

Lehrgang: Diplomarbeit Technik und
Wirtschaftsinformatik 2023-2024

**Titel: PostgreSQL HA Cluster - Konzeption
und Implementation**

Fachexperte: **Norman Süssstrunk**

Student: **Michael Graber**

24.05.2024

Management Summary

Das KSGR (Kantonsspital Graubünden) betreibt über zwanzig PostgreSQL Datenbanken. Diese Datenbanken werden nicht auf einem zentralen Cluster betrieben, sondern oft als single Instanz auf diversen Servern.

In Zukunft wird das KSGR vermehrt auf Kubernetes setzen, was eine noch unbekannte Anzahl weiterer PostgreSQL Datenbanken mit sich bringen wird.

Auch sollen eine ganze Reihe MariaDB / MySQL und Oracle Datenbanken auf PostgreSQL migriert werden.

Um den administrativen Aufwand zu verringern, die Anzahl der Server zu konsolidieren und um die Kontrolle zu erhöhen,

soll nun eine hochverfügbare Clusterlösung für die PostgreSQL Datenbanken evaluiert werden. Ziel dieser Arbeit ist demnach, die Evaluation einer Clusterlösung aus drei Evaluationssystemen zu ermitteln und das geeignetste als Testsystem zu implementieren.

Da, dass KSGR in Zukunft stark auf die Cloud setzen wird, muss ein monolithisches und mindestens ein verteiltes System genauer untersucht werden.

Auf dem Markt lassen sich einige Open-Source, Closed-Source- und proprietäre Lösungen finden.

Der Scope der Arbeit lag dabei nur auf den Open-Source-Lösungen.

In der Evaluation wurden die drei Systeme Patroni, StackGres und YugabyteDB genauer untersucht und jeweils als Evaluationssystem installiert und getestet.

Nach dem Testing und den Benchmark-Tests setzte sich Patroni gegen die beiden anderen Varianten durch.

YugabyteDB hat sich indes für als PostgreSQL Clustersystem für Applikationen, die in Zukunft auf Kubernetes betrieben werden sollen.

Im Anschluss wurde ein Patroni-Testsystem mithilfe des Ansible-Repositories vitabacks / postgresql_cluster installiert.

Das Ansible-Repository bietet eine Reihe von Playbooks, von welchen das Deployment-Playbook, das Maintenance-Playbook sowie ein Playbook zur Erweiterung von Patroni- und Proxy Nodes getestet wurden.

Die Umsetzung mit vitabacks / postgresql_cluster bestand die Tests.

Bis zur Produktivsetzung müssen im Nachgang der Diplomarbeit noch Arbeitspakete umgesetzt werden,

welche während der Diplomarbeit nicht umgesetzt werden konnten, da die Voraussetzungen nicht gegeben waren.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Ausgangslage und Problemstellung	1
1.1.1 Das Kantonsspital Graubünden	1
1.1.2 Die ICT des Kantonsspital Graubünden	3
1.1.3 Rolle in der ICT vom Kantonsspital Graubünden	5
1.1.4 Ausgangslage	6
1.1.5 Problemstellung	11
1.2 Zieldefinition	17
1.3 Abhängigkeiten	23
1.4 Risikomanagement	25
1.4.1 Riskcontrolling	27
1.5 Vorgehensweise und Methoden	32
2 Projektmanagement	33
2.1 Projektplanung	33
2.1.1 Projektcontrolling	33
2.1.2 GANTT-Diagramm	34
2.2 Expertengespräche	38
3 Umsetzung	39
3.1 Evaluation	39
3.1.1 Exkurs Architektur	39
3.1.2 Erheben und Gewichten der Anforderungen	45
3.1.3 Testziele erarbeiten	49
3.1.4 PostgreSQL Benchmarking	52
3.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen	57
3.1.6 Vorauswahl	80
3.1.7 Installation verschiedener Lösungen	80
3.1.8 Testing Evaluationssysteme	98
3.1.9 Gegenüberstellung der Lösungen	101
3.1.10 Entscheid	118
3.2 Aufbau und Implementation Testsystem	119
3.2.1 Architektur	119
3.2.2 Bereitstellen der Grundinfrastruktur	121
3.2.3 Installation und Konfiguration PostgreSQL HA Cluster	121

3.2.4	Technical Review der Umgebung	126
3.3	Testing	130
3.3.1	Protokollierung	133
3.3.2	Review und Auswertung	134
3.4	Maintenance-Tool	134
3.5	Monitoring	138
3.6	Troubleshooting und Lösungsfindung	140
4	Resultate	143
4.1	Zielüberprüfung	143
4.1.1	SQL Optimierungen - Indizes tracken und verwalten	144
4.1.2	Testziele	144
4.2	Schlussfolgerung	144
4.2.1	Kubernetes / rke2	144
4.2.2	YugabyteDB	145
4.2.3	StackGres - Citus	145
4.2.4	Patroni	145
4.2.5	vitabacks / postgresql_cluster	146
4.3	Weiteres Vorgehen / offene Arbeiten	146
4.4	Persönliches Fazit	147
Abbildungsverzeichnis		149
Tabellenverzeichnis		153
Listings		155
Literatur		160
Abkürzungen		166
Glossar		168
Anhang		i
I	Disposition	i
II	Arbeitsrapport	ii
III	Protokoll - Fachgespräche	v
III.I	Fachgespräch 14.02.2024	v
III.II	Fachgespräch 27.03.2024	viii
IV	Statusbericht	xii
IV.I	Status Report 1	xii
IV.II	Status Report 2	xiv

IV.III	Status Report 3	xvii
IV.IV	KSGR Abschlussbericht	xix
V	Kommentare / Anmerkungen	xxiv
VI	Evaluation	xxvii
VI.I	Maintenance - CloudNativePG	xxvii
VI.II	Maintenance - Patroni	xxx
VI.III	Maintenance - StackGres - Citus	xxxiii
VI.IV	Maintenance - YugabyteDB	xxxix
VII	Evaluationssysteme - Installation	xlii
VII.I	rke2	xlii
VII.II	yugabyteDB	xlvi
VII.III	sks9016 - yugabyteDB	lxxiii
VII.IV	Patroni	lxxiv
VII.V	Stackgres mit Citus	xciv
VIII	Evaluationssysteme - Benchmarking	cxv
VIII.I	yugabyteDB	cxv
VIII.II	Patroni	c xvii
VIII.III	StackGres - Citus	cxix
IX	Evaluationssysteme - Testing	cxxii
IX.I	StackGres - Citus	cxxii
IX.II	YugabyteDB	cxxiv
X	Testsystem - Installation	cxxvi
X.I	Prequeries	cxxvii
X.II	Deployment	cxxix
X.III	Maintenance	clxx
X.IV	Prequeries - Erweiterungstests	clxxxiii
X.V	HAproxy erweitern	clxxxiii
X.VI	Patroni Node	clxxxiv
XI	Testsystem - Testing	clxxxv
XII	Maintenance-Tool	clxxxv
XII.I	Maintenance-Tool - Bloated Tables	clxxxv
XII.II	Maintenance-Tool - AUTOVACUUM	cxi
XIII	Monitoring - PRTG	cxcviii
XIV	Exkurs Architekturen - Umsysteme und Prinzipien	cciii
XIV.I	Raft-Consensus	cciii
XIV.II	local-path-provisioner	cciv

1

Einleitung

1.1 Ausgangslage und Problemstellung

1.1.1 Das Kantonsspital Graubünden

Das Kantonsspital Graubünden (KSGR) ist das Zentrumsspital der Südostschweiz, welches Teil der sogenannten Penta Plus Spitäler ist. Die Penta Plus Spitäler sind das Kantonsspital Baden, das Kantonsspital Winterthur, das Spitalzentrum Biel AG, das Kantonsspital Baselland, die Spital STS (Simmental-Thun-Saanenland) AG und eben das Kantonsspital Graubünden.

Das KSGR deckt dabei die Spitalregion Churer Rheintal ab.

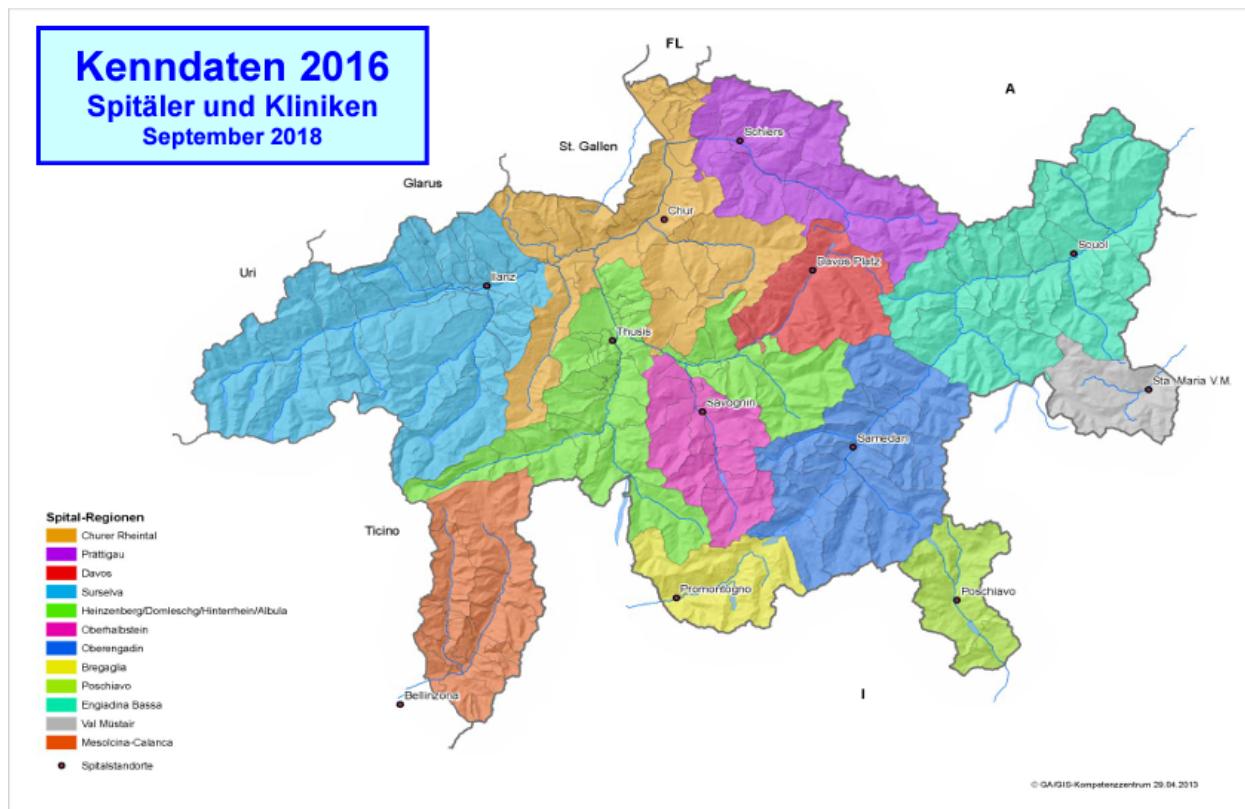


Abbildung 1.1: Spitalregionen Kanton Graubünden[66]

Zudem betreibt das KSGR seit dem 1. Januar 2023 den Standort Walenstadt im Kanton St. Gallen und deckt primär den Wahlkreis Sarganserland ab.



Abbildung 1.2: Wahlkreise Kanton St. Gallen[93]

Da dieser Wahlkreis der Spitalregion Rheintal Werdenberg Sarganserland zugeordnet ist, wird das KSGR auch im restlichen südlichen Teil der Spitalregion aktiv sein.

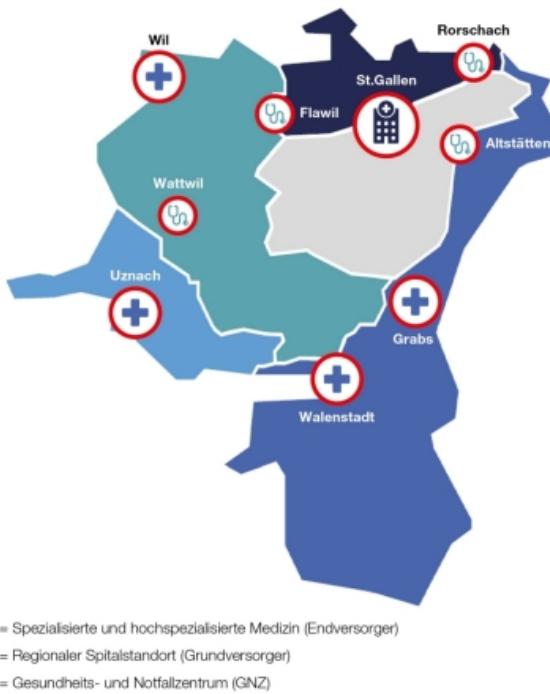


Abbildung 1.3: Spitalregionen / Spitalstrategie Kanton St. Gallen[60]

1.1.2 Die ICT des Kantonsspital Graubünden

Das Kantonsspital Graubünden hat eine Matrixorganisation. Die ICT ist ein eigenständiges Departement und gilt als sogenanntes Querschnittsdepartement, dh. die ICT bedient alle anderen Departemente.

Organigramm des Kantonsspitals Graubünden

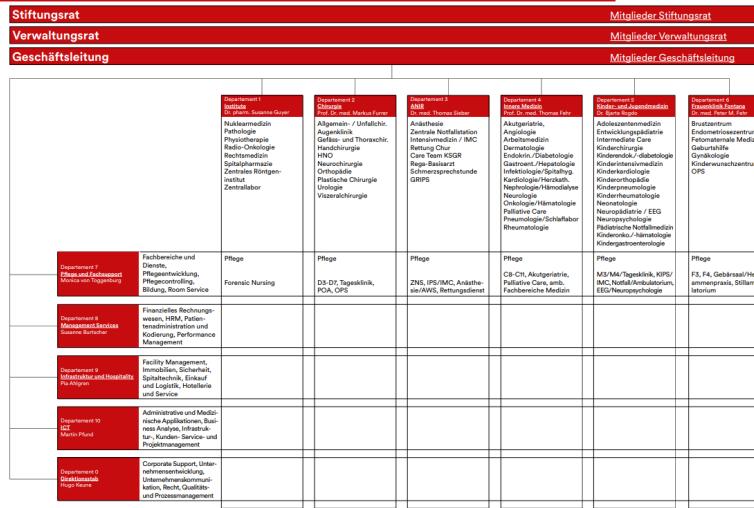


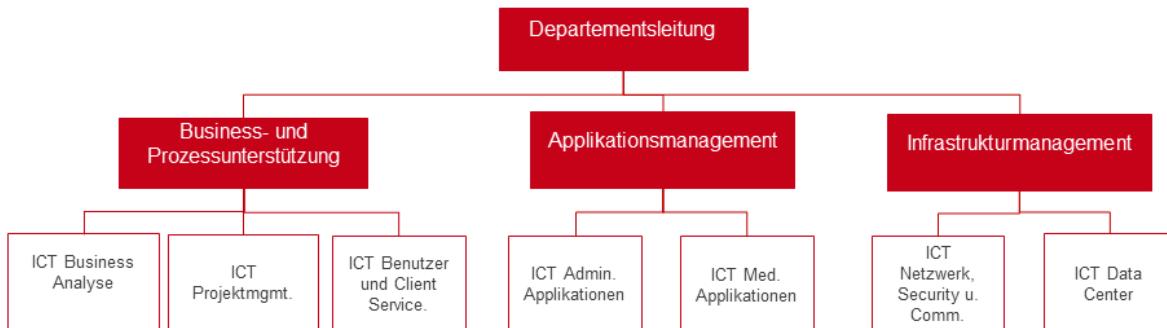
Abbildung 1.4: Organigramm Kantonsspital Graubünden

Die ICT betreibt über 400 Applikationen die auf mehr als 1055 physische und virtuelle Server und Appliances. Das Rückgrat der Infrastruktur ist dabei die Virtualisierungsplattformen VMware ESXi für Server und Citrix für die Thinclients der Enduser. Es werden aber auch Dienstleistungen für andere Spitäler und Kliniken oder andere Einrichtungen des Gesundheitswesens erbracht.

Entsprechend wurde die ICT in ein Applikationsmanagement, ein Infrastrukturmanagement sowie einem unterstützenden Bereich aufgegliedert. Das Applikationsmanagement wurde in je einen Bereich für die Administrativen und Medizinischen Applikationen aufgeteilt. Das Infrastrukturmanagement wiederum wurde in den Bereich Netzwerk und Data Center, welcher für Server zuständig ist, aufgeteilt. Der Bereich Business- und Prozessunterstützung beinhaltet je eine Abteilung für die Businessanalyse, das Projektmanagement und Benutzer- und Clientservices in der auch der Service-Desk untergebracht ist.

(Führungs-)Organisation Departement 10 ab 2023

Kantonsspital
Graubünden



29.09.2023

3

Abbildung 1.5: Organigramm Departement 10 - ICT

Die Organisation der ICT wird sich aber bis spätestens zum Abschluss der Diplomarbeit noch verändern.

1.1.3 Rolle in der ICT vom Kantonsspital Graubünden

Meine Rolle im Kantonsspital Graubünden resp. in der ICT ist die eines DBA. Diese Rolle ist in der Abteilung ICT Data Center.

Da die Kernsysteme auf Oracle Datenbanken und HP-UX laufen, bin ich primär Oracle Database DBA und manage das HP-UX in Zusammenarbeit mit HPE. Die administrative Tätigkeit bei HP-UX besteht primär im Betrieb der HP-UX Cluster Packages (einer sehr rudimentären Art von Containern), überwachen und erweitern des Filesystems, erweitern von SAN Storage Lunes für die Filesystem Erweiterung, Erstellen von PRTG-Sensoren für das Monitoring, SAP Printerqueue Management und andere Tasks die es noch auszuführen gibt. Daneben bin ich auch für andere Datenbanken, teilweise aber nur begrenzt Microsoft SQL Server, MySQL / MariaDB und vermehrt PostgreSQL zuständig. Darüber hinaus bin ich Teilweise in die Linux-Administration involviert und betreue auch noch einige Windows Server für das Zentrale klinische Informationssystem.

1.1.4 Ausgangslage

Die meisten der über 400 Applikationen, die das KSGR betreibt, speichern in den allermeisten Fällen ihre Daten in Datenbanksystemen. Entsprechend der Vielfalt der Applikationen existieren auch eine Vielzahl an Datenbanksystemen und Versionen.

Basierend auf der Liste *DB-Engines Ranking*[57] der Top-Datenbanksysteme. Allerdings werden nicht alle Datenbanksysteme berücksichtigt, entweder weil das Datenbanksystem keine Client/Server Architektur hat oder nicht im Scope der IT oder des Projekts ist.

Folgende Datenbanken sind inventarisiert:

DBMS	Datenbankmodell	Inventarisiert	Kommentar
Oracle Database	Relational, NoSQL, OLAP	Ja	
MySQL	Relational	Ja	
Microsoft SQL Server	Relational, NoSQL, OLAP	Nein	Werden separat administriert und sind daher nicht in diesem Inventar gelistet
PostgreSQL	Relational, NoSQL	Ja	
MongoDB	NoSQL	Ja	
Redis	Key-value	Ja	
Elasticsearch	Search engine	Ja	
IBM DB2	Relational	Ja	
SQLite	Relational	Nein	Lokale Datenbank. Zudem wird die DB nicht via Netzwerk angesprochen
Microsoft Access	Relational	Nein	Nicht im Scope der ICT
Snowflake	Relational	Ja	
Cassandra	Relational	Ja	
MariaDB	Relational	Ja	
Splunk	Search engine	Ja	
Microsoft Azure SQL Database	Relational, NoSQL, OLAP	Nein	Datenbanken sind nicht On-Premise und somit nicht im Scope

Tabelle 1.1: Inventarisierte Datenbanksysteme

Folgende Datenbanksysteme sind demnach im KSGR im Einsatz:

	RDBMS	Instanz	Datenbanken	Appliance
0	MariaDB	2	2	0
1	MongoDB	2	2	0
2	MySQL	28	50	3
3	Oracle Database	27	30	0
4	PostgreSQL	20	20	4
5	Redis	1	1	0
Gesamtergebnis		80	105	7

Tabelle 1.2: Datenbankinventar

Datenbankinventor - Pro RDBMS

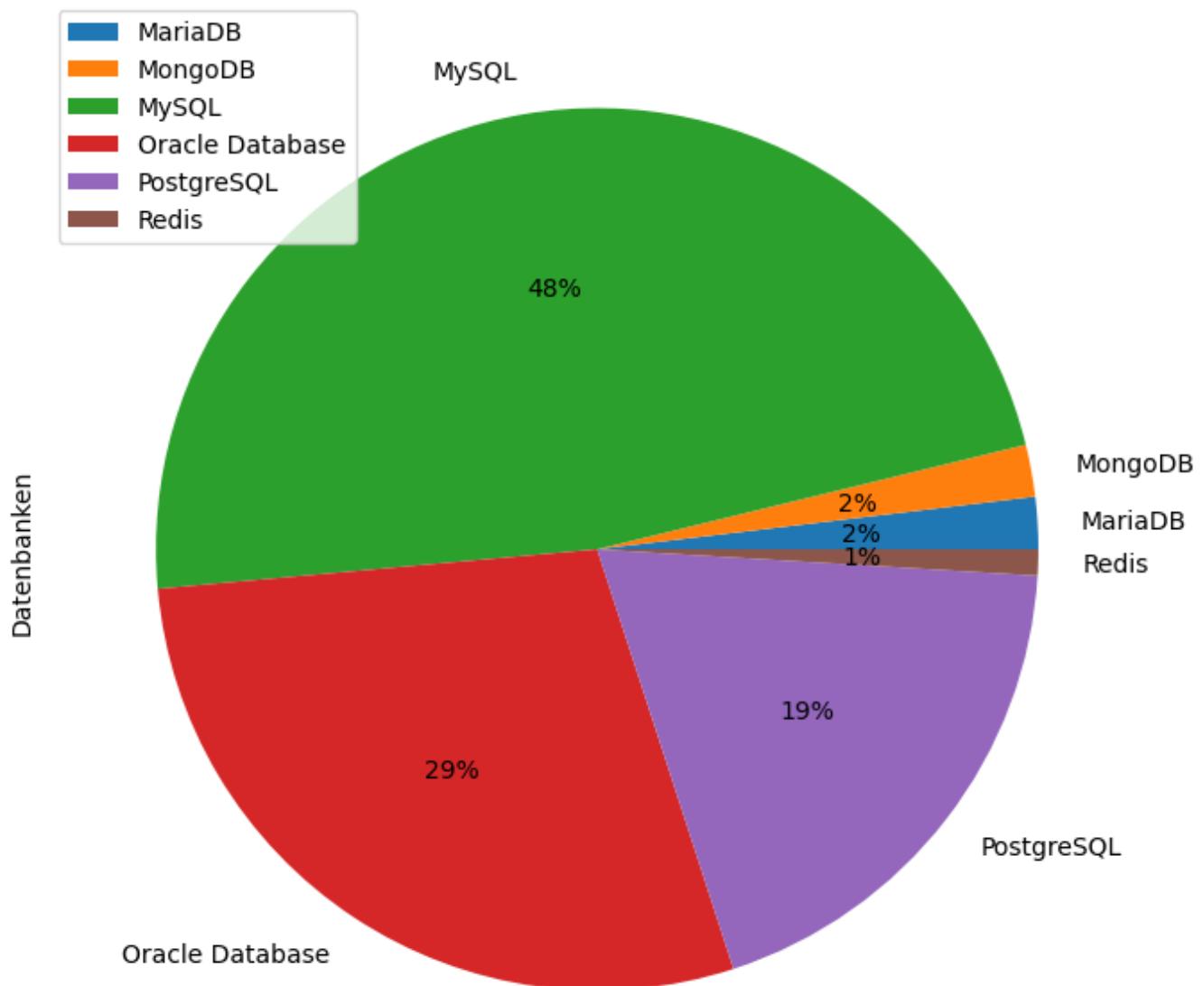


Abbildung 1.6: Datenbanken - Aufgeschlüsselt nach RDBMS - Datenbanken

Datenbankinventor - Pro RDBMS

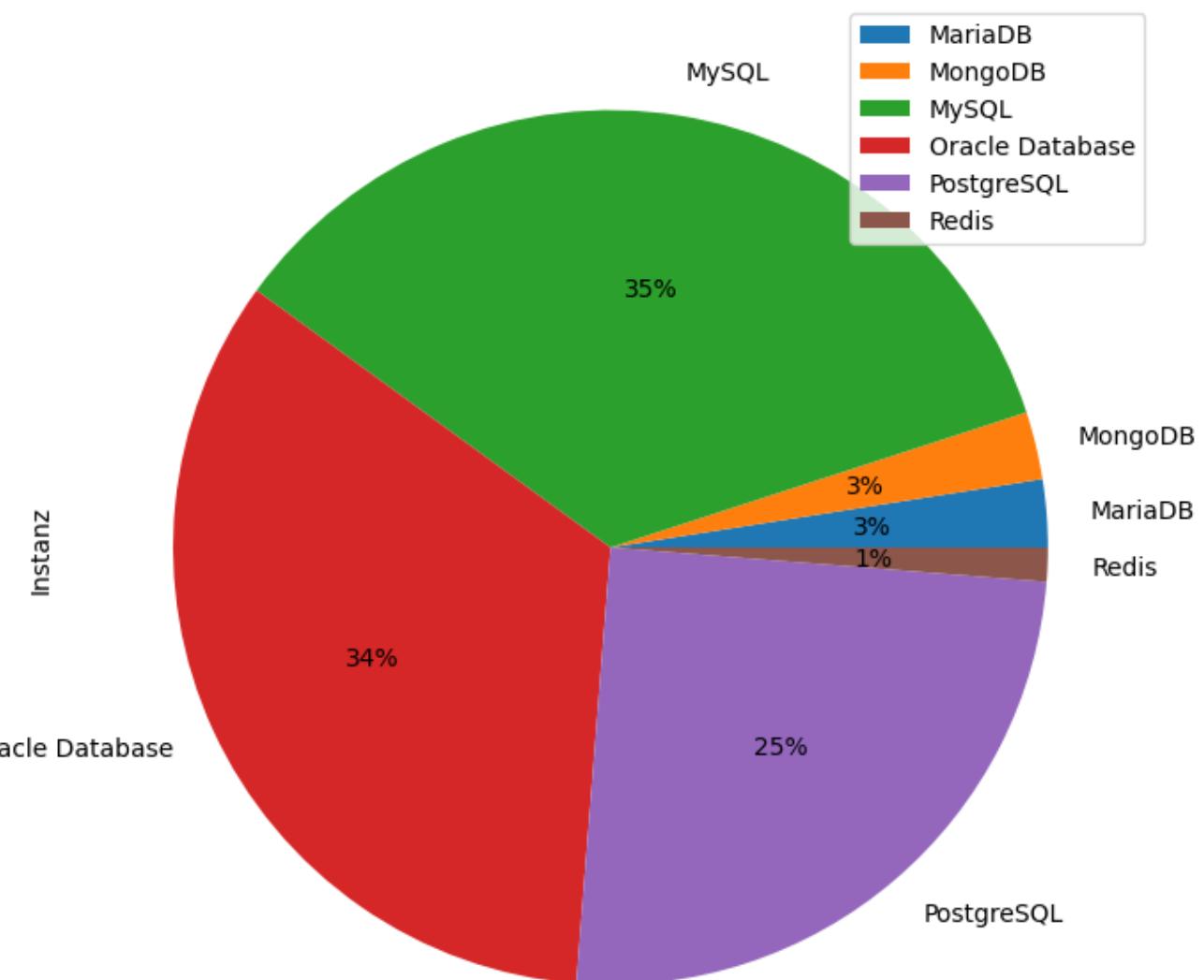


Abbildung 1.7: Datenbanken - Aufgeschlüsselt nach RDBMS - Instanzen

Aufgeschlüsselt auf die Betriebssysteme auf denen die Datenbanken laufen, ergibt sich folgendes Bild:

OS	RDBMS	Appliance	Datenbanken	Instanz
HP-UX	Oracle Database	0	24	21
Linux	MariaDB	0	2	2
	MySQL	3	36	14

Continued on next page

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

OS	RDBMS	Appliance	Datenbanken	Instanz
		0	1	1
Windows Server	Oracle Database	0	1	1
	PostgreSQL	4	8	8
	Redis	0	1	1
	MongoDB	0	2	2
	MySQL	0	14	14
	Oracle Database	0	5	5
	PostgreSQL	0	12	12
	Gesamtergebnis	7	105	80

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

Die Kernsysteme des Spitals werden auf Oracle Datenbanken (Oracle Database) betrieben, die aktuell auf einer HP-UX betrieben werden. Stand heute gibt es kein Clustersystem für die Open-Source Datenbanken wie MariaDB/MySQL oder PostgreSQL.

Durch die Einführung von Kubernetes als Containerplattform wird der Bedarf an PostgreSQL Datenbanken immer grösser. Es werden in naher Zukunft auch verschiedene Oracle Datenbanken sowie MySQL Datenbanken auf PostgreSQL migriert werden.

Aktuell werden die Daten des Zabbix der Netzwerktechniker auf eine MariaDB Datenbank gespeichert, dies soll sich aber ändern. Da das Zabbix alle Netzwerkgeräte Überwacht, pro Sekunde werden im Moment ca. 950 Datenpunkte abgefragt,

pro Sekunde werden 23 Connections geöffnet resp. offen gehalten. Die 23 Connections setzen pro Sekunde über 7'000 Transaktionen ab, wobei über 4'000 direkt von den Clients kommen.

Mittlerweile ist die Datenbank auf über 230GiB angewachsen.

Genauere Informationen sind im Kapitel **Benchmark Settings** zu finden.

1.1.5 Problemstellung

Zusammen mit den bestehenden PostgreSQL Datenbankinstanzen werden die PostgreSQL Datenbanken in der Art, wie sie bisher betrieben werden, nicht mehr betreibbar sein. Die bisherige Strategie erzeugt sehr viel Aufwand und provoziert Risiken, namentlich:

- dezentrale Backups und fragmentierte Backup-Strategien
 - Fehlende Kontrolle
 - Wiederherstellbarkeit nicht garantiert
- Verschiedene Betriebssysteme mit verschiedenen Versionen
 - Fehlernder Überblick
 - Veraltete Betriebssystem- und Datenbankversionen

- Grosser Administrationsaufwand
- Uneinheitliche Absicherung und Härtung
 - Hohe Angreifbarkeit
 - Veraltete Betriebssystem- und Datenbankversionen
 - Grosser Administrationsaufwand
- Uneinheitliche HA-Fähigkeit
 - Hohe Angreifbarkeit
 - Veraltete Betriebssystem- und Datenbankversionen
 - Grosser Administrationsaufwand

Dadurch ergeben sich nach BSI folgende Risiken:

Identifikation						Abschätzung		Behandlung				
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen ergreifen?	WS	SM	Zielwert	Massnahme
1 I		G0.22	Manipulation von Informationen	Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme ausfallen	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja	1	2	Best-Practice bei Härtung der Systeme. Redundanzen einführen	
2 A		G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Fehlfunktionen erleiden.	Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung.	4	4	Ja	2	2	Redundanzen einführen	
3 C, I, A		G0.26	Fehlfunktion von Geräten oder Systemen	Allerdings versuchen Datenbanksysteme, die Auswirkungen so gering wie möglich zu halten. Aufgrund der sehr heterogenen Landschaft ist der Administrationsaufwand für die jetzigen Systeme sehr gross. Zu gross, als das für jede Datenbank und deren Betriebssystem die notwendige Zeit für eine bedarfsgerechte Administration erbracht werden kann.	Fehlfunktionen können innerhalb von Datenbanksystemen die Datenkonsistenz verletzen, Daten können verloren gehen oder ungewollt von Dritten und unberechtigten Personen eingesehen werden. Systeme könnten nicht mehr oder nur noch eingeschränkt verfügbar werden.	2	4	Ja	2	2	Systeme zentralisieren Lifecycle etablieren	
4 C, I, A		G0.27-1	Ressourcenmangel (personelle Ressourcen)	Dadurch bleiben Fehler länger unentdeckt, Hotfixes, Patches, Updates und Upgrades können nicht oder nicht zur richtigen Zeit eingespielt werden. Bei einem akuten Problemfall ist nicht garantiert, dass die Leute erreichbar sind, die notwendig sind	Die Auswirkungen können vielfältig sein, abhängig davon welcher Aspekt unter dem Ressourcenmangel leidet. Grundsätzlich wird aber sowohl die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	3	3	Ja	2	3	Systeme zentralisieren	
5 A		G0.27-2	Ressourcenmangel (technische Ressourcen)	Kann auftreten wenn Ressourcenwachstum zu spät bemerkt wird. So kann die CPU Usage oder das Memory Usage schnell anwachsen. Auch der Storage eines Betriebssystems kann nicht mehr ausreichend für ein System werden.	Wenn die CPU- und Memory-Usage über einen gewissen Schwellwert geht, fängt das Betriebssystem an zu Priorisieren. Dies wird primär der Endanwender in form von Performance Einbussen bemerken. Im schlimmsten Fall steht eine Anwendung nicht mehr zur Verfügung.	2	2	Ja	1	2	Monitoring verschärfen	
6 C, I, A		G0.31	Fehlerhafte Nutzung oder Administration von Geräten und Systemen	Durch die Vielfalt an Datenbankversionen und Betriebssystemen und Plattformen worauf diese betrieben werden, besteht allen voran das Risiko einer fehlerhaften Administration und Konfiguration.	Gefährlicher sind Storage Overflows, besonders wenn die Datenbank nicht mehr alle Informationen schreiben konnte, die sie für einen korrekten Neustart benötigte.	4	3	Ja	2	3	Systeme zentralisieren	
7 C, I, A		G0.32	Missbrauch von Berechtigungen	Obwohl das Microsoft Active Directory die zentrale Benutzerverwaltung ist, sind die wenigsten Datenbanken an dieses angeschlossen. Hinzu kommt der Umstand, dass in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen.	Doch die folgen bleiben nichtsdestotrotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen.	2	4	Ja	2	2	Systeme zentralisieren Übergreifendes Berechtigungskonzept einführen Monitoring der Zugriffe	
8 A, I		G0.45	Datenverlust	Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen wissentlich oder unwissentlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen. Zudem wurden die meisten Datenbanken nur mittels Snapshots oder einem Filesystem Backup gesichert, nicht über eine eigentliche Sicherung mittels WAL. Gerade die fehlende WAL-Archivierung führt im Backupfall dazu, dass alle Transaktionen die zwischen dem letzten Backup nicht mehr vorhanden sind. Hinzu kommt, das für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde. Ein weiterer Aspekt des Risikos besteht in der Tatsache, das aufgrund der grossen Anzahl Datenbanken und deren Heterogenität nur wenige Backups auch wirklich regelmässig geprüft werden.	Aus dem Risiko ergeben sich zwei Auswirkungen, die aber beide ein hohes Mass an Schaden verursachen können. Erstens könnten Backups gar nicht mehr wiederhergestellt werden, dies hätte dann einen totalen Datenverlust zur Folge. Die zweite Ursache erwächst auf der fehlenden WAL-Archivierung, dadurch können zwar die Daten bis zu einem Zeitpunkt X wiederhergestellt werden allerdings sind diese dann nicht zwingend konsistent.	4	5	Ja	1	3	Systeme zentralisieren Einheitliches Backupkonzept Regelmässige Restore-Tests	

Tabelle 1.4: Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken

Daraus ergeben sich folgende Strategien und Handlungsfelder um die Massnahmen zur Risikominimierung umzusetzen:

- Systemabsicherung erarbeiten und einsetzen
- HA-Clustering einführen um die Redundanz zu gewährleisten und Systeme zentral verwalten und betreiben zu können
- Lifecycle-Management für Datenbanken und Betriebssysteme erarbeiten und einsetzen
- Backup-Konzept erarbeiten
- Berechtigungskonzept erarbeiten und einführen

Mit diesen Massnahmen lassen sich die Risiken senken.
Nachfolgend der Vergleich der Ursprünglichen Risiken und den Risiken,
wenn Massnahmen getroffen wurden:

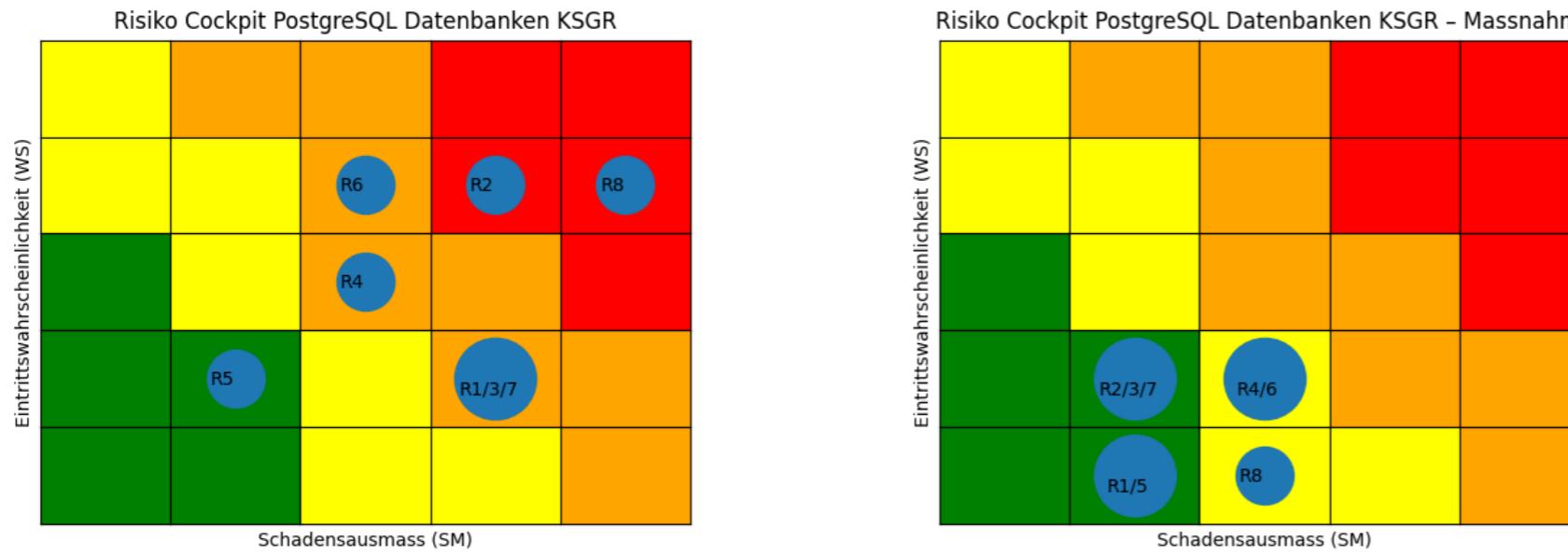


Abbildung 1.8: Risikomanagement PostgreSQL

1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [76, 81] folgende zehn Tasks die zum täglichen Alltag gehören:

Nr.	Aufgabe	Beschreibung	Wichtigkeit
1	Failover	In einem Fehlerfall soll die DB-Node auf einen Standby-Node übergeben werden. Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den primären Node zurückgesetzt werden.	Hoch
2	Failover Restore	Dabei darf es zu keinem Datenverlust kommen, also alle Daten die auf dem Standby-Node erfasst wurden, müssen beim Failover Restore auf den primären DB-Node zurückgeschrieben werden. Die Datenmenge in Datenbanken wachsen in der Regel beständig.	Hoch
3	Filesystem Management	Die Belegung von Tablespace und Filesystem muss deshalb überwacht und ggf. erweitert werden. Läuft eine Disk voll, kommt es im besten Fall zu einem Stillstand der DB, im schlimmsten Fall zu Inkonsistenzen und Datenverlust Nebst den allgemeinen Metriken wie CPU / Memory Usage und der Port-Verfügbarkeit gibt es noch eine Reihe weiterer Aspekte die observiert werden müssen.	Hoch
4	Monitoring	Zum Beispiel, ob es zu verzögerungen bei der Replikation kommt oder die Tablespace genügend Platz haben. Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall. PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. Zudem wird im Rahmen des gleichen Scheduled Tasks ein Cleanup vorgenommen,	Mittel
5	Statistiken / Cleanup Jobs justieren	so dass z.B. gelöschte Datensätze den Disk Space nicht sinnlos belegen. Die Konfiguration dieser Jobs muss an der Metrik der Datenbank angepasst werden, weil gewisse Tasks dann entweder viel zu oft oder viel zu wenig bis gar nicht mehr ausgeführt werden.	Mittel
6	SQL optimierungen	In PostgreSQL können unperfomante SQL Statements ausgelesen werden und zum Teil werden auch Informationen zum Tuning geliefert[55]. Diese müssen regelmäßig ausgelesen werden	Tief
7	Health Checks und Aktionen (Maintenance)	Regelmässig muss die Gesundheit der DBs überprüft werden, etwa ob Tabellen und/oder Indizes sich aufgeblättert haben oder ob Locks vorhanden sind[3]. Während der Hauptarbeitszeit muss dies mindestens alle 90 Minuten geprüft und ggf. reagiert werden.	Hoch
8	Housekeeping	Mit Housekeeping Jobs werden regelmäßig Trace- und Alertlogfiles aufgeräumt, um Platz auf den Disken zu sparen, aber auch, um die Übersichtlichkeit zu wahren.	Mittel
9	Verwalten von DB Objekten	Regelmässig müssen DB Objekte wie Datenbanken, Tabellen, Trigger, Views etc. angepasst oder erstellt werden. Dies richtet sich nach den Bedürfnissen der Kunden resp. deren Applikationen.	Tief
10	User Management	Die Zugriffe der User müssen überwacht, angepasst, erfasst oder gesperrt werden. Auch diese Aufgabe richtet sich nach den Bedürfnissen der Kunden.	Tief

Tabelle 1.5: Administrative Aufgaben

Von diesen Tasks müssen Teile davon zu 50% automatisiert werden wobei alle Muss-Aufgaben automatisiert werden müssen. Diese wären nachfolgende Tasks die automatisiert werden können.

Nr.	Aufgabe	Wichtigkeit	Zu automatisierender Task	Priorität	Muss / Kann	Spätester Termin
1	Failover	Hoch	Automatisierter Failover auf mindestens einen sekundären DB-Node	1	Muss	Abgabe
2	Failover Restore	Hoch	Sobald der Primäre DB-Node wieder vorhanden ist, muss automatisch auf den primären DB-Node zurückgesetzt werden. Das Filesystem muss beim erreichen von 95% Usage automatisiert vergrössert werden.	1	Muss	
3	Filesystem Management	Hoch	Die Vergrösserung muss anhand der Wachstumsrate (die mittels Linux Commands zu ermitteln ist), vergrössert werden	4	Kann	
4	Monitoring	Mittel	Der Status der Clusterumgebung und der Replikation muss im PRTG überwacht werden	2	Muss	
5	Statistiken / Cleanup Jobs justieren	Mittel	Regelmässig müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden Es gibt SQL Abfragen, mit dem fehlende Indizes ermittelt werden können. Diese Indizes sollen automatisiert erstellt werden.	2	Muss	
6	SQL-Optimierungen	Tief	Im gleichen Zug sollen aber auch Indizes, welche nicht verwendet werden, entfernt werden. Sie tragen nicht nur nichts zu performanteren Abfragen bei sondern beziehen unnötige Ressourcen bei Datenmanipulationen[55]. Tabellen und Indizes können sich aufblähen (bloated table / bloated index)	2	Kann	
7	Health Checks und Aktionen (Maintenance)	Hoch	Ist ein Index aufgebläht, kann dies mittels eines REINDEX mit geringem Impact auf die Datenbank gelöst werden[3].	2	Muss	
8	Housekeeping	Mittel	Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden.	3	Kann	
9	Verwalten von DB Objekten	Tief	Keine Automatisierung möglich	5		
10	User Management	Tief	Regelmässige Reports sollen User anzeigen, die seit mehr als einer Woche nicht mehr aktiv waren.	4	Kann	

Tabelle 1.6: Automatisierung Administrativer Aufgaben

Mit dieser Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

- **Ergebnisse**
Mindestens drei Methoden einen PostgreSQL Cluster aufzubauen müssen analysiert und evaluiert werden
- **Resultate**
Aus den mindestens drei Methoden muss die optimale Methode ermittelt werden.
Am Ende muss zudem ein funktionierendes Testsystem bestehen.

Daraus ergeben sich folgende Ziele:

Nr.	Ziel	Beschreibung	Priorität
1	Evaluation	Am Ende der Evaluationsphase müssen mindestens drei Methoden für einen PostgreSQL HA Cluster müssen evaluiert werden. Innerhalb der Evaluation muss analysiert werden, welche Methode oder welches Tool sich hierfür eignen würde.	Hoch
2	Testsystem	Am Ende der Diplomarbeit muss ein funktionierendes Testsystem installiert sein.	Hoch
3	Automatisierter Failover	Ein PostgreSQL Cluster muss im Fehlerfall auf mindestens einen Standby-Node umschwenken. Dabei muss das Timeout so niedrig sein, dass Applikationen nicht auf ein Timeout laufen.	Hoch
4	Automatisierter Failover Restore	Nach einem Failover muss es zu einem Fallback oder Failover Restore kommen, sobald der Primary-Node wieder verfügbar ist.	Hoch
5	Monitoring - Cluster Healthcheck	Die wichtigsten Parameter für das Monitoring des PostgreSQL Clusters (isready, Locks, bloated Tables), der Replikation (Replay Lag, Standby alive) und des PostgreSQL HA Clusters müssen überwacht werden.	Mittel
6	AUTOVACUUM - Parameter verwalten	Täglich müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden	Mittel
7	SQL optimierungen - Indizes tracken und verwalten	Täglich fehlende Indizes automatisiert erstellen und nicht mehr verwendete Indizes automatisiert entfernen	Mittel
8	Maintenance - Indizes säubern	Täglich bloated Indices, also aufgeblähte Indizes, automatisiert erkennen und mittels REINDEX bereinigen	Hoch
9	Housekeeping - Log Rotation	Die Log Rotation muss aktiviert werden. Die Logs müssen aber auch in das KSGR-Log Repository geschrieben werden	Hoch
10	User Management - Monitoring	Nicht verwendete User sollen einmal pro Woche automatisiert erkannt und in einem Report gemeldet werden.	Tief
11	Evaluationsziel	Am Ende der Evaluationsphase muss ein Entscheid getroffen worden sein, welche Methode verwendet wird.	Hoch
12	Installationsziel	Die Testinstallation muss lauffähig sein und zudem alle Anforderungen und Ziele (3 und 4) erfüllen Folgende Testziele müssen erreicht werden: 1. Der PostgreSQL Cluster muss immer lauffähig sein solange noch ein Node up ist, unabhängig davon welche Nodes des PostgreSQL HA Clusters down ist 2. Ein Switchover auf alle sekundären Nodes muss möglich sein 3. Der Fallback auf den primären Node muss erfolgreich sein, unabhängig davon, ob ein Failover oder Switchover stattgefunden hat 4. Das Timeout bei einem Failover / Switchover muss unterhalb der Default Timeouts der Applikationen GitLab und Harbor liegen. 5. Das Replay Lag zwischen Primary und Secondary darf beim Initialen Start nicht über eine Minute dauern oder 1KiB nicht überschreiten	Hoch
13	Testziele		

Tabelle 1.7: Ziele

Im Kantonsspital Graubünden sind bereits einige Systeme im Einsatz, die gegeben sind.

	Produkt	Beschreibung
Storage	HPE 3PAR 8450 SAN Storage System	
Virtualisierungsplattform	VMware® vSphere®	
Primäres Backupsystem	VEEAM Backup System	
Provisioning / Lifecycle Management System	Foreman	Ist zurzeit nur für Linux angedacht
Primäre Linux-Distribution	Debian	
Sekundäre Linux-Distributionen	Rocky Linux Oracle Linux RedHat Enterprise Linux (RedHat Enterprise Linux (RHEL))	RedHat Enterprise Linux (RedHat Enterprise Linux (RHEL)), Rocky Linux oder Oracle Linux wird nur eingesetzt, wenn es nicht anders möglich ist
Primäres Monitoring-System	Paessler Router Traffic Grapher (PRTG)	Monitoring System für alle, ausser dem Netzwerkbereich
Sekundäres Monitoring-System	Zabbix	Wird nur vom Netzwerkbereich verwendet
Container-Plattform	Kubernetes	
Infrastructure as code (Iac) System	Ansible und Terraform	Ansible wird von Foreman verwendet, Terraform wird für die Steuerung der Kubernetes-Plattform verwendet
Log-Plattform / SIEM-System		Wird neu ausgeschrieben
Usermanagement	Microsoft Active Directory	Produkt zurzeit nicht definiert

Tabelle 1.8: Gegebene Systeme

Daraus ergeben sich nach nach Züst, Troxler 2002[106] folgende Abgrenzungen:

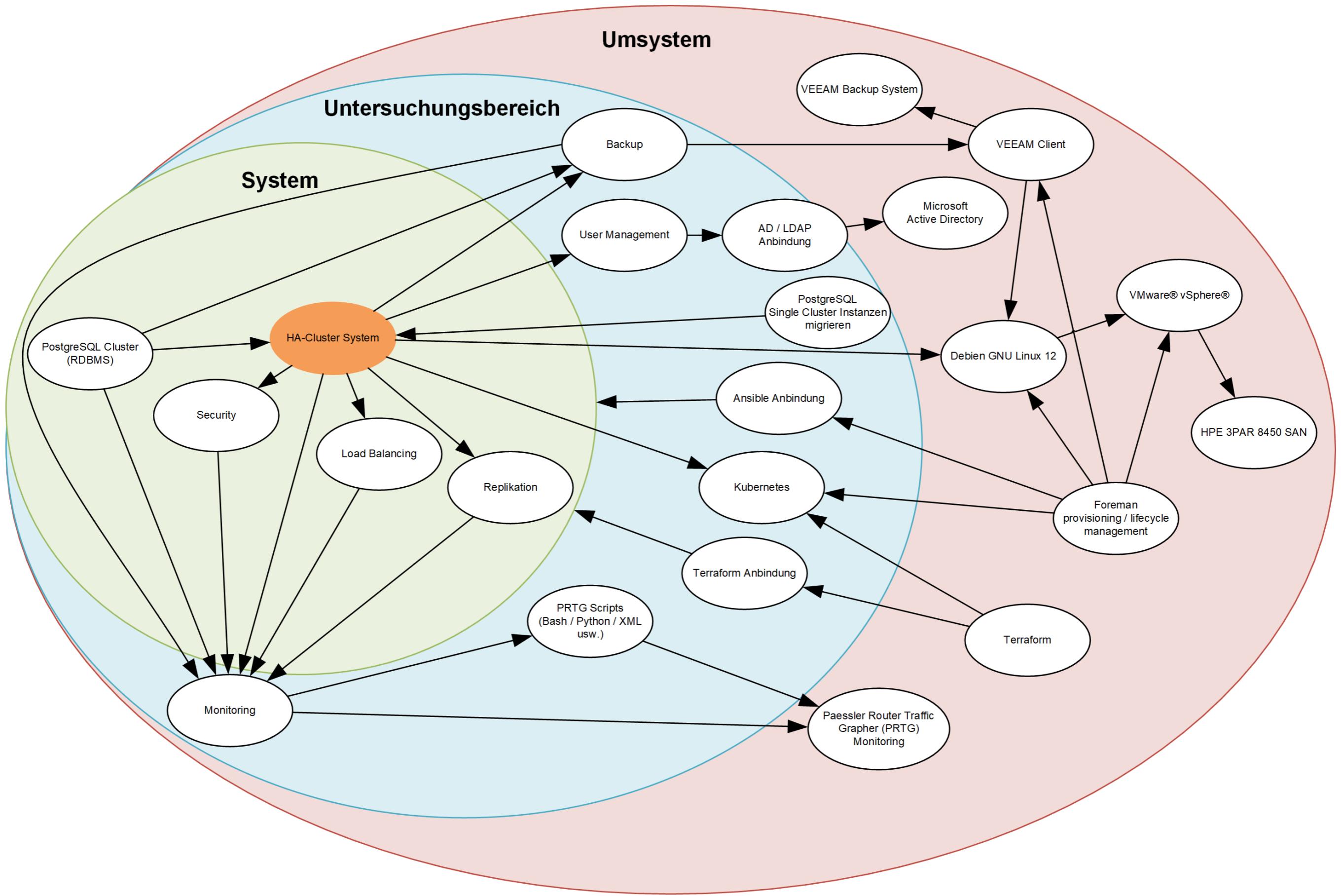


Abbildung 1.9: Systemabgrenzung

1.3 Abhängigkeiten

Es existieren Technische und Organisatorische Abhängigkeiten. Diese haben sowohl ein Risiko als auch einen Impact wenn das Risiko eintrifft. Dies wären folgende:

Nr.	Objekt	Abhängigkeit	Beschreibung	Status	Risiko	Impact
1	Foreman	VMs	Das Lifecycle Management und Provisioning System muss zur Verfügung stehen um in der Evaluationsphase Develop-VMs und in der Installationsphase Test-VMs erstellen zu können.	Im Moment ist Foreman in einer Proof of Concept Phase.	Das Risiko besteht, dass Foreman nicht betriebsbereit ist	VMs müssen von Hand aufgesetzt werden. Entsprechend wird sehr viel mehr Zeit in der Evaluations- und Installationsphase benötigt.
2	Storage	Speicher für VMs / Daten	Es müssen genügend Kapazitäten auf dem Storage vorhanden sein, um die VMs und Datenbanken in Betrieb zu nehmen	Storage wurde bereits erweitert, neue Disks für den SAN Storage wurden bestellt.	Auf dem SAN ist keine Kapazität mehr vorhanden	Es können keine VMs oder Datenbanken erstellt werden
3	Log Management / SIEM System	Sichern der Logfiles für Log Rotation	Sichern der Logfiles für Log Rotation	Ein Log Management System / SIEM muss vorhanden sein, um Logs langfristig sichern zu können.	Die neue Log Management Plattform ist noch nicht betriebsbereit	Log Retention muss stark erhöht werden. Dies wird mehr Storage in Anspruch nehmen.
4	HP-UX Ablöseprojekt	Ressourcen	Das Projekt zur Ablösung der HP-UX Plattform für die Oracle Datenbanken geht in die Konzeptions- und Umsetzungsphase.	Umsetzungsphase.	Als Oracle DBA bin ich stark in das Projekt eingebunden. Es besteht das Risiko eines Ressourcenengpasses	Projekt kann nicht zeitgemäß abgeschlossen werden
5	GitLab	Sicherung	Sicherung von Konfigurationen, Scripts usw.	GitLab ist implementiert und betriebsbereit.	GitLab steht nicht mehr zur Verfügung	Keine Versionierung und Teilsicherungen mehr von Konfigurationsfiles, Scripts usw.
6	PKI	Key Management	Es braucht einen PKI um Keys und Zertifikate handeln zu können	Bestehender PKI wird abgelöst. Ablösungsprojekt in der Initialisierungsphase. Bestehender PKI nicht für Zertifikate im Einsatz	Es steht kein moderner PKI im Einsatz.	Zertifikate können aus Zeitgründen nicht in der Evaluationsphase eingesetzt werden. Für die Testphase müssen Zertifikate manuell ausgestellt werden.
7	Veeam Kasten K10[4]	Backup	Kubernetes Nodes und Pods können nicht mit Klassivem Veeam gesichert werden. Hierfür hat Veeam mit der Version V12 die spezialisierte Veeam Kasten K10 Lösung heraus.	Kann erst beim offiziellen Release von Kubernetes beim KSGR eingeführt werden.	Steht nicht zur Verfügung, bis das Projekt abgeschlossen wird.	Backup muss z.B. einen nfs-Share gesichert werden.

Tabelle 1.9: Abhängigkeiten

1.4 Risikomanagement

Aus den Abhängigkeiten heraus wurden folgende Risiken identifiziert:

Identifikation				Abschätzung		Behandlung			
ID	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen ergreifen?		Zielwert	Massnahme
1	Fehlende Ressourcen	Viele parallele Projekte, Aufträge und der Tagesbetrieb	Ressourcen während der Diplomarbeit sind knapp bemessen	3	4	Ja		2 2	Organisation und Selbstmanagement
2	HP-UX Ablöseprojekt	Das Projekt ist sehr Umfangreich und ist in die Konzeptions- und Umsetzungsphase gestartet	Das Projekt wird parallel zur Diplomarbeit sehr viele Ressourcen und Aufmerksamkeit binden	4	4	Ja		3 3	Ressourcen reservieren
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	HP-UX Plattform, DELL NetWorker / Data Domain Umgebung und HPE 3PAR SAN Storage Umgebung sind über dem Lifecycle und haben in den vergangenen Monaten immer wieder kritische Ausfälle erlebt	Bei einem Event, ausgelöst durch das Alter der HP-UX Plattform, der DELL NetWorker / Data Domain Umgebung oder dem SAN Storage, kann der ganze Betrieb zum erliegen kommen und entsprechend viele Ressourcen aufgrund der Kritikalität binden	4	4	Ja		3 3	Monitoring vorgängig ausbauen und Massnahmen definieren
4	Selbstmanagement und in der Selbstorganisation	Selbstmanagement und Organisation ist nicht meine Stärke	Das Projekt verzettelt sich, Zeit geht verloren. Auch eine Folge könnte der Scope Verlust sein	3	3	Ja		2 2	Werkzeuge im Vorfeld definieren und bereitstellen
5	Scope Verlust während des Projekts	Der Scope kann während des Projekts verloren gehen	Verzettelung und Zeitverlust bis hin zu scheitern	3	4	Ja		2 3	Ziele klar definieren
6	Scope Creep	Der Umfang kann stark steigen wenn Ziele nicht genau genug definiert wurden	Zeitverlust bis hin zu scheitern des Projekts	3	4	Ja		3 3	Ziele SMART definieren
7	SIEM / Log Plattform nicht betriebsbereit	Die öffentliche Ausschreibung für die neue / Log Plattform wurde erst am 23.10.2023 veröffentlicht. Bis zur Implementation kann noch Zeit vergehen.	Logs müssen länger auf dem System selber vorgehalten werden. Zudem müssen ggf. eigene Massnahmen zum Auslesen von Logs getroffen werden	4	1	Nein			
8	Foreman nicht betriebsbereit	Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase. Dadurch besteht das Risiko, dass sie nicht betriebsbereit zum Start der Diplomarbeit ist	Ms müssen von Hand provisioniert werden. Dies bedeutet einen massiven Mehraufwand und verzögert ggf. die Evaluationsphase und mit Sicherheit die Installationsphase	3	5	Ja		3 4	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten.

Tabelle 1.10: Risiko-Matrix der Diplomarbeit

Daraus ergeben sich, mit den Massnahmen, folgende Risikomatrizen:

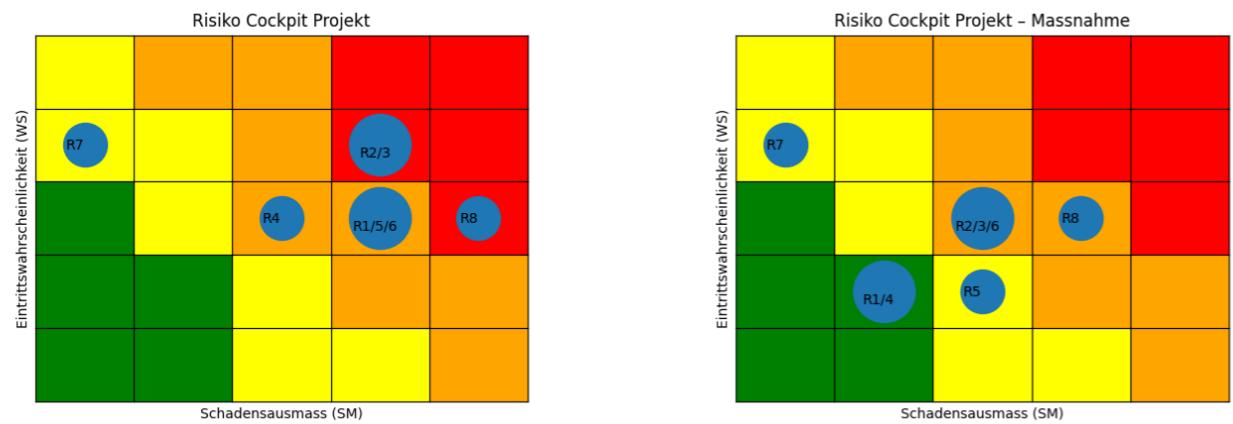


Abbildung 1.10: Risikomanagement Projekt

1.4.1 Riskcontrolling

1.4.1.1 Neu erfasste Risiken

ID	Definiert / Erkannt	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen notwendig	WS.1	SM.1	Massnahmen
9	19.03.2024	Veeam Kasten K10[4] nicht betriebsbereit	Abhängigkeit zum KSGR k8s Projekt. Ohne Kubernetes kein Veeam Kasten.	Keine Kubernetes-Gerechten Sicherungen von Pods usw. Sicherungen inkonsistent o.ä. Ohne Backup kann das Ziel des Projekts nicht erreicht werden	5	5	Ja	5	2	Backup auf einem nfs-Share ablegen. So können Test-DBs wenigstens gesichert werden.
10	29.04.2024	Kubernetes Testumgebung	Abhängigkeit zum KSGR k8s Projekt. Ohne Kubernetes keine YugabyteDB	Es fehlt schlicht ein k8s-Testsystem, wo YugabyteDB installiert werden könnte.	5	1	Nein	5	1	Patroni wird klassisch als Monolith installiert. Für das Projekt besteht zurzeit kein Impact.
11	29.04.2024	Verzug	Evaluationsphase dauerte wesentlich länger als geplant	Rest des Projekts hat einen nicht unerheblichen Verzug	4	5	Ja	4	5	Kaum umsetzbar. Es lässt sich nur schwer etwas streichen.

Tabelle 1.11: Neu Erkannte / Erfasste Risiken

1.4.1.2 Assessment 21.03.2024

ID	Risiko	Assessment-Datum	WS	SM	Status	Ergriffene Massnahmen	Wirksamkeit	Begründung
1	Fehlende Ressourcen	21.03.2024	3	4	hoch	Dokumentation ausserhalb Arbeitszeit	begrenzt	Mentale Ressourcen setzen Limits. ExaCC Server werden mitten während der Diplomarbeit geliefert.
2	HP-UX Ablöseprojekt	21.03.2024	5	4	sehr hoch	Ressourcen reserviert	begrenzt	Von KSGR Seite fehlt eine Stellvertretung. Mithilfe notwendig.
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden Schwächen beim	21.03.2024	4	4	hoch	Externe Partner sensibilisiert	wirksam	Externe Partner können meinen Teil der Aufgaben bei Problemen abfedern. Allerdings nicht vollständig
4	Selbstmanagement und in der Selbstorganisation	21.03.2024	3	3	hoch	- Projektplanung erstellt. - Arbeitspakete geplant	begrenzt	Nicht an alle Tasks gedacht, wie z.B. Risikocontrolling.
5	Scope verlust während des Projekts	21.03.2024	2	2	mittelmässig	Ziele SMART definiert	wirksam	Ziele sind klar definiert. Allerdings gibt es zwangsläufig gewisse Unschärfe.
6	Scope Creep	21.03.2024	3	3	hoch	Ziele SMART definiert	begrenzt	Sehr viele mögliche Lösungen am Markt. SIEM wird nicht rechtzeitig stehen.
7	SIEM / Log Plattform nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Das Schadensmass ist aber zu gering, damit Massnahmen ergriffen werden müssten.
8	Foreman nicht betriebsbereit	21.03.2024	1	1	erledigt	keine		Foreman ist in Betrieb
9	Veeam Kasten K10[4] nicht betriebsbereit	21.03.2024	5	5	sehr hoch	noch keine		

Tabelle 1.12: Risiko-Assessment 21.03.2024

Risiko Cockpit Projekt - Assessment 21.03.2024

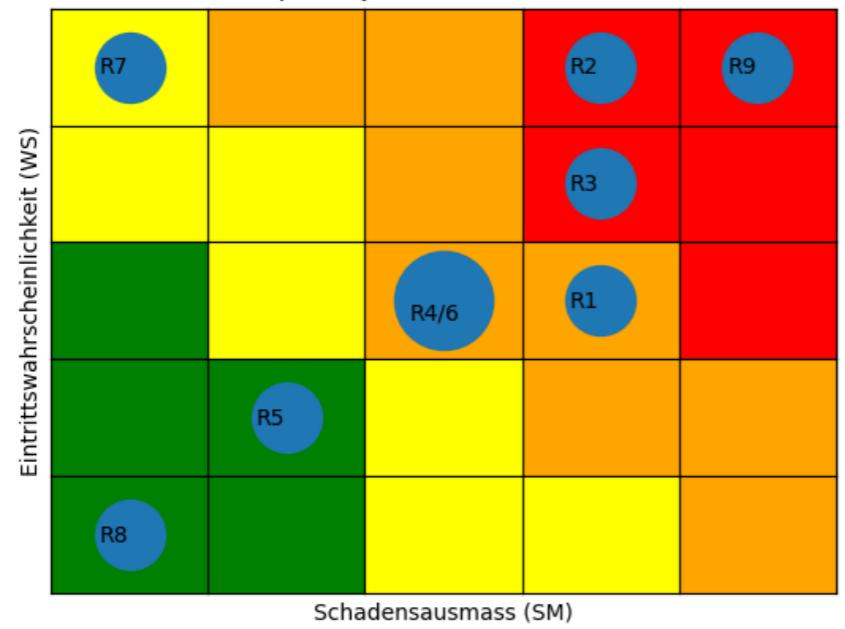


Abbildung 1.11: Riskikomatrix - Assessment 21.03.2024

1.4.1.3 Assessment 29.04.2024

ID	Risiko	Assessment-Datum	WS	SM	Status	Egriffene Massnahmen	Wirksamkeit	Begründung
1	Fehlende Ressourcen	21.03.2024	3	4	hoch	Dokumentation ausserhalb Arbeitszeit	begrenzt	Mentale Ressourcen setzen Limits. ExaCC Server werden mitten während der Diplomarbeit geliefert.
2	HP-UX Ablöseprojekt	21.03.2024	5	4	sehr hoch	Ressourcen reserviert	begrenzt	Von KSGR Seite fehlt eine Stellvertretung. Mithilfe notwendig.
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden Schwächen beim	21.03.2024	4	4	hoch	Externe Partner sensibilisiert	wirksam	Externe Partner können meinen Teil der Aufgaben bei Problemen abfedern. Allerdings nicht vollständig
4	Selbstmanagement und in der Selbstorganisation	21.03.2024	3	3	hoch	- Projektplanung erstellt. - Arbeitspakete geplant	begrenzt	Nicht an alle Tasks gedacht, wie z.B. Risikocontrolling.
5	Scope verlust während des Projekts	21.03.2024	2	2	mittelmässig	Ziele SMART definiert	wirksam	Ziele sind klar definiert. Allerdings gibt es zwangsweise gewisse Unschärfen.
6	Scope Creep	21.03.2024	2	3	hoch	Ziele SMART definiert	wirksam	Vorauswahl getroffen, nur noch drei Evaluierter. SIEM wird nicht rechtzeitig stehen.
7	SIEM / Log Plattform nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Das Schadensmass ist aber zu gering, damit Massnahmen ergriffen werden müssten.
8	Foreman nicht betriebsbereit	21.03.2024	1	1	erledigt	keine		Foreman ist in Betrieb
9	Veeam Kasten K10[4] nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
10	Kubernetes Testumgebung	29.04.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
11	Verzug	29.04.2024	4	5	sehr hoch			

Tabelle 1.13: Risiko-Assessment 29.04.2024

Risiko Cockpit Projekt - Assessment 29.04.2024

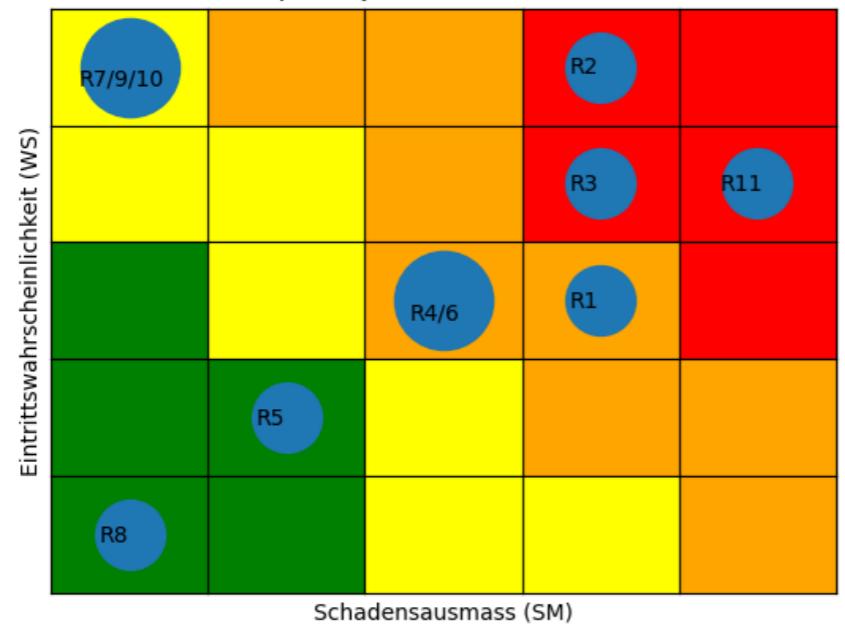


Abbildung 1.12: Riskikomatrix - Assessment 29.04.2024

1.4.1.4 Assessment 13.05.2024 / Abschluss

ID	Risiko	Assessment-Datum	WS	SM	Status	Egriffene Massnahmen	Wirksamkeit	Begründung
1	Fehlende Ressourcen	21.03.2024	3	4	hoch	Dokumentation ausserhalb Arbeitszeit	begrenzt	Mentale Ressourcen setzen Limits. ExaCC Server werden mitten während der Diplomarbeit geliefert.
2	HP-UX Ablöseprojekt	21.03.2024	5	4	sehr hoch	Ressourcen reserviert	begrenzt	Von KSGR Seite fehlt eine Stellvertretung. Mithilfe notwendig.
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden Schwächen beim	21.03.2024	4	4	hoch	Externe Partner sensibilisiert	wirksam	Externe Partner können meinen Teil der Aufgaben bei Problemen abfedern. Allerdings nicht vollständig
4	Selbstmanagement und in der Selbstorganisation	21.03.2024	3	3	hoch	- Projektplanung erstellt. - Arbeitspakete geplant	begrenzt	Nicht an alle Tasks gedacht, wie z.B. Risikocontrolling.
5	Scope verlust während des Projekts	21.03.2024	2	2	mittelmässig	Ziele SMART definiert	wirksam	Ziele sind klar definiert. Allerdings gibt es zwangsweise gewisse Unschärfen.
6	Scope Creep	21.03.2024	2	3	hoch	Ziele SMART definiert	wirksam	Vorauswahl getroffen, nur noch drei Evaluierter. SIEM wird nicht rechtzeitig stehen.
7	SIEM / Log Plattform nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Das Schadensmass ist aber zu gering, damit Massnahmen ergriffen werden müssten.
8	Foreman nicht betriebsbereit	21.03.2024	1	1	erledigt	keine		Foreman ist in Betrieb
9	Veeam Kasten K10[4] nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
10	Kubernetes Testumgebung	29.04.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
11	Verzug	29.04.2024	4	5	sehr hoch			

Tabelle 1.14: Risiko-Assessment 13.05.2024

Risiko Cockpit Projekt - Assessment 13.05.2024

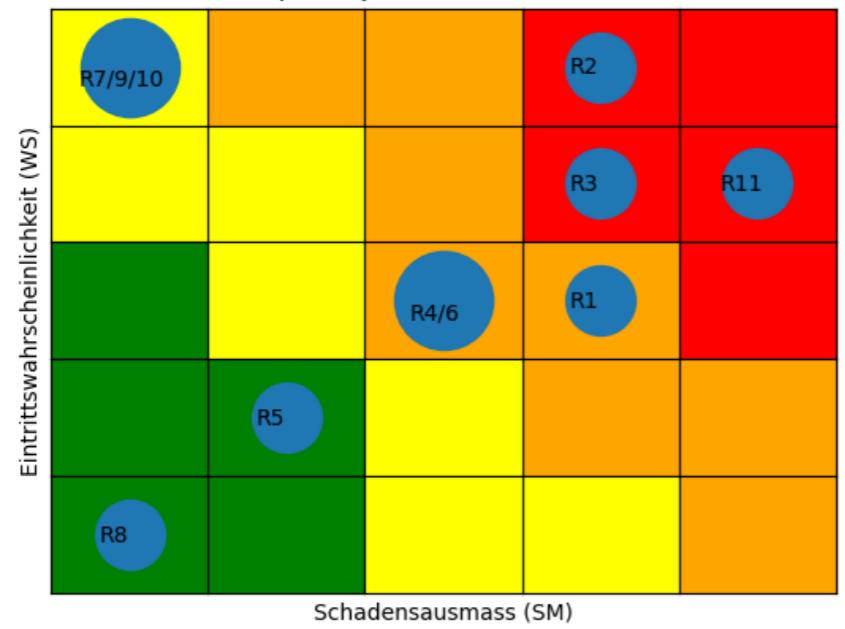


Abbildung 1.13: Riskikomatrix - Assessment 13.05.2024

1.5 Vorgehensweise und Methoden

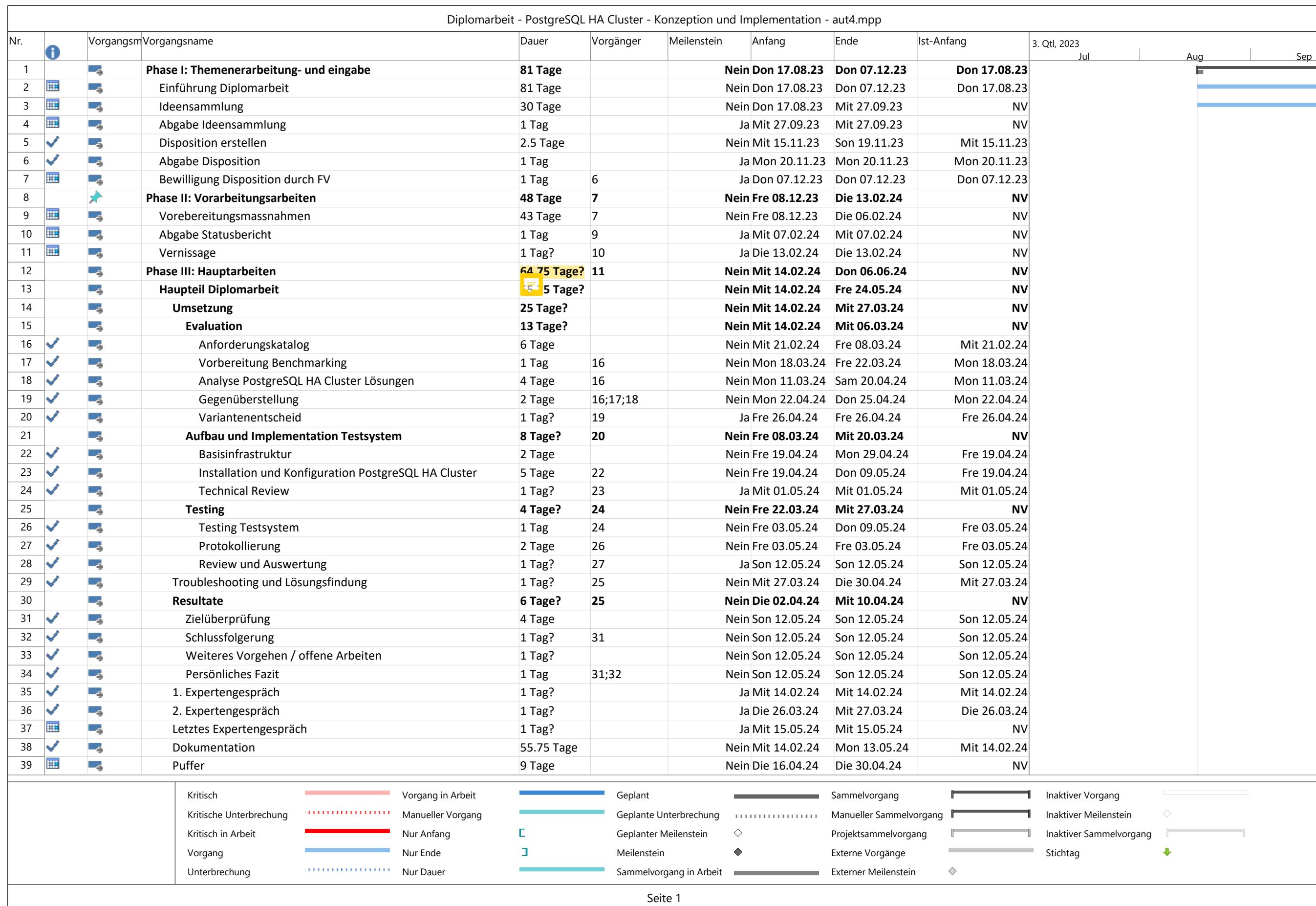
2 Projektmanagement

2.1 Projektplanung

2.1.1 Projektcontrolling

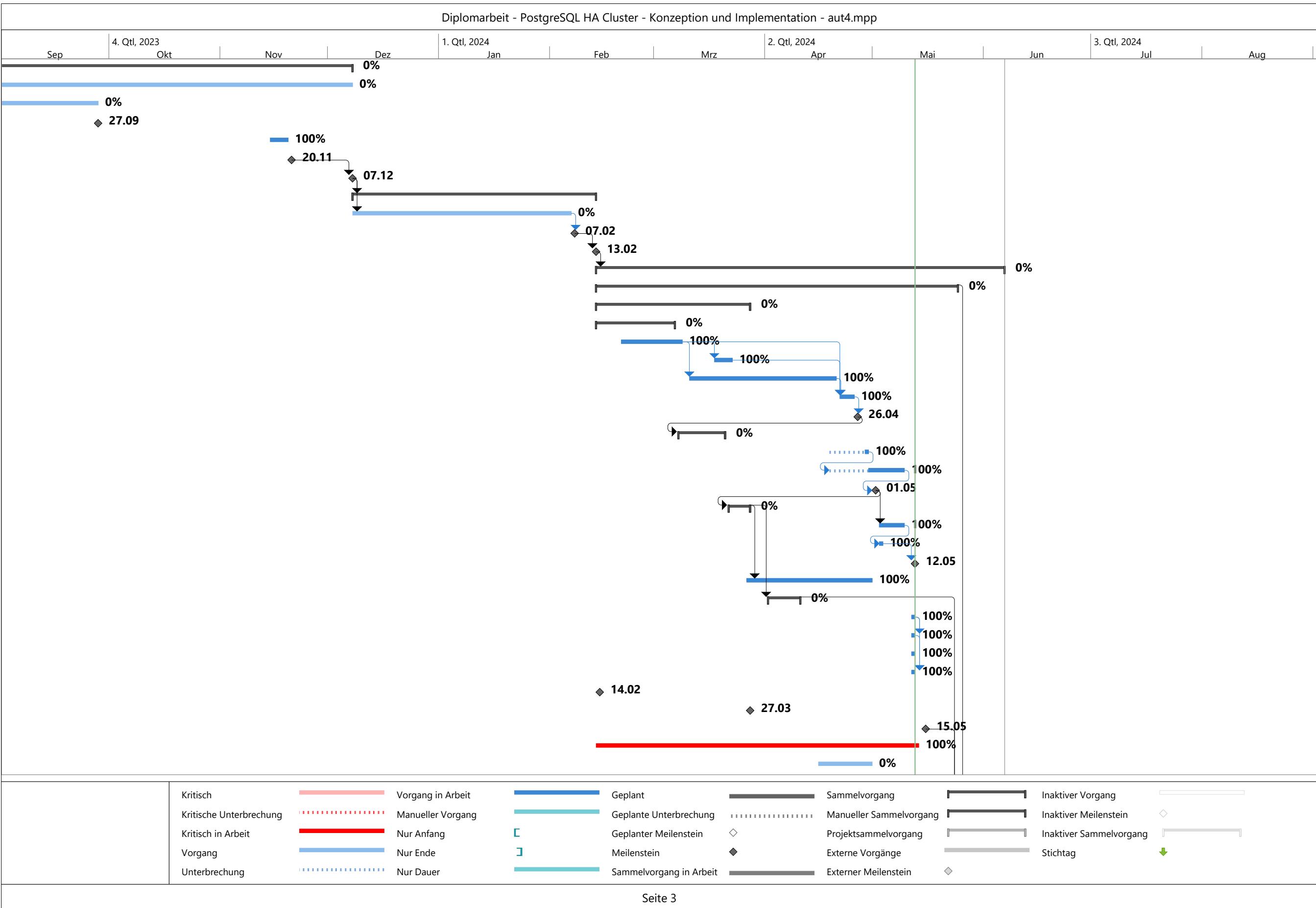
	Phase	Subphase	Nr.	Dauer [h]	Geplante Dauer [h]	Verbleibende Zeit [h]
7	Evaluation	Anforderungskatalog	1.0	4.5	16.0	11.5
10	Evaluation	Vorbereitung Benchmarking	2.0	5.0	4.0	-1.0
6	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	3.0	51.2	16.0	-35.2
8	Evaluation	Gegenüberstellung	4.0	0.5	8.0	7.5
9	Evaluation	Variantenentscheid	5.0	1.0	4.0	3.0
2	Aufbau und Implementation Testsystem	Basisinfrastruktur	6.0	3.0	4.0	1.0
3	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	7.0	20.0	20.0	0.0
4	Aufbau und Implementation Testsystem	Technical Review	8.0	2.0	3.0	1.0
19	Testing	Testing Testsystem	9.0	6.2	8.0	1.8
17	Testing	Protokollierung	10.0	2.0	4.0	2.0
18	Testing	Review und Auswertung	11.0	2.2	2.0	-0.2
20	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	12.0	11.0	8.0	-3.0
16	Resultate	Zielüberprüfung	13.0	1.5	2.0	0.5
14	Resultate	Schlussfolgerung	14.0	1.0	2.0	1.0
15	Resultate	Weiteres Vorgehen / offene Arbeiten	15.0	0.5	1.0	0.5
13	Resultate	Persönliches Fazit	16.0	0.8	2.0	1.2
0	1. Expertengespräch	1. Expertengespräch	17.0	1.0	1.0	0.0
1	2. Expertengespräch	2. Expertengespräch	18.0	1.2	1.0	-0.2
11	Letztes Expertengespräch	Letztes Expertengespräch	19.0	0.0	1.0	1.0
12	Puffer	Puffer	20.0	0.0	16.0	16.0
5	Dokumentation	Dokumentation	21.0	53.2	80.0	26.8
Total				168.0	203.0	35.0

Tabelle 2.1: Projektcontrolling



Diplomarbeit - PostgreSQL HA Cluster - Konzeption und Implementation - aut4.mpp											
Nr.	Vorgangsm. Icon	Vorgangsname	Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang	3. Qtrl, 2023		
									Jul	Aug	Sep
40	💻	Abgabe Diplomarbeit	1 Tag?	13;30;37	Ja	Fre 24.05.24	Fre 24.05.24		NV		
41	👉	Vorbereitung Präsentation	5 Tage	40	Nein	Die 28.05.24	Mon 03.06.24		NV		
42	👉	Präsentation und Fachgespräch	1 Tag?	40;41	Nein	Die 04.06.24	Die 04.06.24		NV		
43	👉	Diplomausstellung	1 Tag?	42	Nein	Mit 05.06.24	Mit 05.06.24		NV		
44	👉	Diplomfeier	1 Tag?	42;43	Nein	Don 06.06.24	Don 06.06.24		NV		

Kritisch		Vorgang in Arbeit		Geplant		Sammelvorgang		Inaktiver Vorgang	
Kritische Unterbrechung		Manueller Vorgang		Geplante Unterbrechung		Manueller Sammelvorgang		Inaktiver Meilenstein	
Kritisch in Arbeit		Nur Anfang		Geplanter Meilenstein		Projektsammelvorgang		Inaktiver Sammelvorgang	
Vorgang		Nur Ende		Meilenstein		Externe Vorgänge		Stichtag	
Unterbrechung		Nur Dauer		Sammelvorgang in Arbeit		Externer Meilenstein			





2.2 Expertengespräche

Folgende Expertengespräche fanden statt:

Fachgespräch	Datum	Fachexperte	Nebenexperte	Studenten	Bemerkungen
1	14.02.2024	Norman Süsstrunk	-	Michael Gruber Curdin Roffler	- Es wurden zwar für alle Studenten von Norman Süsstrunk Zoom-Räume bereitgestellt, aus Effizienzgründen nahmen Curdin Roffler und ich beide am selben Meeting teil
2	26.03.2024	Norman Süsstrunk	-	Michael Gruber	

Tabelle 2.2: Fachgespräche

Das Protokoll ist im Anhang zu finden.

3 Umsetzung

3.1 Evaluation

3.1.1 Exkurs Architektur

3.1.1.1 Sharding

3.1.1.1.1 Vertikales / Horizontales Sharding

Tabellen können horizontal oder vertikal partitioniert werden. Bei der vertikalen Partitionierung werden Tabellen ähnlich zur Normalisierung vertikal aufgeteilt, nur dass der Primary Key derselbe bleibt. Nachfolgend ein Beispiel zur vertikalen Partitionierung:

Primary Key	Column 1	Column 2	Column 3
1	A	B	C
2	D	E	F
3	G	H	I
4	J	K	L
5	M	N	O
6	P	Q	R

Komplette Tabelle

Primary Key	Column 3
1	C
2	F
3	I
4	L
5	O
6	R

Primary Key	Column 1	Column 2
1	A	B
2	D	E
3	G	H
4	J	K
5	M	N
6	P	Q

Vertikale Partitionen

Abbildung 3.1: Sharding - Vertikale Partitionierung

Die horizontale Partitionierung geht den umgekehrten Weg. Die Datensätze einer Tabelle werden anhand bestimmter Regeln in Partitionen mit gleicher Struktur verteilt. Nachfolgend ein Beispiel einer horizontalen Partitionierung:

Primary Key	Column 1	Column 2	Column 3
1	A	B	C
2	D	E	F
3	G	H	I
4	J	K	L
5	M	N	O
6	P	Q	R

Primary Key	Column 1	Column 2	Column 3
1	A	B	C
6	P	Q	R

Primary Key	Column 1	Column 2	Column 3
2	D	E	F
3	G	H	I

Primary Key	Column 1	Column 2	Column 3
4	J	K	L
5	M	N	O

Primary Key	Column 1	Column 2	Column 3
6	P	Q	R
7	S	T	U

Abbildung 3.2: Sharding - Horizontales Partitionierung

Horizontales Partitionieren wird oft für das Sharding von Tabellen benutzt. Die Partitionen entsprechen dann den Shards.

3.1.1.1.2 Key Based Sharding

Hierbei wird das Sharding anhand eines oder mehreren Keys ausgeführt.

3.1.1.1.3 Range Based Sharding

Das Sharding wird dabei anhand von Ranges ausgeführt. Zum Beispiel anhand von Preis-Ranges.

3.1.1.1.4 Directory Based Sharding

Hierfür wird eine Lookup-Tabelle geführt, welche die Schlüssel für das Sharding bereitstellen. Anhand dieser werden dann die entsprechenden Zieltabellen aufgeteilt.

3.1.1.1.5 Hash Based Sharding

Das Hash Based Sharding ist eine Form des Range Based Shardings, bei dem Hashwerte der Datensätze benutzt werden. Je nach Bereich wird der Datensatz dann einem Shard zugewiesen.

3.1.1.2 Monolithische vs. verteilte SQL Systeme

Klassische SQL-Datenbanken sind monolithische Systeme, selbst wenn sie mit Hilfe Replikation eine Primary/Stand-by-Architektur aufweisen. Man kann mit eines SQL-Proxys ein gewisses Mass an Load Balancing betreiben, hat aber immer noch das Problem, dass es einen primären Node gibt, auf dem beschrieben wird. Monolithische Systeme sind daher nicht Cloud Native.

Nur verteilte Systeme, sogenannte Distributed SQL Systeme, sind wirklich Cloud Native fähig, da die Last gleichmäßig auf horizontal skalierenden Nodes verteilt werden kann.

Nachfolgend die schematische Darstellung der beiden Konzepte:

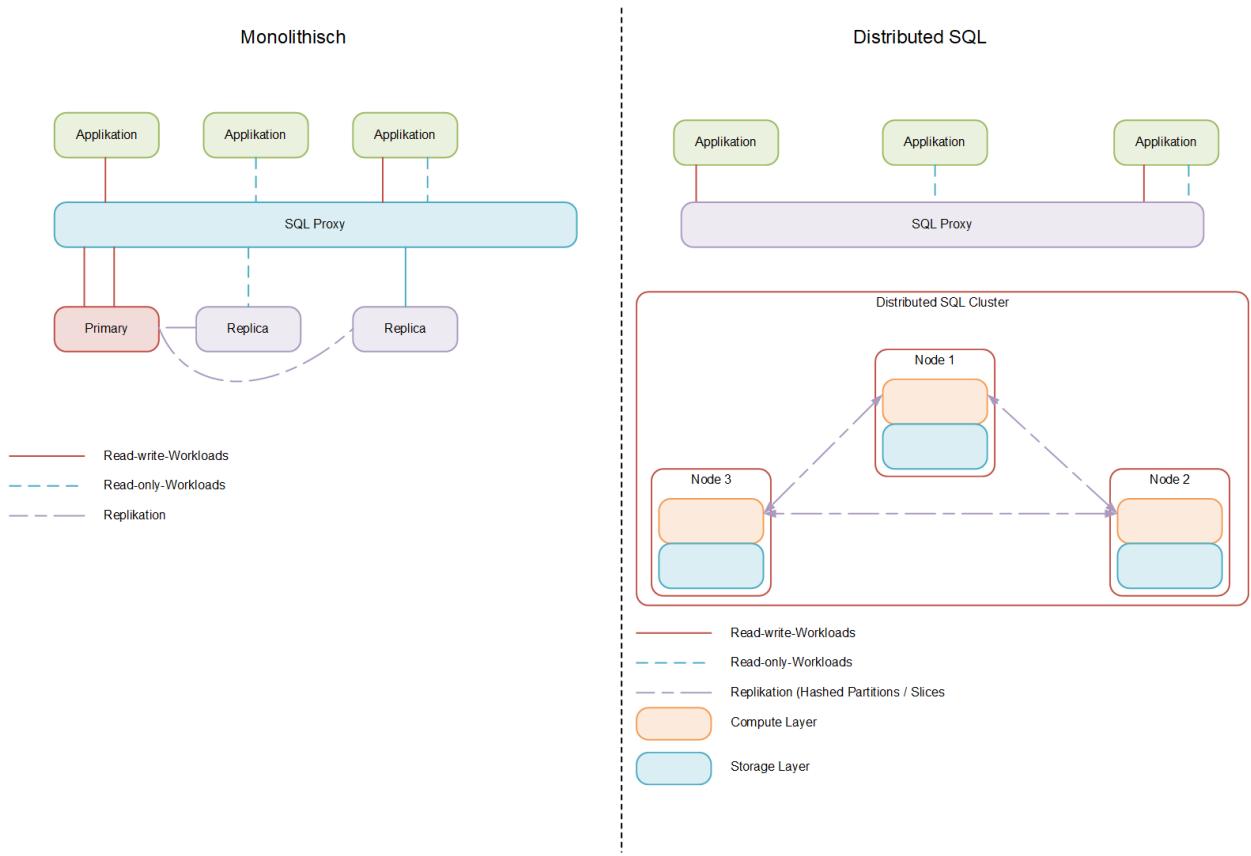


Abbildung 3.3: Monolithische vs. verteilte SQL Systeme

3.1.1.3 High Availability und Replikation

Wenn eine Datenbank HA (High Availability), also **hochverfügbar sein** soll, braucht es eine **Primäre** und mindestens eine **Sekundäre-** oder Failover-Datenbank. Um Datenverlust zu vermeiden, müssen die Daten permanent von der **primären** auf die **sekundäre** Datenbank synchronisiert werden, dies nennt man **Replikation**[79]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

Synchrone Replikation

Wenn bei einer synchronen Replikation eine Transaktion abgesetzt wird, wird der Commit auf der primären Seite erst gesetzt, wenn die Änderung auf der sekundären Seite oder den sekundären Seiten ebenfalls eingetragen und committet ist. Bis zu diesem Moment ist die Transaktion nicht als **committet**.

Dies wird dann zum Problem, wenn keine Verbindung mehr zu **mindesten** einer sekundären Seite vorhanden ist. Zudem wird die synchrone Replikation bei hohen Latenzen zum Bottleneck der Datenbank.

Asynchrone Replikation

Bei der asynchronen Replikation wird eine Transaktion erst auf der eigenen primären Seite committet und erst dann an die sekundären Nodes gesendet. Besonders bei hohen Latenzen bleibt die Datenbank immer perfomant, allerdings kann es je nach Latenz und genereller Auslastung zu Datenverlusten kommen, wenn es zum Failover kommt.

3.1.1.4 Quorum

Ein Quorum-System soll die Integrität und Konsistenz in einem Datenbank-Cluster sicherstellen. Dabei gilt zu beachten, dass nicht eine beliebige Anzahl an Nodes hinzugefügt werden können. Auch hat das Hinzufügen von Nodes immer eine Einbusse an Performance zur Folge, besonders dann, wenn eine synchrone Replikation gewählt wird und auf jedes Commitment von den Replica-Nodes gewartet werden muss.

Quorum

Die Mehrheit der Server, die einen funktionierenden Betrieb gewährleisten können, ohne eine Split-brain-Situation zu erzeugen. Die Formel ist gemeinhin $n/2 + 1$

Throughput

Beschreibt, wie sich die Anzahl Nodes auf die Schreibgeschwindigkeit der Commitments auf die restlichen Nodes auswirkt.

Die Verdopplung der Server halbiert i. d. R. den Throughput.

Fehlertoleranz

Beschreibt, wie viele Nodes ausfallen können, damit der Cluster noch arbeitsfähig ist.

Wobei eine Erhöhung der Nodes von 3 auf 4 die Fehlertoleranz nicht erhöht, da nun eine Split-brain-Situation entstehen kann.

Hier ein Beispiel wie sie in den Artikeln [77, 89, 73] beschrieben werden. Es zeigt auf, ab wie vielen Nodes die Fehlertoleranz erhöht wird und wie sich der repräsentative Throughput verhält.

Anzahl Nodes	Quorum	Fehlertoleranz	Representative Throughput
1	1	0	100
2	2	0	85
3	2	1	82
4	3	1	57
5	3	2	48
6	4	2	41
7	4	3	36

Tabelle 3.1: Quorum Beispiele

3.1.1.5 CAP-Theorem

Das CAP-Theorem besagt, dass nur zwei der drei folgenden drei Merkmale von verteilten Systemen gewährleistet werden können[62].

Konsistenz - Consistency

Die Datenbank ist konsistent, alle Clients sehen gleichzeitig die gleichen Daten unabhängig, auf welchem Node zugegriffen wird. Hierzu muss eine Replikation der Daten an alle Nodes stattfinden und der Commit zurückgegeben werden, also eine synchrone Replikation stattfinden.

Verfügbarkeit - Availability

Jeder Client, der eine Anfrage sendet, muss auch eine Antwort erhalten. Unabhängig davon, wie viele Nodes im Cluster noch aktiv sind.

Ausfalltoleranz / Partitionstoleranz - Partition tolerance

Der Cluster muss auch dann noch funktionsfähig bleiben, wenn es eine beliebige Anzahl von Verbindungsunterbrüchen oder anderen Netzwerkproblemen zwischen den Nodes gibt.

Das CAP-Theorem lässt sich am einfachsten mit folgenden Venn-Diagramm visualisieren:

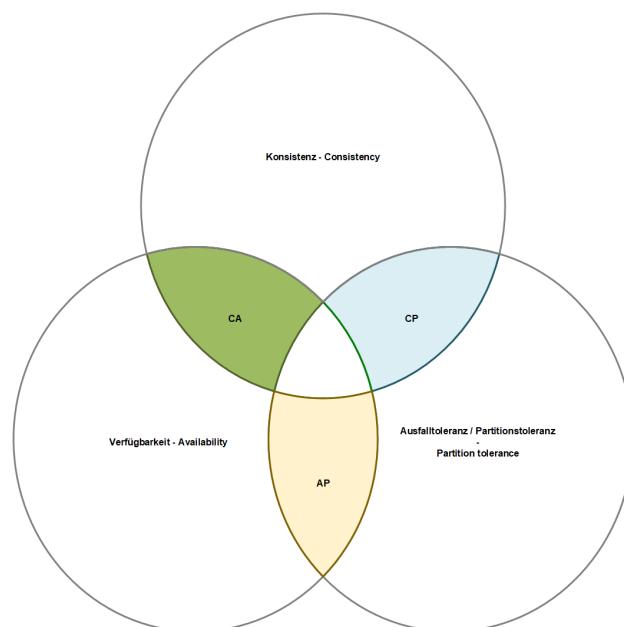


Abbildung 3.4: CAP-Theorem

PostgreSQL, Oracle Database oder IBM DB2 präferieren CA, also Konsistenz und Verfügbarkeit.

3.1.1.6 Skalierung

Datenbanken müssen skalierbar sein. Dabei wird unterschieden zwischen einer vertikalen Skalierung (Scale-up) und horizontaler Skalierung (Scale-out). Bei der vertikalen Skalierung

werden den DB-Servern mehr CPU-Cores und Memory sowie zum Teil Storage hinzugefügt, wobei der Storage in jedem Fall wachsen wird. Beim horizontalen Skalieren werden weitere DB-Nodes in den Cluster eingehängt[75]:

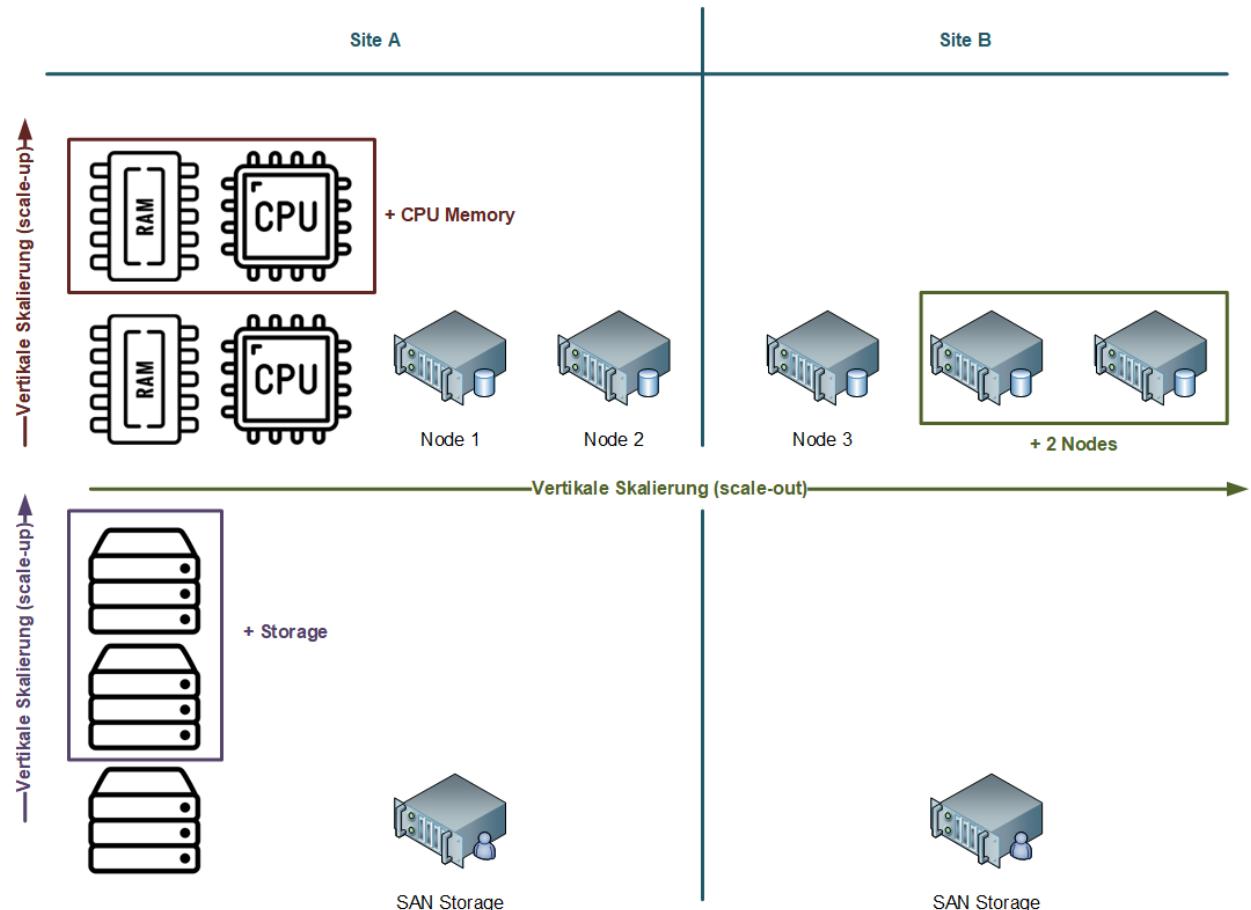


Abbildung 3.5: Datenbank skalierung

Bei monolithischen Datenbanken werden irgendwann die Grenzen der horizontalen Skalierung erreicht und man muss wieder vertikal skalieren, um dem Primary Node genügend Rechnerleistung vorzuhalten.

3.1.2 Erheben und Gewichten der Anforderungen

3.1.2.1 Anforderungen

Das KSGR hat eine Cloud First Strategie.

Das heisst, alle neuen Applikationen und entsprechend deren Datenbanken müssen Cloud Ready bzw. Cloud Native sein. Um die Voraussetzung dafür zu schaffen, muss auch der PostgreSQL Cluster Cloud Ready sein.

Daher müssen zwei von drei genauer evaluierten Lösungen Cloud Native Lösungen sein. Wenn

der Zeitaufwand reicht, können auch eine Cloud Native und Monolithisches System aufgebaut werden.

Nr.	Anforderung	Bezeichnung	Beschreibung	System	Muss / Kann
1	Systemvielfalt		Es muss mindestens eine Monolithische und mindestens 2 zwei Distributed SQL Cluster ermittelt werden	Beides	MUSS
2	Synergien		Skripte und APIs des Monolithischen Systems müssen auch in einem Distributed SQL System verwendet werden können	Beides	MUSS
3	Failover	Automatismus	Das Clustersystem muss bei einem Nodeausfall automatisch auf einen anderen Node umgestellt	Beides	MUSS
4	Failover	Connection - Stabilität	Beim Failover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
5	Failover	Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
6	Switchover	Skript / API	Das System muss ein Skript oder eine API liefern, welche einen geordneten Switchover auf einen anderen Node erlaubt	Beides	MUSS
7	Switchover	Connection - Stabilität	Beim Switchover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
8	Switchover	Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
9	Restore	Skript / API	Das Clustersystem muss ein Skript oder eine API liefern, welche das einfache und ggf. automatisierte Restore eines oder mehreren Nodes ermöglichen	Beides	MUSS
10	Restore	Datensicherheit	Beim Wiederherstellen des Ursprungszustands darf es zu keinem Datenverlust kommen	Beides	MUSS
11	Restore	Connection - Stabilität	Bei der Wiederherstellung einzelner Nodes darf es zu keinen Unterbrechungen auf den Applikationen kommen Das Wiederherstellen des Ursprungszustands muss	Beides	MUSS
12	Restore	Geschwindigkeit	innert weniger Stunden für alle Datenbanken aus dem Backup Wiederhergestellt und im Clustersystem Synchronisiert werden	Beides	MUSS
13	Replikation	Synchrone Replikation	Es muss eine Synchrone Replikation sichergestellt werden	Monolithisch	MUSS
14	Replikation	Failover / Switchover Garantie	Die Replikation muss sicherstellen, dass es bei einem Failover/Switchover zu keinem Fehler kommt	Monolithisch	MUSS
15	Replikation	Throughput	Beschreibt, wie viele Transaktionen pro Zeiteinheit vom Primary an die Replikas gesendet und Committed werden. Dieser Wert ist bei Synchrone Replikation entscheidend da Commits auf allen Replicas abgesetzt sein müssen.	Beides	MUSS
16	Sharding	Datenschutz- und Integrität	Die Datenkonsistenz und Datenintegrität auf den Shards muss sichergestellt werden	Distributed SQL	MUSS
17	Sharding	Schutz vor Datenverlust	Die Synchronisation der Shards muss sicherstellen, dass es zu keinem Datenverlust kommt	Distributed SQL	MUSS
18	Quorum	Quorum-System vorhanden	Das Clustersystem muss über ein Quorum-System besitzen	Beides	MUSS
19	Quorum	Robustheit	Das Quorum des Clustersystems muss robust genug sein, um eine Split-Brain-Situation zu verhindern	Beides	MUSS
20	Connection		Das Clustersystem muss sicherstellen, dass eine Applikation ohne Entwicklungsaufwand mittels dem PostgreSQL Wired Connector zugreifen kann Das Clustersystem muss Skripte oder eine API liefern, mit dem das System zu konfigurieren, verwalten oder überwachen zu können.	Beides	MUSS
21	Management-API	Management-API vorhanden	Zudem müssen mit geringen Arbeitsaufwand damit Nodes hinzugefügt oder entfernt werden können	Beides	MUSS
22	Management-API	Authentifizierung & Autorisierung	Es müssen gängige Standards für Authentifizierung und Autorisierung mitgebracht werden	Beides	MUSS
23	Management-API	Aufwand	Der Aufwand, der benötigt wird um die DB zu verwalten, Nodes hinzuzufügen oder zu entfernen usw. muss gegeneinander verglichen werden.	Beides	MUSS
24	Backup	Backup mit PostgreSQL Standards	Backups müssen mittels PostgreSQL Standards angezogen werden	Beides	MUSS
25	Backup	Restore mit PostgreSQL Standards	Backups müssen mittels PostgreSQL Standards restored werden können	Beides	MUSS
26	Housekeeping - Log Rotation		Das Clustersystem muss die Möglichkeit zur Log Rotation bieten	Beides	MUSS
27	Self Healing		Das Clustersystem muss im Fehlerfall Nodes selber wiederherstellen können Läuft ein Node auf einen Fehler,	Beides	KANN
28	Monitoring - Node Failure		muss das Clustersystem dies erkennen und Melden resp. eine Schnittstelle liefern die abgefragt werden kann Da die meisten PostgreSQL HA Lösungen Open-Source sind,	Beides	MUSS
29	Maintenance Quality		muss sichergestellt werden, dass die gewählte Lösung auch aktiv gepflegt wird. Als Basis dienen hier Informationen wie z.B. GitHub Insights.	Beides	MUSS
30	Performance	tps - Read-Only	Die Transaktionsrate (transactions per second / tps) für DQL Transaktionen	Beides	MUSS
31	Performance	tps - Read-Writes	Die Transaktionsrate (transactions per second / tps) für DML Transaktionen	Beides	MUSS
32	Performance	Ø Latenz - Read-Only	Die Latenzzeit bei DQL Transaktionen	Beides	MUSS
33	Performance	Ø Latenz - Read-Write	Die Latenzzeit bei DML Transaktionen	Beides	MUSS

3.1.2.2 Stakeholder

Rolle	Funktion	Departement	Bereich	Abteilung
Zabbix Stakeholder	Abteilungsleiter	D10 ICT	Infrastrukturmanagement	ICT Netzwerk, Security und Comm.
Stakeholder Data Center Infrastruktur	Abteilungsleiter	D10 ICT	Infrastrukturmanagement	ICT Data Center
k8s Stakeholder	ICT System Ingenieur	D10 ICT	Infrastrukturmanagement	ICT Data Center

Tabelle 3.3: Stakeholder

3.1.2.3 Gewichtung

Die Gewichtung wurde mit Hilfe einer Präferenzmatrix ermittelt.

