

Diplomarbeit Technik und Wirtschaftsinformatik
2023-2024

PostgreSQL HA Cluster - Konzeption und Implementation

22.11.2023

Management Summary

Diplomarbeit Michael Graber

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	10
1.2 Zieldefinition	14
1.3 Abhängigkeiten	20
1.4 Risikomanagement	22
1.4.1 Riskcontrolling	25
1.5 Vorgehensweise und Methoden	28
2 Projektmanagement	29
2.1 Projektplanung	29
2.1.1 Projektcontrolling	30
2.1.2 GANTT-Diagramm	32
2.2 Expertengespräche	33
3 Umsetzung	34
3.1 Evaluation	34
3.1.1 Exkurs Architektur	34
3.1.2 Erheben und Gewichten der Anforderungen	40
3.1.3 Testziele erarbeiten	44
3.1.4 PostgreSQL Benchmarking	47
3.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen	52
3.1.6 Vorauswahl	75
3.1.7 Installation verschiedener Lösungen	75
3.1.8 Testing Evaluationssysteme	93
3.1.9 Gegenüberstellung der Lösungen	96
3.1.10 Entscheid	113
3.2 Aufbau und Implementation Testsystem	114
3.2.1 Architektur	114
3.2.2 Bereitstellen der Grundinfrastruktur	116
3.2.3 Installation und Konfiguration PostgreSQL HA Cluster	116

3.2.4	Technical Review der Umgebung	121
3.3	Testing	125
3.3.1	Protokollierung	128
3.3.2	Review und Auswertung	129
3.4	Maintenance-Tool	129
3.5	Monitoring	129
3.6	Troubleshooting und Lösungsfindung	130
3.7	Ausblick	131
4	Resultate	132
4.1	Zielüberprüfung	132
4.2	Schlussfolgerung	132
4.3	Weiteres Vorgehen / offene Arbeiten	132
4.4	Persönliches Fazit	132
Abbildungsverzeichnis		133
Tabellenverzeichnis		137
Listings		139
Literatur		144
Abkürzungen		150
Glossar		152
Anhang		i
I	Arbeitsrapport	i
II	Protokoll - Fachgespräche	iii
II.I	Fachgespräch 27.03.2024	iv
II.II	Fachgespräch 27.03.2024	v
II.III	Fachgespräch 27.03.2024	vi
II.IV	Fachgespräch 27.03.2024	vii
III	Statusbericht	viii
III.I	Status Report 1	ix
III.II	Status Report 2	xi
III.III	Status Report 3	xiii
IV	Kommentare / Anmerkungen	xvi
V	Evaluation	xix
V.I	Maintenance - CloudNativePG	xix
V.II	Maintenance - Patroni	xxii

V.III	Maintenance - StackGres - Citus	xxv
V.IV	Maintenance - YugabyteDB	xxxi
VI	Evaluationssysteme - Installation	xxxiv
VI.I	rke2	xxxiv
VI.II	yugabyteDB	xli
VI.III	sks9016 - yugabyteDB	lxvi
VI.IV	Patroni	lxvi
VI.V	Stackgres mit Citus	lxxxvi
VII	Evaluationssysteme - Benchmarking	civ
VII.I	yugabyteDB	civ
VII.II	Patroni	cvi
VII.III	StackGres - Citus	cvi
VIII	Evaluationssysteme - Testing	cx
VIII.I	StackGres - Citus	cx
VIII.II	YugabyteDB	cxiii
IX	Testsystem - Installation	c xvii
IX.I	Prequenteries	c xviii
IX.II	Deployment	c xix
IX.III	Maintenance	c xxxi
IX.IV	Prequenteries - Erweiterungstests	c xxxii
IX.V	Haproxy erweitern	c xxxii
IX.VI	Patroni Node	c xxxii
X	Testsystem - Testing	c xxxii
XI	Exkurs Architekturen - Umsysteme und Prinzipien	c xxxii
XI.I	Raft-Konsensus	c xxxii
XI.II	local-path-provisioner	c xxxii
XII	Python Utils	c xxxii
XII.I	zotero.py	c xxxii
XII.II	zotero_bibtex_configuration.yaml	c xxxvii
XII.III	zotero_biblatex_keystore.yaml	c xxxvii
XII.IV	riskmatrix.py	cxliii
XII.V	riskmatrix_plotter_conf.yaml	cxlvii
XII.VI	riskmatrix_xy_axis_tuple_matrix.yaml	cl
XII.VII	cost_benefit_diagram.py	cli
XII.VIII	cost_benefit_diagram_plotter_conf.yaml	c liii
XII.IX	pandas_dataframe_to_latex_table.py	c liii
XII.X	csv_to_latex_diplomarbeit.yaml	c lix
XII.XI	pandas_data_chart_plotter.py	c lxxvii
XII.XII	pandas_data_chart_plotter_conf.yaml	c lxxxii