Die grundlegende Gewichtung wurde folgendermassen vorgenommen:

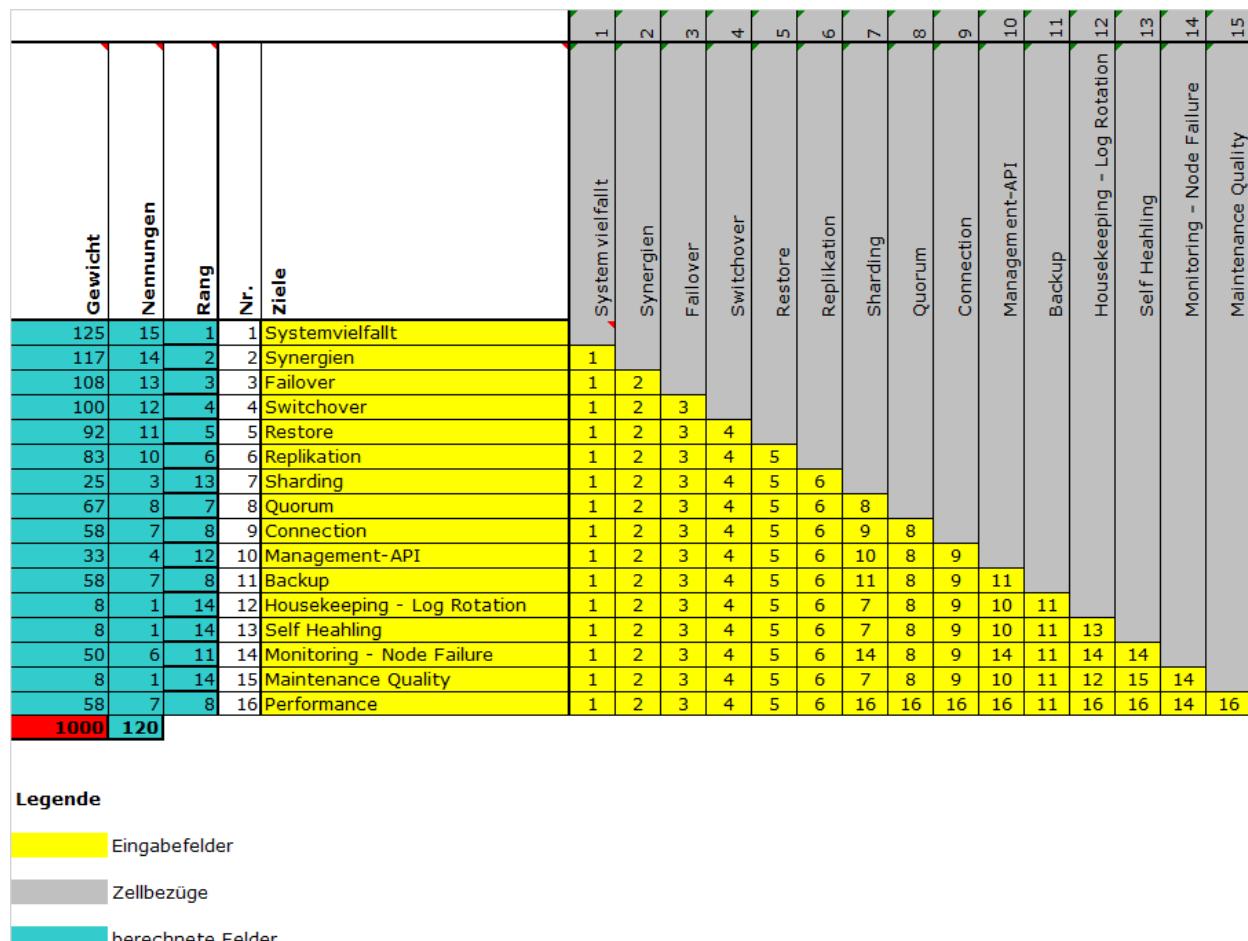


Abbildung 3.6: Präferenzmatrix

3.1.3 Testziele erarbeiten

3.1.3.1 Evaluation - Testfälle

3.1.3.1.1 Patroni

Failover

1. Der Server des primären Node wird manuell heruntergefahren.
2. Während des Failover müssen Daten via SQL eingeführt und ausgelesen werden.
3. Während des Failover muss mindestens eine längere Abfrage gestartet werden.

Switchover

4. Mit der REST-API wird der Switchover auf einen anderen Nod abgesetzt.
5. Mit dem `patronictl`-Command wird der Switchover gesetzt
6. Während dem Switchover müssen Daten via SQL eingeführt und ausgelesen werden.
7. Während dem Switchover muss mindestens eine längere Abfrage gestartet werden.

Restore

8. Mit der REST-API wird der Node erst mit dem `reinitialize` wiederhergestellt und dann mit einem Switchover wieder als Primary gesetzt.
9. Mit dem `patronictl`-Command und Parameter `reinit` der Node wiederhergestellt und abschliessend mit Hilfe Switchover wieder als Primary gesetzt.
10. Mit der REST-API wird der Node mit dem `reinitialize` wiederhergestellt
11. Mit dem `patronictl`-Command und Parameter `reinit` der Node wiederhergestellt
12. Vor, während und nach dem Restore müssen Tabellen mit Foreign-Key-Constraints und Daten geprüft werden.
13. Während des Restore muss mindestens eine längere Abfrage gestartet werden und Daten via SQL eingeführt und ausgelesen werden.

3.1.3.1.2 StackGres - Citus

Failover

1. Der Server des Leader-Coordinator-Node wird manuell heruntergefahren.
2. Während des Failover müssen Daten via SQL eingeführt und ausgelesen werden.
3. Während des Failover muss mindestens eine längere Abfrage gestartet werden.

Sharding

4. Vor, während und nach dem Failover müssen Tabellen mit Foreign-Key-Constraints geprüft werden.
5. Nach einem Failover-Test müssen alle Daten vorhanden sein.

Self Healing

6. Der Node muss wieder hochgefahren werden.
Der Node muss selbstständig Daten synchronisieren.
7. Der Leader muss, falls notwendig, automatisch neu gesetzt werden.

3.1.3.1.3 YugabyteDB

Failover

1. Ein k8s Node wird manuell heruntergefahren,
indem der entsprechende Server heruntergefahren wird.
2. Während des Failover müssen Daten via SQL
eingeführt und ausgelesen werden.
3. Während des Failover muss mindestens eine längere Abfrage gestartet werden.

Sharding

4. Vor, während und nach dem Failover müssen Tabellen mit Foreign-Key-Constraints geprüft werden.
5. Nach einem Failover-Test müssen alle Daten vorhanden sein.

Self Healing

6. Der Node muss wieder hochgefahren werden.
Der Node muss selbstständig Daten synchronisieren.

3.1.3.2 Evaluation - ERD self_healing_test

Die Tests müssen bei allen drei Varianten anhand der Datenbank `self_healing_test` durchgeführt werden.

Dabei werden die Tabellen im Hinblick auf das Citus Schema Based Sharding in Schemas organisiert.

Zwischen den einzelnen Schemas sollen einige Tabellen einen Foreign-Key auf andere Tabellen legen:

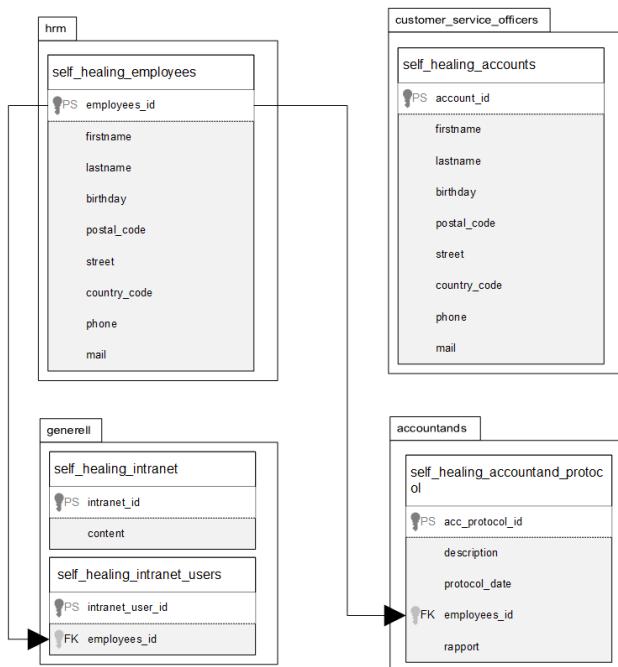


Abbildung 3.7: Testing - ERD DB self_healing_test

3.1.4 PostgreSQL Benchmarking

3.1.4.1 pgBench - Basis-Benchmarking

PostgreSQL bietet ein Benchmarking-Tool[69, 2], mit dem die DB vermessen werden kann.

Damit die Tests aussagekräftig sind, werden mit den Testläufen mehrere Läufe gestartet. Der erste Lauf muss dabei ignoriert werden, denn erst dann wird die DB in den Cache geladen. Wird dies nicht eingehalten, so wird die ganze Testreihe unbrauchbar.

3.1.4.2 Replication Throughput Benchmarking

Etwas komplexer wird es beim Benchmark des Throughput. Den nebst den DB-Latenzen usw. kommt die Netzwerklatenz sowie andere Latenzen hinzu [87].

3.1.4.3 Benchmark Settings

Das Mass aller Dinge ist die Zabbix-DB.

Sie wird vorerst die grössten Zugriffszahlen, das höchste Datenwachstum und die meisten Transaktionen erzeugen.

Es werden alle Switches sowie der grösste Teil der Router erfasst, es sind im Moment etwas mehr als 32'000 Items erfasst.

Ein Item kann ein Gerät, ein Port oder mehrere States pro Port sein:

System information		
Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (enabled/disabled)	250	240 / 10
Number of templates	345	
Number of items (enabled/disabled/not supported)	323479	318628 / 4796 / 55
Number of triggers (enabled/disabled [problem/ok])	136777	135529 / 1248 [148 / 135381]
Number of users (online)	24	3
Required server performance, new values per second	950.86	
High availability cluster	Disabled	

Abbildung 3.8: Benchmark Settings - Zabbix - Systeminformationen

Pro Sekunde werden ca. 950 Datenpunkte abgeholt.

Da der Grossteil der Netzwerksysteme aber erfasst sind, wird die Anzahl Items nicht mehr stark anwachsen.

Auf die Datenbank wird sehr stark zugegriffen. Es werden bis zu 23 Connections pro Sekunde ausgeführt:

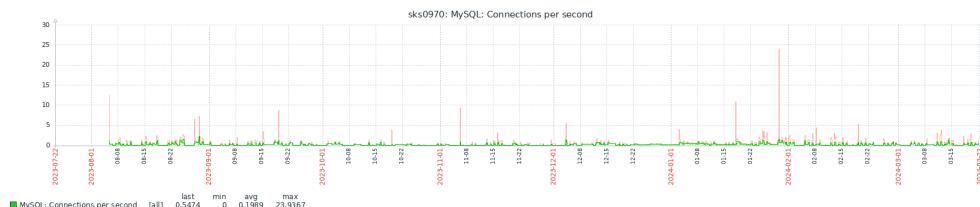


Abbildung 3.9: Benchmark Settings - Zabbix - Connections per Seconds

Pro Sekunde wurden bisher bis zu über 7'000 Queries ausgeführt. Dies schliesst Abfragen von Stored Programs ein:

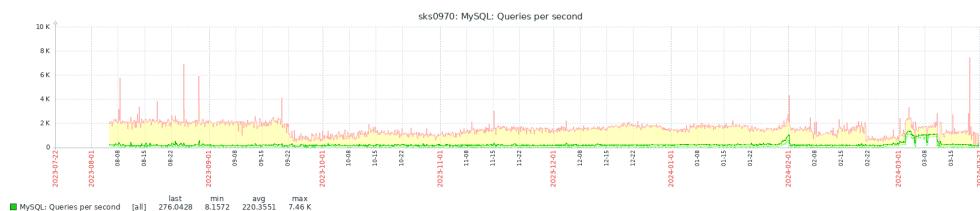


Abbildung 3.10: Benchmark Settings - Zabbix - Queries per Seconds

Reine Client anfragen waren nichtsdestotrotz über 4'000 Queries pro Sekunde:

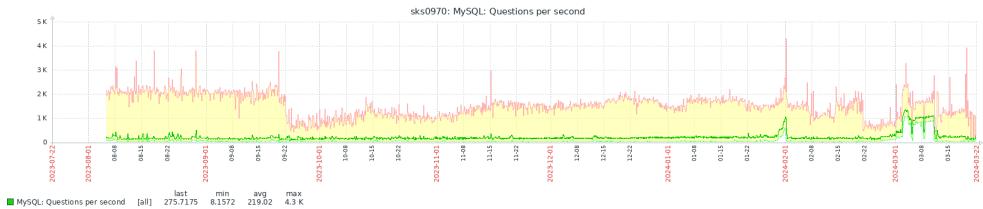


Abbildung 3.11: Benchmark Settings - Zabbix - Client Queries per Seconds

Auch das Wachstum ist beachtlich. Die DB startete mit 180GiB und ist zurzeit bei rund 232GiB, war aber schon bei 238GiB:

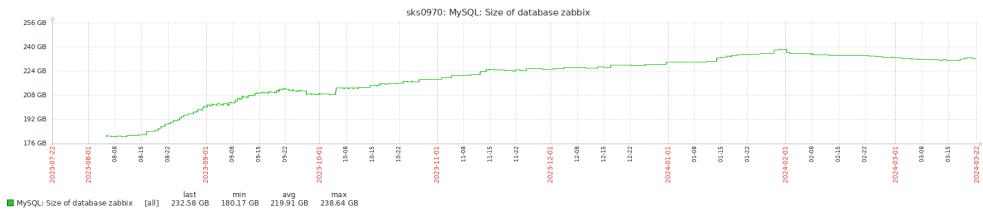


Abbildung 3.12: Benchmark Settings - Zabbix - DB Size

Nun kommen noch die kleineren DBs hinzu. Heisst, für den Mixed Benchmark (DML und DQL Transaktionen) werden folgende Werte und Parameter gesetzt:

Typ	Parameter	pgbench-Parameter	1. Lauf	2. Lauf	3. Lauf	4. Lauf
mixed	Datenbank		pgbench_eval_bench	pgbench_eval_bench	pgbench_eval_bench	pgbench_eval_bench
mixed	DB-Grösse		5GiB	15GiB	50GiB	250GiB
mixed	1. Iteration Lauf ignorieren		ja	ja	ja	ja
mixed	Select only	-S	nein	nein	nein	nein
mixed	Iterationen pro Lauf		4	4	4	4
mixed	Vacuum	-v	ja	ja	ja	ja
mixed	Separate Connects	-C	ja	ja	ja	ja
mixed	Client count	-c	10	50	100	1000
mixed	Anzahl Transaktionen pro Client	-t	10	50	50	7
mixed	Anzahl Transaktionen Total		100	2500	5000	7000
mixed	Anzahl Worker Threads	-j	4	4	4	4

Tabelle 3.4: Benchmark Settings - Mixed Transaktionen

Für den DQL-Only Benchmark wird mit folgenden Konfigurationen gearbeitet:

Typ	Parameter	pgbench-Parameter	1. Lauf	2. Lauf	3. Lauf	4. Lauf
dql	Datenbank		pgbench_eval_bench	pgbench_eval_bench	pgbench_eval_bench	pgbench_eval_bench
dql	DB-Grösse		5GiB	15GiB	50GiB	250GiB
dql	1. Iteration Lauf ignorieren	-S	ja	ja	ja	ja
dql	Select only		ja	ja	ja	ja
dql	Iterationen pro Lauf		4	4	4	4
dql	Vacuum	-v	ja	ja	ja	ja
dql	Separate Connects	-C	ja	ja	ja	ja
dql	Client count	-c	10	50	100	1000
dql	Anzahl Transaktionen pro Client	-t	10	50	50	7
dql	Anzahl Worker Threads	-j	4	4	4	4

Tabelle 3.5: Benchmark Settings - DQL Transaktionen

Bei pgbench kann nicht die Grösse angegeben werden.

Es muss der Skalierungsfaktor angepasst werden.

Dieser legt allerdings fest, wie viele Daten gespeichert werden.

Wird eine 1 eingeben, so werden 100'000 Records angelegt.

Um nun auf eine gewisse **grösse** zu kommen, gibt es verschiedene Formeln.

Die als beste stellte sich folgende heraus[70]:

Zielobjekt	Skalierungsfaktor Formel
DB	$0.0669 * DB - Zielgrsse - 0.5$
Tabelle (pgbench_accounts / ysql_bench_accounts)	$0.0781 * Tabelle - Zielgrsse$
Index (pgbench_accounts_pkey / ysql_bench_accounts_pkey)	$0.4668 * Index - Zielgrsse$

Tabelle 3.6: Benchmark Settings - Skalierungsfaktoren

Daraus errechnen sich für die DB-Größen des Benchmark-Settings folgende Skalierungsfaktoren:

DB Grösse [GiB]	DB Grösse [MiB]	Scale Faktor
5	5120	342
15	15360	1027
50	51200	3425
250	256000	17126

Tabelle 3.7: Benchmark Settings - Datenbankgrößen / Skalierungsfaktor

YugabyteDB speichert die Daten anders als PostgreSQL, nämlich als Key-Value-Store.

Das verhindert, dass die DB Grösse nicht ausgelesen werden kann, nur die Tabellen sind ersichtlich.

Um einen Vergleich zu haben, muss daher die Tabellengrösse berechnet werden.

Der Skalierungsfaktor für die Tabellen ist folgendermassen aufgebaut:

DB Grösse [GiB]	DB Grösse [MiB]	Scale Faktor
5	5120	400
15	15360	1200
50	51200	3999
250	256000	19994

Tabelle 3.8: Benchmark Settings - Tabellengrössen / Skalierungsfaktor

Der Skalierungsfaktor wird pro 100'000 gerechnet, gebe ich also den Faktor 1 ein, werden

100'000 Zeilen in der Tabelle pgbench_accounts resp. ysql_bench_accounts erzeugt.

Entsprechend wachsen die Anzahl Records wie folgt an:

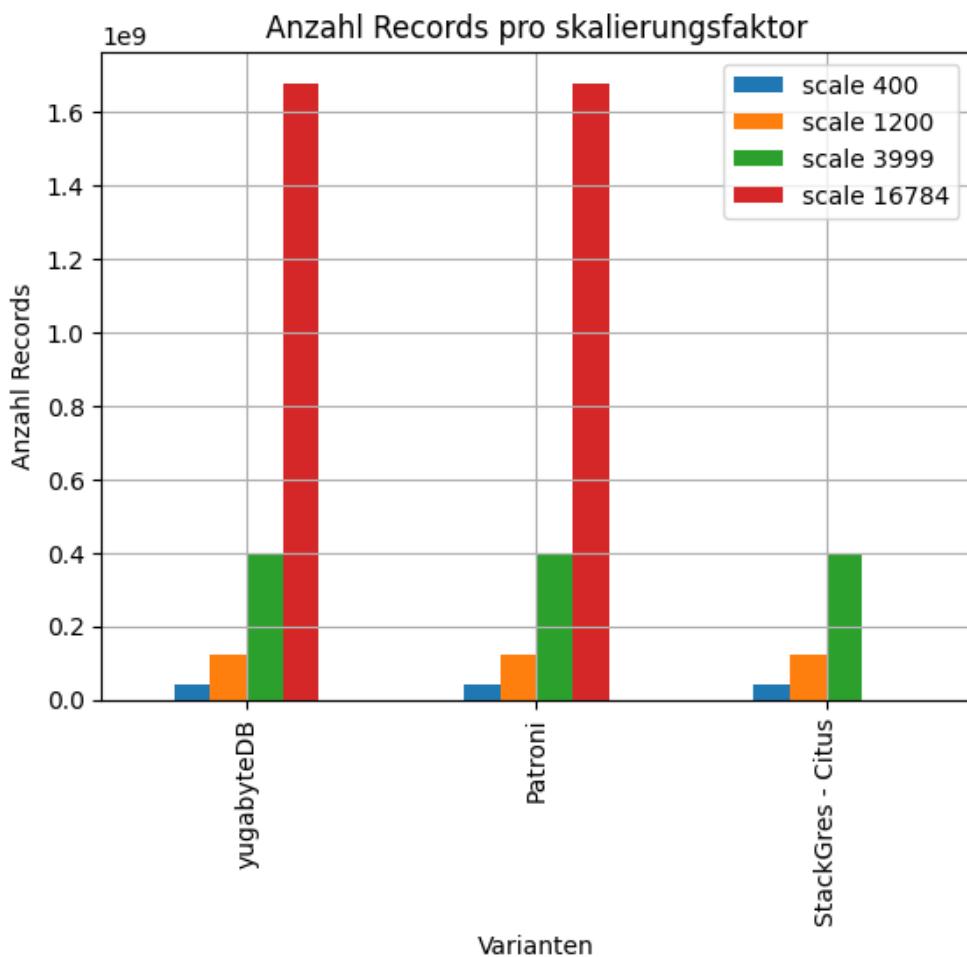


Abbildung 3.13: Benchmark Settings - Anzahl Records / Skalierungsfaktor

3.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen

3.1.5.1 Percona

Percona bietet nebst dem bekannteren Galera-Cluster und MySQL- und MariaDB auch Dienstleistungen [47] aller Art für PostgreSQL an.

Percona arbeitet oft auf Basis von Patroni,
bietet aber auch eigene Erweiterungen und eigene Software an[42].

Da Percona keine Open-Source-Lösungen anbietet, sondern Service-orientiert ist, wird Percona nicht betrachtet.

Allerdings wird immer wieder auf frei zugängliche Dokumentationen von Percona verwiesen werden.

3.1.5.2 EnterpriseDB

EDB (EnterpriseDB) ist ein führendes Software- und Servicehaus für PostgreSQL[25]. Nebst Dienstleistungen für das Management von PostgreSQL-Umgebungen, Migrationen aus Oracle-Umgebungen und **anderem** bietet EDB auch eine Vielzahl an zusätzlicher Software und eigene Replikationslösungen.

EDB bietet PostgreSQL auf Kubernetes mit Hilfe CloudNativePG an, hat aber auch eine eigens entwickelte Distributed SQL / Active-Active Lösungen.

Da die EDB-Lösungen nicht Open-Source sind resp. von EDB aufgekauft Lösungen nicht mehr ohne Subscription betreibbar sind, werden sie nicht berücksichtigt. Allerdings wird immer wieder auf frei zugängliche Dokumentationen von EDB verwiesen werden.

3.1.5.3 KSGR Lösung

Das Kantonsspital Graubünden hat prüft mit keepalived ob die primäre Seite erreichbar und betriebsbereit ist. Trifft dies nicht mehr zu, wird ein Failover durchgeführt[91]. Ist die primäre Seite wieder verfügbar, wird ein Restore auf die primäre Seite gefahren.

Beim Restore wird ein komplettes Backup der sekundären Seite auf die primäre Seite übertragen. Ursache ist, dass die normalerweise für den Datenrestore benötigten PostgreSQL Board Mittel nur für eine relativ kurze Zeit eingesetzt werden können, ehe die Differenzen zwischen den beiden Seiten zu gross werden. Bei kleinen Datenbanken wie jene für Harbor und GitLab ist die Zeit die hierfür benötigt wird, nicht relevant. Sind die Datenbanken auf dem PostgreSQL Cluster jedoch grösser, kann der Restore mehrere Minuten dauern.

3.1.5.4 pgpool-II

pgpool-II ist eine Middleware die zwischen einem PostgreSQL Cluster und einem PostgreSQL-Client gesetzt wird.

3.1.5.4.1 Core-Features

pgpool-II bietet folgende Core-Feature[67]:

- Watchdog für **Automatischer** Failover
- Eigener Quorum-Algorithmus
- Integrierter Connection Pooler

- Eigenes Replikationssystem
- Integriertes Load Balancing
- Limiting Exceeding Connections, also queuen von Connections
- In Memory Query Caching
- Online Node Recovery / Resynchronisation

3.1.5.4.2 Replikation

pgpool-II bietet ein eigenes Replikationssystem an.

Es besteht allerdings die Möglichkeit, die PostgreSQL-Standardreplikationen zu verwenden.

3.1.5.4.3 Proxy

pgpool-II hat bereits einen integrierten Load Balancer.

Einen zusätzlichen Proxy wird daher nicht benötigt.

3.1.5.4.4 Pooling

pgpool-II bietet ebenfalls von Haus aus einen Connection Pooler.

3.1.5.4.5 API / Skripte

pgpool-II bietet mit pgpool ein eigenes Command- und Toolset an.

Mit dem CLI-Tool pcp_command bietet pgpool-II zudem über eine abgesicherte API, die auch via Netzwerk erreichbar ist.

3.1.5.4.6 Maintenance

pgpool-II hat kein GitLab- oder GitHub-Repository. Metriken wie die GitHub Insights, welche Aufschluss über den Zustand des Projekts geben, finden sich nicht.

Die Dokumentation wird zwar aktualisiert, ist aber sehr minimalistisch gehalten. Sie bietet nur wenig Informationen zum Aufbau und Architektur von pgpool-II.

3.1.5.5 pg_auto_failover

pg_auto_failover ist eine PostgreSQL-Erweiterung, die von der Microsoft Subunternehmen Citus Data entwickelt wird.

3.1.5.1 Core-Features

Die wichtigsten Features von pg_auto_failover sind:

- API
- PostgreSQL Extension, also reines PostgreSQL
- State Machine Driven
- Replikations-Quorum
- Citus kompatibel
- Azure VM Support

3.1.5.2 Replikation

pg_auto_failover bietet die Standard PostgreSQL-Replikationen.

3.1.5.3 Proxy

pg_auto_failover benötigt einen HAProxy, um Load Balancing betreiben zu können[30].

3.1.5.4 API / Skripte

pg_auto_failover bietet ein eigenes CLI-Tool, pg_autoctl. Dieses stellt Commands für das Einbinden neuer Nodes,
das Managen von Nodes (Maintenance resp. Switchover),
Konfigurieren oder Monitoren des Systems zur Verfügung[38].

3.1.5.5 Architektur

Die Dokumentation resp. Grafik von pg_auto_failover [8] zeigt auf, wie der Failover funktioniert:

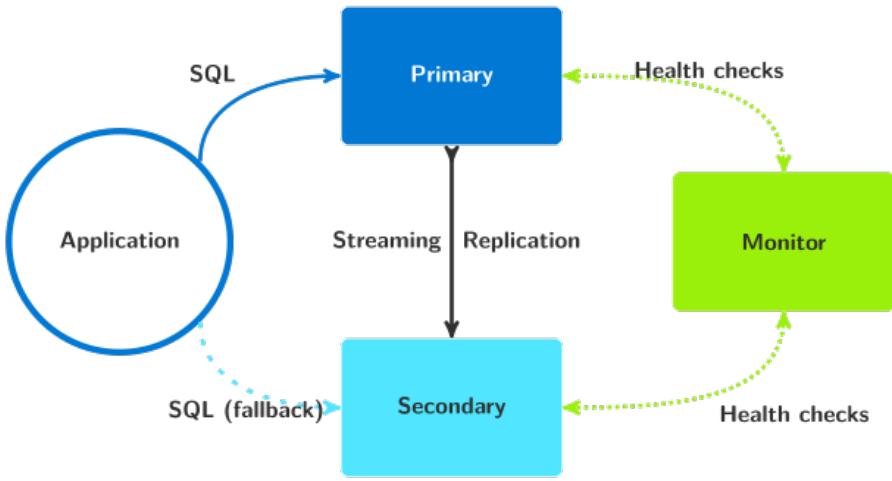


Abbildung 3.14: pg_auto_failover-Architektur - Single Standby

Aber wie die Grafik zeigt, können auch Multi-Nodes können eingebunden werden[41]:

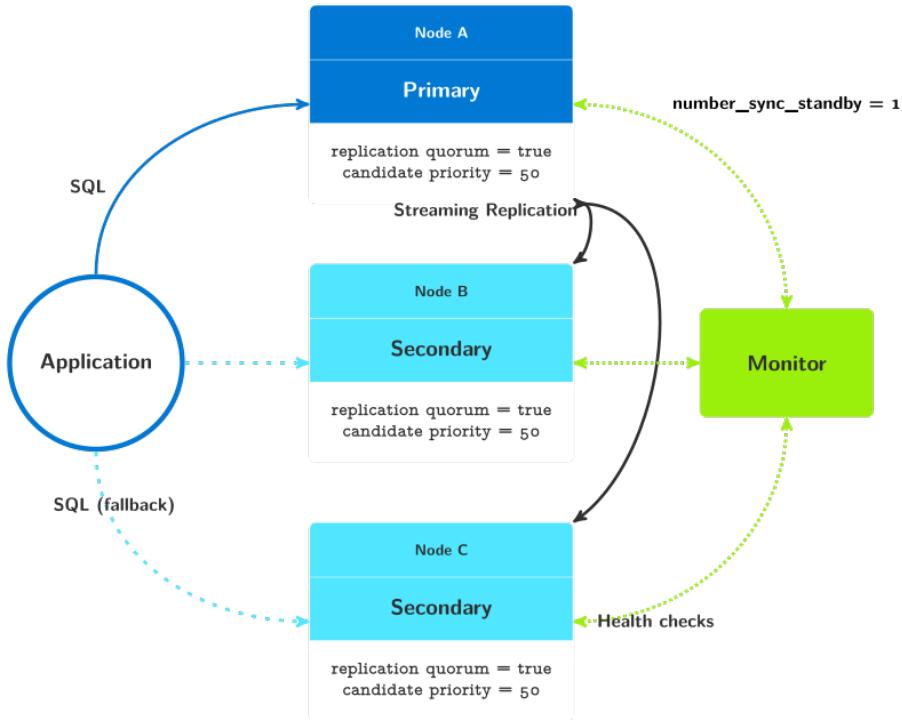


Abbildung 3.15: pg_auto_failover-Architektur - Multi-Node Standby

pg_auto_failover kann Citus einbinden. Allerdings bleibt die Architektur im Kern immer monolithisch.

Die nachfolgende Grafik zeigt die Architektur mit Citus[14]:

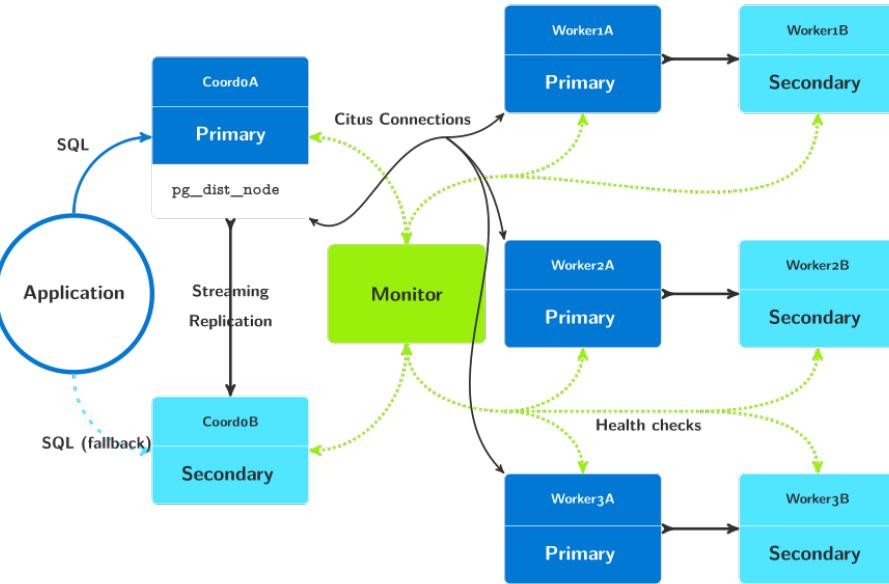


Abbildung 3.16: pg_auto_failover-Architektur - Citus

3.1.5.6 Synergien und Mehrwert

pg_auto_failover bietet eine Docker-Compose-Integration.
Allerdings ist keine Kubernetes-Integration dokumentiert.

Damit bietet pg_auto_failover keine Möglichkeit
Synergien zwischen monolithischer Architektur und einer Cloud-Native-Umsetzung auf
Kubernetes.
Entsprechend ist kein Mehrwert vorhanden.

3.1.5.6 CloudNativePG

CloudNativePG ist eine Containerlösung für PostgreSQL auf Kubernetes.
CloudNativePG wurde ursprünglich von EDB entwickelt.

3.1.5.6.1 Core-Features

Die wichtigsten Features von CloudNativePG sind[16]:

- k8s API Integration
- Automatischer Failover
- Self-Healing von Nodes resp. Replikas
- Skalierbarkeit (vertikal, horizontal bedingt)

- Volumne Backup
- Object Backup
- Rolling PostgreSQL Upgrade / Updates
- Pooling mit PgBouncer
- Offline und Online Import von bestehenden PostgreSQL DBs

3.1.5.6.2 Replikation

CloudNativePG bietet die üblichen PostgreSQL-Replikationen an.

3.1.5.6.3 Proxy

CloudNativePG benötigt keinen zusätzlichen Proxy.

3.1.5.6.4 Pooling

CloudNativePG unterstützt pgBouncer als Connection Pooler.

3.1.5.6.5 API / Skripte

CloudNativePG bietet eine API zum Monitoren und Verwalten von Backups, Clustern und dem System selbst[6].

3.1.5.6.6 Architektur

Kubernetes regelt die Zugriffe mit Hilfe eines entsprechenden Services in die Nodes auf den Pods:

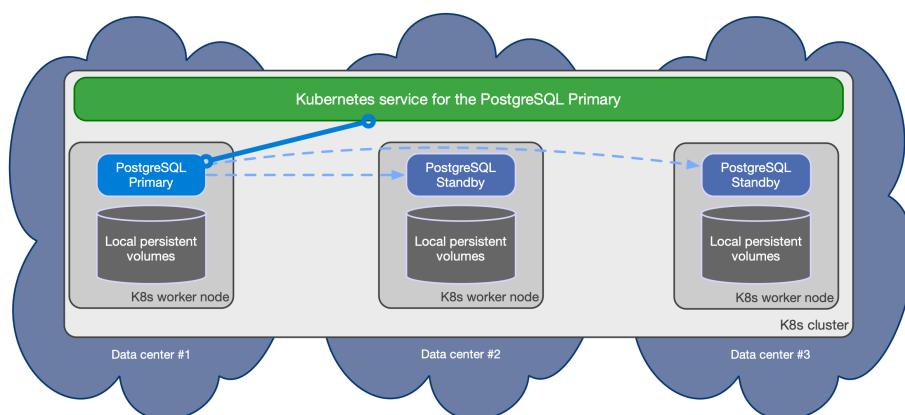


Abbildung 3.17: CloudNativePG - Kubernetes - PostgreSQL

Dabei werden die **Read-write workloads** an den primären Node gesendet:

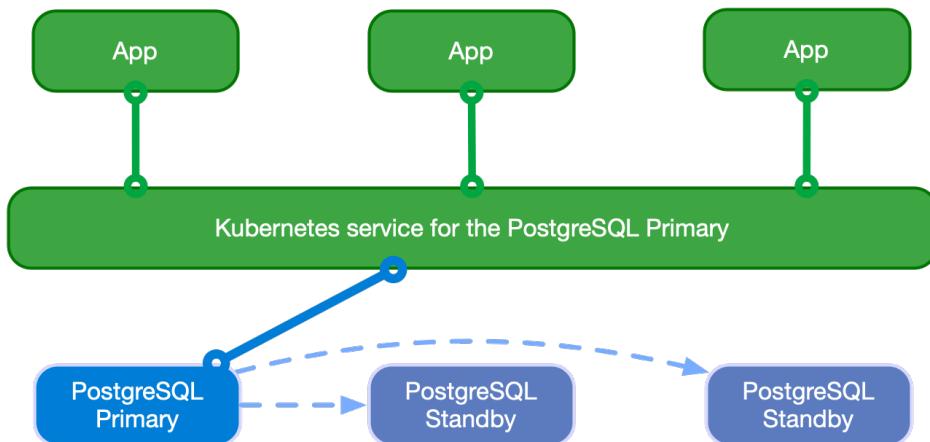


Abbildung 3.18: CloudNativePG - Kubernetes - Read-write workloads

Read-only workloads werden **mit** Round robin an die Replikas zugewiesen:

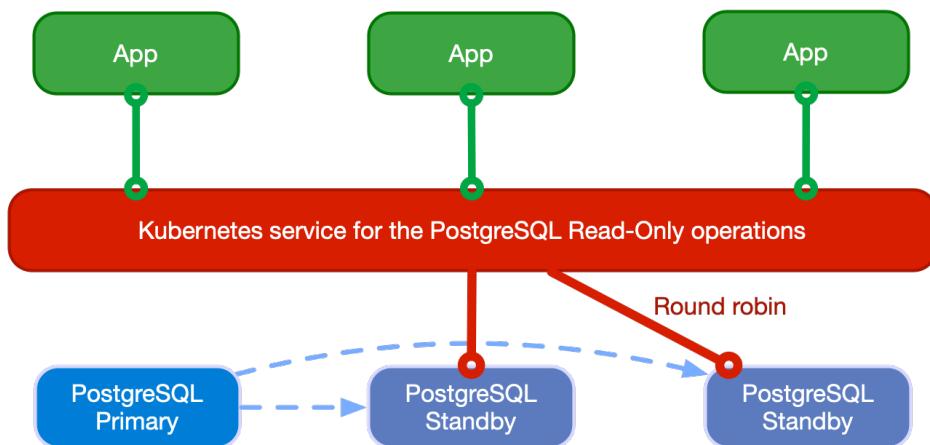


Abbildung 3.19: CloudNativePG - Kubernetes - Read-only workloads

Es könnten auch Lösungen mit Designated Kubernetes-Clustern in einem anderen RZ oder einer anderen Geo-Region realisiert werden.

3.1.5.6.7 Maintenance

Das Projekt hat viele offene Issues.

Der Code wird aber gut gepflegt, nicht mehr verwendeter Code wird auch regelmäßig entfernt. Hin und wieder müssen grössere Aufräumaktionen gestartet, zudem finden die Commits immer wieder am Monatsende statt.

Dafür erfüllt das Projekt die meisten Community Standards.

Die Vollständige Dokumentation ist im [Anhang - Maintenance](#) zu finden.

3.1.5.6.8 Synergien und Mehrwert

CloudNativePG bleibt ein monolithisches System,
welches aber keine Möglichkeit bietet,
auch auf einem normalen Serversetting betrieben zu werden.

Daher bietet CloudNativePG weder einen Benefit durch seine Architektur noch mit der
Möglichkeit,
Synergien nutzen zu können.

3.1.5.7 Patroni

Patroni ist eine von Zalando auf Basis von Python entwickelte HA-Lösung für PostgreSQL.
Patroni wird aktiv von Zalando gepflegt.

3.1.5.7.1 Core-Features

Patroni bietet folgende Core-Features:

- Rest-API und eigenes Skript- und Toolset
- Aktionen und Konfigurationen im Konsensprinzip abgestimmt
- Manueller oder Scheduled Switchover
- Reines PostgreSQL als Basis, Patroni setzt Hilfe von Python darauf auf
- Automatische reintegration von Nodes nach einem Fehler
- Citus kompatibel
- Docker und Docker-compose Dokumentation

3.1.5.7.2 Replikation

Patroni bietet per Default eine eigene Replikation an.
Diese ist allerdings eine asynchrone Replikation.

Patroni unterstützt aber die synchrone Replikation von PostgreSQL.

3.1.5.7.3 Proxy

Patroni benötigt einen HAProxy, um Load Balancing betreiben zu können[30].

3.1.5.7.4 Pooling

Patroni benötigt einen externen Connection Pooler.

Hier wird oft PgBouncer [43] verwendet.

3.1.5.7.5 API / Skripte

Patroni hat ein eigenes Tool- und Commandset, `patronictl`, welches die Verwaltung vereinfacht.

Es umfasst das Ändern und Erfassen von Konfigurationen, das Forcieren eines Failovers als Switchover, Maintenance Handling und Informationsbeschaffung.

Zusätzlich bietet Patroni eine API, welche Daten für das Monitoring bereitstellt, aber auch Betriebsfunktionen zur Verfügung stellt.

3.1.5.7.6 etcd

Patroni benötigt etcd oder Consul als Key-Value-Store.

3.1.5.7.7 Architektur

Das Architektur-Schaubild sieht folgendermassen aus:

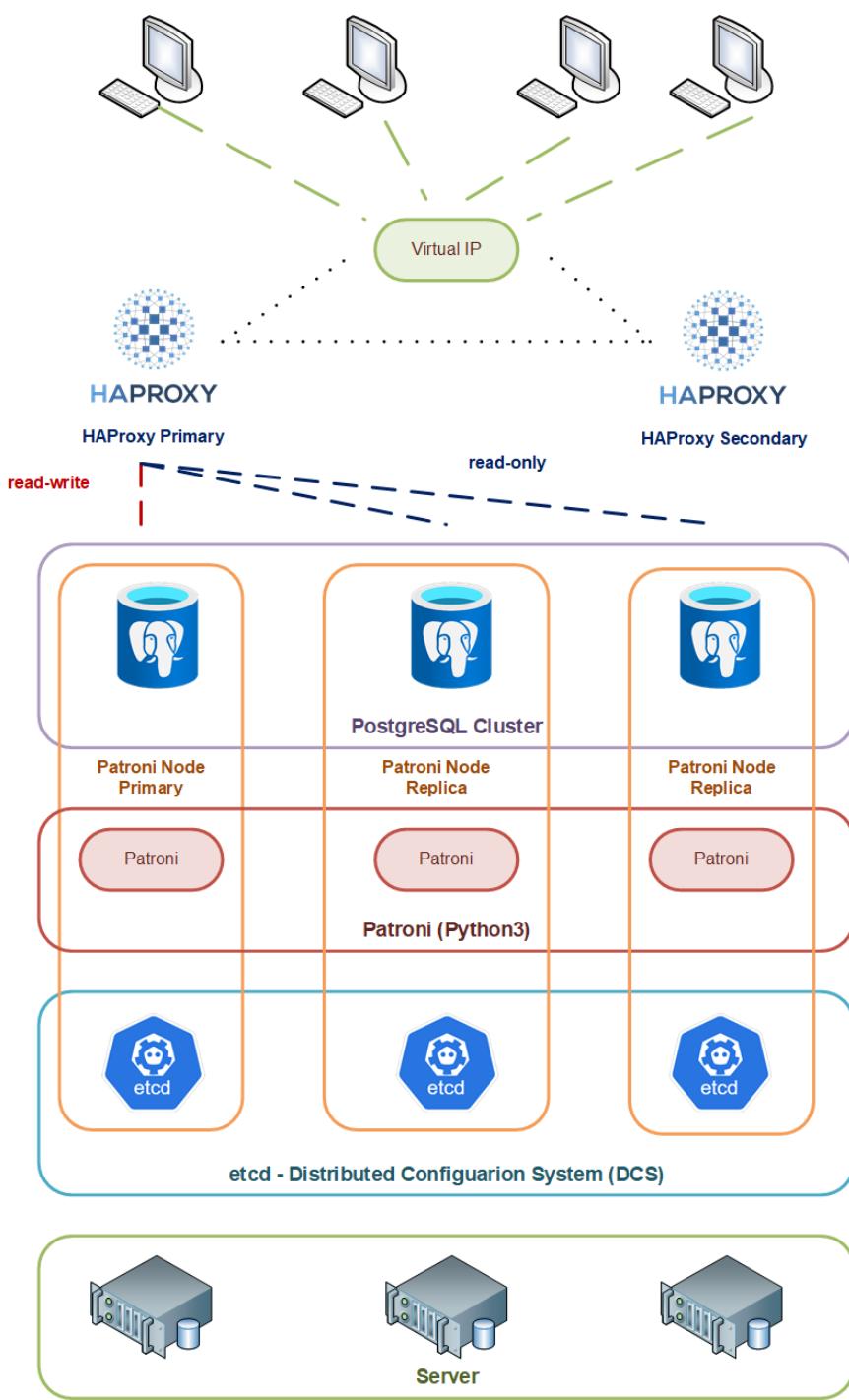


Abbildung 3.20: Patroni-Architektur

3.1.5.7.8 Maintenance

Patroni ist ein sehr gepflegtes Projekt, welches die gängigen Community Standards einhält. Die Details sind im [Anhang - Maintenance](#) zu finden.

3.1.5.7.9 Synergien und Mehrwert

Patroni kann nicht nur mit Citus zu einem Distributed / Sharded SQL System umgebaut werden, es ist auch Kern von StackGres.

Damit könnten die API und Skripte in beiden Welten verwendet werden.

Der Aufwand für die Verwaltung und Optimierung würde stark gesenkt.

Projekte wie vitabaks / postgresql_cluster[105] bieten zudem die Vorlage für eine noch stärkere Automatisierung.

3.1.5.8 StackGres mit Citus

StackGres ist eine PostgreSQL-Implementation die dafür vorgesehenen ist, in einem Kubernetes Cluster betrieben zu werden.

An sich wäre StackGres nur eine Implementation von Patroni in Kubernetes inkl. Load Balancer. Nun kommt das Citus-Plugin ins Spiel, welches aus einer jeden monolithischen, klassischen PostgreSQL Cluster eine Distributed SQL Umgebung macht.

Citus Data, der Entwickler von Citus, wiederum ist in den Microsoft-Konzern eingebettet.

3.1.5.8.1 Core-Features

Die wichtigsten Features von StackGres sind[29]:

- k8s Integration
- Deklaratives k8s CRD
- Automatische Backups
- Grafana und Prometheus integration
- Management Web Konsole
- Erweiterte Replikationsmöglichkeiten
- Integriertes Pooling
- Integrierter Proxy

3.1.5.8.2 Replikation

StackGres bietet asynchrone und synchrone Replikation, Gruppenreplikation sowie kaskadierende Replikation an.

Citus bietet sein eigenes Modell mit dem Sharding an.

3.1.5.8.3 Proxy

StackGres hat den Proxy bereits mit envoy[27] implementiert.

3.1.5.8.4 Pooling

PgBounder[43] ist bereits integriert, es braucht also keinen weiteren Connection Pooler.

3.1.5.8.5 API / Skripte

StackGres wird primär über YAML-Files und Kubernetes gesteuert, bietet aber eine eigene API an.

Citus bietet ebenfalls eine eigene API, mit der Citus vollständig verwaltet werden kann.

3.1.5.8.6 Architektur

3.1.5.8.6.1 StackGres

StackGres packt PostgreSQL, Patroni, PgBouncer und envoy in einen Kubernetes Pod:



Abbildung 3.21: Stackgres - Grundarchitektur

3.1.5.8.6.2 Citus Coordinator und Workers

Citus arbeitet mit einem Coordinator-Node, der jedes Query analysiert und an einen Worker-Node weitergibt.

Die unten stehende Grafik zeigt diese Architektur auf:

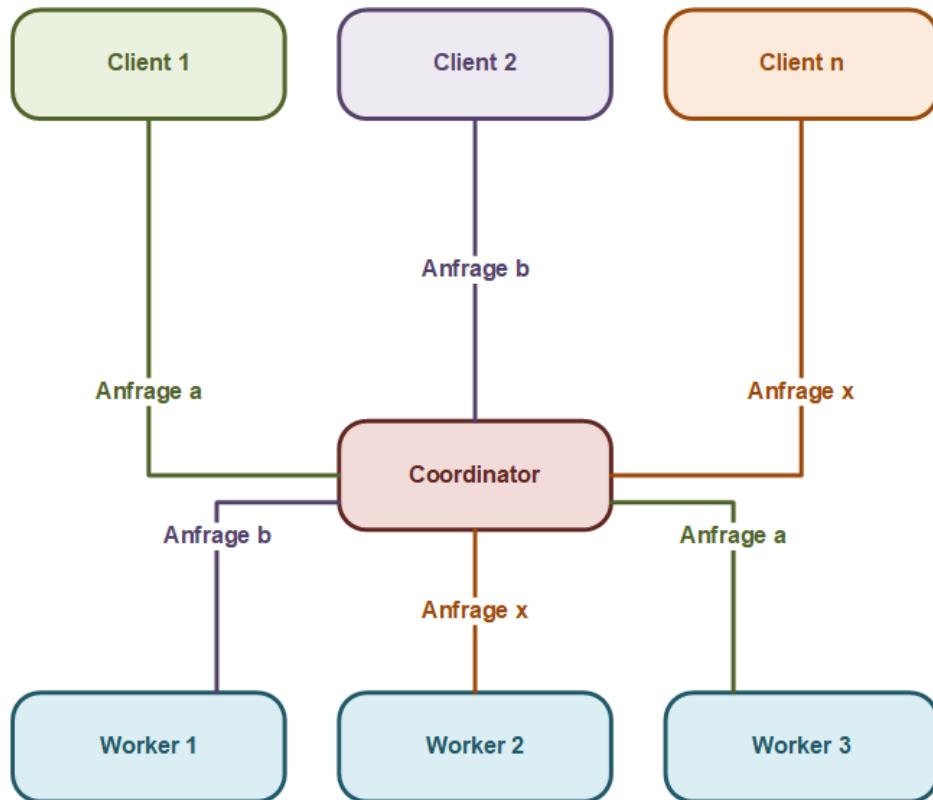


Abbildung 3.22: Citus - Coordinator und Workers

3.1.5.8.6.3 Citus Sharding

Citus bietet zwei Sharding-Modelle an.

Row-based sharding

Beim diesen Sharding werden Tabellen anhand einer Distribution Column aufgeteilt[19, 11]:

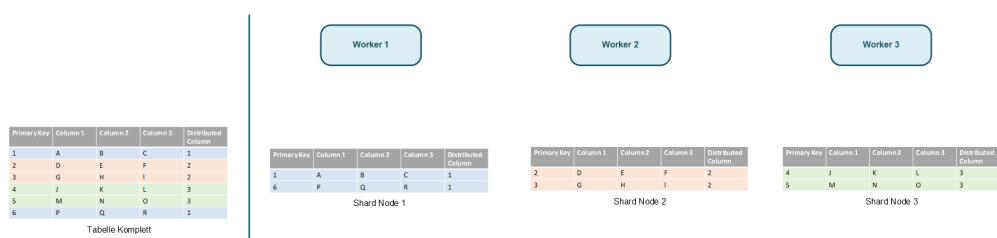


Abbildung 3.23: Citus - Row-Based-Sharding

Schema-based sharding

Beim Schema Based Sharding werden die Tabellen horizontal partitioniert oder gleich ganze Tabellen in den Shard gepackt.

Die Schards werden dabei nach Schema aufgeteilt, wobei ein Schema, wobei ein Schema auf einem Node verbleibt.

Dadurch braucht es im Prinzip keine Anpassung an den Tabellen und Applikationen.

Das Prinzip wird unten aufgezeigt:

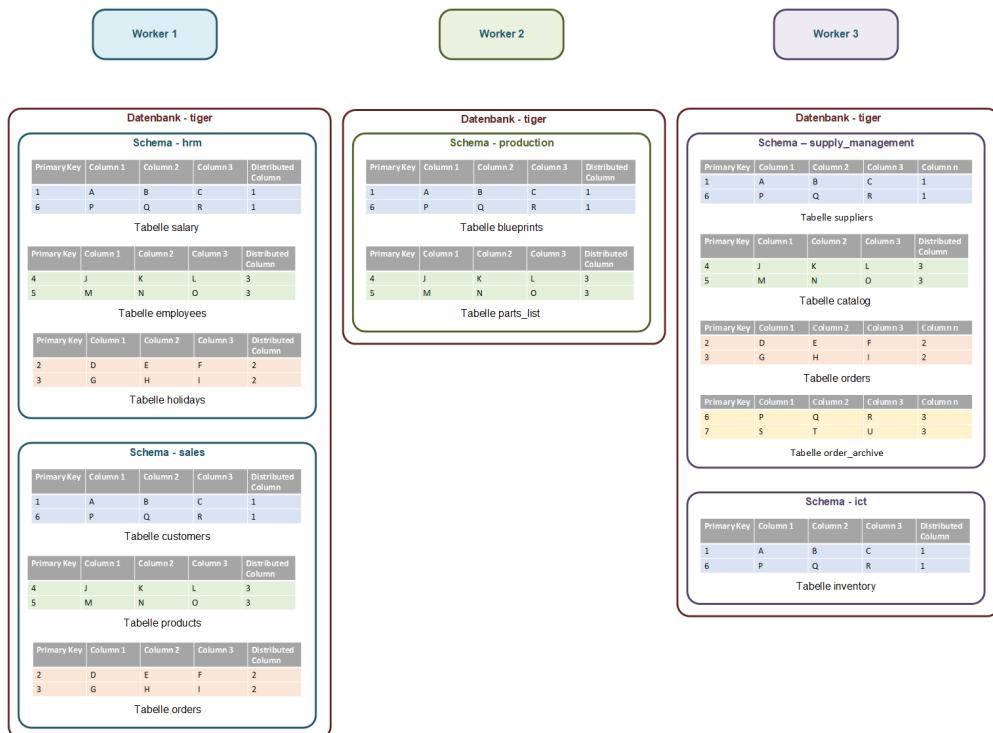


Abbildung 3.24: Citus - Schema-Based-Sharding

Schlussfolgerung

Beide Sharding-Methoden haben eine grosse Schwäche.

In Version 7.2 konnte noch ein Replikationsfaktor angegeben werden[18], ab Version 11 wurde auch diese Variante gestrichen und man konnte noch eine 1:1 Replikation auf einen Worker fahren[13].

Spätestens mit Version 12 steht auch dies nicht mehr zur Verfügung, man muss eine Replication auf alle Kubernetes Nodes erzeugen.

Sie sind nicht vollständig ACID-Konform da Datenverlust entstehen kann, wenn ein Node wegfällt. Dies muss aber bei der Evaluation mit Hilfe von Tests noch bestätigt werden.

Die Shards müssen aber stand heute mit entsprechenden mit Replikation gesichert werden[17]. Daraus ergibt sich aber ein nicht zu vernachlässigbarer Mehraufwand, wenn man self-healing Nodes implementieren möchte.

Jeder Node ist für sich genommen, eine eigene Zone, um sicherzustellen, dass es zu keinem Datenverlust kommt,
 müsste jeder Shared-Node in eine der jeweiligen Zonen repliziert werden.
 Das heisst, es müssten $Shard - Nodes^2$ Replika-Nodes erstellt werden, hier ein Schematisches-Beispiel mit drei Shard-Nodes:

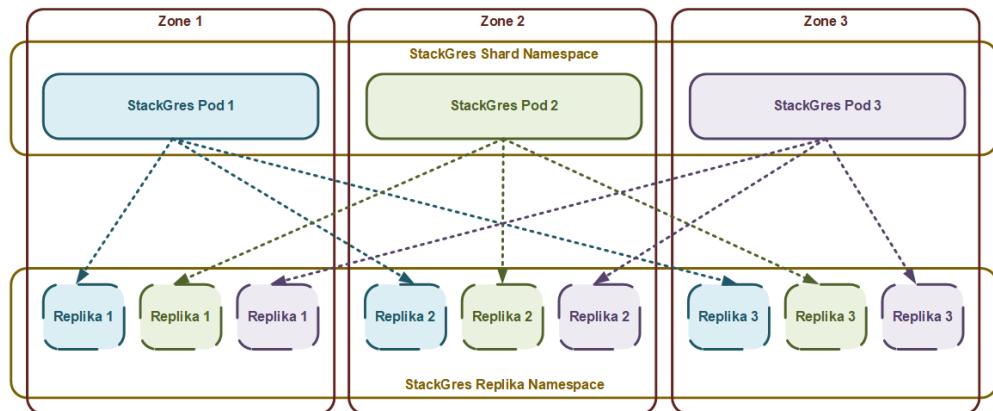


Abbildung 3.25: StackGres-Citus - Shard-Replikation

Die Automation und nur schon die Konfiguration für das Mitskalieren, dürfte einiges an Zeit in Anspruch nehmen.

Eine nicht unwesentliche Folge wäre ein starker Rückgang des Throughput's und Performance-Einbussen.

Alternativ kann ein klassischer Replika-Server verwendet werden, wo die ganze Datenbank gesichert wird.

Bis alle Daten wieder in den StackGres-Citus-Cluster zurückgeholt wurden, das Re-Balancing abgeschlossen ist usw.,

muss der ganze Cluster für die User unerreichbar sein, da dieser in dieser Zeit nicht mehr konsistent ist.

Dieser zweite Ansatz bietet zwar Vorteile beim Throughput, doch im Fehlerfall ist ein HA-Betrieb nur noch begrenzt garantiert werden.

3.1.5.8.7 Maintenance

Bei StackGres und Citus ist Citusdata, welches mittlerweile zu Microsoft gehört, sehr aktiv. Citusdata hält nicht nur die Community Standards ein, sondern Committed zyklisch und räumt alten Code regelmässig auf.

Anders sieht es bei Ongres, dem Maintainer von StackGres, aus.

Weder werden besonders viele Community Standards eingehalten, noch wird regelmässig Committed.

Die Aktivitäten gingen in den letzten Jahren und Monaten zurück, was dazu führt, das immer wieder mal grössere Commits notwendig werden.

Die genaue Analyse ist im [Anhang - Maintenance](#) zu finden.

3.1.5.9 YugabyteDB - Distributed SQL 101

YugabyteDB - Distributed SQL 101 ist eine nahezu komplett PostgreSQL kompatible Datenbank. Sie ist eine Distributed SQL Datenbank, also eine verteilte Datenbank[102].

3.1.5.9.1 Core-Features

Die wichtigsten Features von YugabyteDB sind[10]:

- PostgreSQL kompatibel
- Cassandra-Kompatibilität
- Horizontale Skalierbarkeit
- Global verteilbar
- Cloud Native

3.1.5.9.2 Replikation

Die Replikation erfolgt auf Sharding-Ebene mithilfe des Raft-Protokolls.

Raft wird im [Anhang - Raft-Consensus](#) beschrieben.

3.1.5.9.3 Proxy

YugabyteDB benötigt einen Kubernetes Proxy, zum Beispiel MetalLB.

3.1.5.9.4 Pooling

YugabyteDB hat einen Connection Pooler mit dem YSQL Connection Manager integriert[71].

3.1.5.9.5 API / Skripte

YugabyteDB bietet eigene APIs[7] und CLIs[15] für das Verwalten an.

Diese können auch abgesichert zu werden.

3.1.5.9.6 Architektur

YugabyteDB ist kein reines RDBMS, resp. gar keines. Die Basis besteht aus einem Key-Value-Store. Darüber wurde eine Cassandra-like Query API und eine PostgreSQL like SQL API aufgebaut:

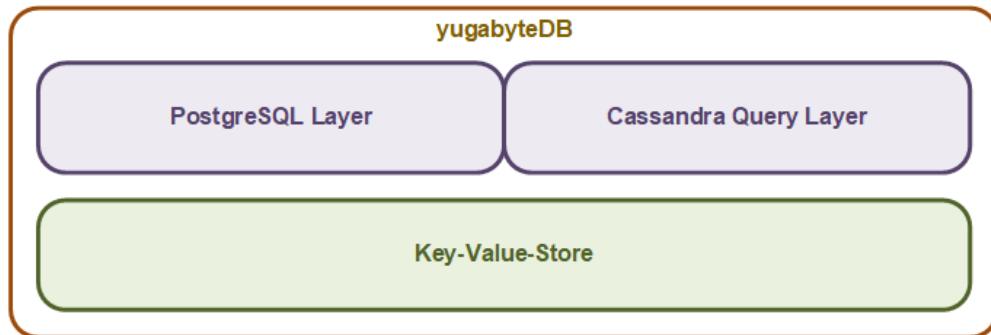


Abbildung 3.26: YugabyteDB - Grundkonzept

Der Basisaufbau wiederum beinhaltet diverse Dienste für das Sharding, die Replikation und Transaktionen:

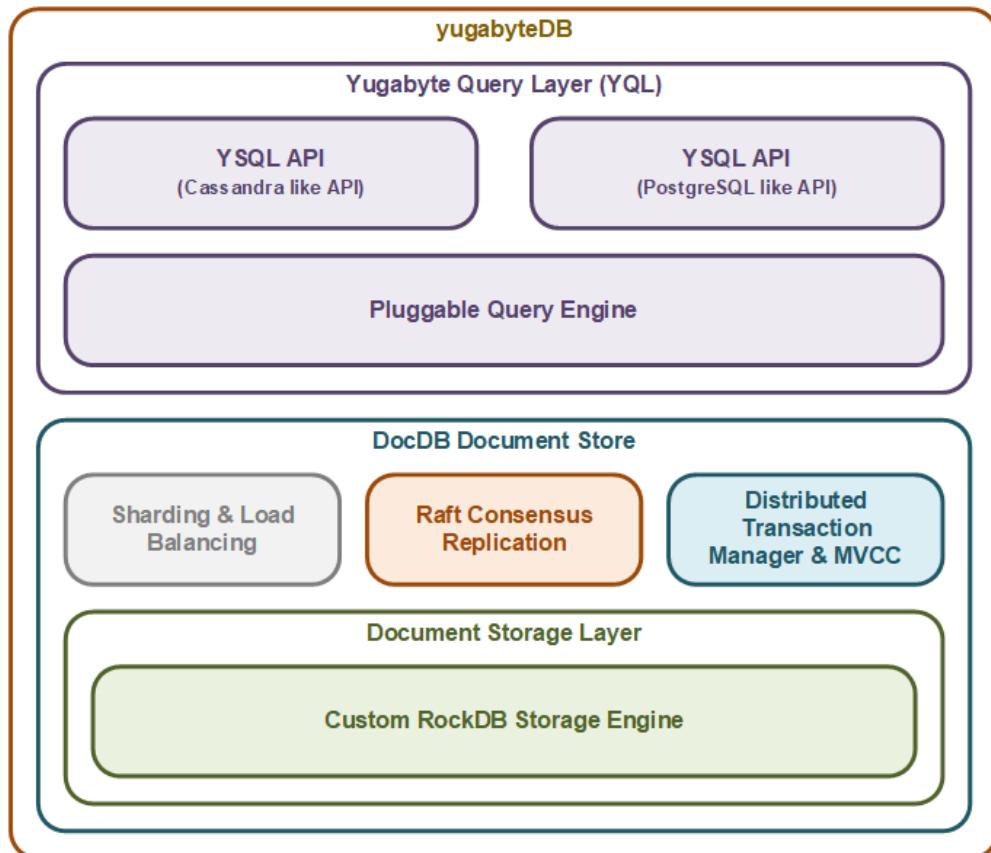


Abbildung 3.27: YugabyteDB - Architektur

3.1.5.9.6.1 YugabyteDB - Sharding

YugabyteDB teilt seine Tabellen in Tablets auf. Die Aufteilung kann gemäss Sharding-Standards gemacht werden:

The diagram illustrates the sharding of a table into three tablets. On the left, a 'Tabelle Komplett' (Complete Table) is shown with 6 rows and 5 columns: Primary Key, Column 1, Column 2, Column 3, and Distributed Column. The data is as follows:

Primary Key	Column 1	Column 2	Column 3	Distributed Column
1	A	B	C	1
2	D	E	F	2
3	G	H	I	2
4	J	K	L	3
5	M	N	O	3
6	P	Q	R	1

Below the table is the label 'Tabelle Komplett'.

On the right, the table is divided into three tablets:

- Tablet 1**: Contains rows 1 and 6. The 'Distributed Column' value is 1.

Primary Key	Column 1	Column 2	Column 3	Distributed Column
1	A	B	C	1
6	P	Q	R	1

- Tablet 2**: Contains rows 2 and 3. The 'Distributed Column' value is 2.

Primary Key	Column 1	Column 2	Column 3	Distributed Column
2	D	E	F	2
3	G	H	I	2

- Tablet 3**: Contains rows 4 and 5. The 'Distributed Column' value is 3.

Primary Key	Column 1	Column 2	Column 3	Distributed Column
4	J	K	L	3
5	M	N	O	3

Below each tablet is its respective label: 'Tablet 1', 'Tablet 2', and 'Tablet 3'.

Abbildung 3.28: YugabyteDB - Sharding

Dabei hat jedes Tablet auf einem Node einen Leader, der an die Follower auf den anderen Nodes repliziert:

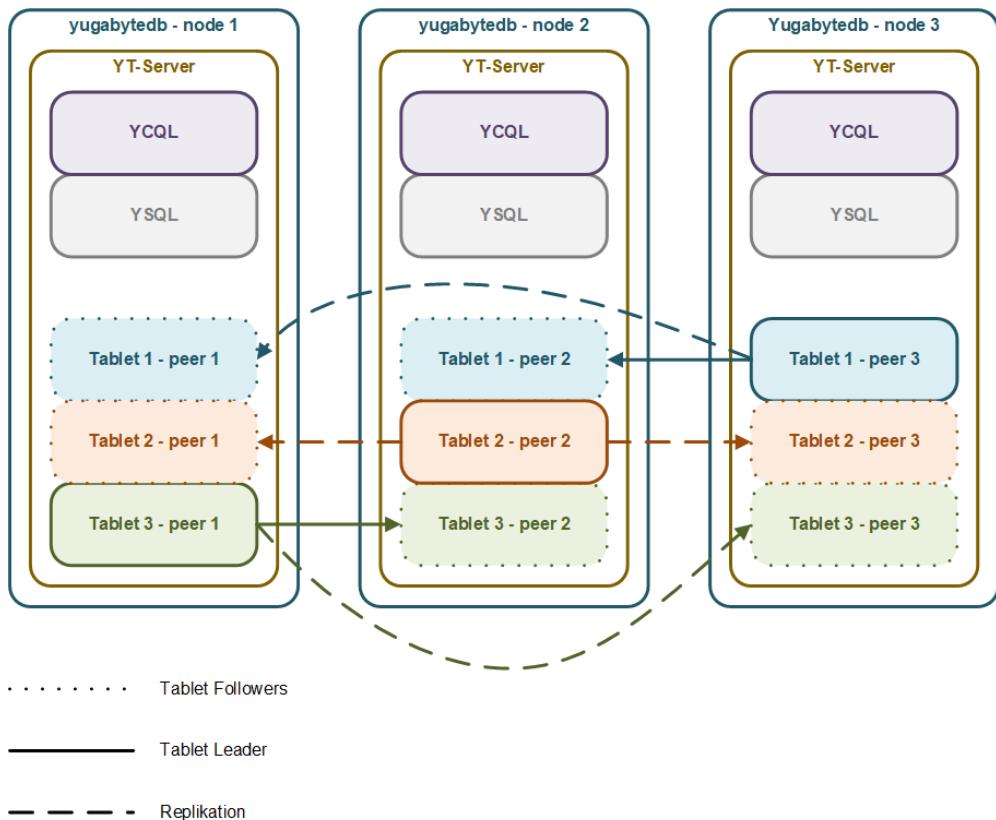


Abbildung 3.29: YugabyteDB - Tablet - Leader und Follower

Mit dem Replikationsfaktor kann angegeben werden, auf wie vielen Nodes ein Tablet repliziert werden soll.

Bei einem 4-Node System können z.B. einige Tablets einen Faktor 3 haben, dass heisst, dass die Daten nur auf 3 Nodes repliziert werden.

Bei einem Replikationsfaktor 4 werden die Daten auf alle Nodes repliziert.

Dies wird mit einem eigenen Service, dem YB-TServer service [49] geregelt:

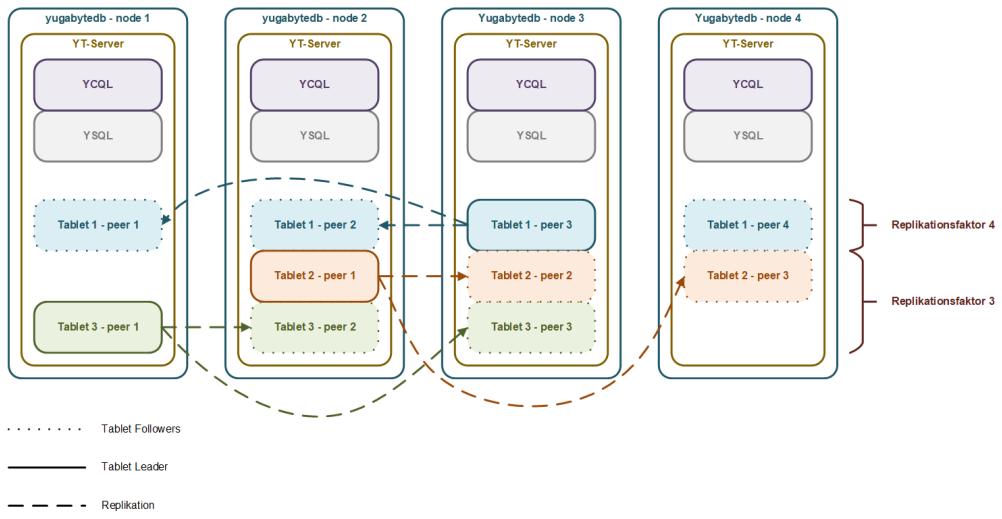


Abbildung 3.30: YugabyteDB - Tablet - Replikationsfaktor

Mehrere Nodes können zu Zonen zusammengebunden werden, die dann z. B. auf verschiedene Rechenzentren verteilt werden:

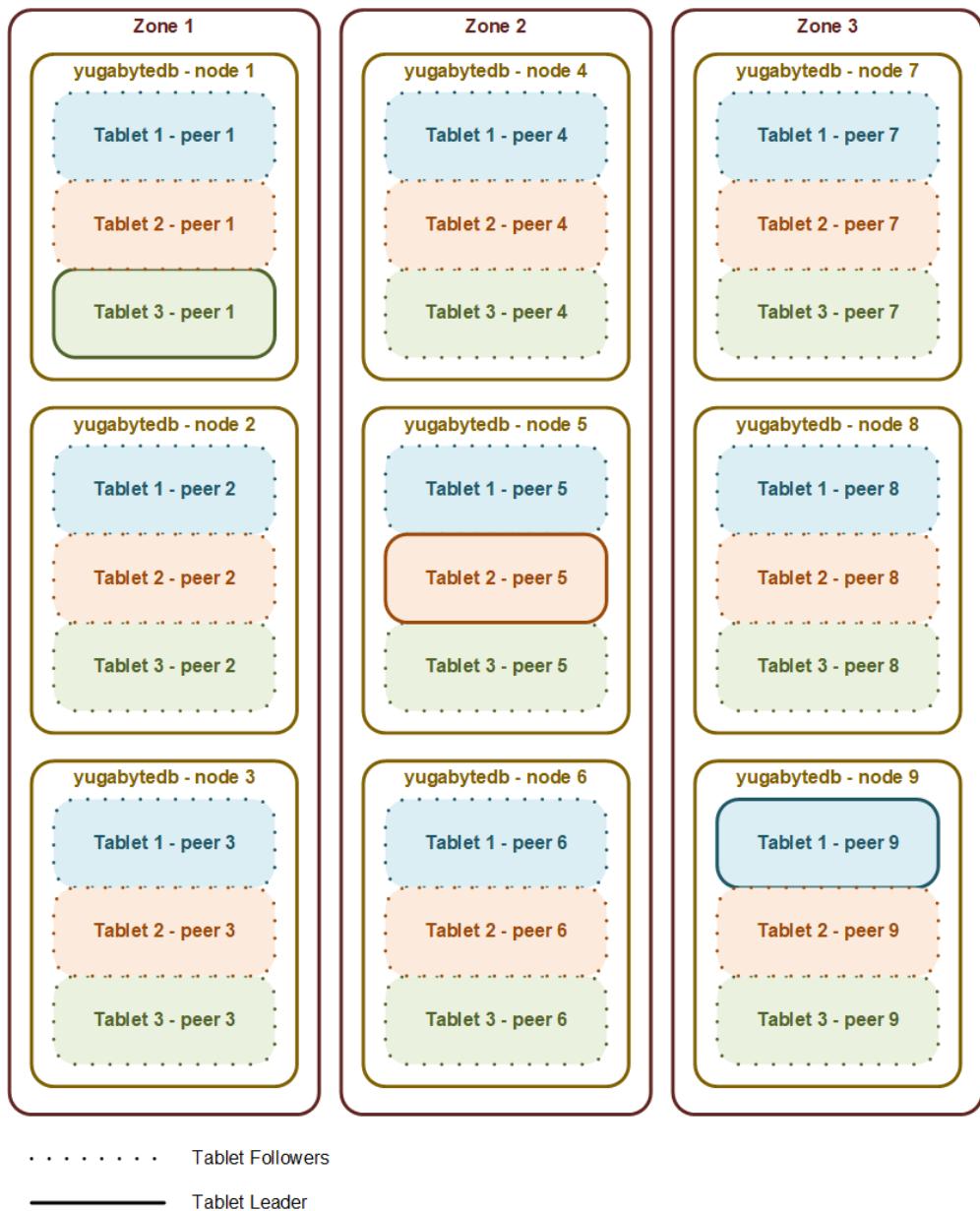


Abbildung 3.31: YugabyteDB - Zonen

Dies wird dann sinnvoll, wenn eine gewisse Ausfalltoleranz erreicht werden soll. Fällt ein Tablet Peer oder ein Node in einer Zone aus, so wird die ganze Zone sofort als nicht mehr arbeitsfähig angesehen. Entsprechend werden in allen Nodes die Tablet-Leader stillgelegt und auf die übrigen Zonen verteilt. YuganyteDB nennt dies Zone outage Tolerance[46].

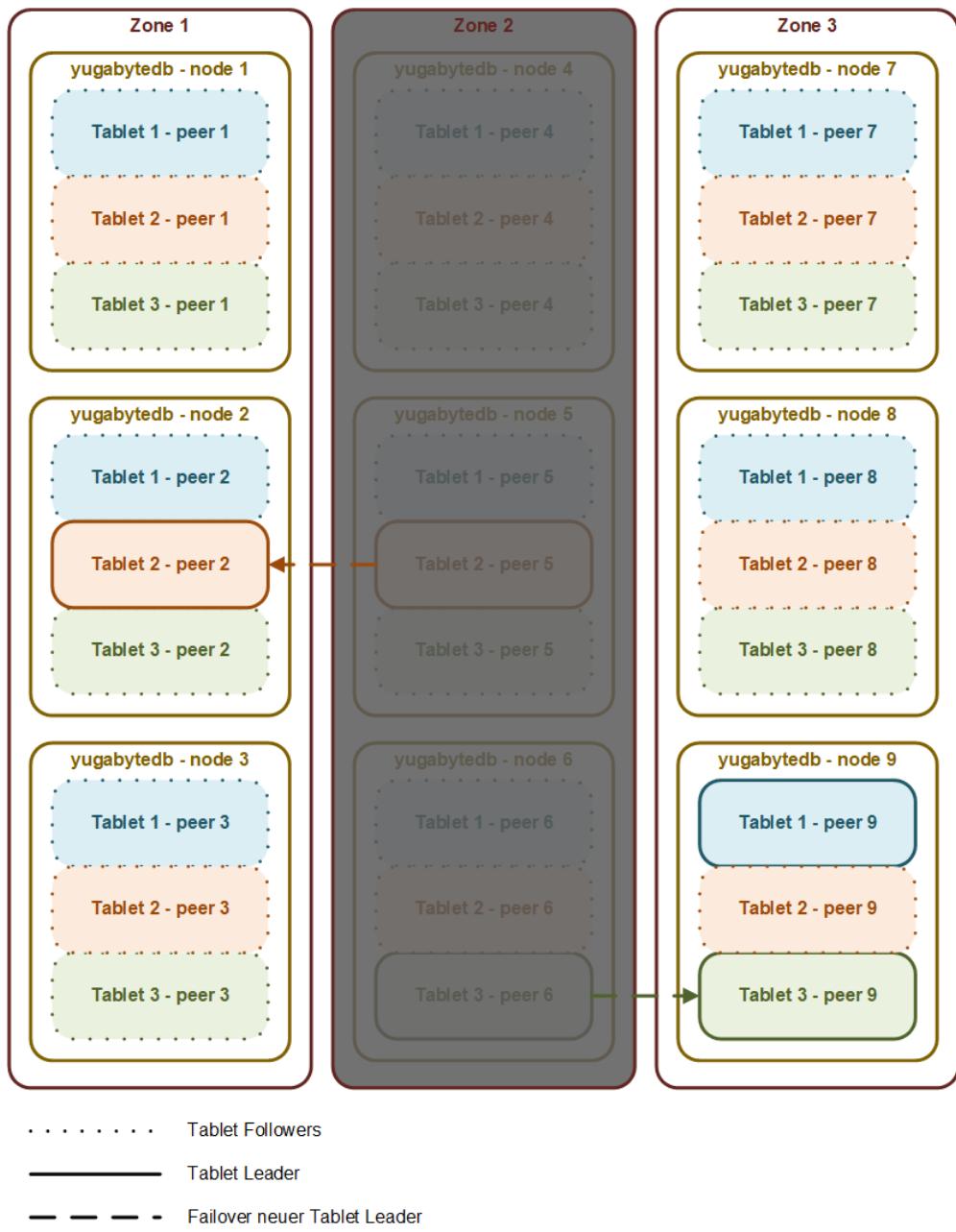


Abbildung 3.32: YugabyteDB - Zone outage Tolerance

3.1.5.9.7 Maintenance

YugabyteDB hat viele gemeldete Issues.

Zudem hält das Projekt nur die notwendigsten Community Standards ein.

Dafür werden in vielen kurzen Sprints viele Commits durchgeführt, obwohl Yugabyte der einzige Maintainer ist.

Die genaue Analyse ist im [Anhang - Maintenance](#) zu finden.

3.1.5.9.8 Synergien und Mehrwert

Der grosse Benefit von YugabyteDB ist sein Distributed SQL Ansatz.

Zudem bietet YugabyteDB eine vollständige Cassandra Integration.

Der Benefit ist auf jeden Fall gegeben.

3.1.6 Vorauswahl

Folgende Lösungen werden nicht evaluiert, sondern bereits zu Beginn ausgeschieden:

Nr.	Lösung	Status	Begründung
1	KSGR-Lösung	Vorausgeschieden	Hat nur einen Standy / Replika-Node. Failover Funktioniert nur bei kleineren Datenmengen wirklich in einer vernünftigen Zeit.
2	pgpool-II	Vorausgeschieden	pgpool-II hat kein GitHub-Repository und bietet daher keine vergleichswerte mittels Github Insights.
3	pg_auto_failover	Vorausgeschieden	pg_auto_failover würde zwar Citus-Support bieten, allerdings gibt es keine gut dokumentierte Implementation für Kubernetes. Erfüllt daher das Kriterium für die Synergien nicht
4	CloudNativePG	Vorausgeschieden	CloudNativePG ist keine vollständige Cloud Native Lösung. Mittels Citus könnte sogar eine Distributed SQL Lösung implementiert werden. Die Grundarchitektur bleibt aber Monolithisch mit einem Primary und Repikas. Und da kein Benefit in Form von Synergien vorhanden sind, fällt CloudNativePG raus.
8	Citus row-based-sharding	Vorausgeschieden	Citus row-based-sharding wäre Hocoeffizient wenn es um Ressourcenverteilung geht und zudem echtes Sharding. Allerdings setzt es anpassungen an den Tabellen der Applikationen voraus. Das KSGR ist allerdings kein Softwarehaus und kann keine Forks durchführen, auch weil viele Applikationen zertifiziert sein müssen. Scheitert daher an der Machbarkeit

Tabelle 3.9: Vorauswahl - Ausgeschieden

Entsprechend werden nur noch nachfolgende Lösungen genauer betrachtet:

Nr.	Lösung	Status	Begründung
5	Patroni	Evaluation	Patroni kann als Monolithisches System genutzt werden, ist aber auch Kern von Stackgres. Die API und Skripte können also in beiden Welten verwendet werden
6	Stackgres mit Citus	Evaluation	Bietet eine einfache und kompakte Möglichkeit für ein Distributed SQL System. Da Patroni unter der Haube ist, kann die API und sonstige Skripte auch auf einem Monolithischen System eingesetzt werden.
7	Yugabyte-DB	Evaluation	Ist eine reine Distributed SQL Lösung und ist Vollständig Cloud Native.

Tabelle 3.10: Vorauswahl - Evaluation

3.1.7 Installation verschiedener Lösungen

Entsprechend wurden folgende Server bereitgestellt:

Server	Typ	Funktion	Full Qualified Device Name	IP
sks1183	Distributed SQL	Server	sks1183.ksgr.ch	10.0.20.97
sks1184	Distributed SQL	Agent	sks1184.ksgr.ch	10.0.20.104
sks1185	Distributed SQL	Agent	sks1185.ksgr.ch	10.0.20.105
sks1232	Monolith	Server	sks1232.ksgr.ch	10.0.20.110
sks1233	Monolith	Server	sks1233.ksgr.ch	10.0.20.111
sks1234	Monolith	Server	sks1234.ksgr.ch	10.0.20.112
sks9016	Benchmark Server	Client	sks9016.ksgr.ch	10.0.21.216
vks0032	Distributed SQL	Virteulle IP	vks0032.ksgr.ch	10.0.20.106
vks0040	Monolith	Virteulle IP	vks0040.ksgr.ch	10.0.20.113

Tabelle 3.11: Evaluationssyssteme

3.1.7.1 rke2 - Evaluationsplattform

Die grundsätzliche Evaluationsplattform für Distributed SQL / Shards sieht folgendermassen aus:

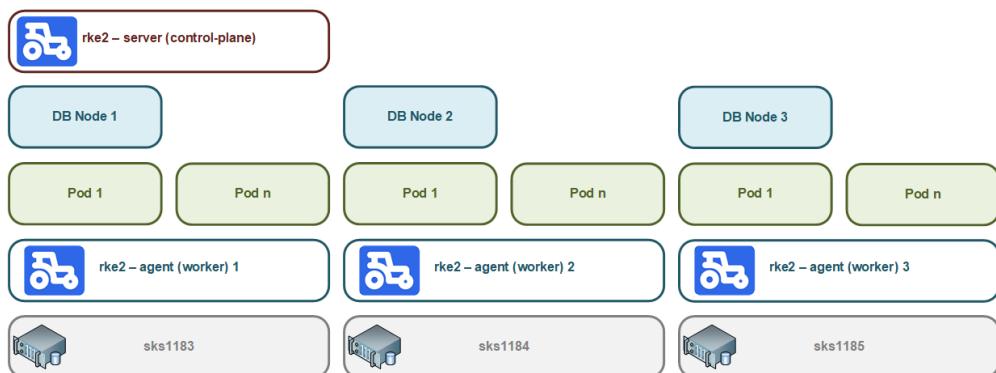


Abbildung 3.33: Evaluationssystem - Distributed SQL / Shards

Die Konfiguration der rke2-Nodes sieht folgendermassen aus:

Kubernetes Runtime	rke2
Container-Enviroment	containerd
Container Network Interface (CNI)	cilium
Cloud Native Storage (CNS)	local-path-provisioner
cluster-cidr	198.18.0.0/16
service-cidr	198.18.0.0/16
External IP Range	10.0.20.106,10.0.20.150-10.0.20.155

Tabelle 3.12: Evaluationssystem - Distributed SQL / Sharding

3.1.7.2 Patroni

3.1.7.2.1 Architektur

Ursprünglich sollte auf jedem Patroni Server (sks1232, sks1233 und sks1234) ein etcd-Node installiert werden.

Auch sollte auf sks1234 der HAProxy installiert werden.

Im Kapitel [Installation](#) wird erklärt, wieso die Architektur nun folgendermassen aussieht:

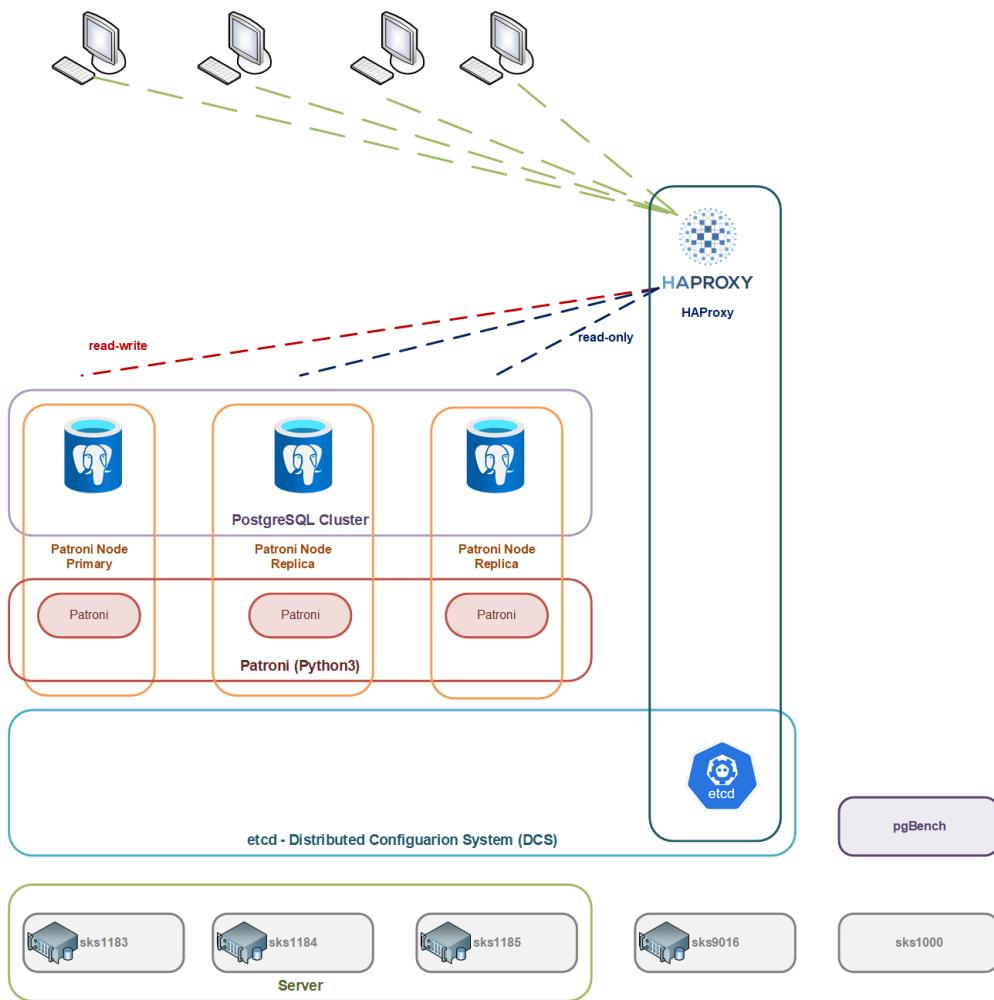


Abbildung 3.34: Patroni - Evaluationsarchitektur

Neu wird auf sks9016 ein etcd-Node und der HAProxy installiert.

3.1.7.2.2 Installation

Wie schon erwähnt, wurde versucht die etcd-Nodes auf die Patroni-Nodes zu installieren.

Erst kam es zu einem Fehler, dass keine Verbindung zum etcd-Node hergestellt werden konnte:

```
1 ERROR: Failed to get list of machines from http://10.0.20.110:2379/v2:  
      EtcdException('Bad response : 404 page not found\n')  
2 INFO: waiting on etcd
```

Listing 3.1: Patroni - etcd API V2 Error

Ursache war, dass etcd V3 ab Version v3.4 die API V2 nicht mehr per Default aktiv ist.

Man könnte bis v3.7 die API V2 noch aktivieren:

```
1 ETCDCCTL_API=2
```

Listing 3.2: Patroni - etcd API V2 Enable

Die nachhaltigere Lösung ist, im Konfigurations-yml-File des Patroni-Nodes Version 3 zu setzen (und dieses Package auch explizit zu installieren):

```
1 ...  
2 etcd3:  
3     host: <ip / hostname>:2379  
4 ...
```

Listing 3.3: Patroni - etcd3 Flag

Der Primary-Node konnte in jeden Fall installiert und deployt werden.

Sobald aber jeweils die Replika-Nodes gestartet wurden, kam es zu einem Fehler.

Die Ursache war, dass es ja bereits einen Host-Key für den jeweiligen Hostnamen resp. die entsprechende IP gab, nämlich den etcd-Node.

So kam es jeweils zu einem Key-Error.

Nach einigem Versuchen, etwa die Keys neu zu beschreiben, brach ich die übung ab.

Der etcd-Node wurde nun nur noch auf dem Server sks9060 installiert.

Resultat war, dass der Cluster lauffähig wurde.

Passwörter können mit dem Bootstrap mitgegeben werden.

Dazu muss im postgresql-Segment das Subsegment authentication erstellt werden.

Der Replikationsuser muss mit einem subsegment replication und der postgres-User mit superuser angegeben werden:

```
1 ...  
2 postgresql:  
3     ...  
4     authentication:  
5         replication:  
6             username: replicator  
7             password: <password>  
8         superuser:  
9             username: postgres  
10            password: <password>  
11 ...
```

Listing 3.4: Patroni - Passwörter

Zuerst lief der Cluster nur asynchron, auch die ersten beiden Benchmarks wurden so ausgeführt. Der Cluster lässt sich nämlich nicht synchron Bootstraben, die Konfiguration muss nachträglich gemacht werden.

Dazu kann ein JSON mit den geänderten Konfiguration übergeben werden, in dieser Konfiguration musste dabei das yml-File mit der Konfiguration angegeben werden:

```
1 patronictl -c /etc/patroni/config.yml edit-config --apply - --force <<'JSON'
2 {
3     synchronous_mode: "on",
4     synchronous_mode_strict: "on",
5     synchronous_node_count: 2,
6     "postgresql":
7         {
8             "parameters": {
9                 "synchronous_commit": "on",
10                "synchronous_standby_names": "*"
11            }
12        }
13    }
14 }JSON
```

Listing 3.5: Patroni - Synchrone Replikation setzen

Die Vergrösserung der Disks hat nur einen begrenzten Impact.

Lediglich das Verzeichnis pg_wal, welches die WAL-Files aufnimmt, muss angepasst werden.

Um die Umstellung ohne Reinstallation durchführen zu können, muss mit symlinks gearbeitet werden.

Die Daten können via Tablespace auf die grösseren mounts gesetzt werden.

Die vollständige Dokumentation der Evaluationsinstallation ist im [Anhang - Installation Patroni](#) zu finden.

3.1.7.3 StackGres - Citus

3.1.7.3.1 Architektur

Für das Benchmarking wurde ein minimales Setting ausgewählt.

Ein Coordinator und einen Shard-Node mit einem Leader- und Replika-Pod.

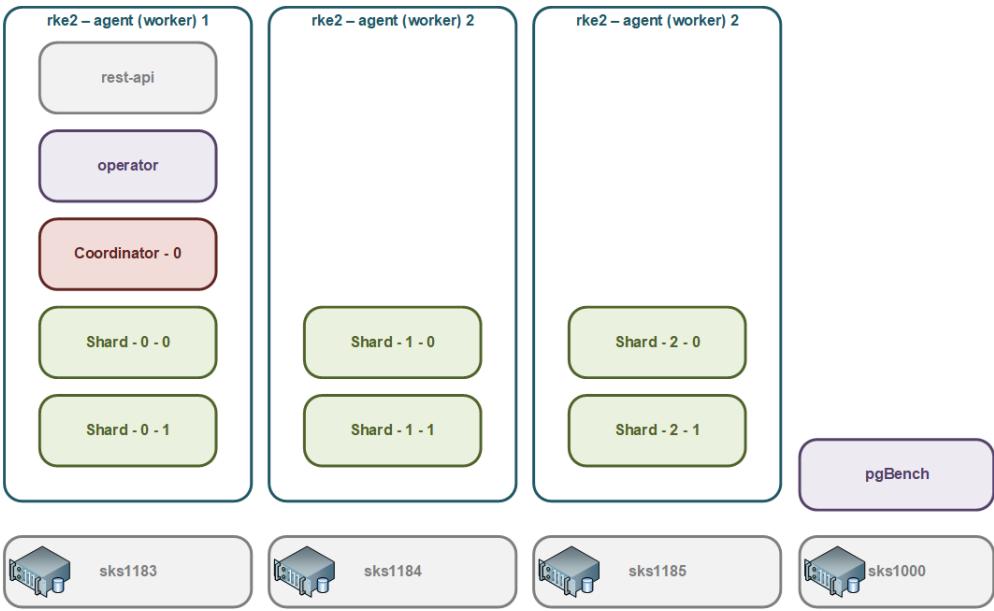


Abbildung 3.35: Stackgres - Citus - Evaluationsarchitektur Benchmarking

Für die Self Healing Tests wurde eine umfangreichere Architektur vorgenommen.

Es stellte sich heraus, dass man die beim [Citus Sharding](#) beschriebene Lösung zum Replizieren leicht umsetzen kann.

Wie dem Architekturnschaubild zu entnehmen ist,
wurden drei Coordinator-Nodes und drei Shard-Nodes gesetzt.
Die Shard-Nodes wurden dabei auf jeweils drei Nodes repliziert:

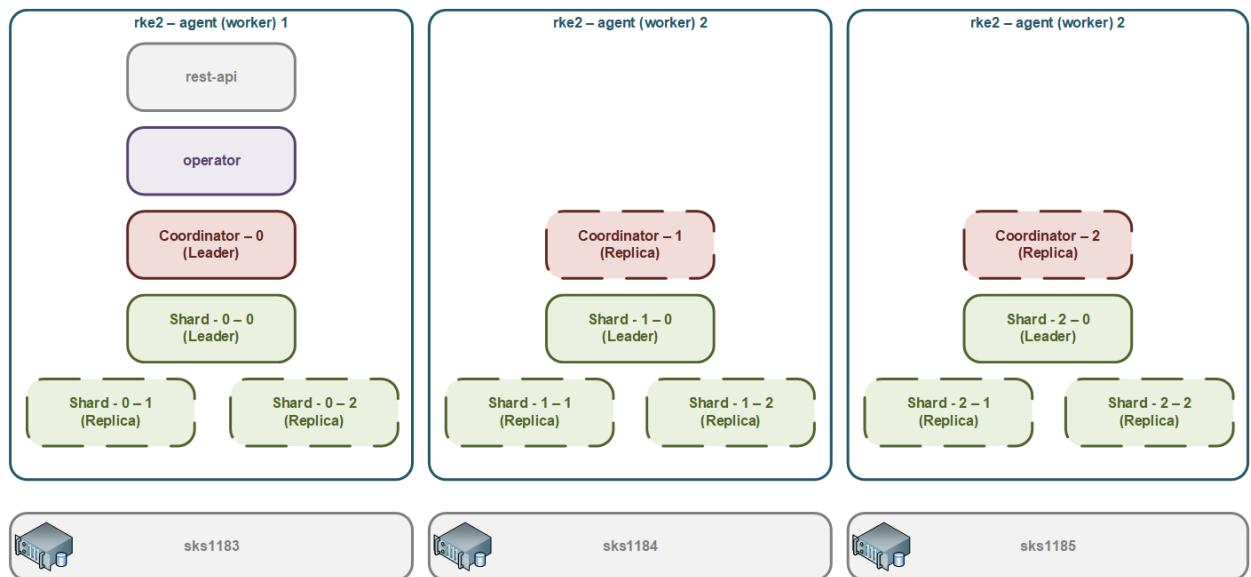


Abbildung 3.36: Stackgres - Citus - Evaluationsarchitektur Self Healing Tests

3.1.7.3.2 Ressourcenhunger

Aus den Architektschemen ist bereits ersichtlich, dass StackGres sehr viele Pods erstellt. StackGres erzeugt mindestens einen Operator- und einen REST-API-Pod, der aber auch für das GUI verwendet wird.

Nun kommt der Coordinator-Pod hinzu und je nach Auswahl pro Node ein Shard-Pod, wobei es mindestens eine Instanz braucht.

Will heissen, im **Worstcase** sind auf einem Node mindestens 4 Pods, auf dem Server kann aber auch noch der k8s-server (control-plane) stehen.

Pro Pod muss mindestens eine CPU gesetzt werden, auch der k8s-Server benötigt mindestens eine CPU, heisst das pro Server mit Minimal Setting 5 CPUs benötigt werden:

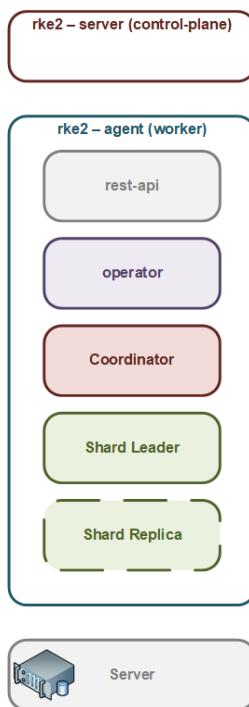


Abbildung 3.37: Stackgres - Citus - Ressourcen - Stack

Auch Memory und Storage muss eingerechnet werden, besonders wenn pro Shard noch mehrere Instanzen deployt werden sollen. Dazu kommt noch eine weitere **eigenheit** von StackGres.

Dazu kommt noch eine weitere Eigenheit von StackGres.

Pro Datenbank wird standardmässig ein Cluster erstellt mit jeweils mindestens einem Coordinator und dem ganzen Stack der daran hängt:

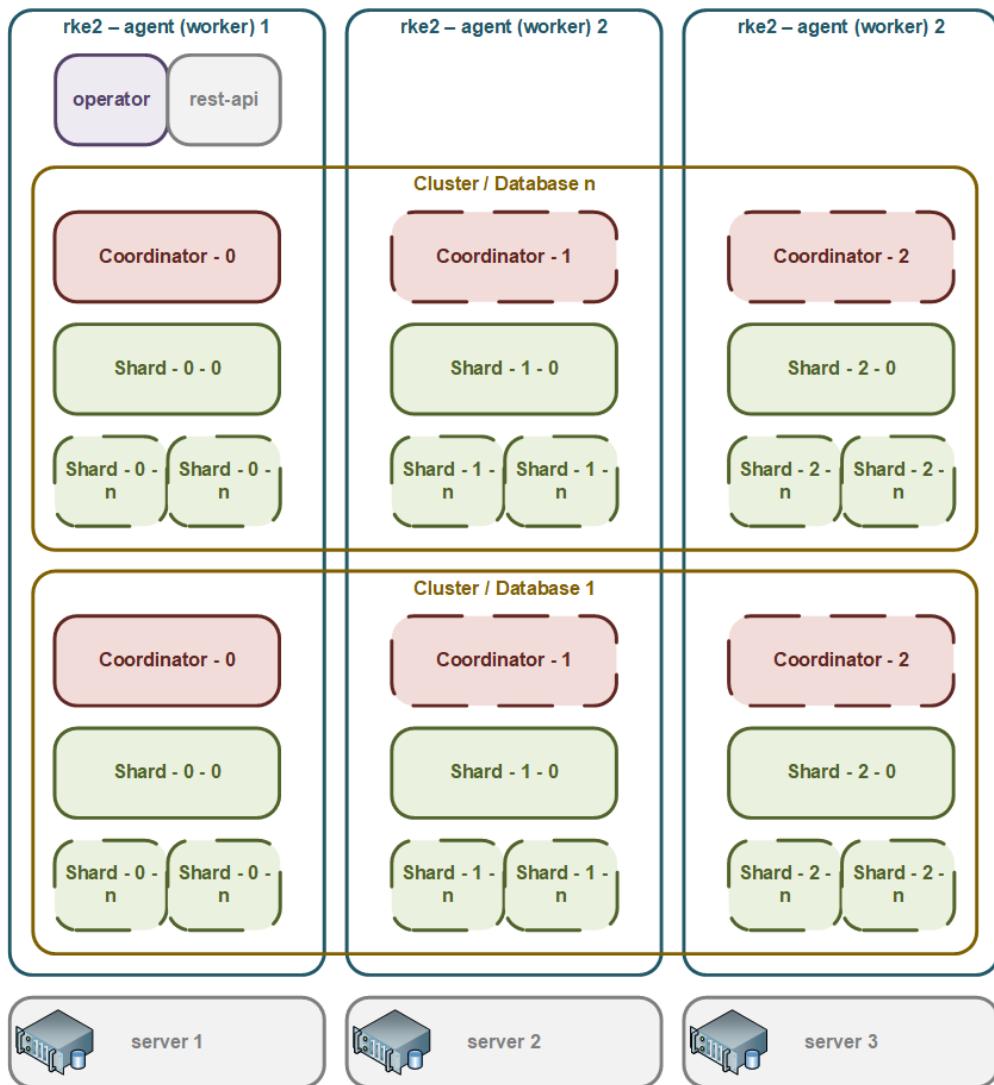


Abbildung 3.38: Stackgres - Citus - Datenbank - Cluster

Entsprechend steigt der Ressourcenbedarf zusätzlich.

3.1.7.3.3 Installation

Ongres bietet für StackGres ein helm-Chart, welches über ein eigenes `values.yaml`-Manifest oder direkt aus dem Repository mit Hilfe von Parametern deployet werden kann.

Beim KSGR wurde das helm-Chart heruntergeladen und entsprechend ein eigenes Manifest geschrieben.

StackGres bietet von Haus aus an, einen Sharded Cluster mit Citus zu installieren.

Dabei muss allerdings das StackGres Extension Repository erreichbar sein, wobei primär über [https](https://) kommuniziert wird.

Hier wird es nun knifflig, sobald Proxys im Spiel sind.

Selbst wenn die Proxy-Settings auf dem Host und im rke2 (CONTAINERD_HTTPS_PROXY / CONTAINERD_HTTP_PROXY / CONTAINERD_NO_PROXY) gesetzt sind, ist dies keine Garantie, das mit Hilfe von https aus dem Pod heraus kommuniziert werden kann, selbst wenn es mit curl möglich ist.

Damit eine Kommunikation stattfinden kann, müssen die Proxy-Zertifikate auf den Host installiert werden.

Alternativ kann die Kommunikation über http erzwungen werden.

StackGres bietet diese Möglichkeit und da es sich um eine Evaluationsumgebung handelt, wurde dieser Weg gewählt.

Zum einen muss der Proxy nach der proxyUrl eingegeben werden, danach müssen die Parameter skipHostnameVerification:true und setHttpScheme:true gesetzt werden.

Die Proxy-URL muss dabei wie folgt aufgebaut werden:

```
1 <proxy scheme>%3A%2F%2F<proxy host>%3A<proxy port>
```

Listing 3.6: StackGres - values.yaml - Extension proxyUrl

Proxy Schema meint dabei http oder https.

Für den KSGR-Proxy sieht der gesamte String entsprechend so aus:

```
1 extensions:
2   repositoryUrls:
3     - https://extensions.stackgres.io/postgres/repository?proxyUrl=http%3A%2F%
4       Fsproxy.svc.first-it.ch%3A8080?skipHostnameVerification:true&setHttpScheme:
5         true
```

Listing 3.7: StackGres - values.yaml - Extension Proxy

Anders als bei YugabyteDB kann man das Web-GUI nicht Zuhilfenahme eines Load Balancer Exposing nach aussen präsentieren, auch wenn eine Cluster-IP gesetzt werden kann.

Soll das Web-GUI und die REST-API permanent von aussen verfügbar sein, muss auf dem rest-api Pod ein permanentes Forwarding implementiert werden.

Umsetzen lässt sich dies mit Hilfe der entsprechenden Keys im values.yaml oder den Parametern beim Deploy.

3.1.7.3.4 Cluster Deployment

Mit der Installation von StackGres steht noch keine Datenbank, es steht nur der Operator- und REST-API Pod.

Dabei muss unterschieden werden, ob eine normale Patroni-Instanz deployt werden soll oder eine Sharded-Instanz (mit Citus).

Dazu braucht es vorgängig folgende Ressourcen, die deployt werden müssen:

StorageClass

Die StorageClass für den Cluster.

Je nachdem empfiehlt es sich, für den Coordinator eine eigene StorageClass zu erzeugen.

SGInstanceProfile

Das Instanz-Profil definiert, wie viel CPU und Memory der Pod erhält.

Die Konfigurationsmöglichkeiten gehen so tief,

dass innerhalb von Pods auch Containern Ressourcen zugewiesen werden können.

Empfehlenswert ist, Coordinator und Worker zu trennen.

Ohne ein separates Instanz-Profil wird ein Standardprofil mit einer CPU und einem GiB

Memory allokiert.

SGPostgresConfig

PostgreSQL-Spezifische Einstellungen können hierüber konfiguriert werden.

Wie das Instanz-Profil auch, ist dies nicht zwingend.

SGShardedCluster

Mit diesem Manifest wird der Cluster oder Sharded Cluster deployt.

Eigene Ressourcen wie Instanz-Profile müssen entsprechend deklariert werden.

Beim SGShardedCluster-Manifest gilt es einige Punkte zu beachten.

Wird beim Coordinator mehr als eine Instanz angegeben, so wird der Coordinator mit Hilfe von Patroni in einem Replica-Cluster betrieben.

Dies hat den Vorteil, dass der Unterbruch bei einem Node Failure kleiner ist.

Bei den shards gibt es allerdings zwei Parameter, die entscheidend sind.

Zum einen clusters, die entsprechend die Anzahl an Shard-Pods erzeugen.

Es müssen dann aber auch die Anzahl Instanzen beim Parameter instancesPerCluster gesetzt werden.

Bei nur einer Instanz wird keine Replikation auf die Nodes erzeugt, bei mehr als einer Instanz wird auf die Nodes verteilt.

Bei drei Nodes und drei Instanzen wird entsprechend auf alle Nodes repliziert, bei zwei Instanzen nur auf zwei von drei Nodes.

Damit die PostgreSQL-DB, hier in Form vom Service postgresServices, von ausserhalb erreichbar ist,

muss die IP-Adresse von MetalLB gebunden werden.

Zentral ist dabei die Annotation für den Primary-Service:

```
1  postgresServices:
2    coordinator:
3      primary:
4        type: LoadBalancer
```

```

5   any:
6     type: LoadBalancer
7   shards:
8     primaries:
9       type: LoadBalancer
10    metadata:
11      annotations:
12        primaryService:
13          metallb.universe.tf/loadBalancerIPs: 10.0.20.106
14        replicasService:
15          metallb.universe.tf/loadBalancerIPs: 10.0.20.153
16        externalTrafficPolicy: "Cluster"

```

Listing 3.8: StackGres-Citus - LoadBalancer -Annotation

Das erzeugen von Persistent Volume Claims für Coordinator- und Shard-Pods wird wie folgt deklariert (hier nur mit der StorageClass stackgres-storage):

```

1 ...
2   coordinator:
3     ...
4   pods:
5     persistentVolume:
6       size: '<Größe>Gi'
7       storageClass: "stackgres-storage"
8 ...
9   shards:
10  ...
11  pods:
12    persistentVolume:
13      size: 'GrößeGi'
14      storageClass: "stackgres-storage"

```

Listing 3.9: StackGres-Citus - StorageClass -PVC Binding

Die Instanz-Profile lassen sich wie folgt zuweisen:

```

1 ...
2   coordinator:
3     instances: 1
4     ...
5     sgInstanceProfile: "<Instanz-Profil Coordinator>"
6   ...
7   shards:
8     ...
9     sgInstanceProfile: "<Instanz-Profil Shard>"

```

Listing 3.10: StackGres-Citus - Instanz-Profile

Beim Benchmarking kam es zu einem sehr unschönen Fehler.

PgBounder verlor die Verbindung.

Nach einer kurzen Suche zeigte sich, dass es wohl einen Bug bei grösseren Workloads gibt, zumindest ist dies meine Interpretation.

Die Lösung bei einigen schien zu sein, dass das Pooling abgeschaltet wird[44].

Für das Benchmarking wurde dies dann auch umgesetzt.

Dies wird folgendermassen gemacht:

```
1 ...
2   coordinator:
3     pods:
4       ...
5         disableConnectionPooling: true
6 ...
```

Listing 3.11: StackGres-Citus - StorageClass -PVC Binding

Bei einem produktiven System müsste dieser Bug aber gefixt werden.

Bei einem Drei-Node Environment, wie es für die Evaluation verwendet wird, kommt es zu einem Konflikt, wenn drei Shard-Pods erzeugt werden.

In diesem Fall muss ein Testing- oder Development-Profil gesetzt werden.

Alternativ können Pod Anti-Affinities oder Pod Affinities gesetzt werden, was nicht einfach ist, da Ongres die Pod Anti-Affinities unzureichend dokumentiert hat.

Daher wurden Testing-Profil-Flags aktiviert:

```
1 apiVersion: stackgres.io/v1alpha1
2 kind: SGShardedCluster
3 metadata:
4   name: <cluster / db name>
5   namespace: <cluster namespace>
6 spec:
7   ...
8   profile: "testing"
```

Listing 3.12: StackGres-Citus - Cluster Profil

Es gäbe zwar die Möglichkeit, Passwörter via Manifest zu setzen, aber dies funktioniert nicht für postgres- und replicator sowie backup.

Laut der StackGres-Dokumentation müssen z. B. die Passwörter vom postgres-User im Nachgang via SQL geändert werden.

Während der Evaluation wurde darauf verzichtet und das generische Passwort aus dem Cluster geholt.

Für den 250GiB-Benchmark wurden ähnlich zur Patroni-Evaluation spezifische PostgreSQL Cluster Konfigurationen im Manifest SGPostgresConfig vorgenommen.

Diese Settings können dann jeweils für die Shards als auch für die Coordinators **Zugewiesen** werden:

```
1 ...
2 coordinator:
3 ...
4 configurations:
5   sgPostgresConfig: "<SGPostgresConfig Name>"
6 ...
7 shards:
8 ...
9 configurations:
10  sgPostgresConfig: "<SGPostgresConfig Name>"
```

Listing 3.13: StackGres-Citus - SGPostgresConfig

Die vollständige Dokumentation der Evaluationsinstallation ist im [Anhang - Installation StackGres - Citus](#) zu finden.

3.1.7.4 YugabyteDB

3.1.7.4.1 Architektur

Die Architektur ist einfach.

Auf allen Worker-Nodes wird je ein `tmaster` und `tserver` Pod gestartet, wie dem Architekturschaubild zu entnehmen ist:

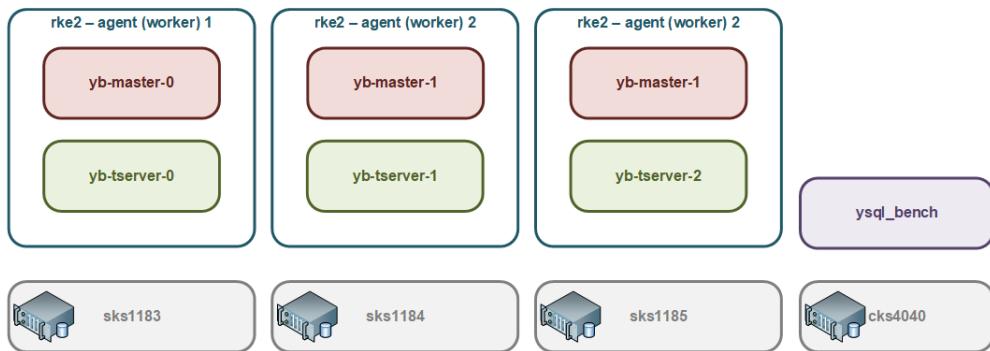


Abbildung 3.39: YugabyteDB - Evaluationsarchitektur

3.1.7.4.2 Installation

Während der Installation des YugabyteDB Evaluations-Environment wurde festgestellt, das man zwei Varianten installieren kann.

YugabyteDB (Repository `yugabyte`) und YugabyteDB Anywhere (Repository `yugawre`):

```

Context: default
Cluster: default
User: default
K9s Rev: v0.31.8 ✨v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU: 1%
MEM: 38%
Name: yw-test-yugaware-pg-upgrade
Optional: false
Type: ConfigMap (a volume populated by a ConfigMap)
Name: yw-test-yugaware-pg-prerun
Optional: false
pg-sample-config:
Type: ConfigMap (a volume populated by a ConfigMap)
Name: yw-test-pg-sample-config
Optional: false
kube-apiserver-rgtwb:
Type: Projected (a volume that contains injected data from multiple sources)
TokenExpirationSeconds: 3600
ConfigMapName: kube-root-ca.crt
ConfigMapOptional: encls
DownloadAPI: true
QoS Class: Burstable
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
Type Reason Age From Message
Normal Scheduled 3m22s default-scheduler Successfully assigned yb-platform/yw-test-yugaware-0 to sks1185
Normal Pulling 2m39s (x3 over 3m22s) kubelet Pulling image "quay.io/yugabyte/yugaware:2.20.2.1-b3"
Warning Failed 2m38s (x3 over 3m21s) kubelet Failed to pull image "quay.io/yugabyte/yugaware:2.20.2.1-b3": failed to pull and unpack image "quay.io/yugabyte/yugaware:2.20.2.1-b3": failed to resolve reference "quay.io/yugabyte/yugaware:2.20.2.1-b3": unexpected status from HEAD request to https://quay.io/v2/yugabyte/yugaware/manifests/2.20.2.1-b3: 401 UNAUTHORIZED
Warning Failed 2m38s (x3 over 3m21s) kubelet Error: ErrImagePull
Normal BackOff 2m11s (x4 over 3m20s) kubelet Back-off pulling image "quay.io/yugabyte/yugaware:2.20.2.1-b3"
Warning Failed 2m11s (x4 over 3m20s) kubelet Error: ImagePullBackoff
Warning FailedToRetrieveImagePullSecret 117s (x8 over 3m23s) kubelet Unable to retrieve some image pull secrets (yugabyte-k8s-pull-secret); attempting to pull the image may not succeed.

```

<namespace> <pod> <describe>

Abbildung 3.40: YugabyteDB - Subscription Yugaware

Es stellte sich auch heraus, dass wenn man YugabyteDB 4 Cores pro Node zur Verfügung geben will (je zwei für den master und tserver), der Server mehr als 4 Cores haben muss.

Andernfalls wird Kubernetes einen der beiden Pods nicht deployen, weil zu wenig Cores zur Verfügung stehen.

Installation von rke2, Cilium und MetallLB, muss nebst dem IPAddressPool auch ein L2Advertisement für den Pool gesetzt werden.

Ansonsten kann die im YugabyteDB values.yaml gesetzte IP für den tserver von aussen nicht angesprochen werden:

```

1 ---
2 apiVersion: metallb.io/v1beta1
3 kind: L2Advertisement
4 metadata:
5   name: l2adv
6   namespace: metallb-system
7 spec:
8   ipAddressPools:
9     - distributed-sql
10

```

Listing 3.14: metallb - Konfig YAML - Detail L2Advertisement

Die Vorschläge zum Lösen des Problems reichten von Deaktivieren von kube-proxy bis hin zu einer Migration zum Cilium-Loadbalancers.

Mit diesem funktionierte dann nicht einmal mehr die Installation von YugabyteDB. Lösung brachte nur ein GitHub-Eintrag[39], wo oben genannter Ansatz empfohlen wurde.

3.1.7.4.3 Konfiguration

Damit nicht der YugabyteDB Anywhere-Service installiert wird, muss das entsprechende Image gesetzt werden:

```
1 ...
2 Image:
3   repository: "yugabytedb/yugabyte"
4   tag: 2.20.2.1-b3
5   pullPolicy: IfNotPresent
6   pullSecretName: ""
7 ...
8 ...
```

Listing 3.15: YugabyteDB - Helm Chart Manifest - Detail Image

Die StorageClass muss im `values.yaml` gesetzt werden, je einmal für den `tmaster` und `tserver`:

```
1 ...
2 storage:
3   ephemeral: false # will not allocate PVs when true
4   master:
5     count: 1
6     size: 3Gi
7     storageClass: "yb-storage"
8   tserver:
9     count: 1
10    size: 3Gi
11    storageClass: "yb-storage"
12 ...
13 ...
```

Listing 3.16: YugabyteDB - Helm Chart Manifest - Detail StorageClass

Dem Node werden je 4 Cores zur Verfügung gestellt. Zwei für den `master` und zwei für den `tserver`. Beide erhalten 4GiB Memory:

```
1 ...
2 resource:
3   master:
4     requests:
5       cpu: "1"
6       memory: 2Gi
7     limits:
8       cpu: "1"
```

```

9      ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating
10     from these formats
11     ## may result in setting an incorrect value for the 'memory_limit_hard_bytes'
12     ' flag.
13     ## Avoid using floating numbers for the numeric part of 'memory'. Doing so
14     may lead to
15     ## the 'memory_limit_hard_bytes' being set to 0, as the function expects
16     integer values.
17     memory: 2Gi
18
19 tserver:
20   requests:
21     cpu: "1"
22     memory: 4Gi
23   limits:
24     cpu: "1"
25     ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating
26     from these formats
27     ## may result in setting an incorrect value for the 'memory_limit_hard_bytes'
28     ' flag.
29     ## Avoid using floating numbers for the numeric part of 'memory'. Doing so
30     may lead to
31     ## the 'memory_limit_hard_bytes' being set to 0, as the function expects
32     integer values.
33     memory: 4Gi
34
35 ...
36

```

Listing 3.17: YugabyteDB - Helm Chart Manifest - Detail Resources

Die Shards oder Tablets wie sie Yugabyte nennt, sollen auf allen drei Nodes repliziert werden:

```

1 ...
2 replicas:
3   master: 3
4   tserver: 3
5   ## Used to set replication factor when isMultiAz is set to true
6   totalMasters: 3
7 ...
8

```

Listing 3.18: YugabyteDB - Helm Chart Manifest - Detail Replika

Wichtig ist auch, dass der YSQL-Dienst aktiv ist, damit PostgreSQL Abfragen abgesetzt werden können.

Deshalb muss der Dienst aktiv sein und darf nicht deaktiviert werden:

```

1 ...
2 # Disable the YSQL
3 disableYsql: false
4 ...

```

Listing 3.19: YugabyteDB - Helm Chart Manifest - Detail Disable YSQL

Nun müssen die Domain und die Service-Endpoints konfiguriert werden.
Der Domainname bleibt vorerst `cluster.local` wie Default hinterlegt.
Die Servicenamen und Ports werden nicht angetastet, wichtig ist die LoadBalancer-IP.
Sie ist entsprechend der gewählten VirtualIP mit `10.0.20.106` zu setzen.

```

1 ...
2 domainName: "cluster.local"
3
4 serviceEndpoints:
5   - name: "yb-master-ui"
6     type: LoadBalancer
7     annotations: {}
8     clusterIP: ""
9     ## Sets the Service's externalTrafficPolicy
10    externalTrafficPolicy: ""
11    app: "yb-master"
12    loadBalancerIP: ""
13    ports:
14      http-ui: "7000"
15
16   - name: "yb-tserver-service"
17     type: LoadBalancer
18     annotations:
19       metallb.universe.tf/loadBalancerIPs: 10.0.20.106
20     clusterIP: ""
21     ## Sets the Service's externalTrafficPolicy
22     externalTrafficPolicy: ""
23     app: "yb-tserver"
24     loadBalancerIP: ""
25     ports:
26       tcp-yql-port: "9042"
27       tcp-yedis-port: "6379"
28       tcp-ysql-port: "5433"
29 ...
30

```

Listing 3.20: YugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints

Beim Testen mit der höchsten Anzahl an Datensätzen zeigte sich, dass der local-path-provisioner nicht sauber konfiguriert waren.
Damit auf jedem Node die Persistence Volume Claims ausgeführt werden, müssen sie deklariert werden und in den StorageClass-Manifesten auch hinterlegt werden.
Genauer muss in der `nodePathMap` folgende Konfiguration vorgenommen werden:

```

1 ...
2         "nodePathMap": [
3             {
4                 "node": "DEFAULT_PATH_FOR_NON_LISTED_NODES",
5                 "paths": ["<Lokaler Pfad>"]
6             },
7             {
8                 "node": "<Nodename>",
9                 "paths": ["<Lokaler Pfad>"]
10            },
11 ...

```

Listing 3.21: local-path-provisioner nodePathMap

Hier ein Beispiel, wie es mit den grossen Volumes aussieht:

```

1 ...
2         "nodePathMap": [
3             {
4                 "node": "DEFAULT_PATH_FOR_NON_LISTED_NODES",
5                 "paths": ["/srv/data/local-path-provisioner"]
6             },
7             {
8                 "node": "sks1183",
9                 "paths": ["/srv/data/local-path-provisioner"]
10            },
11            {
12                "node": "sks1184",
13                "paths": ["/srv/data/local-path-provisioner"]
14            },
15            {
16                "node": "sks1185",
17                "paths": ["/srv/data/local-path-provisioner"]
18            }
19        ]
20 ...

```

Listing 3.22: local-path-provisioner nodePathMap Beispiel

Wird dies nicht gemacht, so wird auf den Default-Path geschrieben.

Das ist zufällig und hat dann zur Folge, dass alle Volumes auf einem Node präsentiert werden.

Was sehr schnell zur Folge hat, dass zu wenig Diskspace vorhanden ist.

Bei YugabyteDB kommt noch dazu, dass es zu Konflikten beim Schreiben von Blocks kommt.

Damit die Persistence Volumes sauber präsentiert werden, muss in der StorageClass die `nodeAffinity` gesetzt werden.

Hier als Beispiel mit den Nodes `sks1183`, `sks1184` und `sks1185`:

```

1   nodeAffinity:
2     required:

```

```

3   nodeSelectorTerms:
4     - matchExpressions:
5       - key: kubernetes.io/hostname
6         operator: In
7         values:
8           - sks1183
9           - sks1184
10          - sks1185

```

Listing 3.23: YugabyteDB - StorageClass nodeAffinity

hostPath

Der hostPath bei der StorageClass muss der gleiche sein, wie der Pfad im Node des nodePathMap von local-path-provisioner. Auch sollten die Pfade auf allen Nodes gleich sein.

Die Problematik mit dem nodePathMap und der nodeAffinity auf der StorageClass hat auch rund zwei Arbeitstage in Anspruch genommen.

Die vollständige Dokumentation der Evaluationsinstallation ist im [Anhang - Installation YugabyteDB](#) zu finden.

3.1.8 Testing Evaluationssysteme

3.1.8.1 Patroni

Patroni funktionierte wie gewollt.

Da kein Connection Pooler auf dem Proxy-Host installiert wurde, kam nicht alles erfüllt werden.

Wichtig dazu ist zu sagen, dass die REST-API und das Command vollständig funktioniert.

Ein Switchover wurde mit folgendem Command ausgeführt:

```

1 root@sks1234:~# patronictl -c /etc/patroni/config.yml switchover
2 Current cluster topology
3 + Cluster: postgres (7357340759952276373) +-----+-----+-----+
4 | Member      | Host        | Role      | State    | TL | Lag in MB |
5 +-----+-----+-----+-----+-----+-----+-----+
6 | postgres01  | 10.0.20.110 | Leader    | running  | 3  |           |
7 | postgres02  | 10.0.20.111 | Sync Standby | streaming | 3  | 0          |
8 | postgres03  | 10.0.20.112 | Sync Standby | streaming | 3  | 0          |
9 +-----+-----+-----+-----+-----+-----+
10 Primary [postgres01]:
11 Candidate ['postgres02', 'postgres03'] []: postgres02
12 When should the switchover take place (e.g. 2024-04-26T16:08 ) [now]: now
13 Are you sure you want to switchover cluster postgres, demoting current leader
   postgres01? [y/N]: y
14 2024-04-26 15:09:02.68997 Successfully switched over to "postgres02"

```

```

15 + Cluster: postgres (7357340759952276373) -----+-----+
16 | Member | Host | Role | State | TL | Lag in MB |
17 +-----+-----+-----+-----+-----+-----+
18 | postgres01 | 10.0.20.110 | Replica | stopped | | unknown || postgres02 |
19 | | 10.0.20.111 | Leader | running | 3 | |
20 | postgres03 | 10.0.20.112 | Replica | running | 3 | 0 |
+-----+-----+-----+-----+-----+-----+

```

Listing 3.24: Patroni - Testing - Switchover

Ein gestoppter Node wurde wie folgt wieder neu aufgebaut:

```

1 root@sks1234:~# patronictl -c /etc/patroni/config.yml reinit postgres
2 + Cluster: postgres (7357340759952276373) +-----+-----+-----+
3 | Member | Host | Role | State | TL | Lag in MB |
4 +-----+-----+-----+-----+-----+-----+
5 | postgres01 | 10.0.20.110 | Sync Standby | streaming | 4 | 0 |
6 | postgres02 | 10.0.20.111 | Leader | running | 4 | |
7 | postgres03 | 10.0.20.112 | Sync Standby | streaming | 4 | 0 |
8 +-----+-----+-----+-----+-----+-----+
9 Which member do you want to reinitialize [postgres03, postgres01]? []: postgres01
10 Are you sure you want to reinitialize members postgres01? [y/N]: y
11 Success: reinitialize for member postgres01

```

Listing 3.25: Patroni - Testing - Reinit

Das vollständige Ergebnis:



Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis	Begründung
Failover	1	Automatismus	Wird der Primary Server vom Netz genommen, führt Patroni einen Failover auf einen Replika-Node	Eingetroffen	Connection-Stabilität kann nur hergestellt werden, wenn entweder die Applikation dazu in der Lage ist oder man einen Connection-Pooler wie pgBouncer einsetzt. Es wurde aber keiner eingesetzt.
Failover	2	Connection-Stabilität	Bestehende Connections dürfen nicht getrennt werden.	Nicht eingetroffen	Auch hier hängt die Stabilität an den Settings der Applikation und einem Connection-Pooler.
Failover	3	Geschwindigkeit	Der Failover muss so schnell stattfinden, dass offene Connections nicht wegen eines Timeouts geschlossen werden.	Bedingt eingetroffen	Connection-Stabilität kann nur hergestellt werden, wenn entweder die Applikation dazu in der Lage ist oder man einen Connection-Pooler wie pgBouncer einsetzt. Es wurde aber keiner eingesetzt.
Switchover	4	Skript / API	Mit der Patroni REST-API wird der Switchover ausgeführt	Eingetroffen	
Switchover	5	Skript / API	Mit dem Patroni Commandset wird er Switchover ausgeführt	Eingetroffen	
Switchover	6	Connection-Stabilität	Bestehende Connections dürfen nicht getrennt werden.	Eingetroffen	
Switchover	7	Geschwindigkeit	Der Switchover muss so schnell stattfinden, dass offene Connections nicht wegen eines Timeouts geschlossen werden.	Nicht eingetroffen	Connection-Stabilität kann nur hergestellt werden, wenn entweder die Applikation dazu in der Lage ist oder man einen Connection-Pooler wie pgBouncer einsetzt. Es wurde aber keiner eingesetzt.
Restore	9	Skript / API	Mit der Patroni REST-API wird der Primary-Node Wiederhergestellt	Eingetroffen	
Restore	10	Skript / API	Mit dem Patroni Commandset der Primary-Node Wiederhergestellt	Eingetroffen	
Restore	11	Skript / API	Mit der Patroni REST-API wird ein Replika-Node Wiederhergestellt	Eingetroffen	
Restore	12	Skript / API	Mit dem Patroni Commandset ein Replika-Node Wiederhergestellt	Eingetroffen	
Restore	13	Datensicherheit	Beim Restore des Primary-Nodes dürfen keine Daten, die seit dem Failover geschrieben wurden, darf es zu keinem Datenverlust kommen	Eingetroffen	Connection-Stabilität kann nur hergestellt werden, wenn entweder die Applikation dazu in der Lage ist oder man einen Connection-Pooler wie pgBouncer einsetzt. Es wurde aber keiner eingesetzt.
Restore	14	Connection-Stabilität	Beim Restore des Primary-Nodes dürfen keine Connections geschlossen werden.	Nicht eingetroffen	

Tabelle 3.13: Testresultate Evaluation Patroni

3.1.8.2 StackGres -Citus

StackGres kann nicht alle Anforderungen erfüllen.

Obwohl es mit envoy und pgBouncer einen Proxy und einen Connection Pooler gibt,

scheint dies nicht über die Coordinator-Nodes selbst zu gehen.

Daher brechen bestehende Connections ab oder laufen irgendwann in ein Timeout, wenn Kubernetes Nodes nicht schnell genug heruntergefahren werden.

Aufgrund des Sharding und das in sich geschlossenen Kubernetes-Environments wurde auf separate Tablespaces verzichtet.

Zuerst wurde versucht, das Sharding mit Version 12 eingeführte Schema Based Sharding umzusetzen.

Wie beim Benchmarking auch, zeigten sich schnell die Grenzen des Citus-Sharding.

Sobald ein Foreign-Key zwischen zwei Tabellen, die in verschiedenen Schemas liegen, existiert, kann kein Schema Based Sharding mehr ausgeführt werden.

Auch hier besteht die Lösung darin, Reference Tables zu erstellen.

Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis	Begründung
Failover	1	Automatismus	Wird der Primary Server vom Netz genommen, führt Patroni einen Failover auf einen Replika-Node	Eingetroffen	
Failover	2	Connection-Stabilität	Bestehende Connections dürfen nicht getrennt werden.	Nicht eingetroffen	
Failover	3	Geschwindigkeit	Der Failover muss so schnell stattfinden, dass offene Connections nicht wegen eines Timeouts geschlossen werden.	Nicht eingetroffen	
Sharding und Datenintegrität	4	Datenkonsistenz	Daten sind Konsistent und Inetger. Eingetroffen	Keine.	Citus setzt envoy ein. Offensichtlich nicht ber einen ganzen Cluster
Sharding	5	Schutz vor Datenverlust	Die Daten müssen Konsistent und schnell auf die Shards verteilt werden	Eingetroffen	
Self Healing	6	Node stellt sich selber wieder her	Shard Node wird automatisch synchronisiert	Eingetroffen	
Self Healing	7	Leader wird automatisch gesetzt	Leader wird entweder beibehalten oder wird neu gesetzt wenn ein Node zurückkehrt	Eingetroffen	

Tabelle 3.14: Testresultate Evaluation StackGres - Citus

Die genauen Details sind im [Anhang - StackGres - Citus Testing](#) zu finden.

3.1.8.3 YugabyteDB

YugabyteDB funktionierte so weit:

Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis
Failover	1	Automatismus	Wird ein Node vom Netz genommen, muss es zu einem Rebalancing kommen	Eingetroffen
Failover	2	Connection-Stabilität	Bestehende Connections dürfen nicht getrennt werden.	Eingetroffen
Failover	3	Geschwindigkeit	Der Failover muss so schnell stattfinden, dass offene Connections nicht wegen eines Timeouts geschlossen werden.	Eingetroffen
Sharding und Datenintegrität	4	Datenkonsistenz	Daten sind Konsistent und Inetger. Eingetroffen	
Sharding	5	Schutz vor Datenverlust	Die Daten müssen Konsistent und schnell auf die Tablets verteilt werden	Eingetroffen
Self Healing	6	Node stellt sich selber wieder her	Tablet wird automatisch synchronisiert	Eingetroffen

Tabelle 3.15: Testresultate Evaluation YugabyteDB

Was es aber bei einer Testinstallation zu prüfen gilt, ist die Zeiteinstellung.

Während des Testing kam es immer wieder vor, dass ein Node Probleme mit der Zeit bekam. Dies fiel immer dann auf, wenn ein Node (meistens `sk1184`), heruntergefahren und später rebooted wurde.

Der Fehler trat auch erst auf, als die Nodes aus einem Grund aus einem Snapshot wiederhergestellt werden mussten.

YugabyteDB stellt dann oft mehr als 500ms Zeitunterschied zwischen dem Tablet-Leader und dem Follower fest.

Sobald dies zutrifft, ist der Server Node nicht mehr arbeitsfähig, da die Zeit für die Synchronisation der Daten benötigt wird[92].

Oft kam auch die Meldung, dass chronyc nicht mehr auf dem Pod installiert sei.

Auf dem Servern scheinen die Zeiten aber synchron zu sein, eine genaue Ursache konnte nicht gefunden werden.

Eine mögliche Ursache ist eine unsaubere Konfiguration von rke2.

Der Beschrieb, wie sich der Fehler dann äussert, ist hier im [Anhang - YugabyteDB Testing](#) zu finden.

3.1.9 Gegenüberstellung der Lösungen

3.1.9.1 Benchmarking - Vorgehen

3.1.9.1.1 YugabyteDB

Zuerst muss die Datenbank erstellt und die Tablespace erzeugt werden, die genauen Schritte sind im [Anhang - YugabyteDB Benchmark SQL](#) zu finden.

Anschliessend muss pro Lauf erst initialisiert werden, dann kann mit dem eigentlichen Benchmarking gestartet werden.

Alle Benchmarking-Commands sind im [Anhang - YugabyteDB Benchmarking Commands](#) zu finden.

3.1.9.1.2 Patroni

Als die 250GiB DB getestet wurde, zeigte sich, dass die Parameter nicht darauf optimiert waren. Die Standby-Server konnten die WAL-Files nicht mehr abarbeiten, so stauten sich auf dem primären Node die Files und die Disk lief jeweils voll.

Es wurde an verschiedenen Stellschrauben geschraubt, etwa an der Anzahl Worker, der Grösse der WAL-Files und anderen.

Mit den Anpassungen wurde die Datenmenge gestemmt.

Filesystem und Cluster mussten beim Initialisieren der letzten Benchmarks permanent überwacht werden.

Dafür wurden folgende Commands verwendet:

```
1 watch --interval=30 df -h --human-readable
2 watch --interval=30 du -h --human-readable /srv/data/
```

```
3 watch --interval=30 patronictl -c /etc/patroni/config.yml list
```

Listing 3.26: Patroni - Benchmarking - Monitoring

Alle Benchmarking-Commands und SQLs zur Ermittlung der Grösse sind im [Anhang - Patroni Benchmarking Commands](#) zu finden.

3.1.9.1.3 StackGres - Citus

Beim Benchmarking zeigten sich die Grenzen des Citus Shardings.

Bereits beim Lesen der Anleitung fiel auf, dass für das Sharding der Tabellen pgbench_accounts und pgbench_history jeweils neu initialisiert wurde[9].

Das hat einen besonderen Grund.

Wenn die Tabellen mit Hilfe von des SELECT-Statements `create_distributed_table`, einem Sharding unterzogen werden sollen, kommt es zu einer Fehlermeldung.

Die Shards würden mit folgenden SQL-Statements erzeugt:

```
1 SELECT create_distributed_table('pgbench_branches', 'bid');
2 SELECT create_distributed_table('pgbench_tellers', 'tid');
3 SELECT create_distributed_table('pgbench_accounts', 'aid');
4 SELECT create_distributed_table('pgbench_history', 'aid');
```

Listing 3.27: Citus - Benchmarking - Distributed Table Sharding

Ursache ist, dass eine Distributed Table keine Foreign Key Constraints erlaubt.

pgbench erzeugt aber gleich mehrere davon:

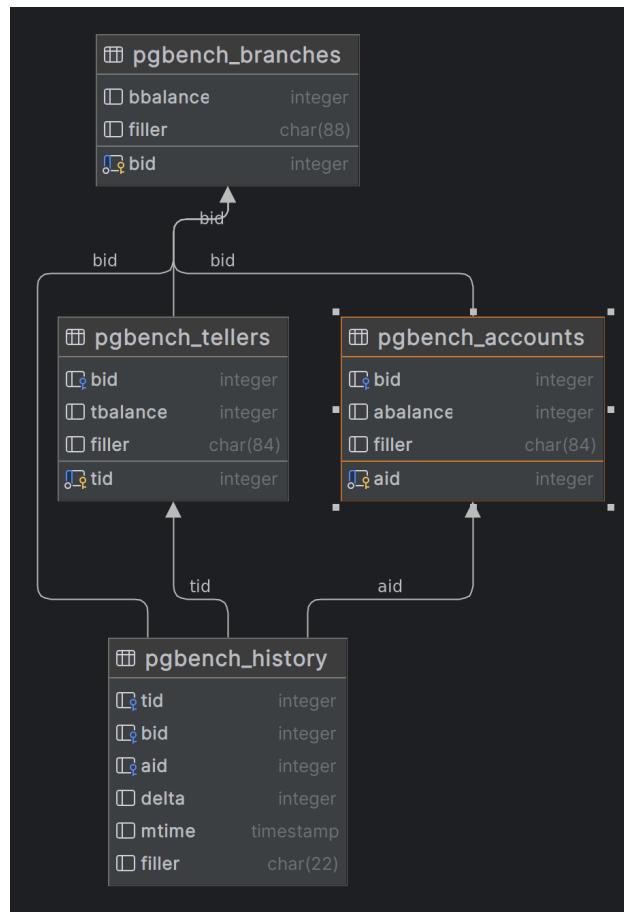


Abbildung 3.41: Benchmarking - ERD pgbench

Ein Schema-Based Sharding ist nicht möglich, da pgbench nur die DB als Parameter übernimmt. Die Tabellen werden zudem immer dropped, bevor sie neu erzeugt und gefüllt werden, das Sharding kann daher nur im Nachgang gemacht werden.

Die Lösung für das Benchmarking bestand also darin, sogenannte Reference Tables zu erstellen:

```

1 SELECT create_reference_table('pgbench_branches');
2 SELECT create_reference_table('pgbench_tellers');
3 SELECT create_reference_table('pgbench_accounts');
4 SELECT create_reference_table('pgbench_history');

```

Listing 3.28: Citus - Benchmarking - Reference Table Sharding

Referenzierte Tabellen werden auf alle Shards repliziert und erfüllen somit die Anforderung an das Sharding.

Diese Art des Sharding wäre allerdings nur für kleinere Tabellen, Multi-Tenant Sharding (wenn Tabellen bei allen Tenants verfügbar sein sollen),

Tabellen, die mit verschiedenen Distributed Tables gejoint werden oder wenn eben, wie in unserem Fall, Foreign-Key Constraints im Spiel sind[22].

Aber auch in diesem Fall muss das Sharding im Nachgang des pgbench-Init's gemacht werden. Bei den kleinen Tabellen geht das relativ **flot**, doch gerade bei der Tabelle pgbench_accounts dauert es sehr lange.

Lang genug, dass eine eigene Betrachtung beim Benchmarking angezeigt wurde.

Leider zeigte sich auch bei den mixed-Benchmarks anders als bei den dql-Benchmarks, dass diese Art des Sharding nicht sehr performant ist.

Wie bei Patroni und YugabyteDB auch, musste für den letzten Benchmark die StorageClass auf die neue Disk verlegt werden.

Aber anders als bei YugabyteDB reichten 250GiB nicht mehr, wie bei Patroni lag die Ursache beim Generieren der Primary- und Foreign-Keys.

Es zeigte sich aber auch rasch, dass der Coordinator die gleiche Grösse annahm wie die Shard-Pods.

Das führte dazu, dass beim letzten Benchmark nur noch 2 Shard-Instanzen deployt werden konnten,

da sonst bei jeder Disk nochmals mindestens 350GiB hinzugefügt hätte werden müssen (da sich der Coordinator den Node mit einem Shard geteilt hätte und der Coordinator auf jedem Node erscheinen könnte).

Es hätten also für die Evaluation 3 x 700GiB (plus noch 3 x 50GiB für den Rest), sprich 3 x 750GiB, allokiert werden müssen.

Auf diesen Mehraufwand wurde verzichtet, um das SAN und somit das Daily Business nicht zu stark zu belasten.

Allerdings war es nicht möglich, die Shards mit rund 250GiB auszuführen.

Trotz grösserer Disk und mehr Ressourcen auf den Shards und Coordinators brach das Sharding der Tabelle pgbench_accounts immer ab.

Die PostgreSQL-Instanz wurde optimiert, das heisst, es wurden so viele Parameter wie möglich der Patroni-Evaluation gesetzt, allerdings gab es von seitens StackGres Einschränkungen.

Das SQL lief für jeweils knapp 13 Minuten, ehe es zu einem Disconnect im Java-Code von einem der beiden Shards kam.

Die Vermutung liegt nahe, dass es ein Memory-Leak im Code gibt, s lassen sich aber nicht genug Informationen aus den Logs gewinnen, um die Fehlerursache und eine mögliche Lösung zu finden.

 Aufgrund von diesem Fehler ist das Benchmarking für StackGres unvollständig.

Alle Benchmarking-Commands und SQLs zur Ermittlung der Grösse sind im [Anhang - StackGres - Citus Benchmarking Commands](#) zu finden.

3.1.9.2 Benchmarks

Der Vergleich zwischen den verschiedenen Varianten.

Bei den Transaktionen pro Sekunden gilt, je höher der Wert, umso besser das Ergebnis.

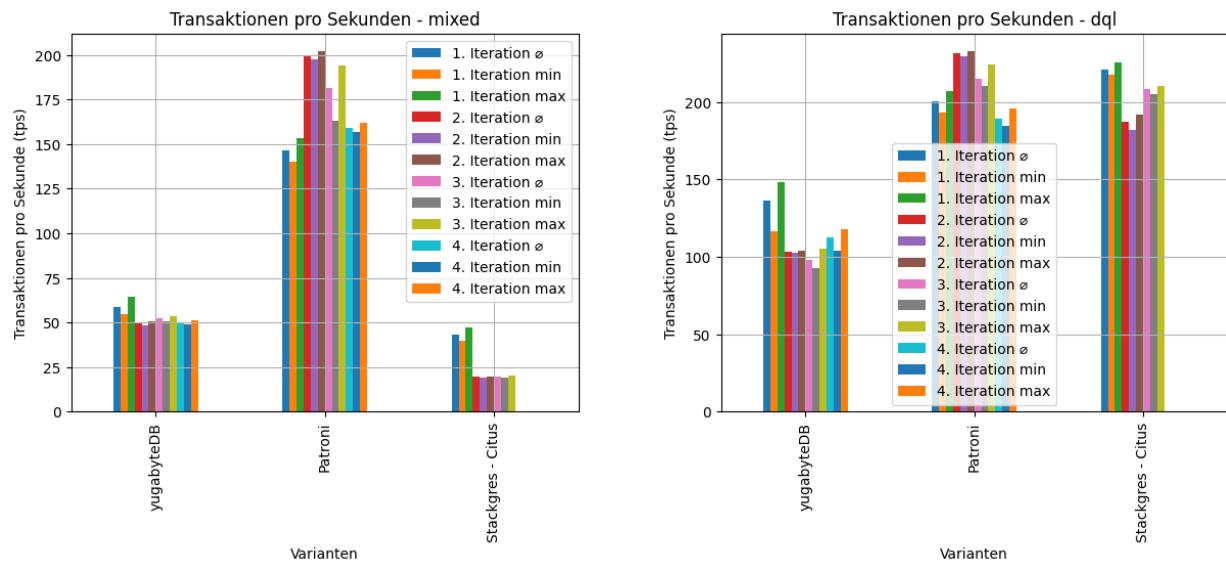


Abbildung 3.42: Benchmarks - tps

Bei der Latenz ist es genau andersrum, je höher der Wert, desto schlechter schnitt die Variante ab.

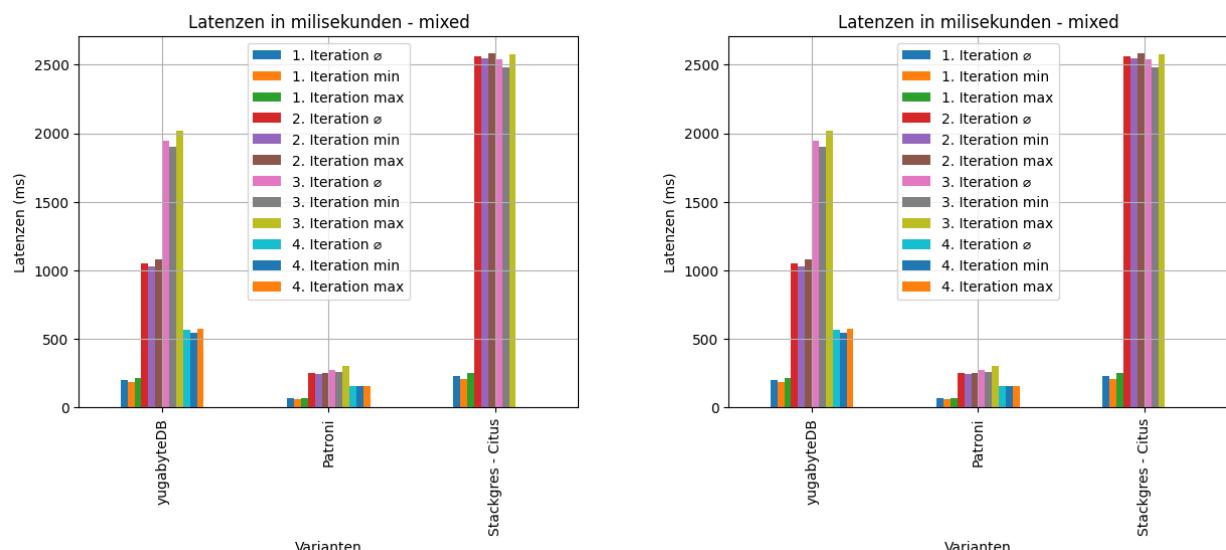


Abbildung 3.43: Benchmarks - latency

Die ersten beiden Läufe mit Patroni wurde erst nur mit der asynchronen Standard-Replikation

von Patroni vorgenommen.

Später wurden die Benchmarks mit der synchronen Replikation wiederholt.

Daraus ergab sich die Möglichkeit, beide Methoden direkt zu vergleichen:

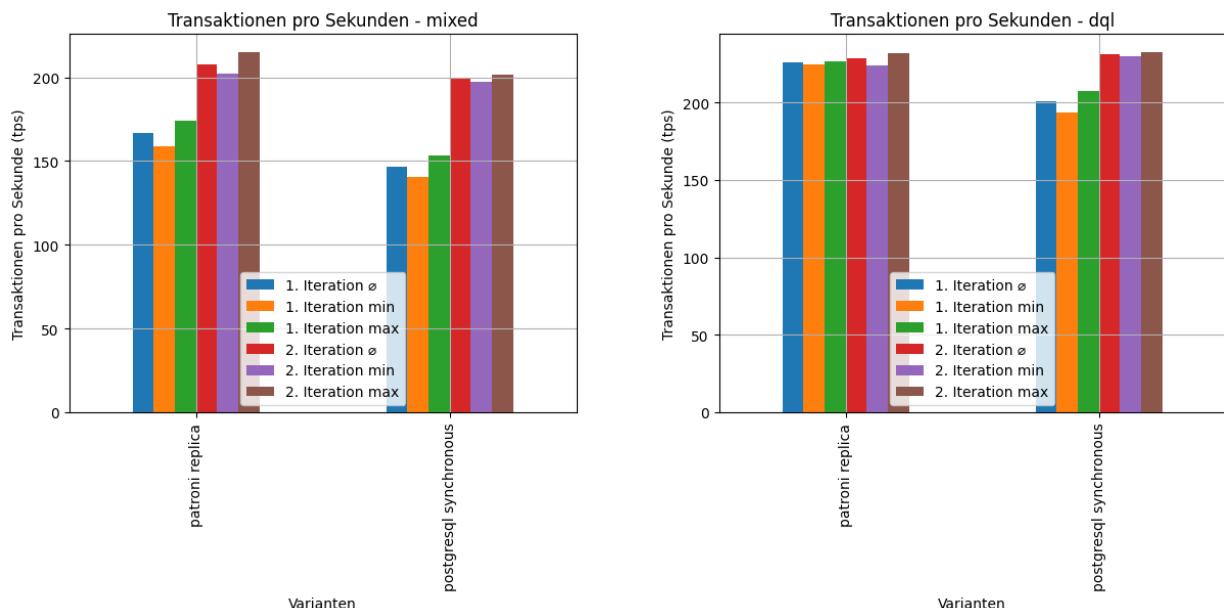


Abbildung 3.44: Benchmarks - tps Patroni Replica

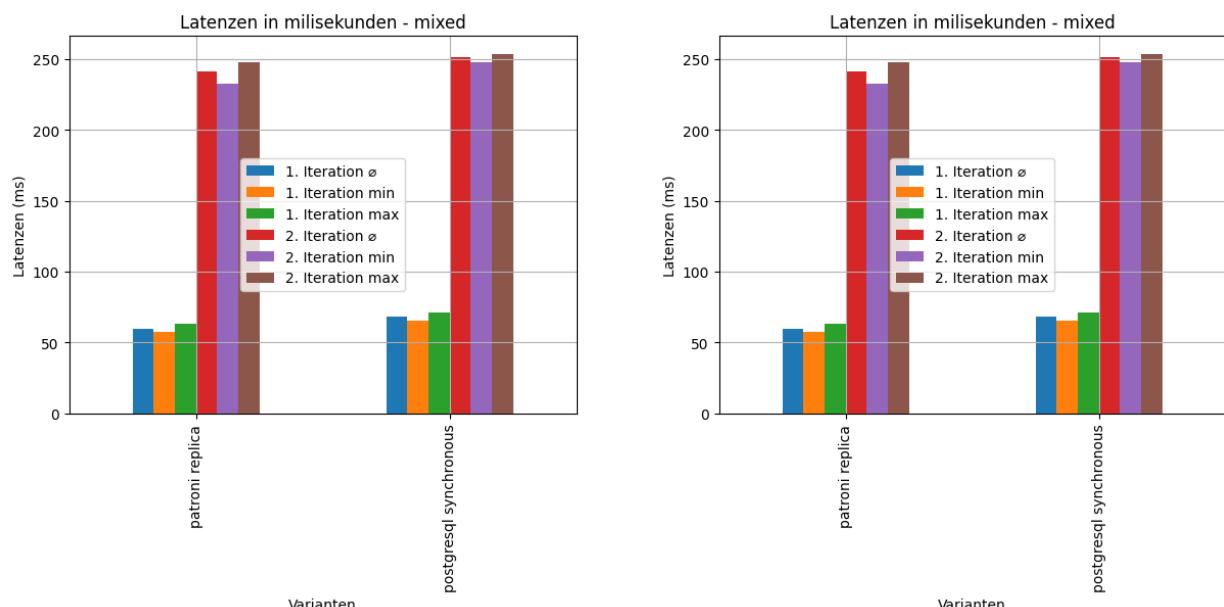


Abbildung 3.45: Benchmarks - latency Patroni Replica

Die asynchrone Replikation ist dabei ein klein wenig schneller als die synchrone Replikation.

Ein weiterer Benchmark sind die Fehler, die bei den DML-Transaktionen beim mixed-Benchmark auftreten können.

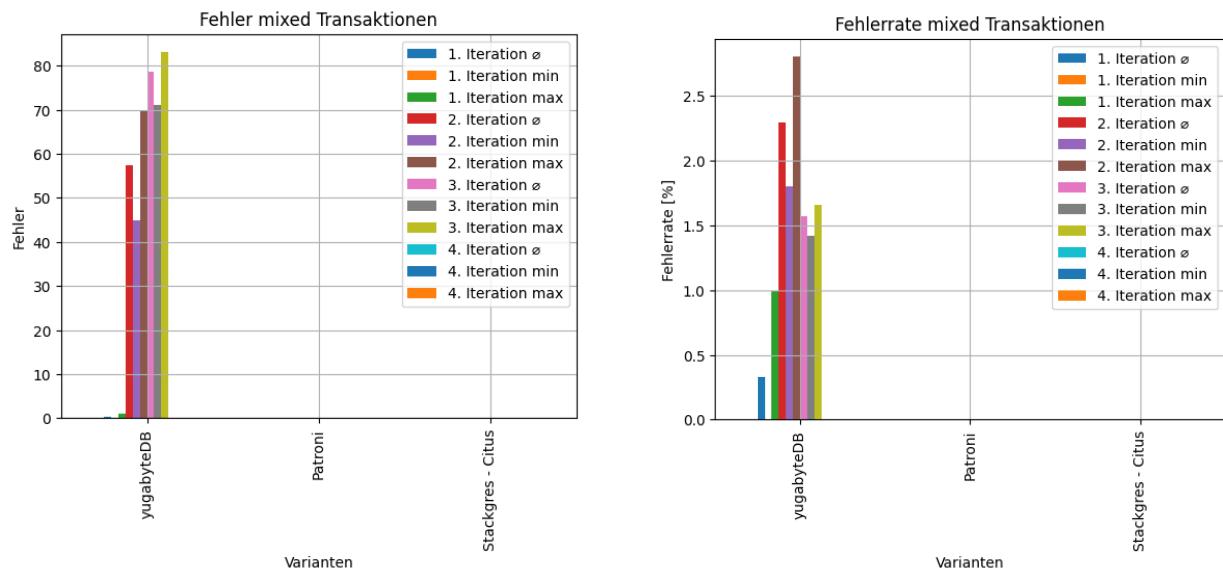


Abbildung 3.46: Benchmarks - Fehler bei mixed-Transaktionen

Ebenfalls ein wichtiger Benchmark ist die Zeit, die benötigt wird, um mit Hilfe von pgbench initialisiert die Tabellen zu erstellen.

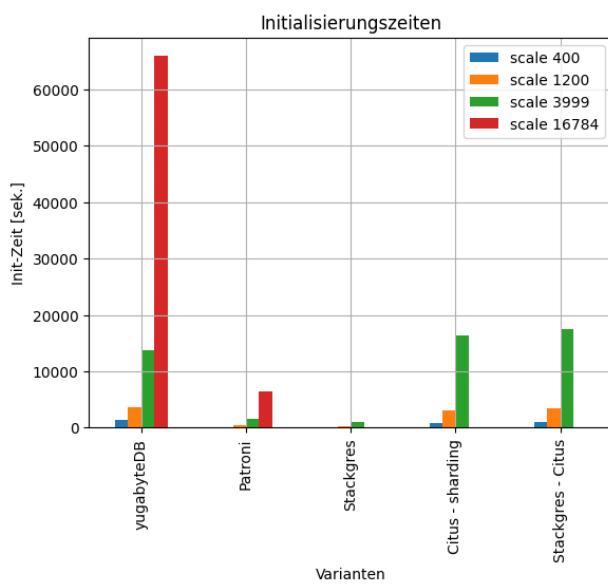


Abbildung 3.47: Benchmarks - Initialisierungszeit - sekunden

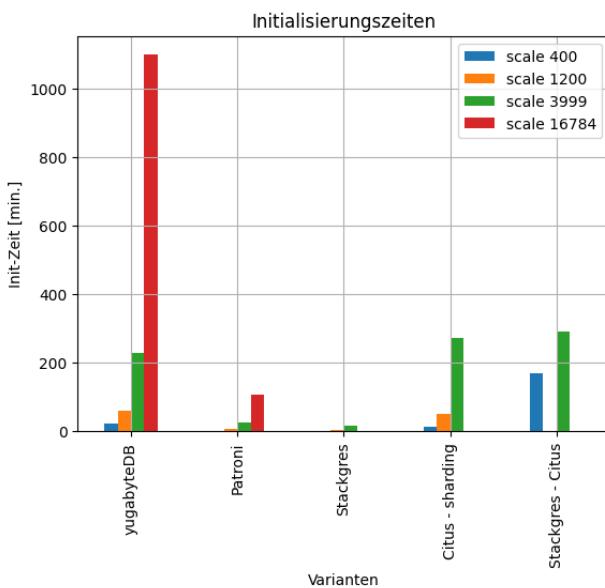


Abbildung 3.48: Benchmarks - Initialisierungszeit - Minuten

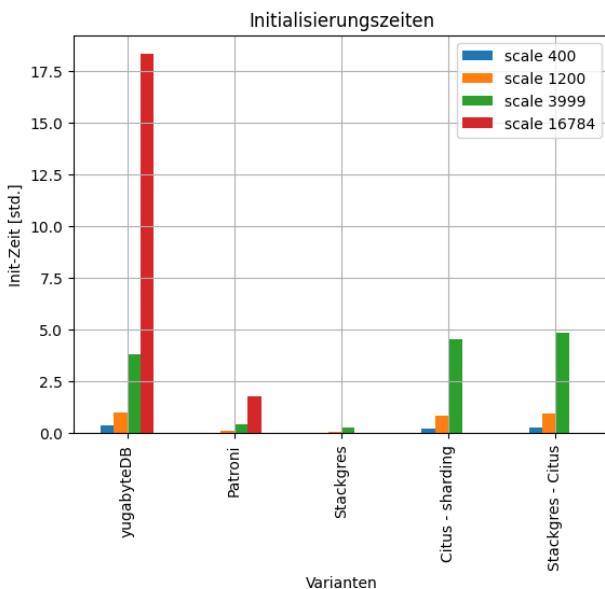


Abbildung 3.49: Benchmarks - Initialisierungszeit - Stunden

Dabei fällt auf, mit Patroni werden die Tabellen am schnellsten geladen. StackGres selber generiert ebenfalls wesentlich schneller als YugabyteDB. Werden dann aber die Tabellen in Shards aufgeteilt, verändert sich die Initialisierungszeit zuungunsten von StackGres - Citus.

3.1.9.2.1 Gegebene Parameter und Annahmen

Es wird mit fünf Jahren gerechnet.

Daher werden also die Zeitaufwände der Betriebstasks pro Jahr berechnet und mit fünf multipliziert.

Folgende getroffenen Annahmen und Parameter sind gesetzt:

Variable	Wert	Beschreibung / Begründung
Anzahl betrachtete Jahre	5	
Anzahl Switchovers pro Jahr	10	Alle zwei Monate wird am KSGR ein Reboot der Linux Server für das Patching vorgenommen
Anzahl Node Recoveries pro Jahr	5	Mindestens zweimal wird ein Failover Test gefahren. Mit drei weiteren Failovers wird gerechnet.
Anzahl Backup Restores pro Jahr	5	Mindestens vier Restore Tests müssen gefahren werden. Mit einem ungeplanten Restore wird gerechnet
Anzahl Quorum erweiterungen pro Jahr	1	Es wird mit nur einer erweiterung pro Jahr gerechnet.
Stundensatz ICT KSGR [CHF]	120	

Tabelle 3.16: Kostenberechnung - Annahmen

3.1.9.2.2 Varianten

Patroni wurde in zwei Varianten aufgeteilt.

Einmal die **Vanilla**-Version, die manuell aufgesetzt und verwaltet wird.

Also so wie es bei der Evaluation gemacht wurde.

Es gibt allerdings ein GitHub-Repository, welches die ganze Installation in Ansible-Playbooks verpackt hat.

Der ganze Prozess wurde analysiert und die Aufwände auch für diese Variante geschätzt.

An der Punkteverteilung ändert sich entsprechend nichts, da die Architektur die gleiche ist.

Diese Variante wird nachfolgend `Patroni` – `postgresql_cluster` oder vereinfacht nur `postgresql_cluster` genannt.

3.1.9.2.3 Zeitvergleiche

Die Zeiten wurden entsprechend den Erfahrungen mit den drei evaluierten Systemen geschätzt.

Phase	Subphase	Patroni - vanilla	Patroni - postgresql-cluster	StackGres - Citus	YugabyteDB
Initialer Aufwand					
	Basisinstallation	5.0	6.0	4.0	5.0
	Basiskonfiguration	5.0	5.0	5.0	5.0
	Backup Konfiguration	1.0	1.0	1.0	2.0
	Monitoring Konfiguration	2.0	2.0	2.0	2.0
Security Aufwand	private container registry Integration	0.0	0.0	1.0	1.0
	PKI Integration	3.0	3.0	3.0	3.0
Erweiterungsaufwand	Automatisierung Backup	1.0	1.0	4.0	4.0
	Automatisierung Skalierung	8.0	4.0	8.0	2.0
	Self-Healing	8.0	4.0	16.0	0.0
	Auto-Recovery	8.0	4.0	16.0	2.0
	DB Self-Service	16.0	16.0	16.0	16.0
Operationsaufwand / 5 Jahre	Switchover	50.0	25.0	50.0	0.0
	Node Recovery	50.0	25.0	100.0	25.0
	Backup Recovery	50.0	25.0	50.0	25.0
	Quorum erweitern	30.0	20.0	5.0	5.0
		237.0	141.0	281.0	97.0

Tabelle 3.17: Gemessene und Extrapolierte Aufwände Bsp.

Die Aufwände für die Betriebstasks wie das Erweitern des Quorums, das Wiederherstellen von Nodes und dem Recovery wurde wie folgt geschätzt:

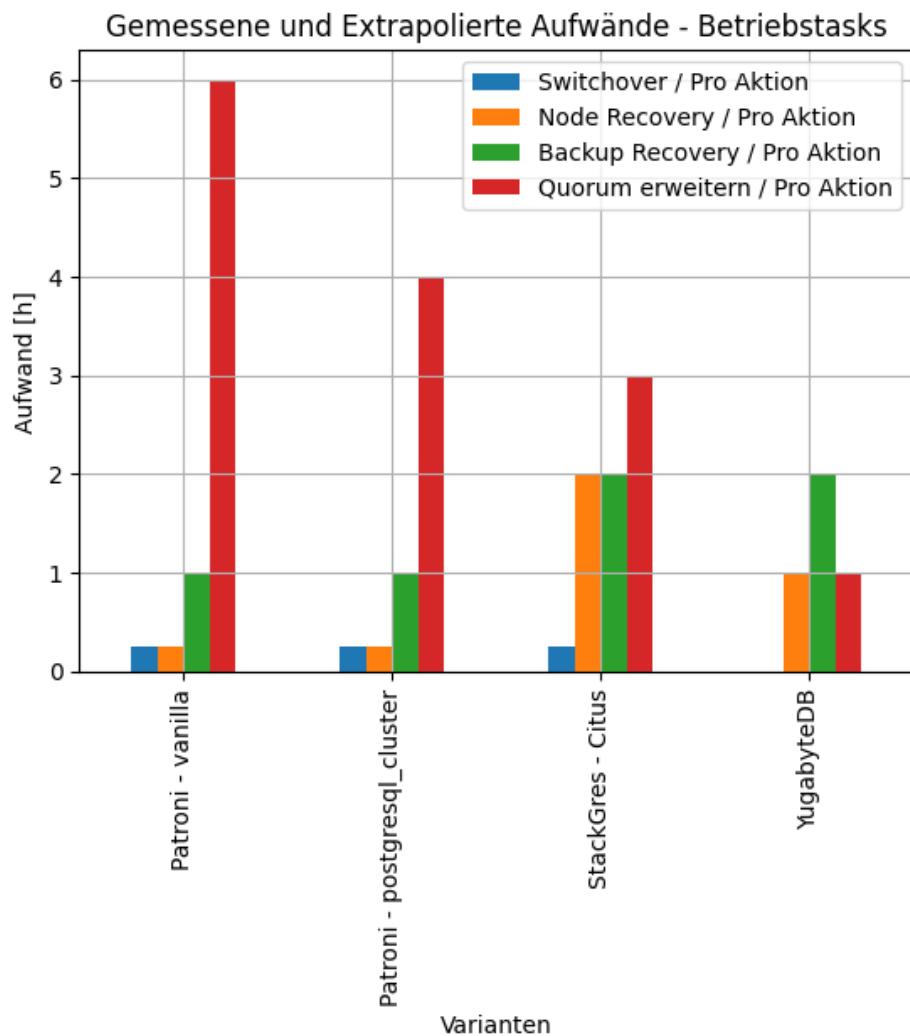


Abbildung 3.50: Zeitaufwände pro Betriebstask

Daraus ergeben sich folgende gesamten Zeitaufwände, wenn sie auf 5 Jahre extrapoliert werden:

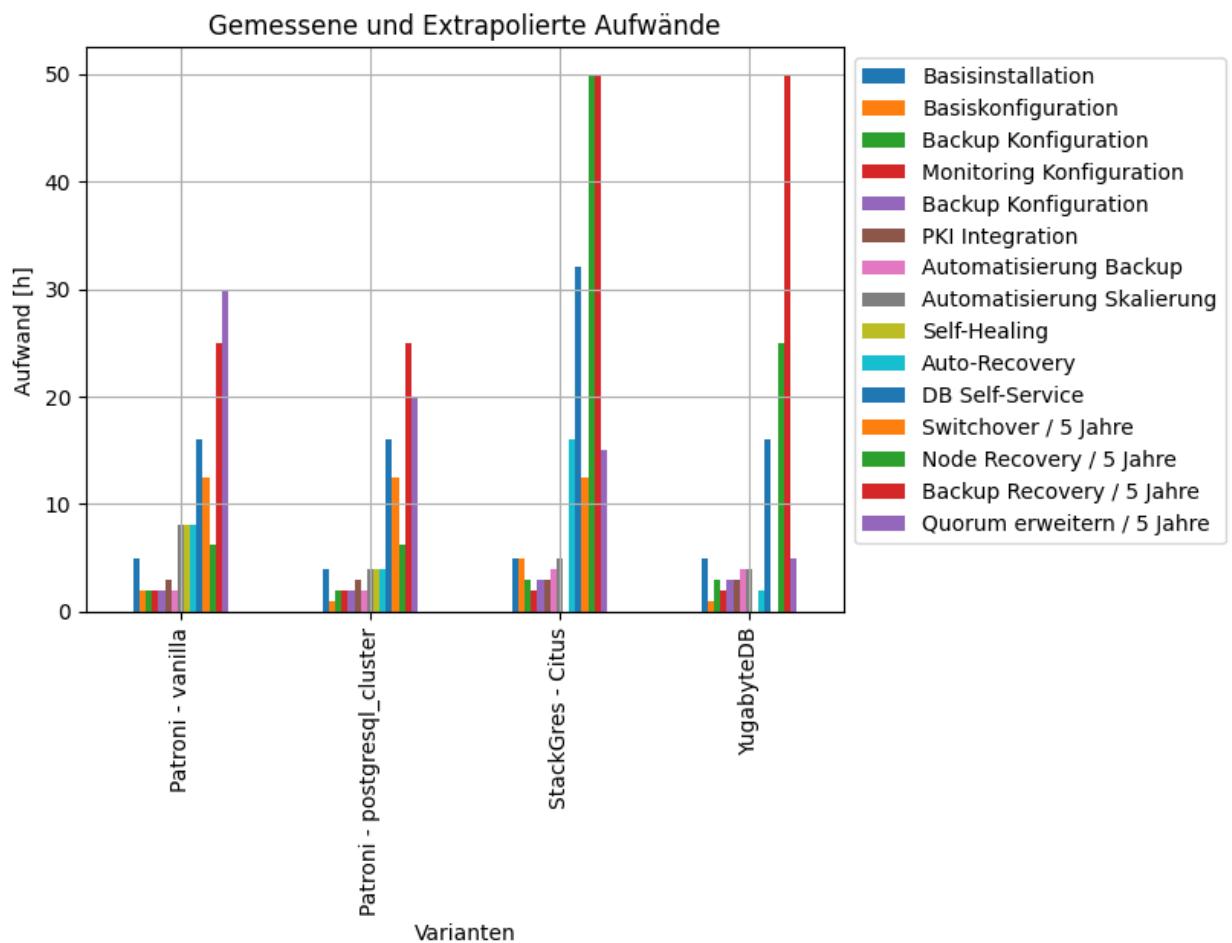


Abbildung 3.51: Zeitaufwände

In der Summe müssen für die jeweiligen Varianten also mit folgenden Aufwänden gerechnet werden:

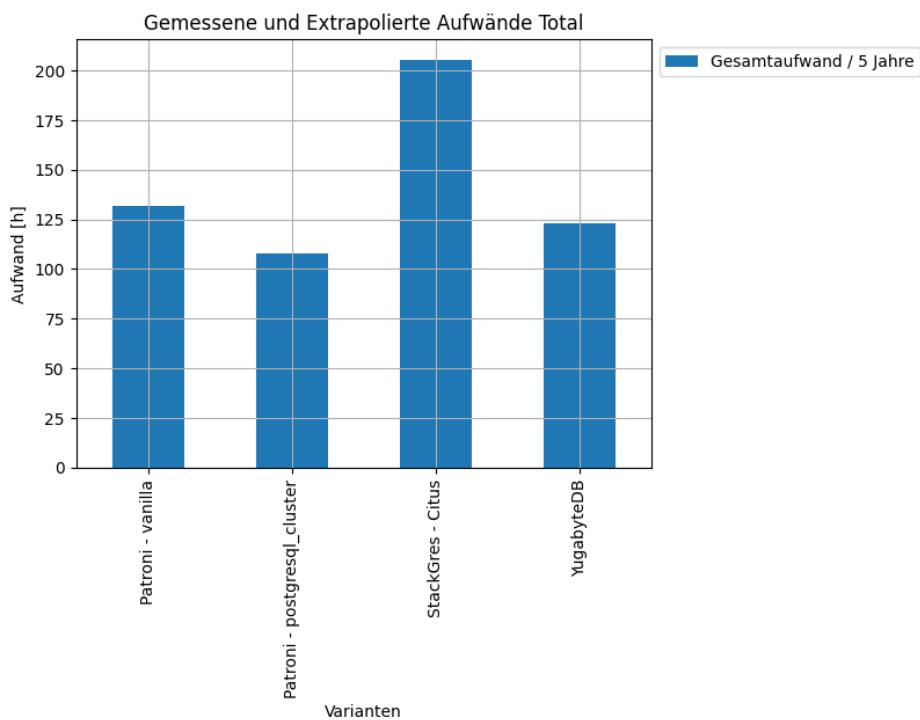


Abbildung 3.52: Zeitaufwände summiert

3.1.9.2.4 Kostenvergleiche

Die Kostenhochrechnung ist simpel.

Da der Stundenansatz in der ICT 120 Franken pro Stunde beträgt, wurden die Zeiten einfach mit 120 multipliziert.

Für die Betriebstasks wurden daher mit folgenden Kosten gerechnet:

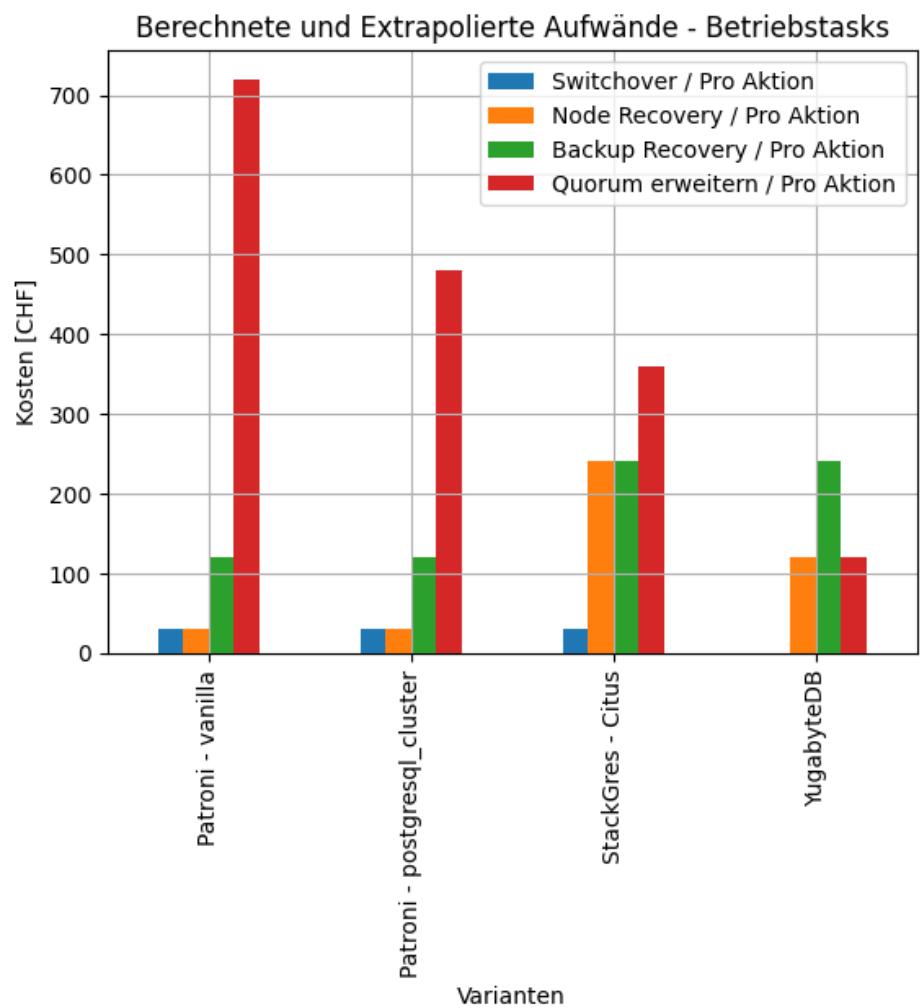


Abbildung 3.53: Kostenaufwände pro Betriebstask

Entsprechend ist auf fünf Jahre mit folgenden Kosten zu rechnen:

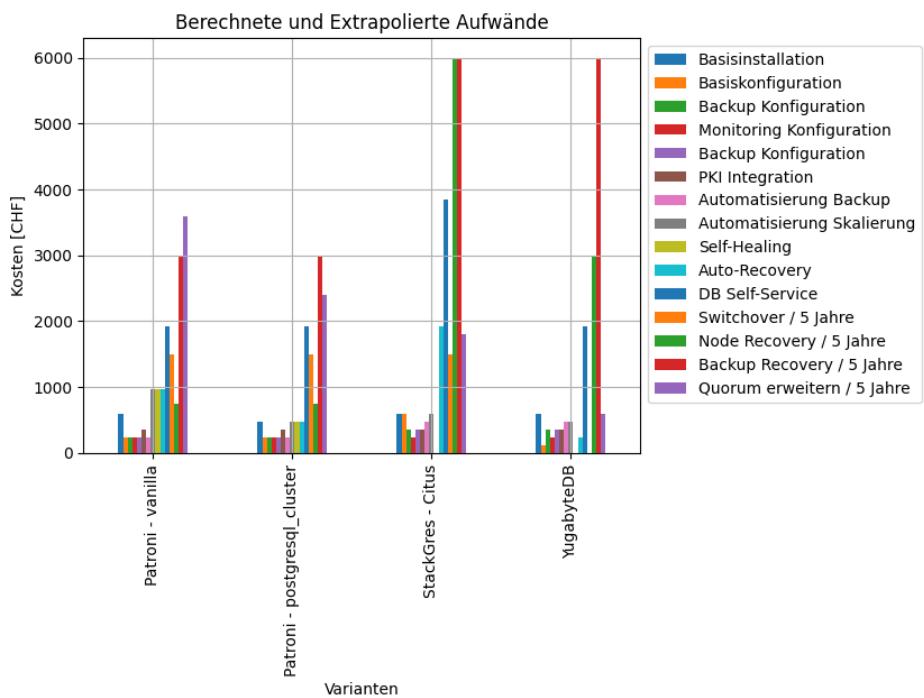


Abbildung 3.54: Kostenaufwände

Die totalen geschätzten Kosten würden sich im Vergleich wie folgt entwickeln:

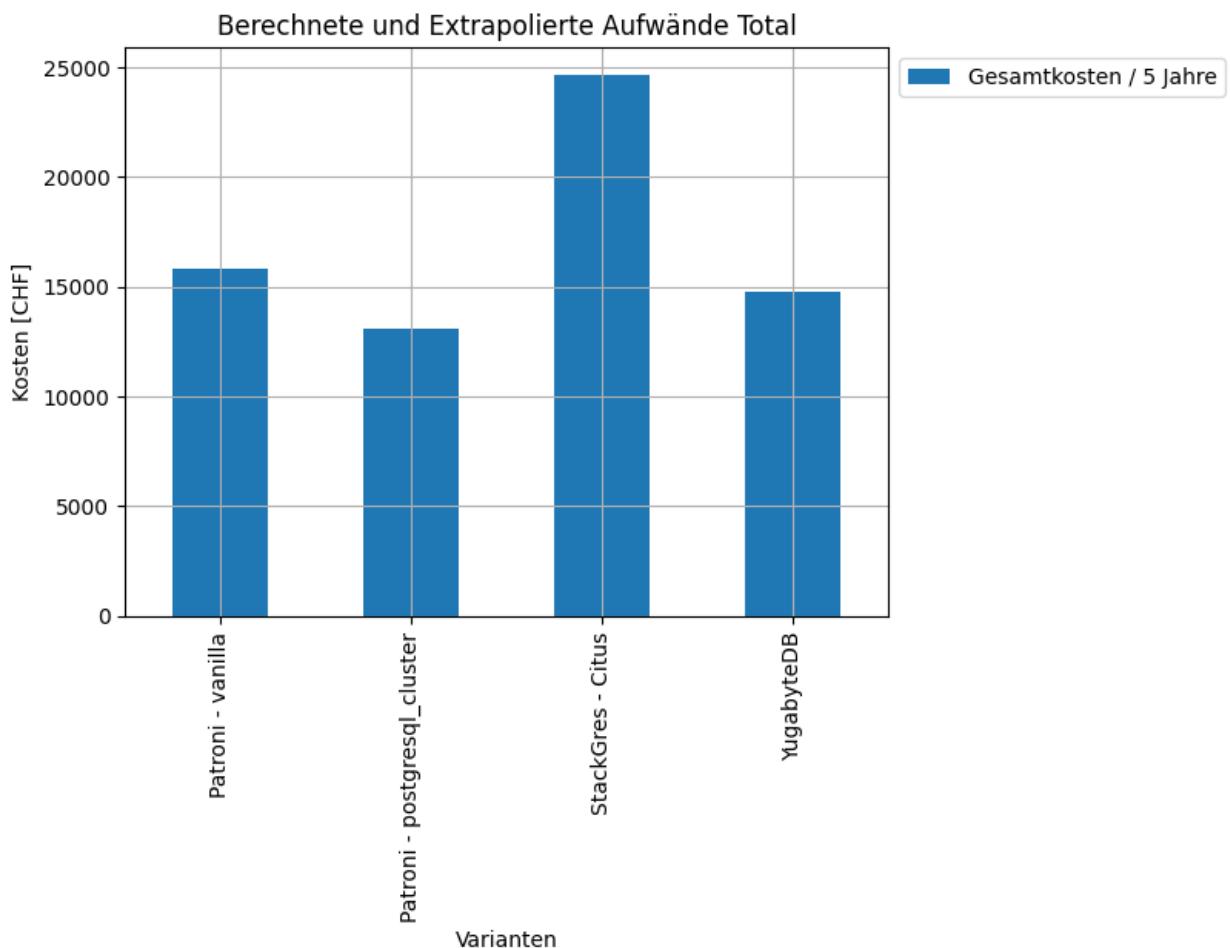


Abbildung 3.55: Kostenaufwände Total

3.1.9.2.5 **Schlussfolgerung**

3.1.9.3 **Kosten-Nutzen**

Patroni, in seinen beiden Ausprägungen als normales Patroni und mit dem postgresql_cluster-Ansible GitHub-Repository ist klar die beste Variante:

		Patroni - vanilla			YugabyteDB			StackGres - Citus			Patroni - postgresql_cluster		
Ziele / Kriterien	Gewicht	Notiz	Punkte	Produkt	Notiz	Punkte	Produkt	Notiz	Punkte	Produkt	Notiz	Punkte	Produkt
1 Systemvielfalt	125		3	375		3	375		3	375		3	375
2 Synergien	117	StackGres	3	350	Keine Synergien möglich	0	0	Patroni	3	350		3	350
3 Failover	108	Kein Pooler	2	217		3	325		1	108		2	217
4 Switchover	100	Kein Pooler	2	200		3	300		2	200		2	200
5 Restore	92		2	183		3	275		3	275		2	183
6 Replikation	83		3	250		3	250	Performance	2	167		3	250
7 Sharding	25	Kein Sharding möglich	0	0	Nur mit Zweck-entfremdung möglich	3	75	Kein Sharding möglich	0	0			
8 Quorum	67		3	200		3	200		3	200		3	200
9 Connection	58		3	175		3	175	Cluster / DB Passwort	2	117		3	175
10 Management-API	33		3	100		3	100		3	100		3	100
11 Backup	58		3	175	Eigenes Backup Tooling	1	58	Eigenes Backup Tooling	1	58		3	175
12 Housekeeping - Log Rotation	8		3	25	Klassisches Veeam eher schwierig	3	25	Klassisches Veeam eher schwierig	3	25		3	25
13 Self Healing	8		2	17		3	25		2	17		2	17
14 Monitoring - Node Failure	50		3	150		3	150		3	150		3	150
15 Maintenance Quality	8		3	25		1	8		2	17		3	25
16 Performance	58		3	175		2	117		1	58		3	175
				2442		2342			2208			2442	
		berechnete Felder	Rang 1			Rang 3			Rang 4			Rang 1	
		Zellbezüge											
		Eingabefelder											

Abbildung 3.56: Kosten-Nutzen-Analyse

Bei den Kosten zeigt sich, dass die Patroni-Variante `postgresql_cluster` klar am günstigsten kommt.

Dies resultiert auf den tiefen Installationskosten und der schnellen Erweiterungen.

Da, dass Repository schon einiges an Ansible-Playbooks mitbringt, ist auch der Erweiterungsaufwand gering.

YugabyteDB liegt auf Platz zwei, gefolgt vom klassischen (Vanilla) Patroni.

Auf dem letzten Platz landet StackGres - Citus, dies weil die Erstellung eines Clusters sehr viel Zeit benötigt.

	Patroni - vanilla	YugabyteDB	StackGres - Citus	Patroni - postgresql_cluster
1 Kosten	15'570.00	14'400.00	24'300.00	12'810.00
2 Nutzwert (Punkte)	2442	2342	2208	2442
	6.38	6.15	11.00	5.25
	Rang 3	Rang 2	Rang 4	Rang 1
<p>berechnete Felder</p> <p>Zellbezüge</p> <p>Eingabefelder</p>				

Abbildung 3.57: Kosten-Nutzen-Ranking

Daraus ergibt sich folgendes Kosten-Nutzen-Diagramm:

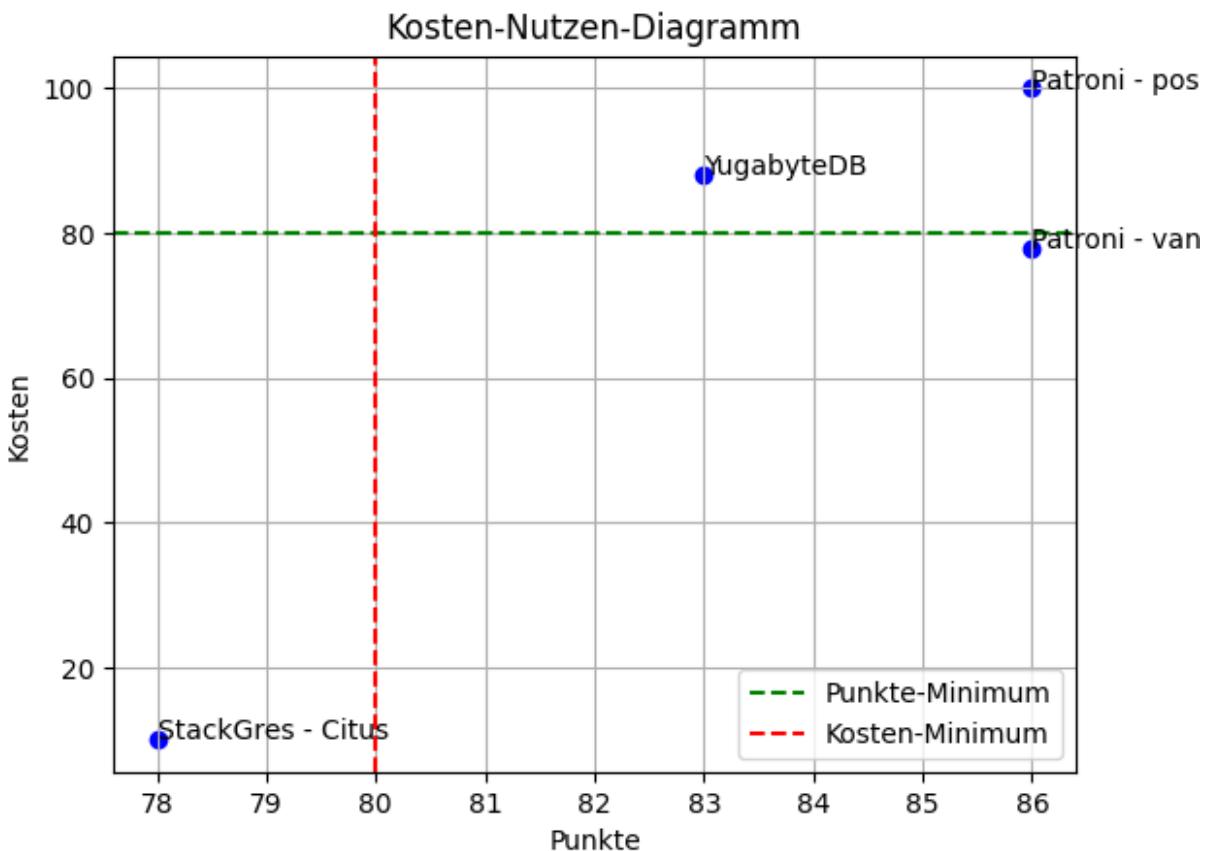


Abbildung 3.58: Kosten-Nutzen-Diagramm

StackGres - Citus scheidet direkt aus, da es nur 77% der geforderten Punktezahl erreicht.
Aber auch das klassische Patroni scheidet aus,
dies, weil die Kosten im Verhältnis zur besten Variante (postgresql_cluster) ebenfalls unter der 80% Marke liegt.

3.1.10 Entscheid

Anhand der Kosten-Nutzen-Analyse steht die Entscheidung fest.
Patroni wird mit der Variante postgresql_cluster als Testsystem aufgebaut.

Die Umsetzung der Cloud Native Lösung, in diesem Fall YugabyteDB, muss verschoben werden.
Dem Kubernetes-Testsystem des KSGR fehlt zum einen noch eine notwendige interne Komponente zur Freigabe,
zum anderen fehlen für den sauberen Betrieb die externen Systeme wie der PKI oder dass SIEM (welches aber in den nächsten Wochen eingeführt wird).

3.2 Aufbau und Implementation Testsystem

3.2.1 Architektur

Das Testsystem wird mit Patroni umgesetzt.

Dabei werden folgende Komponenten eingesetzt:

Aufgabe	System
Orchestrator	Patroni
Proxy	Haproxy
Connection Pooler	PgBouncer
DCS	etcd

Tabelle 3.18: Testsystem - Komponenten

Entsprechend sieht das Architektschema aus:

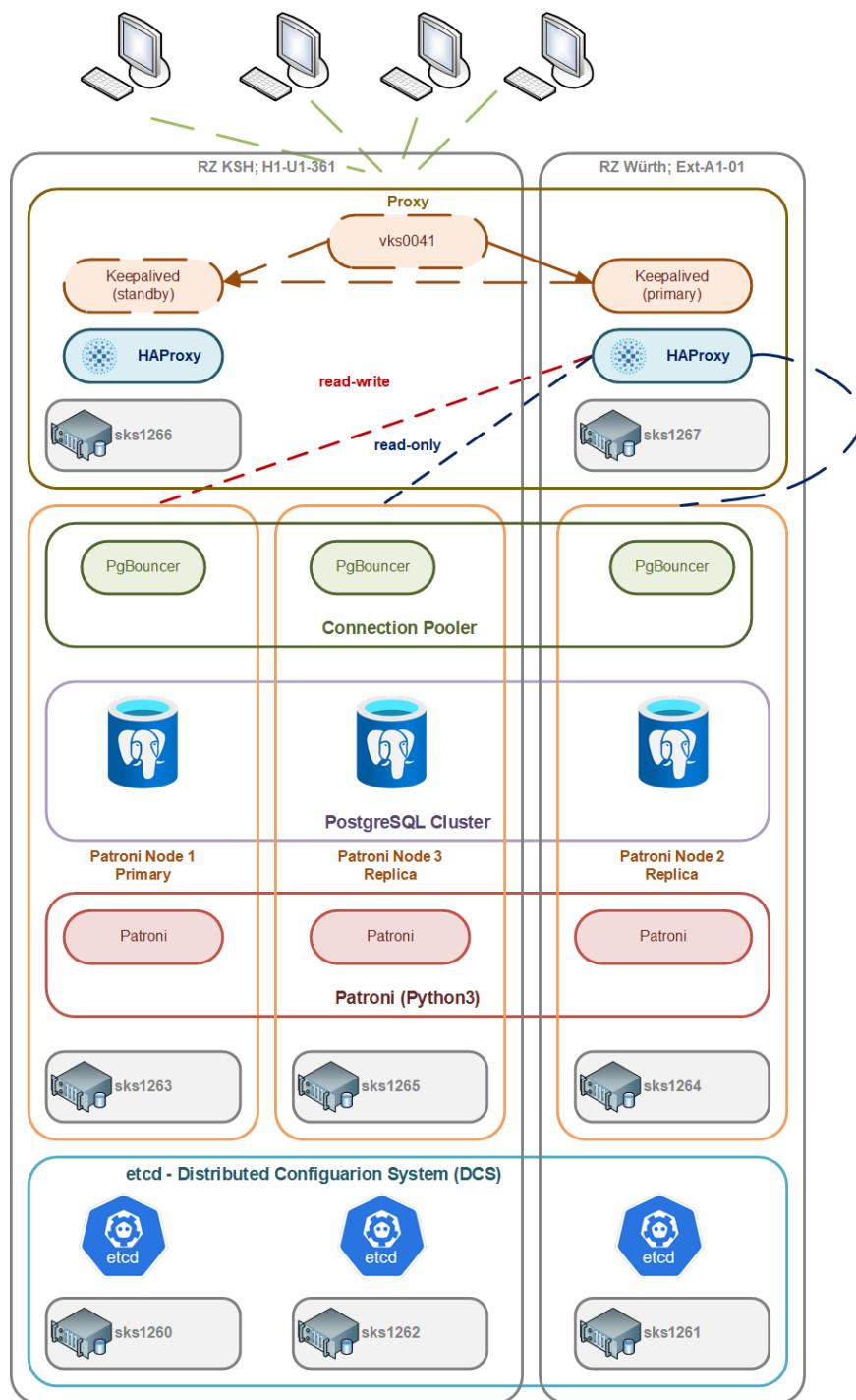


Abbildung 3.59: Testsystem - Architektur

vitabacks / postgresql_cluster hat eine Schwachstelle.

Der Connection Pooler ist nicht auf dem Level des Proxy, sondern auf dem Patroni Node.

Das Resultat bei einem Node Failure ist zwangsläufig, dass die Connections bei einem Switchover / Failover unterbrochen werden.

Sollte die Applikation nicht fähig sein, die Connection zu halten und innerhalb eines Time-outs reconnecten zu können,

kommt es zu einem Finalen Disconnect.

Ohne weiteres lässt sich der Connection Pooler aber nicht auf den Proxy-Server verschieben, zu tief müsste in den Code eingegriffen werden.

3.2.2 Bereitstellen der Grundinfrastruktur

Folgende Server wurden mit Hilfe von Foreman deployt:

Server	Typ	Funktion	Full Qualified Device Name	Domain	IP
sks1260	Monolith	etcd Node 1	sks1260.ksgr.ch	ksgr.ch	10.0.22.170
sks1261	Monolith	etcd Node 2	sks1261.ksgr.ch	ksgr.ch	10.0.22.171
sks1262	Monolith	etcd Node 3	sks1262.ksgr.ch	ksgr.ch	10.0.22.172
sks1263	Monolith	Database Node 1	sks1263.ksgr.ch	ksgr.ch	10.0.22.173
sks1264	Monolith	Database Node 2	sks1264.ksgr.ch	ksgr.ch	10.0.22.174
sks1265	Monolith	Database Node 3	sks1265.ksgr.ch	ksgr.ch	10.0.22.175
sks1266	Monolith	Pooler / Proxy Node 1	sks1266.ksgr.ch	ksgr.ch	10.0.22.176
sks1267	Monolith	Pooler / Proxy Node 2	sks1267.ksgr.ch	ksgr.ch	10.0.22.177
vks0041	Monolith	VirtualIP	vks0041.ksgr.ch	ksgr.ch	10.0.22.178

Tabelle 3.19: Testsystem - Inventar

Um die Erweiterungstests fahren zu können und überhaupt Ansible Playbooks ohne die Red Hat Ansible Automation Platform deployen zu können, musste Ansible auf einem separaten Host installiert werden.

Auch wurde ein Server benötigt, mit dem zuerst ein weiterer HAProxy und später Patroni Node hinzugefügt werden konnte.

Server	Funktion	Full Qualified Device Name	Domain	IP
sks1000	Ansible Host	sks1000.sivc.first-it.ch	sivc.first-it.ch	10.0.20.43
sks9016	HAProxy- und Patroni Node	sks9016.ksgr.ch	ksgr.ch	10.0.28.16

Tabelle 3.20: Testsystem - Auxiliar Inventar

3.2.3 Installation und Konfiguration PostgreSQL HA Cluster

vitabacks / postgresql_cluster GitHub Repository hat zwei zentrale Komponenten.

Das eine ist das Inventory-File, welches die IP-Adressen und Hostnamen der Server beinhaltet.

Dabei ist zu beachten, dass die Host-Einträge überschrieben werden, wenn `hostname=<hostname>` eingetragen wird.

Was entsprechend zu Problemen mit den DNS-Einträgen, dem Active Directory oder anderen Komponenten führen kann.

Die zweite zentrale Komponente ist das `main.yml`-File.

Über dieses YAML-File werden alle Aspekte des Patroni-Clusters einstellen.

Für dieses Testsystem zentrale Konfigurationsabschnitte waren:

Clustername

Der Clustename lautet «`k8s-core-psql`».

Dazu muss folgender Eintrag gesetzt werden: `patroni_cluster_name: k8s-core-psql`

DCS-Typ

Es stehen etcd oder Consul zur Auswahl, entsprechend wird etcd ausgewählt.

Es könnten auch bestehende etcd-Nodes verwendet werden.

Load Balancing

Grundsätzlich funktionierte der Cluster auch ohne Load Balancer.

Allerdings muss in diesem Fall HAProxy als Load Balancer eingesetzt werden.

Virtual IP

Auch kann definiert werden ob eine Virtuelle IP verwendet wird.

Ohne Load Balancer würde die virtuelle IP dem Primary Patroni Node zugewiesen.

Connection Pooler

`PgBouncer` kann aktiviert oder deaktiviert werden, entsprechend wurde dieser aktiviert.

Die Konfiguration wurde belassen.

Ein anderer Connection Pooler steht nicht zur Verfügung.

PostgreSQL - User

Alle notwendigen User können bereits definiert werden.

PostgreSQL - Datenbanken

Selbiges gilt für die Datenbanken.

Folgende Datenbanken wurden eingetragen:

- **`k8s_core_gitlab_prod`**

Leere DB Hülle für die GitLab Datenbank

- **`k8s_core_harbor_prod`**

Leere DB Hülle für die Harbor Datenbank

- **`k8s_core_keycloak_prod`**

Leere DB Hülle für die Keycloak Datenbank

- **gramic_test**

Diese DB wird nur für die Benchmarking-Tests und als Test-DB für die Maintenance-Skripte verwendet.

PostgreSQL - Extensions

Alle Datenbanken benötigen die Extension pgstattuple.

Diese wird verwendet, um Bloated Tables und Indices zu ermitteln.

PostgreSQL - pg_hba.conf

Ist eine IP oder ein Netzwerk nicht erfasst, kann trotz korrekten Zugangsdaten nicht auf die DB zugegriffen werden.

Entsprechend müssen die User und Client Adressen erfasst werden.

PostgreSQL - .pgpass

Im Passwortfile .pgpass werden die Passwörter zu den Usern erfasst.

Zusätzlich kann hier der Zugriff ebenfalls auf bestimmte Adressen eingeschränkt werden.

Als Alternative würde postgresql_pg_ident zur Auswahl stehen.

Patroni - REST-API

Die REST-API kann auf localhost beschränkt werden oder so eingestellt werden, dass die REST-AIP auf die Patroni Adresse oder den Hostnamen hört.

3.2.3.1 Vorbedingungen

Zuerst mussten auf allen Servern die entsprechenden Ports auf der Firewall freigegeben werden. Standardmäßig setzt vitabacks / postgresql_cluster folgende offenen Ports voraus:

Port	Beschreibung
2379	etcd
2380	etcd
5000	HAProxy - Read/Write - Primary
5001	HAProxy - Read Only - Replikas
5002	HAProxy - Read Only - Synchronous Replikas
5003	HAProxy - HAproxy Stats
5432	PostgreSQL
6432	PgBounder
8008	Patroni REST-API

Tabelle 3.21: Testsystem - Benötigte Ports

Es reicht nicht, auf den Servern einen Proxy einzustellen.

Entweder es werden Firewall-Rules erstellt, welche die Ziele von GitHub und den Debian-Repositories freigeben,
oder aber der Proxy wird dem `main.yml` mitgegeben.
Bei den Erweiterungstests ging es nicht mehr anders, der Server `sks9016` ist nur eine Spielwiese für das Foreman Provisioning.
Auch ist dieser Server nicht in der entsprechenden AD-Gruppe der Patroni Test Gruppe, also muss der alte Proxy verwendet werden.
Die Proxy Settings sind zuoberst, wird die Firewall angepasst, wird folgendens eingetragen:

```
1 proxy_env: {}  
2
```

Listing 3.29: main.xym - No Proxy

Der Gruppenproxy funktioniert mit folgenden Proxy-Settings:

```
1 proxy_env:  
2   http_proxy: "http://xksgr_vks0041_inet:<Password Secure / Safe>f@webproxy.sivc.  
      first-it.ch:9090"  
3   https_proxy: "https://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.  
      first-it.ch:9090"  
4
```

Listing 3.30: main.xym - Gruppenproxy

Der Proxy für alles andere benötigt folgenden Eintrag:

```
1 proxy_env:  
2   http_proxy: http://sproxy.sivc.first-it.ch:8080  
3   https_proxy: https://sproxy.sivc.first-it.ch:8080  
4
```

Listing 3.31: main.xym - SProxy

3.2.3.2 Installation / Deploy - `deploy_pgcluster.yml`

Die Installation ist simpel.

Nachdem alle Konfigurationen vollzogen sind, muss nur das entsprechende Playbook ausgeführt werden:

```
1 ansible-playbook deploy_pgcluster.yml  
2
```

Listing 3.32: Deploy - `deploy_pgcluster.yml`

Das reine Deployement dauerte jeweils etwas mehr als fünf Minuten.
Danach war ein voll funktionsfähiger Patroni Cluster betriebsbereit.

3.2.3.3 Maintenance - config_pgcluster.yml

Nachdem die Anpassungen am `main.yml` vorgenommen waren, musste nur das entsprechende Playbook ausgeführt werden:

```
1 ansible-playbook config_pgcluster.yml  
2
```

Listing 3.33: Maintenance - config_pgcluster.yml

☞ Die Nodes, um die der Cluster in den nächsten beiden Schritten erweitert wurde, durften noch nicht ins Inventory eingetragen werden, ansonsten wären Fehlermeldungen und ein Abbruch des Playbooks die Folge gewesen, da es die entsprechenden Nodes noch gar nicht gab.
Ebenso musste ein Proxy in das `main.yml` eingetragen werden, da der Testserver `sks9016` nicht der angepassten Firewall-Rule unterstand.

Zum Testen wurden nachträglich folgende Werte hinzugefügt:

- Der Proxy wurde gesetzt
- Der REST-API wurde die Patroni-Adresse mit Hilfe von `patroni_restapi_listen_addr: "{{ inventory_hostname }}"` hinzugefügt
- Das `pg_hba.conf` musste einmal um den `patroni_replication_username` Usereintrag auf die virtuelle Tabelle `replication` erweitert werden (damit dieser Node replizieren darf) als auch um den generellen Zugriff erweitert werden (damit HAProxy zugreifen darf).
- Zuletzt musste es auch möglich sein, einen Restart vollziehen zu können, wenn es notwendig wäre,
daher wurde `pending_restart: true` gesetzt.

3.2.3.4 HAProxy Node hinzufügen - add_balancer.yml

Um die Erweiterung des Proxy-Layers vorzunehmen, waren zwei Sachen notwendig.
Zuerst musste das Inventory um den Node erweitert werden, der entsprechend als neuer Node markiert wurde:

```
1 [balancers]  
2 10.0.22.176  
3 10.0.22.177  
4 10.0.20.43 new_node=true  
5
```

Listing 3.34: HAProxy Node erweitern - Inventory

Zum Schluss muss nur noch das Playbook ausgeführt werden:

```
1 ansible-playbook add_balancer.yml
```

Listing 3.35: HAProxy Node erweitern - add_balancer.yml

3.2.3.5 Patroni Node hinzufügen - add_pgnode.yml

Das Vorgehen ist gleich wie beim Erweitern von HAProxy.

Das Inventory wurde erweitert:

```
1 [master]
2 10.0.22.173 postgresql_exists=false
3
4 [replica]
5 10.0.22.174 postgresql_exists=false
6 10.0.22.175 postgresql_exists=false
7 10.0.28.16 postgresql_exists=false new_node=true
8
```

Listing 3.36: Patroni Node erweitern - Inventory

Das Deployment erfolgt nun ebenfalls via Playbook:

```
1 ansible-playbook add_pgnode.yml
```

Listing 3.37: Patroni Node erweitern - add_pgnode.yml

Die vollständige Dokumentation ist im [Anhang - Installation Testsystem](#) zu finden.

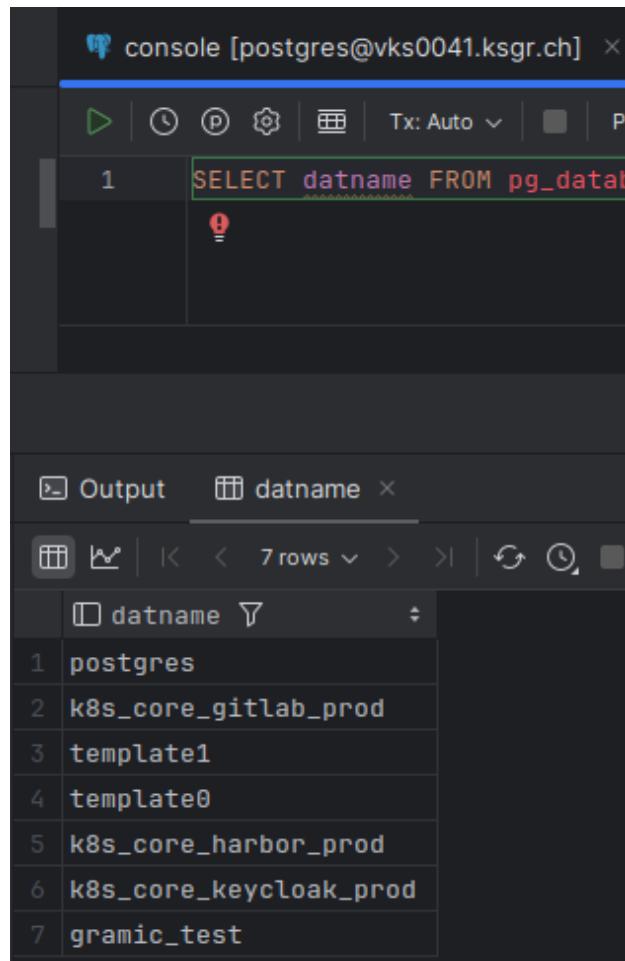
3.2.4 Technical Review der Umgebung

Die beiden Patroni-Nodes `sks1264` und `sks1265` sind synchrone Replikas und der `sks1263` ist der Primary:

Member	Host	Role	State	TL	Lag in MB
sks1263	10.0.22.173	Leader	running	1	0
sks1264	10.0.22.174	Sync Standby	streaming	1	0
sks1265	10.0.22.175	Sync Standby	streaming	1	0

Abbildung 3.60: Technical Review - Patroni Nodes

Alle Datenbanken, die übergeben wurden, sind erstellt worden:



The screenshot shows a PostgreSQL terminal window titled "console [postgres@vks0041.ksgr.ch]". In the command line area, the following SQL query is run:

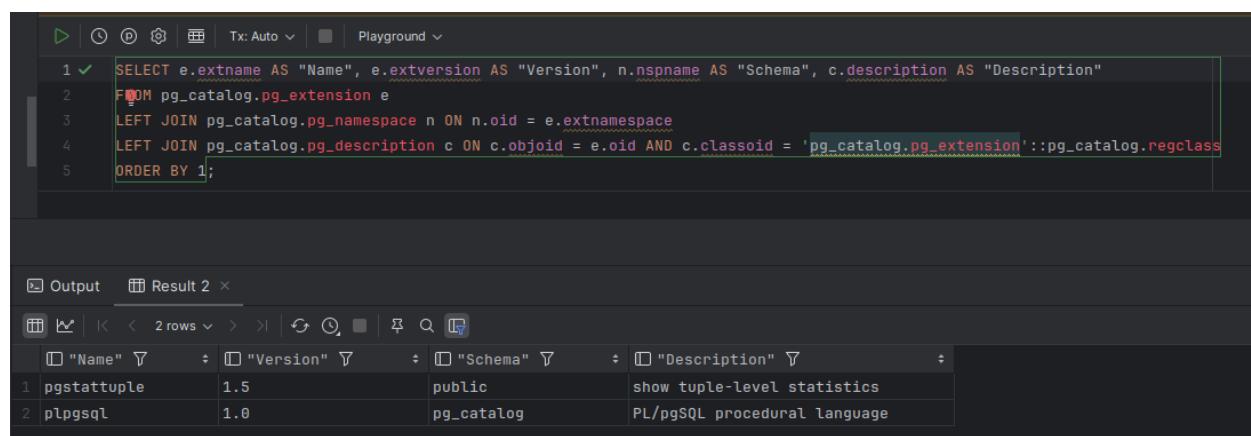
```
1 SELECT datname FROM pg_database
```

The result is displayed in the "Output" tab, showing a list of databases:

datname
postgres
k8s_core_gitlab_prod
template1
template0
k8s_core_harbor_prod
k8s_core_keycloak_prod
gramic_test

Abbildung 3.61: Technical Review - Datenbanken

Die Extensions wurden ebenfalls installiert:



The screenshot shows a PostgreSQL terminal window titled "Playground". A query is run to select information about installed extensions:

```
1 ✓ SELECT e.extname AS "Name", e.extversion AS "Version", n.nspname AS "Schema", c.description AS "Description"
2   FROM pg_catalog.pg_extension e
3   LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
4   LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
5 ORDER BY 1;
```

The result is displayed in the "Output" tab, showing two extensions:

Name	Version	Schema	Description
pgstattuple	1.5	public	show tuple-level statistics
plpgsql	1.0	pg_catalog	PL/pgSQL procedural language

Abbildung 3.62: Technical Review - PostgreSQL Extensions

Die Rollen und User wurden erstellt:

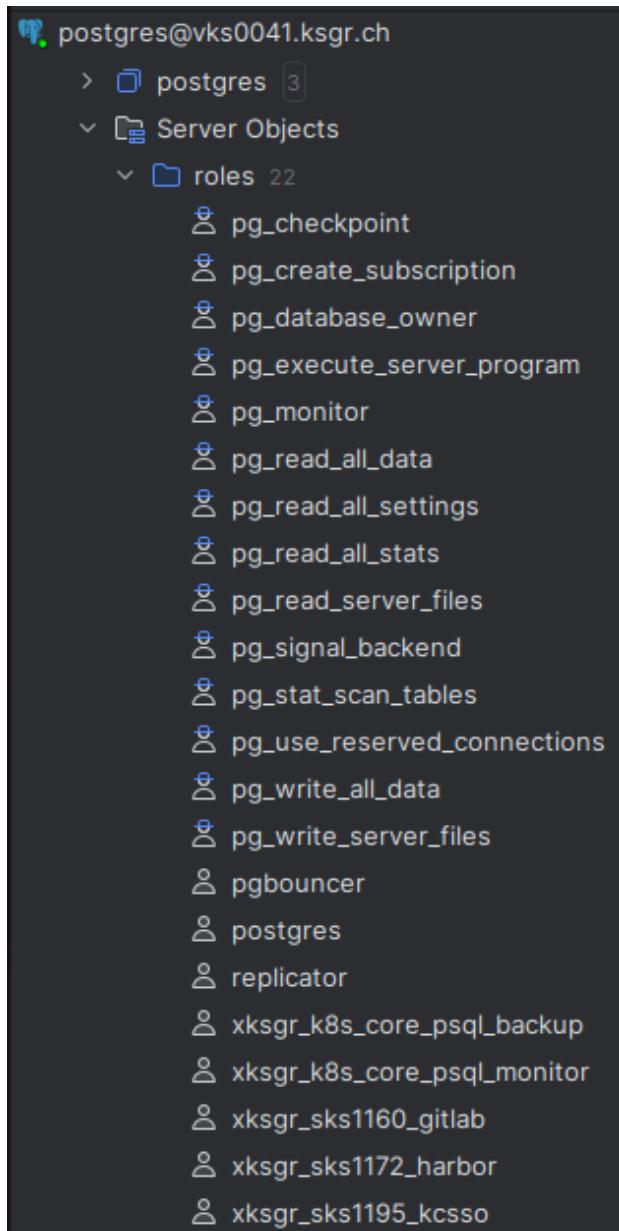


Abbildung 3.63: Technical Review - PostgreSQL User und Rollen

Die beiden HAProxy-Hosts `sks1266` und `sks1267` sind aktiv:



HAProxy version 2.6.12-1+deb12u1, released 2023/12/16

Statistics Report for pid 57563

> General process information

pid 57563 (process #1, rbrgid = 4)
system id = 270963
system maxmem = unlimited, ulimit = 200003
maxconn = 20000, maxconn1 = 20000, maxpipes = 0
current conn = 4, current pipes = 0/0; conn rate = 0/sec, br rate = 2.719 kbps
Running tasks: 0/04, idle = 200 ms
Note: "NOBACKUP" = UP with load-balancing disabled.