1 Einleitung

1.1 Ausgangslage und Problemstellung

1.1.1 Das Kantonsspital Graubünden

Das Kantonsspital Graubünden 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

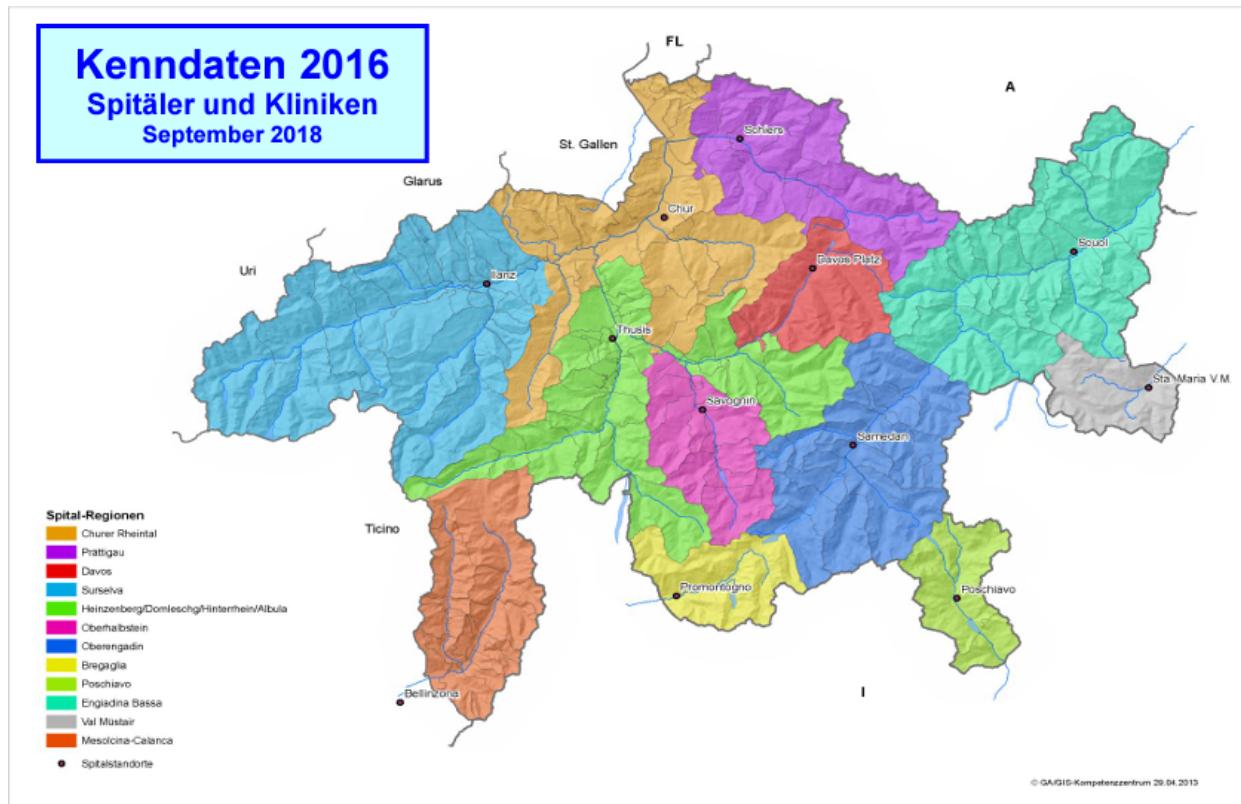


Abbildung 1.1: Spitalregionen Kanton Graubünden[61]