Display option: External resources:
 • Scope:
 • Hide TCP/HTTP servers
 • Hide health check
 • CSV export
 • JSON export (schema)

Stats												Health												Server			
Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			LastChk			Server			
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Rsp	Retr	Redis	Status	Act	Bck	Ck	Dm	Dmme	Thrle
Frontend	0	2	-	2	2	-	100 000	3	1 157	134 395	0	0	0	0	0	0	0	0	0	0	OPEN	0	0	0	0	0	0
Backend	0	0	-	0	0	-	10 000	0	0	0	0	0	0	0	0	0	0	0	0	0	2h20m UP	0	0	0	0	0	0

Health												Replicates												Server			
Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			LastChk			Server			
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Rsp	Retr	Redis	Status	Act	Bck	Ck	Dm	Dmme	Thrle
Frontend	0	1	-	0	1	-	1	4	2h15m	40 652	95 279	0	0	0	0	0	0	0	0	0	2h20m UP	0	0	0	0	0	-
sk1263	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	L7OK200 in 2ms	1/1	Y	-	0	0	Os -
sk1264	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	L7OK200 in 3ms	1/1	Y	-	1	1	2h20m -
sk1265	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	L7OK200 in 2ms	1/1	Y	-	2	2	2h20m -
Backend	0	0	-	0	1	-	1 000	4	2h15m	40 652	95 279	0	0	0	0	0	0	0	0	0	2h20m UP	0	0	0	0	0	-

Replicates												Replicates sync												Server			
Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			LastChk			Server			
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Rsp	Retr	Redis	Status	Act	Bck	Ck	Dm	Dmme	Thrle
Frontend	0	1	-	0	2	-	10 000	3	423	44 656	0	0	1	0	0	0	0	0	0	0	OPEN	0	0	0	0	0	-
sk1263	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h20m DOWN	1/1	Y	-	1	1	2h20m -
sk1264	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	L7OK200 in 2ms	1/1	Y	-	4	2	2h20m -
sk1265	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h20m DOWN	1/1	Y	-	1	1	314 -
Backend	0	0	-	0	0	-	1 000	0	0	7	0	0	0	0	0	0	0	0	0	0	2h20m UP	0	0	0	0	0	-

Abbildung 3.64: Technical Review - HAProxy sks1266

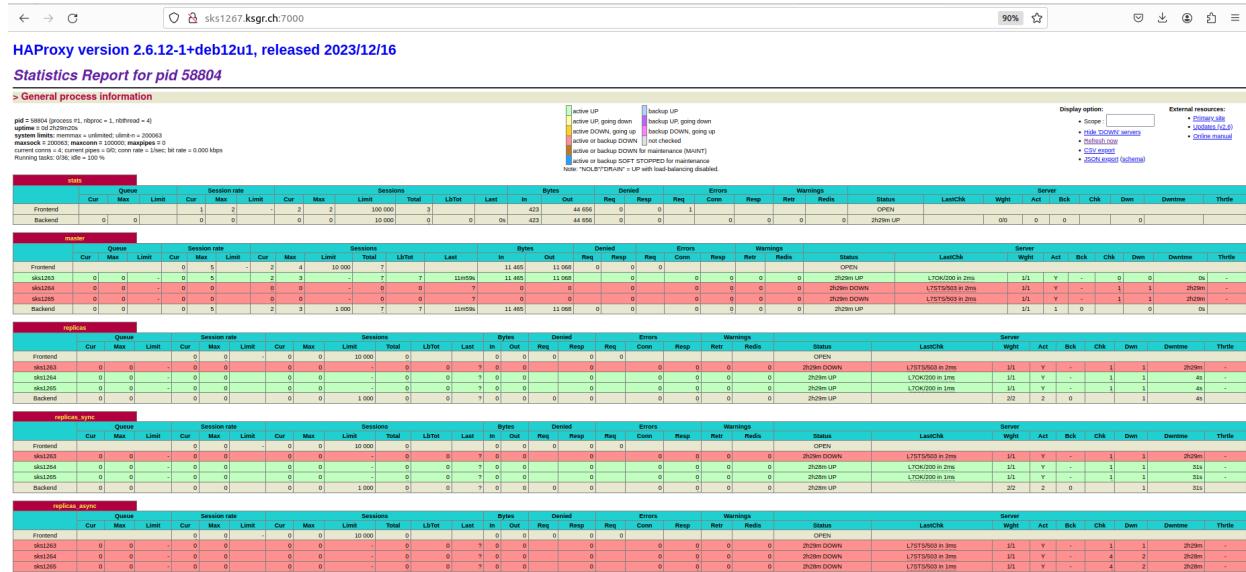


Abbildung 3.65: Technical Review - HAProxy sks1267

Die PgBouncer Konfiguration passt.

Alle via Ansible hinterlegten Datenbanken wurden erfasst:

d	name	host	port	database	force_user	pool_size	min_pool_size	reserve_pool	pool_mode	max_connections	current_connections	paused	disab
0	gramic_test	/var/run/postgresql	5432	gramic_test		20	0	1		1000	0	0	0
0	k8s_core_gitlab_prod	/var/run/postgresql	5432	k8s_core_gitlab_prod		20	0	1		1000	0	0	0
0	k8s_core_harbor_prod	/var/run/postgresql	5432	k8s_core_harbor_prod		20	0	1		1000	0	0	0
0	k8s_core_keycloak_prod	/var/run/postgresql	5432	k8s_core_keycloak_prod		20	0	1		1000	0	0	0
0	pgbouncer		6432	pgbouncer	pgbouncer	2	0	0	statement	1000	0	0	0
0	postgres	/var/run/postgresql	5432	postgres		20	0	1		1000	1	0	0

(6 rows)

Abbildung 3.66: Technical Review - PgBouncer - Datenbanken

Beim Auslesen der Stats und Datenbanken muss der richtige User gewählt werden.

So hat der User pgbouncer keine Berechtigung, lokal zuzugreifen.

Auch nicht auf die lokale Datenbank pgbouncer.

Hierfür muss der User postgres verwendet werden:

```
1 psql -p 6432 -h 127.0.0.1 -U postgres -d pgbouncer
```

Listing 3.38: PgBouncer - Connect

Die Abfrage der Tabellen wird dann mit Hilfe von folgendem Command gemacht:

```
1 show databases;
```

Listing 3.39: PgBouncer - Databases

Die Logs werden einmal pro Tag neu erfasst:

```
root@sks1265:~# ls -l /var/log/postgresql/
total 8
-rw-r--r-- 1 postgres postgres 601 May  8 22:02 pgbouncer.log
-rw-r----- 1 postgres postgres    0 May  8 22:03 postgresql-16-main.log
-rw----- 1 postgres postgres    0 May  9 00:00 postgresql-Thu.log
-rw----- 1 postgres postgres 2841 May  8 22:33 postgresql-Wed.log
```

Abbildung 3.67: Technical Review - PostgreSQL - Log Rotation

Die Einstellungen werden vorerst so belassen.

Sobald dass SIEM betriebsbereit ist, werden das Format und die Frequenz entsprechend den Anforderungen angepasst.

3.3 Testing

3.3.0.1 Testfälle

Basistests

1. Es wird eine Verbindung auf die postgres- und gramic_test DB auf dem Host vks0041.ksgr.ch oder IP 10.0.22.178 auf die Ports 5000 und 5001 hergestellt

2. Es werden Daten aus den Tabellen beide Datenbanken ausgelesen
3. Auf die DB `gramic_test` wird mit `pgbench` ein 5GiB-Init auf den Host `vks0041.ksgr.ch` oder IP `10.0.22.178` und Port `5000` ausgeführt

Failover

4. Der Server des primären Nodes wird manuell rebooted.

Switchover

5. Mit dem `patronictl`-Command wird der Swithcover gesetzt

Restore

6. Der Primary Node Server muss gestoppt werden, ein mixed `pgbench` abgesetzt werden und der Server wieder gestartet werden.
7. Mit dem `patronictl`-Command und Parameter `reinit` wird der Node wiederhergestellt und abschliessend mit Hilfe von Swithcover wieder als Primary gesetzt.
8. Mit dem `patronictl`-Command und Parameter `reinit` wird der Replica-Node wiederhergestellt
9. Vor, während und nach dem Restore müssen Tabellen mit Foreign-Key-Constraints und Daten geprüft werden.

Ansible - Deploy

10. Mit Ansible kann der Patroni Cluster deployed werden.

Ansible - Maintenance

11. Mit Ansible sollen folgende Parameter angepasst werden:
 - Das `pg_hba.conf` File.
So dass der HAProxy Node `10.0.28.16` erweitert werden kann.
 - Die Patroni REST-API soll für den jeweiligen Host von aussen ansprechbar sein.
Entsprechend muss der Eintrag gesetzt werden

Ansible - Patroni Node Extend

12. Mit Hilfe eines Ansible Playbooks kann ein Patroni Node angehängt werden.

Ansible - HAproxy Node Extend

13. Mit Hilfe eines Ansible Playbooks kann ein HAProxy Node angehängt werden.

- ☞ Die beiden Expansionsserver sind nicht in der vks0041 AD-Gruppe und werden auch nicht via Palo Alto ans Init7 Netz gerouted.
Entsprechend muss im `main.yml` der Proxy gesetzt werden.

3.3.1


Protokollierung

Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis	Begründung
Basistests	1	Connection	Verbindung auf vks0041.ksgr.ch oder 10.0.22.178 und Port 5000 und 5001 konnten ausgeführt werden.	Eingetroffen	
Basistests	2	SELECT	Tabellen können ausgelesen werden Tabellen werden erstellt und Daten geschrieben.	Eingetroffen	
Basistests	3	pgbench Init gramic_test	Keiner der Nodes fällt aus oder kann wegen eines zu grossen lags nicht mehr Synchronisiert werden.	Eingetroffen	
Failover	4	Automatismus	Wird der Primary Server vom Netz genommen, führt Patroni einen Failover auf einen Replika-Node	Eingetroffen	
Switchover	5	Skript / API	Mit dem Patroni Commandset wird er Switchover ausgeführt	Eingetroffen	
Restore	6	Automatismus	Ein gestoppter Node wird, wenn nicht zuviel Zeit vergangen ist, automatisch restored	Eingetroffen	
Restore	7	Skript / API	Mit dem Patroni Commandset der Primary-Node Wiederhergestellt	Eingetroffen	
Restore	8	Skript / API	Mit dem Patroni Commandset ein Replika-Node Wiederhergestellt	Eingetroffen	
Restore	9	Datensicherheit	Beim Restore des Primary-Nodes dürfen keine Daten, die seit dem Failover geschrieben wurden, darf es zu keinem Datenverlust kommen	Eingetroffen	
Ansible	10	Deploy	Der gesamte Cluster kann mit dem Playbook deploy_pgcluster.yml deployt werden	Eingetroffen	
Ansible	11	Maintenance	Das pg_hba.conf wurde um den Hosteintrag 10.0.28.16 wurde mittels Playbook config_pgcluster.yml erweitert. Die Patroni REST-API hört nun auf die Node-IP.	Eingetroffen	
Ansible	12	Patroni Node Extend	Der neue Patroni Node 10.0.28.16 wurde mit dem Playbook add_pgnode.yml in den Cluster k8s-core-psql integriert	Eingetroffen	
Ansible	13	HAProxy Node Extend	Der HAProxyNode lässt sich auf 10.28.16 mittels Playbook add_balancer.yml deploeyen.	Eingetroffen	

Tabelle 3.22: Testresultate Testsystem

Der Patroni Node wurde erfolgreich angefügt:

Cluster: k8s-core-psql (7366694452879094871)					
Member	Host	Role	State	TL	Lag in MB
sks1263	10.0.22.173	Leader	running	1	
sks1264	10.0.22.174	Sync Standby	streaming	1	0
sks1265	10.0.22.175	Sync Standby	streaming	1	0
sks9016	10.0.28.16	Replica	streaming	1	0

Abbildung 3.68: Testing - Patroni Node hinzufügen - add_pgnode.yml

3.3.2 Review und Auswertung

3.3.2.1 vitabaks/postgresql_cluster

vitabaks/postgresql_cluster ist sehr schnell und zuverlässig deployt.

Das Ansible-Repository hat allerdings eine Schwachstelle.

Der Connection Pooler ist auf dem Level der Patroni Nodes nicht auf dem Proxy-Level.

Dadurch verliert ein Client die Verbindung, wenn der entsprechende Server nicht mehr erreichbar ist.

Auch der HAProxy hat eine Schwäche.

Es handelt sich um einen Layer 7 Proxy, dadurch ist HAProxy nicht in der Lage, SQL Statements zu analysieren.

Wenn also die Read-Only Statements auf die Replikas geleitet werden sollen, muss die Applikation diese auf den entsprechenden Port zugreifen.

Ist die Applikation selber nicht in der Lage, die Lese- und Schreibprozesse an zwei verschiedene Ports zu leiten, geht der gesamte Load auf den Primary Node.

3.3.2.2 Maintenance-Tool - Bloating Tables

Aufgeblähte Tabellen werden zuverlässig erkannt und die Tabelle vakuumiert.

Zudem werden die Indices neu aufgebaut und somit geordnet.

3.3.2.3 Maintenance-Tool - AUTOVACUUM

Der Parameter autovacuum_vacuum_scale_factor wird zuverlässig berechnet und abgelegt.

3.4 Maintenance-Tool

3.4.0.1 Maintenance-Tool - Generell

Da das Kubernetes-Testsystem des KSGR noch nicht freigegeben ist, wurden die Maintenance-Jobs auf der Evaluationsplattform betrieben.

Dadurch ergaben sich ein paar Einschränkungen.

Ressourcen

Ressourcen wie Python-Skripte sollten in einer produktiven Umgebung z. B. in einem Helm Chart abgebildet werden.

Da die Erstellung eines helm Charts den Rahmen der Diplomarbeit sprengen würde, wurden Python-Skripte als ConfigMap deployt.

In der Produktiven Umgebung ist dies tunlichst zu vermeiden!

HashiCorp Vault / Secrets

Die Zugangsdaten (Username und Passwort) werden mit grosser Wahrscheinlichkeit in HashiCorp Vault[34] abgebildet.

In diesem Fall wird ein Secret deployt und das Secret als Filesystem gemountet.

Die grösse Schwäche hierbei ist, dass die Secrets unverschlüsselt auf dem Filesystem des Pods liegen, selbst wenn mit Base64 ein Hash erzeugt wurde.

Daher ist auch dieses Verfahren nicht für die Produktion geeignet.

Mailing

Mailing steht auf der Evaluationsumgebung nicht zur Verfügung.

Die Mainenance-Jobs sollen nach dem Microservice-Gedanken aufgebaut werden.

So soll die Automatisierbarkeit erhöht und flexibilisiert werden.

3.4.0.2 Maintenance-Tool - Bloated Tables / Indices

3.4.0.2.1 Ziel und Zweck

PostgreSQL hat mit AUTOVACUUM eine Funktionalität, welche Tabellen und Indizes aufräumt und bereinigt.

Allerdings gelten die Parameter für den ganzen PostgreSQL Cluster, einzelne Tabellen werden zwangsläufig nicht optimal vakuumiert.

Daher war ein Ziel, dass Bloated Tables und Indices von einem Maintenance Skript erkannt und bereinigt werden.

3.4.0.2.2 Funktionsweise

Auf allen Datenbanken muss die Extension pgstatttuple installiert werden.

Die Extension sammelt und stellt Informationen über Tabellen, Indizes, Tupels usw., zur Verfügung.

Mit Python werden alle Tabellen gelistet, bei denen pgstatttuple mehr als einen konfigurierten Wert zurückgeben.

Der Schwellwert liegt derzeit bei 1.5%.

Das SQL-Statement hierzu ist wie folgt aufgebaut:

```
1 select
2     relname as table,
3     (pgstattuple(oid)).dead_tuple_percent
4 from pg_class
5 where
```

```

6   relkind = 'r' and
7   (pgstattuple(oid)).dead_tuple_percent > %s
8 order by dead_tuple_percent desc;
9

```

Listing 3.40: Maintenance-Tool - List - Maintenance-Tool - Bloated Tables / Indices

Anschliessend werden die entsprechenden Tabellen mit Hilfe des VACUUM-Befehl manuell vakuumiert:

```

1 vacuum <table>;
2

```

Listing 3.41: Maintenance-Tool - VACUUM - Maintenance-Tool - Bloated Tables / Indices

 Ein VACUUM kann nicht in einer normalen Transaktion abgesetzt werden!

Bei psycopg2 muss der Transaktions-Isolations Level auf ISOLATION_LEVEL_AUTOCOMMIT gesetzt werden.

Anders lässt sich das SQL nicht absetzen.

Im Nachgang werden alle Indizes für die ganze Tabelle neu aufgebaut (REINDEX):

```

1 reindex table <table>;
2

```

Listing 3.42: Maintenance-Tool - REINDEX - Maintenance-Tool - Bloated Tables / Indices

Wichtig ist eine vorangehende Prüfung, ob ein AUTOVACUUM-Job am Laufen ist.

Trifft dies zu, darf das Skript in diesem Lauf nicht ausgeführt werden.

Überprüft kann dies mit folgendem SQL werden:

```

1 select count(*) as active from pg_stat_activity where query like 'autovacuum:%';
2

```

Listing 3.43: Maintenance-Tool - Check AUTOVACUUM - Maintenance-Tool - Bloated Tables / Indices

Dem Microservice-Gedanken entsprechend soll es pro Datenbank einen Job geben, der die Tabellen auf Bloated Tables untersucht und diese bereinigt.

Die vollständige Dokumentation ist im [Anhang - Maintenance-Tool - Bloated Tables](#) zu finden.

3.4.0.3 Maintenance-Tool - AUTOVACUUM

3.4.0.3.1 Ziel und Zweck

Folgende beiden AUTOVACUUM-Parameter sind entscheidend für die Definition, wann AUTOVACUUM startet:

autovacuum_vacuum_threshold

Mindestanzahl geänderter oder gelöschter Tupels die es in einer Tabelle braucht, damit AUTOVACUUM startet.

Der Default liegt bei 50 **toten** Tuples.

autovacuum_vacuum_scale_factor

Definiert, wie Prozent der Datensätze einer Datenbank geändert oder gelöscht werden müssen, bevor vakuumiert wird.

Die Grundformel, wann vakuumiert wird, sieht folgendermassen aus:

$$n_dead_tup > (pg_class.reltuples \times autovacuum_vacuum_scale_factor) + autovacuum_vacuum_threshold$$

Der autovacuum_vacuum_scale_factor über den gesamten PostgreSQL Cluster muss nun also mit der Zeit nachjustiert werden.

Dazu wurde obige Formel entsprechend umgestellt, da autovacuum_vacuum_threshold grundsätzlich statisch bleibt:

$$autovacuum_vacuum_scale_factor = \frac{(n_dead_tup_{max} - autovacuum_vacuum_threshold)}{pg_class.reltuples}$$

Die Aufgabe von diesem Maintenance-Skripts ist es also nun, einmal pro Tag den autovacuum_vacuum_scale_factor neu zu berechnen.

Da keine Mails versendet werden können, soll bei einem sich geänderten Parameter die generierte HTML-Tabelle auf das Verzeichnis des Kubernetes Nodes gespeichert werden.

Im **Produktiven** Betrieb soll die Tabelle als HTML in die Mail eingefügt werden.

3.4.0.3.2 Funktionsweise

In einem ersten Schritt werden alle benötigten Parameter aus der PostgreSQL-Konfiguration oder den entsprechenden Katalogtabellen ausgelesen.

Dies lässt sich mit einem SQL bewerkstelligen, welches wie folgt aufgebaut ist:

```
1 select
2   (
3     select
4       setting as autovacuum_vacuum_scale_factor
5     from pg_settings
6     where
7       name = 'autovacuum_vacuum_scale_factor'
8   ) as autovacuum_vacuum_scale_factor,
9   (
10    select
11      setting as autovacuum_vacuum_threshold
12    from pg_settings
13    where
14      name = 'autovacuum_vacuum_threshold'
15   ) as autovacuum_vacuum_threshold,
16   (
17    select
18      sum(reltuples) as reltuples
19    from pg_class
20   ) as reltuples,
21   (
22    select
23      sender_host
24    from pg_stat_wal_receiver
```

```

25 ) as sender_host
26

```

Listing 3.44: Maintenance-Tool - Parameter - Maintenance-Tool - AUTOVACUUM

Anschliessend wird ein Python Dictionary erstellt, welches wiederum in ein  pandas DataFrame umgewandelt wird.

Mit der Methode `to_html()` wird die Tabelle dann simpel auf das gemountete Host Filesystem geschrieben.

Die Tabelle sieht wie folgt aus:

	Bestehend	Berechnet
autovacuum_vacuum_scale_factor	0.01	0.02

Abbildung 3.69: AUTOVACUUM - Berechneter autovacuum_vacuum_scale_factor

Die vollständige Dokumentation ist im [Anhang - Maintenance-Tool - Maintenance-Tool - AUTOVACUUM](#) zu finden.

3.5 Monitoring

Das Monitoring im PRTG besteht aus zwei Teilen.

Zum einen werden die Standard-Werte wie Ping, CPU Load, Load Average, Disk Free, Traffic und die Uptime:

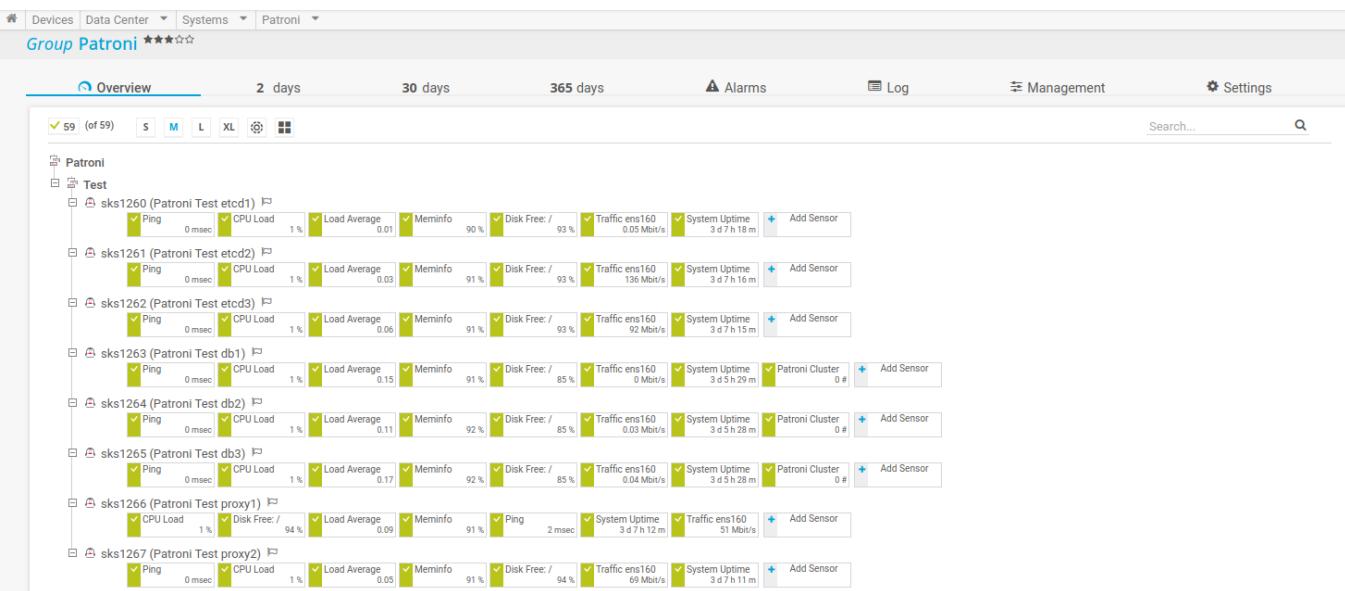


Abbildung 3.70: PRTG - Patroni Monitoring

Zusätzlich wurde ein Python Custom Sensor erstellt.

Dieser nutzt die Patroni REST-API um folgende Werte abzufragen:

Cluster Status

Prüft, ob der Cluster noch im Status running ist.
Dabei wird geprüft, ob der Cluster «running» ist.
Ist der Cluster pausiert, wird eine Warnung ausgegeben.
Andernfalls wird ein Fehler geworfen.

Cluster Unlock

Ein Patroni Node kann nur Primary werden, wenn im DCS (in diesem Fall etcd) der Leader-Lock gesetzt werden kann.
In diesem Fall wird ein Fehler geworfen.

Failsafe Mode Activity

Zusätzlich kann der «failsafe_mode» aktiviert werden.
Dann kann PostgreSQL trotzdem als Primary Node laufen[23].

Replication Lag

Prüft, ob das Replication Lag unterhalb dem Übergebenen Wert liegt.
Es gibt eine Abfrage für die Warngrenze und eine Fehlgrenze.

Replication State

Liest die States der Replikas aus.
Dabei wird eine Warnung ausgegeben, wenn nicht alle Nodes im «sync»Mode sind.
Ein Fehler wird ausgegeben, wenn keiner der Patroni Nodes in diesem State ist.

Die Konfiguration wird im PRTG gemacht.

Dabei müssen im Feld «Additional Parameters» folgende Parameter übergeben werden:

replication_mode

Der Replikationsmodus.
Für diesen Cluster ist es «sync» für synchrone Replikation.

lag_warning

Wenn das Replication Lag 2GB übersteigt, wird eine Warnung ausgeworfen.

lag_error

Wenn das Replication Lag 4GB übersteigt, wird ein Fehler.

Die Werte werden Kommasepariert übergeben:

`replication_mode=sync,lag_warning=2GB,lag_error=4GB`

Abbildung 3.71: PRTG - Patroni Cluster Sensor Setting

Der Sensor sieht im PRTG dann so aus:

Channel	ID	Last Value	Minimum	Maximum
CLUSTER UNLOCK	3	0 #	0 #	0 #
Downtime	-4			
FAILSAFE MODE ACTIVITY	4	0 #	0 #	0 #
REPLICATION LAG	6	0 #	0 #	1 #
REPLICATION STATE	5	0 #	0 #	0 #
sks1263	2	0 #	0 #	0 #

Abbildung 3.72: PRTG - Patroni Cluster Sensor

Die vollständige Dokumentation ist im [Anhang - Monitoring - PRTG](#) zu finden.

3.6 Troubleshooting und Lösungsfindung

Folgende Fehler sind während der Evaluation und der Installation des Testsystems aufgetreten:

Evaluation - Yugaware

Erst wurde das lizenpflichtige Yugaware-Repository verwendet.
Ohne Lizenzkey lässt sich Yugaware nicht installieren.
Die Lösung bestand darin,
auf das Open-Source YugabyteDB-Repository zu wechseln.

Evaluation - etcd für Patroni

Erst wurde versucht, drei etcd-Hosts auf Patroni zu installieren.
Dies führte zu einem Hostnamenskonflikt.
So wurde etcd auf den Standalone Server `sks9016` installiert.

Evaluation - MetalLB

Trotz Load Balancing mit MetalLB war es nicht möglich, von aussen auf YugabyteDB zuzugreifen.
Die Kommunikation wurde nicht von rke2 von Linux an den Load Balancer weitergegeben.
Die Lösung bestand darin, ein sogenanntes L2Advertisement auf den Adress-Pool und
Namespace zu setzen

Evaluation - local-path-provisioner

Alle Persistence Volume Claims wurden auf einen Node gesetzt.
Solange die Volumes nicht zu gross wurden,
war das System lauffähig.
Bei zu grossen Volumes kam es zu einem Fehler weil die Disk in einen Overflow lief.
Die Lösung besteht darin, im `nodePathMap` des `local-path-provisioner` Manifests jeden Node zu
spezifizieren.
Beim StorageClass-Manifest muss eine `nodeAffinity` auf die Nodes gesetzt werden.

Evaluation - StackGres Proxy für Extension

StackGres holt sich die PostgreSQL Extensions aus einem GitHub Repository.
Da die Kommunikation über einen Proxy geht, reicht der rke2 containerd-Proxy
(`CONTAINERD_HTTPS_PROXY` / `CONTAINERD_HTTP_PROXY` / `CONTAINERD_NO_PROXY`) nicht mehr.
Die URL des GitHub Repositories muss um den Proxy erweitert werden.
Dies löst das Problem aber nur, wenn die Zertifikate auf dem Server installiert sind.
Andernfalls muss die Zertifikat-Prüfung umgangen werden, indem die Kommunikation mit Hilfe von
`http` erzwungen wird.
Da es sich nur um eine Evaluationsplattform handelte, kam letztere Lösung zum Einsatz.

Evaluation - Benchmarking automation

Ursprünglich war der Plan, dass die Benchmarks mit Hilfe von `pgbench` resp. `ysql_bench` mit Hilfe
von CronJob ausgeführt wird.
Die beiden Benchmarking-Tools haben allerdings beide kein Passwort-Parameter.
Auch fand sich keine gängige Lösung, das Passwort via Bypass mitgeben zu können.
Um nicht zu viel Zeit darin zu verlieren, wurden die Benchmarks von Hand gemacht.

Testsystem - Proxy

Ansible konnte via Python nicht auf Externe Adressen zugreifen.
Erst wurde manuell versucht, etcd zu installieren.
Es gab zwei Lösungswege.

- Im Ansible `vars/main.yml` konnten Proxy-Settings definiert werden.
Dann mussten trotzdem noch die apt-proxy Settings gesetzt werden.
- Das Netzwerkteam änderte die Zugriffspfade der Server um.
Diese griffen nun, wenn man den Proxy ausschaltet, nicht mehr auf den Proxy zu, sondern die Palo-Firewall.
Die dortigen Rules ermöglichen einen ausgehenden Verkehr.
Diese Variante wird in Zukunft für Kubernetes Standard.

4 Resultate

4.1 Zielüberprüfung

Nr.	Ziel	Priorität	Ziel Umgesetzt / Erreicht	Begründung / Beschreibung
1	Evaluation	Hoch	Ziel erreicht	<p>Es wurden drei Methoden genauer betrachtet und installiert</p> <p>Am Ende der Evaluation wird ein System ausgewählt, welches als Testsystem installiert wurde.</p> <p>Das Testsystem wurde mit vitabaks/postgresql_cluster, einem Ansible-Playbook für Patroni, zuverlässig und stabil deployt.</p>
2	Testsystem	Hoch	Ziel erreicht	<p>Der Cluster lässt sich schnell und mit wenig Aufwand erweitern.</p> <p>Patroni ist zusammen mit HAProxy in der Lage,</p>
3	Automatisierter Failover	Hoch	Ziel zum grossteil erreicht	<p>den Failover Timeout tief zu halten. So Tief, dass eine entsprechend konfigurierte Applikation die Verbindung aufrecht erhalten könnte.</p> <p>Da vitabaks/postgresql_cluster den Connection Pooler nicht auf dem GlSIEM Layer installiert sondern auf dem Patroni Layer, ist der Cluster selber nicht in der Lage die Connections in jedem Fall aufrecht zu erhalten.</p>
4	Automatisierter Failover Restore	Hoch	Ziel erreicht	<p>Patroni fährt einen Node, der ausgefallen ist, selbstständig nach</p>
5	Monitoring - Cluster Healthcheck	Mittel	Ziel erreicht	<p>PRTG überwacht die Standard Vitalparameter wie CPU Load, Load Average, Memory Usage, Disk Usage, Network Traffic und die Uptime.</p> <p>Zusätzlich wurde ein separater Sensor erstellt, der den Cluster Status, die Gesundheit des gesamten Systems und das Replication Lag überwacht.</p>
6	AUTOVACUUM - Parameter verwalten	Mittel	Ziel erreicht	<p>Mit einem Kubernetes CronJob wird täglich der Parameter <code>autovacuum_vacuum_scale_factor</code> berechnet.</p> <p>Es fehlt zurzeit das konkrete Wissen über den PostgreSQL Explain Plan, also jenes Hilfsmittel mit dem SQL Statements analysiert werden.</p>
7	SQL optimierungen - Indizes tracken und verwalten	Mittel	Nicht umgesetzt	<p>Ohne dieses Wissen lässt sich schwer ermitteln, ob es einen Index auf einer Tabelle benötigt.</p> <p>Nicht benötigte Indizes können zwar ermittelt werden, allerdings kann sich der Ausführungsplan eines SQL Statements nach dem löschen stark verändern.</p> <p>Das Risiko einen Index ohne Explaining Plan zu löschen, ist zu gross.</p>
8	Maintenance - Indizes säubern	Hoch	Ziel erreicht	<p>Aufgeblähte Tabellen werden erkannt und vakuumiert.</p> <p>Anschliessend werden die Indizes einer vakuumierten Tabelle neu aufgebaut.</p>
9	Housekeeping - Log Rotation	Hoch	Ziel erreicht	<p>vitabaks/postgresql_cluster und Patroni selbst bieten Konfigurationsmöglichkeiten für Log Rotationen.</p> <p>Da die Umsetzung des GlSIEM Projekts erst gestartet ist, wurden nur rudimentäre Einstellungen vorgenommen.</p>
10	User Management - Monitoring	Tief	Nicht umgesetzt	Ziel wurde nicht verfolgt da es nur eine Tiefe Priorität hatte.
11	Evaluationsziel	Hoch	Ziel erreicht	Mit Patroni wurde eine Variante evaluiert und mit vitabaks/postgresql_cluster auch die Installationsmethode.
12	Installationsziel	Hoch	Ziel erreicht	vitabaks/postgresql_cluster erreicht Ziel 1 und 2
13	Testziele	Hoch	Ziel zum grossteil erreicht	<p>Das erste Teilziel lässt sich in einem Quorum-System schlicht nicht umsetzen.</p> <p>Die restlichen Teilziele wurden erreicht.</p>

Tabelle 4.1: Result - Zielüberprüfung

4.1.1 SQL Optimierungen - Indizes tracken und verwalten

Um Indizes effektiv setzen zu können, muss der Ausführungsplan, der Explaining Plan eines SQL Statements analysiert werden.

EXPLAIN kann allerdings nur ausgeführt werden, wenn ein Statement angehängt wird[5].

Bevor also ein Explaining Plane analysiert werden kann, muss dieser erst generiert werden.

Dazu müssen die Statements gespeichert werden und abschliessend das Query ausgeführt werden.

EXPLAIN bietet weit mehr Informationen.

Wie der EXPLAIN für das automatische Setzen von Indizes benutzt werden kann, benötigt viel Analyse und ein Konzept.

Es lassen sich zwar nicht benutzte Indizes auslesen und auch löschen.

Aus Erfahrung mit Oracle Database zeigte sich aber, dass das Löschen eines Index einen gesamten Ausführungsplan umwerfen kann.

Ohne eine Analyse mit EXPLAIN ist das Risiko für eine solche Aktion zu gross.

Alles in allem bräuchte es für dieses Ziel ein Performance Warehouse.

Also eine Datenbank, die EXPLAIN Resultate, Systemparameter und weitere Metriken speichert.

Nur so lassen sich SQL Statements sauber analysieren und optimieren.

Dieses Unterfangen wäre ein eigenes Thema für eine Diplomarbeit.

4.1.2 Testziele

Das erste Teilziel lautete «Der PostgreSQL Cluster muss immer lauffähig sein, solange noch ein Node up ist, unabhängig davon, welche Nodes des PostgreSQL HA Clusters down ist», ist mit einem Quorum System nicht machbar resp. ein Quorum System verhindert genau das.

Dieses Ziel ist daher nicht umsetzbar und macht retrospektiv keinen Sinn mehr.

4.2 Schlussfolgerung

4.2.1 Kubernetes / rke2

Kubernetes mit rke2 lässt sich vergleichsweise schnell installieren.

Damit die Containerplattform voll funktionsfähig sein soll, braucht es noch einige Module.

Mit MetalLB kann ein Load Balancer, den die Applikationen nutzen können, deployt werden.

Wenn nicht, wie zukünftig beim KSGR ein Storagesystem wie Rook / CephFS ein Storage System aufgebaut werden soll,

kann durch local-path-provisioner das Host-Storagesystem den Containern präsentiert werden.

Für private Zwecke oder als Evaluationssystem kann so rasch ein System aufgebaut werden.

Für Test- oder Produktivsysteme ist dieses Setting allerdings eher weniger geeignet.

4.2.2 YugabyteDB

YugabyteDB bietet auch in der Open-Source Variante einiges.

Besonders die effiziente Speicherung selbst grosser Datenmengen hat in einer Zukunft, wo immer mehr Applikationen in die Cloud verschoben werden und Datenbank Storage sehr teuer wird, einen Trumpf im Ärmel.

YugabyteDB stellt als Distributed SQL System hohe Anforderungen an ein synchrones Zeitsystem.

Auch ist YugabyteDB keine aktuelle Umsetzung von PostgreSQL, für Applikationen die auf eine aktuelle PostgreSQL-Version angewiesen ist, ist YugabyteDB daher nicht geeignet.

4.2.3 StackGres - Citus

Ongres hat mit StackGres den Patroni Stack in eine CloudNative Umgebung gegossen.

Das pro Datenbank ein eigener Cluster deployt wird, sorgt für ein Maximum an Isolation.

Durch diesen Architekturentscheid steigt aber auch der ohnehin grosse Ressourcenverbrauch des gesamten Systems nochmals stark an.

Problematisch ist auch, dass einige Komponenten in einer Blackbox liegen.

Als während des Benchmarking bei der Evaluation das Sharding immer wieder an der gleichen Stelle scheiterte,

war zwar ersichtlich, dass es zu einem Java Fehler gekommen war, doch genauere Informationen liessen sich den Logs nicht entnehmen.

StackGres bietet eine direkte Integration von Citus, im Benchmarking stiess diese Infrastruktur an ihre Grenzen.

Das Sharding-System Citus wiederum hat seine Stärken klar bei lesenden Zugriffen.

Das alte Citus Sharding, bei dem Tabellen **Distributed** wurden, kann nur mit Eingriffen in die Statements der Applikation effizient genutzt werden.

Das Schema Based Sharding bietet hier einen Vorteil, doch beide Sharding-Systeme können Tabellen nicht uneingeschränkt Sharden.

Wurden zwischen Tabellen Foreign-Key Constraints gesetzt, können diese entweder nicht mehr **Distributed** werden oder aber beide müssen im gleichen Schema liegen.

Es können zwar Referenced Tables, Tabellen die auf allen Shards liegen, erzeugt werden, doch diese Methode ist ineffizient beim Schreibprozess.

Zudem rät Citus Data selbst, Referenced Tables nur für Referenzierte Tabellen, die für alle Distributionen und Schemas bereitstehen sollen, zu verwenden.

Und dann auch nur, wenn diese Tabellen nicht zu gross werden.

4.2.4 Patroni

Patroni mit HAProxy bietet die Möglichkeit, einen stabilen monolithischen PostgreSQL HA Cluster aufzubauen.

Basis bleibt dabei ein PostgreSQL Cluster, der von Patroni orchestriert und verwaltet wird.

Patroni benötigt allerdings Zusatztools wie einen Connection Pooler oder einen Proxy sowie z. B. etcd als DCS.

Der Umfang für einen Patroni Cluster steigt selbst bei einem minimalen Setting mit drei PostgreSQL Cluster Nodes auf vier bis fünf Server.

4.2.5 vitabacks / postgresql_cluster

Das Ansible Repository bietet eine ganze Reihe von Playbooks, mit dem ein Patroni Cluster deployt, erweitert und verwaltet werden kann.

Die Dokumentation des GitHub Repositories ist qualitativ hochwertig, ein Cluster ist schnell deployt. Weitere Schritte wie das Härten des gesamten Systems ist ebenfalls bereits angedacht und vorbereitet.

Der grosse Schwachpunkt ist zurzeit die Architektur.

Der Connection Pooler wird auf dem Patroni Node installiert, geht die Verbindung zu diesem verloren, ist die Applikation **Disconnected**.

So wird der Connection Pooler seiner Funktion beraubt.

HAProxy ist zudem ein Layer 7 Proxy, der Cluster ist darauf angewiesen, dass die Applikation Lese- und Schreibanfragen trennen kann und an zwei verschiedene Ports sendet.

Im Moment laufen aber Bestrebungen, mit der Software PgCat sowohl den Proxy als auch den Connection Pooler zu vereinen.

PgCat vereint beides in einer Open-Source-Software und ist in der Lage, die Statements zu analysieren und lesenden Verkehr auf die Replikas zu verteilen.

4.3 Weiteres Vorgehen / offene Arbeiten

Nach dem Abschluss der Diplomarbeit müssen noch einige notwendige Schritte vorgenommen werden. Die anschliessenden Arbeiten werden nach Zeitraum der Umsetzung und Dauer sortiert:

SIEM

Das SIEM in Form von Elastic wurde installiert.

Daher wird einer der nächsten Schritte sein, dass Logging des Testsystems an das SIEM anzubinden.

Backup

Das Testsystem muss in die Backup-Infrastruktur eingebunden werden.

PKI

Die Ausschreibungsphase für den PKI wurde zwar abgeschlossen, doch das System ist noch nicht betriebsbereit.

Sobald das System steht und ein Konzept vorliegt, muss das Testsystem integriert werden.

Hardening

Mit dem PKI müssen die Daten auf dem Storage als auch der Traffic verschlüsselt werden.

KSGR Ansible

Wenn das Hardening und Monitoring umgesetzt sind, muss das vitabacks / postgresql_cluster Repository in den KSGR Red Hat Ansible Automation Platform eingebunden werden.

Dazu muss ggf. ein Fork des GitHub Repositories vorgenommen werden.

Maintenance-Tool

Das Kubernetes Test- und Produktivsystem des KSGR wurde noch nicht freigegeben.
Sobald dies der Fall ist, muss das Maintenance-Tool sauber implementiert werden.
Secrets müssen mit HashiCorp Vault[34] angebunden, das Logging dem KSGR Standard angeglichen werden.

Produktivsystem

Wenn SIEM, Backup, PKI, das Hardening und das Ansible Repository umgesetzt wurde, muss das Produktivsystem installiert werden.

YugabyteDB

Als zweites evaluiertes System, muss YugabyteDB als Test- und Produktivsystems aufgebaut werden.

Performance Warehouse für PostgreSQL

Als Langzeitarbeit soll ein Performance Warehouse aufgebaut werden.
Eine vergleichbare Plattform existiert bereits für Oracle Database.

4.4 Persönliches Fazit

Aus heutiger Sicht würde ich die Ziele weiter reduzieren.

Die Evaluation alleine hat wesentlich mehr Zeit in Anspruch genommen als geplant.

Heute würde ich auf die Umsetzung eines Testsystems verzichten und stattdessen ein oder zwei Systeme mehr evaluieren und installieren.

Generell habe ich unterschätzt, wie tief der Dive in die Konzepte und Architekturen gehen wird.

Den gesamten Zeitplan, aber auch die Dokumentenstruktur würde ich heute anders aufbauen.

Die Evaluation hätte wesentlich mehr Raum und auch den Maintenance-Tools **stunde** eigene Arbeitspakete zur Verfügung.

Das Projektmanagement hat keine eigenen Arbeitspakete erhalten und musste daher immer an anderer Stelle abgebucht werden.

Ein weiterer Punkt, den ich anders machen würde, gerade mit einer etwas ausgedehnteren Evaluationsphase **wäre** mehr Zeit für das Projektmanagement vorhanden gewesen.

Die Diplomarbeit zwang mich, aus meiner Komfortzone herauszutreten.

Ich habe ungleich mehr DevOps und System Engineering Arbeiten als DBA gemacht, von denen gab es schlussendlich nur eine Handvoll.

Erkenntnisse und Lehren habe ich auf zwei Perspektiven gewonnen.

Einmal aus der technischen Sicht.

So konnte ich mit rke2 von Grund auf ein Kubernetes System aufbauen und mit local-path-provisioner sowie MetalLB Sinnvoll erweitern.

Zudem durfte ich tiefe Einblicke in neue und zukunftsfähige Konzepte wie den Distributed SQL Systemen gewinnen.

Als DBA gehörte ich eher zu einer konservativen IT-Sparte, die nach wie vor stark auf **Monolithischer** Systeme setzt und neuen Ansätzen eher misstrauisch gegenübersteht.

Durch die Diplomarbeit habe ich meinen Horizont um verteilte Systeme und dem CloudNative Ansatz erweitert.

Kommt hinzu, dass es für die Evaluation sehr reizvoll war, ein falsch konfiguriertes System einfach zu löschen und angepasst neu zu deployen.

Bei Patroni, dass ein monolithisches System war, brauchte es immer einen Snapshot Restore.

Auf der anderen Seite habe ich auch aus der Sicht der Herangehensweise eine wichtige Lektion gelernt. Beim Aufbau von rke2 sowie dem Umgang mit local-path-provisioner und MetallLB hatte ich mit diversen Problemen zu kämpfen,

doch gelang es mir, durch intensive Recherche jeweils Lösungen für diese Probleme zu finden.

Beim Aufbau des Patroni-Evaluationssystems brachte dieser zeitraubende Ansatz keinen Erfolg mehr.

Daher war ich gezwungen, einen Schritt zurückzutreten und das System auf das absolut notwendige Minimum zu vereinfachen.

Mit nur noch einem etcd Node, der auf einem anderen Server installiert war, gelang der Aufbau des Evaluationssystems.

Diesen Ansatz nahm ich mit Erfolg mit, als ich bei den Maintenance-Tools wieder auf Herausforderungen traf.

Auf einen Satz reduziert, dürfte daher die wichtigste Erkenntnis aus der Diplomarbeit folgende sein:

«Im Zweifelsfall geht man besser einen Schritt zurück und nimmt so viel Komplexität aus einem System wie möglich!».

Abbildungsverzeichnis

1.1	Spitalregionen Kanton Graubünden[66]	1
1.2	Wahlkreise Kanton St. Gallen[93]	2
1.3	Spitalregionen / Spitalstrategie Kanton St. Gallen[60]	3
1.4	Organigramm Kantonsspital Graubünden	4
1.5	Organigramm Departement 10 - ICT	5
1.6	Datenbanken - Aufgeschlüsselt nach RDBMS - Datenbanken	9
1.7	Datenbanken - Aufgeschlüsselt nach RDBMS - Instanzen	10
1.8	Risikomanagement PostgreSQL	16
1.9	Systemabgrenzung	21
1.10	Risikomanagement Projekt	26
1.11	Riskikomatrix - Assessment 21.03.2024	28
1.12	Riskikomatrix - Assessment 29.04.2024	30
1.13	Riskikomatrix - Assessment 13.05.2024	32
3.1	Sharding - Vertikale Partitionierung	39
3.2	Sharding - Horizontales Partitionierung	40
3.3	Monolithische vs. verteilte SQL Systeme	42
3.4	CAP-Theorem	44
3.5	Datenbankskalierung	45
3.6	Präferenzmatrix	49
3.7	Testing - ERD DB self_healing_test	52
3.8	Benchmark Settings - Zabbix - Systeminformationen	53
3.9	Benchmark Settings - Zabbix - Connections per Seconds	53
3.10	Benchmark Settings - Zabbix - Queries per Seconds	53
3.11	Benchmark Settings - Zabbix - Client Queries per Seconds	54
3.12	Benchmark Settings - Zabbix - DB Size	54
3.13	Benchmark Settings - Anzahl Records / Skalierungsfaktor	57
3.14	pg_auto_failover-Architektur - Single Standby	61
3.15	pg_auto_failover-Architektur - Multi-Node Standby	61
3.16	pg_auto_failover-Architektur - Citus	62
3.17	CloudNativePG - Kubernetes - PostgreSQL	63
3.18	CloudNativePG - Kubernetes - Read-write workloads	64
3.19	CloudNativePG - Kubernetes - Read-only workloads	64
3.20	Patroni-Architektur	67
3.21	Stackgres - Grundarchitektur	69
3.22	Citus - Coordinator und Workers	70
3.23	Citus - Row-Based-Sharding	70
3.24	Citus - Schema-Based-Sharding	71
3.25	StackGres-Citus - Shard-Replikation	72
3.26	YugabyteDB - Grundkonzept	74

3.27 YugabyteDB - Architektur	74
3.28 YugabyteDB - Sharding	75
3.29 YugabyteDB - Tablet - Leader und Follower	76
3.30 YugabyteDB - Tablet - Replikationsfaktor	77
3.31 YugabyteDB - Zonen	78
3.32 YugabyteDB - Zone outage Tolerance	79
3.33 Evaluationssystem - Distributed SQL / Shards	81
3.34 Patroni - Evaluationsarchitektur	82
3.35 Stackgres - Citus - Evaluationsarchitektur Benchmarking	85
3.36 Stackgres - Citus - Evaluationsarchitektur Self Healing Tests	85
3.37 Stackgres - Citus - Resourcen - Stack	86
3.38 Stackgres - Citus - Datenbank - Cluster	87
3.39 YugabyteDB - Evaluationsarchitektur	92
3.40 YugabyteDB - Subscription Yugaware	93
3.41 Benchmarking - ERD pgbench	103
3.42 Benchmarks - tps	105
3.43 Benchmarks - latency	105
3.44 Benchmarks - tps Patroni Replica	106
3.45 Benchmarks - latency Patroni Replica	106
3.46 Benchmarks - Fehler bei mixed-Transaktionen	107
3.47 Benchmarks - Initialisierungszeit - sekunden	107
3.48 Benchmarks - Initialisierungszeit - Minuten	108
3.49 Benchmarks - Initialisierungszeit - Stunden	108
3.50 Zeitaufwände pro Betriebstask	111
3.51 Zeitaufwände	112
3.52 Zeitaufwände summiert	113
3.53 Kostenaufwände pro Betriebstask	114
3.54 Kostenaufwände	115
3.55 Kostenaufwände Total	116
3.56 Kosten-Nutzen-Analyse	117
3.57 Kosten-Nutzen-Ranking	117
3.58 Kosten-Nutzen-Diagramm	118
3.59 Testsystem - Architektur	120
3.60 Technical Review - Patroni Nodes	126
3.61 Technical Review - Datenbanken	127
3.62 Technical Review - PostgreSQL Extensions	127
3.63 Technical Review - PostgreSQL User und Rollen	128
3.64 Technical Review - HAProxy sks1266	129
3.65 Technical Review - HAProxy sks1267	129
3.66 Technical Review - PgBouncer - Datenbanken	130
3.67 Technical Review - PostgreSQL - Log Rotation	130
3.68 Testing - Patroni Node hinzufügen - add_pgnode.yml	134
3.69 AUTOVACUUM - Berechneter autovacuum_vacuum_scale_factor	138
3.70 PRTG - Patroni Monitoring	138

3.71 PRTG - Patroni Cluster Sensor Setting	140
3.72 PRTG - Patroni Cluster Sensor	140
 I CloudNativePG - Pulse	xxvii
II CloudNativePG - Code Frequency	xxvii
III CloudNativePG - Community Standards	xxviii
IV CloudNativePG - Contributors Commits	xxviii
V CloudNativePG - Contributors Deletations	xxix
VI CloudNativePG - Contributors Additions	xxix
VII CloudNativePG - Commit Activity	xxix
VIII CloudNativePG - Network Graph	xxx
IX Patroni - Pulse	xxx
X Patroni - Code Frequency	xxxi
XI Patroni - Community Standards	xxxi
XII Patroni - Contributors Commits	xxxii
XIII Patroni - Contributors Deletations	xxxii
XIV Patroni - Contributors Additions	xxxii
XV Patroni - Commit Activity	xxxiii
XVI Patroni - Network Graph	xxxiii
XVII Stackgres - Pulse	xxxiv
XVIII Citus - Pulse	xxxiv
XIX Stackgres - Code Frequency	xxxiv
XX Citus - Code Frequency	xxxv
XXI Stackgres - Community Standards	xxxv
XXII Citus - Community Standards	xxxvi
XXIII Stackgres - Contributors Commits	xxxvi
XXIV Stackgres - Contributors Deletations	xxxvi
XXV Stackgres - Contributors Additions	xxxvii
XXVI Citus - Contributors Commits	xxxvii
XXVII Citus - Contributors Deletations	xxxvii
XXVIII Citus - Contributors Additions	xxxvii
XXIX Stackgres - Commit Activity	xxxviii
XXX Citus - Commit Activity	xxxviii
XXXI Stackgres - Network Graph	xxxix
XXXII Citus - Network Graph	xxxix
XXXIII YugabyteDB - Pulse	xl
XXXIV YugabyteDB - Code Frequency	xl
XXXV YugabyteDB - Community Standards	xli
XXXVI YugabyteDB - Contributors	xli
XXXVII YugabyteDB - Commit Activity	xlii
XXXVIII YugabyteDB - Network Graph	xlii
XXXIX Aproxy - Web-GUI	lxxxiv
XL StackGres Testing - Node sks1184 down	cxxii
XLI StackGres Testing - Pods Down	cxxii

XLII StackGres Testing - Patroni Übersicht	cxxiii
XLIII StackGres Testing - DB Zugriff	cxxiii
XLIV StackGres Testing - Connection Timeout	cxxiv
XLV YugabyteDB - Too big clock skew is detected	cxxiv
XLVI YugabyteDB - Tablet Leader - No Lease	cxxv
XLVII YugabyteDB - CrashLoopBackOff	cxxv
XLVIII YugabyteDB - Too big clock skew is detected - tmaster	cxxvi
XLIX YugabyteDB - Too big clock skew is detected - tserver	cxxvi
L local-path-provisioner[37]	cciv

Tabellenverzeichnis

1.1 Inventarisierte Datenbanksysteme	7
1.2 Datenbankinventar	8
1.3 Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt	11
1.4 Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken	13
1.5 Administrative Aufgaben	17
1.6 Automatisierung Administrativer Aufgaben	18
1.7 Ziele	19
1.8 Gegebene Systeme	20
1.9 Abhängigkeiten	23
1.10 Risiko-Matrix der Diplomarbeit	25
1.11 Neu Erkannte / Erfasste Risiken	27
1.12 Risiko-Assessment 21.03.2024	27
1.13 Risiko-Assessment 29.04.2024	29
1.14 Risiko-Assessment 13.05.2024	31
2.1 Projektcontrolling	33
2.2 Fachgespräche	38
3.1 Quorum Beispiele	43
3.2 Anforderungskatalog	48
3.3 Stakeholder	49
3.4 Benchmark Settings - Mixed Transaktionen	54
3.5 Benchmark Settings - DQL Transaktionen	55
3.6 Benchmark Settings - Skalierungsfaktoren	55
3.7 Benchmark Settings - Datenbankgrößen / Skalierungsfaktor	55
3.8 Benchmark Settings - Tabellengrößen / Skalierungsfaktor	56
3.9 Vorauswahl - Ausgeschieden	80
3.10 Vorauswahl - Evaluation	80
3.11 Evaluationssyssteme	81
3.12 Evaluationssysstem - Distributed SQL / Sharding	81
3.13 Testresultate Evaluation Patroni	99
3.14 Testresultate Evaluation StackGres - Citus	100
3.15 Testresultate Evaluation YugabyteDB	100
3.16 Kostenberechnung - Annahmen	109
3.17 Gemessene und Extrapolierte Aufwände Bsp.	110
3.18 Testsystem - Komponenten	119
3.19 Testsystem - Inventar	121
3.20 Testsystem - Auxiliar Inventar	121
3.21 Testsystem - Benötigte Ports	123
3.22 Testresultate Testsystem	133

4.1	Result - Zielüberprüfung	143
I	Arbeitsrapport	iii
II	Kommentare - Anmerkung	xxv

Listings

3.1 Patroni - etcd API V2 Error	83
3.2 Patroni - etcd API V2 Enable	83
3.3 Patroni - etcd3 Flag	83
3.4 Patroni - Passwörter	83
3.5 Patroni - Synchrone Replikation setzen	84
3.6 StackGres - values.yaml - Extension proxyUrl	88
3.7 StackGres - values.yaml - Extension Proxy	88
3.8 StackGres-Citus - LoadBalancer -Annotation	89
3.9 StackGres-Citus - StorageClass -PVC Binding	90
3.10 StackGres-Citus - Instanz-Profile	90
3.11 StackGres-Citus - StorageClass -PVC Binding	91
3.12 StackGres-Citus - Cluster Profil	91
3.13 StackGres-Citus - SGPostgresConfig	92
3.14 metallb - Konfig YAML - Detail L2Advertisement	93
3.15 YugabyteDB - Helm Chart Manifest - Detail Image	94
3.16 YugabyteDB - Helm Chart Manifest - Detail StorageClass	94
3.17 YugabyteDB - Helm Chart Manifest - Detail Resources	94
3.18 YugabyteDB - Helm Chart Manifest - Detail Replika	95
3.19 YugabyteDB - Helm Chart Manifest - Detail Disable YSQL	95
3.20 YugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints	96
3.21 local-path-provisioner nodePathMap	96
3.22 local-path-provisioner nodePathMap Beispiel	97
3.23 YugabyteDB - StorageClass nodeAffinity	97
3.24 Patroni - Testing - Switchover	98
3.25 Patroni - Testing - Reinit	99
3.26 Patroni - Benchmarking - Monitoring	101
3.27 Citus - Benchmarking - Distributed Table Sharding	102
3.28 Citus - Benchmarking - Reference Table Sharding	103
3.29 main.xyml - No Proxy	124
3.30 main.xyml - Gruppenproxy	124
3.31 main.xyml - SProxy	124
3.32 Deploy - deploy_pgcluster.yml	124
3.33 Maintenance - config_pgcluster.yml	125
3.34 HAProxy Node erweitern - Inventory	125
3.35 HAProxy Node erweitern - add_balancer.yml	126
3.36 Patroni Node erweitern - Inventory	126
3.37 Patroni Node erweitern - add_pgnode.yml	126
3.38 PgBouncer - Connect	130
3.39 PgBouncer - Databases	130

3.40 Maintenance-Tool - List - Maintenance-Tool - Bloated Tables / Indices	135
3.41 Maintenance-Tool - VACUUM - Maintenance-Tool - Bloated Tables / Indices	136
3.42 Maintenance-Tool - REINDEX - Maintenance-Tool - Bloated Tables / Indices	136
3.43 Maintenance-Tool - Check AUTOVACUUM - Maintenance-Tool - Bloated Tables / Indices . .	136
3.44 Maintenance-Tool - Parameter - Maintenance-Tool - AUTOVACUUM	137
 1 Proxy Settings	xlii
2 rke2 server - Verzeichnis erstellen	xliii
3 rke2 server - config.yaml	xliii
4 rke2 server - cilium-config.yaml	xliii
5 rke2 server installieren	xliii
6 rke2 agenten installieren	xliv
7 rke2 agent - config.yaml	xliv
8 -rke2 agent service restart	xliv
9 rke2 server proxy	xliv
10 iptables entries server	xliv
11 local-path-storage auf Linux Bereitstellen	xlv
12 local-path-provisioner definieren	xlv
13 local-path-storage aktualisieren	xlvi
14 MetallB installieren	xlvi
15 MetallB konfigurieren	xlvi
16 MetallB Konfiguration einspielen	xlvii
17 rke2 - 250GiB Disk mount	xlvii
18 local-path-storage 250GiB auf Linux Bereitstellen	xlvii
19 local-path-provisioner Grosse Volumes	xlvii
20 local-path-storage 250GiB aktualisieren	xlviii
21 YugabyteDB - StorageClass setzen	xlviii
22 YugabyteDB - StorageClass / PersistentVolume aktivieren	xlix
23 YugabyteDB - Namespace	xlix
24 YugabyteDB - Helm Chart Manifest	xlix
25 YugabyteDB - Installation	lx
26 YugabyteDB - Deinstallieren	lx
27 YugabyteDB - StorageClass setzen	lx
28 YugabyteDB - StorageClass / PersistentVolume Grosse Volumes aktivieren	lxii
29 YugabyteDB - Namespace 250GiB	lxii
30 YugabyteDB - Helm Chart Manifest 250GiB	lxii
31 YugabyteDB - Cloud - Region - Zone	lxix
32 YugabyteDB - Benchmarking - DB erstellen	lxix
33 YugabyteDB - Benchmarking - Table Size	lxix
34 YugabyteDB - Self Healing Tests - CREATE-SQL	lxx
35 YugabyteDB - Self Healing Tests - Init Data	lxxii
36 YugabyteDB - Self Healing Tests - Failover Data	lxxii
37 YugabyteDB - Self Healing Tests - Recovery Data	lxxiii
38 sks9016 - Download YugabyteDB On-Premise	lxxiii

39	sks9016 - Installation YugabyteDB On-Premise	lxxiv
40	sks9016 - Check YugabyteDB On-Premise	lxxiv
41	Patroni - Firewall Settings	lxxiv
42	Patroni - Proxy Settings	lxxv
43	Patroni - apt-Proxy Settings	lxxv
44	Patroni - PostgreSQL einbinden	lxxv
45	Patroni - Prerequisites installieren	lxxvi
46	Patroni - Stop Patroni und PostgreSQL	lxxvi
47	Patroni - Symlink binaries	lxxvi
48	Patroni - Checks	lxxvi
49	etcd - Installation	lxxvi
50	etcd - Konfiguration	lxxvi
51	etcd - restart	lxxvi
52	etcd - member list	lxxvii
53	Patroni Bootstrap - Konfiguration bereinigen	lxxvii
54	Patroni - Konfiguration - sks1232	lxxvii
55	Patroni - Konfiguration - sks1233	lxxix
56	Patroni - Konfiguration - sks1234	lxxx
57	Patroni Bootstrap - pg_hba	lxxxii
58	Patroni Bootstrap - Patroni-Verzeichnis	lxxxii
59	Patroni Bootstrap - Neu starten	lxxxii
60	Patroni Bootstrap - Disable PostgreSQL	lxxxiii
61	HAProxy - Hostliste	lxxxiii
62	HAProxy - Installation	lxxxiii
63	HAProxy - Safe Alte Config	lxxxiii
64	HAProxy - Konfiguration	lxxxiii
65	HAProxy - Restart	lxxxiv
66	Patroni - 250GiB Disk mount	lxxxv
67	Patroni - 250GiB Verzeichnisse	lxxxv
68	Patroni - 250GiB Cluster Pause	lxxxv
69	Patroni - 250GiB PostgreSQL stoppen	lxxxvi
70	Patroni - 250GiB move pg_wal	lxxxvi
71	Patroni - 250GiB chmod - chown pg_wal	lxxxvi
72	Patroni - 250GiB PostgreSQL - Patroni resume	lxxxvi
73	Patroni - 250GiB Finaler Check	lxxxvi
74	Patroni - 250GiB set Parameter	lxxxviii
75	Patroni - Benchmarking - DB erstellen	lxxxix
76	Patroni - Benchmarking - DB Cleanup	xc
77	Patroni - Benchmarking - Tablespaces erneut erstellen	xc
78	Patroni - Self Healing Tests - CREATE-SQL	xc
79	Patroni - Self Healing Tests - Init Data	xcii
80	Patroni - Self Healing Tests - Failover Data	xciii
81	Patroni - Self Healing Tests - Recovery Data	xciii
82	StackGres-Citus - StorageClass setzen	xciv

83	StackGres-Citus - StorageClass / PersistentVolume aktivieren	xcv
84	StackGres-Citus - Namespace	xcv
85	StackGres-Citus - Helm Chart Manifest	xcv
86	StackGres-Citus - Installation	cii
87	StackGres-Citus - Post-Installation	cii
88	StackGres-Citus - System Username	cii
89	StackGres-Citus - System Passwort	cii
90	StackGres-Citus - System Passwort Cleanup	cii
91	StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator	cii
92	StackGres-Citus - Benchmarking - SGInstanceProfile Shard	ciii
93	StackGres-Citus - Benchmarking - Instanz-Profil Deploy	ciii
94	StackGres-Citus - Benchmarking - SGShardedCluster	ciii
95	StackGres-Citus - Benchmark - Cluster Deploy	civ
96	StackGres-Citus - Benchmark DB Passwort	civ
97	StackGres-Citus - Self Healing Testing - SGInstanceProfile Coordinator	civ
98	StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard	civ
99	StackGres-Citus - Self Healing Testing - Instanz-Profil Deploy	cv
100	StackGres-Citus - Self Healing Testing - SGShardedCluster	cv
101	StackGres-Citus - Self Healing Testing - Cluster Deploy	cvi
102	StackGres-Citus - Self Healing Testing DB Passwort	cvi
103	StackGres-Citus - Deinstallieren	cvi
104	StackGres-Citus - StorageClass setzen	cvi
105	StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren	cvi
106	StackGres-Citus - Namespace 250GiB	cvi
107	StackGres-Citus - Installation 250GiB	cvi
108	StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB	cvi
109	StackGres-Citus - Benchmarking - SGInstanceProfile Shard 250GiB	cvi
110	StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB	cvi
111	StackGres-Citus - Benchmarking - SGPostgresConfig	cvi
112	StackGres-Citus - Benchmark - Deploy SGPostgresConfig	cix
113	StackGres-Citus - Benchmarking - SGShardedCluster 250GiB	cix
114	StackGres-Citus - Benchmark - Cluster Deploy 250GiB	cx
115	StackGres-Citus - Self Healing Tests - CREATE-SQL	cx
116	StackGres-Citus - Self Healing Tests - Init Data	cxi
117	StackGres-Citus - Self Healing Tests - Failover Data	cxi
118	StackGres-Citus - Self Healing Tests - Recovery Data	cxi
119	YugabyteDB - Benchmarking-Commands	cxv
120	yugabyteDB - Benchmarking - Table Size SQL	cvi
121	Patroni - Benchmarking-Commands	cvi
122	Patroni - Benchmarking - Table Size SQL	cix
123	StackGres-Citus - Benchmarking-Commands	cxx
124	StackGres-Citus - Benchmarking - Table Size SQL	cxxii
125	Ansible - Installation	cxxvii
126	Ansible - Repository Clone	cxxvii

127 Ansible - cd Repository	cxxvii
128 Testsystem - Firewall Settings	cxxvii
129 Testsystem - Proxy Settings	cxxviii
130 Testsystem - apt-Proxy Settings	cxxviii
131 Testsystem - Deployment - inventory	cxxix
132 Testsystem - Deployment - main.yml	cxxx
133 Deploy - Anhang - Ansible Ping	cxlvi
134 Deploy - Anhang - deploy_pgcluster.yml	cxlvi
135 Deploy - Anhang - Deployt	cxlvi
136 Testsystem - Anhang - Maintenance - main.yml	clxx
137 Testsystem - Anhang - Maintenance - config_pgcluster.yml	clxxxiii
138 Testsystem - sks9016 - apt-Proxy Settings	clxxxiii
139 Testsystem - sks9016 - Proxy Settings	clxxxiii
140 Testsystem - Anhang - add_balancer.yml - inventory	clxxxiii
141 Testsystem - Anhang - add_balancer.yml	clxxxiv
142 Testsystem - Anhang - add_pgnode.yml - inventory	clxxxiv
143 Testsystem - Anhang - add_pgnode.yml	clxxxv
144 Maintenance-Tool - Bloated Tables / Indices - ksgr_postgresql_maintenance_bloated_tables.pyclxxxv	
145 Maintenance-Tool - Bloated Tables / Indices - Namespace	clxxxviii
146 Maintenance-Tool - Bloated Tables / Indices - Python > ConfigMap	clxxxix
147 Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy	clxxxix
148 Maintenance-Tool - Bloated Tables / Indices - Base64	clxxxix
149 Maintenance-Tool - Bloated Tables / Indices - Secret	clxxxix
150 Maintenance-Tool - Bloated Tables / Indices - Secret Deploy	clxxxix
151 Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test	cxc
152 Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test Deploy	cxc
153 Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-testcxc	
154 Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic- test Deploy	cxi
155 Maintenance-Tool - AUTOVACUUM - ksgr_postgresql_maintenance_autovacuum_calculation.pycxxi	
156 Maintenance-Tool - AUTOVACUUM - Python > ConfigMap	cxcv
157 Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy	cxcv
158 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres	cxcv
159 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres Deploy	cxcvi
160 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum	cxcvi
161 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum Deploy	cxcvi
162 Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation	cxcvii
163 Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation Deploy	cxcviii
164 Monitoring - KSGR - Patroni - Healthcheck.py	cxcviii

Literatur

- [1] *7210-vrrp.pdf.* https://www.cisco.com/c/de_de/support/docs/security/vpn-3000-series-concentrator-7210-vrrp.pdf.
- [2] *About pgbench-tools.* <https://github.com/gregs1104/pgbench-tools>. original-date: 2010-02-17T13:33:28Z 2023.
- [3] Satyadeep Ashwathnarayana und Inc. Netdata. *How to monitor and fix Database bloats in PostgreSQL? / Netdata Blog.* <https://blog.netdata.cloud/postgresql-database-bloat/>. 2022.
- [4] unknown author. *#1 Backup-Lösung für Kubernetes.* <https://www.veeam.com/de/kubernetes-native-backup.html?ck=1697900263871>.
- [5] unknown author. *14.1. Using EXPLAIN.* <https://www.postgresql.org/docs/16/using-explain.html>. 2024.
- [6] unknown author. *API Reference - CloudNativePG.* <https://cloudnative-pg.io/documentation/1.22/cloudnative-pg.v1/>.
- [7] unknown author. *API reference (for YSQL and YCQL).* <https://docs.yugabyte.com/preview/api/>.
- [8] unknown author. *Architecture Basics — pg_auto_failover 2.0 documentation.* <https://pg-auto-failover.readthedocs.io/en/main/architecture.html>.
- [9] unknown author. *Benchmark Setup with Citus and pgbench - Citus 12.1 documentation.* https://docs.citusdata.com/en/stable/extra/write_throughput_benchmark.html.
- [10] unknown author. *Benefits of using YugabyteDB.* <https://docs.yugabyte.com/preview/features/>. Section: preview.
- [11] unknown author. *Choosing Distribution Column - Citus 12.1 documentation.* https://docs.citusdata.com/en/v12.1/sharding/data_modeling.html#distributed-data-modeling.
- [12] unknown author. *Cilium - Cloud Native, eBPF-based Networking, Observability, and Security.* <https://cilium.io>.
- [13] unknown author. *Citus Replication Model: Today and Tomorrow - Replication Groups.* <https://www.citusdata.com/blog/2016/12/15/citus-replication-model-today-and-tomorrow/>.
- [14] unknown author. *Citus Support — pg_auto_failover 2.0 documentation.* <https://pg-auto-failover.readthedocs.io/en/main/citus.html>.
- [15] unknown author. *CLIs and command line tools.* <https://docs.yugabyte.com/preview/admin/>.
- [16] unknown author. *CloudNativePG - Main Features.* <https://cloudnative-pg.io/documentation/1.22/#main-features>.
- [17] unknown author. *Cluster Management - Citus 12.1 documentation - worker-node-failure.* https://docs.citusdata.com/en/v12.1/admin_guide/cluster_management.html#worker-node-failures.
- [18] unknown author. *Cluster Management — Citus Docs 7.2 documentation.* https://docs.citusdata.com/en/v7.2/admin_guide/cluster_management.html.

- [19] unknown author. *Concepts - Citus 12.1 documentation - row-based-sharding*. https://docs.citusdata.com/en/v12.1/get_started/concepts.html#row-based-sharding.
- [20] unknown author. *Consensus Protocol | Raft | Consul | HashiCorp Developer*. <https://developer.hashicorp.com/consul/docs/architecture/consensus>.
- [21] unknown author. *Consul Architecture | Consul | HashiCorp Developer*. <https://developer.hashicorp.com/consul/docs/architecture>.
- [22] unknown author. *Creating and Modifying Distributed Objects (DDL) - Citus 12.1 documentation*. https://docs.citusdata.com/en/stable/develop/reference_ddl.html?highlight=create_reference_table#reference-tables.
- [23] unknown author. *DCS Failsafe Mode — Patroni 3.3.0 documentation*. https://patroni.readthedocs.io/en/latest/dcs_failsafe_mode.html#dcs-failsafe-mode.
- [24] unknown author. *Dynamic Configuration Settings — Patroni 3.2.2 documentation*. https://patroni.readthedocs.io/en/latest/dynamic_configuration.html.
- [25] unknown author. *EDB-Home*. <https://enterprisedb.com/>.
- [26] unknown author. *EDB: Open-Source, Enterprise Postgres Database Management*. <https://www.enterprisedb.com/>.
- [27] unknown author. *Envoy proxy - home*. <https://www.envoyproxy.io/>.
- [28] unknown author. *etcd*. <https://etcd.io/>.
- [29] unknown author. *Features - StackGres Documentation*. <https://stackgres.io/doc/latest/features/>.
- [30] unknown author. *HAProxy Documentation Converter*. <https://docs.haproxy.org/>.
- [31] unknown author. *HAProxy version 2.9.6 - Starter Guide*. <https://docs.haproxy.org/2.9/intro.html#3.2>.
- [32] unknown author. *Helm*. <https://helm.sh/>.
- [33] unknown author. *Introduction | RKE2*. <https://docs.rke2.io/>. 2024.
- [34] unknown author. *Introduction | Vault | HashiCorp Developer*. <https://developer.hashicorp.com/vault/docs/what-is-vault>.
- [35] unknown author. *Introduction to Cilium & Hubble — Cilium 1.15.3 documentation*. <https://docs.cilium.io/en/stable/overview/intro/#what-is-cilium>.
- [36] unknown author. *Keycloak*. <https://www.keycloak.org/>.
- [37] unknown author. *Local Disk Storage for Kubernetes | ITNEXT*. <https://itnext.io/local-disk-storage-for-kubernetes>.
- [38] unknown author. *Manual Pages — pg_auto_failover 2.0 documentation*. <https://pg-auto-failover.readthedocs.io/en/main/ref/manual.html>.
- [39] unknown author. *MetalLB provides Services with IP Addresses but doesn't ARP for the address · Issue #1154 · metallb/metallb*. <https://github.com/metallb/metallb/issues/1154>.
- [40] unknown author. *MetalLB, bare metal load-balancer for Kubernetes*. <https://metallb.universe.tf/>.
- [41] unknown author. *Multi-node Architectures — pg_auto_failover 2.0 documentation*. <https://pg-auto-failover.readthedocs.io/en/main/architecture-multi-standby.html>.

- [42] unknown author. *Percona Software for PostgreSQL*. <https://www.percona.com/postgresql/software>.
- [43] unknown author. *PgBouncer - lightweight connection pooler for PostgreSQL*. <https://www.pgbouncer.org/>.
- [44] unknown author. *Problem with 08P01: server conn crashed? · Issue #714 · pgbouncer/pgbouncer*. <https://github.com/pgbouncer/pgbouncer/issues/714>.
- [45] unknown author. *Red Hat Ansible Automation Platform automation controller*. <https://www.redhat.com/en/technologies/management/ansible/automation-controller>.
- [46] unknown author. *Replication in DocDB - Zone Fault Tolerance*. <https://docs.yugabyte.com/preview/architecture/docdb-replication/replication/>. Section: preview.
- [47] unknown author. *Support and Services for PostgreSQL*. <https://www.percona.com/postgresql/support-and-services>.
- [48] unknown author. *VRRP verstehen | Junos OS | Juniper Networks*. <https://www.juniper.net/documentation/de/de/software/junos/high-availability/topics/topic-map/vrrp-understanding.html>.
- [49] unknown author. *YB-TServer service*. <https://docs.yugabyte.com/preview/architecture/concepts/yb-tserver/>. Section: preview.
- [50] GitLab B.V. und GitLab Inc. *The DevSecOps Platform | GitLab*. <https://about.gitlab.com/>.
- [51] Alexandre Cassen und Read the Docs. *Introduction — Keepalived 1.2.15 documentation*. <https://keepalived.readthedocs.io/en/latest/introduction.html>. 2017.
- [52] Microsoft Corporation. *Azure SQL-Datenbank – ein verwalteter Clouddatenbankdienst | Microsoft Azure*. <https://azure.microsoft.com/de-de/products/azure-sql/database>. 2023.
- [53] Microsoft Corporation. *Datenbank-Software und Datenbankanwendungen | Microsoft Access*. <https://www.microsoft.com/de-de/microsoft-365/access>. 2023.
- [54] Microsoft Corporation. *Microsoft Data Platform | Microsoft*. <https://www.microsoft.com/de-ch/sql-server>.
- [55] Varun Dhawan und data-nerd.blog. *PostgreSQL-Diagnostic-Queries – data-nerd.blog*. <https://data-nerd.blog/2018/12/30/postgresql-diagnostic-queries/>.
- [56] Elektronik-Kompendium.de und Schnabel Schnabel. *SAN - Storage Area Network*. <https://www.elektronik-kompendium.de/sites/net/0906071.htm>. 2023.
- [57] DB-Engines und solidIT consulting & software development gmbh. *DB-Engines Ranking*. <https://db-engines.com/en/ranking>.
- [58] DB-Engines und solidIT consulting & software development gmbh. *relationale Datenbanken - DB-Engines Enzyklopädie*. <https://db-engines.com/de/article/relationale+Datenbanken?ref=RDBMS>.
- [59] The Linux Foundation. *Harbor*. <https://goharbor.io/>. 2023.
- [60] Kanton St. Gallen - Amt für Gesundheitsversorgung und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Weiterentwicklung der Strategie der St.Galler Spitalverbunde / sg.ch*. <https://www.sg.ch/gesundheit-soziales/gesundheit/gesundheitsversorgung--spitaeler-spitex/spitaeler-spitalzukunft.html>.
- [61] Git. *About - Git*. <https://git-scm.com/about>.

- [62] IBM Deutschland GmbH. *Was ist das CAP-Theorem?* / IBM. <https://www.ibm.com/de-de/topics/cap-theorem>. 2023.
- [63] IBM Deutschland GmbH. *Was ist OLAP?* / IBM. <https://www.ibm.com/de-de/topics/olap>.
- [64] Jedox GmbH. *Was ist OLAP? Online Analytical Processing im Überblick*. <https://www.jedox.com/de/blog/was-ist-olap/>. Section: Knowledge.
- [65] Pure Storage Germany GmbH. *Was ist ein Storage Area Network (SAN)?* / Pure Storage. <https://www.purestorage.com/de/knowledge/what-is-storage-area-network.html>.
- [66] Gesundheitsamt Graubünden, Uffizi da sanadad dal Grischun und Ufficio dell'igiene pubblica dei Grigioni. *Kenndaten 2016 Spitäler und Kliniken September 2018*. <https://www.gr.ch/DE/institutionen/verwaltung/djsg/ga/InstitutionenGesundheitswesens/Spitaeler/Dok%20Spitler/Kenndaten%202016%20Spit%C3%A4ler.pdf>.
- [67] The Pgpool Global Development Group. *What is Pgpool-II?* <https://www.pgpool.net/docs/44/en/html/intro-whatis.html>. 2023.
- [68] The PostgreSQL Global Development Group. *25.1. Routine Vacuuming*. <https://www.postgresql.org/docs/16/routine-vacuuming.html>. 2023.
- [69] The PostgreSQL Global Development Group. *pgbench*. <https://www.postgresql.org/docs/16/pgbench.html>. 2023.
- [70] CYBERTEC Guest. *A formula to calculate pgbench scaling factor for target DB size*. <https://www.cybertec-postgresql.com/en/a-formula-to-calculate-pgbench-scaling-factor-for-target-db-size>. 2018.
- [71] Michael Haag. *Built-in Connection Manager Turns Key PostgreSQL Weakness into a Strength*. <https://www.yugabyte.com/blog/connection-pooling-management/>. 2023.
- [72] Inc. HashiCorp. *Terraform by HashiCorp*. <https://www.terraform.io/>.
- [73] Patrick Hunt, Mahadev Konar, Flavio P Junqueira und Benjamin Reed. „ZooKeeper: Wait-free coordination for Internet-scale systems“. In: (2010).
- [74] Splunk Inc. *Splunk / Der Schlüssel zu einem resilierten Unternehmen*. https://www.splunk.com/de_de. 2023.
- [75] Sebastian Insausti. *Scaling PostgreSQL for Large Amounts of Data*. <https://severalnines.com/blog/scaling-postgresql-large-amounts-data/>. 2019.
- [76] Shiv Iyer und MinervaDB. *PostgreSQL DBA Daily Checklist*. <https://minervadb.xyz/postgresql-dba-daily>. 2020.
- [77] Unmesh Joshi. *Quorum*. <https://martinfowler.com/articles/patterns-of-distributed-systems/quorum.html>. 2020.
- [78] Martin Keen und IBM Deutschland GmbH. *IBM Db2*. <https://www.ibm.com/de-de/products/db2>.
- [79] Pasha Kostohrys. *Database replication — an overview*. <https://medium.com/@pkostohrys/database-replication-101-101>. 2020.
- [80] Anatoli Kreyman. *Was ist eigentlich Splunk?* <https://www.kreyman.de/index.php/splunk/76-was-ist-eigentlich-splunk>.

- [81] Pankaj Kushwaha und Unit 3D North Point House. *POSTGRESQL DATABASE MAINTENANCE. Routine backup of daily database... / by Pankaj kushwaha / Medium.* <https://pankajconnect.medium.com/postgresql-database-maintenance-66cd638d25ab>.
- [82] Red Hat Limited. *Was ist Ansible?* <https://www.redhat.com/de/technologies/management/ansible/what-is-ansible>.
- [83] Red Hat Limited. *Was ist CI/CD? Konzepte und CI/CD Tools im Überblick.* <https://www.redhat.com/de/topics/devops/what-is-ci-cd>.
- [84] Switzerland Linuxfabrik GmbH Zurich. *Keepalived — Open Source Admin-Handbuch der Linuxfabrik.* <https://docs.linuxfabrik.ch/software/keepalived.html>. 2023.
- [85] Nico Litzel, Stefan Luber und Vogel IT-Medien GmbH. *Was ist Elasticsearch?* <https://www.bigdata-insider.de/was-ist-elasticsearch-a-939625/>. 2020.
- [86] Hewlett Packard Enterprise Development LP. *Was ist SAN-Speicher? / Glossar.* <https://www.hpe.com/ch/de/what-is/san-storage.html>.
- [87] Julian Markwort. „Benchmarking four Different Replication Solutions“. In: ()..
- [88] Sujoy Nath. *Database Connection Pool.* <https://medium.com/@sujoy.swe/database-connection-pool-641>. 2023.
- [89] Diego Ongaro. „Consensus: Bridging Theory and Practice“. In: (2014).
- [90] Diego Ongaro und John Ousterhout. „In Search of an Understandable Consensus Algorithm“. In: ()..
- [91] Bruno Queirós und LinkedIn Ireland Unlimited Company. *Postgresql replication with automatic failover.* <https://www.linkedin.com/pulse/postgresql-replication-automatic-failover-bruno-queir%C3%A9s/>. 2020.
- [92] Karthik Ranganathan. *Evolving Clock Sync in Distributed Databases / YugabyteDB.* <https://www.yugabyte.com/blog/evolving-clock-sync-for-distributed-databases/>. 2022.
- [93] Kanton St. Gallen - Dienst für politische Rechte und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Wahlkreise für Kantonsratswahlen / sg.ch.* <https://www.sg.ch/politik-verwaltung/abstimmungen-wahlen/wahlen/Wahlkreise-im-Kanton-SG.html>.
- [94] Ed Reckers und SnapLogic Inc. *Was ist die Snowflake-Datenplattform?* <https://www.snaplogic.com/de/blog/snowflake-data-platform>. 2023.
- [95] IONOS SE. *Apache Cassandra: Verteilte Verwaltung großer Datenbanken.* <https://www.ionos.de/digitalguide/hosting/hosting-technik/apache-cassandra-vorgestellt/>. 2021.
- [96] IONOS SE. *Datenbankmanagementsystem (DBMS) erklärt.* <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/>. 2020.
- [97] IONOS SE. *MongoDB – die flexible und skalierbare NoSQL-Datenbank.* <https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysql/>. 2019.
- [98] IONOS SE. *SQLite: Die bekannte Programmiersbibliothek im Detail vorgestellt.* <https://www.ionos.de/digitalguide/websites/web-entwicklung/sqlite/>. 2023.
- [99] IONOS SE. *Terraform.* <https://www.ionos.de/digitalguide/server/tools/was-ist-terraform/>. 2020.

- [100] IONOS SE. *Was ist Redis? Die Datenbank vorgestellt.* <https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-redis/>. 2020.
- [101] IONOS SE. *Was ist SIEM (Security Information and Event Management)?* <https://www.ionos.de/digitalguide/server/sicherheit/was-ist-siem/>. 2020.
- [102] Sami Ahmed Siddiqui. *Distributed SQL 101.* <https://www.yugabyte.com/distributed-sql/>.
- [103] Inc. Snowflake. *Datenbanken, Tabellen und Ansichten – Überblick | Snowflake Documentation.* <https://docs.snowflake.com/de/guides-overview-db>.
- [104] Thomas-Krenn.AG. *Git Grundlagen – Thomas-Krenn-Wiki.* https://www.thomas-krenn.com/de/wiki/Git_Grundlagen.
- [105] vitabaks/postgresql_cluster. https://github.com/vitabaks/postgresql_cluster. original-date: 2019-06-04T13:26:17Z. 2024.
- [106] Rainer Züst. „Einstieg ins Systems Engineering“. In: (2002).

Abkürzungen

ICT	information and communications technology
ibW	ibW Höhere Fachschule Südostschweiz
KSGR	Kantonsspital Graubünden
KPS	KSGR Provisioning System
RDBMS	Relational Database Management System
DBMS	Database Management System
k8s	Kubernetes
HPE	Hewlett Packard Enterprise
HP-UX	Hewlett Packard UNIX
SAP	Systemanalyse Programmierung
SQL	Structured Query Language
DBA	Database Administrator / Datenbankadministrator
HA	High Availability
PRTG	Paessler Router Traffic Grapher
SAN	Storage Area Network
SIEM	Security Information and Event Management
CI/CD	Continuous Integration/Continuous Delivery
SWOT-Analyse	Strengths, Weaknesses, Opportunities, Threats
OLAP	Online Analytical Processing
IaC	Infrastructure as Code
IPERKA	Informieren, Planen, Entscheiden, Realisieren, Kontrollieren, Auswerten
BSI	Bundesamt für Sicherheit in der Informationstechnik
VRRP	Virtual Router Redundancy Protocol
PKI	Public Key Infrastructure
DCS	Distributed Configuration Store
DQL	Data Query Language

DML	Data Manipulation Language
ACID	Atomicity, Consistency, Isolation und Durability
EDB	EnterpriseDB
CRD	Custom Resource Definition
rke2	Ranger Kubernetes Engine 2
BGP	Border Gateway Protocol
NTP	Network Time Protocol
BSD	Berkeley Software Distribution

Glossar

Ansible Ansible ist ein Open-Soure Automatisierungstool zur Provisionierung, Konfiguration, Deployment und Orchestrierung. Ansible verbindet sich auf die Zielgeräte und führt dort die hinterlegten Module aus. Oft werden die verschieden Aufgaben in einem Skript in einem sogenannten Playbook geschrieben werden[82].. 1, 20, 109, 121, 129, 131, 134, 142, 143, 146, 147, 172, cxxvii

AUTOVACUUM Der AUTOVACUUM Job räumt die Tablespaces und Data Files innerhalb von PostgreSQL sowie auf dem Filesystem nach Lösch- und Manipulationstransaktionen auf, aktualisiert Datenbank interne Statistiken und verhindert Datenverlust von selten genutzten Datensätzen[68].. 4, 18, 19, 134, 135, 136, 138, 143, 150, 159, iii, lxxxviii, cxci, cxcv, cxcvi, cxcviii

Cassandra Cassandra ist eine spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[95] wurde.. 7, 73, 74, 80

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuierlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[83].. 166, 172

Cilium Cilium ist ein Netzwerk-Connector zwischen den Services von Container-Applikationen und Container Management-Systemen wie Docker oder k8s. Nebst simplen Networking bietet Cilium auch Netzwerk-Policies, Load-Balancer und andere Features[12, 35].. 93

Connection Pooler Ein Connection Pooler hält permanent eine spezifische Menge an Connections zur Datenbank offen, die er wiederum einer Applikation präsentiert. Schliesst eine Applikation ihre Connection, behält sie der Pooler offen und verwendet sie wieder. Dadurch entfallen für die Applikation das Öffnen der Connection, es wird an Overhead gespart, die Anzahl Connections auf seitens DB reduziert[88].. 58, 59, 63, 66, 69, 73, 119, 120, 121, 122, 134, 143, 145, 146

Consul Consul ist ein Protokoll, welches zur Konsensfindung dient[21, 20].. 66, 122

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[96].. 166

DCS Der DSC ist eine Kernkomponente von Patroni [24]. Realisiert wird der DCS bei Patroni mit etcd.. 119, 122, 166

Debian Debian gehört nebst Slackware Linux zu den ältesten Linux Distribution, die noch immer gepflegt und eingesetzt werden. Sie wurde im August 1993 gestartet und brachte im Laufe der Zeit einige der beliebtesten Distributionen wie Ubuntu hervor.. 20, 124

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine, die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[85].. 7

etcd etcd ist Key-Value-Store, der Konfigurationen in einem HA System speichert. etcd ist Konsensbasiert und dient zum Erhalten eines Quorums[28].. 66, 82, 83, 119, 122, 123, 139, 145, 148, 168, iii, lxxvi

Failover In einem Fehlerfall wird in einem HA-System in den meisten Fällen ein Primary Node auf den Secondary ungeplant geswitched.. 19, 42, 43, 58, 120, 143, 169, lxxii, xciii

Foreman Foreman ist ein Lifecycle Management und Provisioning System für virtuelle und physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 20, 25, 121, 124, lxxv

Git Git ist eine Versionierungssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral, sondern dezentral organisiert und arbeiten daher mit Working Copies von Repositories[61, 104].. 169

GitHub GitHub ist ein Git basierendes System für die Versionierung und bietet dabei auch noch Dienste für CI/CD. GitHub, dass mittlerweile zum Microsoft-Konzern gehört, kann sowohl als **Online** Dienst als auch als On-premises Service konsumiert werden. Hierfür ist der GitHub Enterprise Server notwendig[R398TJSHB, UL2FJNU].. 109, 116, 121, 124, 141, 146, iii, cxxvii

GitLab GitLab ist ein Git basierendes System für die Versionierung und bietet dabei auch noch Dienste für CI/CD. GitLab kann sowohl als **Online** Dienst als auch als On-premises Service konsumiert werden[50].. 19, 58, 122

HAProxy HAProxy ist ein Layer 7 Proxy und Load Balancer[31].. 60, 65, 82, 121, 122, 123, 125, 126, 131, 133, 134, 143, 145, 146

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[59]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 19, 58, 122

helm helm bietet mit seinen helm charts Paketressourcen, die das deployment von Kubernetes-Ressourcen erleichtert[32].. 87, 135, xliv, cvii, cxc, cxvii

HP-UX Dieses UNIX-Derivat ist ein Abkömmling von System III, System V R3 und System V R4 und wurde von HP zum ersten Mal 1982 veröffentlicht.. 5, 11, 25, 166

IBM DB2 IBM DB2 ist eine relationale Datenbank[78], deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 44

keepalived keepalived nutzt VRRP, um eine leichtgewichtige Lösung für ein HA-Failover zu realisieren. keepalived benötigt dazu keinen dritten Node, also einen Quorum-Node. Wenn die definierte sekundärseite keine Antwort mehr von der primären Seite nach einer definierten Anzahl versuchen in einem bestimmten Interval mehr bekommt oder ein per Skript definiertes Event auf der primären Seite eintrifft, wird ein Failover auf die sekundäre Seite ausgeführt. Je nach Konfiguration kann der Restore auf die primäre Seite eingeleitet werden, wenn diese wieder verfügbar ist oder der Restore unterbunden werden[84, 51].. 58

Key-Value-orientierte Siehe Key-Value-Datenbank. 172

Key-Value-Datenbank Eine Key-Value-Datenbank ist ein Typ der NoSQL Datenbanken. Diese Datenbanken haben einen Primary Key und oft mindestens einen Sort Key. Key-Value-Datenbanken können auch Objekte mit Subitems resp. Referenzen dazu speichern. Eine bekannte Key-Value-Datenbank ist Redis. 169, 170, 171

Keycloak Keycloak ist ein Identity und Access Management Tool[36], welches vom KSGR für SSO resp. Authentifizierung und das Management von Secrets für Kubernetes und Applikationen, die in Kubernetes betrieben werden, verwendet wird.. 122

Key-Value-Store Siehe Key-Value-Datenbank. 55, 66, 74, 168

Kubernetes Kubernetes, oder k8s, ist eine Open-Source Containerplattform, die ursprünglich von Google 2014 für die Bereitstellung und Orchestrierung von Containern entwickelt wurde, aber 2015 an eine Tochter Foundation der Linux Foundation gespendet. Kubernetes kommt aus dem Griechischen und bedeutet Steuermann.. 1, 3, 11, 20, 69, 71, 73, 100, 118, 134, 137, 142, 143, 144, 147, 166, 169, 170

Linux Linux ist ein Open-Source-Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und löse vom UNIX Derivat MINIX inspiriert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 170

local-path-provisioner local-path-provisioner ist ein leichtgewichtiger Storage-Provider von Rancher. Er bietet den Applikationen einen persistenten Storage an.. 4, 96, 98, 141, 144, 147, 148, 152, cciv

MariaDB MariaDB ist ein MySQL Fork des ehemaligen MySQL Mitbegründers Michael Widenius, wobei sich der Name Maria aus dem Vornamen einer seiner Töchter ableitet. Nach dem Fork 2009 blieb MariaDB für eine Zeitlang sehr ähnlich mit MySQL und behielt ein ähnliches Versionierungsschema bei. Dies änderte sich 2012, wo dann direkt mit der Version 10 weitergefahren wurde. Beide Datenbanken entfernen sich im Lauf der Zeit immer mehr voneinander und sind nicht mehr in jedem Fall kompatibel oder beliebig austauschbar. Auf den Linux Distributionen trat MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 1, 5, 7, 11, 57

MetalLB MetalLB ist ein für Bare-Metal k8s Systeme ausgelegter Load-Balancer. Er kann sowohl auf Layer 2, mit OS-Boardmitteln arbeiten, bietet aber auch BGP-Routing an, so das Pods direkt von Routern angesteuert werden können, ohne via Host gehen zu müssen[40].. 73, 89, 93, 141, 144, 147, 148

Microsoft Azure SQL Database Microsoft Azure SQL Database oder auch Azure SQL ist eine relationale Datenbank, die von Microsoft für die Azure Cloud optimiert 2010 entwickelt wurde[52].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[53].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[54]. Nebst Microsoft Windows und Windows Server lässt es sich seit Version 2014 ebenfalls auf Linux Betreiben. In der Wirtschaft ist die primäre Plattform aber Windows Server.. 5, 7, 171

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zum ersten Mal 2007 veröffentlicht wurde[97].. 7

MySQL Die Datenbank MySQL wurde ursprünglich als reine relationale Open-Source Datenbank von Firma MySQL AB 1994 entwickelt. Der Name My leitet sich vom Namen My der Tochter des Mitbegründers Michael Widenius ab. Als Sun Microsystem 2008 MySQL übernahm, hielt sich die Option

frei, bei einem Kauf von Sun Microsyszem durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 1, 5, 7, 11, 57

NoSQL NoSQL steht für Not only SQL. Das heisst, Relationale Datenbanken haben komponenten wie Dokumentendatenbanken, Graphendatenbanken, Key-Value-Datenbanken und spaltenorientiert Datenbanken. Viele der grossen Datenbanklösungen wie Oracle Database oder Microsoft SQL Server sind NoSQL Datenbanken resp. bieten diese Option an.. 7, 168, 169, 170, 172, 173

OLAP Eine Online Analytical Processing, kurz OLAP, ist eine Multirelationale resp. Multidimensionale Datenbanklösung. Sie wird oft in Form eines Datenwürfels erklärt, kann aber auf verschiedene Arten umgesetzt werden[64, 63]. OLAP-Systeme bieten eine hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 7, 166

Oracle Linux Oracle Linux ist eine RHEL-Distribution der Firma Oracle und ist mit RHL Binärkompatibel. Sie wird primär für den Betrieb von Oracle Datenbanken verwendet und kommt auf den Oracle Eigenen Appliances ODA und Exadata zum Einsatz. Für den Zweck als DB Plattform kann ein für Oracle Datenbanken optimimierter Kernel verwendet werden. Zu Oracle Linux kann ein kostenpflichtiger Support bezogen werden, allerdings ist die Distribution anders als RHEL auch ohne Lizenz erhältlich.. 20

Oracle Database Die erste verfügbare Version der Oracle Datenbank kam im Jahr 1979 mit Version 2 (statt Version 1) heraus, damals allerdings nur mit den Basisfunktionen. Im Laufe der Zeit wuchs der Funktionsumfang sehr stark an, die Grundlage des Client-Server-Designs kam erstmals im Jahr 1985 mit Version auf den Markt und hat sich im Prinzip bis heute gehalten. Mit der mit Version 8/8i 1997 erschienen Optimizer und mit der Version 9i 2001 erschienen Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangenheit ermöglichen) konnte Oracle sich stark von der Konkurrenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um hochverfügbare Systeme, hohe Performance oder grosse Datenmengen geht.. 5, 7, 11, 44, 144, 147, 171

PKI Eine Public Key Infrastructure ist ein System, das für das Verschlüsseln und Signieren von Daten oder die Zertifizierung verwendet wird.. 118, 146, 147, 166

PostgreSQL Die OpenSource Datenbank PostgreSQL wurde in Form von POSTGRES zum ersten Mal 1986 von der University of California at Berkeley veröffentlicht und zählt zu den beliebtesten Open-Source Datenbanken. Zudem besteht in vielen Bereichen eine gewisse Ähnlichkeit zu Oracles Oracle Database.. 1, 5, 7, 11, 17, 44, 58, 63, 65, 68, 69, 73, 74, 135, 137, 139, 143, 145, 147

PostgreSQL HA Cluster Der HA Cluster des PostgreSQL Clusters. 19, 145

PostgreSQL Cluster Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wie Oracle.. 18, 19, 58, 68, 91, 135, 137, 145, 146, 171, lxxvi, lxxxiii, lxxxviii

PRTG Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum ersten Mal veröffentlicht und war ebenfalls als Netzwerkmonitoring System konzipiert. Wie bei Zabbix lässt sich heute damit ebenfalls fast jedes IT-System damit überwachen. Reichen die zahlreich vorhanden Standard Sensoren nicht, können eigene Sensoren geschrieben werden. PRTG ist nicht Open-Source, man bezahlt anhand gewisser Sensor Packages.. 4, 5, 18, 20, 138, 139, 140, 143, 150, 151, 166, iii, cxcviii, cciii

Quorum In verteilten Systemen resp. Cluster muss sichergestellt werden, dass bei einem Ausfall oder einer Netzwerkunterbrechung zwischen den Nodes es zu keiner Split-brain-Situation kommt. Hierzu wird i. d. R. ein Quorum verwendet. I. d. R. wird jener Teil des Quorums zum Primary oder alleinigen Node, der mit der Mehrheit aller Nodes vereint. Daraus ergeben sich bestimmte Größen, mit 5 Nodes braucht es 3 Nodes, um aktiv zu bleiben und mit 3 Nodes deren 2. Bei diesen Konstellationen wird daher darauf geachtet, eine ungerade Anzahl Nodes im Cluster zu halten, um keine Pat-Situation zu provozieren. Im Kapitel **Quorum** wird genauer auf die Mechanik eines Quorums eingegangen. . 58, 144, 168, 169, cciv

RDBMS Ein RDBMS ist ein Datenbankmanagementsystem für eine relationale Datenbank. Relationale Datenbanken sind Tabellenorganisierte Datenmodelle, die auf Relationen aufbauen, deren Schemata lassen sich normalisieren. Relationale Datenbanken müssen dabei auch Mengenoperationen, Selektion, Projektion und Joins erfüllen, um als Relationale Datenbanken zu gelten[58].. 74, 166

Red Hat Ansible Automation Platform Die Red Hat Ansible Automation Platform, ehemals Ansible Tower, ist eine Automatisierungsplattform für Ansible. Sie bietet eine mit Ansible gesteuerte CI/CD Umgebung an[45].. 121, 146

RedHat Enterprise Linux (RHEL) RHEL wurde in seiner ursprünglichen Form Red Hat Linux (RHL) bis in den Oktober 1994 zurück, wobei die erste Version von RHEL wie es heute existiert, im Jahr 2002 erfolgte. RHEL ist auf lange Wartungszyklen von fünf Jahren und Grosskunden ausgelegt. Ohne entsprechenden Supportvertrag kann keine ISO-Datei bezogen werden. Somit hebt sich RHEL stark von anderen Linux Distributionen ab.. 20

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, d. h. die Daten liegen Primär im Memory und nicht auf dem Storage[100]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 169

rke2 rke2 ist eine leichgewichtige Kubernetes Distribution, die alles Notwendige mitbringt, um einen k8s Cluster zu betreiben[33].. 3, 81, 88, 93, 101, 141, 144, 147, 148

Rocky Linux Rocky Linux basierte auf der offen zugänglichen Linux Distribution CentOS welche RHEL Binärkompatibel war und gilt als inoffizieller Nachfolger von CentOS.. 20

SAN Ein Storage Area Network ist ein dediziertes Netzwerk aus Storage Komponenten. SAN Systeme bieten redundante Pools an Speicher. Die physischen Festplatten werden zu virtuellen Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[56, 86, 65]. 5, 20, 25, 104, 166

SIEM Ein sammelt Daten aus verschiedenen Netzwerkkomponenten oder Geräten von Agents oder Logs. Diese Daten werden permanent analysiert und mit einem definierten Regelwerk gegengeprüft. Ziel

ist es, verdächtige Events zu erkennen und einem Angriff zuvorzukommen oder ihn möglichst früh zu unterbinden[101].. 20, 118, 130, 146, 147, 166

Snowflake Snowflake ist eine Big Data Plattform die Data Warehousing, Data Lakes, Data Engineering und Data Science in einem Service vereint. Die Daten werden in eigenen internen Relationalen und NoSQL-Datenbanken gespeichert[103, 94]. 7

Split-brain Im Kapitel ?? werden die Ursachen und folgenden eines Split-brains genauer besprochen. . 43, 172

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[74, 80]. . 7

SQLite SQLite ist eine Relationale Embedded Datenbank welche seit 2000 existiert. Sie verzichtet auf eine Client-Server-Architektur und kann in vielen Frameworks eingebunden werden[98].. 7

State Machine Eine State Machine resp. Endlicher Automat ist eine abstrahierte Maschine, die eine endliche Anzahl von Zuständen hat, in die gewechselt werden können oder die erreicht wurden. Dadurch ist eine State Machine jederzeit in der Lage, den eigenen Zustand zu kennen und entsprechende Aktionen während des Zustand oder beim Wechseln in einen anderen Zustand auszuführen.. cciii, cciv

Switchover In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geswitched.. 19, 120

SWOT-Analyse Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzeigen. Anhand einer SWOT-Analyse werden anschliessend Strategien abgeleitet, um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder abzumildern.. 166

Terraform Terraform ist ein Werkzeug für die Verwaltung von Infrastruktur mit Software zu steuern, sogenanntes Infrastructure as Code. Terraform wird sehr oft dafür benutzt um Container- und Cloudinfrastruktur ansteuern und verwalten zu können[99, 72].. 20

Transaktion Eine Transaktion ist beinhaltet Schreib-, Lese-, Mutations- oder Löschoperationen auf Daten.. 54

UNIX Die erste Version von UNIX wurde im Jahr 1969 in den Bell Labs entwickelt und übernahm viele Komponenten aus dem gescheiterten Multics-Projekt. Aus dem ursprünglichen UNIX entstanden im Laufe der Zeit viele offene und Proprietäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 166

VMware vSphere Die vSphere® ist ein Typ-1 Hypervisor der Firma VMware® der eine Reihe leistungsfester Tools und Funktionen mitbringt.. iii, xlvii, lxxxv

VRRP VRRP ist ein Protokoll, mit welchem Fallback-Router bereitgestellt werden können. Fällt der primäre Router aus, so kommt der Fallback-Router zum Einsatz. Dank Gewichtung wird der Primäre Router wieder aktiv, sobald er wieder erreichbar ist[48, 1].. 166, 169

Zabbix Das 2001 veröffentlichte Open-Source Monitoring System Zabbix gilt zwar als Netzwerk-Monitoring System, allerdings kann heute nahezu jedes IT-System damit überwacht werden. Zabbix speichert die Metriken und nicht die Auswertungen, das heißtt, solange die Daten vorhanden sind, können Grafiken zu jedem Zeitpunkt generiert werden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge abschliessen.. 11, 20

Selbstständigkeitserklärung

Ich versichere, dass die vorliegende Arbeit von den Autoren selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Alle Inhalte dieser Arbeit dazu gehören neben Texten auch Grafiken, Programmcode, etc., die wörtlich oder sinngemäss aus anderen Quellen stammen, sind als solche eindeutig kenntlich gemacht und korrekt im Quellenverzeichnis gelistet. Dies gilt auch für einzelne Auszüge aus fremden Quellen.

Die Arbeit ist in gleicher oder ähnlicher Form noch nicht veröffentlicht und noch keiner Prüfungsbehörde vorgelegt worden.

Ort, Datum, Unterschrift

Haftungsausschluss

Der vorliegende Bericht wurde von Studierenden im Rahmen einer Diplomarbeit erarbeitet. Es muss an dieser Stelle darauf hingewiesen werden, dass die Arbeit nicht im Rahmen eines Auftragsverhältnisses erstellt wurde. Weder der Ersteller noch die ibW Höhere Fachhochschule Südostschweiz können deshalb für Aktivitäten auf der Basis dieser Diplomarbeit eine Haftung übernehmen.

Disposition

Disposition Diplomarbeit



7

Bewilligung (wird durch die Schulleitung ausgefüllt)

Das Thema dieser Diplomarbeitsdisposition ist bewilligt.

Ort und Datum

Unterschrift Fachvorsteher

Sargans, 4.12.2023

Anmerkungen und Hinweise zur Disposition:

Auf die folgenden Punkte ist bei der Bearbeitung der Diplomarbeit Rücksicht zu nehmen:

Legende:

- ! Hinweis auf kleine Ungereimtheiten*
- !! Grössere Schwächen, Fehler oder Unklarheiten*
- !!! gravierende Schwäche, kann so nicht verwendet werden*

2 Ziele und Nutzen

!! Bei den zu erreichenden Zielen sind Tasks nach Wichtigkeit und Priorität eingestuft worden. Hier gibt es noch ungereimtheiten zwischen Wichtigkeit und Prriorität. Als Beispiel wird ein Task als Tief eingestuft und gleichzeitig als Muss Priorität gesetzt.

3 Abgrenzungen

!! Die Systemabgrenzungsgrafik hilft nicht beim Verständnis deiner Arbeit. Die Abgrenzung deiner Arbeit sollte klar und präzise formuliert werden. Im Moment ist nicht klar, wo genau die Abgrenzungen sind. Die Grafik kann als zusätzliche Information beigefügt werden, wenn sie nicht zu überladen ist und Begriffe aufweist, die in der Arbeit nicht weiter erwähnt werden.

4 Abhängigkeiten und Risiken

! Wie genau hilft eine SWOT-Analyse bei der Zielerreichung? Wenn eine Swot Analyse eingesetzt wird, muss die Erkenntnis daraus in die Arbeit einfließen und einen klaren Mehrwert ausweisen.

Datum	Von	Bis	Dauer [h]	Phase	Subphase	Tätigkeit	Bemerkung	Schwierigkeit	Lösungen
14.02.2024	19:00	20:00	1.0	1. Expertengespräch	1. Expertengespräch				
21.02.2024	15:00	16:00	1.0	Evaluation	Anforderungskatalog	Anforderungskatalog erarbeiten			
22.02.2024	16:00	17:30	1.5	Evaluation	Anforderungskatalog	Anforderungskatalog erarbeiten			
27.02.2024	10:00	11:30	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
27.02.2024	13:00	16:00	3.0	Dokumentation	Dokumentation	Dokumentation erweitern	Viele LaTeX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken	
28.02.2024	09:00	11:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern	Viele LaTeX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken	
01.03.2024	07:00	09:00	2.0	Dokumentation	Dokumentation	Dokumentation Exkurs Architektur	Um Entscheidungen Transparent zu machen, müssen Grundlegende Konzepte aufgezeigt werden. Nicht alle Konzepte wie z.B. Distributed SQL sind bekannt resp. das Zusammenspiel mit Kubernetes.	Konzepte wie Distributed SQL sind nicht einfach zu erklären.	
08.03.2024	07:00	09:00	2.0	Evaluation	Anforderungskatalog	Anforderungskatalog erarbeiten			
11.03.2024	07:00	11:30	4.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Informationen Sammeln	pgpool II	pgpool II hat kein GitHub Repository. Das macht es unmöglich, diese Lösung mit all den anderen zu vergleichen.	pgpool II fällt somit direkt aus der Betrachtung raus, da kein Vergleich möglich ist.
11.03.2024	12:00	13:30	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
11.03.2024	16:45	17:30	0.5	Dokumentation	Dokumentation	Dokumentation erweitern	Stakeholder erfassen		
13.03.2024	17:45	19:45	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Stackgres und Citus analysieren	Citus row-based-sharding	Citus Dokumentation stark Textlastig. Wenig Abbildungen, vieles muss selber gezeichnet werden.	
14.03.2024	19:45	20:45	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen		Citus row-based-sharding		
14.03.2024	20:45	21:30	0.8	Dokumentation	Dokumentation	Projektcontrolling Arbeiten	Citus row-based-sharding Dokumentieren		
16.03.2024	17:45	18:30	0.8	Dokumentation	Dokumentation	Dokumentation erweitern	Zweiter Statusbericht verfassen		
17.03.2024	14:45	16:30	1.8	Dokumentation	Dokumentation	Dokumentation erweitern	ACID Exkurs erfassen		
17.03.2024	19:30	20:00	0.5	Dokumentation	Dokumentation	Dokumentation erweitern	Listings sauber machen.		
17.03.2024	20:15	21:00	0.8	Dokumentation	Dokumentation	Dokumentation erweitern	Neue Listing-Sprache für yaml-Files erstellt, da noch einige folgen werden.		
18.03.2024	14:00	16:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern	Statusbericht 2 fertig Schreiben und Mail an Norman Süsstrunk senden		
18.03.2024	20:20	21:50	1.5	Evaluation	Vorbereitung Benchmarking	pgbench analysieren	Percona ist Dein Freund		
19.03.2024	08:00	10:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb			
19.03.2024	10:00	10:30	0.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Backup Anbindungen		Veeam Kast K10 wird nicht vor Angabe Diplomarbeit fertig sein	Backups lokal speichern. Veeam Integration wird im Nachgang implementiert.
19.03.2024	14:00	16:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern	yugabytedb		
20.03.2024	16:00	16:15	0.2	Dokumentation	Dokumentation	Termin für 2. Fachgespräch organisieren			
21.03.2024	18:00	20:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern	Projektcontrolling gemacht.		
22.03.2024	09:00	11:00	2.0	Evaluation	Vorbereitung Benchmarking	zabbix analysieren			
22.03.2024	13:00	14:30	1.5	Evaluation	Vorbereitung Benchmarking	Benchmark Settings setzen			
22.03.2024	14:30	15:30	1.0	Dokumentation	Dokumentation	Dokumentation erweitern	Projektcontrolling und Dokumentation		
22.03.2024	16:45	19:00	2.8	Dokumentation	Dokumentation	Dokumentation erweitern	Analyse gängiger PostgreSQL HA Cluster Lösungen - Patroni, Stackgres, CloudNativePG dokumentieren / Benchmarking		
24.03.2024	14:30	17:30	3.0	Dokumentation	Dokumentation	Dokumentation erweitern	Analyse gängiger PostgreSQL HA Cluster Lösungen - Patroni, Stackgres, CloudNativePG dokumentieren		
24.03.2024	19:30	22:30	3.0	Dokumentation	Dokumentation	Dokumentation erweitern	Analyse gängiger PostgreSQL HA Cluster Lösungen - Patroni, Stackgres, CloudNativePG dokumentieren / Arbeitsrapport		
25.03.2024	08:00	11:00	3.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	rke2 - local-path-provisioner installieren			
25.03.2024	13:00	14:45	2.8	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb	Anforderungen recht hoch. Es wird ein guillemotleftprivate container registry guillemotright verlangt.	Eine mögliche Lösung könnte sein, rke2 als Registry zu verwenden.	
26.03.2024	12:00	13:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb Installation			
26.03.2024	14:45	15:00	0.2	2. Expertengespräch	2. Expertengespräch			Norman verspätete sich wegen eines Privaten Notfalls.	Termin wird auf morgen verschoben
26.03.2024	15:00	16:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb Installation	Aus versehen YugabyteDB Anywhere (Repository yugaware) installiert.	YugabyteDB (Repository yugabyte) verwenden.	
27.03.2024	10:00	11:30	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb Installation	Diese Installation benötigt zwingend eine Subscription.	Dies ist nach wie vor Open-Source	
27.03.2024	15:00	15:30	0.5	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	MetalLB Installation	Wäre ein No-Go		
27.03.2024	13:00	16:00	4.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	MetalLB Troubleshooting			
27.03.2024	16:30	17:30	1.0	2. Expertengespräch	2. Expertengespräch				
01.04.2024	09:00	09:45	0.8	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmarking	Versuch, das Benchmarking via Cronjobs auszuführen	ysql_bench hat keinen Passwort-Parameter. Ist zudem zu gut geschützt um mit Tricks das Passwort zu übergeben.	Manuell alle 10min ausführen.
01.04.2024	14:00	16:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmarking	Viel Zeit verloren für das manuelle Benchmarking		
02.04.2024	10:00	12:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation	StackGres verfolgt ein anderes Konzept als Yugabyte.		
03.04.2024	09:00	11:30	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation			
03.04.2024	14:00	15:30	0.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation	Installation von Patroni begonnen.		apt-Proxy setzen
09.04.2024	12:30	14:00	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation			
10.04.2024	10:00	11:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation			
12.04.2024	12:30	14:45	2.2	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation			
15.04.2024	18:00	20:00	2.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	rke2 - local-path-provisioner 250GiB	Versuch, grosse Volumes einzubinden	Extension Server nicht erreichbar	Extension Server nicht erreichbar
16.04.2024	13:00	16:00	3.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	rke2 - local-path-provisioner 250GiB	Versuch, grosse Volumes einzubinden	etcd-Server bereitet Probleme	etcd-Server bereitet Probleme
17.04.2024	10:00	11:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmarking / Testing	Grosse Volumes testen	Extension Server nicht erreichbar	Extension Server nicht erreichbar
17.04.2024	12:30	19:00	6.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Deployment / Testing			Proxy gesetzt und http erzwungen
19.04.2024	08:00	11:00	3.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Benchmarking	Auch pgbench muss manuell ausgeführt werden		
19.04.2024	12:00	14:00:00:00	2.0	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	Analyse vitabaks/postgresql_cluster// Architektur	Analyse von vitabaks/postgresql_cluster// Architektur	Analyse von vitabaks/postgresql_cluster auf GitHub.	Node Annotations auf local-path-provisioner und StorageClass setzen
19.04.2024	14:00	20:00	6.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation / Testing	Aufbau analog ursprüngliche Evaluationsplattform von Patroni.	Aufbau analog ursprüngliche Evaluationsplattform von Patroni.	Vereinfachen soweit möglich
19.04.2024	20:00	21:00	1.0	Aufbau und Implementation Testsystem	Basisinfrastruktur	Patroni Test Server DMT / Auftrag	Architektur entsprechend zeichnen.	etcd-Server bereitet Probleme	
20.04.2024	10:00	12:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking	DMT Einträge für Patroni Testserver erstellt.		
20.04.2024	14:00	16:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking / grosse Volumes	Ticket an Niculin Fürst für Foreman Job.		
22.04.2024	09:00	09:15	0.2	Evaluation	Gegenüberstellung		Auf erweiterte Disk umstellen und letzten Benchmark fahren		
22.04.2024	09:30	11:00	1.5	Dokumentation	Dokumentation				
23.04.2024	13:45	14:00	0.2	Evaluation	Gegenüberstellung				
24.04.2024	15:00	17:00	2.0	Dokumentation	Dokumentation				
25.04.2024	14:00	16:00	0.0	Evaluation	Gegenüberstellung				
26.04.2024	13:30	14:30	1.0	Evaluation	Variantenentscheid				
26.04.2024	14:30	17:30	3.0	Dokumentation	Dokumentation	Dokumentation erweitern	iii Prequerries umgesetzt. Snapshots auf VMware vSphere gesetzt.		
29.04.2024	07:00	09:00	2.0	Aufbau und Implementation Testsystem	Basisinfrastruktur				
29.04.2024	09:00	11:00	2.0	Aufbau und Implementation Testsystem	Basisinfrastruktur				

III Protokoll - Fachgespräche

III.I Fachgespräch 14.02.2024

Kantonsspital Graubünden
Departement ICT
Michael Gruber
Datenbank Administrator
Gäuggelistrasse 7
CH-7000 Chur

Standort Informatik
Tel. +41 (0)81 256 68 25
michael.gruber@ksgr.ch
www.ksgr.ch



Protokoll

DIPLOMARBEIT - VERNISSAGE

Dienstag, 14. Februar 2024
18:00 – 21:15 Uhr

Zoom BreakoutSession Nr. 5 / 9

Chur, 13. Mai 2024

Teilnehmer	Bereich od. Funktion	Stellvertretung
Norman Süsstrunk (Vorsitz)	Fachexperte	
Curdin Roffler	Studierender	
Michael Gruber	Studierender	

Beratend ohne Stimmrecht

Protokoll
Michael Gruber

Gäste

Traktanden

	1.	Begrüßung
	2.	Termine
	3.	Coaching
	4.	Administratives
	5.	Fragen und Antworten
		Vorauswahl
	6.	Varia / Aktualitäten
	6.1	Zusammenlegung Breakout Session

I = Informationen

D = Diskussion

E = Entscheid

A = Auftrag

Traktanden	Art	Verantw.	Termin
1. Begrüssung Kurze Vorstellungsrunde von Norman Süsstrunk, Curdin Roffler und Michael Graber	I		
2. Termine 2.1 Flexible Termingestaltung möglich 2.2 Norman braucht eine Woche Vorlaufszeit 2.3 Die Initiative für Expertengespräche soll von den Studenten kommen. Norman wird sich aber spätestens zur Halbzeit melden.	I I A	Curdin Michael	
3. Coaching 2.1 Kriterien müssen klar Aufgenommen werden und Grob gefiltert werden 2.2 Bei Fragen zur Diplomarbeit können die Studenten jederzeit um eine Coaching Session bitten 2.3 Früh genug mit dem Verfassen des Berichts anfangen. Sonst wird die Zeit zu knapp	A A	Curdin Michael Curdin Michael	
4. Administratives 4.1 Spätestens zur Halbzeit muss ein Status an Norman gesendet werden 4.2 Das Intervall des Statusreports muss KSGR Intern abgeklärt werden	A A	Curdin Michael Curdin Michael	
5. Fragen und Antworten 5.1 Darf eine Vorauswahl stattfinden, um den Aufwand zur reduzieren? Eine Vorauswahl ist Sinnvoll und in diesem Rahmen fast zwingend Notwendig, da sonst viel Zuviel Zeit investiert werden müsste	D	Norman Michael	
6. Varia / Aktualitäten 6.1 Zusammenlegung Breakout Session Die Breakout Sessions 5 (mit Curdin Roffler) und 9 (mit Michael Graber) wurden der Einfachheit halber zusammengelegt.			

Pendenzen

Pendenzenliste

Nächster Termin:

Name der Sitzung	
Einreichung Traktanden inkl. Beilagen	
Versand Einladung	

Norman Süsstrunk
Experte
(Vorsitzender)

Michael Graber
Qualifikant
(Protokoll)

Kantonsspital Graubünden
Departement ICT
Michael Graber
Datenbank Administrator
Gäuggelistrasse 7
CH-7000 Chur

Standort Informatik
Tel. +41 (0)81 256 68 25
michael.graber@ksgr.ch
www.ksgr.ch

Protokoll

Sync Diplomarbeit

Mittwoch, 27. März 2024
16:30 – 17:30 Uhr

MS Teams

Chur, 7. April 2024

Teilnehmer	Bereich od. Funktion	Stellvertretung
Norman Süsstrunk (Vorsitz)	Experte	
Michael Graber	Qualifikant	

Beratend ohne Stimmrecht

Protokoll
Michael Graber

Gäste

Traktanden

	1.	Begrüssung
	2.	Fragen und Antworten
	2.1	Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden?
	2.2	Hast Du noch Vorschläge zu PostgreSQL HA Clustern gefunden?
	2.3	Soll ich die Gewichtung mit 100 Punkten oder 1000 machen?
	2.4	Soll die Disposition in den Anhang? Diese ist über 50 Seiten lang?
	2.5	Falls ich die Diplomarbeit zwecks Gegenlesen / Rechtsschreibbeprüfung geben würde, müsste ich dies irgendwo angeben da Fremdleistung?
	3.	Inputs Norman Süsstrunk
	3.1	Siehe Details
	4.	Varia / Aktualitäten
	4.1	...

I = Informationen

D = Diskussion

E = Entscheid

A = Auftrag

Traktanden	Art	Verantw.	Termin
1. Begrüssung	I	Michael	
2. Fragen und Antworten 2.1 Frage: Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden? Gut wäre es, binnen ein bis zwei Wochen zuzusenden. 2.2 Hast Du noch Vorschläge zu PostgreSQL HA Clustern gefunden? Norman hat nicht sehr viel Erfahrung. Hat bisher keine Erfahrung z.B. CloudNativePG. Hat mit einem Team mit Wien auf On-Premises gearbeitet für Non-HA Lösungen PostgreSQL bietet ja von Haus aus keine HA-Lösung 2.3 Soll ich die Gewichtung mit 100 Punkten oder 1000 machen? Mit Hundert Punkten arbeiten. Ist oft eine grobe Abschätzung. Input / Vorschlag: Anforderungen reduzieren auf die wichtigsten. Sonst wird die Zeit knapp. Vorschlag: Durchgehen und vereinfachen und Norman vereinfachte Version senden 2.4 Soll die Disposition in den Anhang? Diese ist über 50 Seiten lang? Disposition nicht in den Anhang setzen. Disposition ist ja Teil der Diplomarbeit 2.5 Falls ich die Diplomarbeit zwecks Gegenlesen / Rechtsschreibprüfung geben würde, müsste ich dies irgendwo angeben da Fremdleistung? Nein, muss nicht angegeben werden. Manchmal werden Personen, die z.B. Mithilfe bei der Korrektur gemacht haben, im Vorwort mit einem Dank bedacht. Es gibt aber keine Formale Anforderungen Aufträge / Kommunikation / nächste Schritte <ul style="list-style-type: none">Vereinfachen der Präferenzmatrix	D D D/A	Michael	
3. Inputs Norman Süsstrunk 3.1 Rechtschreibung Mit Rechtschreibprogramm Prüfen. Mehr Typos, dafür viele. Gäbe eine Ungenügend mit den Fehlern. 3.2 Sprache Nicht zu komplex geschrieben. Technisch beschrieben. 3.3 Inhalt <ul style="list-style-type: none">Sehr viele Sachen drin, die nicht wirklich mit dem Thema zu tun haben			

Traktanden	Art	Verantw.	Termin
<p>Bei der Einleitung => Kenndaten des KSGR. Drin lassen. Hat nichts mit dem Thema zu tun, Nicht noch mehr rein tun, muss nicht raus.</p> <ul style="list-style-type: none"> • Auflisten der DBs müssten nicht rein • Weniger ist mehr. Mehr auf den Punkt kommen • Kondensiert machen mit den wichtigsten Listenings. Es muss danach das System anhand der Doku gebaut werden können • Doku so machen wie man sie macht, nur etwas hübscher für die Arbeit <p>3.4 Kurze Präsentation YugabyteDB</p> <ul style="list-style-type: none"> • Kurz erklärt, wie YugabyteDB funktioniert • Aktuellen Stand der Evaluation präsentiert <p>Aufträge / Kommunikation / nächste Schritte</p> <ul style="list-style-type: none"> • ACID aus dem Exkurs Architektur rausnehmen • Prüfen, ob Dokumentation etwas kompakter wird 		Michael	
<p>4. Varia / Aktualitäten</p> <p>4.1 Protokoll Protokoll verfassen</p> <p>Aufträge / Kommunikation / nächste Schritte</p> <ul style="list-style-type: none"> • Protokoll genehmigen 	A	Michael	
	A	Norman	

Pendenzen
<u>Pendenzenliste</u>
<ul style="list-style-type: none"> • Protokoll binnen 1-2 Wochen • Vorschlag reduzierte Präferenzmatrix

Nächster Termin:	
Name der Sitzung	
Einreichung Traktanden inkl. Beilagen	
Versand Einladung	

Norman Süsstrunk
 Experte
 (Vorsitzender)

Michael Graber
 Qualifikant
 (Protokoll)

IV

Statusbericht

IV.I

Status Report 1

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 13.02.2024	
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	PMA
ICT veratw. Person	Michael Graber	-	
Status	Ampel	Tendenz	Begründung
Gesamtprojekt			
Zeitplanung	🔴	⬇️	Projekt ist Umfangreich und hat viele Teilspekte, die es zu planen und berücksichtigen gilt.
Ressourcen	⚠️	⬇️	Parallel läuft das Grossprojekt "Erneuerung HP UX Plattform", wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.
Kosten	🟢	➡️	Kosten sind noch im Soll-Bereich
Tätigkeiten vergangene Berichtsperiode		Tätigkeiten nächste Berichtsperiode	
<ul style="list-style-type: none"> - Dokumentenstruktur erstellt - Projektplanung erstellt - Vernissage vorbereitet - Statusbericht erstellt 		<ul style="list-style-type: none"> - Anforderungskatalog erarbeiten - Vorbereitung Benchmarking 	
# nächste Lieferobjekte (inkl. allfällige Links)		Status	Erliegdungsgrad
LO-001	Anforderungskatalog	in arbeit	<div style="width: 60%;">60%</div>
LO-002	Vorbereiten Benchmarking	in arbeit	<div style="width: 0%;">0%</div>
LO-003			23.02.2024
LO-004			26.02.2024
LO-005			
# Risiken	Auswirkungsgrad	Massnahmen	Verantw.
R-001	🟡	Organisation und Selbstmanagement	
R-002	🔴	Ressourcen reservieren	
R-003	🔴	Monitoring vorgängig ausbauen und Massnahmen definieren	
R-004	🟡	Werkzeuge im Vorfeld definieren und bereitstellen	
Kostenübersicht		Abhängigkeiten zu anderen Projekten	Massnahmen
Verfügbare Finanzen bis Ende Projekt: $\frac{100 \text{ CHF}}{\text{A}} + 2000 = 24.000 \text{ CHF}$		Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) => Foreman Umgebung	Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten
Bemerkungen / Informationen		Anträge	
Eingereicht	Geprüft	Bemerkungen/Auftrag PMO	
PL:	PMO:		
Datum:	Datum:		
# erledigte Lieferobjekte (inkl. allfällige Links)			

Status

Dokumentenstruktur und Dokument wurde mit **LATEX** angelegt. Aus der Disposition wurde die Ausgangslage, das Riskmanagement, die Abgrenzung, die Zieldefinition sowie Abkürzungen und das Glossar übernommen. Nicht aus der Disposition wurde die SWOT-Analyse übernommen, da diese mehr Fragen aufwerfen denn Antworten liefern würde.

Mit MS Project 2016 wurde die Projektplanung erstellt. Die Struktur richtet sich weitestgehend nach der Dokumentenstruktur.

Risikomanagement

HP-UX Ablösung ExaCC-Projekt

Das grösste Risiko besteht nach wie vor im **Parallelen** Grossprojekt der HP-UX Ablösung. Dieses Projekt wird ab Mitte-Ende Februar eine nicht unerhebliche Menge an Ressourcen für die Architektur, Planung, Migrationsplanung usw., absorbieren.

Als Lösung für dieses Problem, werden **Firmenintern** Ressourcen reserviert. **Nichtsdesto trotz** besteht **gefahr** bezüglich der Priorisierung der beiden **Projekte**.

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, das es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Ein möglicher Ansatz besteht darin, die gängigsten Monolithischen und Distributed SQL Systeme vorzusortieren. Daraus die besten drei (zwei Distributed SQL und ein Monolithisches) genauer zu evaluieren. Dies ist ein Thema, das beim ersten Fachgespräch von meiner Seite traktandiert wird.

Weiteres vorgehen

Als **nächstest** steht die **erarbeitung** des Anforderungskatalog an.
Diese werden bei den internen Stakeholdern abgeholt.

Anschliessend wird anhand des Anforderungskatalog das Benchmarking vorbereitet.
So das die Evaluation mit sauberen Messdaten vonstatten gehen kann.

Anhand des Benchmarkings und der Anforderungen soll im Anschluss ein Testszenario für die Evaluation abgeleitet werden.

IV.II

Status Report 2

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 18.03.2024	
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	PMA
ICT vertrat. Person	Michael Graber	-	
			
Status	Ampel	Tendenz	Begründung
Gesamtprojekt			In vorzg. Grossprojekt Erneuerung HP UX Plattform nimmt viel Zeit in Anspruch. Hinzu kommt, das die Analyse gängiger PostgreSQL HA Lösungen ebenfalls viel Zeit kostet. Dokumentationsaufwand unterschätzt.
Zeitplanung			
Ressourcen			Parallel läuft das Grossprojekt Erneuerung HP UX Plattform, wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.
Kosten			Kosten sind noch im Soll-Bereich
Tätigkeiten vergangene Berichtsperiode		Tätigkeiten nächste Berichtsperiode	
<ul style="list-style-type: none"> - Anforderungskatalog erstellt - Parallel dokumentiert 		<ul style="list-style-type: none"> - Analyse der PostgreSQL HA Clusterlösungen abgeschlossen - Benchmarking abgeschlossen - Variantenentscheid getroffen - Basisystem für Testsystem aufgebaut 	
# nächste Lieferobjekte (inkl. allfällige Links)		Status	Erledigungsgrad
LO-002 Vorbereiten Benchmarking LO-003 Analyse PostgreSQL HA Cluster Lösungen LO-004 Gegenüberstellung LO-005 Variantenentscheid LO-006 Aufbau Basisinfrastruktur Testsystem		offen	
		in Arbeit	
		offen	
		offen	
		offen	
# Risiken		Auswirkungsgrad	Massnahmen
R-001 Fehlende Ressourcen			Organisation und Selbstmanagement
R-002 HP-UX Ablöseprojekt			Ressourcen reservieren
R-003 Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden			Monitoring vorgängig ausbauen und Massnahmen definieren
R-004 Schwächen beim Selbstmanagement und in der Selbstorganisation			Werkzeuge im Vorfeld definieren und bereitstellen
R-005 Scope Verlust während des Projekts			Ziele klar definieren
R-006 Scope Creep			Ziele SMART definieren
R-007 SIEM / Log Plattform nicht betriebsbereit			Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten
R-008 Foreman nicht betriebsbereit			
Kostenübersicht		Abhängigkeiten zu anderen Projekten	
Verfügbare Finanzen bis Ende Projekt: 100 CHF h ⁻¹ * 200 = 20'000 CHF		Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) => Foreman Umgebung	
		Massnahmen	
		Ressourcen reservieren	
		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Bemerkungen / Informationen		Anträge	
Eingereicht		Bemerkungen/Auftrag PMO	
PL:	Geprüft	Bemerkungen/Auftrag PMO	
Datum:	Datum:	Datum:	
# erledigte Lieferobjekte (inkl. allfällige Links)			
LO-001 Anforderungskatalog			

Table 1: Zweiter Statusbericht

Status

Der Anforderungskatalog und die **gewichtung** wurde erstellt. Die Stakeholder wurden entsprechend eingebunden.

Analyse der bestehenden PostgreSQL HA Cluster **lösungen** begonnen. Die Analyse ist allerdings komplex und nimmt mehr Zeit in Anspruch, als erwartet.

Im Moment ist das Projekt in Verzug.

Risikomanagement

HP-UX Ablösung ExaCC-Projekt

Das grösste Risiko besteht nach wie vor im **Parallelen** Grossprojekt der HP-UX Ablösung. Dieses Projekt wird ab Mitte-Ende Februar eine nicht unerhebliche Menge an Ressourcen für die Architektur, Planung, Migrationsplanung usw., absorbieren.

Anfang April werden die beiden Nodes geliefert, entsprechend wird sich der Aufwand Ende April massiv erhöhen.

Als Lösung für dieses Problem, ist nach wie vor die Reservation von Terminen im KSGR am Zug. Dokumentationen werden aber auch am Abend und am Wochenende geschrieben.

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, dass es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Es wurde eine Vorsortierung der gängigen Methoden vorgenommen.

Weiteres vorgehen

Die Analyse muss fertiggestellt werden. Die Architektur und die Server für die Evaluationssysteme stehen und sind bereit für Installation von rke2 usw. Damit müssen noch die Architektur für das **Monolithische System** erstellt werden und dann der Aufbau der Evaluations-DBs in Angriff genommen werden.

Anschliessend wird anhand des Anforderungskatalog das Benchmarking vorbereitet.
So **das** die Evaluation mit sauberen Messdaten vonstatten gehen kann.

Anhand des Benchmarkings und der Anforderungen soll im Anschluss ein Testszenario für die Evaluation abgeleitet werden.

IV.III

Status Report 3

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 29.04.2024		
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	-	
ICT vertrw. Person	Michael Graber	PMA	-	
Gesamtprojekt				
Zeitplanung			Stark im Verzug. Evaluation benötigte weitaus mehr Zeit, als geplant.	
Ressourcen			Parallel läuft das Grossprojekt Erneuerung HP UX Plattform, wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.	
Kosten			Kosten sind noch im Soll-Bereich	
Tätigkeiten vergangene Berichtsperiode				
- Evaliert - Variantenentscheid getroffen - Parallel dokumentiert	Tätigkeiten nächste Berichtsperiode		- Basisystem für Testsystem aufgebaut - Testsystem aufgebaut - Testsystem geprüft und getestet	
# nächste Lieferobjekte (inkl. allfällige Links)		Status	Friedlungsgrad Soll Datum	
LO-006	Aufbau Basisinfrastruktur Testsystem	In Arbeit		Soll Datum
LO-007	Installation Patroni PostgreSQL Cluster	offen		
LO-008	Technical Review Testsystem	offen		
LO-009	Testing	offen		
LO-010	Resultate	offen		
# Risiken				
R-001	Fehlende Ressourcen		Organisation und Selbstmanagement	
R-002	HP-UX Ablöseprojekt		Ressourcen reservieren	
R-003	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden		Monitoring vorgängig ausbauen und Massnahmen definieren	
R-004	Schwächen beim Selbstmanagement und in der Selbstorganisation		Werkzeuge im Vorfeld definieren und bereitstellen	
R-005	Scope Verlust während des Projekts		Ziele klar definieren	
R-006	Scope Creep		Ziele SMART definieren	
R-007	SIEM / Log Plattform nicht betriebsbereit			
R-008	Foreman nicht betriebsbereit		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Kostenübersicht				
Verfügbare Finanzen bis Ende Projekt: 100 CHF h ⁻¹ * 200 = 20'000 CHF		Abhängigkeiten zu anderen Projekten	Massnahmen	
		Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) = Foreman Umgebung	Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Bemerkungen / Informationen				
Eingereicht	Geprüft	Bemerkungen/Auftrag PMO		
PL:	PMO:			
Datum:	Datum:			
# erledigte Lieferobjekte (inkl. allfällige Links)				
LO-001	Anforderungskatalog			
LO-002	Vorbereiten Benchmarking			
LO-003	Analyse PostgreSQL HA Cluster Lösungen			
LO-004	Gegenüberstellung			
LO-005	Variantenentscheid			

Status

Die PostgreSQL HA Lösungen wurden evaluiert.

Es wurden drei Systeme installiert:

- YugabyteDB
- StackGres - Citus
- Patroni

Die Analyse hat mehr Zeit in Anspruch genommen als geplant.

Es tauchten einige **Probleme** auf, welche nur bedingt mit der Arbeit zu tun haben.

Projekt stark im Verzug.

Risikomanagement

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, dass es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Es wurde eine Vorsortierung der gängigen Methoden vorgenommen.

Weiteres vorgehen

Testsystem wird installiert.

Anschliessend wird das System kurz einem **review** unterzogen und getestet.

Im Anschluss wird das Projekt abgeschlossen.

13. Mai 2024

Abschlussbericht Projekt

PostgreSQL HA Cluster - Konzeption und Implementation

Projektcoach	Klicken oder tippen Sie hier, um Text einzugeben.		Projektleiter/-in	Michael Gruber
			Co- Projektleiter	Klicken oder tippen Sie hier, um Text einzugeben.
Antragsteller	Michael Gruber		KST	P Klicken oder tippen Sie hier, um Text einzugeben.
Departement	Departement 10			
PM-Auftragsnr.	OTRS549108			Klicken oder tippen Sie hier, um Text einzugeben.
Projekt - Art	ICT - Hard- und Software			
Gesamtprojekt Start/Ende	Von 14.02.2024 bis 24.05.2024	<input type="checkbox"/> wie geplant <input checked="" type="checkbox"/> ursprünglich anders geplant <small>(kurze Erklärung bei Termine eingehalten)</small>		Abschluss
Initialisierung AIP Board	Klicken oder tippen Sie, um ein Datum einzugeben.	Start Realisierung	01.05.2024	Klicken oder tippen Sie, um ein Datum einzugeben.
Inbetriebnahme / Übergabe an Betrieb der Beschaffung / des Projekts				Klicken oder tippen Sie hier, um Text einzugeben.
Equipment-Nr. / Inventar-Nr. (anzugeben, wenn Anlagen in Betrieb genommen wurden)				Klicken oder tippen Sie hier, um Text einzugeben.
Welcher Kostenstelle wird der künftig Betriebsunterhalt belastet?				10890 Informatik
PMA der Betriebskosten				Klicken oder tippen Sie hier, um Text einzugeben.

Welche Ziele wurden im Projekt erarbeitet/erreicht?

Plan-Ziele	Ziele erreicht?	Begründung
• Termine eingehalten	<input type="checkbox"/> Ja <input checked="" type="checkbox"/> Nein	Der Endtermin wurde eingehalten. Das Restliche Projekt litt unter Verzug
• Interne Personalressourcen eingehalten	<input checked="" type="checkbox"/> Ja <input type="checkbox"/> Nein < 5% <input type="checkbox"/> Nein > 5%	Klicken oder tippen Sie hier, um Text einzugeben.
• Externe Kosten eingehalten	<input checked="" type="checkbox"/> Ja <input type="checkbox"/> Nein < 5% <input type="checkbox"/> Nein > 5%	Klicken oder tippen Sie hier, um Text einzugeben.

Ergebnisse

Drei Clustersysteme analysiert.

Patroni als zu bauenden Testsystem evaluiert.

Patroni mittels vitabacks / postgresql_cluster als Testsystem umgesetzt und getestet.

Erfolgserlebnisse

Sehr viel Erfahrung mit Kubernetes / rke2 sowie Ansible gesammelt.

Herausforderungen bei Kubernetes / rke2 gemeistert.

Funktionierendes Patroni Testsystem aufgebaut.

Schwierigkeiten / unvorhergesehene Risiken

- YugabyteDB kann als Open-Source DB oder als Subscription Version Yugaware installiert werden
- etcd wurde während der Evaluation auf den Patroni Nodes installiert. Es kam zu einem Namenskonflikt der etcd- und Patroni Nodes
- Der Kubernetes Load Balancer MetalLB benötigte ein L2Advertisement auf den Adresspool. Ohne waren die Evaluationsdatenbanken von aussen nicht erreichbar
- Das Persistente Storage System local-path-provisioner wurde erst falsch konfiguriert.
Daher wurden alle Volumes auf einen Kubernetes Node geschrieben, so dass ab einer gewissen DB Grösse der Storage nicht mehr ausreichte
- Auf den Kubernetes Hosts wurden die sproxy-Zertifikate nicht installiert.
Dadurch konnte StackGres nicht auf sein Extension GitHub Repository zugreifen.
Auch die Maintenance-Tools mussten via Proxy ihre Dependencis installieren
- Das Benchmarking konnte nicht automatisiert werden
- Mit vitabacks / postgresql_cluster konnte erst nicht auf Repositories von außerhalb zugreifen.
Zum einen wurden die Server auf die Zukünftige Firewall ohne Proxy geroutet und entsprechende Firewall Rules erstellt.
Der andere Lösungsweg bestand darin, im Zentralen yml-File (main.yml) den Proxy sauber einzubinden
- Veeam Kasten K10 stand während der Diplomarbeit nicht zur Verfügung.
Daher konnten bei den Distributed SQL Systemen das Backup nur sehr rudimentär betrachtet werden
- Während der gesamten Diplomarbeit stand, dass KSGR Kubernetes Testsystem nicht zur Verfügung. Daher musste das Maintenance-Tool auf die Evaluationsumgebung aufgebaut werden.
- Das HP-UX Ersatzprojekt nahm während der Diplomarbeit stark an Fahrt auf.
Entsprechend musste die Diplomarbeit immer wieder unterbrochen werden.
- Die Evaluationsphase nahm weit mehr Zeit in Anspruch als geplant.
Entsprechend kam es zu Verzug beim Projekt

Offene Punkte

- Das neue SIEM in Form von Elastic muss an das Testsystem angebunden werden
- Mittels des PKI und HashiCorp Vault muss die Testdatenbank sowie der Traffic verschlüsselt werden
- Das Testsystem muss an die Veeam Backup Infrastruktur angebunden werden
- vitabacks / postgresql_cluster muss in die KSGR Ansible Umgebung eingebunden werden
- Die beiden Maintenance-Tools müssen in das KSGR Kubernetes Testsystem eingebunden werden
- Das Produktivsystem muss aufgebaut und in Betrieb genommen werden

Zukunftsprognose / Trends

- YugabyteDB muss als Test- und Produktivsystem für die Cloud Native Applikationen aufgebaut werden
- Es muss ein Performance Warehouse für PostgreSQL konzipiert und aufgebaut werden

Erfahrungen / Empfehlungen

- rke2 bietet die Möglichkeit, mit einfachen Mitteln einen Kubernetes Cluster aufzubauen.
Allerdings braucht es zusätzliche Komponenten wie local-path-provisioner oder MetalLB um das

System für Datenbanken nutzbar zu machen.

Für Produktive Umgebung braucht es daher eine komplexere und stabilere Umgebung

- YugabyteDB hat den Vorteil, selbst grössere Datenmengen komprimiert zu speichern.
Setzt aber ein funktionierendes und synchrones Zeitsystem voraus.
Zudem ist nicht der neueste PostgreSQL Stand implementiert.
- StackGres - Citus benötigt Architekturbedingt grosse Ressourcenmengen.
Das Sharding von Citus hat Grenzen, so lassen sich Foreign Key Constraints nicht ohne weiteres sharden.
StackGres ist zudem bis zu einem gewissen Grad eine Blackbox, Fehler lassen sich dadurch nur schwer evaluieren
- Patroni ist mit vitabacks / postgresql_cluster rasch deployt.
Dessen Konzept birgt aber eine Schwäche, der Connection Pooler ist auf dem gleichen Server installiert wie Patroni. Dadurch wird eine Stärke des Poolers aufgehoben.
Patroni selbst ist State oft he Art wenn es um PostgreSQL Clustering geht.

Was sind die Auswirkungen der Gross-Investition / des Projekts auf:

(zu jedem Stichwort ist eine Aussage zu machen, welche Auswirkungen eingetreten sind)

Personal: keine Veränderung Mehrbedarf Mehrbedarf andere Abteilung

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Immobilien / Räume: weitere Räume nötig Umbau bestehende Räume nötig
 wenig bauliche Massnahmen nötig Keine

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

zusätzliche Geräte nötig bestehende Geräte mussten ersetzt werden
 zusätzliche Büroeinrichtungen nötig

Mobilien / Geräte: bestehende Büroeinrichtungen mussten ersetzt werden
 zusätzliche ICT-Infrastruktur nötig
 bestehende ICT-Infrastruktur musste ersetzt werden
 Keine

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Material (Verbrauch-): neues Material nötig (Bezeichnung): Klicken oder tippen Sie hier, um Text einzugeben.
 Mehrbedarf bestehendes Material nötig
 Keine Veränderung zum IST vor Start der Gross-Investition / des Projekts

Wartung-/Servicevertrag: zusätzlicher Vertrag nötig (Vertragsart): Klicken oder tippen Sie hier, um Text einzugeben.
 bestehender Vertrag wurden verlängert
 Keine Veränderungen zum IST vor Start der Gross-Investition / des Projekts

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Prozesse: Prozessanpassungen im eigenen Fachbereich nötig
 Anpassungen bei anderen nötig
 Keine Anpassungen nötig

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Wirtschaftlichkeit: mehr Erträge tiefere Kosten höhere Kosten
 Medizinischer Nutzen betrieblicher Nutzen
 Patienten Nutzen keine Auswirkungen

Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Antrag

Klicken oder tippen Sie hier, um Text einzugeben.

Beilagen

*Auszug Projekt- / Beschaffungskosten --> Finanzauszug ** Tabelle Auftragsvergaben

Klicken oder tippen Sie hier, um Text einzugeben.

*ist zwingend beizulegen (von Projektkoordinator/-in einfordern)

** ist zwingend beizulegen, wenn im Projekt Aufträge an Externe vergeben wurden, die der Submissionspflicht unterstehen (durch Projektleiter [i.d.R. ST oder IT] auszufüllen)

13.05.2024

Michael Graber

Datum

Unterschrift Projektleitung

Klicken oder tippen Sie, um ein Datum einzugeben.

Datum

Unterschrift Projektcoach

Hier werden Kommentare und Anmerkungen, welche für das Fazit wichtig sein könnten, gesammelt.

Woche	Beschreibung / Event / Problem
KW10	<p>Vier ganze Tage war ich in Thalwil für die Oracle Multitenant-Schulung für das ExaCC Projekt (Ablösung HP-UX).</p> <p>Am Freitag war ich ebenfalls fast den ganzen Tag dran.</p> <p>Weitere Termine werden folgen, das Risiko durch das Projekt tritt langsam ein.</p> <p>Projekt Zeitlich im Verzug.</p>
KW11	<p>Nebst dem HP-UX Ablösungsprojekt schlagen auch diverse Betriebsthemen ein.</p> <p>Die analyse der PostgreSQL HA Cluster nimmt ebenfalls mehr Zeit in Anspruch, als erwartet.</p> <ul style="list-style-type: none"> - HP-UX Probleme am Montag. <p>Backups sind über das Weekend nicht durchgelaufen.</p>
KW12	<p>Die ganze Montagsplanung wurde über den Haufen geworfen.</p> <ul style="list-style-type: none"> - Besprechung bezüglich Backup. <p>Veeam Kasten steht noch nicht zur Verfügung.</p> <ul style="list-style-type: none"> - Mittwochvormittag in Zürich, am Nachmittag Probleme mit dfs-Shares.
KW12	<p>So wenig Zeit.</p> <ul style="list-style-type: none"> - Mit Norman Termin für nächste Woche Fachgespräch organisiert. <p>Freue mich darauf.</p>
KW12	<ul style="list-style-type: none"> - Alle Gängigen PostgreSQL HA Lösungen dokumentiert. Aufwand für Die Dokumentation weit grösser als erwartet. - YugabyteDB entpuppt sich als recht fordernd.
KW13	<p>Es benötigt eine «private container registry», mir fehlt die Expertise dazu.</p> <ul style="list-style-type: none"> - Der Aufbau der Projektplanung entpuppt sich begrenzt nutzbar. <p>Das erstellen der Evaluationsinfrastr</p> <ul style="list-style-type: none"> - Das Problem mit dem «private container registry» rührte daher,
KW13	<p>dass das YugabyteDB Anywhere (Repository yugaware) verwendet wurde.</p> <p>Kurz ein Schock, dass YugabyteDB ausgeschieden ist.</p>
KW13	<p>Später bemerkte ich, dass man das Repo yugabytedb auswählen muss.</p> <ul style="list-style-type: none"> - MetalLB benötigt zwingend L2Advertisement,
KW13	<ul style="list-style-type: none"> damit Linux die Kommunikation von aussen nach innen leiten kann.

VI

Evaluation

VI.I

Maintenance - CloudNativePG

Das Projekt hat eine vergleichsweise hohe Anzahl an aktiven Issues, wobei viele neue dazugekommen sind:

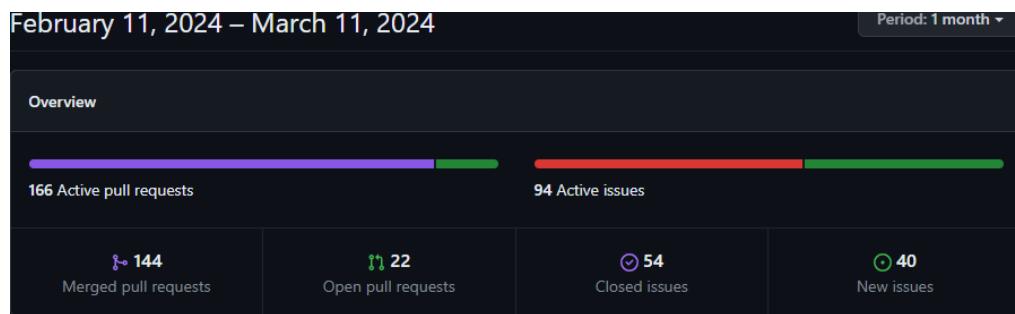


Abbildung I: CloudNativePG - Pulse

Der Code ist aber gut gepflegt, Code wird nicht nur regelmässig hinzugefügt, sondern auch entfernt. Auffällig ist, dass im April 2022 eine grosse Menge Code entfernt wurde:

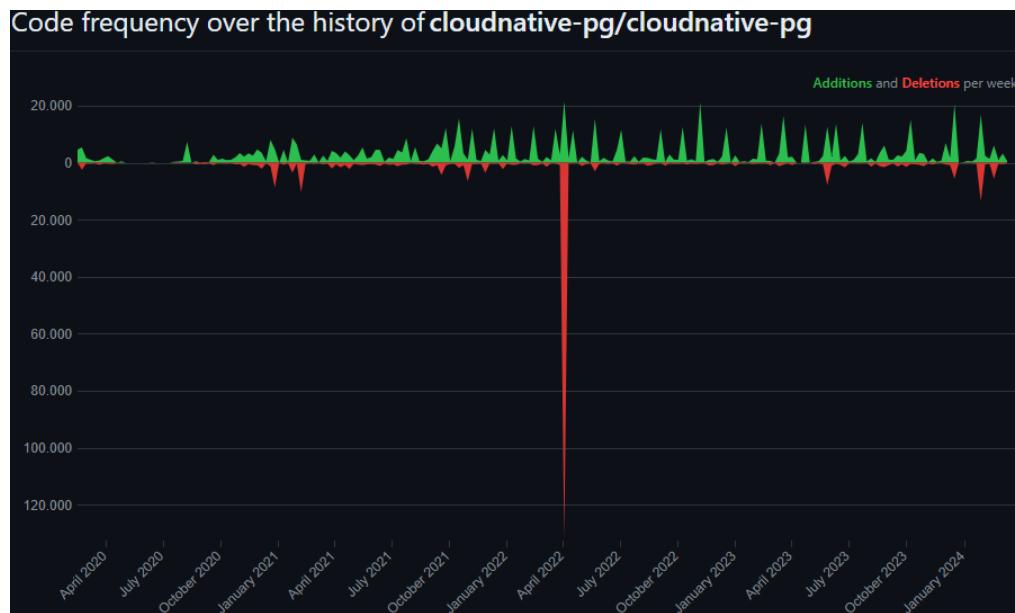


Abbildung II: CloudNativePG - Code Frequency

Das Projekt hält die meisten Standards von GitHub ein:

Community Standards



Abbildung III: CloudNativePG - Community Standards

Die Contributors committen zwar regelmässig auf das Projekt, allerdings fügen sie ungleich mehr dazu, als sie alten Code bereinigen.

Das führt dann dazu, dass es zu grösseren Aufräumarbeiten kommt wie im April 2022.

Es kann der Eindruck gewonnen werden, dass der Code wenig aufgeräumt wird und viel Ballast mit sich schleppst,

was ein Sicherheitsrisiko darstellen kann:

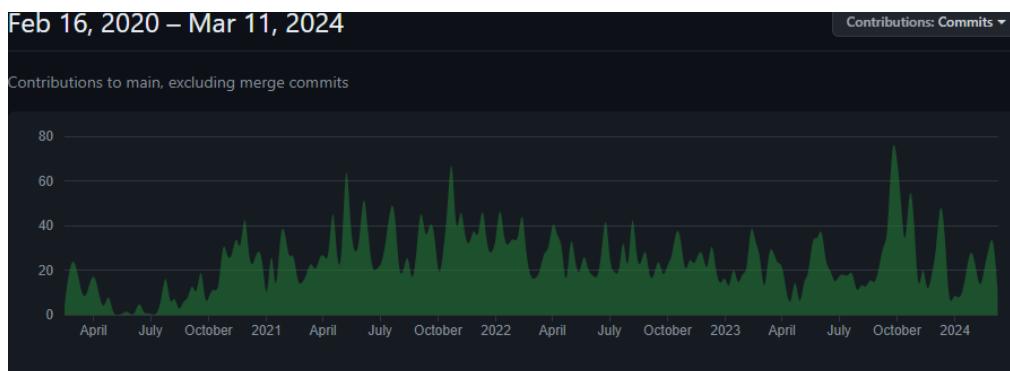


Abbildung IV: CloudNativePG - Contributors Commits



Abbildung V: CloudNativePG - Contributors Deletations

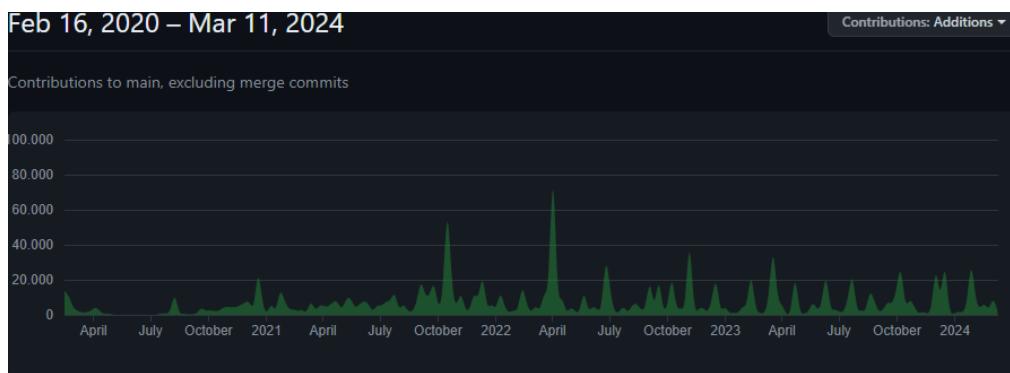


Abbildung VI: CloudNativePG - Contributors Additions

Commits werden regelmässig abgesetzt, allerdings gibt es immer wieder gehäufte Commits.
Oft um die Monatswechsel herum:

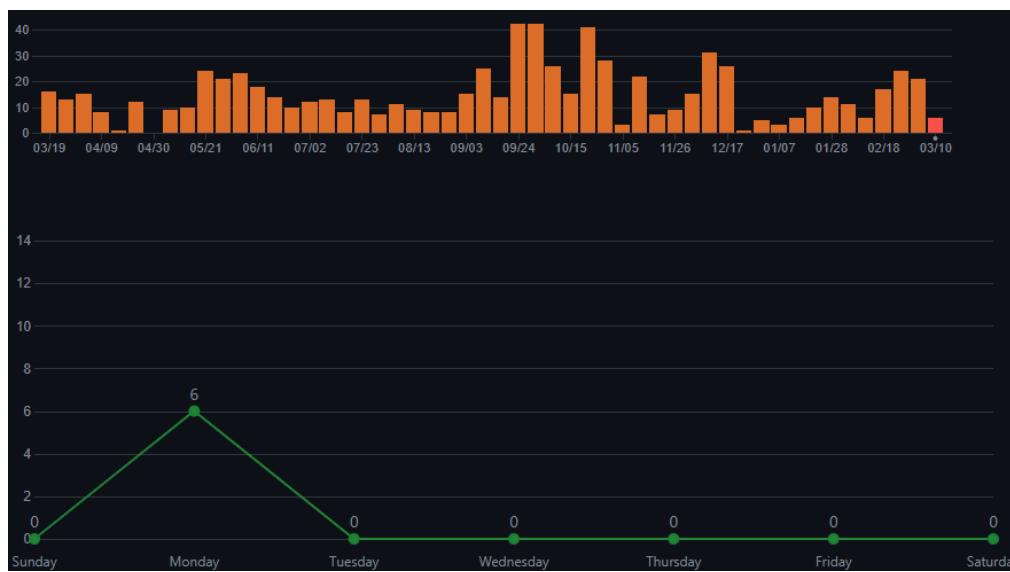


Abbildung VII: CloudNativePG - Commit Activity

Nebst dem Projekt cloudnative-pg der «© The CloudNativePG Contributors» ist CloudNativePG-Gründer EDB noch immer ein grosser Contributor.

Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.

The repository network shows the 100 most recently pushed forks.



Abbildung VIII: CloudNativePG - Network Graph

VI.II Maintenance - Patroni

Patroni wird von Zalando regelmäßig gepflegt. Das Projekt hat eine überschaubare Anzahl an Issues:

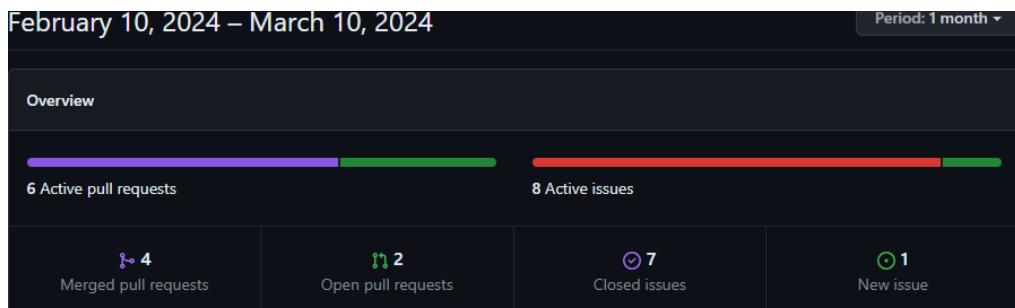


Abbildung IX: Patroni - Pulse

Code wird regelmäßig hinzugefügt und entfernt:

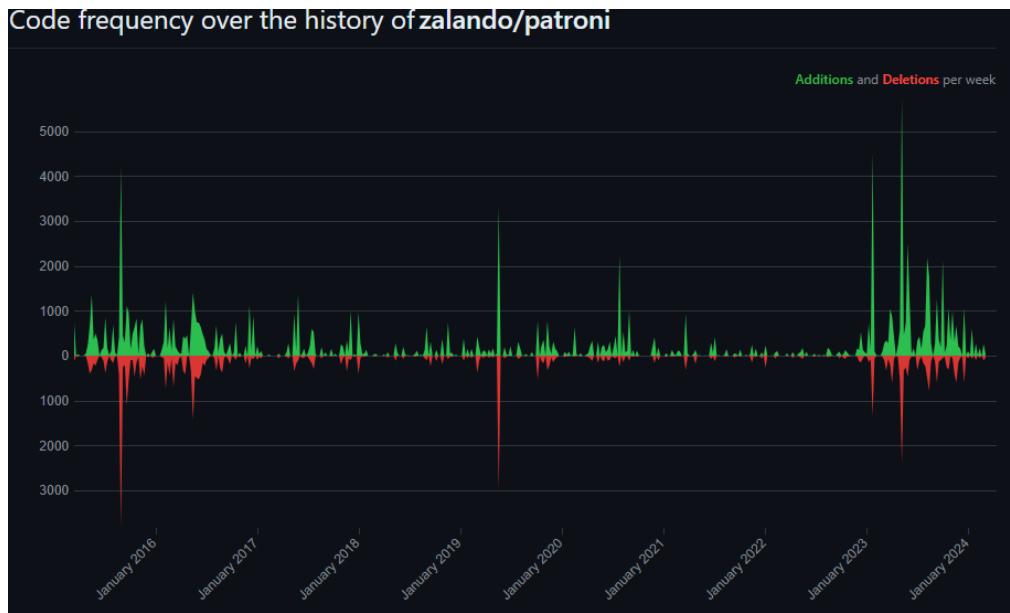


Abbildung X: Patroni - Code Frequency

Das Projekt hält auch die gängigen Standards auf Github ein:

Abbildung XI: Patroni - Community Standards

Die Contributors commiten, löschen und erweitern Patroni regelmässig:

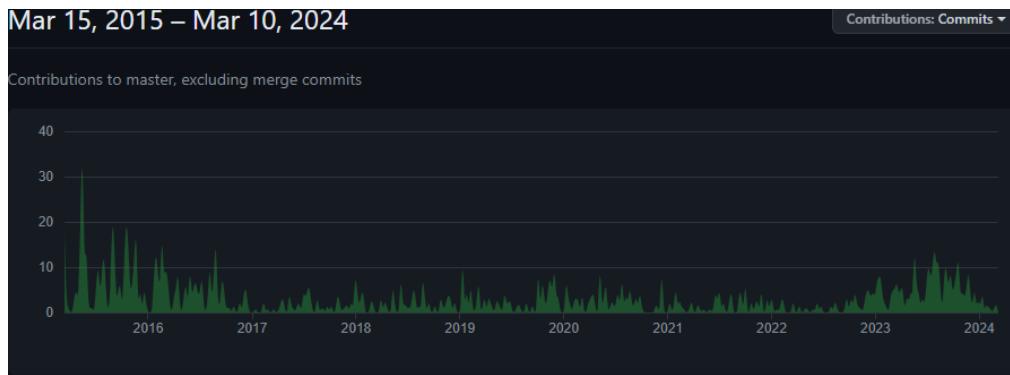


Abbildung XII: Patroni - Contributors Commits



Abbildung XIII: Patroni - Contributors Deletations



Abbildung XIV: Patroni - Contributors Additions

Commits werden nach wie vor immer noch regelmäßig eingespielt, auch wenn die Frequenz etwas nachgelassen hat:

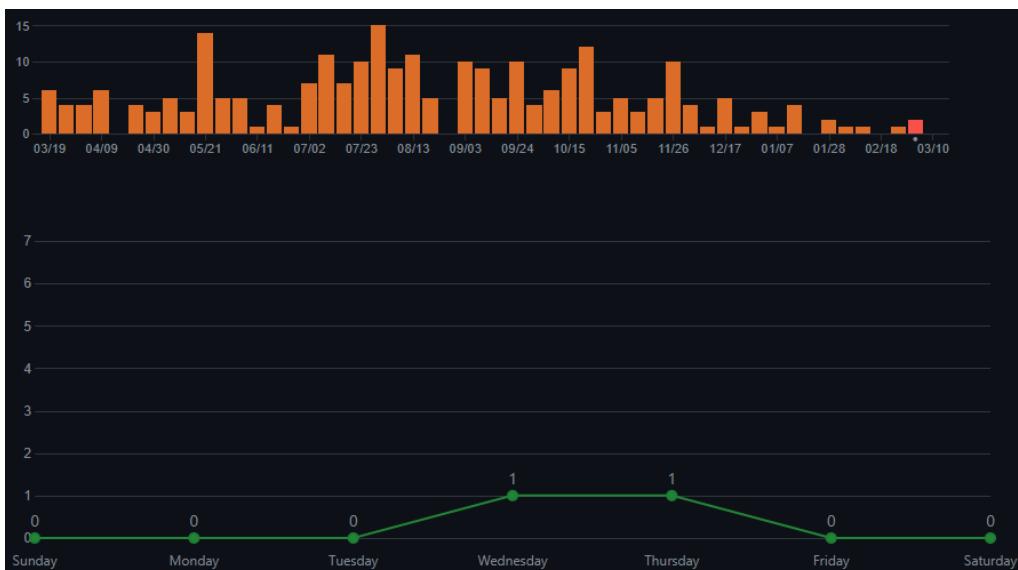


Abbildung XV: Patroni - Commit Activity

Nebst Zalando selbst hat auch EnterpriseDB[26] ein grösseres Repository eingebunden. Dies, weil EnterpriseDB stark auf Patroni setzt.

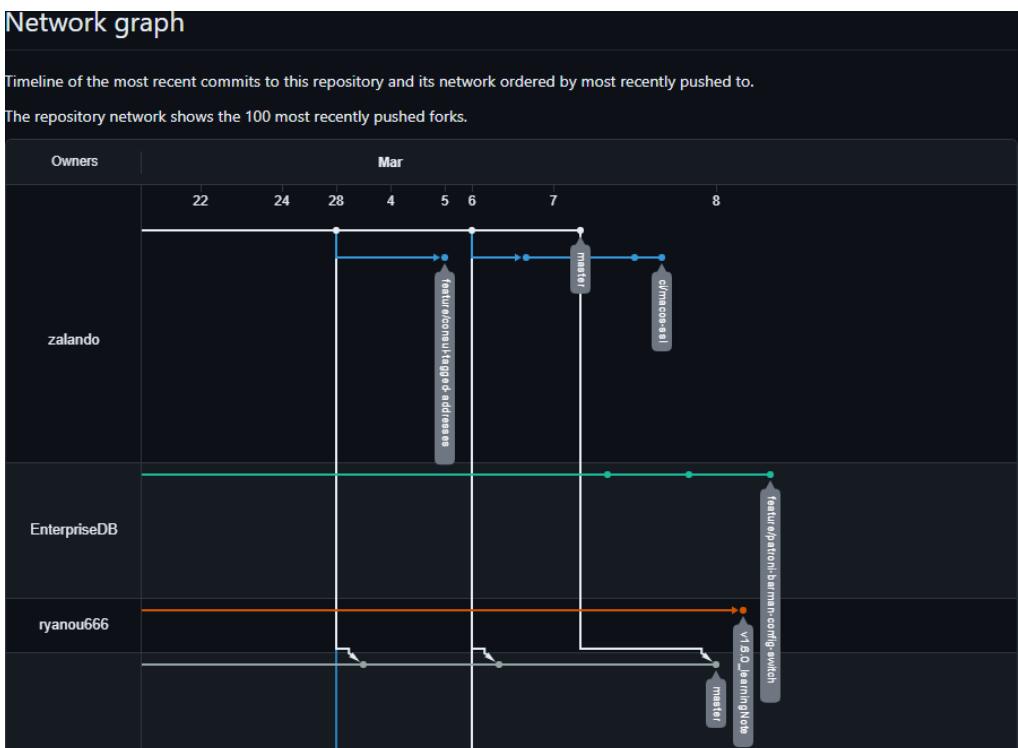


Abbildung XVI: Patroni - Network Graph

VI.III Maintenance - StackGres - Citus

Bei StackGres gab es im letzten Monat keine wirkliche Bewegung:

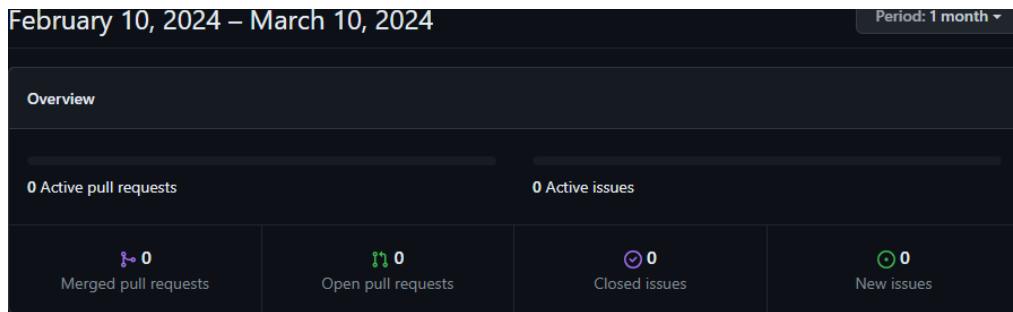


Abbildung XVII: Stackgres - Pulse

Anders sieht es bei Citus aus, die Firma, die mittlerweile zu Microsoft gehört, schliesst Issues rasch und hat eine verhältnismässig hohe Requstrate:

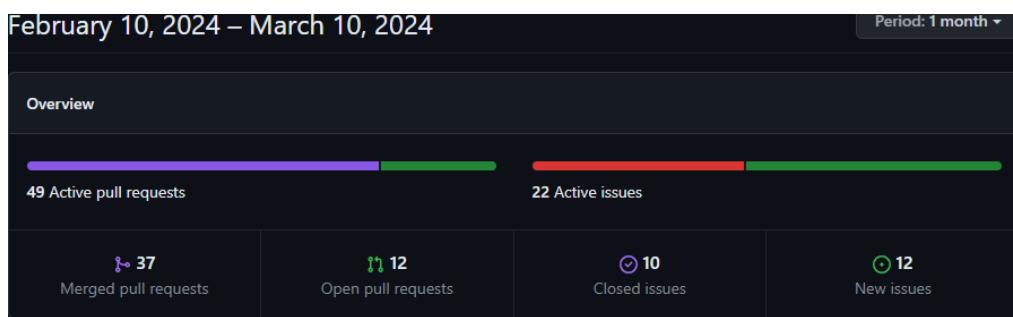


Abbildung XVIII: Citus - Pulse

Bei StackGres wird sehr viel Code hinzugefügt oder gelöscht, beim älteren Citus wurden weniger Änderungen verzeichnet:

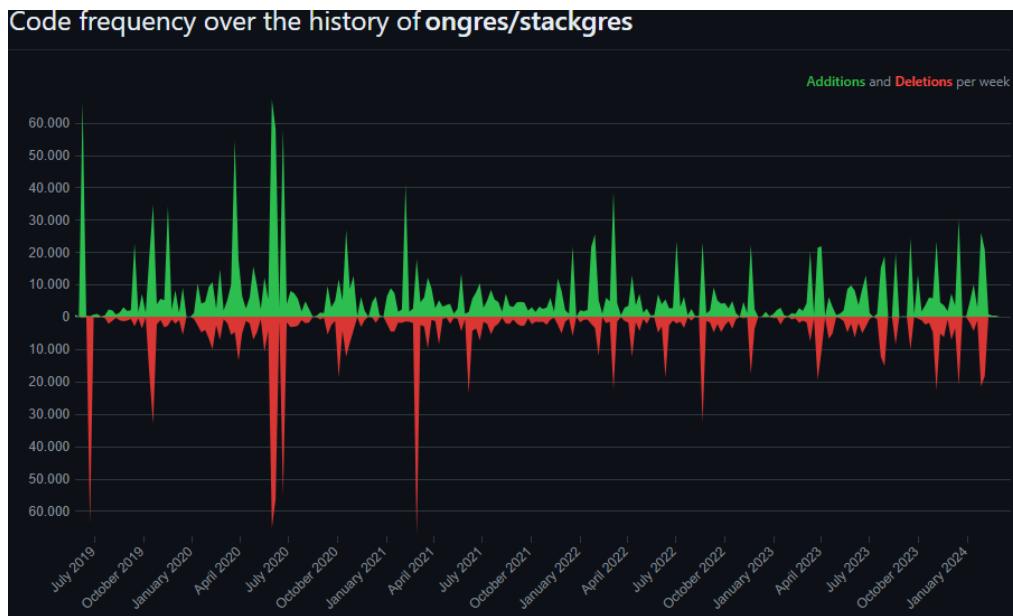


Abbildung XIX: Stackgres - Code Frequency

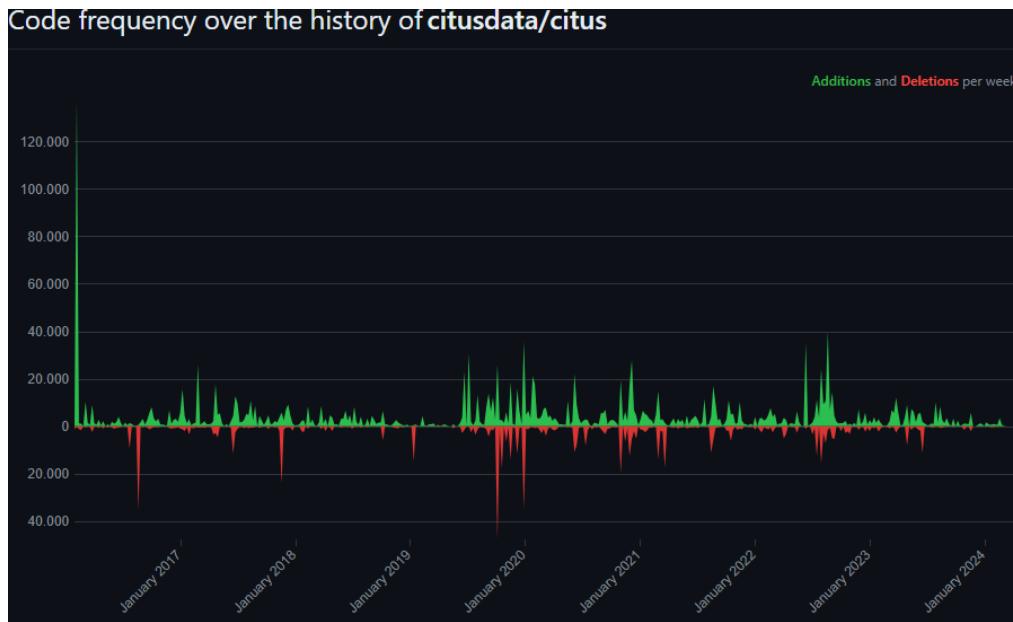


Abbildung XX: Citus - Code Frequency

Citus legt einen hohen Stellenwert auf die Community-Standards, StackGres selbst schneidet hier nur mittelmässig ab:

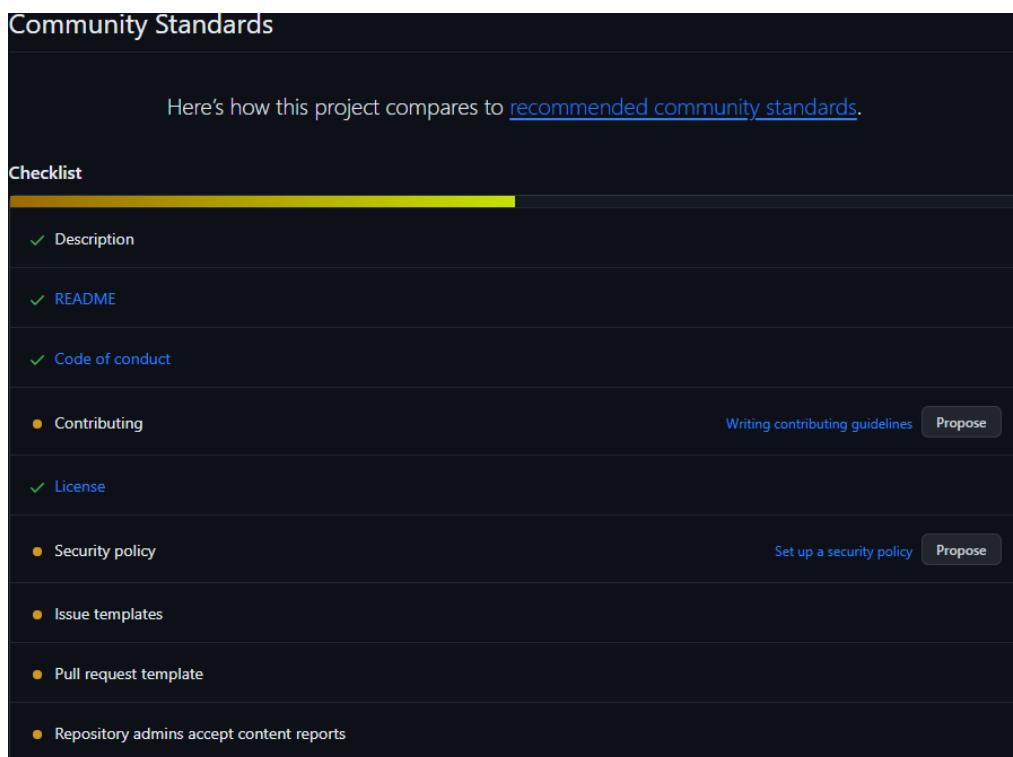


Abbildung XXI: Stackgres - Community Standards

Community Standards

Here's how this project compares to [recommended community standards](#).

Checklist

- ✓ Description
- ✓ README
- ✓ Code of conduct
- ✓ Contributing
- ✓ License
- ✓ Security policy
- Issue templates
- ✓ Pull request template
- Repository admins accept content reports

Abbildung XXII: Citus - Community Standards

Die Stackgres Contributors pflegen aktiv Additions ein, löschen regelmäßig und Commiten ebenfalls auf die main-Branch. Citus, dessen Repository länger Committed wird, hat weniger Bewegung auf die main-Branch.

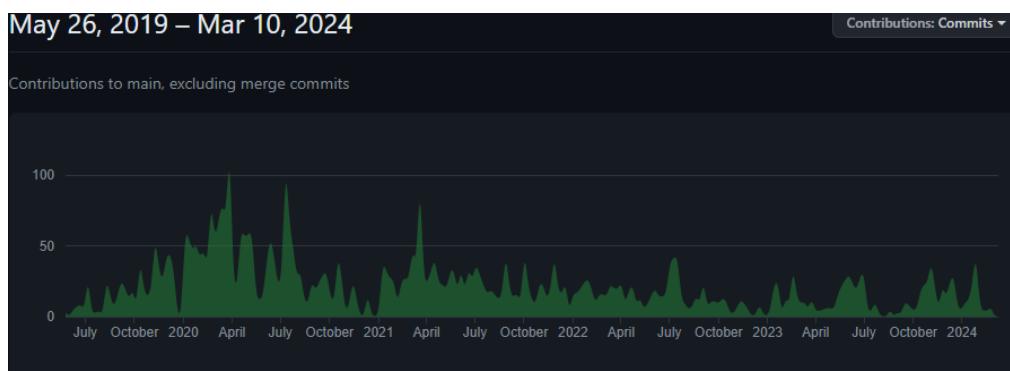


Abbildung XXIII: Stackgres - Contributors Commits

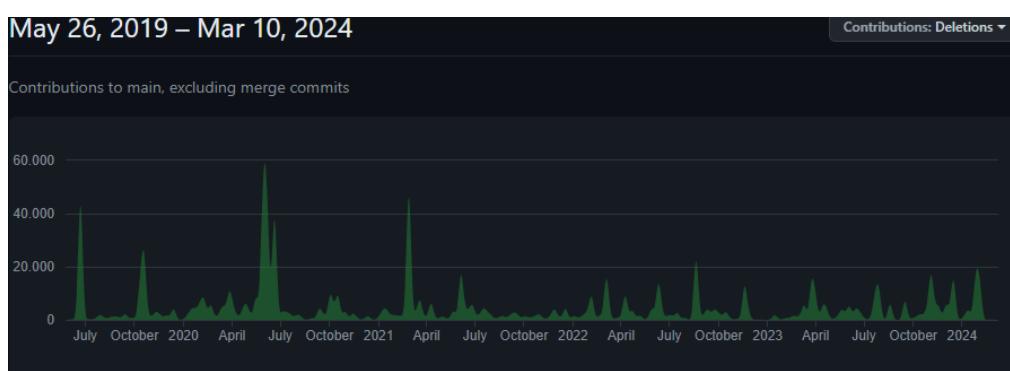


Abbildung XXIV: Stackgres - Contributors Deletations

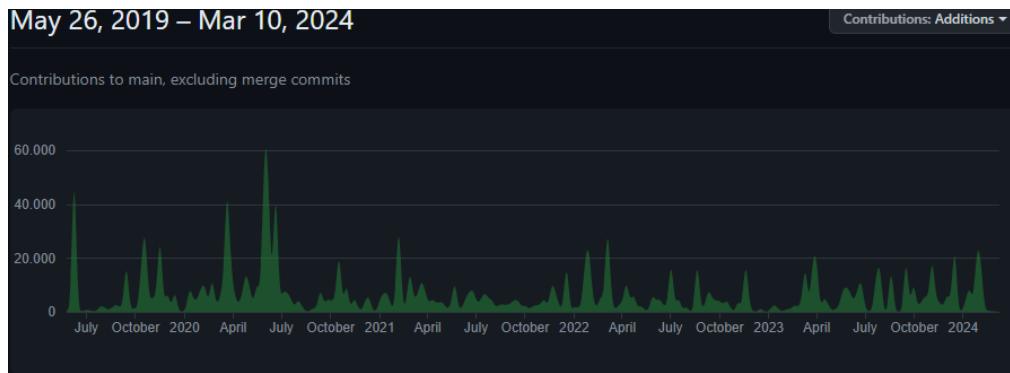


Abbildung XXV: Stackgres - Contributors Additions

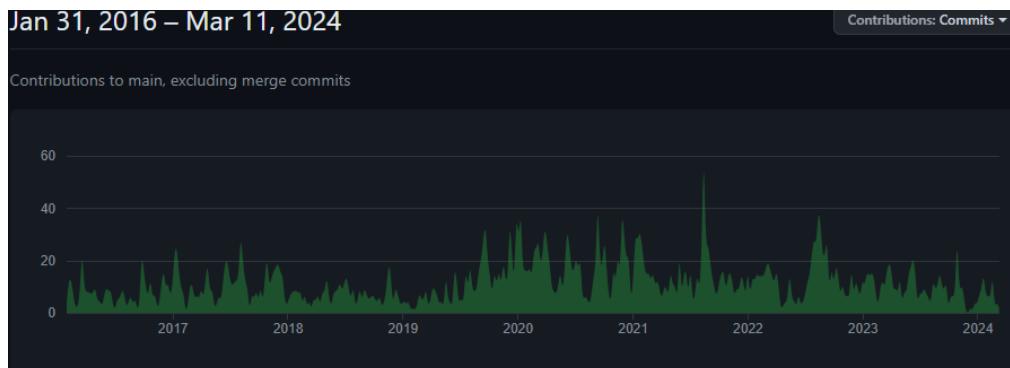


Abbildung XXVI: Citus - Contributors Commits

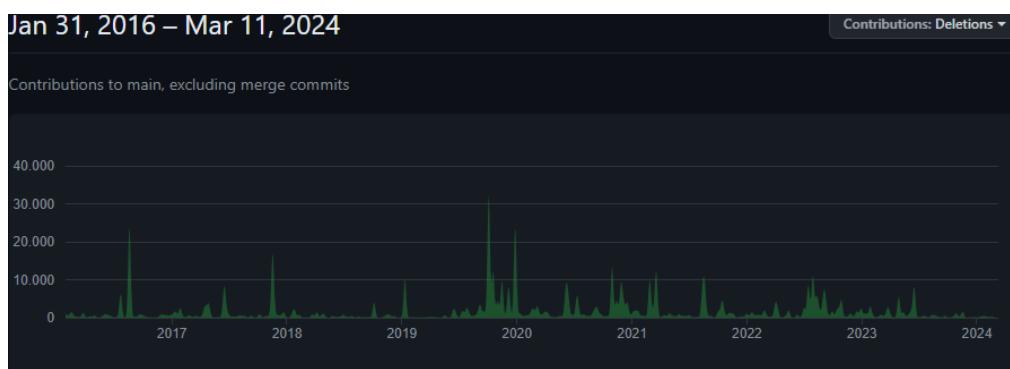


Abbildung XXVII: Citus - Contributors Deletations



Abbildung XXVIII: Citus - Contributors Additions

Gerade Ende Januar gab es bei StackGres eine grössere Anzahl Commits, anhand der Statistik wird ersichtlich, dass i. d. R. einmal pro Monat grössere Mengen an Commits eingespielt werden. Bei Citus gibt es ebenfalls regelmässig grössere Mengen an Commits, allerdings scheint bei citusdata mehr mit kürzeren Sprints gearbeitet zu werden als bei Ongres denn die Commits sind regelmässiger:

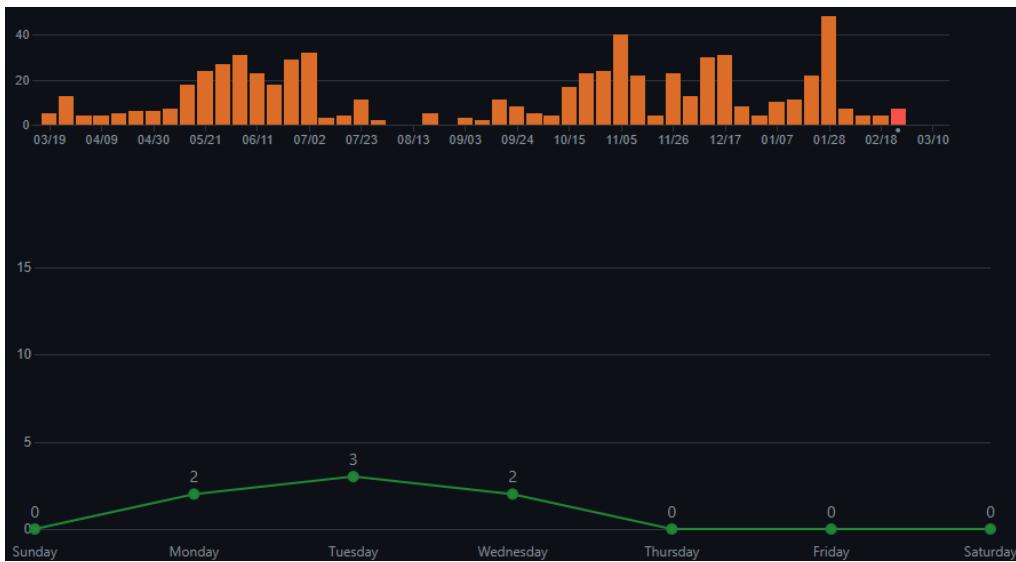


Abbildung XXIX: Stackgres - Commit Activity



Abbildung XXX: Citus - Commit Activity

In letzter Zeit haben nur Ongres, der Entwickler von StackGres, als auch citusdata, grössere Commits auf das Repository gefahren. Andere grössere Entwickler wie EnterpriseDB sind abwesend.

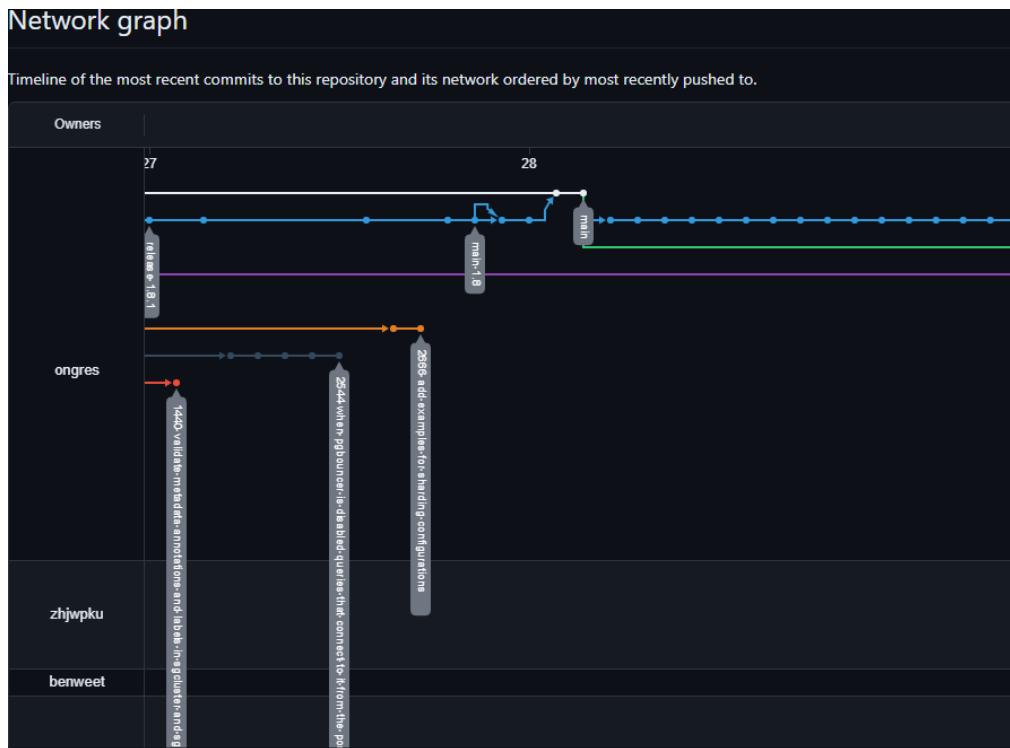


Abbildung XXXI: Stackgres - Network Graph

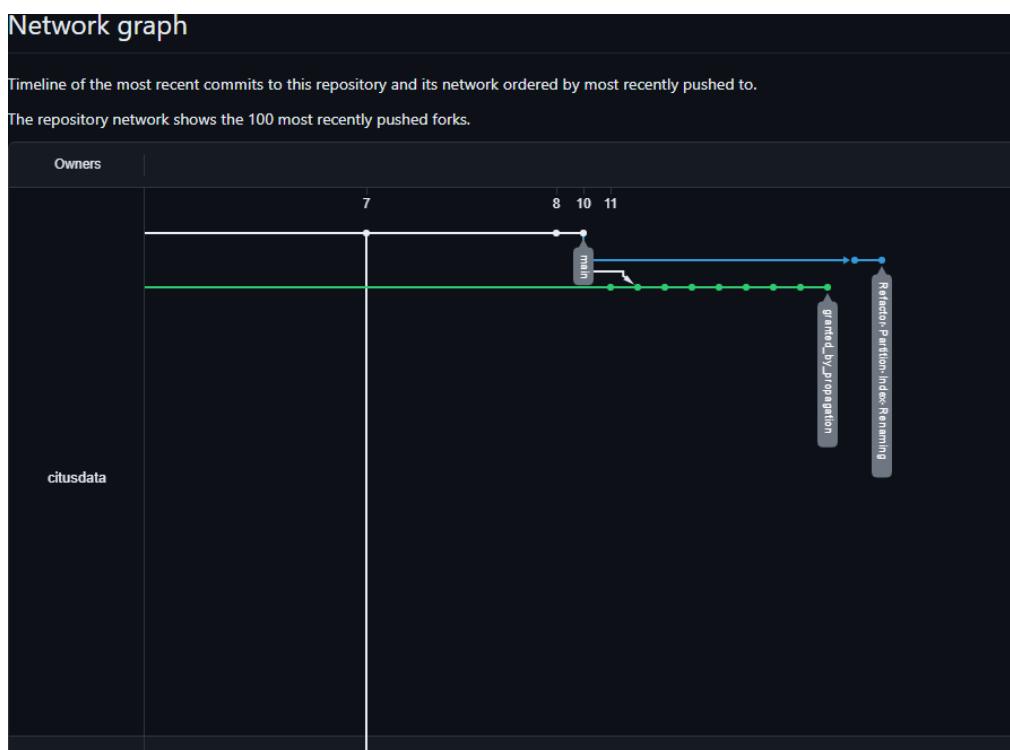


Abbildung XXXII: Citus - Network Graph

VI.IV

Maintenance - YugabyteDB

Das Projekt hat eine sehr hohe Anzahl an aktiven Issues, wobei viele neue dazugekommen sind:

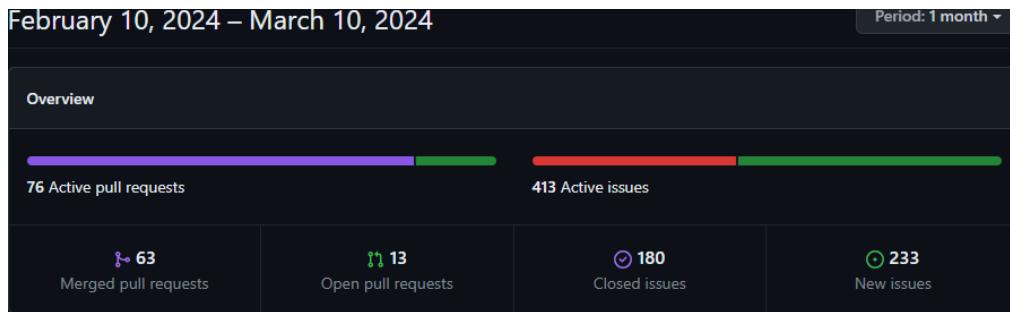


Abbildung XXXIII: YugabyteDB - Pulse

Die Code Frequency kann nicht ausgegeben werden, es gab zu viele Commits:



Abbildung XXXIV: YugabyteDB - Code Frequency

Das Projekt hält nur die wichtigsten Community Standards ein:

Community Standards

Here's how this project compares to [recommended community standards](#).

Checklist

Item	Status	Action
✓ Description		
✓ README		
● Code of conduct		Propose
● Contributing		Writing contributing guidelines Propose
✓ License		
● Security policy		Set up a security policy Propose
✓ Issue templates		
● Pull request template		
● Repository admins accept content reports		

Abbildung XXXV: YugabyteDB - Community Standards

Es werden immer wieder Commits abgesetzt, allerdings sind diese nicht weiter aufgeteilt in Commits, Additions und Deletations:

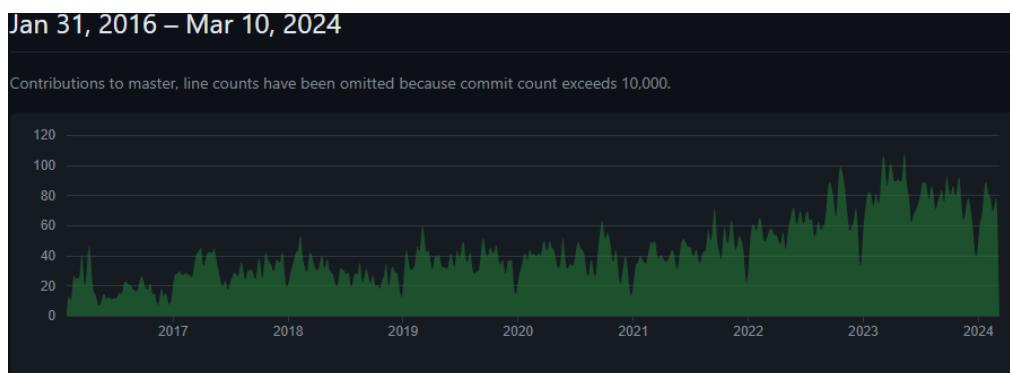


Abbildung XXXVI: YugabyteDB - Contributors

Die Commits wiederum werden regelmässig ausgeführt, es wird scheinbar in kurzen Sprints gearbeitet:



Abbildung XXXVII: YugabyteDB - Commit Activity

YugabyteDB ist der Maintainer seines Produkts.

Es gibt keine anderen grossen Contributors:

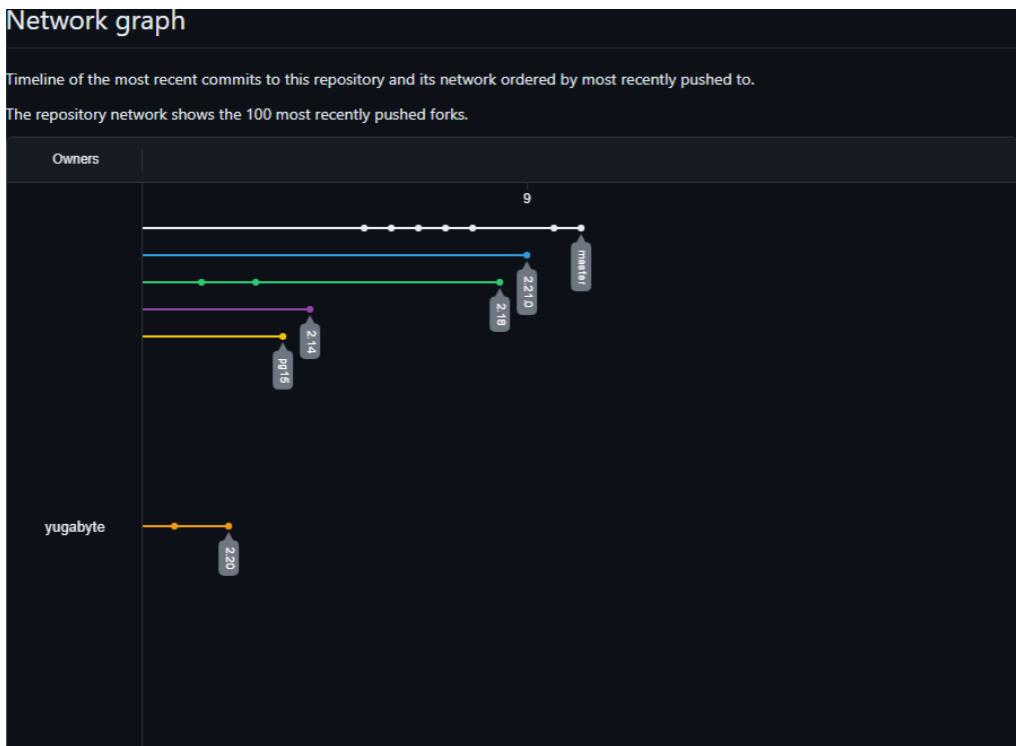


Abbildung XXXVIII: YugabyteDB - Network Graph

VII Evaluationssysteme - Installation

VII.I rke2

VII.I.I Vorbereitung

Da Package aus WAN-Repositories geladen werden, muss eine Proxy-Connection nach aussen gemacht werden können:

```
1 sudo nano /etc/profile.d/proxy.sh
```

```

2
3 export https_proxy=http://sproxy.ssvc.first-it.ch:8080 export HTTPS_PROXY=http://sproxy.ssvc.first-it.ch:8080
4 export http_proxy=http://sproxy.ssvc.first-it.ch:8080
5 export HTTP_PROXY=http://sproxy.ssvc.first-it.ch:8080
6 export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
7 export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
8
9 source /etc/profile.d/proxy.sh

```

Listing 1: Proxy Settings

VII.I.II Installation

VII.I.II.I server - sks1183

Es gibt kein apt-Package. Daher muss zuerst das tarball-Package heruntergeladen werden.

Zuerst wird das Verzeichnis für rke2 erstellt:

```

1 mkdir -p /etc/rancher/rke2/
2 mkdir -p /var/lib/rancher/rke2/server/manifests/

```

Listing 2: rke2 server - Verzeichnis erstellen

```

1 # /etc/rancher/rke2/
2 cluster-cidr: "198.18.0.0/16"
3 service-cidr: "198.19.0.0/16"
4 cni:
5   - cilium
6 disable:
7   - rke2-canal

```

Listing 3: rke2 server - config.yaml

Cilium muss separat manifestiert werden:

```

1 # /var/lib/rancher/rke2/server/manifests/rke2-cilium-config.yaml
2 ---
3 apiVersion: helm.cattle.io/v1
4 kind: HelmChartConfig
5 metadata:
6   name: rke2-cilium
7   namespace: kube-system
8 spec:
9   valuesContent: |-
10     eni:
11       enabled: true

```

Listing 4: rke2 server - cilium-config.yaml

Das Package kann nun installiert und aktiviert werden:

```

1 curl -sfL https://get.rke2.io | INSTALL_RKE2_VERSION=v1.29.0+rke2r1 sh -
2 systemctl enable rke2-server.service
3 systemctl start rke2-server.service

```

Listing 5: rke2 server installieren

VII.I.II. II agents - sks1184 / sks1185

Der Agent muss direkt heruntergeladen, installiert und aktiviert werden:

```
1 curl -sSL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" INSTALL_RKE2_VERSION=v1.29.0+rke2r1 sh -
2 systemctl enable rke2-agent.service
3 mkdir -p /etc/rancher/rke2/
```

Listing 6: rke2 agenten installieren

Die Konfiguration muss nun konfiguriert werden. Dem Agents müssen den Server und den Server Token erhalten:

```
1 # /etc/rancher/rke2/config.yaml
2 server: https://10.0.20.97:9345
3 token: K1042bf32f28282edad37cbac4b77ccfa1cd44a26f0ea2c19111ed664013954a326::server:7a430a28b29501b778543f0882a156b8
```

Listing 7: rke2 agent - config.yaml

Nun muss der Dienst restarted werden:

```
1 systemctl start rke2-agent.service
```

Listing 8: -rke2 agent service restart

VII.I.II. III Cluster Konfiguration

VII.I.II. III.I server und agents

Auch für Kubernetes und die Pots müssen die Proxy-Einstellungen gemacht werden:

```
1 nano /etc/default/rke2-server
2 HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
3 HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
4 NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
5
6 CONTAINERD_HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
7 CONTAINERD_HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
8 CONTAINERD_NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
```

Listing 9: rke2 server proxy

Entsprechend muss die Firewall gesetzt werden:

```
1 nano /etc/iptables/rules.v4
2
3 # Generated by iptables-save v1.8.9 (nf_tables)
4 *filter
5 :INPUT DROP [0:0]
6 :FORWARD ACCEPT [0:0]
7 :OUTPUT ACCEPT [0:0]
8 -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
9 -A INPUT -p udp -m udp --sport 53 -j ACCEPT
10 -A INPUT -p icmp -j ACCEPT
11 -A INPUT -i lo -j ACCEPT
12 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
13 -A INPUT -s 10.0.9.115/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.115" -j ACCEPT
14 -A INPUT -s 10.0.9.76/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.76" -j ACCEPT
15 -A INPUT -s 10.0.36.147/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.36.147" -j ACCEPT
```

```

16 -A INPUT -s 10.0.9.35/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.35" -j ACCEPT
17 -A INPUT -s 10.0.9.37/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.37" -j ACCEPT
18 -A INPUT -s 10.0.9.74/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.74" -j ACCEPT
19 -A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.75" -j ACCEPT-A INPUT -s 10.0.9.36/32 -p udp -m udp --dport 161 -m comment --
comment "Allow SNMP for probe 10.0.9.36" -j ACCEPT
20 -A INPUT -s 10.0.9.14/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.14" -j ACCEPT
21 -A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
22 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6443 -j ACCEPT
23 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 9345 -j ACCEPT
24 COMMIT
25 # Completed
26
27 systemctl restart iptables

```

Listing 10: iptables entries server

VII.I.II.IV local-path-provisioner

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

```

1 root@sks1183:~# mkdir /var/local-path-provisioner
2 root@sks1183:~# chmod -R 777 /var/local-path-provisioner/
3
4 root@sks1184:~# mkdir /var/local-path-provisioner
5 root@sks1184:~# chmod -R 777 /var/local-path-provisioner/
6
7 root@sks1185:~# mkdir /var/local-path-provisioner
8 root@sks1185:~# chmod -R 777 /var/local-path-provisioner/

```

Listing 11: local-path-storage auf Linux Bereitstellen

Anschliessend musste rke2 entsprechend angepasst werden.