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



Abbildung 1.2: Wahlkreise Kanton St. Gallen[88]

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

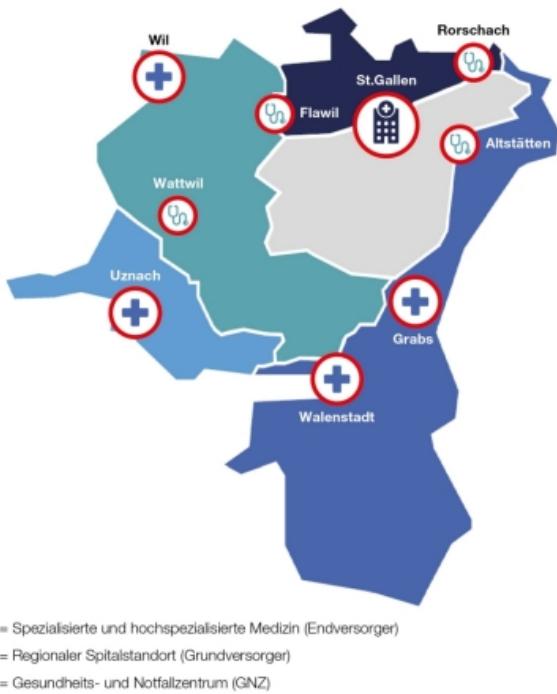


Abbildung 1.3: Spitalregionen / Spitalstrategie Kanton St. Gallen[55]

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

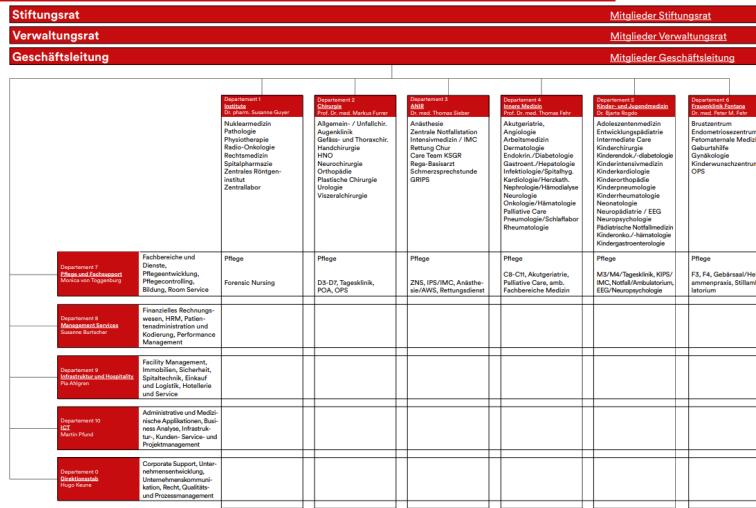


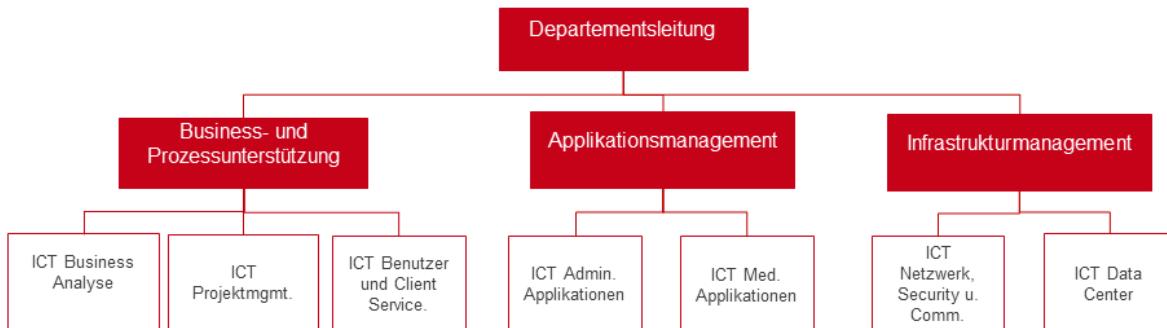
Abbildung 1.4: Organigramm Kantonsspital Graubünden