Damit automatisch der local-path auf das Verzeichnis /var/local-path-provisioner/ geht, muss dies in einem entsprechenden Manifest geschrieben werden:

```

1 kind: ConfigMap
2 apiVersion: v1
3 metadata:
4   name: local-path-config
5   namespace: local-path-storage
6 data:
7   config.json: |-
8     {
9       "nodePathMap": [
10         {
11           "node": "DEFAULT_PATH_FOR_NON_LISTED_NODES",
12           "paths": ["/var/local-path-provisioner"]
13         }
14       ]
15     }
16   setup: |-
17     #!/bin/sh
18     set -eu
19     mkdir -m 0777 -p "$VOL_DIR"
20   teardown: |-
21     #!/bin/sh

```

```

22     set -eu
23     rm -rf "$VOL_DIR"
24   helperPod.yaml: |-
25     apiVersion: v1
26     kind: Pod
27     metadata:
28       name: helper-pod
29     spec:
30       priorityClassName: system-node-critical
31       tolerations:
32         - key: node.kubernetes.io/disk-pressure
33           operator: Exists
34           effect: NoSchedule
35       containers:
36         - name: helper-pod
37           image: busybox

```

Listing 12: local-path-provisioner definieren

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/rke2/local-path-provisioner.yaml
```

Listing 13: local-path-storage aktualisieren

VII.I.II.V MetallB - Proxy / Load Balancer

MetallB musste installiert werden:

```
1 kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.4/config/manifests/metallb-native.yaml
```

Listing 14: MetallB installieren

Das Konfigurationsmanifest wurde eingespielt:

```

1 apiVersion: metallb.io/v1beta1
2 kind: IPAddressPool
3 metadata:
4   name: distributed-sql
5   namespace: metallb-system
6 spec:
7   addresses:
8     - 10.0.20.106-10.0.20.106
9     - 10.0.20.150-10.0.20.155
10 ---
11 apiVersion: metallb.io/v1beta1
12 kind: L2Advertisement
13 metadata:
14   name: l2adv
15   namespace: metallb-system
16 spec:
17   ipAddressPools:
18     - distributed-sql

```

Listing 15: MetallB konfigurieren

Das Manifest musste danach eingespielt werden:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/rke2/metallb-values.yaml
```

Listing 16: MetallB Konfiguration einspielen

VII.I.II.VI local-path-provisioner - Grosse Volumes

Sobald die neue Disk im VMware vSphere erfasst wurde, muss die Disk auf den Servern gemountet werden:

```
1 echo " - - - " | tee /sys/class/scsi_host/host*/scan && lsblk
2
3 fdisk /dev/sdb
4   n -> all default values
5   t -> 8e
6   p -> Kontrolle
7   w
8
9 pvcreate /dev/sdb1 && \
10 vgcreate vgdata /dev/sdb1 && \
11 lvcreate -l 100%FREE -n lvdata vgdata && \
12 mkfs.ext4 /dev/vgdata/lvdata && \
13 mkdir -p /srv/data && \
14 cp /etc/fstab /tmp/fstab.bak && \
15 echo "/dev/vgdata/lvdata /srv/data ext4 defaults 0 0" >> /etc/fstab && \
16 systemctl daemon-reload && mount -a && lsblk
```

Listing 17: rke2 - 250GiB Disk mount

Nun muss das Verzeichnis local-path-provisioner auf der Disk /srv/data/ erstellt und berechtigt werden:

```
1 root@sks1183:~# mkdir -p /srv/data/local-path-provisioner
2 root@sks1183:~# chmod 777 /srv/data/local-path-provisioner
3
4 root@sks1184:~# mkdir -p /srv/data/local-path-provisioner
5 root@sks1184:~# chmod 777 /srv/data/local-path-provisioner
6
7 root@sks1185:~# mkdir -p /srv/data/local-path-provisioner
8 root@sks1185:~# chmod 777 /srv/data/local-path-provisioner
```

Listing 18: local-path-storage 250GiB auf Linux Bereitstellen

Spätestens wenn über 200GiB präsentiert werden sollen, muss das Binding auf den jeweiligen Node stattfinden:

```
1 kind: ConfigMap
2 apiVersion: v1
3 metadata:
4   name: local-path-config
5   namespace: local-path-storage
6 data:
7   config.json: |- 
8     {
9       "nodePathMap": [
10         {
11           "node": "DEFAULT_PATH_FOR_NON_LISTED_NODES",
12           "paths": ["/srv/data/local-path-provisioner"]
13         },
14         {
15       }
```

```

15         "node": "sk1183",
16         "paths": ["/srv/data/local-path-provisioner"]
17     },
18     {
19         "node": "sk1184",
20         "paths": ["/srv/data/local-path-provisioner"]
21     },
22     {
23         "node": "sk1185",
24         "paths": ["/srv/data/local-path-provisioner"]
25     }
26   ]
27 }
28 setup: |-
29   #!/bin/sh
30   set -eu
31   mkdir -m 0777 -p "$VOL_DIR"
32 teardown: |-
33   #!/bin/sh
34   set -eu
35   rm -rf "$VOL_DIR"
36 helperPod.yaml: |-
37   apiVersion: v1
38   kind: Pod
39   metadata:
40     name: helper-pod
41   spec:
42     priorityClassName: system-node-critical
43     tolerations:
44       - key: node.kubernetes.io/disk-pressure
45         operator: Exists
46         effect: NoSchedule
47     containers:
48       - name: helper-pod
49         image: busybox

```

Listing 19: local-path-provisioner Grosse Volumes

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/rke2/local-path-provisioner_srv_default.yaml
```

Listing 20: local-path-storage 250GiB aktualisieren

VII.II **yugabyteDB**

VII.II.I **Prerequisites**

VII.II.I.I **StorageClass setzen**

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```

1 apiVersion: storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4   name: yb-storage

```

```

5 provisioner: rancher.io/local-path
6 parameters:
7   nodePath: /var/local-path-provisioner
8 volumeBindingMode: WaitForFirstConsumer
9 reclaimPolicy: Delete
10 --
11 apiVersion: v1
12 kind: PersistentVolume
13 metadata:
14   name: yb-storage-pv
15   labels:
16     type: local
17 spec:
18   accessModes:
19     - ReadWriteOnce
20   capacity:
21     storage: 3Gi
22   storageClassName: "yb-storage"
23   hostPath:
24     path: /var/local-path-provisioner

```

Listing 21: YugabyteDB - StorageClass setzen

Die StorageClass und das PersistentVolume muss aktiviert werden:

```

1 gramic@cks4040:~$ kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/yugabytedb/yugabytedb/storageclass.yaml
2 storageclass.storage.k8s.io/yb-storage unchanged
3 persistentvolume/yb-storage-pv created

```

Listing 22: YugabyteDB - StorageClass / PersistentVolume aktivieren

VII.II.II Installation

Zuerst muss ein Namespace erstellt werden:

```
1 kubectl create namespace yb-platform
```

Listing 23: YugabyteDB - Namespace

Das vollständige helm Chart Manifest:

```

1 # Default values for Yugabyte.
2 # This is a YAML-formatted file.
3 # Declare variables to be passed into your templates.
4 Component: "yugabytedb"
5
6 fullnameOverride: ""
7 nameOverride: ""
8
9 Image:
10   repository: "yugabytedb/yugabyte"
11   tag: 2.20.2.1-b3
12   pullPolicy: IfNotPresent
13   pullSecretName: ""
14
15 storage:

```

```

16   ephemeral: false # will not allocate PVs when true
17
18   master:
19     count: 1
20     size: 3Gi
21     storageClass: "yb-storage"
22
23   tserver:
24     count: 1
25     size: 3Gi
26     storageClass: "yb-storage"
27
28 resource:
29   master:
30     requests:
31       cpu: "1"
32       memory: 2Gi
33     limits:
34       cpu: "1"
35       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
36       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
37       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
38       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
39       memory: 2Gi
40
41   tserver:
42     requests:
43       cpu: "1"
44       memory: 4Gi
45     limits:
46       cpu: "1"
47       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
48       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
49       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
50       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
51       memory: 4Gi
52
53 replicas:
54   master: 3
55   tserver: 3
56   ## Used to set replication factor when isMultiAz is set to true
57   totalMasters: 3
58
59 partition:
60   master: 0
61   tserver: 0
62
63 updateStrategy:
64   type: RollingUpdate
65
66 # Used in Multi-AZ setup
67 masterAddresses: ""
68
69 isMultiAz: false
70 AZ: ""
71
72 # Disable the YSQL

```

```

70 disableYsql: false
71
72 tls:
73   # Set to true to enable the TLS.
74   enabled: false
75   nodeToNode: true
76   clientToServer: true
77   # Set to false to disallow any service with unencrypted communication from joining this cluster
78   insecure: false
79   # Set enabled to true to use cert-manager instead of providing your own rootCA
80   certManager:
81     enabled: false
82     # Will create own ca certificate and issuer when set to true
83     bootstrapSelfsigned: true
84     # Use ClusterIssuer when set to true, otherwise use Issuer
85     useClusterIssuer: false
86     # Name of ClusterIssuer to use when useClusterIssuer is true
87     clusterIssuer: cluster-ca
88     # Name of Issuer to use when useClusterIssuer is false
89     issuer: Yugabyte-ca
90     certificates:
91       # The lifetime before cert-manager will issue a new certificate.
92       # The re-issued certificates will not be automatically reloaded by the service.
93       # It is necessary to provide some external means of restarting the pods.
94       duration: 2160h # 90d
95       renewBefore: 360h # 15d
96       algorithm: RSA # ECDSA or RSA
97       # Can be 2048, 4096 or 8192 for RSA
98       # Or 256, 384 or 521 for ECDSA
99       keySize: 2048
100
101 ## When certManager.enabled=false, rootCA.cert and rootCA.key are used to generate TLS certs.
102 ## When certManager.enabled=true and bootstrapSelfsigned=true, rootCA is ignored.
103 ## When certManager.enabled=true and bootstrapSelfsigned=false, only rootCA.cert is used
104 ## to verify TLS certs generated and signed by the external provider.
105 rootCA:
106   cert: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ01CQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJRd0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakF1RncweE9UQX1NRGd3TURRd01qSmFGdzB5T1RBcU1EVXdNRFF3TWpKEy"
107   key: "LS0tLS1CRUdJTiBSUOEgUFJJVkfURSBLRVkTLS0tLQpNSU1FcEFJQkFBSONBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBx0XJpWDZqRXJ1YnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQL0J1M2RxMW4yb0RDZkZUTDB4aTI0V01kTFFyckEy"
108
109 ## When tls.certManager.enabled=false
110 ## nodeCert and clientCert will be used only when rootCA.key is empty.
111 ## Will be ignored and genSignedCert will be used to generate
112 ## node and client certs if rootCA.key is provided.
113 ## cert and key are base64 encoded content of certificate and key.
114 nodeCert:
115   cert: ""
116   key: ""
117 clientCert:
118   cert: ""
119   key: ""

```

```

120 gflags:
121   master:
122     default_memory_limit_to_ram_ratio: 0.85
123   tserver: {}
124 #   use_cassandra_authentication: false
125
126 PodManagementPolicy: Parallel
127
128 enableLoadBalancer: true
129
130 ybc:
131   enabled: false
132 ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#resource-requests-and-limits-of-pod-and-container
133 ## Use the above link to learn more about Kubernetes resources configuration.
134 # resources:
135 #   requests:
136 #     cpu: "1"
137 #     memory: 1Gi
138 #   limits:
139 #     cpu: "1"
140 #     memory: 1Gi
141
142 ybCleanup: {}
143 ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#resource-requests-and-limits-of-pod-and-container
144 ## Use the above link to learn more about Kubernetes resources configuration.
145 # resources:
146 #   requests:
147 #     cpu: "1"
148 #     memory: 1Gi
149 #   limits:
150 #     cpu: "1"
151 #     memory: 1Gi
152
153 domainName: "cluster.local"
154
155 serviceEndpoints:
156   - name: "yb-master-ui"
157     type: LoadBalancer
158     annotations: {}
159     clusterIP: ""
160     ## Sets the Service's externalTrafficPolicy
161     externalTrafficPolicy: ""
162     app: "yb-master"
163     loadBalancerIP: ""
164     ports:
165       http-ui: "7000"
166
167   - name: "yb-tserver-service"
168     type: LoadBalancer
169     annotations:
170       metallb.universe.tf/loadBalancerIPs: 10.0.20.106
171     clusterIP: ""
172     ## Sets the Service's externalTrafficPolicy
173     externalTrafficPolicy: ""

```

```

174 app: "yb-tserver"
175 loadBalancerIP: ""
176 ports:
177   tcp-yql-port: "9042"
178   tcp-yedis-port: "6379"
179   tcp-ysql-port: "5433"
180
181 Services:
182 - name: "yb-masters"
183   label: "yb-master"
184   skipHealthChecks: false
185   memory_limit_to_ram_ratio: 0.85
186   ports:
187     http-ui: "7000"
188     tcp-rpc-port: "7100"
189
190 - name: "yb-tservers"
191   label: "yb-tserver"
192   skipHealthChecks: false
193   ports:
194     http-ui: "9000"
195     tcp-rpc-port: "9100"
196     tcp-yql-port: "9042"
197     tcp-yedis-port: "6379"
198     tcp-ysql-port: "5433"
199     http-ycql-met: "12000"
200     http-yedis-met: "11000"
201     http-ysql-met: "13000"
202     grpc-ybc-port: "18018"
203
204
205 ## Should be set to true only if Istio is being used. This also adds
206 ## the Istio sidecar injection labels to the pods.
207 ## TODO: remove this once
208 ## https://github.com/yugabyte/yugabyte-db/issues/5641 is fixed.
209 ##
210 istioCompatibility:
211   enabled: false
212
213 ## Settings required when using multicluster environment.
214 multicluster:
215   ## Creates a ClusterIP service for each yb-master and yb-tserver
216   ## pod.
217   createServicePerPod: false
218   ## creates a ClusterIP service whos name does not have release name
219   ## in it. A common service across different clusters for automatic
220   ## failover. Useful when using new naming style.
221   createCommonTserverService: false
222
223   ## Enable it to deploy YugabyteDB in a multi-cluster services enabled
224   ## Kubernetes cluster (KEP-1645). This will create ServiceExport.
225   ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#registering\_a\_service\_for\_export
226   ## You can use this gist for the reference to deploy the YugabyteDB in a multi-cluster scenario.
227   ## Gist - https://gist.github.com/baba230896/78cc9bb6f4ba0b3d0e611cd49ed201bf

```

```

228 createServiceExports: false
229
230 ## Mandatory variable when createServiceExports is set to true.
231 ## Use: In case of GKE, you need to pass GKE Hub Membership Name.
232 ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#enabling
233 kubernetesClusterId: ""
234
235 ## mcsApiVersion is used for the MCS resources created by the
236 ## chart. Set to net.gke.io/v1 when using GKE MCS.
237 mcsApiVersion: "multicloud.x-k8s.io/v1alpha1"
238
239 serviceMonitor:
240     ## If true, two ServiceMonitor CRs are created. One for yb-master
241     ## and one for yb-tserver
242     ## https://github.com/coreos/prometheus-operator/blob/master/Documentation/api.md#servicemonitor
243     ##
244     enabled: false
245     ## interval is the default scrape_interval for all the endpoints
246     interval: 30s
247     ## extraLabels can be used to add labels to the ServiceMonitors
248     ## being created
249     extraLabels: {}
250     # release: prom
251
252     ## Configurations of ServiceMonitor for yb-master
253 master:
254     enabled: true
255     port: "http-ui"
256     interval: ""
257     path: "/prometheus-metrics"
258
259     ## Configurations of ServiceMonitor for yb-tserver
260 tserver:
261     enabled: true
262     port: "http-ui"
263     interval: ""
264     path: "/prometheus-metrics"
265 ycql:
266     enabled: true
267     port: "http-ycql-met"
268     interval: ""
269     path: "/prometheus-metrics"
270 ysql:
271     enabled: true
272     port: "http-ysql-met"
273     interval: ""
274     path: "/prometheus-metrics"
275 yedis:
276     enabled: true
277     port: "http-yedis-met"
278     interval: ""
279     path: "/prometheus-metrics"
280
281 commonMetricRelabelings:

```

```

282 # https://git.io/JJW5p
283 # Save the name of the metric so we can group_by since we cannot by __name__ directly...
284 - sourceLabels: ["__name__"]
285   regex: "(.*"
286   targetLabel: "saved_name"
287   replacement: "$1"
288 # The following basically retrofit the handler_latency_* metrics to label format.
289 - sourceLabels: ["__name__"]
290   regex: "handler_latency_(yb_[^_]*_([_]*_)_([_]*)(.*"
291   targetLabel: "server_type"
292   replacement: "$1"
293 - sourceLabels: ["__name__"]
294   regex: "handler_latency_(yb_[^_]*_([_]*_)_([_]*)(.*"
295   targetLabel: "service_type"
296   replacement: "$2"
297 - sourceLabels: ["__name__"]
298   regex: "handler_latency_(yb_[^_]*_([_]*_)_([_]*)(_sum|_count)?"
299   targetLabel: "service_method"
300   replacement: "$3"
301 - sourceLabels: ["__name__"]
302   regex: "handler_latency_(yb_[^_]*_([_]*_)_([_]*)(_sum|_count)?"
303   targetLabel: "__name__"
304   replacement: "rpc_latency$4"
305
306 resources: {}
307
308 nodeSelector: {}
309
310 affinity: {}
311
312 statefulSetAnnotations: {}
313
314 networkAnnotation: {}
315
316 commonLabels: {}
317
318 master:
319   ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
320   ## This might override the default affinity from service.yaml
321   # To successfully merge, we need to follow rules for merging nodeSelectorTerms that kubernentes
322   # has. Each new node selector term is ORed together, and each match expression or match field in
323   # a single selector is ANDed together.
324   # This means, if a pod needs to be scheduled on a label 'custom_label_1' with a value
325   # 'custom_value_1', we need to add this 'subterm' to each of our pre-defined node affinity
326   # terms.
327   #
328   # Pod anti affinity is a simpler merge. Each term is applied separately, and the weight is tracked.
329   # The pod that achieves the highest weight is selected.
330   ## Example.
331   # affinity:
332   #   podAntiAffinity:
333   #     requiredDuringSchedulingIgnoredDuringExecution:
334   #       - labelSelector:
335   #         matchExpressions:
```

```

336 #       - key: app
337 #         operator: In
338 #
339 #         values:
340 #           - "yb-master"
341 #
342 # For further examples, see examples/yugabyte/affinity_overrides.yaml
343 affinity: {}
344
345 ## Extra environment variables passed to the Master pods.
346 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
347 ## Example:
348 # extraEnv:
349 #   - name: NODE_IP
350 #     valueFrom:
351 #       fieldRef:
352 #         fieldPath: status.hostIP
353 extraEnv: []
354
355 # secretEnv variables are used to expose secrets data as env variables in the master pod.
356 # TODO Add namespace also to support copying secrets from other namespace.
357 # secretEnv:
358 #   - name: MYSQL_LDAP_PASSWORD
359 #     valueFrom:
360 #       secretKeyRef:
361 #         name: secretName
362 #         key: password
363 secretEnv: []
364
365 ## Annotations to be added to the Master pods.
366 podAnnotations: {}
367
368 ## Labels to be added to the Master pods.
369 podLabels: {}
370
371 ## Tolerations to be added to the Master pods.
372 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
373 ## Example:
374 # tolerations:
375 #   - key: dedicated
376 #     operator: Equal
377 #     value: experimental
378 #     effect: NoSchedule
379 tolerations: []
380
381 ## Extra volumes
382 ## extraVolumeMounts are mandatory for each extraVolumes.
383 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
384 ## Example:
385 # extraVolumes:
386 #   - name: custom-nfs-vol
387 #     persistentVolumeClaim:
388 #       claimName: some-nfs-claim
389 extraVolumes: []

```

```

390
391 ## Extra volume mounts
392 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volumemount-v1-core
393 ## Example:
394 # extraVolumeMounts:
395 # - name: custom-nfs-vol
396 #   mountPath: /home/yugabyte/nfs-backup
397 extraVolumeMounts: []
398
399 ## Set service account for master DB pods. The service account
400 ## should exist in the namespace where the master DB pods are brought up.
401 serviceAccount: ""
402
403 ## Memory limit hard % (between 1-100) of the memory limit.
404 memoryLimitHardPercentage: 85
405
406
407 tserver:
408 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
409 ## This might override the default affinity from service.yaml
410 # To successfully merge, we need to follow rules for merging nodeSelectorTerms that kubernentes
411 # has. Each new node selector term is ORed together, and each match expression or match field in
412 # a single selector is ANDed together.
413 # This means, if a pod needs to be scheduled on a label 'custom_label_1' with a value
414 # 'custom_value_1', we need to add this 'subterm' to each of our pre-defined node affinity
415 # terms.
416 #
417 # Pod anti affinity is a simpler merge. Each term is applied separately, and the weight is tracked.
418 # The pod that achieves the highest weight is selected.
419 ## Example.
420 # affinity:
421 #   podAntiAffinity:
422 #     requiredDuringSchedulingIgnoredDuringExecution:
423 #       - labelSelector:
424 #         matchExpressions:
425 #           - key: app
426 #             operator: In
427 #             values:
428 #               - "yb-tserver"
429 #     topologyKey: kubernetes.io/hostname
430 # For further examples, see examples/yugabyte/affinity_overrides.yaml
431 affinity: {}
432
433 ## Extra environment variables passed to the TServer pods.
434 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
435 ## Example:
436 # extraEnv:
437 # - name: NODE_IP
438 #   valueFrom:
439 #     fieldRef:
440 #       fieldPath: status.hostIP
441 extraEnv: []
442
443 ## secretEnv variables are used to expose secrets data as env variables in the tserver pods.

```

```

444 ## If namespace field is not specified we assume that user already
445 ## created the secret in the same namespace as DB pods.
446 ## Example
447 # secretEnv:
448 # - name: MYSQL_LDAP_PASSWORD
449 #   valueFrom:
450 #     secretKeyRef:
451 #       name: secretName
452 #       namespace: my-other-namespace-with-ldap-secret
453 #       key: password
454 secretEnv: []
455
456 ## Annotations to be added to the TServer pods.
457 podAnnotations: {}
458
459 ## Labels to be added to the TServer pods.
460 podLabels: {}
461
462 ## Tolerations to be added to the TServer pods.
463 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
464 ## Example:
465 # tolerations:
466 # - key: dedicated
467 #   operator: Equal
468 #   value: experimental
469 #   effect: NoSchedule
470 tolerations: []
471
472 ## Sets the --server_broadcast_addresses flag on the TServer, no
473 ## preflight checks are done for this address. You might need to add
474 ## 'use_private_ip: cloud' to the gflags.master and gflags.tserver.
475 serverBroadcastAddress: ""
476
477 ## Extra volumes
478 ## extraVolumesMounts are mandatory for each extraVolumes.
479 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
480 ## Example:
481 # extraVolumes:
482 # - name: custom-nfs-vol
483 #   persistentVolumeClaim:
484 #     claimName: some-nfs-claim
485 extraVolumes: []
486
487 ## Extra volume mounts
488 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volumemount-v1-core
489 ## Example:
490 # extraVolumeMounts:
491 # - name: custom-nfs-vol
492 #   path: /home/yugabyte/nfs-backup
493 extraVolumeMounts: []
494
495 ## Set service account for tserver DB pods. The service account
496 ## should exist in the namespace where the tserver DB pods are brought up.
497 serviceAccount: ""

```

```

498
499 ## Memory limit hard % (between 1-100) of the memory limit.
500 memoryLimitHardPercentage: 85
501
502
503 helm2Legacy: false
504
505 ip_version_support: "v4_only" # v4_only, v6_only are the only supported values at the moment
506
507 # For more https://docs.yugabyte.com/latest/reference/configuration/yugabyted/#environment-variables
508 authCredentials:
509   ysql:
510     user: "yadmin"
511     password: "TES2&Daggerfall"
512     database: ""
513   ycql:
514     user: ""
515     password: ""
516     keyspace: ""
517
518 oldNamingStyle: true
519
520 preflight:
521   # Set to true to skip disk IO check, DNS address resolution, and
522   # port bind checks
523   skipAll: false
524   # Set to true to skip port bind checks
525   skipBind: false
526
527   ## Set to true to skip ulimit verification
528   ## SkipAll has higher priority
529   skipUlimit: false
530
531 ## Pod securityContext
532 ## Ref: https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#security-context
533 ## The following configuration runs YB-Master and YB-TServer as a non-root user
534 podSecurityContext:
535   enabled: false
536   ## Mark it false, if you want to stop the non root user validation
537   runAsNonRoot: true
538   fsGroup: 10001
539   runAsUser: 10001
540   runAsGroup: 10001
541
542 ## Added to handle old universe which has volume annotations
543 ## K8s universe <= 2.5 to >= 2.6
544 legacyVolumeClaimAnnotations: false

```

Listing 24: yugabyteDB - Helm Chart Manifest

Die Installation erfolgt dann wie folgt:

```
1 helm install yw-test yugabytedb/yugabyte --version 2.19.3 -n yb-platform -f /home/gramic/PycharmProjects/rke2_settings/yugabytedb/yugabytedb/values.yaml
```

Listing 25: yugabyteDB - Installation

VII.II.III Rekonfiguration mit 250GiB Storage

VII.II.III.I Bereinigen

Zuerst wurde YugabyteDB deinstalliert und alle bestehenden Daten gelöscht:

```
1 helm delete yw-evaluation -n yb-platform
2 kubectl delete pvc --namespace yb-platform -l app=yb-master
3 kubectl delete pvc --namespace yb-platform -l app=yb-tserver
4 kubectl delete namespace yb-platform
5 kubectl delete pv yb-storage-pv
6 kubectl delete storageclass yb-storage
```

Listing 26: YugabyteDB - Deinstallieren

VII.II.III.II StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden. Wichtig ist hier, dass die Node Affinity gesetzt wird, damit die Persistence Volumes auf den Node geschrieben werden:

```
1 # https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/prepare-environment/kubernetes/#configure-storage-class
2 # https://github.com/rancher/local-path-provisioner
3 apiVersion: storage.k8s.io/v1
4 kind: StorageClass
5 metadata:
6   name: yb-storage
7 provisioner: rancher.io/local-path
8 parameters:
9   nodePath: /srv/data/local-path-provisioner
10 volumeBindingMode: WaitForFirstConsumer
11 reclaimPolicy: Delete
12 --
13 apiVersion: v1
14 kind: PersistentVolume
15 metadata:
16   name: yb-storage-pv
17   labels:
18     type: local
19 spec:
20   accessModes:
21     - ReadWriteOnce
22   capacity:
23     storage: 1Gi
24   storageClassName: "yb-storage"
25   hostPath:
26     path: /srv/data/local-path-provisioner
27   nodeAffinity:
28     required:
29       nodeSelectorTerms:
30         - matchExpressions:
31           - key: kubernetes.io/hostname
32             operator: In
33             values:
34               - sks1183
35               - sks1184
```

Listing 27: YugabyteDB - StorageClass setzen

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/yugabytedb/yugabytedb/storageclass_250gib.yaml
```

Listing 28: YugabyteDB - StorageClass / PersistentVolume Grosses Volumen aktivieren

VII.II.III.III Installation - 250GiB

Zuerst muss wieder der Namespace erstellt werden:

```
1 kubectl create namespace yb-platform
```

Listing 29: YugabyteDB - Namespace 250GiB

Das values.yaml wurde für den letzten grossen Test angepasst.

Es werden nun 12GiB Memory allokiert für die tserver Nodes.

Es werden PVCs erstellt mit 240GiB Volume:

```
1 # Default values for Yugabyte.
2 # This is a YAML-formatted file.
3 # Declare variables to be passed into your templates.
4 Component: "yugabytedb"
5
6 fullnameOverride: ""
7 nameOverride: ""
8
9 Image:
10   repository: "yugabytedb/yugabyte"
11   tag: 2.20.2.1-b3
12   pullPolicy: IfNotPresent
13   pullSecretName: ""
14
15 storage:
16   ephemeral: false # will not allocate PVs when true
17   master:
18     count: 1
19     size: 2Gi
20     storageClass: "yb-storage"
21   tserver:
22     count: 1
23     size: 240Gi
24     storageClass: "yb-storage"
25
26 resource:
27   master:
28     requests:
29       cpu: "2"
30       memory: 1Gi
31     limits:
32       cpu: "2"
33       memory: 1Gi
34   tserver:
35     requests:
```

```

36   cpu: "2"
37   memory: 12Gi
38   limits:
39     cpu: "2"
40     memory: 12Gi
41
42 replicas:
43   master: 3
44   tserver: 3
45   ## Used to set replication factor when isMultiAz is set to true
46   totalMasters: 3
47
48 partition:
49   master: 0
50   tserver: 0
51
52 updateStrategy:
53   type: RollingUpdate
54
55 # Used in Multi-AZ setup
56 masterAddresses: ""
57
58 isMultiAz: false
59 AZ: ""
60
61 # Disable the YSQL
62 disableYsql: false
63
64 tls:
65   # Set to true to enable the TLS.
66   enabled: false
67   nodeToNode: true
68   clientToServer: true
69   # Set to false to disallow any service with unencrypted communication from joining this cluster
70   insecure: false
71   # Set enabled to true to use cert-manager instead of providing your own rootCA
72   certManager:
73     enabled: false
74     # Will create own ca certificate and issuer when set to true
75     bootstrapSelfsigned: true
76     # Use ClusterIssuer when set to true, otherwise use Issuer
77     useClusterIssuer: false
78     # Name of ClusterIssuer to use when useClusterIssuer is true
79     clusterIssuer: cluster-ca
80     # Name of Issuer to use when useClusterIssuer is false
81     issuer: Yugabyte-ca
82   certificates:
83     duration: 2160h # 90d
84     renewBefore: 360h # 15d
85     algorithm: RSA # ECDSA or RSA
86     # Can be 2048, 4096 or 8192 for RSA
87     # Or 256, 384 or 521 for ECDSA
88     keySize: 2048
89   rootCA:

```

```

90   cert: ""
91   LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ01CQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJrd0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakFlRncweE9UQX1NRGd3TURRd01qSmFGdzB5T1RBeU1EVXdNRFF3TWpK
92   =
93   key: ""
94   LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBx0XJpWDZqRXJ1YnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQLOJ1M2RxMW4yb0RDZkZUTDB4aTIOV01kTFFyckEy
95   =
96   nodeCert:
97     cert: ""
98     key: ""
99   clientCert:
100    cert: ""
101    key: ""
102
103 gflags:
104   master:
105     default_memory_limit_to_ram_ratio: 0.85
106   tserver:
107     ysql_beta_features: true          # gramic, wird üfr vacuum öbentigt
108     follower_unavailable_considered_failed_sec: 300 # gramic, autobalancing auf 5 minuten heruntersetzen
109
110 PodManagementPolicy: Parallel
111
112 enableLoadBalancer: true
113
114 ybc:
115   enabled: false
116
117 ybCleanup:
118   resources:
119     requests:
120       cpu: "1"
121       memory: 0.5Gi
122     limits:
123       cpu: "1"
124       memory: 0.5Gi
125
126 domainName: "cluster.local"
127
128 serviceEndpoints:
129   - name: "yb-master-ui"
130     type: LoadBalancer
131     annotations:
132       metallb.universe.tf/loadBalancerIPs: 10.0.20.151
133     clusterIP: ""
134     ## Sets the Service's externalTrafficPolicy
135     externalTrafficPolicy: "Cluster"
136     app: "yb-master"
137     loadBalancerIP: ""
138     ports:
139       http-ui: "7000"
140
141   - name: "yb-tserver-service"
142     type: LoadBalancer

```

```

140 annotations:
141   metallb.universe.tf/loadBalancerIPs: 10.0.20.106
142 clusterIP: ""
143 ## Sets the Service's externalTrafficPolicy
144 externalTrafficPolicy: "Cluster"
145 app: "yb-tserver"
146 loadBalancerIP: ""
147 ports:
148   tcp-yql-port: "9042"
149   tcp-yedis-port: "6379"
150   tcp-ysql-port: "5433"
151
152 Services:
153 - name: "yb-masters"
154   label: "yb-master"
155   skipHealthChecks: false
156   memory_limit_to_ram_ratio: 0.85
157   ports:
158     http-ui: "7000"
159     tcp-rpc-port: "7100"
160
161 - name: "yb-tservers"
162   label: "yb-tserver"
163   skipHealthChecks: false
164   ports:
165     http-ui: "9000"
166     tcp-rpc-port: "9100"
167     tcp-yql-port: "9042"
168     tcp-yedis-port: "6379"
169     tcp-ysql-port: "5433"
170     http-ycql-met: "12000"
171     http-yedis-met: "11000"
172     http-ysql-met: "13000"
173     grpc-ybc-port: "18018"
174
175
176 ## Should be set to true only if Istio is being used. This also adds
177 ## the Istio sidecar injection labels to the pods.
178 ## TODO: remove this once
179 ## https://github.com/yugabyte/yugabyte-db/issues/5641 is fixed.
180 ##
181 istioCompatibility:
182   enabled: false
183
184 ## Settings required when using multicluster environment.
185 multicluster:
186   ## Creates a ClusterIP service for each yb-master and yb-tserver
187   ## pod.
188   createServicePerPod: false
189   ## creates a ClusterIP service whos name does not have release name
190   ## in it. A common service across different clusters for automatic
191   ## failover. Useful when using new naming style.
192   createCommonTserverService: false
193

```

```

194 ## Enable it to deploy YugabyteDB in a multi-cluster services enabled
195 ## Kubernetes cluster (KEP-1645). This will create ServiceExport.
196 ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#registering_a_service_for_export
197 ## You can use this gist for the reference to deploy the YugabyteDB in a multi-cluster scenario.
198 ## Gist - https://gist.github.com/baba230896/78cc9bb6f4ba0b3d0e611cd49ed201bf
199 createServiceExports: false
200
201 ## Mandatory variable when createServiceExports is set to true.
202 ## Use: In case of GKE, you need to pass GKE Hub Membership Name.
203 ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#enabling
204 kubernetesClusterId: ""
205
206 ## mcsApiVersion is used for the MCS resources created by the
207 ## chart. Set to net.gke.io/v1 when using GKE MCS.
208 mcsApiVersion: "multicloud.x-k8s.io/v1alpha1"
209
210 serviceMonitor:
211     ## If true, two ServiceMonitor CRs are created. One for yb-master
212     ## and one for yb-tserver
213     ## https://github.com/coreos/prometheus-operator/blob/master/Documentation/api.md#servicemonitor
214     ##
215     enabled: false
216     ## interval is the default scrape_interval for all the endpoints
217     interval: 30s
218     ## extraLabels can be used to add labels to the ServiceMonitors
219     ## being created
220     extraLabels: {}
221     # release: prom
222
223     ## Configurations of ServiceMonitor for yb-master
224 master:
225     enabled: true
226     port: "http-ui"
227     interval: ""
228     path: "/prometheus-metrics"
229
230     ## Configurations of ServiceMonitor for yb-tserver
231 tserver:
232     enabled: true
233     port: "http-ui"
234     interval: ""
235     path: "/prometheus-metrics"
236 ycql:
237     enabled: true
238     port: "http-ycql-met"
239     interval: ""
240     path: "/prometheus-metrics"
241 ysql:
242     enabled: true
243     port: "http-ysql-met"
244     interval: ""
245     path: "/prometheus-metrics"
246 yedis:
247     enabled: true

```

```

248   port: "http-yedis-met"
249   interval: ""
250   path: "/prometheus-metrics"
251
252 commonMetricRelabelings:
253 # https://git.io/JJW5p
254 # Save the name of the metric so we can group_by since we cannot by __name__ directly...
255 - sourceLabels: ["__name__"]
256   regex: "(.*"
257   targetLabel: "saved_name"
258   replacement: "$1"
259 # The following basically retrofit the handler_latency_* metrics to label format.
260 - sourceLabels: ["__name__"]
261   regex: "handler_latency_(yb_[^_]*)([^_]*)([^_]*)(.*"
262   targetLabel: "server_type"
263   replacement: "$1"
264 - sourceLabels: ["__name__"]
265   regex: "handler_latency_(yb_[^_]*)([^_]*)([^_]*)(.*"
266   targetLabel: "service_type"
267   replacement: "$2"
268 - sourceLabels: ["__name__"]
269   regex: "handler_latency_(yb_[^_]*)([^_]*)([^_]*)(_sum|_count)?"
270   targetLabel: "service_method"
271   replacement: "$3"
272 - sourceLabels: ["__name__"]
273   regex: "handler_latency_(yb_[^_]*)([^_]*)([^_]*)(_sum|_count)?"
274   targetLabel: "__name__"
275   replacement: "rpc_latency$4"
276
277 resources: {}
278
279 nodeSelector: {}
280
281 affinity: {}
282
283 statefulSetAnnotations: {}
284
285 networkAnnotation: {}
286
287 commonLabels: {}
288
289 master:
290   ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
291   # For further examples, see examples/yugabyte/affinity_overrides.yaml
292   affinity: {}
293
294   ## Extra environment variables passed to the Master pods.
295   ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
296   extraEnv: []
297
298   # secretEnv variables are used to expose secrets data as env variables in the master pod.
299   # TODO Add namespace also to support copying secrets from other namespace.
300   secretEnv: []
301

```

```

302 ## Annotations to be added to the Master pods.
303 podAnnotations: {}
304
305 ## Labels to be added to the Master pods.
306 podLabels: {}
307
308 ## Tolerations to be added to the Master pods.
309 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
310 tolerations: []
311
312 ## Extra volumes
313 ## extraVolumesMounts are mandatory for each extraVolumes.
314 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
315 extraVolumes: []
316
317 ## Extra volume mounts
318 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volumemount-v1-core
319 extraVolumeMounts: []
320
321 ## Set service account for master DB pods. The service account
322 ## should exist in the namespace where the master DB pods are brought up.
323 serviceAccount: ""
324
325 ## Memory limit hard % (between 1-100) of the memory limit.
326 memoryLimitHardPercentage: 85
327
328
329 tserver:
330 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
331 ## This might override the default affinity from service.yaml
332 affinity: {}
333
334 ## Extra environment variables passed to the TServer pods.
335 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
336 extraEnv: []
337
338 ## secretEnv variables are used to expose secrets data as env variables in the tserver pods.
339 ## If namespace field is not specified we assume that user already
340 ## created the secret in the same namespace as DB pods.
341 secretEnv: []
342
343 ## Annotations to be added to the TServer pods.
344 podAnnotations: {}
345
346 ## Labels to be added to the TServer pods.
347 podLabels: {}
348
349 ## Tolerations to be added to the TServer pods.
350 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
351 tolerations: []
352
353 ## Sets the --server_broadcast_addresses flag on the TServer, no
354 ## preflight checks are done for this address. You might need to add
355 ## 'use_private_ip: cloud' to the gflags.master and gflags.tserver.

```

```

356 serverBroadcastAddress: ""
357
358 ## Extra volumes
359 ## extraVolumesMounts are mandatory for each extraVolumes.
360 extraVolumes: []
361
362 ## Extra volume mounts
363 extraVolumeMounts: []
364
365 ## Set service account for tserver DB pods. The service account
366 ## should exist in the namespace where the tserver DB pods are brought up.
367 serviceAccount: ""
368
369 ## Memory limit hard % (between 1-100) of the memory limit.
370 memoryLimitHardPercentage: 85
371
372 helm2Legacy: false
373
374 ip_version_support: "v4_only" # v4_only, v6_only are the only supported values at the moment
375
376 # For more https://docs.yugabyte.com/latest/reference/configuration/yugabyted/#environment-variables
377 authCredentials:
378   ysql:
379     user: "yadmin"
380     password: "TES2&Daggerfall"
381     database: ""
382   ycql:
383     user: ""
384     password: ""
385     keyspace: ""
386
387 oldNamingStyle: true
388
389 preflight:
390   # Set to true to skip disk IO check, DNS address resolution, and
391   # port bind checks
392   skipAll: false
393   # Set to true to skip port bind checks
394   skipBind: false
395
396   ## Set to true to skip ulimit verification
397   ## SkipAll has higher priority
398   skipUlimit: false
399
400 ## Pod securityContext
401 ## Ref: https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#security-context
402 ## The following configuration runs YB-Master and YB-TServer as a non-root user
403 podSecurityContext:
404   enabled: false
405   ## Mark it false, if you want to stop the non root user validation
406   runAsNonRoot: true
407   fsGroup: 10001
408   runAsUser: 10001
409   runAsGroup: 10001

```

```

410
411 ## Added to handle old universe which has volume annotations
412 ## K8s universe <= 2.5 to >= 2.6
413 legacyVolumeClaimAnnotations: false

```

Listing 30: YugabyteDB - Helm Chart Manifest 250GiB

VII.II.IV SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle `pgbench_eval_bench` erstellt.

Zudem wird je ein Tablespace für die Indizes (`eval_index_tablespace`) und Daten (`eval_data_tablespace`) erstellt.

Die Tablespace werden auf die drei Tablet-Server verteilt.

Um die Tablets zu verteilen, müssen die Cloud, Region und Zone mitgegeben werden.

Dies wird folgendermassen via SQL ausgelesen:

```

1 select
2   cloud,
3   region,
4   zone
5 from yb_servers();

```

Listing 31: YugabyteDB - Cloud - Region - Zone

Mit diesen Informationen lassen sich jetzt die Replikation für die Tablets einstellen.

Es muss auf drei Replikas repliziert werden:

```

1 -- Tabelle pgbench_eval_bench erstellen
2 drop database if exists pgbench_eval_bench;
3 create database pgbench_eval_bench;
4
5 -- Tablespace für Indices
6 drop tablespace if exists eval_index_tablespace;
7 CREATE TABLESPACE eval_index_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');
8
9 -- Genereller Tablespace erstellen
10 drop tablespace if exists eval_data_tablespace;
11 CREATE TABLESPACE eval_data_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');

```

Listing 32: YugabyteDB - Benchmarking - DB erstellen

Die Tabellen werden automatisch von `ysql_bench` beim Initialisieren erstellt, daher sind keine weiteren Schritte notwendig.

Die Grösse der Tabellen lässt sich mit folgendem SQL auslesen:

```

1 select
2   table_name,
3   pg_size.pretty(pg_total_relation_size(quote_ident(table_name))), 
4   pg_total_relation_size(quote_ident(table_name))
5 from information_schema.tables
6 where table_schema = 'public'
7 order by 3 desc;

```

Listing 33: YugabyteDB - Benchmarking - Table Size

VII.II.V SQL Statements - Testing

Entsprechend dem ERD müssen die Tabellen erstellt werden: [Evaluation - ERD self_healing_test](#) Die Tabelle heisst entsprechend `self_healing_test`. Auch hier soll ein Index-Tablespace (`self_healing_indices_tablespace`) und ein Daten-Tablespace (`self_healing_datas_tablespace`) erstellt werden.

Neu dazu kommt der Longtext-Tablespace `self_healing_longtexts_tablespace`. Erst werden die Rollen erstellt, gefolgt von den Usern und Schemas.

Die Schemas müssen entsprechend mittels GRANT berechtigt werden. Die Tabellen müssen entsprechend den Schemas erstellt werden, wobei einige Tabellen in 3 Tablets pro tserver gesplittet werden sollen (Also insgesamt 9). Das gesamte CREATE-Skript:

```
1 -- self-healing-Tabelle
2 create database self_healing_test;
3
4 -- Tablespace für Indices
5 drop tablespace if exists self_healing_indices_tablespace;
6 CREATE TABLESPACE self_healing_indices_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');
7
8 -- Genereller Tablespace erstellen
9 drop tablespace if exists self_healing_datas_tablespace;
10 CREATE TABLESPACE self_healing_datas_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');
11
12 -- longtext Tablespace erstellen
13 drop tablespace if exists self_healing_longtexts_tablespace;
14 CREATE TABLESPACE self_healing_longtexts_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');
15
16 -- Rollen erstellen
17 drop role if exists hrm;
18 create role hrm;
19 drop role if exists accountands;
20 create role accountands;
21 drop role if exists customer_service_officers;
22 create role customer_service_officers;
23 drop role if exists legal_affairs;
24 create role legal_affairs;
25
26 -- User erstellen
27 drop user if exists hrm_1;
28 drop user if exists hrm_2;
29 create user hrm_1 with password 'hrm1' role hrm;
30 create user hrm_2 with password 'hrm2' role hrm;
31
32 drop user if exists cso_1;
33 drop user if exists cso_2;
34 create user cso_1 with password 'cso1' role customer_service_officers;
35 create user cso_2 with password 'cso2' role customer_service_officers;
36
37 drop user if exists la_1;
38 drop user if exists la_2;
39 create user la_1 with password 'la1' role legal_affairs;
40 create user la_2 with password 'la2' role legal_affairs;
41
42 -- Schemas erstellen
43 drop schema if exists hrm;
```

```

44 create schema hrm authorization hrm;
45 drop schema if exists accountands;create schema accountands authorization accountands;
46 drop schema if exists customer_serviceOfficers;
47 create schema customer_serviceOfficers authorization customer_serviceOfficers;
48 drop schema if exists generell;
49 create schema generell;
50
51 -- GRANTS erstellen
52 grant all on all tables in schema hrm to legal_affairs;
53 grant all on all tables in schema accountands to legal_affairs;
54 grant all on all tables in schema customer_serviceOfficers to legal_affairs;
55 grant all on all tables in schema generell to legal_affairs;
56 grant all on all tables in schema hrm to yadmin;
57 grant all on all tables in schema accountands to yadmin;
58 grant all on all tables in schema customer_serviceOfficers to yadmin;
59 grant all on all tables in schema generell to yadmin;
60
61 -- self_healing_accounts für Schema customer_serviceOfficers
62 drop table if exists customer_serviceOfficers.self_healing_accounts;
63 create table customer_serviceOfficers.self_healing_accounts (
64     account_id int primary key,
65     firstname varchar(255) not null,
66     lastname varchar(255) not null,
67     birthday date not null,
68     postal_code varchar(50),
69     street varchar(255),
70     country_code varchar(2),
71     phone varchar(25),
72     mail varchar(255) check (mail like '%@%')
73 ) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
74 create unique index accounts_personal_mark on customer_serviceOfficers.self_healing_accounts(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;
75
76 -- self_healing_employees für Schema hrm
77 drop table if exists hrm.self_healing_employees;
78 create table hrm.self_healing_employees (
79     employees_id int primary key,
80     firstname varchar(255) not null,
81     lastname varchar(255) not null,
82     birthday date not null,
83     postal_code varchar(50),
84     street varchar(255),
85     country_code varchar(2),
86     phone varchar(25),
87     mail varchar(255) check (mail like '%@%')
88 ) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
89 create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;
90
91 -- self_healing_accountand_protocol für Schema accountands
92 drop table if exists accountands.self_healing_accountand_protocol;
93 create table accountands.self_healing_accountand_protocol (
94     acc_protocol_id int primary key,
95     description varchar(100) not null,
96     protocol_date date not null,
97     employees_id int not null,

```

```

98     rapport TEXT,
99     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict) tablespace self_healing_longtexts_tablespace SPLIT
100    INTO 3 TABLETS;
101
102 -- self_healing_intranet für public Schema
103 drop table if exists generell.self_healing_intranet;
104 create table generell.self_healing_intranet (
105     intranet_id int primary key,
106     content text
107 ) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;
108
109 -- self_healing_intranet für public Schema
110 drop table if exists generell.self_healing_intranet_users;
111 create table generell.self_healing_intranet_users (
112     intranet_user_id int primary key,
113     employees_id int not null,
114     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
115 );
116 create unique index intranet_unique_combi on generell.self_healing_intranet_users(intranet_user_id, employees_id);

```

Listing 34: YugabyteDB - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```

1 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
2 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
3 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
4
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
6 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
7 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
8
9 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (100, 'bla', '07.04.2024', 100, 'blabla');
10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (200, 'yada', '07.04.2024', 100, 'ydayadyada',
11 );
11 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (300, 'something', '07.04.2024', 300, 'something');
12
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
15 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');
16
17 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(100, 100);
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(200, 200);
19 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(300, 300);
20
21 select * from customer_serviceOfficers.self_healing_accounts;
22 select * from hrm.self_healing_employees;
23 select * from accountands.self_healing_accountand_protocol;
24 select * from generell.self_healing_intranet_users;

```

Listing 35: YugabyteDB - Self Healing Tests - Init Data

Während des Failover-Test müssen Daten beschrieben werden:

```

1 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');

```

```

2 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
3 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
4 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005'); insert into hrm.self_healing_employees (employees_id,
    firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
6
7 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (400, 'bla', '07.04.2024', 200, 'blabla');
8 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (500, 'yada', '07.04.2024', 600, 'ydayadyada');
9
10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (1000, 'something', '07.04.2024', 300, 'something');
11
12 insert into generell.self_healing_intranet(intranet_id, content) VALUES (200, 'yadada');
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (600, 'bla bla');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (900, 'talking and talking');
15
16 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(400, 400);
17 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(500, 500);
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(600, 600);
19
20 select * from customer_serviceOfficers.self_healing_accounts;
21 select * from hrm.self_healing_employees;
22 select * from accountands.self_healing_accountand_protocol;
23 select * from generell.self_healing_intranet;
24 select * from generell.self_healing_intranet_users;

```

Listing 36: YugabyteDB - Self Healing Tests - Failover Data

Nach dem Recovery müssen die Daten entsprechend vorhanden sein und es müssen weitere Daten beschrieben werden können:

```

1 select * from customer_serviceOfficers.self_healing_accounts;
2 select * from hrm.self_healing_employees;
3 select * from accountands.self_healing_accountand_protocol;
4 select * from generell.self_healing_intranet;
5 select * from generell.self_healing_intranet_users;
6
7 insert into generell.self_healing_intranet(intranet_id, content) VALUES (700, 'yadada');
8 insert into generell.self_healing_intranet(intranet_id, content) VALUES (800, 'bla bla');
9 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1100, 'talking and talking');
10
11 select * from customer_serviceOfficers.self_healing_accounts;
12 select * from hrm.self_healing_employees;
13 select * from accountands.self_healing_accountand_protocol;
14 select * from generell.self_healing_intranet;
15 select * from generell.self_healing_intranet_users;

```

Listing 37: YugabyteDB - Self Healing Tests - Recovery Data

VII.III sks9016 - yugabyteDB

VII.III.I yugabyteDB - Download und Installation yugabyteDB

Ohne yugabyteDB zu installieren, lässt sich ysql_bench nicht ausführen. Daher muss das ganze Package erst heruntergeladen werden:

```
1 root@sks9016:~# wget https://downloads.yugabyte.com/releases/2.21.0.0/yugabyte-2.21.0.0-b545-linux-x86_64.tar.gz
```

Listing 38: sks9016 - Download yugabyteDB On-Premise

Im nächsten Schritt wird es im /opt entpackt und das post_install.sh-Skript ausgeführt:

```

1 root@sks9016:/opt# tar xvfz yugabyte-2.21.0.0-b545-linux-x86_64.tar.gz && cd yugabyte-2.21.0.0/
2 ...
3 root@sks9016:/opt/yugabyte-2.21.0.0# ./bin/post_install.sh
4

```

Listing 39: sks9016 - Installation yugabyteDB On-Premise

Um nun zu testen, ob das Ganze funktioniert, kann eine Verbindung zum Evaluationssystem hergestellt werden:

```

1 root@sks9016:/opt/yugabyte-2.21.0.0# cd /opt/yugabyte-2.21.0.0/postgres/bin/
2 root@sks9016:/opt/yugabyte-2.21.0.0/postgres/bin# ./ysqlsh "host=10.0.20.106 user=yadmin"
3 Password for user yadmin:
4 ysqlsh (11.2-YB-2.21.0.0-b0)
5 Type "help" for help.
6
7 No entry for terminal type "xterm-256color";
8 using dumb terminal settings.
9 yugabyte=# exit
10

```

Listing 40: sks9016 - Check yugabyteDB On-Premise

Damit ist der Benchmarking-Server ready.

VII.IV Patroni

VII.IV.I Prerequisites

Ganz am Anfang steht die Firewall.

Die Rules müssen auf sks1232, sks1233, sks1234 und sks9016 gesetzt werden:

```

1 # sks1232 / sks1233 / sks1234 / sks9016(10.0.28.16)
2 nano /etc/iptables/rules.v4
3 *filter
4 :INPUT ACCEPT [0:0]
5 :FORWARD ACCEPT [0:0]
6 :OUTPUT ACCEPT [0:0]
7 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
8 -A INPUT -s 10.0.9.115/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.115" -j ACCEPT
9 -A INPUT -s 10.0.9.76/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.76" -j ACCEPT
10 -A INPUT -s 10.0.36.147/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.36.147" -j ACCEPT
11 -A INPUT -s 10.0.9.35/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.35" -j ACCEPT
12 -A INPUT -s 10.0.9.37/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.37" -j ACCEPT
13 -A INPUT -s 10.0.9.74/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.74" -j ACCEPT
14 -A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.75" -j ACCEPT
15 -A INPUT -s 10.0.9.36/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.36" -j ACCEPT
16 -A INPUT -s 10.0.9.14/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for probe 10.0.9.14" -j ACCEPT
17 -A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
18 # generell

```

```

19 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 443 -j ACCEPT
20 # postgres
21 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
22 # patroni-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2379 -j ACCEPT
23 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2380 -j ACCEPT
24 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2376 -j ACCEPT
25 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6432 -j ACCEPT
26 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8008 -j ACCEPT
27 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 7000 -j ACCEPT
28 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8080 -j ACCEPT
29 COMMIT
30 # Completed
31
32 systemctl restart iptables
33 systemctl status iptables

```

Listing 41: Patroni - Firewall Settings

Danach muss der Proxy gesetzt werden:

```

1 # sks1232 / sks1233 / sks1234
2 # Proxy setzen
3 # nano /etc/profile.d/proxy.sh
4 export https_proxy=http://sproxy.sivc.first-it.ch:8080
5 export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
6 export http_proxy=http://sproxy.sivc.first-it.ch:8080
7 export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
8 export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
9 export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
10 # source /etc/profile.d/proxy.sh

```

Listing 42: Patroni - Proxy Settings

Damit das PostgreSQL-Repository eingebunden werden kann,
muss dem apt-Proxy gesetzt werden.

Da via Foreman installiert wurde, muss dieser ausgenommen werden:

```

1 # sks1232 / sks1233 / sks1234
2 # apt-Proxy setzen
3 # nano /etc/apt/apt.conf.d/proxy.conf
4 Acquire::http::Proxy "http://sproxy.sivc.first-it.ch:8080";
5 Acquire::https::Proxy "http://sproxy.sivc.first-it.ch:8080";
6 Acquire::http::proxy::foreman.ksgr.ch "DIRECT";

```

Listing 43: Patroni - apt-Proxy Settings

Im nächsten Schritt kann das PostgreSQL-Repository eingebunden werden.

 Achtung, die von PostgreSQL beschriebene Variante wurde in Debian 10 als Deprecated gesetzt, mit Debian 13 wird diese Repository-Integration zu einem Fehler werden.

```

1 # sks1232 / sks1233 / sks1234
2 # PostgreSQL Repository einbinden
3 sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
4 wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
5
6 # Ausloggen und wieder einloggen

```

```
7 apt update
```

Listing 44: Patroni - PostgreSQL einbinden

Nun muss der PostgreSQL Cluster, Patroni, python3-etcd und python3-psycopg2 installiert werden:

```
1 apt install postgresql-16 postgresql-server-dev-16 patroni python3-etcd python3-psycopg2
```

Listing 45: Patroni - Prerequisites installieren

Im nächsten Schritt müssen Patroni und der PostgreSQL Cluster gestoppt werden:

```
1 systemctl stop postgresql patroni
```

Listing 46: Patroni - Stop Patroni und PostgreSQL

Anschliessend muss noch vom PostgreSQL-Verzeichnis /usr/lib/postgresql/16/bin/ ein Symlink nach /usr/sbin/ gesetzt werden:

```
1 ln -s /usr/lib/postgresql/16/bin/* /usr/sbin/
```

Listing 47: Patroni - Symlink binaries

Zu guter Letzt sollte geprüft werden, ob alle Versionen passen und am richtigen Ort sind:

```
1 which patroni
2 which psql
3 patroni --version
```

Listing 48: Patroni - Checks

Damit kann zum etcd übergegangen werden.

VII.IV.II Installation etcd

Auf sks9016 sollte erst das Repository angepasst werden und anschliessend der etcd-server installiert werden:

```
1 apt update
2 apt install etcd-server
```

Listing 49: etcd - Installation

Die Konfiguration ist simpel.

Die IP muss gesetzt werden, ein Listener auf Localhost und IP gesetzt werden:

```
1 # nano /etc/default/etcd
2 ETCD_LISTEN_PEER_URLS="http://10.0.28.16:2380"
3 ETCD_LISTEN_CLIENT_URLS="http://localhost:2379,http://10.0.28.16:2379"
4 ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.28.16:2380"
5 ETCD_INITIAL_CLUSTER="default=http://10.0.28.16:2380,"
6 ETCD_ADVERTISE_CLIENT_URLS="http://10.0.28.16:2379"
7 ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
8 ETCD_INITIAL_CLUSTER_STATE="new"
```

Listing 50: etcd - Konfiguration

Der Service sollte neu gestartet und seine Lauffähigkeit getestet werden:

```
1 systemctl restart etcd
2 systemctl is-enabled etcd
3 systemctl status etcd
```

Listing 51: etcd - restart

Es sollte nun ein Member gelistet werden:

```
1 etcdctl member list
```

Listing 52: etcd - member list

Damit kann nun das Bootstrapping gestartet werden.

VII.IV.III Bootstrapping

Zuerst müssen die yml-Konfigurationen gesetzt werden.

Am besten wird das bestehende File jeweils gelöscht:

```
1 rm /etc/patroni/config.yml
2 nano /etc/patroni/config.yml
```

Listing 53: Patroni Bootstrap - Konfiguration bereinigen

Die yml-Files sehen wie folgt aus:

```
1 # Scope of PostgreSQL
2 scope: postgres
3 # Namespace for the PostgreSQL database
4 namespace: /db/
5 # Name of the PostgreSQL instance
6 name: postgres01
7 # Patroni REST API Configuration
8 restapi:
9     # The IP address and port on which the REST API should listen
10    listen: 10.0.20.110:8008
11
12    # The IP address and port to which clients should connect
13    connect_address: 10.0.20.110:8008
14 # Patroni Etcd Configuration
15 etcd3:
16    # The host address and port of the Etcd server
17    host: 10.0.28.16:2379
18 # Patroni Bootstrap Configuration
19 bootstrap:
20    # Configuration parameters for distributed configuration store (DCS)
21    dcs:
22        ttl: 30
23        loop_wait: 10
24        retry_timeout: 10
25        maximum_lag_on_failover: 1048576
26        postgresql:
27            # Use pg_rewind during bootstrap
28            use_pg_rewind: true
29            # Change pg_wal
30            create_replica_methods:
31                - basebackup
32            basebackup:
33                max_rate: '100M'
34                waldir: "/srv/data/pg_wal"
35        # Initdb configuration
36        # Initdb configuration
```

```

37 initdb:
38   - auth: scram-sha-256
39   - encoding: UTF8
40   - data-checksums
41 # pg_hba.conf entries for replication and general access
42 pg_hba:
43   - host replication replicator 127.0.0.1/32 scram-sha-256
44   - host replication replicator 10.0.20.110/0 scram-sha-256
45   - host replication replicator 10.0.20.111/0 scram-sha-256
46   - host replication replicator 10.0.20.112/0 scram-sha-256
47   - host all all 0.0.0.0/0 scram-sha-256
48 # Adding default user admin with password admin
49 users:
50   admin:
51     password: admin
52     options:
53       - createrole
54       - createdb
55 # Patroni PostgreSQL Configuration
56 postgresql:
57   # PostgreSQL server listening address and port
58   listen: 10.0.20.110:5432
59   # Connect address for PostgreSQL clients
60   connect_address: 10.0.20.110:5432
61   # Data directory for PostgreSQL
62   data_dir: /var/lib/patroni
63   # Path to the pgpass file
64   pgpass: /tmp/pgpass
65   # Authentication configuration
66   authentication:
67     # Replication of user credentials
68     replication:
69       username: replicator
70       password: replicator
71     # Superuser credentials
72     superuser:
73       username: postgres
74       password: postgres
75   # Additional PostgreSQL parameters
76   parameters:
77
78     # Directory for Unix socket
79     unix_socket_directories: '.'
80     # Password encryption method
81     password_encryption: 'scram-sha-256'
82 # Patroni Tags Configuration
83 tags:
84   # Prevents a node from being promoted in case of failure
85   nofailover: false
86   # Prevents the load balancer from considering this node
87   noloadbalance: false
88   # Prevents a replica from being created by cloning
89   clonefrom: false
90   # Prevents synchronous replication from being enforced

```

```
91 | nosync: false
```

Listing 54: Patroni - Konfiguration - sks1232

```
1 # Scope of PostgreSQL
2 scope: postgres
3 # Namespace for the PostgreSQL database
4 namespace: /db/
5 # Name of the PostgreSQL instance
6 name: postgres02
7 # Patroni REST API Configuration
8 restapi:
9     # The IP address and port on which the REST API should listen
10    listen: 10.0.20.111:8008
11
12    # The IP address and port to which clients should connect
13    connect_address: 10.0.20.111:8008
14 # Patroni Etcd Configuration
15 etcd3:
16    # The host address and port of the Etcd server
17    host: 10.0.28.16:2379
18 # Patroni Bootstrap Configuration
19 bootstrap:
20    # Configuration parameters for distributed configuration store (DCS)
21    dcs:
22        ttl: 30
23        loop_wait: 10
24        retry_timeout: 10
25        maximum_lag_on_failover: 1048576
26        postgresql:
27            # Use pg_rewind during bootstrap
28            use_pg_rewind: true
29            # Change pg_wal
30            create_replica_methods:
31                - basebackup
32            basebackup:
33                max-rate: '100M'
34                waldir: "/srv/data/pg_wal"
35        # Initdb configuration
36        # Initdb configuration
37        initdb:
38            - auth: scram-sha-256
39            - encoding: UTF8
40            - data-checksums
41        # pg_hba.conf entries for replication and general access
42        pg_hba:
43            - host replication replicator 127.0.0.1/32 scram-sha-256
44            - host replication replicator 10.0.20.110/0 scram-sha-256
45            - host replication replicator 10.0.20.111/0 scram-sha-256
46            - host replication replicator 10.0.20.112/0 scram-sha-256
47            - host all all 0.0.0.0/0 scram-sha-256
48        # Adding default user admin with password admin
49        users:
50            admin:
```

```

51     password: admin
52     options:
53         - createrole
54         - createdb
55 # Patroni PostgreSQL Configuration
56 postgresql:
57     # PostgreSQL server listening address and port
58     listen: 10.0.20.111:5432
59     # Connect address for PostgreSQL clients
60     connect_address: 10.0.20.111:5432
61     # Data directory for PostgreSQL
62     data_dir: /var/lib/patroni
63     # Path to the pgpass file
64     pgpass: /tmp/pgpass
65
66     # Authentication configuration
67 authentication:
68     # Replication of user credentials
69     replication:
70         username: replicator
71         password: replicator
72     # Superuser credentials
73     superuser:
74         username: postgres
75         password: postgres
76     # Additional PostgreSQL parameters
77     parameters:
78         # Directory for Unix socket
79         unix_socket_directories: '.'
80         # Password encryption method
81         password_encryption: 'scram-sha-256'
82 # Patroni Tags Configuration
83 tags:
84     # Prevents a node from being promoted in case of failure
85     nofailover: false
86     # Prevents the load balancer from considering this node
87     noloadbalance: false
88     # Prevents a replica from being created by cloning
89     clonefrom: false
90     # Prevents synchronous replication from being enforced
91     nosync: false

```

Listing 55: Patroni - Konfiguration - sks1233

```

1 # Scope of PostgreSQL
2 scope: postgres
3 # Namespace for the PostgreSQL database
4 namespace: /db/
5 # Name of the PostgreSQL instance
6 name: postgres03
7 # Patroni REST API Configuration
8 restapi:
9     # The IP address and port on which the REST API should listen
10    listen: 10.0.20.112:8008

```

```

11
12     # The IP address and port to which clients should connect
13     connect_address: 10.0.20.112:8008
14 # Patroni Etcd Configuration
15 etcd3:
16     # The host address and port of the Etcd server
17     host: 10.0.28.16:2379
18 # Patroni Bootstrap Configuration
19 bootstrap:
20     # Configuration parameters for distributed configuration store (DCS)
21     dcs:
22         ttl: 30
23         loop_wait: 10
24         retry_timeout: 10
25         maximum_lag_on_failover: 1048576
26     postgresql:
27         # Use pg_rewind during bootstrap
28         use_pg_rewind: true
29         # Change pg_wal
30         create_replica_methods:
31             - basebackup
32         basebackup:
33             max-rate: '100M'
34             waldir: "/srv/data/pg_wal"
35 # Initdb configuration
36 initdb:
37     - auth: scram-sha-256
38     - encoding: UTF8
39     - data-checksums
40 # pg_hba.conf entries for replication and general access
41 pg_hba:
42     - host replication replicator 127.0.0.1/32 scram-sha-256
43     - host replication replicator 10.0.20.110/0 scram-sha-256
44     - host replication replicator 10.0.20.111/0 scram-sha-256
45     - host replication replicator 10.0.20.112/0 scram-sha-256
46     - host all all 0.0.0.0/0 scram-sha-256
47 # Adding default user admin with password admin
48 users:
49     admin:
50         password: admin
51         options:
52             - createrole
53             - createdb
54 # Patroni PostgreSQL Configuration
55 postgresql:
56     # PostgreSQL server listening address and port
57     listen: 10.0.20.112:5432
58     # Connect address for PostgreSQL clients
59     connect_address: 10.0.20.112:5432
60     # Data directory for PostgreSQL
61     data_dir: /var/lib/patroni
62     # Path to the pgpass file
63     pgpass: /tmp/pgpass
64     # Authentication configuration

```

```

65 authentication:
66   # Replication of user credentials
67   replication:
68     username: replicator
69     password: replicator
70   # Superuser credentials
71   superuser:
72     username: postgres
73     password: postgres
74   # Additional PostgreSQL parameters
75   parameters:
76     unix_socket_directories: '.'
77   # Password encryption method
78   password_encryption: 'scram-sha-256'
79 # Patroni Tags Configuration
80 tags:
81   # Prevents a node from being promoted in case of failure
82   nofailover: false
83   # Prevents the load balancer from considering this node
84   noloadbalance: false
85   # Prevents a replica from being created by cloning
86   clonefrom: false
87   # Prevents synchronous replication from being enforced
88   nosync: false

```

Listing 56: Patroni - Konfiguration - sks1234

Sehr wichtig ist, dass das pg_hba.conf-File mitgegeben wird.

Ansonsten kann nicht auf die Datenbank zugegriffen werden.

Für den User replication müssen die IP-Adressen aller Nodes gesetzt werden.

Im Evaluations-Enviroment soll zudem der Zugriff von überall mit Passwort möglich sein.

Dies wird mit der letzten Zeile ermöglicht:

```

1 host replication replicator 127.0.0.1/32 scram-sha-256
2 host replication replicator 10.0.20.110/0 scram-sha-256
3 host replication replicator 10.0.20.111/0 scram-sha-256
4 host replication replicator 10.0.20.112/0 scram-sha-256
5 host all all 0.0.0.0/0 scram-sha-256

```

Listing 57: Patroni Bootstrap - pg_hba

Nun muss das Verzeichnis für PostgreSQL im Patroni-Verzeichnis erstellt und berechtigt werden:

```

1 mkdir -p /var/lib/patroni
2 chown -R postgres:postgres /var/lib/patroni
3 chmod 700 /var/lib/patroni

```

Listing 58: Patroni Bootstrap - Patroni-Verzeichnis

Jetzt können die Dienste auf den Patroni-Servern neu gestartet werden, Patroni sollte nun lauffähig sein:

```

1 systemctl start patroni
2 systemctl status patroni
3 patronictl -c /etc/patroni/config.yml list

```

Listing 59: Patroni Bootstrap - Neu starten

Wenn es lauffähig ist, muss noch aufgeräumt werden.

Der bestehende PostgreSQL Cluster muss disabled werden:

```
1 sudo systemctl disable --now postgresql
```

Listing 60: Patroni Bootstrap - Disable PostgreSQL

VII.IV.IV Haproxy

Final kann nun HAProxy installiert werden.

Zuerst müssen die Hosts hinterlegt werden:

```
1 #nano /etc/hosts
2 10.0.20.110    postgres01
3 10.0.20.111    postgres02
4 10.0.20.112    postgres03
```

Listing 61: HAProxy - Hostliste

Nun müssen die Repositories aktualisiert werden und HAProxy installiert werden:

```
1 apt update
2 apt install haproxy
```

Listing 62: HAProxy - Installation

Mit der Installation kommt ein Konfig-File mit.

Das soll gesichert werden:

```
1 mv /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.orig
```

Listing 63: HAProxy - Safe Alte Config

Nun muss die neue Konfiguration geschrieben werden:

```
1 #nano /etc/haproxy/haproxy.cfg
2 # Global configuration settings
3 global
4   # Maximum connections globally
5   maxconn 100
6   # Logging settings
7   log 127.0.0.1 local2
8
9 # Default settings
10 defaults
11   # Global log configuration
12   log global
13   # Set mode to TCP
14   mode tcp
15   # Number of retries
16   retries 2
17   # Client timeout
18   timeout client 30m
19   # Connect timeout
20   timeout connect 4s
21   # Server timeout
22   timeout server 30m
```

```

23 # Check timeout
24 timeout check 5s
25
26 # Stats configuration listen stats
27   # Set mode to HTTP
28 mode http
29 # Bind to port 8080
30 bind *:8080
31 # Enable stats
32 stats enable
33 # Stats URI
34 stats uri /
35
36 # PostgreSQL configuration
37 listen postgres
38   # Bind to port 5432
39 bind *:5432
40 # Enable HTTP check
41 option httpchk
42 # Expect status 200
43 http-check expect status 200
44 # Server settings
45 default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
46 # Define PostgreSQL servers
47 server postgres01 10.0.20.110:5432 maxconn 100 check port 8008
48 server postgres02 10.0.20.111:5432 maxconn 100 check port 8008
49 server postgres03 10.0.20.112:5432 maxconn 100 check port 8008

```

Listing 64: HAProxy - Konfiguration

Nun kann HAProxy neu gestartet werden:

```

1 systemctl restart haproxy
2 systemctl status haproxy

```

Listing 65: HAProxy - Restart

Nun kann HAProxy geöffnet werden: <http://10.0.28.16:8080/>

Es sollte nun so aussehen:

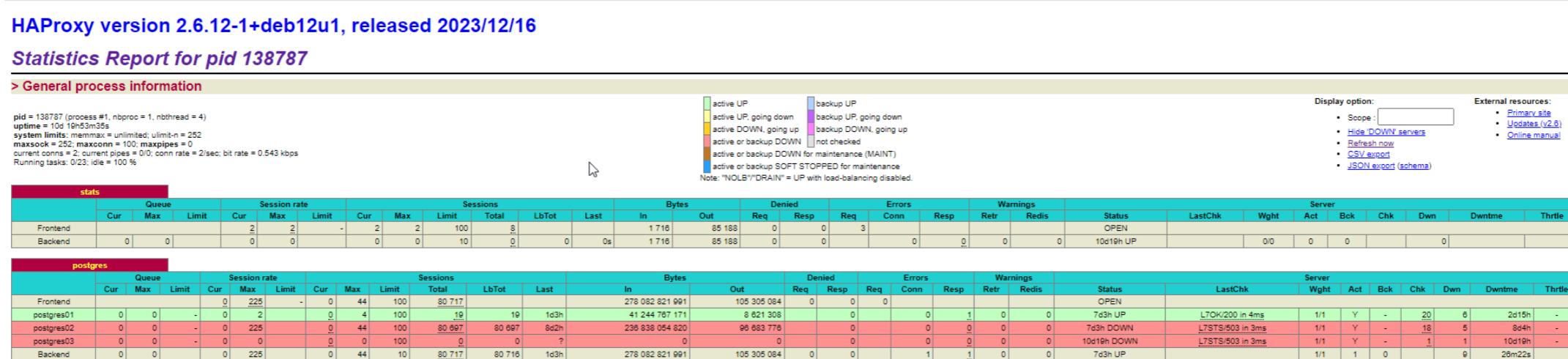


Abbildung XXXIX: HAProxy - Web-GUI

VII.IV.V Rekonfiguration mit 250GiB Storage

Sobald die neue Disk im VMware vSphere erfasst wurde, muss die Disk auf den Servern gemountet werden:

```
1 echo " - - - " | tee /sys/class/scsi_host/host*/scan && lsblk
2
3 fdisk /dev/sdb
4   n -> all default values
5   t -> 8e
6   p -> Kontrolle
7   w
8
9 pvcreate /dev/sdb1 && \
10 vgcreate vgdata /dev/sdb1 && \
11 lvcreate -l 100%FREE -n lvdata vgdata && \
12 mkfs.ext4 /dev/vgdata/lvdata && \
13 mkdir -p /srv/data && \
14 cp /etc/fstab /tmp/fstab.bak && \
15 echo "/dev/vgdata/lvdata /srv/data ext4 defaults 0 0" >> /etc/fstab && \
16 systemctl daemon-reload && mount -a && lsblk
```

Listing 66: Patroni - 250GiB Disk mount

Nun können die Verzeichnisse angelegt werden.

Nebst dem Verzeichnis für die Indizes und Daten braucht es auch ein neues pg_wal-Verzeichnis:

```
1 mkdir -p /srv/data/eval_index_tablespace
2 chown -R postgres:postgres /srv/data/eval_index_tablespace
3 chmod 700 /srv/data/eval_index_tablespace
4
5 mkdir -p /srv/data/eval_data_tablespace
6 chown -R postgres:postgres /srv/data/eval_data_tablespace
7 chmod 700 /srv/data/eval_data_tablespace
8
9 mkdir -p /srv/data/self_healing_test_index_tablespace
10 chown -R postgres:postgres /srv/data/self_healing_test_index_tablespace
11 chmod 700 /srv/data/self_healing_test_index_tablespace
12
13 mkdir -p /srv/data/self_healing_test_data_tablespace
14 chown -R postgres:postgres /srv/data/self_healing_test_data_tablespace
15 chmod 700 /srv/data/self_healing_test_data_tablespace
16
17 mkdir -p /srv/data/pg_wal
18 chown -R postgres:postgres /srv/data/pg_wal
19 chmod 700 /srv/data/pg_wal
```

Listing 67: Patroni - 250GiB Verzeichnisse

Um das WAL-Verzeichnis umzuhängen, muss erst die der Patroni-Cluster gestoppt werden:

```
1 patronictl -c /etc/patroni/config.yml pause
```

Listing 68: Patroni - 250GiB Cluster Pause

Wichtig ist, dass auch PostgreSQL gestoppt ist.

Da die PostgreSQL Binaries nun im Patroni-Verzeichnis liegen und auch dort das Socket-File läuft, muss das Verzeichnis zwingend als Daten-Verzeichnisparameter (-D) angegeben werden,

sonst wird es zu einem Socket-Fehler kommen.

Zwingend ist zudem, das Command als postgres-User auszuführen:

```
1 sudo su - postgres
2 /usr/sbin/pg_ctl stop -D /var/lib/patroni/
```

Listing 69: Patroni - 250GiB PostgreSQL stoppen

Ganz wichtig ist nun, das ganze pg_wal-Verzeichnis zu verschieben.

So das die WAL-Files kopiert werden.

Da ein symlink gezogen wird, muss das bestehende Verzeichnis garantiert gelöscht sein:

```
1 mv /var/lib/patroni/pg_wal/* /srv/data/pg_wal
2 ln -s /srv/data/pg_wal /var/lib/patroni/pg_wal
3 ls -l /var/lib/patroni/pg_wal
```

Listing 70: Patroni - 250GiB move pg_wal

Die Berechtigung muss anschliessend wieder mit root-Rechten gesetzt werden:

```
1 chown -R postgres:postgres /var/lib/patroni/pg_wal
2 chmod 700 /var/lib/patroni/pg_wal
```

Listing 71: Patroni - 250GiB chmod - chown pg_wal

Jetzt muss erst PostgreSQL wieder gestartet werden.

Wenn alle Datenbanken auf allen Patroni-Nodes wieder lauffähig sind, kann der Cluster wieder aktiviert werden:

```
1 /usr/sbin/pg_ctl start -D /var/lib/patroni/
2 patronictl -c /etc/patroni/config.yml resume
```

Listing 72: Patroni - 250GiB PostgreSQL - Patroni resume

Zum Schluss sollte noch zwingend geprüft werden, ob der Cluster funktionsfähig ist:

```
1 patronictl -c /etc/patroni/config.yml list
```

Listing 73: Patroni - 250GiB Finaler Check

Wenn Patroni nun wieder einsatzfähig ist, müssen noch die optimierten Parameter gesetzt werden.

Folgende Parameter wurden gesetzt:

max_connections

Damit die 7000 Transaktionen sicher abgesetzt werden können.

superuser_reserved_connections

Mindestens 10 postgres-Verbindungen müssen gemacht werden können.

max_worker_processes

Damit genügend Worker vorhanden sind, um die WAL-Files abzuarbeiten, wurde deren Zahl auf 16 erhöht.

wal_log_hints

Stellt sicher, dass der gesamte Diskpage-Inhalt ins WAL geschrieben wird.

max_wal_senders

Um den Replikations-Throughput zu erhöhen, wurden 16 Sender aktiviert

max_replication_slots

Damit die 16 Sender auch auf der Gegenseite abgearbeitet werden, müssen die entsprechenden Slots bereitgestellt werden.

wal_keep_size

Damit die WAL-Files nicht zu gross werden und somit auch das Replication-Lag, wurde die Grösse auf 1 GiB reduziert.

Das zwingt den Primary dazu, die WAL-Files nur bis zu maximal 1 GiB aufzubewahren, bevor sie archiviert werden.

Das ist wichtig, wenn Diskspace nicht im Überfluss vorhanden ist.

wal_level

Der Level wurde auf logical gesetzt, damit alle Informationen geschrieben werden.

wal_buffers

Der Buffer wurde auf 16 MiB erhöht.

wal_writer_delay

Wenn der WAL-Writer einen Flush durchgeführt hat, geht er in ein Timeout.

Dies wurde auf 20 ms heruntergesetzt, damit rasch wieder geschrieben wird.

wal_writer_flush_after

Definiert, ab welcher Grösse der Cache des WAL-Writers auf die Disk geschrieben wird.

Wurde auf 1 MiB gesetzt um rasch auf die Disk schreiben zu können.

min_wal_size

Damit die WAL-Files nicht zu schnell repliziert werden und die Standby-Server, aber auch der Primary-Server nicht überlastet werden, sollen die Files mindestens 1 GiB gross werden, bevor sie repliziert werden.

max_wal_size

Auf der anderen Seite soll trotzdem nicht zu lange gewartet werden, die maximale Grösse wurde auf 5 GiB begrenzt.

commit_delay

Maximal 20 ms soll gewartet werden, bevor ein Transactions-Commit geflushed wird.

commit_siblings

Maximal 10 Transktionen sollen offenbleiben, bevor es zu einem Flush kommt.

checkpoint_timeout

Die maximale Zeit zwischen den Checkpoints soll 5min betragen.

archive_mode

Um Diskspace zu sparen sollen die WAL-Files nicht aktiviert werden.

checkpoint_completion_target

Ab 95% des Checkout-Timeouts soll mit dem Abschluss begonnen werden.

max_standby_archive_delay

Erst nach zehn Minuten sollen die Standby-Queries abgebrochen werden.

Ist notwendig, da es wegen der grossen Datenmenge und entsprechendem lag sonst schnell zum Abbruch kommen kann.

max_standby_streaming_delay

Beim normalen Standby soll die Zeit nur 3 min sein.

wal_receiver_status_interval

Alle 60 sekunden soll der Status ausgetauscht werden.

max_logical_replication_workers

Acht logische Replication worker sollen zum Einsatz kommen.

max_sync_workers_per_subscription

Es soll die maximale Anzahl an workern (8) für das Parallelisieren verwendet werden.

shared_buffers

Der gesamte Server hat 16 GiB Memory, $\frac{1}{4}$ davon soll als Buffer dienen, also 4 GiB.

maintenance_work_mem

1 GiB soll zur Verfügung stehen für den AUTOVACUUM-Job oder Maintenance-Jobs wie das Indexieren usw.

work_mem

Der ganze PostgreSQL Cluster soll 12 GiB Memory zur Verfügung haben.

temp_file_limit

Damit für das Erzeugen von Primary Keys, Foreign-Keys oder Indizes genügend Platz vorhanden ist, sollen temporäre Tabellen 200 GiB Diskspace allokiert werden.

vacuum_cost_delay

Der AUTOVACUUM-Job soll maximal 2 ms ruhen.

vacuum_cost_limit

Der gesamte Prozess darf maximal zehn Sekunden schlafen.

bgwriter_delay

Der Hintergrundprozess soll nur 10 ms ruhen.

bgwriter_lru_maxpages

Maximal dürfen 800 Buffers vom Hintergrundprozess beschrieben werden.

bgwriter_lru_multiplier

Insgesamt sollen beim nächsten Schreibzyklus 5 x soviel neue Writer erzeugt werden, wie im aktuellen Zyklus benötigt werden.

Entsprechend auch hier der Command mit patronictl:

```
1 patronictl -c /etc/patroni/config.yml edit-config --apply --force << 'JSON'
2 {
3     synchronous_mode: "on",
4     synchronous_mode_strict: "on",
5     synchronous_node_count: 2,
6     "postgresql":
7         {
8             "parameters": {
9                 "synchronous_commit": "on",
10                "synchronous_standby_names": "*",
11            }
12        }
13    }
14 }
```

```

11 "max_connections": 1100,
12 "superuser_reserved_connections":10,
13 "max_worker_processes": 16,
14 "wal_log_hints": "on",      "max_wal_senders": 32,
15 "max_replication_slots": 32,
16 "wal_keep_size": "1GB",
17 "wal_level": "logical",
18 "wal_buffers": "16MB",
19 "wal_writer_delay": "20ms",
20 "wal_writer_flush_after": "1MB",
21 "min_wal_size": "1GB",
22 "max_wal_size": "5GB",
23 "commit_delay": 20,
24 "commit_siblings": 10,
25 "checkpoint_timeout": "5min",
26 "checkpoint_completion_target": 0.95,
27 "archive_mode": "off",
28 "max_standby_archive_delay": "10min",
29 "max_standby_streaming_delay": "3min",
30 "wal_receiver_status_interval": "1s",
31 "hot_standby_feedback": "on",
32 "wal_receiver_timeout": "60s",
33 "max_logical_replication_workers": 8,
34 "max_sync_workers_per_subscription": 8,
35 "shared_buffers": "4GB",
36 "maintenance_work_mem": "1GB",
37 "work_mem": "12GB",
38 "temp_file_limit": "200GB",
39 "vacuum_cost_delay": "2ms",
40 "vacuum_cost_limit": 10000,
41 "bgwriter_delay": "10ms",
42 "bgwriter_lru_maxpages": 800,
43 "bgwriter_lru_multiplier": "5.0"
44 }
45 }
46 }
47 JSON

```

Listing 74: Patroni - 250GiB set Parameter

VII.IV.VI SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle pgbench_eval_bench erstellt.

Zudem wird je ein Tablespace für die Indizes (eval_index_tablespace) und Daten (eval_data_tablespace) erstellt.

Dies muss aber zweimal gemacht werden, einmal für die normalen Benchmarks und einmal mit den grossen Volumes:

```

1 -- Datenbank pgbench_eval_bench erstellen
2 drop database if exists pgbench_eval_bench;
3 create database pgbench_eval_bench;
4
5 -- Tablespace für Indices
6 drop tablespace if exists eval_index_tablespace;
7 CREATE TABLESPACE eval_index_tablespace owner postgres location '/var/lib/patroni/eval_index_tablespace';

```

```

8
9 -- Genereller Tablespace erstellen
10 drop tablespace if exists eval_data_tablespace;CREATE TABLESPACE eval_data_tablespace owner postgres location '/var/lib/patroni/eval_data_tablespace';

```

Listing 75: Patroni - Benchmarking - DB erstellen

Wird auf die grossen Volumes gewechselt, müssen vorher die Tabellen bereinigt und gelöscht werden:

```

1 -- Daten löschen
2 truncate table pgbench_accounts;
3 truncate table pgbench_tellers;
4 truncate table pgbench_history;
5 truncate table pgbench_branches;
6
7 -- Tabellen löschen
8 drop table pgbench_accounts;
9 drop table pgbench_tellers;
10 drop table pgbench_history;
11 drop table pgbench_branches;

```

Listing 76: Patroni - Benchmarking - DB Cleanup

Anschliessend werden die neuen Tablespaces erzeugt:

```

1 -- Tablespace für Indices
2 drop tablespace if exists eval_index_tablespace;
3 CREATE TABLESPACE eval_index_tablespace owner postgres location '/srv/data/eval_index_tablespace';
4
5 -- Genereller Tablespace erstellen
6 drop tablespace if exists eval_data_tablespace;
7 CREATE TABLESPACE eval_data_tablespace owner postgres location '/srv/data/eval_data_tablespace';

```

Listing 77: Patroni - Benchmarking - Tablespaces erneut erstellen

VII.IV.VII SQL Statements - Testing

Entsprechend dem ERD müssen die Tabellen erstellt werden: [Evaluation - ERD self_healing_test](#) Die Tabelle heisst entsprechend self_healing_test. Die Tests wurden allerdings erst gemacht, als auf die grossen Volumes gewechselt war. Das gesamte CREATE-Skript:

```

1 -- self-healing-Tabelle
2 drop table if exists self_healing_test;
3 create database self_healing_test;
4
5 -- Tablespace für Indices
6 drop tablespace if exists self_healing_indices_tablespace;
7 CREATE TABLESPACE self_healing_indices_tablespace owner postgres location '/srv/data/self_healing_test_index_tablespace';
8
9 -- Genereller Tablespace erstellen
10 drop tablespace if exists self_healing_datas_tablespace;
11 CREATE TABLESPACE self_healing_datas_tablespace owner postgres location '/srv/data/self_healing_test_data_tablespace';
12
13 -- Rollen erstellen
14 drop role if exists hrm;
15 create role hrm;
16 drop role if exists accountands;
17 create role accountands;

```

```

18 drop role if exists customer_serviceOfficers;
19 create role customer_serviceOfficers;
20 drop role if exists legal_affairs;create role legal_affairs;
21
22 -- User erstellen
23 drop user if exists hrm_1;
24 drop user if exists hrm_2;
25 create user hrm_1 with password 'hrm1' role hrm;
26 create user hrm_2 with password 'hrm2' role hrm;
27
28 drop user if exists cso_1;
29 drop user if exists cso_2;
30 create user cso_1 with password 'cso1'role customer_serviceOfficers;
31 create user cso_2 with password 'cso2' role customer_serviceOfficers;
32
33 drop user if exists la_1;
34 drop user if exists la_2;
35 create user la_1 with password 'la1' role legal_affairs;
36 create user la_2 with password 'la2' role legal_affairs;
37
38 -- Schemas erstellen
39 drop schema if exists hrm;
40 create schema hrm authorization hrm;
41 drop schema if exists accountands;
42 create schema accountands authorization accountands;
43 drop schema if exists customer_serviceOfficers;
44 create schema customer_serviceOfficers authorization customer_serviceOfficers;
45 drop schema if exists generell;
46 create schema generell;
47
48 -- GRANTS erstellen
49 grant all on all tables in schema hrm to legal_affairs;
50 grant all on all tables in schema accountands to legal_affairs;
51 grant all on all tables in schema customer_serviceOfficers to legal_affairs;
52 grant all on all tables in schema generell to legal_affairs;
53 grant all on all tables in schema hrm to postgres;
54 grant all on all tables in schema accountands to postgres;
55 grant all on all tables in schema customer_serviceOfficers to postgres;
56 grant all on all tables in schema generell to postgres;
57
58 -- self_healing_accounts für Schema customer_serviceOfficers
59 drop table if exists customer_serviceOfficers.self_healing_accounts;
60 create table customer_serviceOfficers.self_healing_accounts (
61     account_id int primary key,
62     firstname varchar(255) not null,
63     lastname varchar(255) not null,
64     birthday date not null,
65     postal_code varchar(50),
66     street varchar(255),
67     country_code varchar(2),
68     phone varchar(25),
69     mail varchar(255) check (mail like '%@%')
70 ) tablespace self_healing_datas_tablespace;
71 create unique index accounts_personal_mark on customer_serviceOfficers.self_healing_accounts(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;

```

```

72
73 -- self_healing_employees für Schema hrm
74 drop table if exists hrm.self_healing_employees;
75 create table hrm.self_healing_employees (
76     employees_id int primary key,
77     firstname varchar(255) not null,
78     lastname varchar(255) not null,
79     birthday date not null,
80     postal_code varchar(50),
81     street varchar(255),
82     country_code varchar(2),
83     phone varchar(25),
84     mail varchar(255) check (mail like '%@%')
85 ) tablespace self_healing_datas_tablespace;
86
87 -- self_healing_accountand_protocol für Schema accountands
88 drop table if exists accountands.self_healing_accountand_protocol;
89 create table accountands.self_healing_accountand_protocol (
90     acc_protocol_id int primary key,
91     description varchar(100) not null,
92     protocol_date date not null,
93     employees_id int not null,
94     rapport TEXT,
95     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
96 ) tablespace self_healing_datas_tablespace;
97
98 -- self_healing_intranet für public Schema
99 drop table if exists generell.self_healing_intranet;
100 create table generell.self_healing_intranet (
101     intranet_id int primary key,
102     content text
103 ) tablespace self_healing_datas_tablespace;
104
105 -- self_healing_intranet für public Schema
106 drop table if exists generell.self_healing_intranet_users;
107 create table generell.self_healing_intranet_users (
108     intranet_user_id int primary key,
109     employees_id int not null,
110     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
111 ) tablespace self_healing_datas_tablespace;
112 create unique index intranet_unique_combi on generell.self_healing_intranet_users(intranet_user_id, employees_id) tablespace self_healing_indices_tablespace;

```

Listing 78: Patroni - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```

1 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
2 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
3 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
4
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
6 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
7 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
8
9 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (100, 'bla', '07.04.2024', 100, 'blabla');

```

```

10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (200, 'yada', '07.04.2024', 100, 'ydayadyada');
11 );
12 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (300, 'something', '07.04.2024', 300, 'something');
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
15 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');
16
17 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(100, 100);
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(200, 200);
19 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(300, 300);
20
21 select * from customer_service_officers.self_healing_accounts;
22 select * from hrm.self_healing_employees;
23 select * from accountands.self_healing_accountand_protocol;
24 select * from generell.self_healing_intranet_users;

```

Listing 79: Patroni - Self Healing Tests - Init Data

Während des Failover-Test müssen Daten beschrieben werden:

```

1 insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
2 insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
3 insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
4
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
6 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
7 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
8
9 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (400, 'bla', '07.04.2024', 200, 'blabla');
10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (500, 'yada', '07.04.2024', 600, 'ydayadyada');
11 );
12 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (1000, 'something', '07.04.2024', 300, 'something');
13
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (200, 'yadada');
15 insert into generell.self_healing_intranet(intranet_id, content) VALUES (600, 'bla bla');
16 insert into generell.self_healing_intranet(intranet_id, content) VALUES (900, 'talking and talking');
17
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(400, 400);
19 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(500, 500);
20 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(600, 600);
21
22 select * from customer_service_officers.self_healing_accounts;
23 select * from hrm.self_healing_employees;
24 select * from accountands.self_healing_accountand_protocol;
25 select * from generell.self_healing_intranet_users;

```

Listing 80: Patroni - Self Healing Tests - Failover Data

Nach dem Recovery müssen die Daten entsprechend vorhanden sein und es müssen weitere Daten beschrieben werden können:

```

1 select * from customer_service_officers.self_healing_accounts;
2 select * from hrm.self_healing_employees;
3 select * from accountands.self_healing_accountand_protocol;

```

```

4 select * from generell.self_healing_intranet;
5 select * from generell.self_healing_intranet_users;
6 insert into generell.self_healing_intranet(intranet_id, content) VALUES (700, 'yadada');insert into generell.self_healing_intranet(intranet_id, content) VALUES (800, 'bla bla');
7 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1100, 'talking and talking');
8
9 select * from customer_serviceOfficers.self_healing_accounts;
10 select * from hrm.self_healing_employees;
11 select * from accountands.self_healing_accountand_protocol;
12 select * from generell.self_healing_intranet;
13 select * from generell.self_healing_intranet_users;

```

Listing 81: Patroni - Self Healing Tests - Recovery Data

VII.V Stackgres mit Citus

VII.V.I Prerequisites

VII.V.I.I StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```

1 # https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/prepare-environment/kubernetes/#configure-storage-class
2 # https://github.com/rancher/local-path-provisioner
3 apiVersion: storage.k8s.io/v1
4 kind: StorageClass
5 metadata:
6   name: stackgres-storage
7 provisioner: rancher.io/local-path
8 parameters:
9   nodePath: /srv/data/local-path-provisioner
10 volumeBindingMode: WaitForFirstConsumer
11 reclaimPolicy: Delete
12 --
13 apiVersion: v1
14 kind: PersistentVolume
15 metadata:
16   name: stackgres-storage-pv
17   labels:
18     type: local
19 spec:
20   accessModes:
21     - ReadWriteOnce
22   capacity:
23     storage: 1Gi
24   storageClassName: "stackgres-storage"
25   hostPath:
26     path: /srv/data/local-path-provisioner
27   nodeAffinity:
28     required:
29       nodeSelectorTerms:
30         - matchExpressions:
31           - key: kubernetes.io/hostname
32             operator: In
33             values:

```

```
34     - sks1183
35     - sks1184
36     - sks1185
```

Listing 82: StackGres-Citus - StorageClass setzen

Die StorageClass und das PersistentVolume muss aktiviert werden:

```
1 gramic@cks4040:~$ kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/storageclass.yaml
2 storageclass.storage.k8s.io/stackgres-storage created
3 persistentvolume/stackgres-storage-pv created
```

Listing 83: StackGres-Citus - StorageClass / PersistentVolume aktivieren

VII.V.II Installation

Auch bei StackGres braucht es zuerst einen entsprechenden Namespace:

```
1 kubectl create namespace sg-platform
```

Listing 84: StackGres-Citus - Namespace

Das Manifest beinhaltet nur die wichtigsten Parameter.

Der Grossteil wird über das Cluster-Deployment gesetzt:

```
1 # -- The container registry host (and port) where the images will be pulled from.
2 containerRegistry: quay.io
3 # -- Image pull policy used for images loaded by the Operator
4 imagePullPolicy: "IfNotPresent"
5 # Section to configure Operator Installation ServiceAccount
6 serviceAccount:
7   # -- If 'true' the Operator Installation ServiceAccount will be created
8   create: true
9   # -- Section to configure Operator ServiceAccount annotations
10  annotations: {}
11  # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
12  repoCredentials: []
13
14 # Section to configure Operator Pod
15 operator:
16   # Section to configure Operator image
17   image:
18     # -- Operator image name
19     name: "stackgres/operator"
20     # -- Operator image tag
21     tag: "1.9.0"
22     # -- Operator image pull policy
23     pullPolicy: "IfNotPresent"
24     # -- Operator Pod annotations
25     annotations: {}
26     # -- Operator Pod resources. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#resourcerequirements-v1-core
27   # resources: {}
28   resources:
29     requests:
30       cpu: "1"
31       memory: 1Gi
```

```

32   limits:
33     cpu: "1"
34     memory: 1Gi
35   # -- Operator Pod node selector
36   nodeSelector: {}
37   # -- Operator Pod tolerations. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#toleration-v1-core
38   tolerations: []
39   # -- Operator Pod affinity. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#affinity-v1-core
40   affinity: {}
41   # Section to configure Operator ServiceAccount
42   serviceAccount:
43     # -- Section to configure Operator ServiceAccount annotations
44     annotations: {}
45     # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
46     repoCredentials: []
47   # Section to configure Operator Service
48   service:
49     # -- Section to configure Operator Service annotations
50     annotations: {}
51
52 # Section to configure REST API Pod
53 restapi:
54   # -- REST API Pod name
55   name: stackgres-restapi
56   # Section to configure REST API image
57   image:
58     # -- REST API image name
59     name: "stackgres/restapi"
60     # -- REST API image tag
61     tag: "1.9.0"
62     # -- REST API image pull policy
63     pullPolicy: "IfNotPresent"
64   # -- REST API Pod annotations
65   annotations: {}
66   resources:
67     requests:
68       cpu: "1"
69       memory: 1Gi
70     limits:
71       cpu: "1"
72       memory: 1Gi
73   # -- REST API Pod node selector
74   nodeSelector: {}
75   # -- REST API Pod tolerations. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#toleration-v1-core
76   tolerations: []
77   # -- REST API Pod affinity. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#affinity-v1-core
78   affinity: {}
79   # Section to configure REST API ServiceAccount
80   serviceAccount:
81     # -- REST API ServiceAccount annotations
82     annotations: {}
83     # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
84     repoCredentials: []
85   # Section to configure REST API Service

```

```

86 service:
87   # -- REST API Service annotations
88   annotations: {}
89
90 # Section to configure Web Console container
91 adminui:
92   # Section to configure Web Console image
93   image:
94     # -- Web Console image name
95     name: "stackgres/admin-ui"
96     # -- Web Console image tag
97     tag: "1.9.0"
98     # -- Web Console image pull policy
99     pullPolicy: "IfNotPresent"
100 resources:
101   requests:
102     cpu: "1"
103     memory: 1Gi
104   limits:
105     cpu: "1"
106     memory: 1Gi
107 # Section to configure Web Console service.
108 service:
109   # -- When set to 'true' the HTTP port will be exposed in the Web Console Service
110 exposeHTTP: true
111 # -- The type used for the service of the UI:
112 type: ClusterIP
113 # -- (string) LoadBalancer will get created with the IP specified in
114 loadBalancerIP:           # gramic, load-balancer-IP
115 # -- (array) If specified and supported by the platform,
116 loadBalancerSourceRanges:
117 # -- (integer) The HTTPS port used to expose the Service on Kubernetes nodes
118 nodePort:
119 # -- (integer) The HTTP port used to expose the Service on Kubernetes nodes
120 nodePortHTTP:
121
122 # Section to configure Operator Installation Jobs
123 jobs:
124   # Section to configure Operator Installation Jobs image
125   image:
126     # -- Operator Installation Jobs image name
127     name: "stackgres/jobs"
128     # -- Operator Installation Jobs image tag
129     tag: "1.9.0"
130     # -- Operator Installation Jobs image pull policy
131     pullPolicy: "IfNotPresent"
132     # -- Operator Installation Jobs annotations
133     annotations: {}
134     # -- Operator Installation Jobs resources. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#resourcerequirements-v1-core
135     resources: {}
136     # -- Operator Installation Jobs node selector
137     nodeSelector: {}
138     # -- Operator Installation Jobs tolerations. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#toleration-v1-core
139     tolerations: []

```

```

140 # -- Operator Installation Jobs affinity. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#affinity-v1-core
141 affinity: {}
142
143 # Section to configure deployment aspects.
144 deploy:
145   # -- When set to 'true' the Operator will be deployed.
146   operator: true
147   # -- When set to 'true' the Web Console / REST API will be deployed.
148   restapi: true
149
150 # Section to configure the Operator, REST API and Web Console certificates and JWT RSA key-pair.
151 cert:
152   # -- If set to 'true' the CertificateSigningRequest used to generate the certificate used by
153   #   Webhooks will be approved by the Operator Installation Job.
154   autoapprove: true
155   # -- When set to 'true' the Operator certificate will be created.
156   createForOperator: true
157   # -- When set to 'true' the Web Console / REST API certificate will be created.
158   createForWebApi: true
159   # -- (string) The Secret name with the Operator Webhooks certificate issued by the Kubernetes cluster CA
160   #   of type kubernetes.io/tls. See https://kubernetes.io/docs/concepts/configuration/secret/#tls-secrets
161   secretName:
162     # -- When set to 'true' the Operator certificates will be regenerated if 'createForOperator' is set to 'true', and the certificate is expired or invalid.
163   regenerateCert: true
164   # -- (integer) The duration in days of the generated certificate for the Operator after which it will expire and be regenerated.
165   #   If not specified it will be set to 730 (2 years) by default.
166   certDuration: 730
167   # -- (string) The Secret name with the Web Console / REST API certificate
168   #   of type kubernetes.io/tls. See https://kubernetes.io/docs/concepts/configuration/secret/#tls-secrets
169   webSecretName:
170     # -- When set to 'true' the Web Console / REST API certificates will be regenerated if 'createForWebApi' is set to 'true', and the certificate is expired or invalid.
171   regenerateWebCert: true
172   # -- When set to 'true' the Web Console / REST API RSA key pair will be regenerated if 'createForWebApi' is set to 'true', and the certificate is expired or invalid.
173   regenerateWebRsa: true
174   # -- (integer) The duration in days of the generated certificate for the Web Console / REST API after which it will expire and be regenerated.
175   #   If not specified it will be set to 730 (2 years) by default.
176   webCertDuration:
177     # -- (integer) The duration in days of the generated RSA key pair for the Web Console / REST API after which it will expire and be regenerated.
178   #   If not specified it will be set to 730 (2 years) by default.
179   webRsaDuration:
180     # -- (string) The private RSA key used to create the Operator Webhooks certificate issued by the
181     #   Kubernetes cluster CA.
182   key:
183     # -- (string) The Operator Webhooks certificate issued by Kubernetes cluster CA.
184   crt:
185     # -- (string) The private RSA key used to generate JWTs used in REST API authentication.
186   jwtRsaKey:
187     # -- (string) The public RSA key used to verify JWTs used in REST API authentication.
188   jwtRsaPub:
189     # -- (string) The private RSA key used to create the Web Console / REST API certificate
190   webKey:
191     # -- (string) The Web Console / REST API certificate
192   webCrt:
193     # Section to configure cert-manager integration to generate Operator certificates

```

```

194 certManager:
195   # -- When set to 'true' then Issuer and Certificate for Operator and Web Console / REST API
196   #   Pods will be generated
197   autoConfigure: false
198   # -- The requested duration (i.e. lifetime) of the Certificates. See https://cert-manager.io/docs/reference/api-docs/#cert-manager.io%2fv1
199   duration: "2160h"
200   # -- How long before the currently issued certificates expiry cert-manager should renew the certificate. See https://cert-manager.io/docs/reference/api-docs/#cert-manager.io%2fv1
201   renewBefore: "360h"
202   # -- The private key cryptography standards (PKCS) encoding for this certificates private key to be encoded in. See https://cert-manager.io/docs/reference/api-docs/#cert-
203   manager.io/v1.CertificatePrivateKey
204   encoding: PKCS1
205   # -- Size is the key bit size of the corresponding private key for this certificate. See https://cert-manager.io/docs/reference/api-docs/#cert-manager.io/v1.
206   CertificatePrivateKey
207   size: 2048
208
209 # Section to configure RBAC for Web Console admin user
210 rbac:
211   # -- When set to 'true' the admin user is assigned the 'cluster-admin' ClusterRole by creating
212   #   ClusterRoleBinding.
213   create: true
214
215 # Section to configure Web Console authentication
216 authentication:
217   # -- Specify the authentication mechanism to use. By default is 'jwt', see https://stackgres.io/doc/latest/api/rbac#local-secret-mechanism.
218   #   If set to 'oidc' then see https://stackgres.io/doc/latest/api/rbac/#openid-connect-provider-mechanism.
219   type: jwt
220   # -- (boolean) When 'true' will create the secret used to store the 'admin' user credentials to access the UI.
221   createAdminSecret: true
222   # -- The admin username that will be required to access the UI
223   user: admin
224   # -- (string) The admin password that will be required to access the UI
225   #password: "TES2&Daggerfall"
226   password:
227   # Section to configure Web Console OIDC authentication
228 oidc:
229     # tlsVerification -- (string) Can be one of 'required', 'certificate-validation' or 'none'
230
231 # Section to configure Prometheus integration.
232 prometheus:
233
234   allowAutobind: true
235
236 # Section to configure Grafana integration
237 grafana:
238   # -- When set to 'true' embed automatically Grafana into the Web Console by creating the
239   #   StackGres dashboards and the read-only role used to read it from the Web Console
240   autoEmbed: false
241   # -- The schema to access Grafana. By default http. (used to embed manually and
242   #   automatically grafana)
243   schema: http
244   # -- (string) The service host name to access grafana (used to embed manually and
245   #   automatically Grafana).
246   webHost:
247   # -- The datasource name used to create the StackGres Dashboards into Grafana
248   dataSourceName: Prometheus

```

```

245 # -- The username to access Grafana. By default admin. (used to embed automatically
246 # Grafana)
247 user: admin
248 # -- The password to access Grafana. By default prom-operator (the default in for
249 # kube-prometheus-stack helm chart). (used to embed automatically Grafana)
250 password: prom-operator
251 # -- Use following fields to indicate a secret where the grafana admin credentials are stored (replace user/password)
252
253 # -- (string) The namespace of secret with credentials to access Grafana. (used to
254 # embed automatically Grafana, alternative to use 'user' and 'password')
255 secretNamespace:
256 # -- (string) The name of secret with credentials to access Grafana. (used to embed
257 # automatically Grafana, alternative to use 'user' and 'password')
258 secretName:
259 # -- (string) The key of secret with username used to access Grafana. (used to embed
260 # automatically Grafana, alternative to use 'user' and 'password')
261 secretUserKey:
262 # -- (string) The key of secret with password used to access Grafana. (used to
263 # embed automatically Grafana, alternative to use 'user' and 'password')
264 secretPasswordKey:
265 # -- (string) The ConfigMap name with the dashboard JSONs
266 # that will be created in Grafana. If not set the default
267 # StackGres dashboards will be created. (used to embed automatically Grafana)
268 dashboardConfigMap:
269 # -- (array) The URLs of the PostgreSQL dashboards created in Grafana (used to embed manually
270 urls:
271 # Create and copy/paste grafana API token:
272 # - Grafana > Configuration > API Keys > Add API key (for viewer) > Copy key value
273 token:
274
275 # Section to configure extensions
276 extensions:
277 repositoryUrls:
278 - https://extensions.stackgres.io/postgres/repository?proxyUrl=http%3A%2F%2Fsproxy.ssvc.first-it.ch%3A8080?skipHostnameVerification:true&setHttpScheme:true
279 cache:
280 # -- When set to 'true' enable the extensions cache.
281 enabled: false
282 # -- An array of extensions pattern used to pre-loaded extensions into the extensions cache
283 preloadedExtensions:
284 - x86_64/linux/timescaledb-1\.7\.4-pg12
285 # Section to configure the extensions cache PersistentVolume
286 persistentVolume:
287 size: 60Gi
288 storageClass: "stackgres-storage"
289 hostPath:
290 developer:
291 version:
292 # -- (string) Set 'quarkus.log.level'. See https://quarkus.io/guides/logging#root-logger-configuration
293 logLevel:
294 # -- If set to 'true' add extra debug to any script controlled by the reconciliation cycle of the operator configuration
295 showDebug: false
296 # -- Set 'quarkus.log.console.format' to '%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c{4.}] (%t) %s%e%n'. See https://quarkus.io/guides/logging#logging-format
297 showStackTraces: false
298 # -- Only work with JVM version and allow connect

```

```

299 # on port 8000 of operator Pod with jdb or similar
300 enableJvmDebug: false
301 # -- Only work with JVM version and if 'enableJvmDebug' is 'true'
302 #   suspend the JVM until a debugger session is started
303 enableJvmDebugSuspend: false
304 # -- (string) Set the external Operator IP
305 externalOperatorIp:
306 # -- (integer) Set the external Operator port
307 externalOperatorPort:
308 # -- (string) Set the external REST API IP
309 externalRestApiIp:
310 # -- (integer) Set the external REST API port
311 externalRestApiPort:
312 # -- If set to 'true' and 'extensions.cache.enabled' is also 'true'
313 #   it will try to download extensions from images (experimental)
314 allowPullExtensionsFromImageRepository: false
315 # -- It set to 'true' disable arbitrary user that is set for OpenShift clusters
316 disableArbitraryUser: false
317 # Section to define patches for some StackGres Pods
318 patches:
319   # Section to define volumes to be used by the operator container
320 operator:
321   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
322   volumes: []
323   # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
324   volumeMounts: []
325   # Section to define volumes to be used by the restapi container
326 restapi:
327   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
328   volumes: []
329   # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
330   volumeMounts: []
331   # Section to define volumes to be used by the adminui container
332 adminui:
333   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
334   volumes: []
335   # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
336   volumeMounts: []
337   # Section to define volumes to be used by the jobs container
338 jobs:
339   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
340   volumes: []
341   # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
342   volumeMounts: []
343   # Section to define volumes to be used by the cluster controller container
344 clusterController:
345   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
346   volumes: []
347   # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
348   volumeMounts: []
349   # Section to define volumes to be used by the distributedlogs controller container
350 distributedlogsController:
351   # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
352   volumes: []

```

```
353     # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
354     volumeMounts: []
```

Listing 85: StackGres-Citus - Helm Chart Manifest

Die Installation erfolgt dann wie folgt:

```
1 helm install -n sg-platform stackgres-operator stackgres-charts/stackgres-operator -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/values.yaml
```

Listing 86: StackGres-Citus - Installation

Damit nun aber der Cluster sauber läuft und auf den Cluster zugegriffen werden kann, müssen folgende Steps ebenfalls nachträglich ausgeführt werden:

```
1 # Check if the operator was successfully deployed and is available:
2 kubectl describe deployment -n sg-platform stackgres-operator
3 kubectl wait -n sg-platform deployment/stackgres-operator --for condition=Available
4 # Check if the restapi was successfully deployed and is available:
5 kubectl describe deployment -n sg-platform stackgres-restapi
6 kubectl wait -n sg-platform deployment/stackgres-restapi --for condition=Available
7 # To access StackGres Operator UI from localhost, run the below commands:
8 POD_NAME=$(kubectl get pods --namespace sg-platform -l "stackgres.io/restapi=true" -o jsonpath=".items[0].metadata.name")
9 kubectl port-forward "$POD_NAME" 8443:9443 --namespace sg-platform
```

Listing 87: StackGres-Citus - Post-Installation

Jetzt kann das GUI unter folgendem Link geöffnet werden: <https://localhost:8443>

👉 Port Forwarding

Wenn das Port Forwarding so gemacht wird, muss die CLI offenbleiben.

Sonst wird die Verbindung umgehend gekappt.

Daher empfiehlt es sich, mehrere Terminals offen zu halten.

Der Username ist admin aber das Passwort ist generisch.