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, haben in den allermeisten Fällen ihre Daten in Datenbanksysteme speichern. Entsprechend der Vielfalt der Applikationen existieren auch eine vielzahl an Datenbanksystemen und Versionen.

Basierend auf der Liste *DB-Engines Ranking*[52] 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	Ko...
Oracle Database	Relational, NoSQL, OLAP	Ja	
MySQL	Relational	Ja	
Microsoft SQL Server	Relational, NoSQL, OLAP	Nein	We...
PostgreSQL	Relational, NoSQL	Ja	
MongoDB	NoSQL	Ja	
Redis	Key-value	Ja	
Elasticsearch	Search engine	Ja	
IBM DB2	Relational	Ja	
SQLite	Relational	Nein	Lob...
Microsoft Access	Relational	Nein	Nic...
Snowflake	Relational	Ja	
Cassandra	Relational	Ja	
MariaDB	Relational	Ja	
Splunk	Search engine	Ja	
Microsoft Azure SQL Database	Relational, NoSQL, OLAP	Nein	Da...

Tabelle 1.1: Inventarisierte DBMS

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

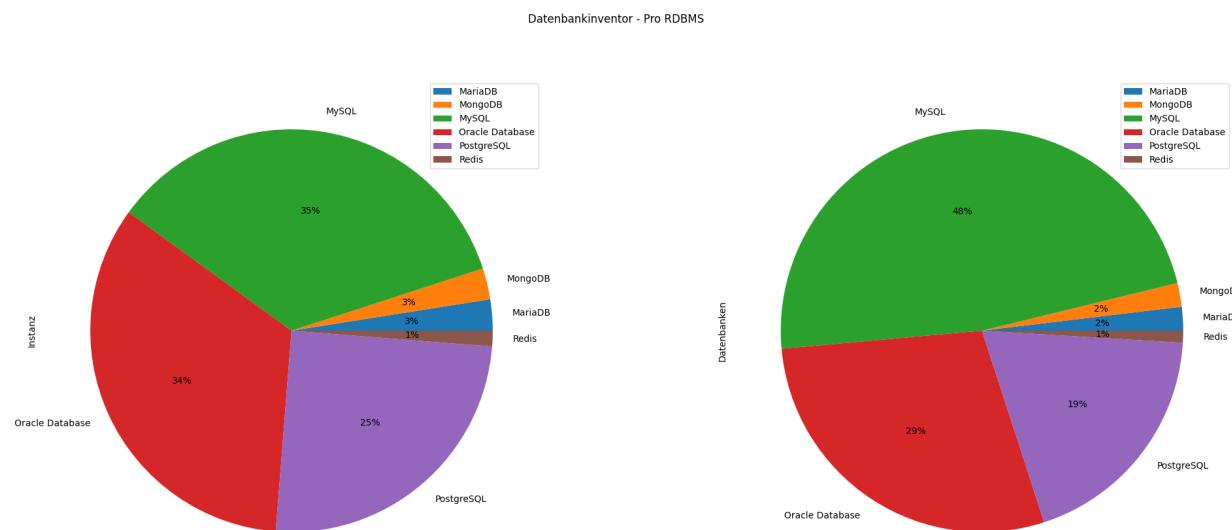


Abbildung 1.6: Datenbanken - Aufgeschlüsselt nach RDBMS

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
	Oracle Database	0	1	1
	PostgreSQL	4	8	8
	Redis	0	1	1
Windows Server	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

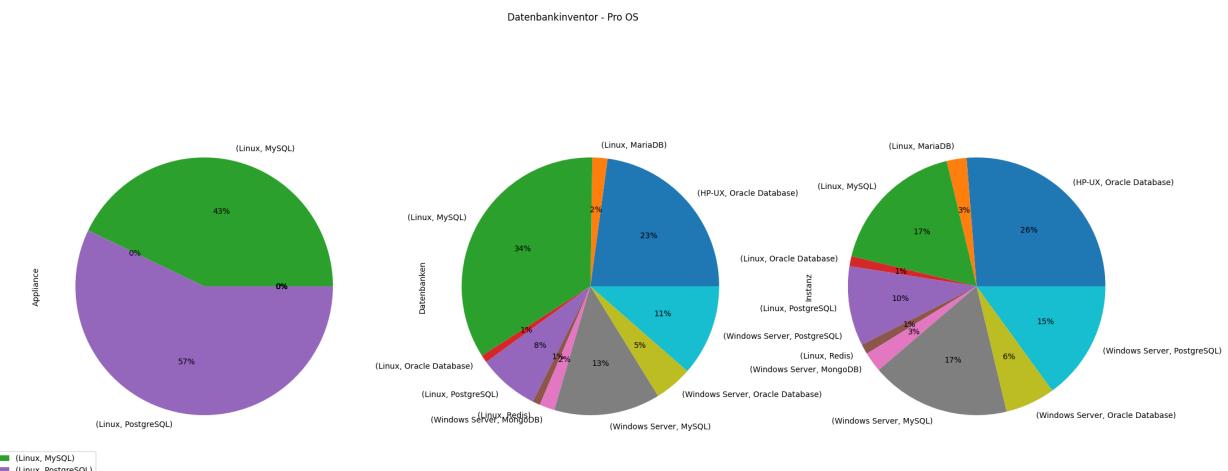


Abbildung 1.7: Datenbanken - Aufgeschlüsselt nach Betriebssystem

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 1'200 Datenpunkte abgefragt und xxx in die Datenbank und wird im Laufe der Zeit mehrere Terrabyte gross werden.

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 viele Aufwände 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						
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	
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, dass keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Die Auswirkung ist abhängig von den Systemen und deren Sicherheitsmaßnahmen.	
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, dass keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Sofern kein Ersatz vorhanden ist, kann dies zu einer Verlust von Daten führen.	
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.	Daher sind Fehlfunktionen von Systemen zu erwarten.	
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 Auswirkung ist abhängig von der Anzahl der betroffenen Personen.	
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 Anzahl der technischen Ressourcen zu gross ist, kann dies zu einem Systemausfall führen.	
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.	Gefährlich, da es zu schwerwiegenden Fehlern führen kann.	
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.	Daraus resultiert eine Menge an Sicherheitslücken.	
8 A, I		G0.45	Datenverlust	Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, dass 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, dass für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde.	Aus dem Datenverlust kann wiederholter Datenverlust resultieren.	
				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.	Erstens kann dies zu Datenverlust führen. Die zweite Gefahr dadurch ist, dass die Backups nicht mehr gültig sind.	

Tabelle 1.4: Risiko-Matrix aktuelle Situation

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
- Backupkonzept erarbeiten
- Berechtigungskonzept erarbeiten und einführen

Mit diesen Massnahmen lassen sich die Risiken senken.