Der Username liesse sich mit folgendem Command auslesen:

```
1 kubectl get secret -n sg-platform stackgres-restapi-admin --template '{{ printf "username = %s\n" (.data.k8sUsername | base64decode) }}'
```

Listing 88: StackGres-Citus - System Username

Dieses lässt sich mit folgendem Command auslesen:

```
1 kubectl get secret -n sg-platform stackgres-restapi-admin --template '{{ printf "password = %s\n" (.data.clearPassword | base64decode) }}'
```

Listing 89: StackGres-Citus - System Passwort

Am Schluss sollte das Passwort aber noch gesäubert werden:

```
1 kubectl patch secret --namespace sg-platform stackgres-restapi-admin --type json -p '[{"op":"remove","path":"/data/clearPassword"}]'
```

Listing 90: StackGres-Citus - System Passwort Cleanup

VII.V.III Deployment - Benchmarking

Zuerst wurde das Instanz-Profil für den Coordinator und die Shards deployt:

```
1 apiVersion: stackgres.io/v1
2 kind: SGInstanceProfile
3 metadata:
4   namespace: sg-platform
```

```
5   name: sg-pgbench-coordinator
6 spec:
7   cpu: "4"
8   memory: "4Gi"
```

Listing 91: StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator

```
1 apiVersion: stackgres.io/v1
2 kind: SGInstanceProfile
3 metadata:
4   namespace: sg-platform
5   name: sg-pgbench-shard
6 spec:
7   cpu: "4"
8   memory: "8Gi"
```

Listing 92: StackGres-Citus - Benchmarking - SGInstanceProfile Shard

Deploy wird ebenfalls via kubectl:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_pgbench_coord.yaml
2 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_pgbench_shard.yaml
```

Listing 93: StackGres-Citus - Benchmarking - Instanz-Profil Deploy

Auf ein Deployment einer SGPostgresConfig wurde bei den ersten drei Benchmarks verzichtet, da für Patroni in den ersten drei Benchmarks keine spezifischen Anpassungen erfuhr.

Das Manifest für den Benchmark-Cluster sieht entsprechend den Vorgaben wie folgt aus:

```
1 apiVersion: stackgres.io/v1alpha1
2 kind: SGShardedCluster
3 metadata:
4   name: sg-pgbench
5   namespace: sg-platform
6 spec:
7   type: citus
8   database: pgbench_eval_bench
9   postgres:
10    version: '16'
11   coordinator:
12    instances: 1
13   pods:
14     persistentVolume:
15      #       size: '75Gi'
16      #       size: '230Gi'
17      storageClass: "stackgres-storage"
18      disableConnectionPooling: true
19      sgInstanceProfile: "sg-pgbench-coordinator"
20   shards:
21     clusters: 3
22     instancesPerCluster: 1
23     pods:
24       persistentVolume:
25         size: '75Gi'
26         storageClass: "stackgres-storage"
27         sgInstanceProfile: "sg-pgbench-shard"
28     postgresServices:
29       coordinator:
```

```

30   primary:
31     type: LoadBalancer
32   any:
33     type: LoadBalancer
34   shards:
35     primaries:
36       type: LoadBalancer
37   metadata:
38     annotations:
39       primaryService:
40         metallb.universe.tf/loadBalancerIPs: 10.0.20.106
41     replicasService:
42       metallb.universe.tf/loadBalancerIPs: 10.0.20.153
43     externalTrafficPolicy: "Cluster"
44   profile: "testing"

```

Listing 94: StackGres-Citus - Benchmarking - SGShardedCluster

Der Deploy:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGShardedCluster_pgbench.yaml
```

Listing 95: StackGres-Citus - Benchmark - Cluster Deploy

Damit nun aber eine Verbindung auf die DB (IP 10.0.20.106) gemacht werden kann, muss das Passwort ausgelesen werden:

```
1 kubectl get secrets -n sg-platform sg-pgbench -o jsonpath='{.data.superuser-password}' | base64 -d
```

Listing 96: StackGres-Citus - Benchmark DB Passwort

VII.V.IV Deployment - Self Healing Tests

Auch hier wurde zuerst das Instanz-Profil für den Coordinator und die Shards deployt:

```

1 #Not needed actually
2 apiVersion: stackgres.io/v1
3 kind: SGInstanceProfile
4 metadata:
5   namespace: sg-platform
6   name: sg-self-healing-coordinator
7 spec:
8   cpu: "1"
9   memory: "2Gi"

```

Listing 97: StackGres-Citus - Self Healing Testing - SGInstanceProfile Coordinator

```

1 #Not needed actually
2 apiVersion: stackgres.io/v1
3 kind: SGInstanceProfile
4 metadata:
5   namespace: sg-platform
6   name: sg-self-healing-shard
7 spec:
8   cpu: "1"
9   memory: "2Gi"

```

Listing 98: StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard

Deploy wird ebenfalls via kubectl:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_self_healing_coord.yaml
  kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_self_healing_shard.yaml
```

Listing 99: StackGres-Citus - Self Healing Testing - Instanz-Profil Deploy

Auch beim Self Healing Testing wurde auf ein Deployment einer SGPostgresConfig verzichtet.

Dafür wurden drei Coordinator-Instanzen und drei Shard-Instanzen pro Cluster deklariert.

Das Manifest für den Benchmark-Cluster sieht entsprechend den Vorgaben wie folgt aus:

```
1 apiVersion: stackgres.io/v1alpha1
2 kind: SGShardedCluster
3 metadata:
4   name: sg-healing-test
5   namespace: sg-platform
6 spec:
7   type: citus
8   database: self_healing_test
9   postgres:
10     version: '16'
11   coordinator:
12     syncInstances: 3
13     replication: "strict-sync"
14   pods:
15     persistentVolume:
16       size: '2Gi'
17       storageClass: "stackgres-storage"
18     sgInstanceProfile: "sg-self-healing-coordinator"
19   shards:
20     clusters: 3
21     instancesPerCluster: 3
22     pods:
23       persistentVolume:
24         size: '5Gi'
25         storageClass: "stackgres-storage"
26     sgInstanceProfile: "sg-self-healing-shard"
27   postgresServices:
28     coordinator:
29       primary:
30         type: LoadBalancer
31       any:
32         type: LoadBalancer
33     shards:
34       primaries:
35         type: LoadBalancer
36   metadata:
37     annotations:
38       primaryService:
39         metallb.universe.tf/loadBalancerIPs: 10.0.20.152
40     replicasService:
41       metallb.universe.tf/loadBalancerIPs: 10.0.20.151
42       externalTrafficPolicy: "Cluster"
```

```
43 | profile: "testing"
```

Listing 100: StackGres-Citus - Self Healing Testing - SGShardedCluster

Der Deploy:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGShardedCluster_self_healing_test.yaml
```

Listing 101: StackGres-Citus - Self Healing Testing - Cluster Deploy

Das Passwort für den Cluster (IP 10.0.20.152) muss ebenfalls ausgelesen werden:

```
1 kubectl get secrets -n sg-platform sg-healing-test -o jsonpath='{.data.superuser-password}' | base64 -d
```

Listing 102: StackGres-Citus - Self Healing Testing DB Passwort

VII.V Rekonfiguration mit 250GiB Storage

VII.V.I Bereinigen

Zuerst wurde auch hier der komplette Namespace bereinigt, wobei es zwingend ist, vorher im GUI die Cluster zu löschen und mittels CLI den Operator zu deinstallieren. Andernfalls wird nicht alles sauber entfernt, was zu schwer nachvollziehbaren Sideeffects führt:

```
1 helm delete stackgres-operator --namespace sg-platform
2 kubectl delete namespace sg-platform
3 kubectl delete pv stackgres-storage-pv
4 kubectl delete storageclass stackgres-storage
5 kubectl delete pvc --namespace sg-platform
6 kubectl delete pvc --namespace sg-platform
```

Listing 103: StackGres-Citus - Deinstallieren

VII.V.II StorageClass setzen

```
1 # https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/prepare-environment/kubernetes/#configure-storage-class
2 # https://github.com/rancher/local-path-provisioner
3 apiVersion: storage.k8s.io/v1
4 kind: StorageClass
5 metadata:
6   name: stackgres-storage-big
7 provisioner: rancher.io/local-path
8 parameters:
9   nodePath: /srv/data/local-path-provisioner
10 volumeBindingMode: WaitForFirstConsumer
11 reclaimPolicy: Delete
12 --
13 apiVersion: v1
14 kind: PersistentVolume
15 metadata:
16   name: stackgres-storage-pv
17   labels:
18     type: local
19 spec:
20   accessModes:
21     - ReadWriteOnce
```

```

22 capacity:
23   storage: 340Gi
24 storageClassName: "stackgres-storage"
25 hostPath:
26   path: /srv/data/local-path-provisioner

```

Listing 104: StackGres-Citus - StorageClass setzen

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/storageclass_big.yaml
```

Listing 105: StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren

VII.V.V.III Installation

```
1 kubectl create namespace sg-platform
```

Listing 106: StackGres-Citus - Namespace 250GiB

Natürlich muss auch wieder das helm-Manifest deploy werden:

```
1 helm install -n sg-platform stackgres-operator stackgres-charts/stackgres-operator -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/values.yaml
```

Listing 107: StackGres-Citus - Installation 250GiB

VII.V.V.IV Deployment - Benchmarking mit 250GiB

Die Ressourcen für die Shards mussten erhöht werden:

```

1 apiVersion: stackgres.io/v1
2 kind: SGInstanceProfile
3 metadata:
4   namespace: sg-platform
5   name: sg-pgbench-coordinator
6 spec:
7   cpu: "4"
8   memory: "4Gi"

```

Listing 108: StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB

```

1 apiVersion: stackgres.io/v1
2 kind: SGInstanceProfile
3 metadata:
4   namespace: sg-platform
5   name: sg-pgbench-shard
6 spec:
7   cpu: "4"
8   memory: "12Gi"

```

Listing 109: StackGres-Citus - Benchmarking - SGInstanceProfile Shard 250GiB

```

1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_pgbench_coord.yaml
2 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_pgbench_shard.yaml

```

Listing 110: StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB

Nun wurde ein SGPostgresConfig erstellt.

Die Settings entsprechen der Patroni-Konfiguration.

Bei StackGres gibt es allerdings einige Parameter, die für die Konfiguration gesperrt sind.

In diesem Fall waren es folgende:

- wal_log_hints
- wal_level
- archive_mode

Das Manifest sieht folgendermassen aus:

```
1 #Not needed actually
2 apiVersion: stackgres.io/v1
3 kind: SGPostgresConfig
4 metadata:
5   namespace: sg-platform
6   name: sg-evaluation-250gib
7 spec:
8   postgresVersion: '16'
9   postgresql.conf:
10    random_page_cost: '1.5'
11    password_encryption: 'scram-sha-256'
12    log_checkpoints: 'on'
13    max_connections: '1100'
14    superuser_reserved_connections: '10'
15    max_worker_processes: '16'
16    #   wal_log_hints: 'on'
17    max_wal_senders: '32'
18    max_replication_slots: '32'
19    wal_keep_size: '1GB'
20    #   wal_level: 'logical'
21    wal_buffers: '16MB'
22    wal_writer_delay: '20ms'
23    wal_writer_flush_after: '1MB'
24    min_wal_size: '1GB'
25    max_wal_size: '5GB'
26    commit_delay: '20'
27    commit_siblings: '10'
28    checkpoint_timeout: '5min'
29    checkpoint_completion_target: '0.95'
30    #   archive_mode: 'off'
31    max_standby_archive_delay: '10min'
32    max_standby_streaming_delay: '3min'
33    wal_receiver_status_interval: '1s'
34    hot_standby_feedback: 'on'
35    wal_receiver_timeout: '60s'
36    max_logical_replication_workers: '8'
37    max_sync_workers_per_subscription: '8'
38    shared_buffers: '4GB'
39    maintenance_work_mem: '1GB'
40    work_mem: '12GB'
41    temp_file_limit: '200GB'
```

```

42 vacuum_cost_delay: '2ms'
43 vacuum_cost_limit: '10000'
44 bgwriter_delay: '10ms'
45 bgwriter_lru_maxpages: '800'
46 bgwriter_lru_multiplier: '5.0'

```

Listing 111: StackGres-Citus - Benchmarking - SGPostgresConfig

Der Deploy:

```

1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGPostgresConfig.yaml

```

Listing 112: StackGres-Citus - Benchmark - Deploy SGPostgresConfig

Das Manifest für den Benchmark-Cluster wurde die Shard-Cluster Anzahl auf 2 reduziert und die PVC entsprechend dimensioniert.

Zudem wurde das SGPostgresConfig-Profil sgPostgresConfig eingebunden:

```

1 apiVersion: stackgres.io/v1alpha1
2 kind: SGShardedCluster
3 metadata:
4   name: sg-pgbench
5   namespace: sg-platform
6 spec:
7   type: citus
8   database: pgbench_eval_bench
9   postgres:
10     version: '16'
11   coordinator:
12     instances: 1
13   pods:
14     persistentVolume:
15       size: '230Gi'
16       storageClass: "stackgres-storage"
17     disableConnectionPooling: true    # gramic, 19.04.2024: create_distributed_table auf pgbench_accounts ässt sich nicht üausfren wegen pgbouncer problemen
18     sgInstanceProfile: "sg-pgbench-coordinator"
19   configurations:
20     sgPostgresConfig: "sg-evaluation-250gib"
21   shards:
22     clusters: 2 # gramic, 19.04.2024: 250GiB Tabelle
23     instancesPerCluster: 1
24     pods:
25       persistentVolume:
26         size: '230Gi'
27         storageClass: "stackgres-storage"
28       sgInstanceProfile: "sg-pgbench-shard"
29     configurations:
30       sgPostgresConfig: "sg-evaluation-250gib"
31   postgresServices:
32     coordinator:
33       primary:
34         type: LoadBalancer
35       any:
36         type: LoadBalancer
37     shards:
38       primaries:
39         type: LoadBalancer

```

```

40 metadata:
41   annotations:
42     primaryService:
43       metallb.universe.tf/loadBalancerIPs: 10.0.20.106
44     replicasService:
45       metallb.universe.tf/loadBalancerIPs: 10.0.20.153
46     externalTrafficPolicy: "Cluster"
47   profile: "testing"

```

Listing 113: StackGres-Citus - Benchmarking - SGShardedCluster 250GiB

Der Deploy:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGShardedCluster_pgbench.yaml
```

Listing 114: StackGres-Citus - Benchmark - Cluster Deploy 250GiB

VII.V.II SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle pgbench_eval_bench bereits beim Deployment erstellt.

Allerdings ohne Tablespace.

Daher sind keine weiteren Schritte an dieser Stelle notwendig, die Tabellen werden von pgbench beim Initialisieren erstellt.

VII.V.III SQL Statements - Testing

Auch hier wird die Tabelle bereits beim Deployment des Clusters erstellt.

Es müssen aber natürlich noch gemäss ERD die Tabellen erstellt werden: [Evaluation - ERD self_healing_test](#) Auch hier wird auf Tablespace verzichtet.

Erst werden die Rollen erstellt, gefolgt von den Usern und Schemas.

Die Schemas müssen entsprechend mittels GRANT berechtigt werden. Die Tabellen müssen entsprechend den Schemas erstellt werden, es werden Reference Table Shards erzeugt.

Das gesamte CREATE-Skript:

```

1 -- Rollen erstellen
2 drop role if exists hrm;
3 create role hrm;
4 drop role if exists accountands;
5 create role accountands;
6 drop role if exists customer_service_officers;
7 create role customer_service_officers;
8 drop role if exists legal_affairs;
9 create role legal_affairs;
10
11 -- User erstellen
12 drop user if exists hrm_1;
13 drop user if exists hrm_2;
14 create user hrm_1 with password 'hrm1' role hrm;
15 create user hrm_2 with password 'hrm2' role hrm;
16
17 drop user if exists cso_1;
18 drop user if exists cso_2;
19 create user cso_1 with password 'cso1' role customer_service_officers;
20 create user cso_2 with password 'cso2' role customer_service_officers;
21
22 drop user if exists la_1;

```

```

23 drop user if exists la_2;
24 create user la_1 with password 'la1' role legal_affairs;
25 create user la_2 with password 'la2' role legal_affairs;
26 -- Schemas erstellen
27 drop schema if exists hrm;
28 create schema hrm authorization hrm;
29 drop schema if exists accountands;
30 create schema accountands authorization accountands;
31 drop schema if exists customer_service_officers;
32 create schema customer_service_officers authorization customer_service_officers;
33 drop schema if exists generell;
34 create schema generell;
35
36 -- GRANTS erstellen
37 grant all on all tables in schema hrm to legal_affairs;
38 grant all on all tables in schema accountands to legal_affairs;
39 grant all on all tables in schema customer_service_officers to legal_affairs;
40 grant all on all tables in schema generell to legal_affairs;
41 grant all on all tables in schema hrm to postgres;
42 grant all on all tables in schema accountands to postgres;
43 grant all on all tables in schema customer_service_officers to postgres;
44 grant all on all tables in schema generell to postgres;
45
46 -- self_healing_accounts für Schema customer_service_officers
47 drop table if exists customer_service_officers.self_healing_accounts;
48 create table customer_service_officers.self_healing_accounts (
49     account_id int primary key,
50     firstname varchar(255) not null,
51     lastname varchar(255) not null,
52     birthday date not null,
53     postal_code varchar(50),
54     street varchar(255),
55     country_code varchar(2),
56     phone varchar(25),
57     mail varchar(255) check (mail like '%@%')
58 );
59 create unique index accounts_personal_mark on customer_service_officers.self_healing_accounts(firstname, lastname, birthday);
60 SELECT create_reference_table('customer_service_officers.self_healing_accounts');
61
62 -- self_healing_employees für Schema hrm
63 drop table if exists hrm.self_healing_employees;
64 create table hrm.self_healing_employees (
65     employees_id int primary key,
66     firstname varchar(255) not null,
67     lastname varchar(255) not null,
68     birthday date not null,
69     postal_code varchar(50),
70     street varchar(255),
71     country_code varchar(2),
72     phone varchar(25),
73     mail varchar(255) check (mail like '%@%')
74 );
75 create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday);
76 SELECT create_reference_table('hrm.self_healing_employees');

```

```

77
78 -- self_healing_accountand_protocol für Schema accountands drop table if exists accountands.self_healing_accountand_protocol;
79 create table accountands.self_healing_accountand_protocol (
80     acc_protocol_id int primary key,
81     description varchar(100) not null,
82     protocol_date date not null,
83     employees_id int not null,
84     rapport TEXT,
85     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
86 );
87 SELECT create_reference_table('accountands.self_healing_accountand_protocol');
88
89 -- self_healing_intranet für public Schema
90 drop table if exists generell.self_healing_intranet;
91 create table generell.self_healing_intranet (
92     intranet_id int primary key,
93     content text
94 );
95 SELECT create_reference_table('generell.self_healing_intranet');
96
97 -- self_healing_intranet für public Schema
98 drop table if exists generell.self_healing_intranet_users;
99 create table generell.self_healing_intranet_users (
100    intranet_user_id int primary key,
101    employees_id int not null,
102    foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
103 );
104 create unique index intranet_unique_combi on generell.self_healing_intranet_users(intranet_user_id, employees_id);
105 SELECT create_reference_table('generell.self_healing_intranet_users');

```

Listing 115: StackGres-Citus - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```

1 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
2 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
3 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
4
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
6 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
7 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');
8
9 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (100, 'bla', '07.04.2024', 100, 'blabla');
10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (200, 'yada', '07.04.2024', 100, 'ydayadyada');
11 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (300, 'something', '07.04.2024', 300, 'something');
12
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
15 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');
16
17 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(100, 100);
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(200, 200);
19 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(300, 300);

```

```

20
21 select * from customer_serviceOfficers.self_healing_accounts; select * from hrm.self_healing_employees;
22 select * from accountands.self_healing_accountand_protocol;
23 select * from generell.self_healing_intranet_users;

```

Listing 116: StackGres-Citus - Self Healing Tests - Init Data

Während dem Failover-Test müssen Daten beschrieben werden:

```

1 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
2 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
3 insert into customer_serviceOfficers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
4
5 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
6 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
7 insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');
8
9 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (400, 'bla', '07.04.2024', 200, 'blabla');
10 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (500, 'yada', '07.04.2024', 600, 'ydayadyada');
   );
11 insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport) values (1000, 'something', '07.04.2024', 300, 'something');
12
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (200, 'yadada');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (600, 'bla bla');
15 insert into generell.self_healing_intranet(intranet_id, content) VALUES (900, 'talking and talking');
16
17 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(400, 400);
18 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(500, 500);
19 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(600, 600);
20
21 select * from customer_serviceOfficers.self_healing_accounts;
22 select * from hrm.self_healing_employees;
23 select * from accountands.self_healing_accountand_protocol;
24 select * from generell.self_healing_intranet;
25 select * from generell.self_healing_intranet_users;

```

Listing 117: StackGres-Citus - Self Healing Tests - Failover Data

Nach dem Recovery müssen die Daten entsprechend vorhanden sein und es müssen weitere Daten beschrieben werden können:

```

1 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(1000, 400);
2 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(2000, 500);
3 insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(3000, 600);
4
5 select count(*) from customer_serviceOfficers.self_healing_accounts;
6 select count(*) from hrm.self_healing_employees;
7 select count(*) from accountands.self_healing_accountand_protocol;
8 select count(*) from generell.self_healing_intranet;
9 select count(*) from generell.self_healing_intranet_users;

```

Listing 118: StackGres-Citus - Self Healing Tests - Recovery Data

VIII Evaluationssysteme - Benchmarking

VIII.I YugabyteDB

Pro Lauf werden erst die Daten initialisiert werden.

Wichtig ist dabei, dass die beiden Tablespaces eval_index_tablespace und eval_data_tablespace mitgegeben werden.

Anschliessend werden die Benchmarks an sich ausgeführt.

Die Resultate werden weggeschrieben.

```
1 ######
2 # 1. Lauf      #
3 # ca. 5GiB    #
4 #####
5 # Init
6 ./ysql_bench -h 10.0.20.106 -p 5433 -i -s 400 --foreign-keys -F 100 -I dtgvpf --index-
  tablespace=eval_index_tablespace --tablespace=eval_data_tablespace -U yadmin
  pgbench_eval_bench
7
8 # Benchmarking mixed
9 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -U yadmin pgbench_eval_bench >
  /home/gramic/1_1_yugabytedb_mixed_benchmark.txt
10 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -U yadmin pgbench_eval_bench >
  /home/gramic/1_2_yugabytedb_mixed_benchmark.txt
11 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -U yadmin pgbench_eval_bench >
  /home/gramic/1_3_yugabytedb_mixed_benchmark.txt
12 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -U yadmin pgbench_eval_bench >
  /home/gramic/1_4_yugabytedb_mixed_benchmark.txt
13
14 # Benchmarking dql
15 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -S -U yadmin
  pgbench_eval_bench > /home/gramic/1_1_yugabytedb_dql_benchmark.txt
16 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -S -U yadmin
  pgbench_eval_bench > /home/gramic/1_2_yugabytedb_dql_benchmark.txt
17 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -S -U yadmin
  pgbench_eval_bench > /home/gramic/1_3_yugabytedb_dql_benchmark.txt
18 ./ysql_bench -h 10.0.20.106 -p 5433 -c 10 -C -j 4 -v -t 10 -S -U yadmin
  pgbench_eval_bench > /home/gramic/1_3_yugabytedb_dql_benchmark.txt
19
20 #####
21 # 2. Lauf      #
22 # ca. 15GiB   #
23 #####
24 # Init
25 ./ysql_bench -h 10.0.20.106 -p 5433 -i -s 1200 --foreign-keys -F 100 -I dtgvpf --index-
  tablespace=eval_index_tablespace --tablespace=eval_data_tablespace -U yadmin
  pgbench_eval_bench
26
27 # Benchmarking mixed
28 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench >
  /home/gramic/2_1_yugabytedb_mixed_benchmark.txt
29 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench >
  /home/gramic/2_2_yugabytedb_mixed_benchmark.txt
30 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench >
```

```

    /home/gramic/2_3_yugabytedb_mixed_benchmark.txt
31 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench >
    /home/gramic/2_4_yugabytedb_mixed_benchmark.txt
32 # Benchmarking dql
33 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/2_1_yugabytedb_dql_benchmark.txt
34 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/2_2_yugabytedb_dql_benchmark.txt
35 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/2_3_yugabytedb_dql_benchmark.txt
36 ./ysql_bench -h 10.0.20.106 -p 5433 -c 50 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/2_4_yugabytedb_dql_benchmark.txt
37
38 #####
39 # 3. Lauf #
40 # ca. 50GiB #
41 #####
42 ./ysql_bench -h 10.0.20.106 -p 5433 -i -s 3999 --foreign-keys -F 100 -I dtgvpf --index-
    tablespace=eval_index_tablespace --tablespace=eval_data_tablespace -U yadmin
    pgbench_eval_bench
43
44 # Benchmarking mixed
45 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench
    > /home/gramic/3_1_yugabytedb_mixed_benchmark.txt
46 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench
    > /home/gramic/3_2_yugabytedb_mixed_benchmark.txt
47 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench
    > /home/gramic/3_3_yugabytedb_mixed_benchmark.txt
48 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -U yadmin pgbench_eval_bench
    > /home/gramic/3_4_yugabytedb_mixed_benchmark.txt
49
50 # Benchmarking dql
51 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/3_1_yugabytedb_dql_benchmark.txt
52 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/3_2_yugabytedb_dql_benchmark.txt
53 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/3_3_yugabytedb_dql_benchmark.txt
54 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin
    pgbench_eval_bench > /home/gramic/3_4_yugabytedb_dql_benchmark.txt
55
56
57 #####
58 # 4. Lauf #
59 # ca. 250GiB #
60 #####
61 ./ysql_bench -h 10.0.20.106 -p 5433 -i -s 16784 --foreign-keys -F 100 -I dtgvpf --index-
    tablespace=eval_index_tablespace --tablespace=eval_data_tablespace -U yadmin
    pgbench_eval_bench
62
63 # Benchmarking mixed
64 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -U yadmin pgbench_eval_bench
    > /home/gramic/4_1_yugabytedb_mixed_benchmark.txt
65 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -U yadmin pgbench_eval_bench

```

```

> /home/gramic/4_2_yugabytedb_mixed_benchmark.txt
66 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -U yadmin pgbench_eval_bench
    > /home/gramic/4_3_yugabytedb_mixed_benchmark.txt./ysql_bench -h 10.0.20.106 -p
        5433 -c 25 -C -j 4 -v -t 280 -U yadmin pgbench_eval_bench > /home/gramic/4
            _4_yugabytedb_mixed_benchmark.txt
67
68 # Benchmarking dql
69 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -S -U yadmin
    pgbench_eval_bench > /home/gramic/4_1_yugabytedb_dql_benchmark.txt
70 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -S -U yadmin
    pgbench_eval_bench > /home/gramic/4_2_yugabytedb_dql_benchmark.txt
71 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -S -U yadmin
    pgbench_eval_bench > /home/gramic/4_3_yugabytedb_dql_benchmark.txt
72 ./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -S -U yadmin
    pgbench_eval_bench > /home/gramic/4_4_yugabytedb_dql_benchmark.txt

```

Listing 119: YugabyteDB - Benchmarking-Commands

Die Grösse der Tabellen lässt sich wie folgt auslesen:

```

1 select
2   table_name ,
3   pg_size.pretty(pg_total_relation_size(quote_ident(table_name))),
4   pg_total_relation_size(quote_ident(table_name))
5 from information_schema.tables
6 where table_schema = 'public'
7 order by 3 desc;

```

Listing 120: YugabyteDB - Benchmarking - Table Size SQL

VIII.II Patroni

Pro Lauf werden erst die Daten initialisiert werden.

Wichtig ist dabei, dass die beiden Tablespaces eval_index_tablespace und eval_data_tablespace mitgegeben werden.

Anschliessend werden die Benchmarks an sich ausgeführt.

Die Resultate werden weggeschrieben.

```

1 #####
2 # 1. Lauf      #
3 # ca. 5GiB     #
4 #####
5 # Init
6 pgbench --host=10.0.28.16 --port=5432 --initialize --scale=400 --foreign-keys --
    fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace
    =eval_data_tablespace --username=postgres pgbench_eval_bench
7
8 # Benchmarking mixed
9 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
    home/itgramic/1_1_patroni_mixed_benchmark.txt
10 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
    home/itgramic/1_2_patroni_mixed_benchmark.txt
11 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
    home/itgramic/1_3_patroni_mixed_benchmark.txt

```

```

12 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_4_patroni_mixed_benchmark.txt
13 # Benchmarking dql pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_1_patroni_dql_benchmark.txt
14 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_2_patroni_dql_benchmark.txt
15 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_3_patroni_dql_benchmark.txt
16 pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_4_patroni_dql_benchmark.txt
17 #####
18 #####2. Lauf#####
19 # 2. Lauf #
20 # ca. 15GiB #
21 #####
22 # Init
23 pgbench --host=10.0.28.16 --port=5432 --initialize --scale=1200 --foreign-keys --
    fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace
    =eval_data_tablespace --username=postgres pgbench_eval_bench
24
25 # Benchmarking mixed
26 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_1_patroni_mixed_benchmark.txt
27 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_2_patroni_mixed_benchmark.txt
28 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_3_patroni_mixed_benchmark.txt
29 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_4_patroni_mixed_benchmark.txt
30 # Benchmarking dql
31 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_1_patroni_dql_benchmark.txt
32 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_2_patroni_dql_benchmark.txt
33 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_3_patroni_dql_benchmark.txt
34 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_4_patroni_dql_benchmark.txt
35 #####
36 #####3. Lauf#####
37 # 3. Lauf #
38 # ca. 50GiB #
39 #####
40 pgbench --host=10.0.28.16 --port=5432 --initialize --scale=3999 --foreign-keys --
    fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace
    =eval_data_tablespace --username=postgres pgbench_eval_bench
41
42 # Benchmarking mixed
43 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_1_patroni_mixed_benchmark.txt
44 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_2_patroni_mixed_benchmark.txt
45 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_3_patroni_mixed_benchmark.txt

```

```

46 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /
    home/itgramic/3_4_patroni_mixed_benchmark.txt
47 # Benchmarking dql pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres
    pgbench_eval_bench > /home/itgramic/3_1_patroni_dql_benchmark.txt
48 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench >
    /home/itgramic/3_2_patroni_dql_benchmark.txt
49 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench >
    /home/itgramic/3_3_patroni_dql_benchmark.txt
50 pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench >
    /home/itgramic/3_4_patroni_dql_benchmark.txt
51
52
53 ######
54 # 4. Lauf #
55 # ca. 250GiB #
56 #####
57 pgbench --host=10.0.28.16 --port=5432 --initialize --scale=16784 --foreign-keys --
    fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace
    =eval_data_tablespace --username=postgres pgbench_eval_bench
58
59 # Benchmarking mixed
60 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_1_patroni_mixed_benchmark.txt
61 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_2_patroni_mixed_benchmark.txt
62 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_3_patroni_mixed_benchmark.txt
63 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_4_patroni_mixed_benchmark.txt
64 # Benchmarking dql
65 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench >
    /home/itgramic/4_1_patroni_dql_benchmark.txt
66 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench >
    /home/itgramic/4_2_patroni_dql_benchmark.txt
67 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench >
    /home/itgramic/4_3_patroni_dql_benchmark.txt
68 pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench >
    /home/itgramic/4_4_patroni_dql_benchmark.txt

```

Listing 121: Patroni - Benchmarking-Commands

Die Grösse der Tabellen lässt sich wie folgt auslesen:

```
1 SELECT pg_size.pretty( pg_database_size('pgbench_eval_bench') );
```

Listing 122: Patroni - Benchmarking - Table Size SQL

VIII.III StackGres - Citus

Pro Lauf werden erst die Daten initialisiert **werden**.

Wichtig ist dabei, dass keine Tablespace mitgegeben werden, da diese nicht existieren.

Anschliessend werden die Benchmarks an sich ausgeführt.

Die Resultate werden weggeschrieben.

```

1 ##########
2 #   1. Lauf      ##  ca. 5GiB      #
3 #########
4 # Init
5 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=400 --foreign-keys --
   fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
6
7 # Benchmarking mixed
8 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
  home/itgramic/1_1_stackgresmixed_benchmark.txt
9 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
  home/itgramic/1_2_stackgresmixed_benchmark.txt
10 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
  home/itgramic/1_3_stackgresmixed_benchmark.txt
11 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /
  home/itgramic/1_4_stackgresmixed_benchmark.txt
12
13 # Benchmarking dql
14 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench >
  /home/itgramic/1_1_stackgresdql_benchmark.txt
15 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench >
  /home/itgramic/1_2_stackgresdql_benchmark.txt
16 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench >
  /home/itgramic/1_3_stackgresdql_benchmark.txt
17 pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench >
  /home/itgramic/1_4_stackgresdql_benchmark.txt
18
19 #########
20 #   2. Lauf      #
21 #  ca. 15GiB     #
22 #########
23 # Init
24 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=1200 --foreign-keys --
   fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
25
26 # Benchmarking mixed
27 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /
  home/itgramic/2_1_stackgres_mixed_benchmark.txt
28 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /
  home/itgramic/2_2_stackgres_mixed_benchmark.txt
29 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /
  home/itgramic/2_3_stackgres_mixed_benchmark.txt
30 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /
  home/itgramic/2_4_stackgres_mixed_benchmark.txt
31 # Benchmarking dql
32 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench >
  /home/itgramic/2_1_stackgres_dql_benchmark.txt
33 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench >
  /home/itgramic/2_2_stackgres_dql_benchmark.txt
34 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench >
  /home/itgramic/2_3_stackgres_dql_benchmark.txt
35 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench >
  /home/itgramic/2_4_stackgres_dql_benchmark.txt
36

```

```

37 #####
38 #   3. Lauf      ##  ca. 50GiB      #
39 #####
40 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=3999 --foreign-keys --
    fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
41
42 # Benchmarking mixed
43 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /
    home/itgramic/3_1_stackgres_mixed_benchmark.txt
44 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /
    home/itgramic/3_2_stackgres_mixed_benchmark.txt
45 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /
    home/itgramic/3_3_stackgres_mixed_benchmark.txt
46 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /
    home/itgramic/3_4_stackgres_mixed_benchmark.txt
47 # Benchmarking dql
48 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench
    > /home/itgramic/3_1_stackgres_dql_benchmark.txt
49 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench
    > /home/itgramic/3_2_stackgres_dql_benchmark.txt
50 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench
    > /home/itgramic/3_3_stackgres_dql_benchmark.txt
51 pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench
    > /home/itgramic/3_4_stackgres_dql_benchmark.txt
52
53
54 #####
55 #   4. Lauf      #
56 #  ca. 250GiB   #
57 #####
58 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=16784 --foreign-keys --
    fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
59
60 # Benchmarking mixed
61 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_1_stackgresmixed_benchmark.txt
62 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_2_stackgresmixed_benchmark.txt
63 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_3_stackgresmixed_benchmark.txt
64 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /
    home/itgramic/4_4_stackgresmixed_benchmark.txt
65 # Benchmarking dql
66 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench
    > /home/itgramic/4_1_stackgresdql_benchmark.txt
67 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench
    > /home/itgramic/4_2_stackgresdql_benchmark.txt
68 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench
    > /home/itgramic/4_3_stackgresdql_benchmark.txt
69 pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench
    > /home/itgramic/4_4_stackgresdql_benchmark.txt

```

Listing 123: StackGres-Citus - Benchmarking-Commands

Die Grösse der Tabellen lässt sich wie folgt auslesen:

```
1 SELECT * FROM citus_tables;
```

Listing 124: StackGres-Citus - Benchmarking - Table Size SQL

Das Resultat ist aber an sich zu gross, da bei den Reference Tables alle Shards und die Coordinators zusammengerechnet werden.

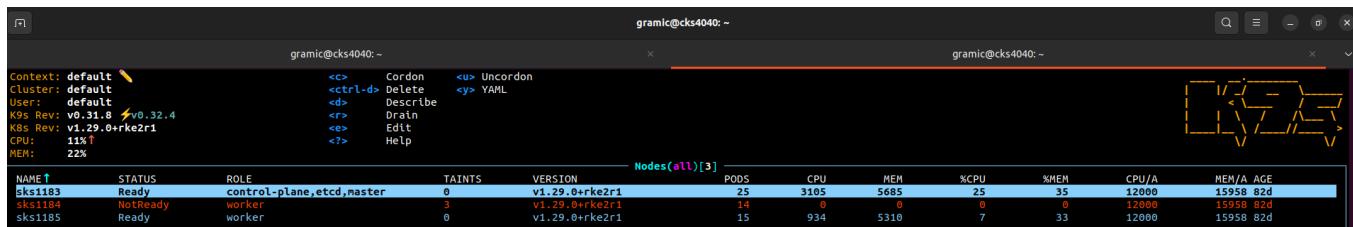
Die korrekte Grösse muss wie folgt kalkuliert werden:

$$realsize = \frac{size}{[coordinator_{syncInstances} + (shard_{clusters} \times shard_{instancesPerCluster})]}$$

IX Evaluationssysteme - Testing

IX.I StackGres - Citus

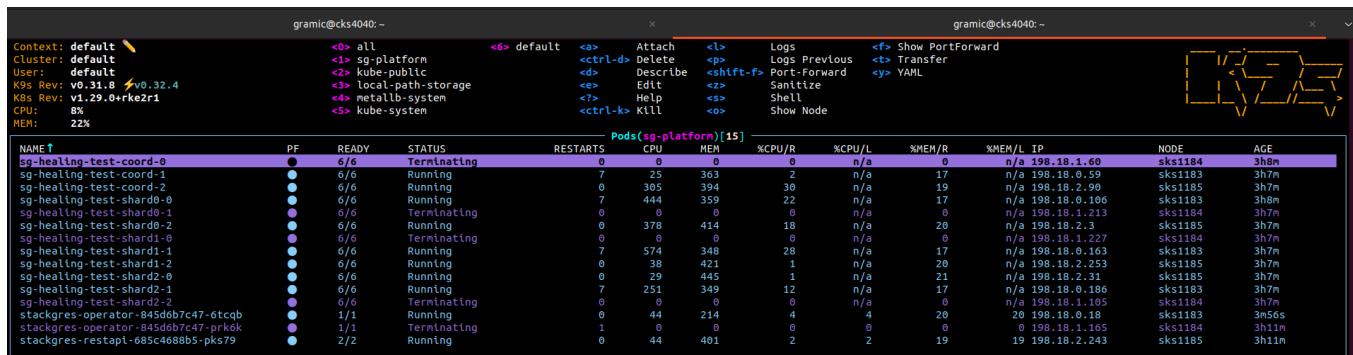
Der Node ging down, als der Server `sks1184` heruntergefahren wurde:



NAME	STATUS	ROLE	TRAINTS	VERSION	PODS	CPU	MEM	%CPU	%MEM	CPU/A	MEM/A, AGE
sks1183	Ready	control-plane,etcd,master	0	v1.29.0+rke2r1	25	3105	5685	25	35	12000	15958 82d
sks1184	NotReady	worker	0	v1.29.0+rke2r1	14	0	0	0	0	12000	15958 82d
sks1185	Ready	worker	0	v1.29.0+rke2r1	15	934	5310	7	33	12000	15958 82d

Abbildung XL: StackGres Testing - Node sks1184 down

Entsprechend wurden die Pods ebenfalls auf terminating gesetzt:



NAME	PF	READY	STATUS	RESTARTS	PODS(sg-platform)	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE	AGE
sg-healing-test-coord-0	●	6/6	Terminating	0	0	0	0	n/a	0	n/a	198.18.1.60	sks1184	3h8m	
sg-healing-test-coord-1	●	6/6	Running	7	25	363	2	n/a	17	n/a	198.18.0.59	sks1183	3h7m	
sg-healing-test-coord-2	●	6/6	Running	0	305	394	30	n/a	19	n/a	198.18.2.90	sks1185	3h7m	
sg-healing-test-shard0-0	●	6/6	Running	7	444	359	22	n/a	17	n/a	198.18.0.106	sks1183	3h8m	
sg-healing-test-shard0-1	●	6/6	Terminating	0	0	0	0	n/a	0	n/a	198.18.1.213	sks1184	3h7m	
sg-healing-test-shard0-2	●	6/6	Running	0	378	414	18	n/a	20	n/a	198.18.2.3	sks1185	3h7m	
sg-healing-test-shard1-0	●	6/6	Terminating	0	0	0	0	n/a	0	n/a	198.18.1.227	sks1184	3h7m	
sg-healing-test-shard1-1	●	6/6	Running	7	574	348	28	n/a	17	n/a	198.18.0.163	sks1183	3h7m	
sg-healing-test-shard1-2	●	6/6	Running	0	38	421	1	n/a	20	n/a	198.18.2.253	sks1185	3h7m	
sg-healing-test-shard2-0	●	6/6	Running	0	29	445	1	n/a	21	n/a	198.18.2.31	sks1185	3h7m	
sg-healing-test-shard2-1	●	6/6	Running	7	251	349	12	n/a	17	n/a	198.18.0.186	sks1183	3h7m	
sg-healing-test-shard2-2	●	6/6	Terminating	0	0	0	0	n/a	0	n/a	198.18.1.105	sks1184	3h7m	
stackgres-operator-845db7c47-6tcqb	●	1/1	Running	0	44	214	4	4	20	2	198.18.0.18	sks1183	3m56s	
stackgres-operator-845db7c47-prk6k	●	1/1	Terminating	1	0	0	0	0	0	0	198.18.1.165	sks1184	3h11m	
stackgres-restapi-685c4688b5-pks79	●	2/2	Running	0	44	401	2	2	19	19	198.18.2.243	sks1185	3h11m	

Abbildung XLI: StackGres Testing - Pods Down

Der Patroni-Leader des Coordinators, aber auch die der Shards wurden einem Failover ausgeführt:

```

gramic@cks4040: $ kubectl exec -it sg-healing-test-coord-0 -n sg-platform -c patroni -- patronictl list
+ Citus cluster: sg-healing-test --+-----+-----+-----+-----+
| Group | Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+-----+
| 0 | sg-healing-test-coord-0 | 198.18.1.60:7433 | Leader | running | 2 | |
| 0 | sg-healing-test-coord-1 | 198.18.0.59:7433 | Sync Standby | streaming | 2 | 0 |
| 0 | sg-healing-test-coord-2 | 198.18.2.90:7433 | Sync Standby | streaming | 2 | 0 |
| 1 | sg-healing-test-shard0-0 | 198.18.0.106:7433 | Replica | streaming | 2 | 0 |
| 1 | sg-healing-test-shard0-1 | 198.18.1.213:7433 | Leader | running | 2 | |
| 1 | sg-healing-test-shard0-2 | 198.18.2.3:7433 | Replica | streaming | 2 | 0 |
| 2 | sg-healing-test-shard1-0 | 198.18.1.227:7433 | Replica | streaming | 2 | 0 |
| 2 | sg-healing-test-shard1-1 | 198.18.0.163:7433 | Replica | streaming | 2 | 0 |
| 2 | sg-healing-test-shard1-2 | 198.18.2.253:7433 | Leader | running | 2 | |
| 3 | sg-healing-test-shard2-0 | 198.18.2.31:7433 | Replica | streaming | 2 | 0 |
| 3 | sg-healing-test-shard2-1 | 198.18.0.186:7433 | Replica | streaming | 2 | 0 |
| 3 | sg-healing-test-shard2-2 | 198.18.1.105:7433 | Leader | running | 2 | |
+-----+-----+-----+-----+-----+-----+-----+
gramic@cks4040: $ kubectl exec -it sg-healing-test-coord-0 -n sg-platform -c patroni -- patronictl list
Error from server: error dialing backend: proxy error from 127.0.0.1:9345 while dialing 10.0.20.104:10250, code 502: 502 Bad Gateway
gramic@cks4040: $ kubectl exec -it sg-healing-test-coord-1 -n sg-platform -c patroni -- patronictl list
+ Citus cluster: sg-healing-test --+-----+-----+-----+-----+
| Group | Member | Host | Role | State | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+-----+
| 0 | sg-healing-test-coord-0 | 198.18.1.60:7433 | Replica | running | 2 | 0 |
| 0 | sg-healing-test-coord-1 | 198.18.0.59:7433 | Sync Standby | streaming | 3 | 0 |
| 0 | sg-healing-test-coord-2 | 198.18.2.90:7433 | Leader | running | 3 | |
| 1 | sg-healing-test-shard0-0 | 198.18.0.106:7433 | Replica | streaming | 3 | 0 |
| 1 | sg-healing-test-shard0-1 | 198.18.1.213:7433 | Replica | running | 2 | 0 |
| 1 | sg-healing-test-shard0-2 | 198.18.2.3:7433 | Leader | running | 3 | |
| 2 | sg-healing-test-shard1-0 | 198.18.1.227:7433 | Replica | streaming | 2 | 0 |
| 2 | sg-healing-test-shard1-1 | 198.18.0.163:7433 | Replica | streaming | 2 | 0 |
| 2 | sg-healing-test-shard1-2 | 198.18.2.253:7433 | Leader | running | 2 | |
| 3 | sg-healing-test-shard2-0 | 198.18.2.31:7433 | Leader | running | 3 | |
| 3 | sg-healing-test-shard2-1 | 198.18.0.186:7433 | Replica | streaming | 3 | 0 |
| 3 | sg-healing-test-shard2-2 | 198.18.1.105:7433 | Replica | running | 2 | 0 |
+-----+-----+-----+-----+-----+-----+-----+

```

Abbildung XLII: StackGres Testing - Patroni Übersicht

Während dieser Zeit ist die DB immer erreichbar:

The screenshot shows a PostgreSQL client interface. At the top, there is a query editor window containing the following SQL statements:

```

184 -! After rebuild
185 ✓ select * from customer_serviceOfficers.self_healing_accounts;
186 ✓ select * from hrm.self_healing_employees;
187 ✓ select * from accountands.self_healing_accountand_protocol;
188 ✓ select * from generell.self_healing_intranet;
189 ✓ select * from generell.self_healing_intranet_users;
190

```

Below the query editor is a table viewer window titled "self_healing_test.hrm.self_healing_employees". The table has two columns: "intranet_user_id" and "employees_id". The data is as follows:

	intranet_user_id	employees_id
1	100	100
2	200	200
3	300	300
4	400	400
5	500	500
6	600	600
7	700	400
8	800	500
9	900	600

Abbildung XLIII: StackGres Testing - DB Zugriff

Allerdings werden längere Transaktionen geschlossen:

Citus cluster: sg-healing-test						
Group	Member	Host	Role	State	TL	Lag in MB
0	sg-healing-test-coord-0	198.18.1.106:7433	Sync Standby	streaming	3	0
0	sg-healing-test-coord-1	198.18.0.59:7433	Sync Standby	streaming	3	0
0	sg-healing-test-coord-2	198.18.2.90:7433	Leader	running	3	
1	sg-healing-test-shard0-0	198.18.0.106:7433	Replica	streaming	3	0
1	sg-healing-test-shard0-1	198.18.1.71:7433	Replica	streaming	3	0
1	sg-healing-test-shard0-2	198.18.2.3:7433	Leader	running	3	
2	sg-healing-test-shard1-0	198.18.1.155:7433	Replica	streaming	2	0
2	sg-healing-test-shard1-1	198.18.0.163:7433	Replica	streaming	2	0
2	sg-healing-test-shard1-2	198.18.2.253:7433	Leader	running	2	
3	sg-healing-test-shard2-0	198.18.2.31:7433	Leader	running	3	
3	sg-healing-test-shard2-1	198.18.0.186:7433	Replica	streaming	3	0
3	sg-healing-test-shard2-2	198.18.1.163:7433	Replica	streaming	3	0

Abbildung XLIV: StackGres Testing - Connection Timeout

IX.II YugabyteDB

Zum einen kann der Fehler irgendwann auftreten.

In diesem Fall wird erst im Log die Fehlermeldung geworfen, dass die Zeitdifferenz zu gross ist:

```

Logs(yb-platform/yb-tserver-1)[tail]
Autoscroll:On FullScreen:Off Timestamps:Off Wrap:Off

yb-cleanup Permitted disk usage for core dump files in kb: 25691590
yb-cleanup Disk usage by core dump files in kb: 4
yb-cleanup Permitted disk usage for yb-tserver*log.* files in kb: 5000000
yb-cleanup Disk usage by yb-tserver*log.* files in kb: 126952
yb-cleanup Permitted disk usage for postgres*log* files in kb: 100000
yb-cleanup Disk usage by postgres*log* files in kb: 9235
yb-cleanup Permitted disk usage for core dump files in kb: 25691590
yb-cleanup Disk usage by core dump files in kb: 4
yb-cleanup Permitted disk usage for yb-tserver*log.* files in kb: 5000000
yb-cleanup Disk usage by yb-tserver*log.* files in kb: 126953
yb-tserver bash: chronyc: command not found
yb-cleanup Permitted disk usage for postgres*log* files in kb: 100000
yb-cleanup Disk usage by postgres*log* files in kb: 9314
yb-tserver bash: chronyc: command not found
yb-tserver Fatal failure details written to /mnt/disk0/yb-data/tserver/logs/yb-tserver.FATAL.details.2024-04-16T18_23_18.pid28.txt
yb-tserver F20240416 18:23:18 ./src/yb/server/hybrid_clock.cc:177] Too big clock skew is detected: 0.506s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usecs=60000000
yb-tserver    @ 0x55e8dc4c46377 google::LogMessage::SendToLog()
yb-tserver    @ 0x55e8dc4c472dd google::LogMessage::Flush()
yb-tserver    @ 0x55e8dc4c47959 google::LogMessageFatal::LogMessageFatal()
yb-tserver    @ 0x55e8dde9756c yb::server::HybridClock::NowWithError()
yb-tserver    @ 0x55e8dde95601 yb::server::HybridClock::NowRange()
yb-tserver    @ 0x55e8ddfb8e39 yb::tablet::TransactionCoordinator::Impl::Poll()
yb-tserver    @ 0x55e8dddaad3c yb::rpc::Poller::Poll()
yb-tserver    @ 0x55e8ddda566 _ZNboost4astddetail18completion_handlerIZN2yb3rpc95scheduler4Impl11HandleTimerERKN5_6system10error_codeEEUlVE_NS0_10lo_context19basic_executor_typeINSt3__19allocat
yb-tserver    @ 0x55e8ddda566 boost::astddetail::scheduler::Run()
yb-tserver    @ 0x55e8dddf92f57 yb::rpc::IOThreadPool::Impl::Execute()
yb-tserver    @ 0x55e8ddde51acba yb::Thread::SuperviseThread()
yb-tserver    @ 0x7f1de2471694 start_thread
yb-tserver    @ 0x7f1de27fe41d __clone

```

Abbildung XLV: YugabyteDB - Too big clock skew is detected

Eine Folge ist, dass kein neuer Leader bestimmt werden kann:

localhost:7000/table?id=00004000000030008000000000000400b						
Priority	7	string NULLABLE VALUE				
mail	8	string NULLABLE VALUE				
Tablet ID	Partition	SplitDepth	State	Hidden	Message	RaftConfig
a2c249ed8ebc4f068c00735c9acc40c5	hash_split: [0xAAAA, 0xFFFF]	0	Running	0	Tablet reported with an active leader	<ul style="list-style-type: none"> LEADER: yb-tserver-1.yb-tservers.yb-platform.svc.cluster.local (HAS_LEASE) Remaining ht_lease (may be stale): -148310 ms UUID: 015d6a1c121c4648879d2dd2f6f7747c FOLLOWER: yb-tserver-2.yb-tservers.yb-platform.svc.cluster.local UUID: d44aa240148c4e15a0684e4ac6dc9a9d
ada5becc50e948588337990cf8f61bf8	hash_split: [0x5555, 0xAAA9]	0	Running	0	Tablet reported with an active leader	<ul style="list-style-type: none"> FOLLOWER: yb-tserver-1.yb-tservers.yb-platform.svc.cluster.local UUID: 015d6a1c121c4648879d2dd2f6f7747c LEADER: yb-tserver-2.yb-tservers.yb-platform.svc.cluster.local (NO_MAJORITY_REPLICATEDLEASE) Cannot replicate lease for past 4 heartbeats UUID: d44aa240148c4e15a0684e4ac6dc9a9d
a6609fe063354432bc38b435fe1413d	hash_split: [0x0000, 0x5554]	0	Running	0	Tablet reported with an active leader	<ul style="list-style-type: none"> FOLLOWER: yb-tserver-1.yb-tservers.yb-platform.svc.cluster.local UUID: 015d6a1c121c4648879d2dd2f6f7747c LEADER: yb-tserver-2.yb-tservers.yb-platform.svc.cluster.local (NO_MAJORITY_REPLICATEDLEASE) Cannot replicate lease for past 29 heartbeats UUID: d44aa240148c4e15a0684e4ac6dc9a9d

Abbildung XLVI: YugabyteDB - Tablet Leader - No Lease

Als nächstes wird der komplette tserver in einem CrashLoopBackOff fallen:

```

Context: default
Cluster: default
User: default
K9s Rev: v0.31.8 ⚡ v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU: 1%
MEM: 36%

```

NAME	PF	READY	STATUS	RESTARTS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE	AGE
yb-master-0	●	2/2	Running	0	11	238	0	n/a	23	n/a	192.168.1.60	sk51184	34h
yb-master-1	●	2/2	Running	0	5	194	0	n/a	15	n/a	192.18.0.31	sk51183	34h
yb-master-2	●	2/2	Running	0	4	210	0	n/a	20	n/a	192.18.2.180	sk51185	7m54s
yb-tserver-0	●	2/2	Running	0	8	1101	0	n/a	8	n/a	192.18.2.207	sk51185	7m54s
yb-tserver-1	●	1/2	CrashLoopBackOff	5	1	1	0	n/a	0	n/a	192.18.1.77	sk51184	34h
yb-tserver-2	●	2/2	Running	0	27	7591	1	n/a	61	n/a	192.18.0.136	sk51183	34h

Abbildung XLVII: YugabyteDB - CrashLoopBackOff

Der ganze Cluster an sich aber bleibt arbeitsfähig.

Anders sieht es aus, wenn auch tmaster-Nodes von Start weg betroffen sind.

Es werden aber primär nur die Logs überall geschrieben:

X..I Ansible - Installation

Auf sks1000 wurde zuerst Ansible installiert:

```
1 sudo apt update && sudo apt install -y python3-pip sshpass git  
2 pip3 install ansible  
3
```

Listing 125: Ansible - Installation

Das GitHub Repository von vitabacks / postgresql_cluster muss lokal geklont werden:

```
1 git clone https://github.com/vitabaks/postgresql_cluster.git  
2
```

Listing 126: Ansible - Repository Clone

Das Verzeichnis ist postgresql_cluster beinhaltet die Playbooks und Ressourcen.

```
1 cd postgresql_cluster/  
2
```

Listing 127: Ansible - cd Repository

Die wichtigen Files liegen nun an folgenden Verzeichnissen:

- **inventory:** postgresql_cluster/inventory
- **main.yml:** postgresql_cluster/vars/main.yml
- **inventory:** postgresql_cluster/deploy_pgcluster.yml
- **inventory:** postgresql_cluster/config_pgcluster.yml
- **inventory:** postgresql_cluster/add_balancer.yml
- **inventory:** postgresql_cluster/add_pgnode.yml

X.I Prequenteries

Auf allen Hosts müssen die Firewalls angepasst werden:

```
1 # nano /etc/iptables/rules.v4  
2 # Generated by iptables-save v1.8.9 (nf_tables)  
3 *filter  
4 :INPUT ACCEPT [0:0]  
5 :FORWARD ACCEPT [0:0]  
6 :OUTPUT ACCEPT [0:0]  
7 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT  
8 -A INPUT -s 10.0.9.115/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for  
probe 10.0.9.115" -j ACCEPT  
9 -A INPUT -s 10.0.9.76/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for  
probe 10.0.9.76" -j ACCEPT  
10 -A INPUT -s 10.0.36.147/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for  
probe 10.0.36.147" -j ACCEPT  
11 -A INPUT -s 10.0.9.35/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for  
probe 10.0.9.35" -j ACCEPT  
12 -A INPUT -s 10.0.9.37/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for  
probe 10.0.9.37" -j ACCEPT
```

```

13 -A INPUT -s 10.0.9.74/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for
    probe 10.0.9.74" -j ACCEPT
14 -A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for
    probe 10.0.9.75" -j ACCEPT
15 -A INPUT -s 10.0.9.36/32 -p udp -m udp --dport 161 -m comment --comment "Allow SNMP for
    probe 10.0.9.36" -j ACCEPT-A INPUT -s 10.0.9.14/32 -p udp -m udp --dport 161 -m
    comment --comment "Allow SNMP for probe 10.0.9.14" -j ACCEPT
16 -A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
17 # generell
18 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 443 -j ACCEPT
19 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8080 -j ACCEPT
20 # postgres
21 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
22 # pgbounder
23 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6432 -j ACCEPT
24 # etcd
25 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2379 -j ACCEPT
26 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2380 -j ACCEPT
27 # patroni
28 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8008 -j ACCEPT
29 # haproxy
30 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5000 -j ACCEPT
31 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5001 -j ACCEPT
32 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5002 -j ACCEPT
33 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5003 -j ACCEPT
34 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 7000 -j ACCEPT
35 COMMIT
36 # Completed
37 # systemctl restart iptables
38

```

Listing 128: Testsystem - Firewall Settings

Als nächstes muss der Proxy gesetzt werden:

```

1 # nano /etc/profile.d/proxy.sh
2 export http_proxy="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first
   -it.ch:9090";
3 export https_proxy="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.
   first-it.ch:9090";
4 export no_proxy="127.0.0.1,localhost,10.0.0.0/8,.ksgr.ch,.sivc.first-it.ch"
5
6 export HTTP_PROXY="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first
   -it.ch:9090";
7 export HTTPS_PROXY="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.
   first-it.ch:9090";
8 export NO_PROXY="127.0.0.1,localhost,10.0.0.0/8,.ksgr.ch,.sivc.first-it.ch"
9 # source /etc/profile.d/proxy.sh
10

```

Listing 129: Testsystem - Proxy Settings

Auch der apt-Proxy muss gesetzt werden:

```

1 # nano /etc/apt/apt.conf.d/proxy.conf
2 Acquire::http::Proxy "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.
   first-it.ch:9090";

```

```

3 Acquire::https::Proxy "http://xksgr_vks0041_inet<Password Secure / Safe>@webproxy.svc.
  first-it.ch:9090";
4 Acquire::http::proxy::foreman.ksgr.ch "DIRECT";

```

Listing 130: Testsystem - apt-Proxy Settings

X.II Deployment

Zuerst muss das Inventory angepasst werden:

```

1 # if dcs_exists: false and dcs_type: "etcd"
2 [etcd_cluster] # recommendation: 3, or 5-7 nodes
3 10.0.22.170
4 10.0.22.171
5 10.0.22.172
6
7 # if dcs_exists: false and dcs_type: "consul"
8 [consul_instances] # recommendation: 3 or 5-7 nodes
9
10 # if with_haproxy_load_balancing: true
11 [balancers]
12 10.0.22.176
13 10.0.22.177
14
15 # PostgreSQL nodes
16 [master]
17 10.0.22.173 postgresql_exists=false
18
19 [replica]
20 10.0.22.174 postgresql_exists=false
21 10.0.22.175 postgresql_exists=false
22
23 [postgres_cluster:children]
24 master
25 replica
26
27 # if pgbackrest_install: true and "repo_host" is set
28 [pgbackrest] # optional (Dedicated Repository Host)
29
30
31 # Connection settings
32 [all:vars]
33 ansible_connection='ssh'
34 ansible_ssh_port='22'
35 ansible_user='itgramic'
36 ansible_ssh_pass='<Secret ;->' # "sshpass" package is required for use "
  ansible_ssh_pass"
37 #ansible_ssh_private_key_file=
38 ansible_python_interpreter='/usr/bin/env python3'
39
40 [pgbackrest:vars]
41

```

Listing 131: Testsystem - Deployment - inventory

Nun muss das main.yml dran. Wichtig sind folgende Einstellungen.

```
1--  
2#-----  
3# Proxy variables (optional) for download packages using a proxy server  
4proxy_env: {} # yamllint disable rule:braces  
5#-----  
6#proxy_env:  
7#   http_proxy: "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it  
     .ch:9090"  
8#   https_proxy: "https://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-  
   it.ch:9090"  
9  
10# Cluster variables  
11cluster_vip: "" # IP address for client access to the databases in the cluster (optional).  
12cluster_vip: "10.0.22.178" # IP address for client access to the databases in the cluster (optional).  
13vip_interface: "{{ ansible_default_ipv4.interface }}" # interface name (e.g., "ens32").  
14# Note: VIP-based solutions such as keepalived or vip-manager may not function correctly  
#       in cloud environments like AWS.  
15  
16#patroni_cluster_name: "postgres-cluster" # the cluster name (must be unique for each cluster)  
17patroni_cluster_name: "k8s-core-psql" # the cluster name (must be unique for each cluster)  
18patroni_install_version: "3.3.0" # or 'latest'  
19  
20patroni_superuser_username: "postgres"  
21patroni_superuser_password: "<Password Secure / Safe>"  
22patroni_replication_username: "replicator"  
23patroni_replication_password: "<Password Secure / Safe>"  
24  
25#synchronous_mode: false # or 'true' for enable synchronous database replication  
26synchronous_mode: true # or 'true' for enable synchronous database replication  
27synchronous_mode_strict: false # if 'true' then block all client writes to the master,  
#                               when a synchronous replica is not available  
28#synchronous_node_count: 1 # number of synchronous standby databases  
29synchronous_node_count: 2 # number of synchronous standby databases  
30  
31# Load Balancing  
32with_haproxy_load_balancing: true # or 'true' if you want to install and configure the  
#                                 load-balancing  
33haproxy_listen_port:  
34  master: 5000  
35  replicas: 5001  
36  replicas_sync: 5002  
37  replicas_async: 5003  
38# The following ('_direct') ports are used for direct connections to the PostgreSQL  
#       database,  
39# bypassing the PgBouncer connection pool (if 'pgbouncer_install' is 'true').  
40# Uncomment the relevant lines if you need to set up direct connections.  
41  stats: 7000  
42haproxy_maxconn:
```

```

43 global: 100000
44 master: 10000
45 replica: 10000
46
47 haproxy_timeout:
48   client: "60m"
49   server: "60m"
50
51 # keepalived (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'true')
52 keepalived_virtual_router_id: "{{ cluster_vip.split('.')[3] | int }}" # The last octet of
      'cluster_vip' IP address is used by default.
53
54 # virtual_router_id - must be unique in the network (available values are 0..255).
55
56 # vip-manager (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'false'
57   ')
58
59 vip_manager_version: "2.4.0" # version to install
60 vip_manager_conf: "/etc/patroni/vip-manager.yml"
61 vip_manager_interval: "1000" # time (in milliseconds) after which vip-manager wakes up
      and checks if it needs to register or release ip addresses.
62
63 vip_manager_iface: "{{ vip_interface }}" # interface to which the virtual ip will be
      added
64
65 vip_manager_ip: "{{ cluster_vip }}" # the virtual ip address to manage
66 vip_manager_mask: "24" # netmask for the virtual ip
67
68
69 # DCS (Distributed Consensus Store)
70 dcs_exists: false # or 'true' if you don't want to deploy a new etcd cluster
71 dcs_type: "etcd" # or 'consul'
72
73 # if dcs_type: "etcd" and dcs_exists: false
74 etcd_version: "3.5.11" # version for deploy etcd cluster
75 etcd_data_dir: "/var/lib/etcd"
76 etcd_cluster_name: "etcd-{{ patroni_cluster_name }}" # ETCD_INITIAL_CLUSTER_TOKEN
77
78 # if dcs_type: "etcd" and dcs_exists: true
79 patroni_etcd_hosts: [] # list of servers of an existing etcd cluster
80
81 patroni_etcd_namespace: "service" # (optional) etcd namespace (prefix)
82 patroni_etcd_username: "" # (optional) username for etcd authentication
83 patroni_etcd_password: "" # (optional) password for etcd authentication
84 patroni_etcd_protocol: "" # (optional) http or https, if not specified http is used
85
86 # if dcs_type: "consul"
87 consul_version: "1.15.8"
88 consul_config_path: "/etc/consul"
89 consul_configd_path: "{{ consul_config_path }}/conf.d"
90 consul_data_path: "/var/lib/consul"
91
92 consul_domain: "consul" # Consul domain name
93 consul_datacenter: "dc1" # Datacenter label (can be specified for each host in the
      inventory)
94
95 consul_disable_update_check: true # Disables automatic checking for security bulletins
      and new version releases
96
97 consul_enable_script_checks: true # This controls whether health checks that execute
      scripts are enabled on this agent
98
99 consul_enable_local_script_checks: true # Enable them when they are defined in the local

```

```

        configuration files
90 consul_ui: false # Enable the consul UI?
91 consul_syslog_enable: true # Enable logging to syslog
92 consul_iface: "{{ ansible_default_ipv4.interface }}" # specify the interface name with a
    Private IP (ex. "enp7s0")
93 # TLS
94 # You can enable TLS encryption by dropping a CA certificate, server certificate, and
    server key in roles/consul/files/
95 consul_tls_enable: false
96 consul_tls_ca_crt: "ca.crt"
97 consul_tls_server_crt: "server.crt"
98 consul_tls_server_key: "server.key"
99 # DNS
100 consul_recursors: [] # List of upstream DNS servers
101 consul_dnsmasq_enable: true # Enable DNS forwarding with Dnsmasq
102 consul_dnsmasq_cache: 0 # dnsmasq cache-size (0 - disable caching)
103 consul_dnsmasq_servers: "{{ nameservers }}" # Upstream DNS servers used by dnsmasq
104 consul_join: [] # List of LAN servers of an existing consul cluster, to join.
105
106 # https://developer.hashicorp.com/consul/docs/discovery/services
107 consul_services:
108   - name: "{{ patroni_cluster_name }}"
109     id: "{{ patroni_cluster_name }}-master"
110     tags: ['master', 'primary']
111     port: "{{ pgbouncer_listen_port }}" # or "{{ postgresql_port }}" if
112     pgbouncer_install: false
113     checks:
114       - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/primary",
115         interval: "2s" }
116       - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" } # comment out
117         this check if pgbouncer_install: false
118   - name: "{{ patroni_cluster_name }}"
119     id: "{{ patroni_cluster_name }}-replica"
120     tags: ['replica']
121     port: "{{ pgbouncer_listen_port }}"
122     checks:
123       - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/replica?lag
124         ={{ patroni_maximum_lag_on_replica }}", interval: "2s" }
125       - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }

# PostgreSQL variables
126 postgresql_version: "16"
127 # postgresql_data_dir: see vars/Debian.yml or vars/RedHat.yml
128 postgresql_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{
129   inventory_hostname }},127.0.0.1" to listen on a specific IP address.
130 postgresql_port: "5432"
131 postgresql_encoding: "UTF8" # for bootstrap only (initdb)
132 postgresql_locale: "en_US.UTF-8" # for bootstrap only (initdb)
133 postgresql_data_checksums: true # for bootstrap only (initdb)
134 postgresql_password_encryption_algorithm: "scram-sha-256" # or "md5" if your clients do
    not work with passwords encrypted with SCRAM-SHA-256

# (optional) list of users to be created (if not already exists)
135 postgresql_users:

```

```

135 - { name: "{{ pgbouncer_auth_username }}", password: "{{ pgbouncer_auth_password }}",
136   flags: "LOGIN", role: "" }
137 - { name: "xksgr_sks1160_gitlab", password: "<Password Secure / Safe>", flags: "
138   SUPERUSER" }
139 - { name: "xksgr_sks1172_harbor", password: "<Password Secure / Safe>", flags: "
140   SUPERUSER" }
141 - { name: "xksgr_sks1195_kcsso", password: "<Password Secure / Safe>", flags: "SUPERUSER"
142   " }
143 - { name: "xksgr_k8s_core_psql_monitor", password: "<Password Secure / Safe>", flags: "
144   LOGIN", role: "pg_monitor" }
145 - { name: "xksgr_k8s_core_psql_backup", password: "<Password Secure / Safe>", flags: "
146   SUPERUSER" }
147
148 # (optional) list of databases to be created (if not already exists)
149 #postgresql_databases: []
150 postgresql_databases:
151   - { db: "k8s_core_gitlab_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "
152     en_US.UTF-8", owner: "xksgr_sks1160_gitlab" }
153   - { db: "k8s_core_harbor_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "
154     en_US.UTF-8", owner: "xksgr_sks1172_harbor" }
155   - { db: "k8s_core_keycloak_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype:
156     "en_US.UTF-8", owner: "xksgr_sks1195_kcsso" }
157   - { db: "gramic_test", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF
158     -8", owner: "postgres" }
159
160 # (optional) list of schemas to be created (if not already exists)
161 postgresql_schemas: []
162   - { schema: "myschema", db: "mydatabase", owner: "mydb-user" }
163
164 # (optional) list of database extensions to be created (if not already exists)
165 postgresql_extensions: []
166 postgresql_extensions:
167   - { ext: "pgstattuple", db: "postgres" }
168   - { ext: "pgstattuple", db: "k8s_core_gitlab_prod" }
169   - { ext: "pgstattuple", db: "k8s_core_harbor_prod" }
170   - { ext: "pgstattuple", db: "k8s_core_keycloak_prod" }
171   - { ext: "pgstattuple", db: "gramic_test" }
172
173 # postgresql parameters to bootstrap dcs (are parameters for example)
174 postgresql_parameters:
175   - { option: "max_connections", value: "500" }
176   - { option: "superuser_reserved_connections", value: "5" }
177   - { option: "password_encryption", value: "{{ postgresql_password_encryption_algorithm
178     }}" }
179   - { option: "max_locks_per_transaction", value: "512" }
180   - { option: "max_prepared_transactions", value: "0" }
181   - { option: "huge_pages", value: "try" } # or "on" if you set "vm_nr_hugepages" in
182     kernel parameters
183   - { option: "shared_buffers", value: "{{ (ansible_memtotal_mb * 0.25) | int }}MB" } #
184     by default, 25% of RAM
185   - { option: "effective_cache_size", value: "{{ (ansible_memtotal_mb * 0.75) | int }}MB"
186     } # by default, 75% of RAM
187   - { option: "work_mem", value: "128MB" } # please change this value
188   - { option: "maintenance_work_mem", value: "256MB" } # please change this value

```

```

175 - { option: "checkpoint_timeout", value: "15min" }
176 - { option: "checkpoint_completion_target", value: "0.9" }
177 - { option: "min_wal_size", value: "2GB" }
178 - { option: "max_wal_size", value: "8GB" } # or 16GB/32GB
179 - { option: "wal_buffers", value: "32MB" }
180 - { option: "default_statistics_target", value: "1000" }
181 - { option: "seq_page_cost", value: "1" }
182 - { option: "random_page_cost", value: "1.1" } # or "4" for HDDs with slower random
   access
183 - { option: "effective_io_concurrency", value: "200" } # or "2" for traditional HDDs
   with lower I/O parallelism
184 - { option: "synchronous_commit", value: "on" } # or 'off' if you can you lose single
   transactions in case of a crash
185 - { option: "autovacuum", value: "on" } # never turn off the autovacuum!
186 - { option: "autovacuum_max_workers", value: "5" }
187 - { option: "autovacuum_vacuum_scale_factor", value: "0.01" } # or 0.005/0.001
188 - { option: "autovacuum_analyze_scale_factor", value: "0.01" }
189 - { option: "autovacuum_vacuum_cost_limit", value: "500" } # or 1000/5000
190 - { option: "autovacuum_vacuum_cost_delay", value: "2" }
191 - { option: "autovacuum_naptime", value: "1s" }
192 - { option: "max_files_per_process", value: "4096" }
193 - { option: "archive_mode", value: "on" }
194 - { option: "archive_timeout", value: "1800s" }
195 - { option: "archive_command", value: "cd ." } # not doing anything yet with WAL-s
196 - { option: "wal_level", value: "logical" }
197 - { option: "wal_keep_size", value: "2GB" }
198 - { option: "max_wal_senders", value: "10" }
199 - { option: "max_replication_slots", value: "10" }
200 - { option: "hot_standby", value: "on" }
201 - { option: "wal_log_hints", value: "on" }
202 - { option: "wal_compression", value: "on" }
203 - { option: "shared_preload_libraries", value: "pg_stat_statements,auto_explain" }
204 - { option: "pg_stat_statements.max", value: "10000" }
205 - { option: "pg_stat_statements.track", value: "all" }
206 - { option: "pg_stat_statements.track_utility", value: "false" }
207 - { option: "pg_stat_statements.save", value: "true" }
208 - { option: "auto_explain.log_min_duration", value: "10s" } # enable auto_explain for
   10-second logging threshold. Decrease this value if necessary
209 - { option: "auto_explain.log_analyze", value: "true" }
210 - { option: "auto_explain.log_buffers", value: "true" }
211 - { option: "auto_explain.log_timing", value: "false" }
212 - { option: "auto_explain.log_triggers", value: "true" }
213 - { option: "auto_explain.log_verbose", value: "true" }
214 - { option: "auto_explain.log_nested_statements", value: "true" }
215 - { option: "auto_explain.sample_rate", value: "0.01" } # enable auto_explain for 1%
   of queries logging threshold
216 - { option: "track_io_timing", value: "on" }
217 - { option: "log_lock_waits", value: "on" }
218 - { option: "log_temp_files", value: "0" }
219 - { option: "track_activities", value: "on" }
220 - { option: "track_activity_query_size", value: "4096" }
221 - { option: "track_counts", value: "on" }
222 - { option: "track_functions", value: "all" }
223 - { option: "log_checkpoints", value: "on" }

```

```

224 - { option: "logging_collector", value: "on" }
225 - { option: "log_truncate_on_rotation", value: "on" }
226 - { option: "log_rotation_age", value: "1d" }
227 - { option: "log_rotation_size", value: "0" }
228 - { option: "log_line_prefix", value: "'%t [%p-%l] %r %q%u@%d '" }
229 - { option: "log_filename", value: "postgresql-%a.log" }
230 - { option: "log_directory", value: "{{ postgresql_log_dir }}" }
231 - { option: "hot_standby_feedback", value: "on" } # allows feedback from a hot standby
   to the primary that will avoid query conflicts
232 - { option: "max_standby_streaming_delay", value: "30s" }
233 - { option: "wal_receiver_status_interval", value: "10s" }
234 - { option: "idle_in_transaction_session_timeout", value: "10min" } # reduce this
   timeout if possible
235 - { option: "jit", value: "off" }
236 - { option: "max_worker_processes", value: "24" }
237 - { option: "max_parallel_workers", value: "8" }
238 - { option: "max_parallel_workers_per_gather", value: "2" }
239 - { option: "max_parallel_maintenance_workers", value: "2" }
240 - { option: "tcp_keepalives_count", value: "10" }
241 - { option: "tcp_keepalives_idle", value: "300" }
242 - { option: "tcp_keepalives_interval", value: "30" }

243
244 # Set this variable to 'true' if you want the cluster to be automatically restarted
245 # after changing the 'postgresql_parameters' variable that requires a restart in the '
   config_pgcluster.yml' playbook.
246 # By default, the cluster will not be automatically restarted.
247 pending_restart: false
248
249 # specify additional hosts that will be added to the pg_hba.conf
250 postgresql_pg_hba:
251 - { type: "local", database: "all", user: "{{ patroni_superuser_username }}", address: "",
   method: "trust" }
252 - { type: "local", database: "all", user: "{{ pgbouncer_auth_username }}", address: "",
   method: "trust" } # required for pgbouncer auth_user
253 - { type: "local", database: "replication", user: "all", address: "", method: "trust" }
254 - { type: "host", database: "replication", user: "all", address: "127.0.0.1/32", method:
   "trust" }
255 - { type: "host", database: "replication", user: "all", address: "::1/128", method: "trust" }
256 - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
   address: "10.0.22.173/24", method: "{{ postgresql_password_encryption_algorithm }}" }
257 - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
   address: "10.0.22.174/24", method: "{{ postgresql_password_encryption_algorithm }}" }
258 - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
   address: "10.0.22.175/24", method: "{{ postgresql_password_encryption_algorithm }}" }
259 - { type: "local", database: "all", user: "all", address: "", method: "{{ postgresql_password_encryption_algorithm }}" }
260 - { type: "host", database: "all", user: "all", address: "127.0.0.1/32", method: "{{ postgresql_password_encryption_algorithm }}" }
261 - { type: "host", database: "all", user: "all", address: "::1/128", method: "{{ postgresql_password_encryption_algorithm }}" }
262 - { type: "host", database: "k8s_core_gitlab_prod", user: "xksgr_sks1160_gitlab",
   address: "10.0.20.88/24", method: "{{ postgresql_password_encryption_algorithm }}" }
263 - { type: "host", database: "k8s_core_harbor_prod", user: "xksgr_sks1172_harbor",

```

```

address: "10.0.20.92/24", method: "{{ postgresql_password_encryption_algorithm }}" }
264 - { type: "host", database: "k8s_core_keycloak_prod", user: "xksgr_sks1195_kcsso",
  address: "10.0.20.98/24", method: "{{ postgresql_password_encryption_algorithm }}" }
265 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.63/24",
  method: "{{ postgresql_password_encryption_algorithm }}" }
266 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.43/24",
  method: "{{ postgresql_password_encryption_algorithm }}" }
267 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.77/24",
  method: "{{ postgresql_password_encryption_algorithm }}" }

268
269 # list of lines that Patroni will use to generate pg_ident.conf
270 postgresql_pg_ident: []
271
272 # the password file (~/.pgpass)
273 postgresql_pgpass:
274   - "localhost:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
275   - "{{ inventory_hostname }}:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
276   - "*:{{ pgbouncer_listen_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
277   - "*:{{ postgresql_port }}:*:{{ patroni_replication_username }}:{{ patroni_replication_password }}"
278   - "10.0.20.88:5432:k8s_core_gitlab_prod:xksgr_sks1160_gitlab:<Password Secure / Safe>"
279   - "10.0.20.92:5432:k8s_core_harbor_prod:xksgr_sks1172_harbor:j<Password Secure / Safe>"
280   - "10.0.20.98:5432:k8s_core_keycloak_prod:xksgr_sks1195_kcsso:<Password Secure / Safe>"

281
282 # PgBouncer parameters
283 pgbouncer_install: true # or 'false' if you do not want to install and configure the
  pgbouncer service
284 pgbouncer_processes: 1 # Number of pgbouncer processes to be used. Multiple processes
  use the so_REUSEPORT option for better performance.
285 pgbouncer_conf_dir: "/etc/pgbouncer"
286 pgbouncer_log_dir: "/var/log/pgbouncer"
287 pgbouncer_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}"
  to listen on a specific IP address.
288 pgbouncer_listen_port: 6432
289 pgbouncer_max_client_conn: 10000
290 pgbouncer_max_db_connections: 1000
291 pgbouncer_max_prepared_statements: 1024
292 pgbouncer_default_pool_size: 20
293 pgbouncer_query_timeout: 120
294 pgbouncer_default_pool_mode: "session"
295 pgbouncer_admin_users: "{{ patroni_superuser_username }}" # comma-separated list of
  users, who are allowed to change settings
296 pgbouncer_stats_users: "{{ patroni_superuser_username }}" # comma-separated list of
  users who are just allowed to use SHOW command
297 pgbouncer_ignore_startup_parameters: "extra_float_digits,geqo,search_path"
298 pgbouncer_auth_type: "{{ postgresql_password_encryption_algorithm }}"
299 pgbouncer_auth_user: true # or 'false' if you want to manage the list of users for
  authentication in the database via userlist.txt
300 pgbouncer_auth_username: pgbouncer # user who can query the database via the user_search
  function
301 pgbouncer_auth_password: "<Password Secure / Safe>"
```

```

302 pgbouncer_auth_dbname: "postgres"
303 pgbouncer_client_tls_sslmode: "disable"
304 pgbouncer_client_tls_key_file: ""
305 pgbouncer_client_tls_cert_file: ""
306 pgbouncer_client_tls_ca_file: ""
307 pgbouncer_client_tls_protocols: "secure" # allowed values: tlsv1.0, tlsv1.1, tlsv1.2,
     tlsv1.3, all, secure (tlsv1.2,tlsv1.3)
308 pgbouncer_client_tls_ciphers: "default" # allowed values: default, secure, fast, normal,
     all (not recommended)
309
310 pgbouncer_pools:
311   - { name: "postgres", dbname: "postgres", pool_parameters: "" }
312   - { name: "k8s_core_gitlab_prod", dbname: "k8s_core_gitlab_prod", pool_parameters: "" }
313   - { name: "k8s_core_harbor_prod", dbname: "k8s_core_harbor_prod", pool_parameters: "" }
314   - { name: "k8s_core_keycloak_prod", dbname: "k8s_core_keycloak_prod", pool_parameters: ""
     }
315   - { name: "gramic_test", dbname: "gramic_test", pool_parameters: "" }
316
317 # Extended variables (optional)
318 patroni_restapi_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{  

     inventory_hostname }}" to listen on a specific IP address.
319 patroni_restapi_port: 8008
320 patroni_ttl: 30
321 patroni_loop_wait: 10
322 patroni_retry_timeout: 10
323 patroni_master_start_timeout: 300
324 patroni_maximum_lag_on_failover: 1048576 # (1MB) the maximum bytes a follower may lag to  

     be able to participate in leader election.
325 patroni_maximum_lag_on_replica: "100MB" # the maximum of lag that replica can be in order  

     to be available for read-only queries.
326
327 # https://patroni.readthedocs.io/en/latest/yaml_configuration.html#postgresql
328 patroni_callbacks: []
329
330 # https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#standby-cluster
331 # Requirements:
332 # 1. the cluster name for Standby Cluster must be unique ('patroni_cluster_name' variable
     )
333 # 2. the IP addresses (or network) of the Standby Cluster servers must be added to the  

     pg_hba.conf of the Main Cluster ('postgresql_pg_hba' variable).
334 patroni_standby_cluster:
335   host: "" # an address of remote master
336   port: "5432" # a port of remote master
337
338 # Permanent replication slots.
339 # These slots will be preserved during switchover/failover.
340 # https://patroni.readthedocs.io/en/latest/dynamic_configuration.html
341 patroni_slots: []
342
343 patroni_log_destination: stderr # or 'logfile'
344 # if patroni_log_destination: logfile
345 patroni_log_dir: /var/log/patroni
346 patroni_log_level: info
347 patroni_log_traceback_level: error

```

```

348 patroni_log_format: "%(asctime)s %(levelname)s: %(message)s"
349 patroni_log_dateformat: ""
350 patroni_log_max_queue_size: 1000
351 patroni_log_file_num: 4
352 patroni_log_file_size: 25000000 # bytes
353 patroni_log_loggers_patroni_postmaster: warning
354 patroni_log_loggers_urllib3: warning # or 'debug'
355
356 patroni_watchdog_mode: automatic # or 'off', 'required'
357 patroni_watchdog_device: /dev/watchdog
358
359 patroni_postgresql_use_pg_rewind: true # or 'false'
360 # try to use pg_rewind on the former leader when it joins cluster as a replica.
361
362 patroni_remove_data_directory_on_rewind_failure: false # or 'true' (if use_pg_rewind: 'true')
363 # avoid removing the data directory on an unsuccessful rewind
364 # if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica.
365
366 patroni_remove_data_directory_on_diverged_timelines: false # or 'true'
367 # if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica
368 # if it notices that timelines are diverging and the former master can not start
     streaming from the new master.
369
370 # https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#bootstrap
371 patroni_cluster_bootstrap_method: "initdb" # or "wal-g", "pgbackrest", "pg_probackup"
372
373 # https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#building-replicas
374 patroni_create_replica_methods:
375   - basebackup
376
377 pgbackrest:
378   - { option: "command", value: "/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore" }
379   - { option: "keep_data", value: "True" }
380   - { option: "no_params", value: "True" }
381 wal_g:
382   - { option: "command", value: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST" }
383   - { option: "no_params", value: "True" }
384 basebackup:
385   - { option: "max-rate", value: "100M" }
386   - { option: "checkpoint", value: "fast" }
387 #   - { option: "waldir", value: "{{ postgresql_wal_dir }}" }
388 pg_probackup:
389   - { option: "command", value: "{{ pg_probackup_restore_command }}" }
390   - { option: "no_params", value: "true" }
391
392 # "restore_command" written to recovery.conf when configuring follower (create replica)
393 postgresql_restore_command: ""
394
395 # pg_probackup
396 pg_probackup_install: false # or 'true'
397 pg_probackup_install_from_postgrespro_repo: true # or 'false',
398 pg_probackup_version: "{{ postgresql_version }}"

```

```

399 pg_probackup_instance: "pg_probackup_instance_name"
400 pg_probackup_dir: "/mnt/backup_dir"
401 pg_probackup_threads: "4"
402 pg_probackup_add_keys: "--recovery-target=latest --skip-external-dirs --no-validate"
403 # Ensure there is a space at the beginning of each part to prevent commands from
404   concatenating.
405 pg_probackup_command_parts:
406   - "pg_probackup-{{ pg_probackup_version }}"
407   - " restore -B {{ pg_probackup_dir }}"
408   - " --instance {{ pg_probackup_instance }}"
409   - " -j {{ pg_probackup_threads }}"
410   - " {{ pg_probackup_add_keys }}"
411 pg_probackup_restore_command: "{{ pg_probackup_command_parts | join('') }}"
412 pg_probackup_patroni_cluster_bootstrap_command: "{{ pg_probackup_command_parts | join('') }}"
413
414 # WAL-G
415 wal_g_install: false # or 'true'
416 wal_g_version: "3.0.0"
417 wal_g_json: # config https://github.com/wal-g/wal-g#configuration
418   - { option: "AWS_ACCESS_KEY_ID", value: "{{ AWS_ACCESS_KEY_ID | default('') }}" } # define values or pass via --extra-vars
419   - { option: "AWS_SECRET_ACCESS_KEY", value: "{{ AWS_SECRET_ACCESS_KEY | default('') }}" } # define values or pass via --extra-vars
420   - { option: "WALG_S3_PREFIX", value: "{{ WALG_S3_PREFIX | default('') }}" } # define values or pass via --extra-vars
421   - { option: "WALG_COMPRESSION_METHOD", value: "brotli" } # or "lz4", "lzma", "zstd"
422   - { option: "WALG_DELTA_MAX_STEPS", value: "6" } # determines how many delta backups can be between full backups
423   - { option: "PGDATA", value: "{{ postgresql_data_dir }}" }
424   - { option: "PGHOST", value: "{{ postgresql_unix_socket_dir }}" }
425   - { option: "PGPORT", value: "{{ postgresql_port }}" }
426   - { option: "PGUSER", value: "{{ patroni_superuser_username }}" }
427   - { option: "AWS_S3_FORCE_PATH_STYLE", value: "true" } # to use Minio.io S3-compatible storage
428   - { option: "AWS_ENDPOINT", value: "http://minio:9000" } # to use Minio.io S3-compatible storage
429   - { option: "", value: "" }
430 wal_g_archive_command: "wal-g wal-push %p"
431 wal_g_patroni_cluster_bootstrap_command: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST"
432
433 # Define job_parts outside of wal_g_cron_jobs
434 # Ensure there is a space at the beginning of each part to prevent commands from
435   concatenating.
436 wal_g_backup_command:
437   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
438   - " && wal-g backup-push {{ postgresql_data_dir }} > {{ postgresql_log_dir }}/walg_backup.log 2>&1"
439 wal_g_delete_command:
440   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
441   - " && wal-g delete retain FULL 4 --confirm > {{ postgresql_log_dir }}/walg_delete.log"

```

```

2>&1"

440
441 wal_g_cron_jobs:
442   - name: "WAL-G: Create daily backup"
443     user: "postgres"
444     file: /etc/cron.d/walg
445     minute: "30"
446     hour: "3"
447     day: "*"
448     month: "*"
449     weekday: "*"
450     job: "{{ wal_g_backup_command | join('') }}"
451   - name: "WAL-G: Delete old backups" # retain 4 full backups (adjust according to your
452     company's backup retention policy)
453     user: "postgres"
454     file: /etc/cron.d/walg
455     minute: "30"
456     hour: "6"
457     day: "*"
458     month: "*"
459     weekday: "*"
460     job: "{{ wal_g_delete_command | join('') }}"
461
462 # pgBackRest
463 pgbackrest_install: false # or 'true'
464 pgbackrest_install_from_pgdg_repo: false # or 'false'
465 pgbackrest_stanza: "{{ patroni_cluster_name }}" # specify your --stanza
466 pgbackrest_repo_type: "posix" # or "s3", "gcs", "azure"
467 pgbackrest_repo_host: "" # dedicated repository host (optional)
468 pgbackrest_repo_user: "postgres"
469 pgbackrest_conf_file: "/etc/pgbackrest/pgbackrest.conf"
470 pgbackrest_conf:
471   global: # [global] section
472     - { option: "log-level-file", value: "detail" }
473     - { option: "log-path", value: "/var/log/pgbackrest" }
474     - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }} " }
475     - { option: "repo1-path", value: "/var/lib/pgbackrest" }
476     - { option: "repo1-retention-full", value: "4" }
477     - { option: "repo1-retention-archive", value: "4" }
478     - { option: "start-fast", value: "y" }
479     - { option: "stop-auto", value: "y" }
480     - { option: "resume", value: "n" }
481     - { option: "link-all", value: "y" }
482     - { option: "spool-path", value: "/var/spool/pgbackrest" }
483     - { option: "archive-async", value: "y" } # Enables asynchronous WAL archiving (
484       details: https://pgbackrest.org/user-guide.html#async-archiving)
485     - { option: "archive-get-queue-max", value: "1GiB" }
486   stanza: # [stanza_name] section
487     - { option: "process-max", value: "4" }
488     - { option: "log-level-console", value: "info" }
489     - { option: "recovery-option", value: "recovery_target_action=promote" }
490     - { option: "pg1-socket-path", value: "{{ postgresql_unix_socket_dir }} " }
491     - { option: "pg1-path", value: "{{ postgresql_data_dir }} " }

```

```

491 #     - { option: "", value: "" }
492 # (optional) dedicated backup server config (if "repo_host" is set)
493 pgbackrest_server_conf:
494   global:
495     - { option: "log-level-file", value: "detail" }
496     - { option: "log-level-console", value: "info" }
497     - { option: "log-path", value: "/var/log/pgbackrest" }
498     - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}"} }
499     - { option: "repo1-path", value: "/var/lib/pgbackrest" }
500     - { option: "repo1-retention-full", value: "4" }
501     - { option: "repo1-retention-archive", value: "4" }
502     - { option: "repo1-bundle", value: "y" }
503     - { option: "repo1-block", value: "y" }
504     - { option: "start-fast", value: "y" }
505     - { option: "stop-auto", value: "y" }
506     - { option: "resume", value: "n" }
507     - { option: "link-all", value: "y" }
508     - { option: "archive-check", value: "y" }
509     - { option: "archive-copy", value: "n" }
510     - { option: "backup-standby", value: "y" }
511 #     - { option: "", value: "" }
512 # the stanza section will be generated automatically
513
514 pgbackrest_archive_command: "pgbackrest --stanza={{ pgbackrest_stanza }} archive-push %p"
515
516 pgbackrest_patroni_cluster_restore_command:
517   '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore' # restore from
      latest backup
518 #   '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --type=time "--target=2020-06-01
      11:00:00+03" --delta restore' # Point-in-Time Recovery (example)
519
520 # By default, the cron jobs is created on the database server.
521 # If 'repo_host' is defined, the cron jobs will be created on the pgbackrest server.
522 pgbackrest_cron_jobs:
523   - name: "pgBackRest: Full Backup"
524     file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
525     user: "postgres"
526     minute: "30"
527     hour: "6"
528     day: "*"
529     month: "*"
530     weekday: "0"
531     job: "pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup"
532     # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --
      type=full --stanza={{ pgbackrest_stanza }} backup; fi"
533   - name: "pgBackRest: Diff Backup"
534     file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
535     user: "postgres"
536     minute: "30"
537     hour: "6"
538     day: "*"
539     month: "*"
540     weekday: "1-6"
541     job: "pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup"

```

```

542     # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --  
543     type=diff --stanza={{ pgbackrest_stanza }} backup; fi"  
544  
544 # PITR mode (if patroni_cluster_bootstrap_method: "pgbackrest" or "wal-g"):  
545 # 1) The database cluster directory will be cleaned (for "wal-g") or overwritten (for "  
546     pgbackrest" --delta restore).  
546 # 2) And also the patroni cluster "{{ patroni_cluster_name }}" will be removed from the  
547     DCS (if exist) before recovery.  
548  
548 disable_archive_command: true # or 'false' to not disable archive_command after restore  
549 keep_patroni_dynamic_json: true # or 'false' to remove patroni.dynamic.json after  
      restore (if exists)  
550  
551 # Netdata - https://github.com/netdata/netdata  
552 netdata_install: false # or 'true' for install Netdata on postgresql cluster nodes (with  
      kickstart.sh)  
553 netdata_install_options: "--stable-channel --disable-telemetry --dont-wait"  
554 netdata_conf:  
555   web_bind_to: "*"  
556   # https://learn.netdata.cloud/docs/store/change-metrics-storage  
557   memory_mode: "dbengine" # The long-term metrics storage with efficient RAM and disk  
      usage.  
558   page_cache_size: 64 # Determines the amount of RAM in MiB that is dedicated to caching  
      Netdata metric values.  
559   dbengine_disk_space: 1024 # Determines the amount of disk space in MiB that is  
      dedicated to storing Netdata metric values.  
560  
561 ...  
562  
563

```

Listing 132: Testsystem - Deployment - main.yml

Jetzt muss geprüft werden, ob die Hosts für Ansible erreichbar sind:

```

1 root@sk1000:~/postgresql_cluster# ansible all -m ping
2 10.0.22.176 | SUCCESS => {
3     "changed": false,
4     "ping": "pong"
5 }
6 10.0.22.177 | SUCCESS => {
7     "changed": false,
8     "ping": "pong"
9 }
10 10.0.22.172 | SUCCESS => {
11     "changed": false,
12     "ping": "pong"
13 }
14 10.0.22.170 | SUCCESS => {
15     "changed": false,
16     "ping": "pong"
17 }
18 10.0.22.171 | SUCCESS => {
19     "changed": false,
20     "ping": "pong"

```

```

21 }
22 10.0.22.175 | SUCCESS => {
23     "changed": false,
24     "ping": "pong"
25 }
26 10.0.22.173 | SUCCESS => {
27     "changed": false,
28     "ping": "pong"
29 }
30 10.0.22.174 | SUCCESS => {
31     "changed": false,
32     "ping": "pong"
33 }

```

Listing 133: Deploy - Anhang - Ansible Ping

Deployt wird wie folgt:

```

1 ansible-playbook deploy_pgcluster.yml
2

```

Listing 134: Deploy - Anhang - deploy_pgcluster.yml

Nach dem Deployment sollte folgende Ausgabe zu sehen sein:

```

1 root@skks1000:~/postgresql_cluster# ansible-playbook deploy_pgcluster.yml
2
3 PLAY [Deploy PostgreSQL HA Cluster (based on "Patroni")]
*****
4
5 TASK [Gathering Facts]
*****
6 ok: [10.0.22.170]
7 ok: [10.0.22.172]
8 ok: [10.0.22.171]
9 ok: [10.0.22.176]
10 ok: [10.0.22.177]
11 ok: [10.0.22.173]
12 ok: [10.0.22.175]
13 ok: [10.0.22.174]
14
15 TASK [Include main variables]
*****
16 ok: [10.0.22.170]
17 ok: [10.0.22.171]
18 ok: [10.0.22.172]
19 ok: [10.0.22.176]
20 ok: [10.0.22.177]
21 ok: [10.0.22.173]
22 ok: [10.0.22.174]
23 ok: [10.0.22.175]
24
25 TASK [Include system variables]

```

```

*****
26 ok: [10.0.22.171] ok: [10.0.22.170]
27 ok: [10.0.22.176]
28 ok: [10.0.22.177]
29 ok: [10.0.22.172]
30 ok: [10.0.22.173]
31 ok: [10.0.22.174]
32 ok: [10.0.22.175]
33
34 TASK [Include OS-specific variables]
*****
35 ok: [10.0.22.171]
36 ok: [10.0.22.170]
37 ok: [10.0.22.172]
38 ok: [10.0.22.176]
39 ok: [10.0.22.177]
40 ok: [10.0.22.173]
41 ok: [10.0.22.174]
42 ok: [10.0.22.175]
43
44 TASK [System information]
*****
45 ok: [10.0.22.170] => {
46     "system_info": {
47         "Architecture": "x86_64",
48         "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
49         "Disk space total": "45.14 GB",
50         "Kernel": "6.1.0-18-amd64",
51         "Memory": "7.71 GB",
52         "OS": "Debian 12.5",
53         "Product name": "VMware Virtual Platform",
54         "Virtualization type": "VMware"
55     }
56 }
57 ok: [10.0.22.171] => {
58     "system_info": {
59         "Architecture": "x86_64",
60         "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
61         "Disk space total": "45.14 GB",
62         "Kernel": "6.1.0-18-amd64",
63         "Memory": "7.71 GB",
64         "OS": "Debian 12.5",
65         "Product name": "VMware Virtual Platform",
66         "Virtualization type": "VMware"
67     }
68 }
69 ok: [10.0.22.176] => {
70     "system_info": {
71         "Architecture": "x86_64",
72         "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
73         "Disk space total": "45.14 GB",

```

```

74     "Kernel": "6.1.0-18-amd64",
75     "Memory": "7.71 GB",           "OS": "Debian 12.5",
76     "Product name": "VMware Virtual Platform",
77     "Virtualization type": "VMware"
78   }
79 }
80 ok: [10.0.22.177] => {
81   "system_info": {
82     "Architecture": "x86_64",
83     "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
84     "Disk space total": "45.14 GB",
85     "Kernel": "6.1.0-18-amd64",
86     "Memory": "7.71 GB",
87     "OS": "Debian 12.5",
88     "Product name": "VMware Virtual Platform",
89     "Virtualization type": "VMware"
90   }
91 }
92 ok: [10.0.22.172] => {
93   "system_info": {
94     "Architecture": "x86_64",
95     "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
96     "Disk space total": "45.14 GB",
97     "Kernel": "6.1.0-18-amd64",
98     "Memory": "7.71 GB",
99     "OS": "Debian 12.5",
100    "Product name": "VMware Virtual Platform",
101    "Virtualization type": "VMware"
102  }
103 }
104 ok: [10.0.22.173] => {
105   "system_info": {
106     "Architecture": "x86_64",
107     "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
108     "Disk space total": "45.14 GB",
109     "Kernel": "6.1.0-20-amd64",
110     "Memory": "7.71 GB",
111     "OS": "Debian 12.5",
112     "Product name": "VMware Virtual Platform",
113     "Virtualization type": "VMware"
114   }
115 }
116 ok: [10.0.22.174] => {
117   "system_info": {
118     "Architecture": "x86_64",
119     "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
120     "Disk space total": "45.14 GB",
121     "Kernel": "6.1.0-20-amd64",
122     "Memory": "7.71 GB",
123     "OS": "Debian 12.5",
124     "Product name": "VMware Virtual Platform",
125     "Virtualization type": "VMware"
126   }
127 }

```

```

128 ok: [10.0.22.175] => {
129     "system_info": {
130         "Architecture": "x86_64",
131         "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
132         "Disk space total": "45.14 GB",
133         "Kernel": "6.1.0-20-amd64",
134         "Memory": "7.71 GB",
135         "OS": "Debian 12.5",
136         "Product name": "VMware Virtual Platform",
137         "Virtualization type": "VMware"
138     }
139 }
140
140 TASK [pre-checks : PostgreSQL | check that data directory "/var/lib/postgresql/16/main"
141     is not initialized]
142 ****
143
144 ok: [10.0.22.173]
145 ok: [10.0.22.175]
146 ok: [10.0.22.174]
147
148 TASK [Update apt cache]
149 ****
150
151 ok: [10.0.22.170]
152 ok: [10.0.22.176]
153 ok: [10.0.22.172]
154 ok: [10.0.22.177]
155 ok: [10.0.22.171]
156 ok: [10.0.22.173]
157 ok: [10.0.22.175]
158 ok: [10.0.22.174]
159
160 changed: [10.0.22.170]
161 changed: [10.0.22.176]
162 changed: [10.0.22.172]
163 changed: [10.0.22.171]
164 changed: [10.0.22.177]
165 changed: [10.0.22.173]
166 changed: [10.0.22.175]
167
168 TASK [Make sure that the iproute is installed]
169 ****
170
171 ok: [10.0.22.170]
172 ok: [10.0.22.176]
173 ok: [10.0.22.172]
174 ok: [10.0.22.171]
175 ok: [10.0.22.177]
176 ok: [10.0.22.173]
177 ok: [10.0.22.175]

```

```

173 ok: [10.0.22.174]
174 PLAY [etcd_cluster.yml | Deploy etcd Cluster]
*****  

175 TASK [Gathering Facts]
*****  

176 ok: [10.0.22.170]
177 ok: [10.0.22.172]
178 ok: [10.0.22.171]
179
180 TASK [Include main variables]
*****  

181 ok: [10.0.22.170]
182 ok: [10.0.22.171]
183 ok: [10.0.22.172]
184
185 TASK [Include system variables]
*****  

186 ok: [10.0.22.170]
187 ok: [10.0.22.171]
188 ok: [10.0.22.172]
189
190 TASK [Include OS-specific variables]
*****  

191 ok: [10.0.22.170]
192 ok: [10.0.22.171]
193 ok: [10.0.22.172]
194
195 TASK [Update apt cache]
*****  

196 ok: [10.0.22.170]
197 ok: [10.0.22.172]
198 ok: [10.0.22.171]
199
200 TASK [Make sure the gnupg and apt-transport-https packages are present]
*****  

201 ok: [10.0.22.170]
202 ok: [10.0.22.172]
203 ok: [10.0.22.171]
204
205 TASK [sysctl : Build a sysctl_conf dynamic variable]
*****  

206 ok: [10.0.22.170] => (item=etcd_cluster)
207 ok: [10.0.22.171] => (item=etcd_cluster)
208 ok: [10.0.22.172] => (item=etcd_cluster)
209
210 TASK [etcd : Make sure the unzip/tar packages are present]

```

```

*****
211 changed: [10.0.22.170] changed: [10.0.22.172]
212 changed: [10.0.22.171]
213
214 TASK [etcd : Download "etcd" package]
*****
215 changed: [10.0.22.171] => (item=https://github.com/etcd-io/etcd/releases/download/v3
216 .5.11/etcd-v3.5.11-linux-amd64.tar.gz)
217 [WARNING]: Module remote_tmp /tmp/root/ansible did not exist and was created with a mode
218 of 0700, this may cause issues when running as another user. To avoid this, create the
219 remote_tmp dir with the correct
220 permissions manually
221 changed: [10.0.22.172] => (item=https://github.com/etcd-io/etcd/releases/download/v3
222 .5.11/etcd-v3.5.11-linux-amd64.tar.gz)
223 changed: [10.0.22.170] => (item=https://github.com/etcd-io/etcd/releases/download/v3
224 .5.11/etcd-v3.5.11-linux-amd64.tar.gz)
225
226 TASK [etcd : Extract "etcd" into /tmp]
*****
227 changed: [10.0.22.170]
228 changed: [10.0.22.172]
229 changed: [10.0.22.171]
230
231 TASK [etcd : Copy "etcd" and "etcdctl" binary files to /usr/local/bin/]
*****
232 changed: [10.0.22.170] => (item=etcd)
233 changed: [10.0.22.172] => (item=etcd)
234 changed: [10.0.22.171] => (item=etcd)
235 changed: [10.0.22.170] => (item=etcdctl)
236 changed: [10.0.22.172] => (item=etcdctl)
237 changed: [10.0.22.171] => (item=etcdctl)
238
239 TASK [etcd : Add etcd user]
*****
240 changed: [10.0.22.170]
241 changed: [10.0.22.171]
242 changed: [10.0.22.172]
243
244 TASK [etcd : Create etcd conf directory]
*****
245 changed: [10.0.22.170]

```

```

246 changed: [10.0.22.172]
247 changed: [10.0.22.171] [WARNING]: sftp transfer mechanism failed on [10.0.22.170]. Use
     ANSIBLE_DEBUG=1 to see detailed information
248 [WARNING]: sftp transfer mechanism failed on [10.0.22.172]. Use ANSIBLE_DEBUG=1 to see
     detailed information
249 [WARNING]: sftp transfer mechanism failed on [10.0.22.171]. Use ANSIBLE_DEBUG=1 to see
     detailed information
250
251 TASK [etcd : Generate conf file "/etc/etcd/etcd.conf"]
 ****
252 changed: [10.0.22.170]
253 changed: [10.0.22.172]
254 changed: [10.0.22.171]
255
256 TASK [etcd : Copy systemd service file]
 ****
257 changed: [10.0.22.170]
258 changed: [10.0.22.171]
259 changed: [10.0.22.172]
260
261 TASK [etcd : Enable and start etcd service]
 ****
262 changed: [10.0.22.172]
263 changed: [10.0.22.171]
264 changed: [10.0.22.170]
265
266 TASK [etcd : Wait for port 2379 to become open on the host]
 ****
267 ok: [10.0.22.170]
268 ok: [10.0.22.172]
269 ok: [10.0.22.171]
270
271 TASK [etcd : Wait until the etcd cluster is healthy]
 ****
272 ok: [10.0.22.172]
273 ok: [10.0.22.171]
274 ok: [10.0.22.170]
275
276 TASK [etcd : cluster health]
 ****
277 ok: [10.0.22.170] => {
278     "msg": "http://10.0.22.170:2379 is healthy: successfully committed proposal: took =
2.588786ms\n"
279 }
280 ok: [10.0.22.171] => {
281     "msg": "http://10.0.22.171:2379 is healthy: successfully committed proposal: took =
5.296369ms\n"
282 }
```

```

283 ok: [10.0.22.172] => {
284     "msg": "http://10.0.22.172:2379 is healthy: successfully committed proposal: took =
2.024139ms\n"}
285
286 PLAY [consul.yml | Configure Consul instances]
*****
287 skipping: no hosts matched
288
289 PLAY [deploy_pgcluster.yml | Postgres Cluster Configuration]
*****
290
291 TASK [Gathering Facts]
*****
292 ok: [10.0.22.174]
293 ok: [10.0.22.173]
294 ok: [10.0.22.175]
295
296 TASK [Include main variables]
*****
297 ok: [10.0.22.173]
298 ok: [10.0.22.174]
299 ok: [10.0.22.175]
300
301 TASK [Include system variables]
*****
302 ok: [10.0.22.173]
303 ok: [10.0.22.174]
304 ok: [10.0.22.175]
305
306 TASK [Include OS-specific variables]
*****
307 ok: [10.0.22.173]
308 ok: [10.0.22.174]
309 ok: [10.0.22.175]
310
311 TASK [add-repository : Add repository apt-key]
*****
312 changed: [10.0.22.173] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.
    asc'})
313 changed: [10.0.22.175] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.
    asc'})
314 changed: [10.0.22.174] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.
    asc'})
315
316 TASK [add-repository : Add repository]
*****

```

```

317 changed: [10.0.22.175] => (item={'repo': 'deb https://apt.postgresql.org/pub/repos/apt/
    bookworm-pgdg main'})
318 changed: [10.0.22.173] => (item={'repo': 'deb https://apt.postgresql.org/pub/repos/apt/
    bookworm-pgdg main'})changed: [10.0.22.174] => (item={'repo': 'deb https://apt.
    postgresql.org/pub/repos/apt/ bookworm-pgdg main'})
319
320 TASK [packages : Install system packages]
*****
321 ok: [10.0.22.173] => (item=['python3'])
322 ok: [10.0.22.175] => (item=['python3'])
323 ok: [10.0.22.174] => (item=['python3'])
324 ok: [10.0.22.173] => (item=python3)
325 ok: [10.0.22.175] => (item=python3)
326 ok: [10.0.22.174] => (item=python3)
327 changed: [10.0.22.175] => (item=python3-dev)
328 changed: [10.0.22.173] => (item=python3-dev)
329 changed: [10.0.22.174] => (item=python3-dev)
330 changed: [10.0.22.175] => (item=python3-psycopg2)
331 changed: [10.0.22.173] => (item=python3-psycopg2)
332 changed: [10.0.22.175] => (item=python3-setuptools)
333 changed: [10.0.22.174] => (item=python3-psycopg2)
334 changed: [10.0.22.173] => (item=python3-setuptools)
335 changed: [10.0.22.174] => (item=python3-setuptools)
336 changed: [10.0.22.175] => (item=python3-pip)
337 changed: [10.0.22.173] => (item=python3-pip)
338 ok: [10.0.22.175] => (item=curl)
339 ok: [10.0.22.173] => (item=curl)
340 ok: [10.0.22.175] => (item=less)
341 ok: [10.0.22.173] => (item=less)
342 ok: [10.0.22.175] => (item=sudo)
343 ok: [10.0.22.173] => (item=sudo)
344 changed: [10.0.22.174] => (item=python3-pip)
345 ok: [10.0.22.174] => (item=curl)
346 ok: [10.0.22.174] => (item=less)
347 ok: [10.0.22.174] => (item=sudo)
348 changed: [10.0.22.175] => (item=vim)
349 changed: [10.0.22.173] => (item=vim)
350 ok: [10.0.22.175] => (item=gcc)
351 ok: [10.0.22.173] => (item=gcc)
352 changed: [10.0.22.175] => (item=jq)
353 changed: [10.0.22.173] => (item=jq)
354 ok: [10.0.22.175] => (item=iptables)
355 ok: [10.0.22.173] => (item=iptables)
356 changed: [10.0.22.174] => (item=vim)
357 ok: [10.0.22.174] => (item=gcc)
358 changed: [10.0.22.175] => (item=acl)
359 changed: [10.0.22.173] => (item=acl)
360 changed: [10.0.22.175] => (item=dnsutils)
361 changed: [10.0.22.174] => (item=jq)
362 changed: [10.0.22.173] => (item=dnsutils)
363 ok: [10.0.22.174] => (item=iptables)
364 changed: [10.0.22.174] => (item=acl)
365 changed: [10.0.22.174] => (item=dnsutils)

```

```

366
367 TASK [packages : PostgreSQL | ensure postgresql database-cluster manager package]
*****changed: [10.0.22.173]
368 changed: [10.0.22.174]
369 changed: [10.0.22.175]
370
371 TASK [packages : PostgreSQL | disable initializing of a default postgresql cluster]
*****
372 changed: [10.0.22.175]
373 changed: [10.0.22.173]
374 changed: [10.0.22.174]
375
376 TASK [packages : PostgreSQL | disable log rotation with logrotate for postgresql]
*****
377 changed: [10.0.22.173]
378 changed: [10.0.22.174]
379 changed: [10.0.22.175]
380
381 TASK [packages : Install PostgreSQL packages]
*****
382 changed: [10.0.22.174] => (item=postgresql-16)
383 changed: [10.0.22.173] => (item=postgresql-16)
384 changed: [10.0.22.175] => (item=postgresql-16)
385 ok: [10.0.22.173] => (item=postgresql-client-16)
386 ok: [10.0.22.174] => (item=postgresql-client-16)
387 ok: [10.0.22.175] => (item=postgresql-client-16)
388 ok: [10.0.22.173] => (item=postgresql-contrib-16)
389 ok: [10.0.22.175] => (item=postgresql-contrib-16)
390 ok: [10.0.22.174] => (item=postgresql-contrib-16)
391 changed: [10.0.22.174] => (item=postgresql-server-dev-16)
392 changed: [10.0.22.175] => (item=postgresql-server-dev-16)
393 changed: [10.0.22.173] => (item=postgresql-server-dev-16)
394 [WARNING]: sftp transfer mechanism failed on [10.0.22.174]. Use ANSIBLE_DEBUG=1 to see
            detailed information
395 [WARNING]: sftp transfer mechanism failed on [10.0.22.173]. Use ANSIBLE_DEBUG=1 to see
            detailed information
396 [WARNING]: sftp transfer mechanism failed on [10.0.22.175]. Use ANSIBLE_DEBUG=1 to see
            detailed information
397
398 TASK [sudo : Add user to /etc/sudoers.d/]
*****
399 changed: [10.0.22.174] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL',
        })
400 changed: [10.0.22.173] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL',
        })
401 changed: [10.0.22.175] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL',
        })
402
403 TASK [swap : Ensure swap exists]

```

```

*****
404 ok: [10.0.22.173] ok: [10.0.22.175]
405 ok: [10.0.22.174]
406
407 TASK [swap : Swap exists]
*****
408 ok: [10.0.22.173] => {
409     "msg": "swap_size_mb: 3904"
410 }
411 ok: [10.0.22.174] => {
412     "msg": "swap_size_mb: 3904"
413 }
414 ok: [10.0.22.175] => {
415     "msg": "swap_size_mb: 3904"
416 }
417
418 TASK [swap : Disable all existing swaps]
*****
419 changed: [10.0.22.173]
420 changed: [10.0.22.174]
421 changed: [10.0.22.175]
422
423 TASK [swap : Remove swap from /etc/fstab]
*****
424 changed: [10.0.22.174]
425 changed: [10.0.22.173]
426 changed: [10.0.22.175]
427
428 TASK [swap : Remove swap file (if exists)]
*****
429 ok: [10.0.22.174]
430 ok: [10.0.22.175]
431 ok: [10.0.22.173]
432
433 TASK [swap : Create swap file]
*****
434 changed: [10.0.22.175]
435 changed: [10.0.22.173]
436 changed: [10.0.22.174]
437
438 TASK [swap : Set permissions on swap file]
*****
439 changed: [10.0.22.173]
440 changed: [10.0.22.174]
441 changed: [10.0.22.175]
442
443 TASK [swap : Make swap file if necessary]

```

```

*****
444 changed: [10.0.22.175] changed: [10.0.22.173]
445 changed: [10.0.22.174]
446
447 TASK [swap : Run swapon on the swap file]
*****
448 changed: [10.0.22.173]
449 changed: [10.0.22.174]
450 changed: [10.0.22.175]
451
452 TASK [swap : Manage swap file entry in fstab]
*****
453 changed: [10.0.22.173]
454 changed: [10.0.22.175]
455 changed: [10.0.22.174]
456
457 TASK [sysctl : Build a sysctl_conf dynamic variable]
*****
458 ok: [10.0.22.173] => (item=master)
459 ok: [10.0.22.173] => (item=postgres_cluster)
460 ok: [10.0.22.174] => (item=postgres_cluster)
461 ok: [10.0.22.174] => (item=replica)
462 ok: [10.0.22.175] => (item=postgres_cluster)
463 ok: [10.0.22.175] => (item=replica)
464
465 TASK [sysctl : Setting kernel parameters]
*****
466 changed: [10.0.22.173] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
467 changed: [10.0.22.175] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
468 changed: [10.0.22.173] => (item={'name': 'vm.swappiness', 'value': '1'})
469 changed: [10.0.22.175] => (item={'name': 'vm.swappiness', 'value': '1'})
470 changed: [10.0.22.173] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
471 changed: [10.0.22.175] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
472 changed: [10.0.22.173] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
473 changed: [10.0.22.175] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
474 changed: [10.0.22.175] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864',
    })
475 changed: [10.0.22.174] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
476 changed: [10.0.22.175] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})
477 changed: [10.0.22.174] => (item={'name': 'vm.swappiness', 'value': '1'})
478 changed: [10.0.22.175] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
479 changed: [10.0.22.174] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
480 changed: [10.0.22.175] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
481 changed: [10.0.22.174] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
482 changed: [10.0.22.175] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
483 changed: [10.0.22.174] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864',
    })
484 changed: [10.0.22.173] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864',
    })

```

```

485 changed: [10.0.22.175] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
486 changed: [10.0.22.174] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})changed:
        [10.0.22.175] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
487 changed: [10.0.22.174] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
488 changed: [10.0.22.175] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000
        65535'})
489 changed: [10.0.22.174] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
490 changed: [10.0.22.175] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000
        '})
491 changed: [10.0.22.174] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
492 changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
493 changed: [10.0.22.173] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})
494 changed: [10.0.22.173] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
495 changed: [10.0.22.173] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
496 changed: [10.0.22.173] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
497 changed: [10.0.22.175] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192
        '})
498 changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
499 changed: [10.0.22.175] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
500 changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
501 changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
502 changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000
        65535'})
503 changed: [10.0.22.174] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000
        '})
504 changed: [10.0.22.175] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
505 changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000
        65535'})
506 changed: [10.0.22.173] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000
        '})
507 changed: [10.0.22.174] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192
        '})
508 changed: [10.0.22.173] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192
        '})
509 changed: [10.0.22.174] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
510 changed: [10.0.22.173] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
511 changed: [10.0.22.174] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
512 changed: [10.0.22.173] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
513
514 TASK [transparent_huge_pages : Create systemd service "disable-transparent-huge-pages.
    service"]
*****
515 changed: [10.0.22.174]
516 changed: [10.0.22.173]
517 changed: [10.0.22.175]
518
519 RUNNING HANDLER [transparent_huge_pages : Start disable-transparent-huge-pages service]
*****
520 changed: [10.0.22.174]
521 changed: [10.0.22.173]
522 changed: [10.0.22.175]
523

```

```

524 TASK [pam_limits : Linux PAM limits | add or modify nofile limits]
*****
525 changed: [10.0.22.174] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value':
65536})changed: [10.0.22.175] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value':
65536})
526 changed: [10.0.22.173] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value':
65536})
527 changed: [10.0.22.174] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value':
200000})
528 changed: [10.0.22.175] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value':
200000})
529 changed: [10.0.22.173] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value':
200000})
530
531 TASK [locales : Generate locales]
*****
532 ok: [10.0.22.174] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})
533 ok: [10.0.22.175] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})
534 ok: [10.0.22.173] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})
535
536 TASK [locales : Set locale "en_US.utf-8" into /etc/default/locale]
*****
537 changed: [10.0.22.173] => (item=LANG=en_US.utf-8)
538 changed: [10.0.22.174] => (item=LANG=en_US.utf-8)
539 changed: [10.0.22.175] => (item=LANG=en_US.utf-8)
540 changed: [10.0.22.173] => (item=LANGUAGE=en_US.utf-8)
541 changed: [10.0.22.174] => (item=LANGUAGE=en_US.utf-8)
542 changed: [10.0.22.175] => (item=LANGUAGE=en_US.utf-8)
543 changed: [10.0.22.173] => (item=LC_ALL=en_US.utf-8)
544 changed: [10.0.22.174] => (item=LC_ALL=en_US.utf-8)
545 changed: [10.0.22.175] => (item=LC_ALL=en_US.utf-8)
546
547 PLAY [balancers.yml | Configure HAProxy load balancers]
*****
548
549 TASK [Gathering Facts]
*****
550 ok: [10.0.22.176]
551 ok: [10.0.22.177]
552
553 TASK [Include main variables]
*****
554 ok: [10.0.22.176]
555 ok: [10.0.22.177]
556
557 TASK [Include system variables]
*****
```

```

558 ok: [10.0.22.176]
559 ok: [10.0.22.177]
560 TASK [Include OS-specific variables]
*****
561 ok: [10.0.22.176]
562 ok: [10.0.22.177]
563
564 TASK [Update apt cache]
*****
565 ok: [10.0.22.176]
566 ok: [10.0.22.177]
567
568 TASK [Make sure the gnupg and apt-transport-https packages are present]
*****
569 ok: [10.0.22.176]
570 ok: [10.0.22.177]
571
572 TASK [sysctl : Build a sysctl_conf dynamic variable]
*****
573 ok: [10.0.22.176] => (item=balancers)
574 ok: [10.0.22.177] => (item=balancers)
575
576 TASK [sysctl : Setting kernel parameters]
*****
577 changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
578 changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
579 changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
580 changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
581 changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000
      65535'})
582 changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000
      65535'})
583 changed: [10.0.22.176] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000
      '})
584 changed: [10.0.22.177] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000
      '})
585 changed: [10.0.22.177] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192
      '})
586 changed: [10.0.22.176] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192
      '})
587 changed: [10.0.22.176] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
588 changed: [10.0.22.177] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
589 changed: [10.0.22.176] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
590 changed: [10.0.22.177] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
591
592 TASK [haproxy : Install HAProxy package]
*****
593 changed: [10.0.22.176]

```

```

594 changed: [10.0.22.177]
595 TASK [haproxy : Make sure the kernel parameter "net.ipv4.ip_nonlocal_bind" are enabled]
***** ok: [10.0.22.177]
596 ok: [10.0.22.176]
597
598 TASK [haproxy : Add haproxy group]
*****
599 ok: [10.0.22.176]
600 ok: [10.0.22.177]
601
602 TASK [haproxy : Add haproxy user]
*****
603 changed: [10.0.22.176]
604 changed: [10.0.22.177]
605
606 TASK [haproxy : Create directories]
*****
607 changed: [10.0.22.177] => (item=/etc/haproxy)
608 changed: [10.0.22.176] => (item=/etc/haproxy)
609 ok: [10.0.22.177] => (item=/run/haproxy)
610 ok: [10.0.22.176] => (item=/run/haproxy)
611 changed: [10.0.22.177] => (item=/var/lib/haproxy/dev)
612 changed: [10.0.22.176] => (item=/var/lib/haproxy/dev)
613 [WARNING]: sftp transfer mechanism failed on [10.0.22.177]. Use ANSIBLE_DEBUG=1 to see
detailed information
614 [WARNING]: sftp transfer mechanism failed on [10.0.22.176]. Use ANSIBLE_DEBUG=1 to see
detailed information
615
616 TASK [haproxy : Generate conf file "/etc/haproxy/haproxy.cfg"]
*****
617 changed: [10.0.22.177]
618 changed: [10.0.22.176]
619
620 TASK [haproxy : Generate systemd service file "/etc/systemd/system/haproxy.service"]
*****
621 changed: [10.0.22.177]
622 changed: [10.0.22.176]
623
624 TASK [confd : Download and copy "confd" binary file to /usr/local/bin/]
*****
625 changed: [10.0.22.177] => (item=https://github.com/kelseyhightower/confd/releases/
download/v0.16.0/confd-0.16.0-linux-amd64)
626 changed: [10.0.22.176] => (item=https://github.com/kelseyhightower/confd/releases/
download/v0.16.0/confd-0.16.0-linux-amd64)
627
628 TASK [confd : Create conf directories]
*****

```

```

629 changed: [10.0.22.176] => (item=/etc/confd/conf.d)changed: [10.0.22.177] => (item=/etc/
    confd/conf.d)
630 changed: [10.0.22.176] => (item=/etc/confd/templates)
631 changed: [10.0.22.177] => (item=/etc/confd/templates)
632
633 TASK [confd : Generate conf file "/etc/confd/confd.toml"]
*****
634 changed: [10.0.22.177]
635 changed: [10.0.22.176]
636
637 TASK [confd : Generate conf file "/etc/confd/conf.d/haproxy.toml"]
*****
638 changed: [10.0.22.176]
639 changed: [10.0.22.177]
640
641 TASK [confd : Generate template "/etc/confd/templates/haproxy.tmpl"]
*****
642 changed: [10.0.22.177]
643 changed: [10.0.22.176]
644
645 TASK [confd : Copy systemd service file]
*****
646 changed: [10.0.22.177]
647 changed: [10.0.22.176]
648
649 TASK [keepalived : Install keepalived packages]
*****
650 changed: [10.0.22.177]
651 changed: [10.0.22.176]
652
653 TASK [keepalived : Make sure the kernel parameters "net.ipv4.ip_nonlocal_bind", "net.ipv4
    .ip_forward" are enabled]
*****
654 ok: [10.0.22.176] => (item=net.ipv4.ip_nonlocal_bind)
655 ok: [10.0.22.177] => (item=net.ipv4.ip_nonlocal_bind)
656 ok: [10.0.22.177] => (item=net.ipv4.ip_forward)
657 ok: [10.0.22.176] => (item=net.ipv4.ip_forward)
658
659 TASK [keepalived : Make sure the "/usr/libexec/keepalived" directory exists]
*****
660 changed: [10.0.22.176]
661 changed: [10.0.22.177]
662
663 TASK [keepalived : Create vrrp_script "/usr/libexec/keepalived/haproxy_check.sh"]
*****

```

```

664 changed: [10.0.22.177]
665 changed: [10.0.22.176]
666 TASK [keepalived : Generate conf file "/etc/keepalived/keepalived.conf"]
*****
667 changed: [10.0.22.177]
668 changed: [10.0.22.176]
669
670 RUNNING HANDLER [haproxy : Restart haproxy service]
*****
671 changed: [10.0.22.177]
672 changed: [10.0.22.176]
673
674 RUNNING HANDLER [haproxy : Check HAProxy is started and accepting connections]
*****
675 ok: [10.0.22.177]
676 ok: [10.0.22.176]
677
678 RUNNING HANDLER [confd : Restart confd service]
*****
679 changed: [10.0.22.177]
680 changed: [10.0.22.176]
681
682 RUNNING HANDLER [keepalived : Restart keepalived service]
*****
683 changed: [10.0.22.177]
684 changed: [10.0.22.176]
685
686 RUNNING HANDLER [keepalived : Wait for the cluster ip address (VIP) "10.0.22.178" is
running]
*****
687 ok: [10.0.22.176]
688 ok: [10.0.22.177]
689
690 PLAY [deploy_pgcluster.yml | Install and configure pgBackRest]
*****
691
692 TASK [Gathering Facts]
*****
693 ok: [10.0.22.173]
694 ok: [10.0.22.175]
695 ok: [10.0.22.174]
696
697 TASK [Include main variables]
*****
698 ok: [10.0.22.173]

```

```

699 ok: [10.0.22.174]
700 ok: [10.0.22.175]
701 TASK [Include OS-specific variables]
*****
702 ok: [10.0.22.173]
703 ok: [10.0.22.174]
704 ok: [10.0.22.175]
705
706 PLAY [deploy_pgcluster.yml | PostgreSQL Cluster Deployment]
*****
707
708 TASK [Gathering Facts]
*****
709 ok: [10.0.22.175]
710 ok: [10.0.22.173]
711 ok: [10.0.22.174]
712
713 TASK [Include main variables]
*****
714 ok: [10.0.22.173]
715 ok: [10.0.22.174]
716 ok: [10.0.22.175]
717
718 TASK [Include system variables]
*****
719 ok: [10.0.22.173]
720 ok: [10.0.22.174]
721 ok: [10.0.22.175]
722
723 TASK [Include OS-specific variables]
*****
724 ok: [10.0.22.173]
725 ok: [10.0.22.174]
726 ok: [10.0.22.175]
727
728 TASK [cron : Make sure that the cron package is installed]
*****
729 ok: [10.0.22.175]
730 ok: [10.0.22.174]
731 ok: [10.0.22.173]
732
733 TASK [pgbouncer : Install pgbouncer package]
*****
734 changed: [10.0.22.175]
735 changed: [10.0.22.173]
736 changed: [10.0.22.174]

```

```
737
738 TASK [pgbouncer : Ensure config directory "/etc/pgbouncer" exist]
*****changed: [10.0.22.173]
739 changed: [10.0.22.175]
740 changed: [10.0.22.174]
741
742 TASK [pgbouncer : Ensure log directory "/var/log/pgbouncer" exist]
*****
743 changed: [10.0.22.173]
744 changed: [10.0.22.174]
745 changed: [10.0.22.175]
746
747 TASK [pgbouncer : Check if pgbouncer systemd service exists]
*****
748 ok: [10.0.22.173]
749 ok: [10.0.22.174]
750 ok: [10.0.22.175]
751
752 TASK [pgbouncer : Stop and disable standard init script]
*****
753 changed: [10.0.22.174]
754 changed: [10.0.22.175]
755 changed: [10.0.22.173]
756
757 TASK [pgbouncer : Configure pgbouncer systemd service file]
*****
758 changed: [10.0.22.175] => (item=pgbouncer)
759 changed: [10.0.22.173] => (item=pgbouncer)
760 changed: [10.0.22.174] => (item=pgbouncer)
761
762 TASK [pgbouncer : Ensure pgbouncer service is enabled]
*****
763 changed: [10.0.22.173] => (item=pgbouncer)
764 changed: [10.0.22.174] => (item=pgbouncer)
765 changed: [10.0.22.175] => (item=pgbouncer)
766
767 TASK [pgbouncer : Enable log rotation with logrotate]
*****
768 changed: [10.0.22.175] => (item=pgbouncer)
769 changed: [10.0.22.174] => (item=pgbouncer)
770 changed: [10.0.22.173] => (item=pgbouncer)
771
772 TASK [pgbouncer : Configure pgbouncer.ini]
*****
773 changed: [10.0.22.175] => (item=pgbouncer)
774 changed: [10.0.22.173] => (item=pgbouncer)
```

```
775 changed: [10.0.22.174] => (item=pgbouncer)
776 TASK [pgpass : Configure a password file (/var/lib/postgresql/.pgpass)]
    ****
    changed: [10.0.22.175]
777 changed: [10.0.22.174]
778 changed: [10.0.22.173]
779
780 RUNNING HANDLER [Restart pgbouncer service]
    ****
781 changed: [10.0.22.174] => (item=pgbouncer)
782 changed: [10.0.22.173] => (item=pgbouncer)
783 changed: [10.0.22.175] => (item=pgbouncer)
784
785 RUNNING HANDLER [Wait for port "6432" to become open on the host]
    ****
786 ok: [10.0.22.173]
787 ok: [10.0.22.174]
788 ok: [10.0.22.175]
789
790 TASK [patroni : Copy patroni requirements.txt file]
    ****
791 changed: [10.0.22.174]
792 changed: [10.0.22.173]
793 changed: [10.0.22.175]
794
795 TASK [patroni : Install setuptools]
    ****
796 changed: [10.0.22.175]
797 changed: [10.0.22.173]
798 changed: [10.0.22.174]
799
800 TASK [patroni : Install requirements]
    ****
801 changed: [10.0.22.174]
802 changed: [10.0.22.173]
803 changed: [10.0.22.175]
804
805 TASK [patroni : Install patroni 3.3.0]
    ****
806 changed: [10.0.22.174]
807 changed: [10.0.22.173]
808 changed: [10.0.22.175]
809
810 TASK [patroni : Create conf directory]
    ****
811 changed: [10.0.22.174]
812 changed: [10.0.22.173]
```

```

813 changed: [10.0.22.175]
814 TASK [patroni : Generate conf file "/etc/patroni/patroni.yml"]
*****
     changed: [10.0.22.173]
815 changed: [10.0.22.174]
816 changed: [10.0.22.175]
817
818 TASK [patroni : Copy systemd service file "/etc/systemd/system/patroni.service"]
*****
819 changed: [10.0.22.173]
820 changed: [10.0.22.174]
821 changed: [10.0.22.175]
822
823 TASK [patroni : Prepare PostgreSQL | make sure the postgresql log directory "/var/log/postgresql" exists]
*****
824 changed: [10.0.22.173]
825 changed: [10.0.22.174]
826 changed: [10.0.22.175]
827
828 TASK [patroni : Prepare PostgreSQL | make sure PostgreSQL data directory "/var/lib/postgresql/16/main" exists]
*****
829 changed: [10.0.22.174]
830 changed: [10.0.22.173]
831 changed: [10.0.22.175]
832
833 TASK [patroni : Prepare PostgreSQL | make sure the postgresql config files exists]
*****
834 ok: [10.0.22.173]
835 ok: [10.0.22.174]
836 ok: [10.0.22.175]
837
838 TASK [patroni : Prepare PostgreSQL | generate default postgresql config files]
*****
839 changed: [10.0.22.173]
840 changed: [10.0.22.175]
841 changed: [10.0.22.174]
842
843 TASK [patroni : Prepare PostgreSQL | make sure the data directory "/var/lib/postgresql/16/main" is empty]
*****
844 changed: [10.0.22.174] => (item=absent)
845 changed: [10.0.22.173] => (item=absent)
846 changed: [10.0.22.175] => (item=absent)
847 changed: [10.0.22.173] => (item=directory)
848 changed: [10.0.22.174] => (item=directory)
849 changed: [10.0.22.175] => (item=directory)

```

```

850
851 TASK [patroni : Start patroni service on the Master server]
*****changed: [10.0.22.173]*****
852
853 TASK [patroni : Wait for port 8008 to become open on the host]
*****ok: [10.0.22.173]*****
854
855
856 TASK [patroni : Check PostgreSQL is started and accepting connections on Master]
*****ok: [10.0.22.173]*****
857
858
859 TASK [patroni : Wait for the cluster to initialize (master is the leader with the lock)]
*****ok: [10.0.22.173]*****
860
861
862 TASK [patroni : Prepare PostgreSQL | generate pg_hba.conf]
*****changed: [10.0.22.173]*****
863 changed: [10.0.22.175]
864 changed: [10.0.22.174]
865
866
867 TASK [patroni : Prepare PostgreSQL | reload for apply the pg_hba.conf]
*****ok: [10.0.22.175]*****
868 ok: [10.0.22.174]
869
870
871
872 TASK [patroni : Make sure the postgresql users are present, and password does not differ
      from the specified]
*****failed: [10.0.22.173] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and (pgbackrest_install|bool or wal_g_install|bool) and (existing_pgcluster is not defined or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template loop_control.label: 'None' has no attribute 'name'. 'None' has no attribute 'name'", "skip_reason": "Conditional result was False"}*****
873 ...ignoring
874 failed: [10.0.22.174] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and (pgbackrest_install|bool or wal_g_install|bool) and (existing_pgcluster is not defined or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template loop_control.label: 'None' has no attribute 'name'. 'None' has no attribute 'name'", "skip_reason": "Conditional result was False"}*****
875 ...ignoring
876 failed: [10.0.22.175] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and ("

```

```

    pgbackrest_install|bool or wal_g_install|bool) and (existing_pgcluster is not defined
    or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template
    loop_control.label: 'None' has no attribute 'name'. 'None' has no attribute 'name'", "
    skip_reason": "Conditional result was False"}
878 ...ignoring
879 TASK [patroni : Start patroni service on Replica servers]
*****
880 changed: [10.0.22.174]
881 changed: [10.0.22.175]
882
883 TASK [patroni : Wait for port 8008 to become open on the host]
*****
884 ok: [10.0.22.174]
885 ok: [10.0.22.175]
886
887 TASK [patroni : Check that the patroni is healthy on the replica server]
*****
888 ok: [10.0.22.174]
889 ok: [10.0.22.175]
890
891 TASK [patroni : Turning off postgresql autostart from config "start.conf" (will be
    managed by patroni)]
*****
892 changed: [10.0.22.173]
893 changed: [10.0.22.174]
894 changed: [10.0.22.175]
895
896 TASK [patroni : Disable "postgresql@16-main" service]
*****
897 ok: [10.0.22.173]
898 ok: [10.0.22.174]
899 ok: [10.0.22.175]
900
901 TASK [patroni : Add PATRONICCTL_CONFIG_FILE environment variable into /etc/environment]
*****
902 changed: [10.0.22.173]
903 changed: [10.0.22.175]
904 changed: [10.0.22.174]
905
906 TASK [postgresql-users : Make sure the PostgreSQL users are present]
*****
907 changed: [10.0.22.173] => (item=pgbouncer)
908 changed: [10.0.22.173] => (item=xksgr_sks1160_gitlab)
909 changed: [10.0.22.173] => (item=xksgr_sks1172_harbor)
910 changed: [10.0.22.173] => (item=xksgr_sks1195_kcso)
911 changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_monitor)
912 changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_backup)

```

```

913
914 TASK [postgresql-users : Grant roles to users]
*****changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_monitor)
915
916 TASK [postgresql-databases : Make sure the PostgreSQL databases are present]
*****
917 changed: [10.0.22.173] => (item={'db': 'k8s_core_gitlab_prod', 'encoding': 'UTF8', ,
918   'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1160_gitlab',
919   })
920 changed: [10.0.22.173] => (item={'db': 'k8s_core_harbor_prod', 'encoding': 'UTF8', ,
921   'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1172_harbor',
922   })
923 changed: [10.0.22.173] => (item={'db': 'k8s_core_keycloak_prod', 'encoding': 'UTF8', ,
924   'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1195_kcsoo',
925   })
926 changed: [10.0.22.173] => (item={'db': 'gramic_test', 'encoding': 'UTF8', 'lc_collate': ,
927   'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'postgres'})
928
929 TASK [pgbouncer/config : Ensure config directory "/etc/pgbouncer" exist]
*****
930 ok: [10.0.22.173]
931 ok: [10.0.22.174]
932 ok: [10.0.22.175]
933
934 TASK [pgbouncer/config : Update pgbouncer.ini]
*****
935 ok: [10.0.22.173] => (item=pgbouncer)
936 ok: [10.0.22.175] => (item=pgbouncer)
937 ok: [10.0.22.174] => (item=pgbouncer)
938
939 TASK [pgbouncer/config : Check if 'user_search' function exists]
*****
940 ok: [10.0.22.173]
941
942 TASK [pgbouncer/config : Create 'user_search' function for pgbouncer 'auth_query' option]
*****
943 changed: [10.0.22.173]
944

```

```

945 TASK [deploy-finish : Get postgresql users list]
*****
946 ok: [10.0.22.173]
947 TASK [deploy-finish : PostgreSQL list of users]
*****
948 ok: [10.0.22.173] => {
949     "users_result.stdout_lines": [
950         "                                     List of roles",
951         "                                     Role name           |   Attributes",
952         "                                     ",           |
953         "                                     pgbouncer          |   ",
954         "                                     postgres            | Superuser, Create role, Create DB, Replication,
955         "                                     Bypass RLS",
956         "                                     replicator          | Replication",
957         "                                     xksgr_k8s_core_psql_backup | Superuser",
958         "                                     xksgr_k8s_core_psql_monitor |   ",
959         "                                     xksgr_sks1160_gitlab    | Superuser",
960         "                                     xksgr_sks1172_harbor    | Superuser",
961         "                                     xksgr_sks1195_kcpresso | Superuser"
962     ]
963 }
964 TASK [deploy-finish : Get postgresql database list]
*****
965 ok: [10.0.22.173]
966
967 TASK [deploy-finish : PostgreSQL list of databases]
*****
968 ok: [10.0.22.173] => {
969     "dbs_result.stdout_lines": [
970         "           name           |       owner        | encoding |   collate   |",
971         "           ctype          |   size      |   tablespace  ",
972         "           ",
973         "           gramic_test    |   postgres      |   UTF8      | en_US.UTF-8 | en_US.
974         "           UTF-8 | 7492 kB | pg_default",
975         "           k8s_core_gitlab_prod | xksgr_sks1160_gitlab | UTF8      | en_US.UTF-8 | en_US.
976         "           UTF-8 | 7492 kB | pg_default",
977         "           k8s_core_harbor_prod | xksgr_sks1172_harbor | UTF8      | en_US.UTF-8 | en_US.
978         "           UTF-8 | 7492 kB | pg_default",
979         "           k8s_core_keycloak_prod | xksgr_sks1195_kcpresso | UTF8      | en_US.UTF-8 | en_US.
980         "           UTF-8 | 7492 kB | pg_default",
981         "           postgres        |   postgres      |   UTF8      | en_US.UTF-8 | en_US.
982         "           UTF-8 | 7492 kB | pg_default",
983         "(5 rows)"
984     ]

```

```

979 }
980 TASK [deploy-finish : Check postgresql cluster health]
*****  

981     ok: [10.0.22.173]  

982
982 TASK [deploy-finish : PostgreSQL Cluster health]
*****  

983 ok: [10.0.22.173] => {
984     "patronictl_result.stdout_lines": [
985         "+ Cluster: k8s-core-psql (7367246950910762117) -----+",
986         "| Member | Host | Role | State | TL | Lag in MB |",
987         "+-----+-----+-----+-----+-----+-----+",
988         "| sks1263 | 10.0.22.173 | Leader | running | 1 | |",
989         "| sks1264 | 10.0.22.174 | Sync Standby | streaming | 1 | 0 |",
990         "| sks1265 | 10.0.22.175 | Sync Standby | streaming | 1 | 0 |",
991         "+-----+-----+-----+-----+-----+-----+"
992     ]
993 }
994
995 TASK [deploy-finish : PostgreSQL Cluster connection info]
*****  

996 ok: [10.0.22.173] => {
997     "msg": [
998         "+-----+",
999         "address (VIP) 10.0.22.178",
1000        "port 5000 (read/write) master",
1001        "port 5001 (read only) all replicas",
1002        "port 5002 (read only) synchronous replica only",
1003        "port 5003 (read only) asynchronous replicas only",
1004        "+-----+"
1005     ]
1006 }
1007
1008 PLAY RECAP
*****
1009 10.0.22.170 : ok=28    changed=11    unreachable=0    failed=0    skipped=56
1010             rescued=0    ignored=0
1011 10.0.22.171 : ok=28    changed=11    unreachable=0    failed=0    skipped=49
1012             rescued=0    ignored=0
1013 10.0.22.172 : ok=28    changed=11    unreachable=0    failed=0    skipped=49
1014             rescued=0    ignored=0
1015 10.0.22.173 : ok=95    changed=52    unreachable=0    failed=0    skipped
1016             =308    rescued=0    ignored=1
1017 10.0.22.174 : ok=81    changed=46    unreachable=0    failed=0    skipped
1018             =294    rescued=0    ignored=1
1019 10.0.22.175 : ok=81    changed=46    unreachable=0    failed=0    skipped
1020             =294    rescued=0    ignored=1
1021 10.0.22.176 : ok=39    changed=20    unreachable=0    failed=0    skipped=89
1022             rescued=0    ignored=0
1023 10.0.22.177 : ok=39    changed=20    unreachable=0    failed=0    skipped=83
1024             rescued=0    ignored=0

```

Listing 135: Deploy - Anhang - Deployt

X.III Maintenance

Das Inventory bleibt gleich.

Geändert wurde nur das main.yml:

```

1---
2#
3# Proxy variables (optional) for download packages using a proxy server
4proxy_env:
5  http_proxy: http://sproxy.sivc.first-it.ch:8080
6  https_proxy: http://sproxy.sivc.first-it.ch:8080
7
8# Cluster variables
9#cluster_vip: "" # IP address for client access to the databases in the cluster (
10#   optional).
10cluster_vip: "10.0.22.178" # IP address for client access to the databases in the
11#   cluster (optional).
11vip_interface: "{{ ansible_default_ipv4.interface }}" # interface name (e.g., "ens32").
12# Note: VIP-based solutions such as keepalived or vip-manager may not function correctly
13#       in cloud environments like AWS.
13
14#patroni_cluster_name: "postgres-cluster" # the cluster name (must be unique for each
15#   cluster)
15patroni_cluster_name: "k8s-core-psql" # the cluster name (must be unique for each
16#   cluster)
16patroni_install_version: "3.3.0" # or 'latest'
17
18patroni_superuser_username: "postgres"
19patroni_superuser_password: "<Password Secure / Safe>"
20patroni_replication_username: "replicator"
21patroni_replication_password: "<Password Secure / Safe>"
22
23#synchronous_mode: false # or 'true' for enable synchronous database replication
24synchronous_mode: true # or 'true' for enable synchronous database replication
25synchronous_mode_strict: false # if 'true' then block all client writes to the master,
26#   when a synchronous replica is not available
26#synchronous_node_count: 1 # number of synchronous standby databases
27synchronous_node_count: 2 # number of synchronous standby databases
28
29# Load Balancing
30with_haproxy_load_balancing: true # or 'true' if you want to install and configure the
31#   load-balancing
31haproxy_listen_port:
32  master: 5000
33  replicas: 5001
34  replicas_sync: 5002
35  replicas_async: 5003
36# The following ('_direct') ports are used for direct connections to the PostgreSQL
37#   database,

```

```

37 # bypassing the PgBouncer connection pool (if 'pgbouncer_install' is 'true').
38 # Uncomment the relevant lines if you need to set up direct connections.
39 stats: 7000
40
41 haproxy_maxconn:
42   global: 100000
43   master: 10000
44   replica: 10000
45
46 haproxy_timeout:
47   client: "60m"
48   server: "60m"
49
50
51 # keepalived (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'true')
52 keepalived_virtual_router_id: "{{ cluster_vip.split('.')[3] | int }}" # The last octet of
53   'cluster_vip' IP address is used by default.
54
55 # virtual_router_id - must be unique in the network (available values are 0..255).
56
57 # vip-manager (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'false'
58   ')
59
60
61 # VIP Manager
62
63 vip_manager_version: "2.4.0" # version to install
64
65 vip_manager_conf: "/etc/patroni/vip-manager.yml"
66
67 vip_manager_interval: "1000" # time (in milliseconds) after which vip-manager wakes up
68   and checks if it needs to register or release ip addresses.
69
70
71 # DCS (Distributed Consensus Store)
72
73 dcs_exists: false # or 'true' if you don't want to deploy a new etcd cluster
74 dcs_type: "etcd" # or 'consul'
75
76
77 # if dcs_type: "etcd" and dcs_exists: false
78 etcd_version: "3.5.11" # version for deploy etcd cluster
79 etcd_data_dir: "/var/lib/etcd"
80
81 etcd_cluster_name: "etcd-{{ patroni_cluster_name }}" # ETCD_INITIAL_CLUSTER_TOKEN
82
83
84 # if dcs_type: "consul" and dcs_exists: true
85 patroni_etcd_hosts: [] # list of servers of an existing etcd cluster
86 patroni_etcd_namespace: "service" # (optional) etcd namespace (prefix)
87 patroni_etcd_username: "" # (optional) username for etcd authentication
88 patroni_etcd_password: "" # (optional) password for etcd authentication
89 patroni_etcd_protocol: "" # (optional) http or https, if not specified http is used
90
91
92 # if dcs_type: "consul"
93
94 consul_version: "1.15.8"
95
96 consul_config_path: "/etc/consul"
97 consul_configd_path: "{{ consul_config_path }}/conf.d"
98
99 consul_data_path: "/var/lib/consul"
100
101 consul_domain: "consul" # Consul domain name
102
103 consul_datacenter: "dc1" # Datacenter label (can be specified for each host in the
104   inventory)
105
106 consul_disable_update_check: true # Disables automatic checking for security bulletins
107   and new version releases

```

```

85 consul_enable_script_checks: true # This controls whether health checks that execute
86     scripts are enabled on this agent
87 consul_enable_local_script_checks: true # Enable them when they are defined in the local
88     configuration files
89 consul_ui: false # Enable the consul UI?
90 consul_syslog_enable: true # Enable logging to syslog
91 consul_iface: "{{ ansible_default_ipv4.interface }}" # specify the interface name with a
92     Private IP (ex. "enp7s0")
93 # TLS
94 # You can enable TLS encryption by dropping a CA certificate, server certificate, and
95     server key in roles/consul/files/
96 consul_tls_enable: false
97 consul_tls_ca_crt: "ca.crt"
98 consul_tls_server_crt: "server.crt"
99 consul_tls_server_key: "server.key"
100 # DNS
101 consul_recursors: [] # List of upstream DNS servers
102 consul_dnsmasq_enable: true # Enable DNS forwarding with Dnsmasq
103 consul_dnsmasq_cache: 0 # dnsmasq cache-size (0 - disable caching)
104 consul_dnsmasq_servers: "{{ nameservers }}" # Upstream DNS servers used by dnsmasq
105 consul_join: [] # List of LAN servers of an existing consul cluster, to join.
106
107 # https://developer.hashicorp.com/consul/docs/discovery/services
108 consul_services:
109     - name: "{{ patroni_cluster_name }}"
110         id: "{{ patroni_cluster_name }}-master"
111         tags: ['master', 'primary']
112         port: "{{ pgbouncer_listen_port }}" # or "{{ postgresql_port }}" if
113         pgbouncer_install: false
114         checks:
115             - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/primary",
116                 interval: "2s" }
117                 - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" } # comment out
118                 this check if pgbouncer_install: false
119             - name: "{{ patroni_cluster_name }}"
120                 id: "{{ patroni_cluster_name }}-replica"
121                 tags: ['replica']
122                 port: "{{ pgbouncer_listen_port }}"
123                 checks:
124                     - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/replica?lag
125 ={{ patroni_maximum_lag_on_replica }}", interval: "2s" }
126                     - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }
127
128 # PostgreSQL variables
129 postgresql_version: "16"
130 # postgresql_data_dir: see vars/Debian.yml or vars/RedHat.yml
131 postgresql_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }},127.0.0.1" to listen on a specific IP address.
132 postgresql_port: "5432"
133 postgresql_encoding: "UTF8" # for bootstrap only (initdb)
134 postgresql_locale: "en_US.UTF-8" # for bootstrap only (initdb)
135 postgresql_data_checksums: true # for bootstrap only (initdb)
136 postgresql_password_encryption_algorithm: "scram-sha-256" # or "md5" if your clients do
137     not work with passwords encrypted with SCRAM-SHA-256

```

```

129
130 # (optional) list of users to be created (if not already exists)
131 postgresql_users:
132   - { name: "{{ pgbouncer_auth_username }}", password: "{{ pgbouncer_auth_password }}",
133     flags: "LOGIN", role: "" }
134   - { name: "xksgr_sks1160_gitlab", password: "<Password Secure / Safe>", flags: "
135     SUPERUSER" }
136   - { name: "xksgr_sks1172_harbor", password: "<Password Secure / Safe>", flags: "
137     SUPERUSER" }
138   - { name: "xksgr_sks1195_kcso", password: "<Password Secure / Safe>", flags: "SUPERUSER"
139   }
140   - { name: "xksgr_k8s_core_psql_monitor", password: "<Password Secure / Safe>", flags: "
141     LOGIN", role: "pg_monitor" }
142   - { name: "xksgr_k8s_core_psql_backup", password: "<Password Secure / Safe>", flags: "
143     SUPERUSER" }
144
145 # (optional) list of databases to be created (if not already exists)
146 #postgresql_databases: []
147 postgresql_databases:
148   - { db: "k8s_core_gitlab_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "
149     en_US.UTF-8", owner: "xksgr_sks1160_gitlab" }
150   - { db: "k8s_core_harbor_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "
151     en_US.UTF-8", owner: "xksgr_sks1172_harbor" }
152   - { db: "k8s_core_keycloak_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype
153     : "en_US.UTF-8", owner: "xksgr_sks1195_kcso" }
154   - { db: "gramic_test", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF
155     -8", owner: "postgres" }
156
157 # (optional) list of schemas to be created (if not already exists)
158 #postgresql_schemas: []
159 # - { schema: "myschema", db: "mydatabase", owner: "mydb-user" }
160
161 # (optional) list of database extensions to be created (if not already exists)
162 #postgresql_extensions: []
163 postgresql_extensions:
164   - { ext: "pgstattuple", db: "postgres" }
165   - { ext: "pgstattuple", db: "k8s_core_gitlab_prod" }
166   - { ext: "pgstattuple", db: "k8s_core_harbor_prod" }
167   - { ext: "pgstattuple", db: "k8s_core_keycloak_prod" }
168   - { ext: "pgstattuple", db: "gramic_test" }
169
170 # postgresql parameters to bootstrap dcs (are parameters for example)
171 postgresql_parameters:
172   - { option: "max_connections", value: "500" }
173   - { option: "superuser_reserved_connections", value: "5" }
174   - { option: "password_encryption", value: "{{ postgresql_password_encryption_algorithm
175     }}" }
176   - { option: "max_locks_per_transaction", value: "512" }
177   - { option: "max_prepared_transactions", value: "0" }
178   - { option: "huge_pages", value: "try" } # or "on" if you set "vm_nr_hugepages" in
179     kernel parameters
180   - { option: "shared_buffers", value: "{{ (ansible_memtotal_mb * 0.25) | int }}MB" } #
181     by default, 25% of RAM
182   - { option: "effective_cache_size", value: "{{ (ansible_memtotal_mb * 0.75) | int }}MB"
183     } # by default, 75% of RAM

```

```

169 - { option: "work_mem", value: "128MB" } # please change this value
170 - { option: "maintenance_work_mem", value: "256MB" } # please change this value
171 - { option: "checkpoint_timeout", value: "15min" }
172 - { option: "checkpoint_completion_target", value: "0.9" }
173 - { option: "min_wal_size", value: "2GB" }
174 - { option: "max_wal_size", value: "8GB" } # or 16GB/32GB
175 - { option: "wal_buffers", value: "32MB" }
176 - { option: "default_statistics_target", value: "1000" }
177 - { option: "seq_page_cost", value: "1" }
178 - { option: "random_page_cost", value: "1.1" } # or "4" for HDDs with slower random
   access
179 - { option: "effective_io_concurrency", value: "200" } # or "2" for traditional HDDs
   with lower I/O parallelism
180 - { option: "synchronous_commit", value: "on" } # or 'off' if you can you lose single
   transactions in case of a crash
181 - { option: "autovacuum", value: "on" } # never turn off the autovacuum!
182 - { option: "autovacuum_max_workers", value: "5" }
183 - { option: "autovacuum_vacuum_scale_factor", value: "0.01" } # or 0.005/0.001
184 - { option: "autovacuum_analyze_scale_factor", value: "0.01" }
185 - { option: "autovacuum_vacuum_cost_limit", value: "500" } # or 1000/5000
186 - { option: "autovacuum_vacuum_cost_delay", value: "2" }
187 - { option: "autovacuum_naptime", value: "1s" }
188 - { option: "max_files_per_process", value: "4096" }
189 - { option: "archive_mode", value: "on" }
190 - { option: "archive_timeout", value: "1800s" }
191 - { option: "archive_command", value: "cd ." } # not doing anything yet with WAL-s
192 - { option: "wal_level", value: "logical" }
193 - { option: "wal_keep_size", value: "2GB" }
194 - { option: "max_wal_senders", value: "10" }
195 - { option: "max_replication_slots", value: "10" }
196 - { option: "hot_standby", value: "on" }
197 - { option: "wal_log_hints", value: "on" }
198 - { option: "wal_compression", value: "on" }
199 - { option: "shared_preload_libraries", value: "pg_stat_statements,auto_explain" }
200 - { option: "pg_stat_statements.max", value: "10000" }
201 - { option: "pg_stat_statements.track", value: "all" }
202 - { option: "pg_stat_statements.track_utility", value: "false" }
203 - { option: "pg_stat_statements.save", value: "true" }
204 - { option: "auto_explain.log_min_duration", value: "10s" } # enable auto_explain for
   10-second logging threshold. Decrease this value if necessary
205 - { option: "auto_explain.log_analyze", value: "true" }
206 - { option: "auto_explain.log_buffers", value: "true" }
207 - { option: "auto_explain.log_timing", value: "false" }
208 - { option: "auto_explain.log_triggers", value: "true" }
209 - { option: "auto_explain.log_verbose", value: "true" }
210 - { option: "auto_explain.log_nested_statements", value: "true" }
211 - { option: "auto_explain.sample_rate", value: "0.01" } # enable auto_explain for 1%
   of queries logging threshold
212 - { option: "track_io_timing", value: "on" }
213 - { option: "log_lock_waits", value: "on" }
214 - { option: "log_temp_files", value: "0" }
215 - { option: "track_activities", value: "on" }
216 - { option: "track_activity_query_size", value: "4096" }
217 - { option: "track_counts", value: "on" }

```

```

218 - { option: "track_functions", value: "all" }
219 - { option: "log_checkpoints", value: "on" }
220 - { option: "logging_collector", value: "on" }
221 - { option: "log_truncate_on_rotation", value: "on" }
222 - { option: "log_rotation_age", value: "1d" }
223 - { option: "log_rotation_size", value: "0" }
224 - { option: "log_line_prefix", value: "'%t [%p-%l] %r %q%u@%d '" }
225 - { option: "log_filename", value: "postgresql-%a.log" }
226 - { option: "log_directory", value: "{{ postgresql_log_dir }}" }
227 - { option: "hot_standby_feedback", value: "on" } # allows feedback from a hot standby
      to the primary that will avoid query conflicts
228 - { option: "max_standby_streaming_delay", value: "30s" }
229 - { option: "wal_receiver_status_interval", value: "10s" }
230 - { option: "idle_in_transaction_session_timeout", value: "10min" } # reduce this
      timeout if possible
231 - { option: "jit", value: "off" }
232 - { option: "max_worker_processes", value: "24" }
233 - { option: "max_parallel_workers", value: "8" }
234 - { option: "max_parallel_workers_per_gather", value: "2" }
235 - { option: "max_parallel_maintenance_workers", value: "2" }
236 - { option: "tcp_keepalives_count", value: "10" }
237 - { option: "tcp_keepalives_idle", value: "300" }
238 - { option: "tcp_keepalives_interval", value: "30" }

239

240 # Set this variable to 'true' if you want the cluster to be automatically restarted
241 # after changing the 'postgresql_parameters' variable that requires a restart in the 'config_pgcluster.yml' playbook.
242 # By default, the cluster will not be automatically restarted.
243 pending_restart: false
244 pending_restart: true # gramic, maintenance test
245
246 # specify additional hosts that will be added to the pg_hba.conf
247 postgresql_pg_hba:
248   - { type: "local", database: "all", user: "{{ patroni_superuser_username }}", address: "",
      method: "trust" }
249   - { type: "local", database: "all", user: "{{ pgbouncer_auth_username }}", address: "",
      method: "trust" } # required for pgbouncer auth_user
250   - { type: "local", database: "replication", user: "all", address: "", method: "trust" }
251   - { type: "host", database: "replication", user: "all", address: "127.0.0.1/32", method:
      "trust" }
252   - { type: "host", database: "replication", user: "all", address: "::1/128", method: "trust" }
253   - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
      address: "10.0.22.173/24", method: "{{ postgresql_password_encryption_algorithm }}" }
254   - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
      address: "10.0.22.174/24", method: "{{ postgresql_password_encryption_algorithm }}" }
255   - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
      address: "10.0.22.175/24", method: "{{ postgresql_password_encryption_algorithm }}" }
256   - { type: "host", database: "replication", user: "{{ patroni_replication_username }}",
      address: "10.0.28.16/24", method: "{{ postgresql_password_encryption_algorithm }}" } # gramic,
      maintenance test
257   - { type: "local", database: "all", user: "all", address: "", method: "{{ postgresql_password_encryption_algorithm }}" }
258   - { type: "host", database: "all", user: "all", address: "127.0.0.1/32", method: "{{ postgresql_password_encryption_algorithm }}" }

```

```

      postgresql_password_encryption_algorithm }}" }
259 - { type: "host", database: "all", user: "all", address: "::1/128", method: "{{ postgresql_password_encryption_algorithm }}" }
260 - { type: "host", database: "k8s_core_gitlab_prod", user: "xksgr_sks1160_gitlab",
261   address: "10.0.20.88/24", method: "{{ postgresql_password_encryption_algorithm }}" }
262 - { type: "host", database: "k8s_core_harbor_prod", user: "xksgr_sks1172_harbor",
263   address: "10.0.20.92/24", method: "{{ postgresql_password_encryption_algorithm }}" }
264 - { type: "host", database: "k8s_core_keycloak_prod", user: "xksgr_sks1195_kcso",
265   address: "10.0.20.98/24", method: "{{ postgresql_password_encryption_algorithm }}" }
266 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.63/24",
267   method: "{{ postgresql_password_encryption_algorithm }}" }
268 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.43/24",
269   method: "{{ postgresql_password_encryption_algorithm }}" }
270 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.77/24",
271   method: "{{ postgresql_password_encryption_algorithm }}" }
272 - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.28.16/24",
273   method: "{{ postgresql_password_encryption_algorithm }}" } # gramic,
274   maintenance test
275
276 # list of lines that Patroni will use to generate pg_ident.conf
277 postgresql_pg_ident: []
278
279 # the password file (~/.pgpass)
280 postgresql_pgpass:
281   - "localhost:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
282   - "{{ inventory_hostname }}:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
283   - "*:{{ pgbouncer_listen_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
284   - "*:{{ postgresql_port }}:*:{{ patroni_replication_username }}:{{ patroni_replication_password }}"
285   - "10.0.20.88:5432:k8s_core_gitlab_prod:xksgr_sks1160_gitlab:<Password Secure / Safe>"
286   - "10.0.20.92:5432:k8s_core_harbor_prod:xksgr_sks1172_harbor:<Password Secure / Safe>"
287   - "10.0.20.98:5432:k8s_core_keycloak_prod:xksgr_sks1195_kcso:<Password Secure / Safe>"
288
289 # PgBouncer parameters
290 pgbouncer_install: true # or 'false' if you do not want to install and configure the
291   pgbouncer service
292 pgbouncer_processes: 1 # Number of pgbouncer processes to be used. Multiple processes
293   use the so_REUSEPORT option for better performance.
294 pgbouncer_conf_dir: "/etc/pgbouncer"
295 pgbouncer_log_dir: "/var/log/pgbouncer"
296 pgbouncer_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
297 pgbouncer_listen_port: 6432
298 pgbouncer_max_client_conn: 10000
299 pgbouncer_max_db_connections: 1000
300 pgbouncer_max_prepared_statements: 1024
301 pgbouncer_default_pool_size: 20
302 pgbouncer_query_wait_timeout: 120
303 pgbouncer_default_pool_mode: "session"
304 pgbouncer_admin_users: "{{ patroni_superuser_username }}" # comma-separated list of
305   users, who are allowed to change settings

```

```

295 pgbounce_stats_users: "{{ patroni_superuser_username }}" # comma-separated list of
   users who are just allowed to use SHOW command
296 pgbounce_ignore_startup_parameters: "extra_float_digits,geqo,search_path"
297 pgbounce_auth_type: "{{ postgresql_password_encryption_algorithm }}"
298 pgbounce_auth_user: true # or 'false' if you want to manage the list of users for
   authentication in the database via userlist.txt
299 pgbounce_auth_username: pgbounce # user who can query the database via the user_search
   function
300 pgbounce_auth_password: "<Password Secure / Safe>"
301 pgbounce_auth_dbname: "postgres"
302 pgbounce_client_tls_sslmode: "disable"
303 pgbounce_client_tls_key_file: ""
304 pgbounce_client_tls_cert_file: ""
305 pgbounce_client_tls_ca_file: ""
306 pgbounce_client_tls_protocols: "secure" # allowed values: tlsv1.0, tlsv1.1, tlsv1.2,
   tlsv1.3, all, secure (tlsv1.2,tlsv1.3)
307 pgbounce_client_tls_ciphers: "default" # allowed values: default, secure, fast, normal,
   all (not recommended)
308
309 pgbounce_pools:
310   - { name: "postgres", dbname: "postgres", pool_parameters: "" }
311   - { name: "k8s_core_gitlab_prod", dbname: "k8s_core_gitlab_prod", pool_parameters: "" }
312   - { name: "k8s_core_harbor_prod", dbname: "k8s_core_harbor_prod", pool_parameters: "" }
313   - { name: "k8s_core_keycloak_prod", dbname: "k8s_core_keycloak_prod", pool_parameters: ""
     }
314   - { name: "gramic_test", dbname: "gramic_test", pool_parameters: "" }
315
316 # Extended variables (optional)
317 #patroni_restapi_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
318 patroni_restapi_listen_addr: "{{ inventory_hostname }}" # gramic, maintenance test
319 patroni_restapi_port: 8008
320 patroni_ttl: 30
321 patroni_loop_wait: 10
322 patroni_retry_timeout: 10
323 patroni_master_start_timeout: 300
324 patroni_maximum_lag_on_failover: 1048576 # (1MB) the maximum bytes a follower may lag to
   be able to participate in leader election.
325 patroni_maximum_lag_on_replica: "100MB" # the maximum of lag that replica can be in order
   to be available for read-only queries.
326
327 # https://patroni.readthedocs.io/en/latest/yaml_configuration.html#postgresql
328 patroni_callbacks: []
329
330 # https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#standby-cluster
331 # Requirements:
332 # 1. the cluster name for Standby Cluster must be unique ('patroni_cluster_name' variable
   )
333 # 2. the IP addresses (or network) of the Standby Cluster servers must be added to the
   pg_hba.conf of the Main Cluster ('postgresql_pg_hba' variable).
334 patroni_standby_cluster:
335   host: "" # an address of remote master
336   port: "5432" # a port of remote master
337 # primary_slot_name: "" # which slot on the remote master to use for replication (

```

```

    optional)
338 # restore_command: "" # command to restore WAL records from the remote master to
      # standby leader (optional)
339 # recovery_min_apply_delay: "" # how long to wait before actually apply WAL records on
      # a standby leader (optional)
340
341 # Permanent replication slots.
342 # These slots will be preserved during switchover/failover.
343 # https://patroni.readthedocs.io/en/latest/dynamic\_configuration.html
344 patroni_slots: []
345 #   - slot: "logical_replication_slot" # the name of the permanent replication slot.
346 #     type: "logical" # the type of slot. Could be 'physical' or 'logical' (if the slot is
      # logical, you have to define 'database' and 'plugin').
347 #     plugin: "pgoutput" # the plugin name for the logical slot.
348 #     database: "postgres" # the database name where logical slots should be created.
349 #   - slot: "test_logical_replication_slot"
350 #     type: "logical"
351 #     plugin: "pgoutput"
352 #     database: "test"
353
354 patroni_log_destination: stderr # or 'logfile'
355 # if patroni_log_destination: logfile
356 patroni_log_dir: /var/log/patroni
357 patroni_log_level: info
358 patroni_log_traceback_level: error
359 patroni_log_format: "%(asctime)s %(levelname)s: %(message)s"
360 patroni_log_dateformat: ""
361 patroni_log_max_queue_size: 1000
362 patroni_log_file_num: 4
363 patroni_log_file_size: 25000000 # bytes
364 patroni_log_loggers_patroni_postmaster: warning
365 patroni_log_loggers_urllib3: warning # or 'debug'
366
367 patroni_watchdog_mode: automatic # or 'off', 'required'
368 patroni_watchdog_device: /dev/watchdog
369
370 patroni_postgresql_use_pg_rewind: true # or 'false'
371 # try to use pg_rewind on the former leader when it joins cluster as a replica.
372
373 patroni_remove_data_directory_on_rewind_failure: false # or 'true' (if use_pg_rewind: '
      # true')
374 # avoid removing the data directory on an unsuccessful rewind
375 # if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica.
376
377 patroni_remove_data_directory_on_diverged_timelines: false # or 'true'
378 # if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica
379 # if it notices that timelines are diverging and the former master can not start
      # streaming from the new master.
380
381 # https://patroni.readthedocs.io/en/latest/replica\_bootstrap.html#bootstrap
382 patroni_cluster_bootstrap_method: "initdb" # or "wal-g", "pgbackrest", "pg_probackup"
383
384 # https://patroni.readthedocs.io/en/latest/replica\_bootstrap.html#building-replicas
385 patroni_create_replica_methods:

```

```

386 #   - pgbackrest
387 #   - wal_g
388 #   - pg_probackup
389   - basebackup
390
391 pgbackrest:
392   - { option: "command", value: "/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore" }
393   - { option: "keep_data", value: "True" }
394   - { option: "no_params", value: "True" }
395 wal_g:
396   - { option: "command", value: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST" }
397   - { option: "no_params", value: "True" }
398 basebackup:
399   - { option: "max-rate", value: "100M" }
400   - { option: "checkpoint", value: "fast" }
401 #   - { option: "waldir", value: "{{ postgresql_wal_dir }}" }
402 pg_probackup:
403   - { option: "command", value: "{{ pg_probackup_restore_command }}" }
404   - { option: "no_params", value: "true" }
405
406 # "restore_command" written to recovery.conf when configuring follower (create replica)
407 postgresql_restore_command: ""
408
409 # pg_probackup
410 pg_probackup_install: false # or 'true'
411 pg_probackup_install_from_postgrespro_repo: true # or 'false'
412 pg_probackup_version: "{{ postgresql_version }}"
413 pg_probackup_instance: "pg_probackup_instance_name"
414 pg_probackup_dir: "/mnt/backup_dir"
415 pg_probackup_threads: "4"
416 pg_probackup_add_keys: "--recovery-target=latest --skip-external-dirs --no-validate"
417 # Ensure there is a space at the beginning of each part to prevent commands from concatenating.
418 pg_probackup_command_parts:
419   - "pg_probackup-{{ pg_probackup_version }}"
420   - " restore -B {{ pg_probackup_dir }}"
421   - " --instance {{ pg_probackup_instance }}"
422   - " -j {{ pg_probackup_threads }}"
423   - " {{ pg_probackup_add_keys }}"
424 pg_probackup_restore_command: "{{ pg_probackup_command_parts | join('') }}"
425 pg_probackup_patroni_cluster_bootstrap_command: "{{ pg_probackup_command_parts | join('') }}"
426
427 # WAL-G
428 wal_g_install: false # or 'true'
429 wal_g_version: "3.0.0"
430 wal_g_json: # config https://github.com/wal-g/wal-g#configuration
431   - { option: "AWS_ACCESS_KEY_ID", value: "{{ AWS_ACCESS_KEY_ID | default('') }}" } # define values or pass via --extra-vars
432   - { option: "AWS_SECRET_ACCESS_KEY", value: "{{ AWS_SECRET_ACCESS_KEY | default('') }}" } # define values or pass via --extra-vars
433   - { option: "WALG_S3_PREFIX", value: "{{ WALG_S3_PREFIX | default('') }}" } # define values or pass via --extra-vars

```

```

434 - { option: "WALG_COMPRESSION_METHOD", value: "brotli" } # or "lz4", "lzma", "zstd"
435 - { option: "WALG_DELTA_MAX_STEPS", value: "6" } # determines how many delta backups
436   can be between full backups
437 - { option: "PGDATA", value: "{{ postgresql_data_dir }}" }
438 - { option: "PGHOST", value: "{{ postgresql_unix_socket_dir }}" }
439 - { option: "PGPORT", value: "{{ postgresql_port }}" }
440 - { option: "PGUSER", value: "{{ patroni_superuser_username }}" }
441 wal_g_archive_command: "wal-g wal-push %p"
442 wal_g_patroni_cluster_bootstrap_command: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST"
443
444 # Define job_parts outside of wal_g_cron_jobs
445 # Ensure there is a space at the beginning of each part to prevent commands from
446   concatenating.
447 wal_g_backup_command:
448   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
449   - " && wal-g backup-push {{ postgresql_data_dir }} > {{ postgresql_log_dir }}/
450     walg_backup.log 2>&1"
451 wal_g_delete_command:
452   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
453   - " && wal-g delete retain FULL 4 --confirm > {{ postgresql_log_dir }}/walg_delete.log
454     2>&1"
455
456 wal_g_cron_jobs:
457   - name: "WAL-G: Create daily backup"
458     user: "postgres"
459     file: /etc/cron.d/walg
460     minute: "30"
461     hour: "3"
462     day: "*"
463     month: "*"
464     weekday: "*"
465     job: "{{ wal_g_backup_command | join('') }}"
466   - name: "WAL-G: Delete old backups" # retain 4 full backups (adjust according to your
467     company's backup retention policy)
468     user: "postgres"
469     file: /etc/cron.d/walg
470     minute: "30"
471     hour: "6"
472     day: "*"
473     month: "*"
474     weekday: "*"
475     job: "{{ wal_g_delete_command | join('') }}"
476
477 # pgBackRest
478 pgbackrest_install: false # or 'true'
479 pgbackrest_install_from_pgdg_repo: false # or 'false'
480 pgbackrest_stanza: "{{ patroni_cluster_name }}" # specify your --stanza
481 pgbackrest_repo_type: "posix" # or "s3", "gcs", "azure"
482 pgbackrest_repo_host: "" # dedicated repository host (optional)
483 pgbackrest_repo_user: "postgres"
484 pgbackrest_conf_file: "/etc/pgbackrest/pgbackrest.conf"

```

```

480 # config https://pgbackrest.org/configuration.html
481 pgbackrest_conf:
482   global: # [global] section
483     - { option: "log-level-file", value: "detail" }
484     - { option: "log-path", value: "/var/log/pgbackrest" }
485     - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
486     - { option: "repo1-path", value: "/var/lib/pgbackrest" }
487     - { option: "repo1-retention-full", value: "4" }
488     - { option: "repo1-retention-archive", value: "4" }
489     - { option: "start-fast", value: "y" }
490     - { option: "stop-auto", value: "y" }
491     - { option: "resume", value: "n" }
492     - { option: "link-all", value: "y" }
493     - { option: "spool-path", value: "/var/spool/pgbackrest" }
494     - { option: "archive-async", value: "y" } # Enables asynchronous WAL archiving (
495       details: https://pgbackrest.org/user-guide.html#async-archiving)
496     - { option: "archive-get-queue-max", value: "1GiB" }
497   stanza: # [stanza_name] section
498     - { option: "process-max", value: "4" }
499     - { option: "log-level-console", value: "info" }
500     - { option: "recovery-option", value: "recovery_target_action=promote" }
501     - { option: "pg1-socket-path", value: "{{ postgresql_unix_socket_dir }}" }
502     - { option: "pg1-path", value: "{{ postgresql_data_dir }}" }
503     - { option: "", value: "" }
504   # (optional) dedicated backup server config (if "repo_host" is set)
505 pgbackrest_server_conf:
506   global:
507     - { option: "log-level-file", value: "detail" }
508     - { option: "log-level-console", value: "info" }
509     - { option: "log-path", value: "/var/log/pgbackrest" }
510     - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
511     - { option: "repo1-path", value: "/var/lib/pgbackrest" }
512     - { option: "repo1-retention-full", value: "4" }
513     - { option: "repo1-retention-archive", value: "4" }
514     - { option: "repo1-bundle", value: "y" }
515     - { option: "repo1-block", value: "y" }
516     - { option: "start-fast", value: "y" }
517     - { option: "stop-auto", value: "y" }
518     - { option: "resume", value: "n" }
519     - { option: "link-all", value: "y" }
520     - { option: "archive-check", value: "y" }
521     - { option: "archive-copy", value: "n" }
522     - { option: "backup-standby", value: "y" }
523     - { option: "", value: "" }
524   # the stanza section will be generated automatically
525
526 pgbackrest_archive_command: "pgbackrest --stanza={{ pgbackrest_stanza }} archive-push %p"
527
528 pgbackrest_patroni_cluster_restore_command:
529   '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore' # restore from
      latest backup
530 # '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --type=time "--target=2020-06-01
      11:00:00+03" --delta restore' # Point-in-Time Recovery (example)

```

```

531 # By default, the cron jobs is created on the database server.
532 # If 'repo_host' is defined, the cron jobs will be created on the pgbackrest server.
533 pgbackrest_cron_jobs:
534   - name: "pgBackRest: Full Backup"
535     file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
536     user: "postgres"
537     minute: "30"
538     hour: "6"
539     day: "*"
540     month: "*"
541     weekday: "0"
542     job: "pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup"
543     # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --
544     # type=full --stanza={{ pgbackrest_stanza }} backup; fi"
545   - name: "pgBackRest: Diff Backup"
546     file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
547     user: "postgres"
548     minute: "30"
549     hour: "6"
550     day: "*"
551     month: "*"
552     weekday: "1-6"
553     job: "pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup"
554     # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --
555     # type=diff --stanza={{ pgbackrest_stanza }} backup; fi"
556
557 # PITR mode (if patroni_cluster_bootstrap_method: "pgbackrest" or "wal-g"):
558 # 1) The database cluster directory will be cleaned (for "wal-g") or overwritten (for "
559 #    pgbackrest" --delta restore).
560 # 2) And also the patroni cluster "{{ patroni_cluster_name }}" will be removed from the
561 #    DCS (if exist) before recovery.
562
563 disable_archive_command: true # or 'false' to not disable archive_command after restore
564 keep_patroni_dynamic_json: true # or 'false' to remove patroni.dynamic.json after
565     restore (if exists)
566
567 # Netdata - https://github.com/netdata/netdata
568 netdata_install: false # or 'true' for install Netdata on postgresql cluster nodes (with
569     kickstart.sh)
570 netdata_install_options: "--stable-channel --disable-telemetry --dont-wait"
571 netdata_conf:
572   web_bind_to: "*"
573   # https://learn.netdata.cloud/docs/store/change-metrics-storage
574   memory_mode: "dbengine" # The long-term metrics storage with efficient RAM and disk
575     usage.
576   page_cache_size: 64 # Determines the amount of RAM in MiB that is dedicated to caching
577     Netdata metric values.
578   dbengine_disk_space: 1024 # Determines the amount of disk space in MiB that is
579     dedicated to storing Netdata metric values.
580
581 ...

```

Listing 136: Testsystem - Anhang - Maintenance - main.yml

Nun muss ebenfalls das Playbook ausgeführt werden:

```
1 ansible-playbook config_pgcluster.yml
2
```

Listing 137: Testsystem - Anhang - Maintenance - config_pgcluster.yml

X.IV Prequenteries - Erweiterungstests

Auf dem sks9016 muss zuerst der apt-Proxy gesetzt werden:

```
1 # nano /etc/apt/apt.conf.d/proxy.conf
2 Acquire::http::Proxy "http://sproxy.sivc.first-it.ch:8080";
3 Acquire::https::Proxy "http://sproxy.sivc.first-it.ch:8080";
4 Acquire::http::proxy::foreman.ksgr.ch "DIRECT";
5
```

Listing 138: Testsystem - sks9016 - apt-Proxy Settings

Auch der Proxy muss gesetzt werden:

```
1 # nano /etc/profile.d/proxy.sh
2 export https_proxy=http://sproxy.sivc.first-it.ch:8080
3 export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
4 export http_proxy=http://sproxy.sivc.first-it.ch:8080
5 export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
6 export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
7 export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
8 # source /etc/profile.d/proxy.sh
9
```

Listing 139: Testsystem - sks9016 - Proxy Settings

Firewall war keine auf dem Host installiert.

X.V Haproxy erweitern

Zuerst musste das Inventory um den entsprechenden Node erweitert werden:

```
1 [etcd_cluster] # recommendation: 3, or 5-7 nodes
2 10.0.22.170
3 10.0.22.171
4 10.0.22.172
5
6 # if dcs_exists: false and dcs_type: "consul"
7 [consul_instances] # recommendation: 3 or 5-7 nodes
8
9 # if with_haproxy_load_balancing: true
10 [balancers]
11 10.0.22.176
12 10.0.22.177
13 10.0.18.16 new_node=true
14
```

```

15 # PostgreSQL nodes
16 [master]
17 10.0.22.173 postgresql_exists=false
18 [replica]
19 10.0.22.174 postgresql_exists=false
20 10.0.22.175 postgresql_exists=false
21
22 [postgres_cluster:children]
23 master
24 replica
25
26 # if pgbackrest_install: true and "repo_host" is set
27 [pgbackrest] # optional (Dedicated Repository Host)
28
29 # Connection settings
30 [all:vars]
31 ansible_connection='ssh'
32 ansible_ssh_port='22'
33 ansible_user='itgramic'
34 ansible_ssh_pass='<Secret ;->' # "sshpass" package is required for use "
35     ansible_ssh_pass"
36 #ansible_ssh_private_key_file=
37 ansible_python_interpreter='/usr/bin/env python3'
38
39 [pgbackrest:vars]
40

```

Listing 140: Testsystem - Anhang - add_balancer.yml - inventory

Nun muss das Playbook add_balancer.yml ausgeführt werden:

```

1 ansible-playbook add_balancer.yml
2

```

Listing 141: Testsystem - Anhang - add_balancer.yml

X.VI Patroni Node

Zuerst musste das Inventory um den entsprechenden Node erweitert werden:

```

1 [etcd_cluster] # recommendation: 3, or 5-7 nodes
2 10.0.22.170
3 10.0.22.171
4 10.0.22.172
5
6 # if dcs_exists: false and dcs_type: "consul"
7 [consul_instances] # recommendation: 3 or 5-7 nodes
8
9 # if with_haproxy_load_balancing: true
10 [balancers]
11 10.0.22.176
12 10.0.22.177
13
14 # PostgreSQL nodes

```

```

15 [master]
16 10.0.22.173 postgresql_exists=false
17 [replica]10.0.22.174 postgresql_exists=false
18 10.0.22.175 postgresql_exists=false
19 10.0.28.16 postgresql_exists=false new_node=true
20
21 [postgres_cluster:children]
22 master
23 replica
24
25 # if pgbackrest_install: true and "repo_host" is set
26 [pgbackrest] # optional (Dedicated Repository Host)
27
28 # Connection settings
29 [all:vars]
30 ansible_connection='ssh'
31 ansible_ssh_port='22'
32 ansible_user='itgramic'
33 ansible_ssh_pass='<Secret ;->' # "sshpass" package is required for use "
34     ansible_ssh_pass"
35 #ansible_ssh_private_key_file=
36 ansible_python_interpreter='/usr/bin/env python3'
37
38 [pgbackrest:vars]

```

Listing 142: Testsystem - Anhang - add_pgnode.yml - inventory

Das Deployment erfolgt nun ebenfalls via Playbook add_pgnode.yml:

```

1 ansible-playbook add_pgnode.yml
2

```

Listing 143: Testsystem - Anhang - add_pgnode.yml

XI **Testsystem - Testing**

XII **Maintenance-Tool**

XII.I **Maintenance-Tool - Bloated Tables**

Das Python-Skript:

```

1 import os
2 import psycopg2
3 from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
4 import yaml
5 from kubernetes import client, config
6 import base64
7 import sys
8
9 # Pass Exception Class
10 class OneException(Exception):
11     pass
12

```

```

13 # Read the ConfigMap
14 #
15 # Get the ConfigMap from the Namespace
16 def read_config(configmap):
17     # config = os.environ['config-map-connections.yaml']
18     script_dir = os.path.dirname(os.path.abspath(__file__))
19     configfile = os.path.join(script_dir, configmap)
20     config = dict()
21     with open(configfile, "r") as file:
22         config = yaml.load(file, Loader=yaml.FullLoader)
23     print(config)
24     return config
25
26 # Open Database Connection
27 #
28 # Opens a Database Connection to the over given Database.
29 # Returns the opened Connection.
30 def database_connection(config, database, user, password):
31     # user = config['connectiondata']['username']
32     host = config['host']
33     port = config['port']
34     conn = psycopg2.connect(database=database,
35                             user=user,
36                             host=host,
37                             password=password,
38                             port=port)
39
40     return conn
41
42 # Get the Secret
43 #
44 # Get the Secret from the Namespace
45 def get_secret(secret_name):
46     secret = open(secret_name, "r").read()
47     print("secret", secret)
48     return secret
49
50 # Get the Username
51 #
52 # Get the Username from the Secret from the Namespace
53 def get_username(secret_name):
54     username = open(secret_name, "r").read()
55     print("username", username)
56     return username
57
58 # Cleanup Bloated Database
59 #
60 # Connected to the Database, check the AUTOVACUUM and VACUUM the Table and
61 # REINDEX all Indices of the Table
62 def cleanup_bloated_database(config):
63     # Get Configuration
64     dead_tuple_percent = config['dead_tuple_percent']
65     database = config['database']
66     secret_username = config['secret_username']

```

```

67     # Get Secret of the Database
68     secret = get_secret(secret_password)
69     # Get Username for the Database
70     user = get_username(secret_username)
71
72     # Open Database Connection
73     connection = database_connection(config, database, user, secret)
74
75     # Get the AUTOVACUUM State
76     active_autovaacum = check_autovacuum(connection)
77
78     # Check AUTOVACUUM State
79     #
80     # Run Maintenance Job if no AUTOVACUUM is running
81     if active_autovaacum == 0:
82         # Get bloated Tables
83         bloated_tables = get_bloated_tables(connection, dead_tuple_percent)
84         # Cleanup bloated Tables
85         cleanup_bloated_tables(connection, bloated_tables)
86     else:
87         print('autovacuum maintenance active!')
88
89     # Close Database Connection
90     connection.close()
91
92 # Check AUTOVACUUM
93 #
94 # Check if an AUTOVACUUM Job is running.
95 # For this, a Select on the Table pg_stat_activity on all Queries like autovacuum:% is
96 # used.
97 def check_autovacuum(connection):
98     cur = connection.cursor()
99     sql = "select count(*) as active from pg_stat_activity where query like 'autovacuum
100 :%';"
101     cur.execute(sql)
102     active_autovaacum = cur.fetchone()
103     active_autovaacum = active_autovaacum[0]
104
105     return active_autovaacum
106
107 # Get Bloated Tables
108 #
109 # Select the Bloated Tables.
110 def get_bloated_tables(connection, dead_tuple_percent):
111     cur = connection.cursor()
112     sql = """select
113         relname as table,
114         (pgstattuple(oid)).dead_tuple_percent
115     from pg_class
116     where
117         relkind = 'r' and
118         (pgstattuple(oid)).dead_tuple_percent > %s
119     order by dead_tuple_percent desc;
120         """

```

```

119     cur.execute(sql, (dead_tuple_percent,))
120     return cur
121 # Cleanup Bloated Tables
122 #
123 # Run through the Bloated Tables and VACUUM and REINDEX
124 def cleanup_bloated_tables(connection, cur):
125     bloated_tables = cur.fetchall()
126     for bloated_table in bloated_tables:
127         table = bloated_table[0]
128         vacuum_table(connection, table)
129         reindex_table(connection, table)
130     print("finished")
131
132 # VACUUM Table
133 #
134 # VACUUM the Table
135 def vacuum_table(connection, table):
136     old_isolation = connection.isolation_level
137     connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
138     cur = connection.cursor()
139     sql = "vacuum " + table + ";"
140     cur.execute(sql)
141     connection.set_isolation_level(old_isolation)
142     print("vacuum table " + table + " successfully")
143
144 # REINDEX
145 #
146 # REINDEX the whole Table
147 def reindex_table(connection, table):
148     cur = connection.cursor()
149     sql = "reindex table " + table + ";"
150     cur.execute(sql)
151     connection.commit()
152     print("reindex table " + table + " successfully")
153
154 # Main Method
155 def maintenance_bloated_databases(confimap):
156     # confimap = confimap_Arg[0]
157     try:
158         # print(confimap)
159         config = read_config(confimap)
160         cleanup_bloated_database(config)
161     except OneException as e:
162         print(repr(e))
163
164 arg1 = sys.argv[1]
165 maintenance_bloated_databases( arg1 )
166

```

Listing 144: Maintenance-Tool - Bloated Tables / Indices
 ksgr_postgresql_maintenance_bloated_tables.py

Zuerst muss der Namespace `ksgr-postgresql-maintenance` erstellt werden:

```
1 kubectl create namespace ksgr-postgresql-maintenance  
2
```

Listing 145: Maintenance-Tool - Bloated Tables / Indices - Namespace

Das Python-Skript `ksgr_postgresql_maintenance_bloated_tables.py` kann in ein ConfigMap umgewandelt werden:

```
1 kubectl create configmap ksgr-postgresql-maintenance-bloated-tables --from-file /home/  
    gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/  
    ksgr_postgresql_maintenance_bloated_tables.py --dry-run=client -o yaml > /home/gramic/  
    PycharmProjects/ksgr_postgresql_maintenance/cleaned/  
    ksgr_postgresql_maintenance_bloated_tables.yml -n ksgr-postgresql-maintenance  
2
```

Listing 146: Maintenance-Tool - Bloated Tables / Indices - Python > ConfigMap

Entsprechend lässt es sich deployen:

```
1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/  
    ksgr_postgresql_maintenance_bloated_tables.yml -n ksgr-postgresql-maintenance  
2
```

Listing 147: Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy

Der Username und das Passwort für den Cluster wird als Secret gespeichert.

Dazu werden erst die Daten in Base64 Strings umgewandelt:

```
1 USER=postgres  
2 PASSWORD=<Password Secure / Safe>  
3 echo -n ${USER} | base64  
4 echo -n ${PASSWORD} | base64  
5
```

Listing 148: Maintenance-Tool - Bloated Tables / Indices - Base64

Diese werden nun in ein Secret gegeben:

```
1 apiVersion: v1  
2 kind: Secret  
3 metadata:  
4   name: secret-vks0041  
5   namespace: ksgr-postgresql-maintenance  
6 type: Opaque  
7 data:  
8   user: <Base64 User>  
9   password: <Base64 Password>  
10
```

Listing 149: Maintenance-Tool - Bloated Tables / Indices - Secret

Das Secret kann deployt werden:

```
1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/secret_vks0041  
    .yaml  
2
```

Listing 150: Maintenance-Tool - Bloated Tables / Indices - Secret Deploy

Die ConfigMap configmap-vks0041-gramic-test beinhaltet den Hostname, den Port, den Datenbanknamen sowie den Pfad zu den Secrets:

```
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: configmap-vks0041-gramic-test
5   namespace: ksgr-postgresql-maintenance
6 data:
7   configmap-vks0041-gramic-test.yaml: |-
8     host: "10.0.22.178"
9     port: "5000"
10    database: "gramic_test"
11    secret_username: "/mnt/etc/secret-volume/secret-vks0041/user"
12    secret_password: "/mnt/etc/secret-volume/secret-vks0041/password"
13    dead_tuple_percent: "1.5"
14
```

Listing 151: Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test

Das Deployment sieht entsprechend aus:

```
1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/
  configmap-vks0041-gramic_test.yaml
2
```

Listing 152: Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test Deploy

Das finale Deployment erfolgt mittels einer CronJob.

Jeden Tag um Mitternacht wird ein Pod deployt, welcher Skript ausführt.

Dabei müssen erst folgende Dependencis installiert werden:

- psycopg2-binary
- PyYAML
- kubernetes

Normalerweise würden diese PyPi-Packages in einem helm Chart als Ressource mitgegeben werden.

Dann wird das Skript mit dem ConfigMap ausgeführt.

Dazu wird das Secret und das ConfigMap in das Pod-Filesystem **gemounted**.

```
1 apiVersion: batch/v1
2 kind: CronJob
3 metadata:
4   name: ksgr-maintenance-bloating-vks0041-gramic-test
5   namespace: ksgr-postgresql-maintenance
6 spec:
7   schedule: "0 0 * * *"
8   jobTemplate:
9     spec:
10       template:
11         metadata:
12           labels:
13             workload: cronjob
```

```

14     type: ksgr-postgresql-maintenance
15   spec:
16     containers:
17       - name: ksgr-maintenance-bloating-vks0041-gramic-test-container
18         image: python:3.11-bookworm
19         command: ["sh", "-c"]
20         args:
21           [
22             "python -m pip install --proxy http://sproxy.sivc.first-it.ch:8080
psycopg2-binary PyYAML kubernetes;
23             python /tmp/python/ksgr_postgresql_maintenance_bloated_tables.py /tmp/
conf/configmap-vks0041-gramic-test.yaml"
24           ]
25         volumeMounts:
26           - name: ksgr-postgresql-maintenance-bloated-tables
27             mountPath: /tmp/python
28           - name: configmap-vks0041-gramic-test
29             mountPath: /tmp/conf
30           - name: secret-volume
31             readOnly: true
32             mountPath: /mnt/etc/secret-volume/secret-vks0041
33
34         restartPolicy: Never
35       volumes:
36         - name: sproxy
37           configMap:
38             name: sproxy
39         - name: ksgr-postgresql-maintenance-bloated-tables
40           configMap:
41             name: ksgr-postgresql-maintenance-bloated-tables
42         - name: configmap-vks0041-gramic-test
43           configMap:
44             name: configmap-vks0041-gramic-test
45         - name: secret-volume
46           secret:
47             secretName: secret-vks0041
48

```

Listing 153: Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-test

Das Deployment:

```

1 kubectl apply -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/ksgr-
   maintenance-bloating-vks0041-gramic_test-cronjob.yaml
2

```

Listing 154: Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-test
Deploy

XII.II Maintenance-Tool - AUTOVACUUM

Das Python-Skript:

```

1 import os
2 import sys

```

```

3 from datetime import date
4 import pandas as pdimport psycopg2
5 import yaml
6
7
8 # Pass Exception Class
9 class OneException(Exception):
10     pass
11
12 # Read the ConfigMap
13 #
14 # Get the ConfigMap from the Namespace
15 def read_config(confimap):
16     # config = os.environ['config-map-connections.yaml']
17     script_dir = os.path.dirname(os.path.abspath(__file__))
18     configfile = os.path.join(script_dir, confimap)
19     config = dict()
20     with open(configfile, "r") as file:
21         config = yaml.load(file, Loader=yaml.FullLoader)
22     print(config)
23     return config
24
25
26 # Open Database Connection
27 #
28 # Opens a Database Connection to the over given Database.
29 # Returns the opened Connection.
30 def database_connection(config, database, user, password):
31     # user = config['connectiondata']['username']
32     host = config['host']
33     port = config['port_ro']
34     conn = psycopg2.connect(database=database,
35                            user=user,
36                            host=host,
37                            password=password,
38                            port=port)
39     return conn
40
41
42 # Get the Secret
43 #
44 # Get the Secret from the Namespace
45 def get_secret(secret_name):
46     secret = open(secret_name, "r").read()
47     return secret
48
49 # Get the Username
50 #
51 # Get the Username from the Secret from the Namespace
52 def get_username(secret_name):
53     username = open(secret_name, "r").read()
54     print("username", username)
55     return username
56

```

```

57 #     Get PostgreSQL Parameters
58 ##     Get autovacuum_vacuum_scale_factor and autovacuum_vacuum_threshold from pg_settings
59 #     Get sum of reltuples from pg_class and get the sender_host (Primary) IP Address from
60 #     pg_stat_wal_receiver
61 def get_paramaters(connection):
62     cur = connection.cursor()
63     sql = """select
64     (
65         select
66             setting as autovacuum_vacuum_scale_factor
67             from pg_settings
68             where
69                 name = 'autovacuum_vacuum_scale_factor'
70             ) as autovacuum_vacuum_scale_factor,
71     (
72         select
73             setting as autovacuum_vacuum_threshold
74             from pg_settings
75             where
76                 name = 'autovacuum_vacuum_threshold'
77             ) as autovacuum_vacuum_threshold,
78     (
79         select
80             sum(reltuples) as reltuples
81             from pg_class
82         ) as reltuples,
83     (
84         select
85             sender_host
86             from pg_stat_wal_receiver
87         ) as sender_host
88 ;"""
89     cur.execute(sql)
90     postgresql_parameters = cur.fetchone()
91     autovacuum_vacuum_scale_factor = postgresql_parameters[0]
92     autovacuum_vacuum_threshold = postgresql_parameters[1]
93     reltuples = postgresql_parameters[2]
94     sender_host = postgresql_parameters[3]
95
96     postgresql_parameter = dict(autovacuum_vacuum_scale_factor=
97                                 autovacuum_vacuum_scale_factor,
98                                 autovacuum_vacuum_threshold=autovacuum_vacuum_threshold,
99                                 reltuples=reltuples,
100                                sender_host=sender_host)
101
102 #
103 #     Calculate needed autovacuum_vacuum_scale_factor and check with the configured ones.
104 #     Create HTML Table if values are different
105 def calculate_check_autovacuum(postgres_params, configmap_autovacuum):
106     autovacuum_vacuum_scale_factor_actual = float(postgres_params.get(
107         'autovacuum_vacuum_scale_factor'))

```

```

107     autovacuum_vacuum_threshold = float(postgres_params.get('autovacuum_vacuum_threshold',
108 ))  

109     reltuples = float(postgres_params.get('reltuples'))      sender_host = postgres_params.  

110     get('sender_host')  

111     max_death_tuples = configmap_autovacuum.get('max_death_tuples')  

112     autovacuum_vacuum_scale_factor = round(((max_death_tuples -  

113     autovacuum_vacuum_threshold) / reltuples), 2)  

114  

115     if autovacuum_vacuum_scale_factor != autovacuum_vacuum_scale_factor_actual:  

116         result_table = create_pandas_dataframe(autovacuum_vacuum_scale_factor_actual,  

117     autovacuum_vacuum_scale_factor, ['autovacuum_vacuum_scale_factor'])  

118         result_path = configmap_autovacuum.get('calc_result')  

119         filename_prefix = configmap_autovacuum.get('filename_prefix')  

120         filename = filename_prefix + str(date.today()) + '.html'  

121         write_calculating_result(result_table, result_path, filename)  

122  

123 # Pandas DataFrame  

124 #  

125 # Create a Pandas DataFrame from the overgiven Parameters  

126 def create_pandas_dataframe(autovacuum_vacuum_scale_factor_actual,  

127     autovacuum_vacuum_scale_factor, index):  

128     data = {'Bestehend': autovacuum_vacuum_scale_factor_actual, 'Berechnet':  

129     autovacuum_vacuum_scale_factor}  

130     result_df = pd.DataFrame(data, index=index)  

131     return result_df  

132  

133  

134 # Write HTML  

135 #  

136 # Write the Result Pandas DataFrame to HTML Table  

137 def write_calculating_result(result_table, path, filename):  

138     # html = result_table.to_html(classes='table table-striped')  

139     html = result_table.to_html()  

140     filepath = path + '/' + filename  

141     text_file = open(filepath, "w")  

142     text_file.write(html)  

143     text_file.close()  

144  

145 # Maintenance Main  

146 #  

147 # Calculate the AUTOVACUUM Parameters  

148 def maintenance_autovacuum_calculation(configmap_conn, configmap_autovacuum):  

149     try:  

150         # Read the Configs  

151         config = read_config(configmap_conn)  

152         config_autovac = read_config(configmap_autovacuum)  

153  

154         # Get Configuration  

155         database = config['database']  

156         secret_username = config['secret_username']  

157         secret_password = config['secret_password']  

158  

159         # Get Secret of the Database  

160         secret = get_secret(secret_password)

```

```

155
156     # Get Username for the Database           user = get_username(secret_username)
157
158     # Open Database Connection
159     connection = database_connection(config, database, user, secret)
160
161     # Get the PostgreSQL Parameters
162     postgres_params = get_paramaters(connection)
163
164     # Calculate and Check the Autovacuum Vacuum Scale Factor
165     calculate_check_autovacuum(postgres_params, config_autovac)
166
167     # Close Database Connection
168     connection.close()
169
170     # cleanup_bloated_database(config)
171 except OneException as e:
172     print(repr(e))
173
174 arg1 = sys.argv[1]
175 arg2 = sys.argv[2]
176 maintenance_autovacuum_calculation(arg1, arg2)
177

```

Listing 155: Maintenance-Tool - AUTOVACUUM
 ksgr_postgresql_maintenance_autovacuum_calculation.py

Das Python-Skript ksgr_postgresql_maintenance_autovacuum_calculation.py kann in ein ConfigMap umgewandelt werden:

```

1 kubectl create configmap ksgr-postgresql-maintenance-autovacuum-calculation --from-file /
  home/gramic/PycharmProjects/ksgr_postgresql_maintenance/
  ksgr_postgresql_maintenance_autovacuum_calculation.py --dry-run=client -o yaml > /home
  /gramic/PycharmProjects/ksgr_postgresql_maintenance/
  ksgr_postgresql_maintenance_autovacuum_calculation.yml -n ksgr-postgresql-maintenance
2

```

Listing 156: Maintenance-Tool - AUTOVACUUM - Python > ConfigMap

Entsprechend lässt es sich deployen:

```

1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/
  ksgr_postgresql_maintenance_autovacuum_calculation.yml -n ksgr-postgresql-maintenance
2

```

Listing 157: Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy

Ein neues Secret muss nicht deployt werden.

Im ConfigMap configmap-vks0041-postgres wird der Hostname, den Read-Write Port, der Read-Only Port, die Datenbank und die Secret-Pfade gespeichert:

```

1 apiVersion: v1
2 kind: ConfigMap
3 metadata:

```

```

4   name: configmap-vks0041-postgres
5   namespace: ksgr-postgresql-maintenance
6 data:
7   configmap-vks0041-postgres.yaml: |-
8     host: "10.0.22.178"
9     port_rw: "5000"
10    port_ro: "5001"
11    database: "postgres"
12    secret_username: "/mnt/etc/secret-volume/secret-vks0041/user"
13    secret_password: "/mnt/etc/secret-volume/secret-vks0041/password"
14

```

Listing 158: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres

Das Deployment sieht entsprechend aus:

```

1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/configmap-
  vks0041-postgres.yaml
2

```

Listing 159: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres Deploy

Für das AUTOVACUUM wird zudem die ConfigMap configmap-vks0041-autovacuum erstellt.
In diesem werden der Pfad zum Node (Host) Filesystem, die Anzahl toter Tupels und dem Filenamen-Präfix:

```

1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: configmap-vks0041-autovacuum
5   namespace: ksgr-postgresql-maintenance
6 data:
7   configmap-vks0041-autovacuum.yaml: |-
8     calc_result: "/mnt/data"
9     max_death_tuples: 1000
10    filename_prefix: "autovacuum-parameter-"
11

```

Listing 160: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum

Das Deployment sieht entsprechend aus:

```

1 kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/configmap-
  vks0041-autovacuum.yaml
2

```

Listing 161: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum Deploy

Das finale Deployment erfolgt mit Hilfe von eines CronJob.
Jeden Tag um Mitternacht wird ein Pod deployt, welcher Skript ausführt.
Dabei müssen erst folgende Dependencis installiert werden:

- psycopg2-binary
- PyYAML
- kubernetes

- pandas

Normalerweise würden diese PyPi-Packages in einem helm Chart als Ressource mitgegeben werden.
 Dann wird das Skript mit dem beiden ConfigMaps ausgeführt.
 Dazu wird das Secret und die ConfigMaps in das Pod-Filesystem gemounted.

```

1 apiVersion: batch/v1
2 kind: CronJob
3 metadata:
4   name: ksgr-maintenance-autovacuum-calulation
5   namespace: ksgr-postgresql-maintenance
6 spec:
7   schedule: "0 0 * * *"
8   jobTemplate:
9     spec:
10       template:
11         metadata:
12           labels:
13             workload: cronjob
14             type: ksgr-postgresql-maintenance
15       spec:
16         containers:
17           - name: ksgr-maintenance-autovacuum-calulation-container
18             image: python:3.11-bookworm
19             command: ["sh", "-c"]
20             args:
21               [
22                 "python -m pip install --proxy http://sproxy.sivc.first-it.ch:8080
psycopg2-binary PyYAML kubernetes pandas;
23                 python /tmp/python/ksgr_postgresql_maintenance_autovacuum_calculation.
py /tmp/conf/postgres/configmap-vks0041-postgres.yaml /tmp/conf/autovacuum/configmap-
vks0041-autovacuum.yaml"
24               ]
25             volumeMounts:
26               - name: ksgr-postgresql-maintenance-autovacuum-calculation
27                 mountPath: /tmp/python
28               - name: configmap-vks0041-postgres
29                 mountPath: /tmp/conf/postgres
30               - name: configmap-vks0041-autovacuum
31                 mountPath: /tmp/conf/autovacuum
32               - name: secret-volume
33                 readOnly: true
34                 mountPath: /mnt/etc/secret-volume/secret-vks0041
35               - name: ksgr-data
36                 mountPath: /mnt/data
37             restartPolicy: Never
38             volumes:
39               - name: ksgr-postgresql-maintenance-autovacuum-calculation
40                 configMap:
41                   name: ksgr-postgresql-maintenance-autovacuum-calculation
42               - name: configmap-vks0041-postgres
43                 configMap:
44                   name: configmap-vks0041-postgres
45               - name: configmap-vks0041-autovacuum

```

```

46     configMap:
47         name: configmap-vks0041-autovacuum
48     - name: ksgr-data
49         hostPath:
50             path: /mnt/data
51     - name: secret-volume
52         secret:
53             secretName: secret-vks0041
54

```

Listing 162: Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-calulation

Das Deployment sieht entsprechend aus:

```

1 kubectl apply -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/
   ksgr_postgresql_maintenance_autovacuum_calculation_cronjob.yaml
2

```

Listing 163: Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-calulation Deploy

XIII Monitoring - PRTG

Bevor das Python-Skript abgelegt werden kann, müssen einige PyPI-Packages auf allen Probe-Server installiert werden:

- json
- socket
- requests

☞ Bei PRTG sind die Status folgendermassen aufgeteilt:

0 = OK, 1 = Warning und 2 = Error.

Dies gilt es zu beachten.

Das Python-Skript KSGR - Patroni - Healthcheck.py sieht wie folgt aus:

```

1 # -*- coding: utf-8 -*-
2
3 """
4 Created: 2024-05-05
5
6 @author: Michael Graber, Kantonsspital Graubuenden, Switzerland
7 """
8 import json
9 import socket
10 import sys
11
12 import requests
13 from paesslerag_prtg_sensor_api.sensor.result import CustomSensorResult
14 from paesslerag_prtg_sensor_api.sensor.units import ValueUnit
15
16
17 # node_state = 0 # ok

```

```

18 #     node_state = 1 # warning
19 #     node_state = 2 # error
20
21 def check_patroni_cluster(host, replication_mode, lag_warning, lag_error):      # Get the
22     IP
23
24     resolution = socket.gethostbyname_ex(host)
25     ip_address_block = resolution[2]
26     ip_address = ip_address_block[0]
27     print(str(ip_address))
28
29     # Variable declaration
30     patroni_comment = ''
31     cluster_state = 2
32     cluster_unlocked = 2
33     failsafe_mode_is_active = 2
34     sync_replication_state = 2
35     replication_lage_state = 2
36
37     # Get the Cluster State
38     patroni_cluster_state_url = 'http://'+ip_address+':8008/patroni'
39     patroni_cluster_state_response = requests.get(patroni_cluster_state_url)
40     patroni_cluster_state_response_json = patroni_cluster_state_response.json()
41     patroni_cluster_state_response_code = patroni_cluster_state_response.status_code
42
43     patroni_cluster_name = patroni_cluster_state_response_json.get('patroni').get('name')
44     patroni_cluster_state = patroni_cluster_state_response_json.get('state')
45     patroni_role = patroni_cluster_state_response_json.get('role')
46
47     if 'paused' in patroni_cluster_state_response_json:
48         patroni_cluster_paused = patroni_cluster_state_response_json.get('paused')
49     else:
50         patroni_cluster_paused = False
51
52     if patroni_cluster_state == 'running' and patroni_cluster_paused:
53         cluster_state = 1
54         patroni_comment = '(Cluster '+patroni_cluster_name+', paused)'
55     elif not(patroni_cluster_state):
56         cluster_state = 2
57         patroni_comment = '(Cluster '+patroni_cluster_name+', not running)'
58     else:
59         cluster_state = 0
60
61     if 'cluster_unlocked' in patroni_cluster_state_response_json:
62         patroni_cluster_unlocked = patroni_cluster_state_response_json.get(
63             'cluster_unlocked')
64         if 'failsafe_mode_is_active' in patroni_cluster_state_response_json:
65             patroni_failsafe_mode_is_active = patroni_cluster_state_response_json.get(
66                 'failsafe_mode_is_active')
67             if patroni_cluster_unlocked == True and patroni_failsafe_mode_is_active ==
68                 True:
69                 patroni_comment += ', (DCS Failsafe Mode)'
70                 cluster_unlocked = 2
71                 failsafe_mode_is_active = 2
72             else:

```

```

68         cluster_unlocked = 2
69         failsafe_mode_is_active = 0      else:
70     cluster_unlocked = 0
71     failsafe_mode_is_active = 0
72
73 #   Get the Cluster Node States
74 patroni_nodes_state_url = 'http://'+ ip_address + ':8008/cluster'
75 patroni_cluster_nodes_response = requests.get(patroni_nodes_state_url)
76 patroni_cluster_nodes_response_json = patroni_cluster_nodes_response.json()
77 patroni_cluster_nodes_response_code = patroni_cluster_nodes_response.status_code
78
79 replica_count = 0
80 sync_replica_count = 0
81 if patroni_role == 'master':
82     patroni_replication = patroni_cluster_state_response_json.get('replication')
83     for patroni_replication_member in patroni_replication:
84         patroni_sync_state = patroni_replication_member.get('sync_state')
85         if patroni_sync_state == replication_mode:
86             sync_replica_count = sync_replica_count + 1
87             replica_count = replica_count + 1
88 else:
89     cluster_members = patroni_cluster_nodes_response_json.get('members')
90     for cluster_node in cluster_members:
91         node_role = cluster_node.get('role')
92         if node_role == 'leader':
93             leader_api = cluster_node.get('api_url')
94             patroni_leader_state_response = requests.get(leader_api)
95             patroni_leader_state_response_json = patroni_leader_state_response.json()
96             patroni_leader_state_response_code = patroni_leader_state_response.
97 status_code
98
99     patroni_replication = patroni_leader_state_response_json.get('replication')
100    )
101    for patroni_replication_member in patroni_replication:
102        patroni_sync_state = patroni_replication_member.get('sync_state')
103        if patroni_sync_state == replication_mode:
104            sync_replica_count = sync_replica_count + 1
105            replica_count = replica_count + 1
106
107    if sync_replica_count == sync_replica_count:
108        sync_replication_state = 0
109    elif sync_replica_count > 0:
110        sync_replication_state = 1
111        patroni_comment += ' (Not all Replicas run in sync standby mode),'
112    else:
113        sync_replication_state = 2
114        patroni_comment += ' (None of the Replicas run in sync standby mode),'
115
116 #   Get the Replication lag warning
117 lag_warning_url = 'http://'+ ip_address + ':8008//replica?lag=' + lag_warning
118 lag_warning_response = requests.get(lag_warning_url)
119 lag_warning_response_code = lag_warning_response.status_code

```

```

120     lag_error_url = 'http://' + ip_address + ':8008//replica?lag=' + lag_error
121     lag_error_response = requests.get(lag_error_url)      lag_error_response_code =
122     lag_error_response.status_code
123
124     if patroni_role == 'master':
125         replication_lage_state = 0
126     else:
127         if lag_warning_response_code != 200:
128             replication_lage_state = 1
129             patroni_comment += ' (replication lag over ' + lag_warning + ')'
130         elif lag_error_response_code != 200:
131             replication_lage_state = 2
132             patroni_comment += ' (replication lag over ' + lag_error + ')'
133         else:
134             replication_lage_state = 0
135
136     csr = CustomSensorResult(text="This sensor runs on %s" % data["host"])
137
138     # Patroni Cluster State
139     channel_name = "PATRONI CLUSTER - " + patroni_cluster_name
140     csr.add_primary_channel(name=patroni_cluster_name,
141                             value=cluster_state,
142                             unit=ValueUnit.CUSTOM,
143                             is_float=False,
144                             is_limit_mode=True,
145                             limit_max_warning=0.75,
146                             limit_max_error=1.5,
147                             limit_warning_msg=channel_name + " Warning",
148                             limit_error_msg=channel_name + " Error")
149
150     channel_name = "CLUSTER UNLOCK"
151     csr.add_channel(name=channel_name,
152                     value=cluster_unlocked,
153                     unit=ValueUnit.CUSTOM,
154                     is_float=False,
155                     is_limit_mode=True,
156                     limit_max_warning=0.75,
157                     limit_max_error=1.5,
158                     limit_warning_msg=channel_name + " Warning",
159                     limit_error_msg=channel_name + " Error")
160
161     channel_name = "FAILSAFE MODE ACTIVITY"
162     csr.add_channel(name=channel_name,
163                     value=failsafe_mode_is_active,
164                     unit=ValueUnit.CUSTOM,
165                     is_float=False,
166                     is_limit_mode=True,
167                     limit_max_warning=0.75,
168                     limit_max_error=1.5,
169                     limit_warning_msg=channel_name + " Warning",
170                     limit_error_msg=channel_name + " Error")
171
172     channel_name = "REPLICATION STATE"
173     csr.add_channel(name=channel_name,

```

```

173         value=sync_replication_state ,
174         unit=ValueUnit.CUSTOM ,                                     is_float=False ,
175         is_limit_mode=True ,
176         limit_max_warning=0.75 ,
177         limit_max_error=1.5 ,
178         limit_warning_msg=channel_name + " Warning" ,
179         limit_error_msg=channel_name + " Error")
180
181     channel_name = "REPLICATION LAG"
182     csr.add_channel(name=channel_name ,
183                     value=replication_lage_state ,
184                     unit=ValueUnit.CUSTOM ,
185                     is_float=False ,
186                     is_limit_mode=True ,
187                     limit_max_warning=0.75 ,
188                     limit_max_error=1.5 ,
189                     limit_warning_msg=channel_name + " Warning" ,
190                     limit_error_msg=channel_name + " Error")
191
192     # Overwrite the Sensor Text and create a custom Warning/Error Message
193     patroni_comment = patroni_comment.strip()
194     csr.text = "This sensor runs on %s" % data["host"] + ' ' + patroni_comment
195
196     return csr
197
198 if __name__ == "__main__":
199     try:
200         # Get Parameters
201         data1 = sys.argv[1]
202         data = json.loads(data1)
203         data_overgiven = json.loads(data1)
204
205         # Get the overgiven Parameter String and cast to a Dictionary
206         params = data_overgiven["params"]
207         perams = dict(param.split('=') for param in params.split(',',)))
208
209         # Get the used Binary
210         replication_mode = perams["replication_mode"]
211         lag_warning = perams["lag_warning"]
212         lag_error = perams["lag_error"]
213
214         # Call Function / Method
215         csr = check_patroni_cluster(data["host"], replication_mode, lag_warning,
216                                     lag_error)
217
218         # Return the Value
219         print(csr.json_result)
220
221     except Exception as e:
222         csr = CustomSensorResult(text="Python Script execution error")
223         csr.error = "Python Script execution error: %s" % str(e)
224         print(csr.json_result)

```

Listing 164: Monitoring - KSGR - Patroni - Healthcheck.py

Das Python-Skript muss auf allen PRTG Probes, welches die entsprechenden Sensoren ausführt, angelegt werden,

XIV Exkurs Architekturen - Umsysteme und Prinzipien

XIV.I Raft-Consensus

Raft ist eine Kombination aus einer Log-Replikation und einer State Machine[90].

Ein Node kann dabei einen von drei States einnehmen:

Leader

Empfängt als einziger Requests von den Clients und hat die Hoheit über die Transaktionen im Log

Candidate

Wenn ein Leader ausgefallen ist oder für einen Follower nicht mehr innerhalb eines zufälligen Timeouts erreichbar ist,
ernennt dieser sich zum Kandidaten als neuer Leader und wählt sich selbst.
Dazu versendet er alle seine Informationen inklusive seinem Log.
Gewählt wird dann jener Candidate, der ein konsistentes, vollständiges und jüngste Log besitzt.

Follower

Follower können nur Commands empfangen und Response Messages senden.

Das Prozedere für die Bestimmung eines Leaders kann viele Ursachen haben.

Etwa weil ein System neu gestartet wird oder ein Leader nicht mehr verfügbar oder erreichbar ist.

Dann werden folgende Schritte vorgenommen:

1. Alle Nodes starten als Follower und warten auf einen Leader
2. Wenn kein Leader vorhanden ist, startet mindestens ein Follower die Elektion eines Leaders.
Mindestens eine Request Vote wird versendet.
3. Die Follower wählen dann ihren neuen Leader aus.
4. Der neue Leader sendet nun Append Entries als Heartbeat-Message und Anweisungen an die Follower, Log Einträge zu replizieren.

Das zufällige Heartbeat-Timeout der Follower verhindert dabei,
dass sich alle Follower gleichzeitig als Candidate ausrufen und sich selber wählen.
Sonst hätte man eine Split-Brain Situation.

Ist ein Leader aktiv, wird der Log konsens wie folgt ausgeführt:

1. Der Leader erhält einen Request von einem Client Daten zu schreiben
2. Der Leader legt dieses als neuen Eintrag ein

3. Der Leader sendet Append Entry Messages zu den Followern
4. Eine Mehrheit der Follower müssen nun bestätigen,
dass sie diese erhalten und in ihr Log geschrieben haben.
5. Der Leader schreibt nun ein Commit in seine State Machine
6. Der Leader sendet nun Append Entry Messages, dass er die Einträge committed hat.
Der Leader befiehlt den Followern, es ihm gleichzutun
7. Nun ist eine konsistente Transaktion auf den Cluster geschrieben

Dem Log wird immer der Index des vorangegangenen Eintrags gesendet.

Fällt ein Follower aus und wird später wieder dem Quorum hinzugefügt,

meldet er einen Fehler, wenn er neue Daten erhält, ihm fehlen ja Einträge.

Der Leader sendet dem Follower also den Eintrag.

Fehlen weitere vorangehende Einträge, wird der Vorgang solange ausgeführt,
bis wieder ein konsistenter Stand erreicht ist.

Das System ist so selbstheilend.

XIV.II local-path-provisioner

local-path-provisioner wartet auf einen PVC (Persistence Volume Claim).

Sobald ein Pod existiert, wird auf dem Worker Node ein Volume erzeugt, welches normalerweise unter dem Pfad `var/volumes` auf dem Node zu finden ist.

Anschliessend wird ein PV (Persistence Volume) und dem Volume auf dem Node angefügt.

Anschliessend wird das PV an den PVC gebunden:

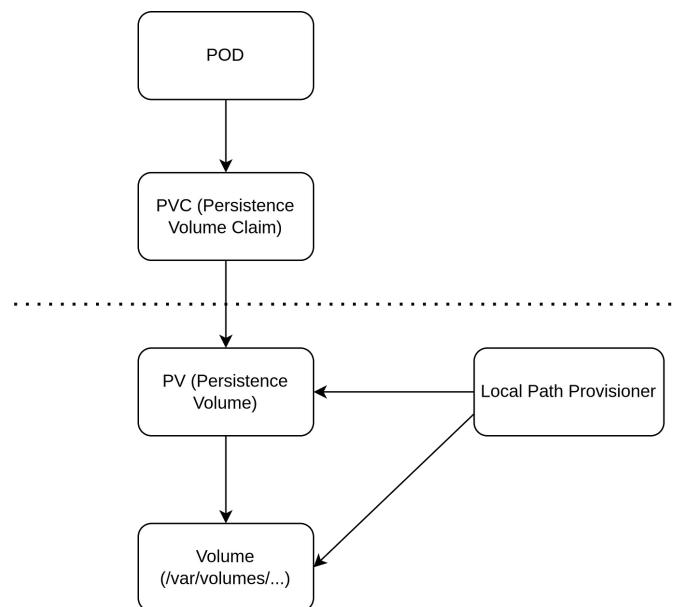


Abbildung L: local-path-provisioner[37]