Die Risiken werden wie folgt gesenkt:

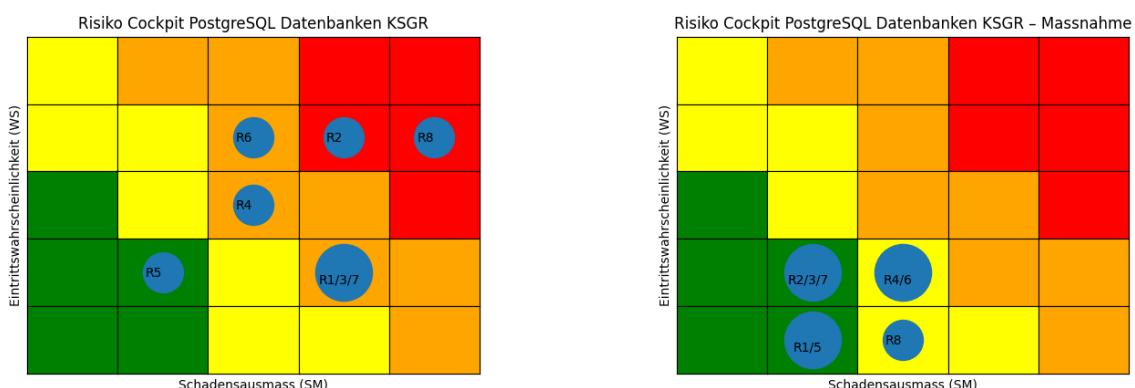


Abbildung 1.8: Risikomanagement PostgreSQL

1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [71, 77] 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 auf den Primären DB-Node zurückgeschrieben werden beim Failover Restore Die Datenmenge von 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 Überwacht 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 inoperante SQL Statements ausgelesen werden und zum Teil werden auch Informationen zum Tuning geliefert[50]. 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[2]. 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[50]. 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[2].	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 der 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 werden, 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 Secondary Nodes muss möglich sein 3. Der Fallback auf den Primary 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
Logplattform / SIEM System		Wird neu Ausgeschrieben. Produkt zurzeit nicht definiert
Usermanagement	Microsoft Active Directory	

Tabelle 1.8: Gegebene Systeme

Daraus ergeben sich nach nach Züst, Troxler 2002[103] folgende Abgrenzungen:

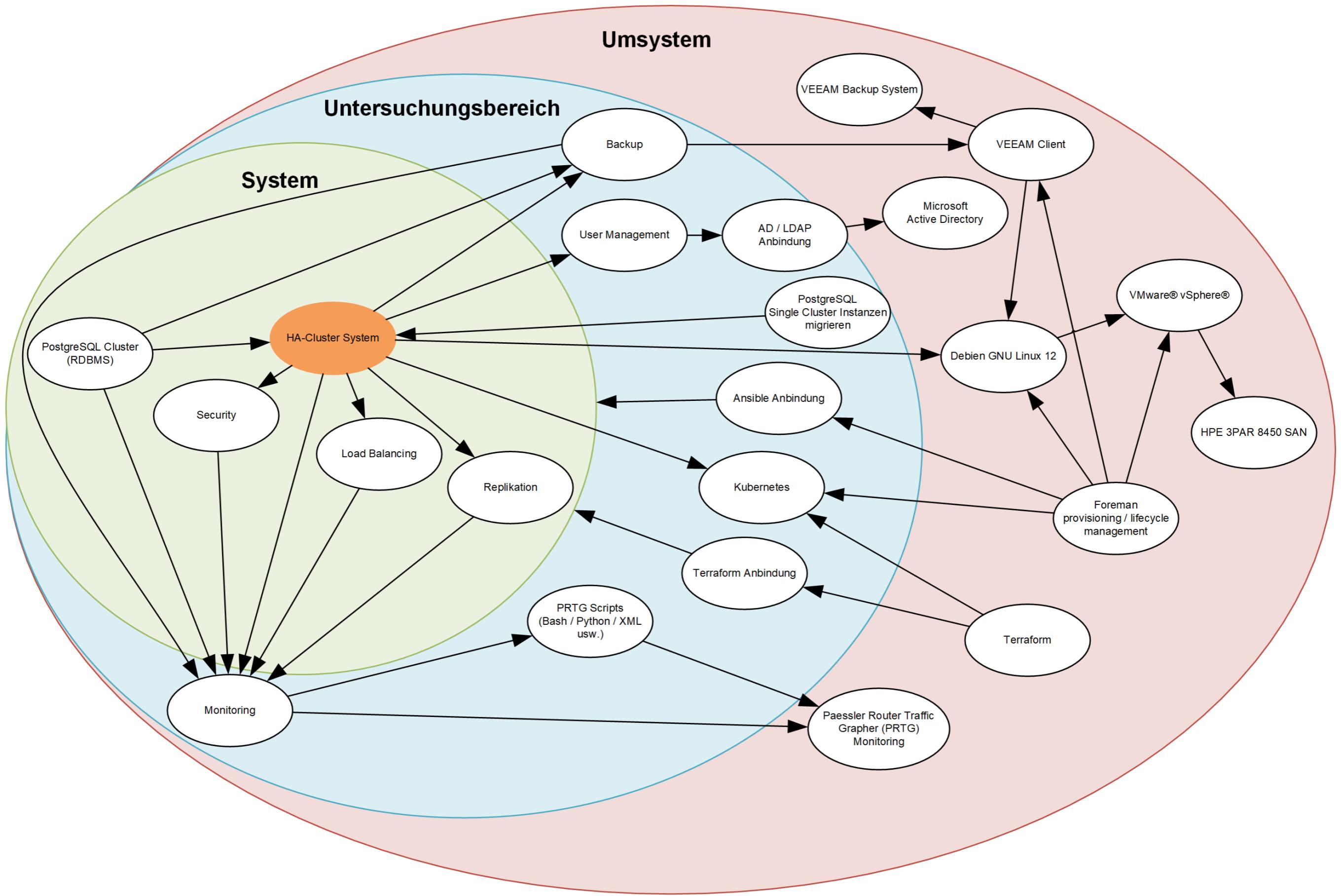


Abbildung 1.9: Systemabgrenzung

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[3]	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?	WS	SM	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 vor gehalten 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:

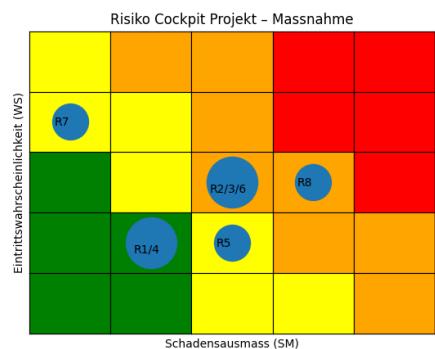
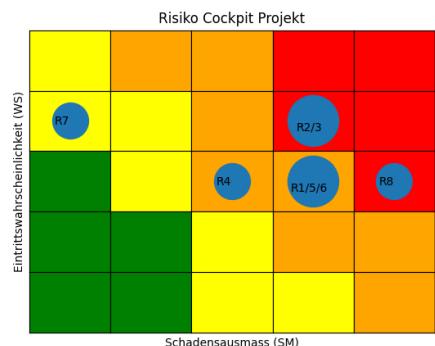


Abbildung 1.10: Risikomanagement Projekt

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[3] 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[3] 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

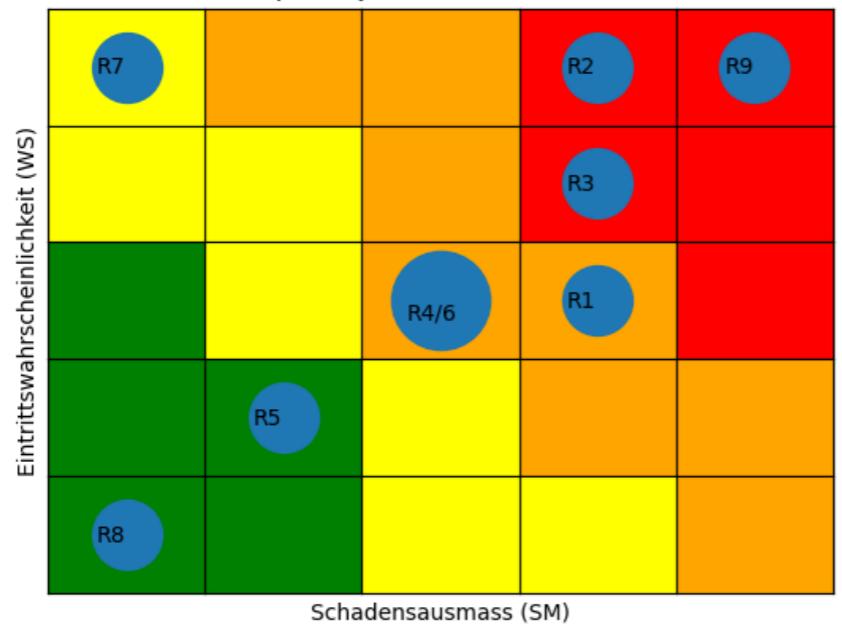


Abbildung 1.11: Riskikomatrix - Assessment 21.03.2024

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[3] 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

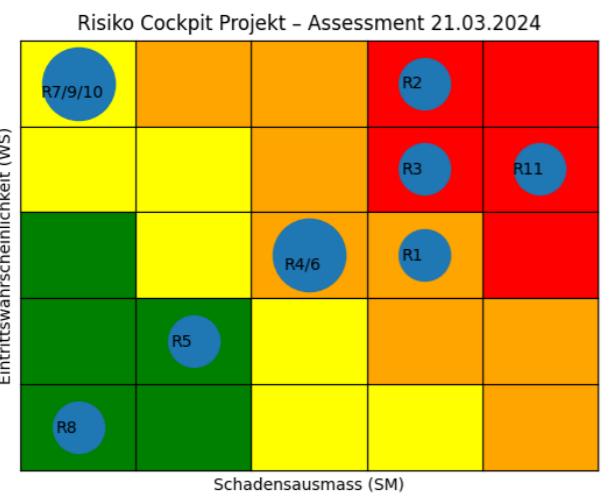


Abbildung 1.12: Riskikomatrix - Assessment 29.04.2024

1.5 Vorgehensweise und Methoden

2 **Projektmanagement**

2.1 **Projektplanung**

2.1.1 Projektcontrolling

	Phase	Subphase	Dauer [h]	Geplante Dauer [h]	Verbleibende Zeit [h]
0	1. Expertengespräch	1. Expertengespräch	1.0	1.0	0.0
1	2. Expertengespräch	2. Expertengespräch	1.2	1.0	-0.2
2	Aufbau und Implementation Testsystem	Basisinfrastruktur	3.0	4.0	1.0
3	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	10.0	20.0	10.0
4	Aufbau und Implementation Testsystem	Technical Review	2.0	3.0	1.0
5	Dokumentation	Dokumentation	38.8	80.0	41.2
6	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	51.2	16.0	-35.2
7	Evaluation	Anorderungskatalog	4.5	16.0	11.5
8	Evaluation	Gegenüberstellung	0.5	8.0	7.5
9	Evaluation	Variantenentscheid	1.0	4.0	3.0
10	Evaluation	Vorbereitung Benchmarking	5.0	4.0	-1.0
11	Letztes Expertengespräch	Letztes Expertengespräch	0.0	1.0	1.0
12	Puffer	Puffer	0.0	16.0	16.0
13	Resultate	Persönliches Fazit	0.0	2.0	2.0
14	Resultate	Schlussfolgerung	0.0	2.0	2.0
15	Resultate	Weiteres Vorgehen / offene Arbeiten	0.0	1.0	1.0
16	Resultate	Zielüberprüfung	0.0	2.0	2.0
17	Testing	Protokollierung	0.0	4.0	4.0
18	Testing	Review und Auswertung	0.0	2.0	2.0
19	Testing	Testing Testsystem	0.0	8.0	8.0
20	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	11.0	8.0	-3.0
Total			129.2	203.0	73.8

Tabelle 2.1: Projektcontrolling

Projektcontrolling

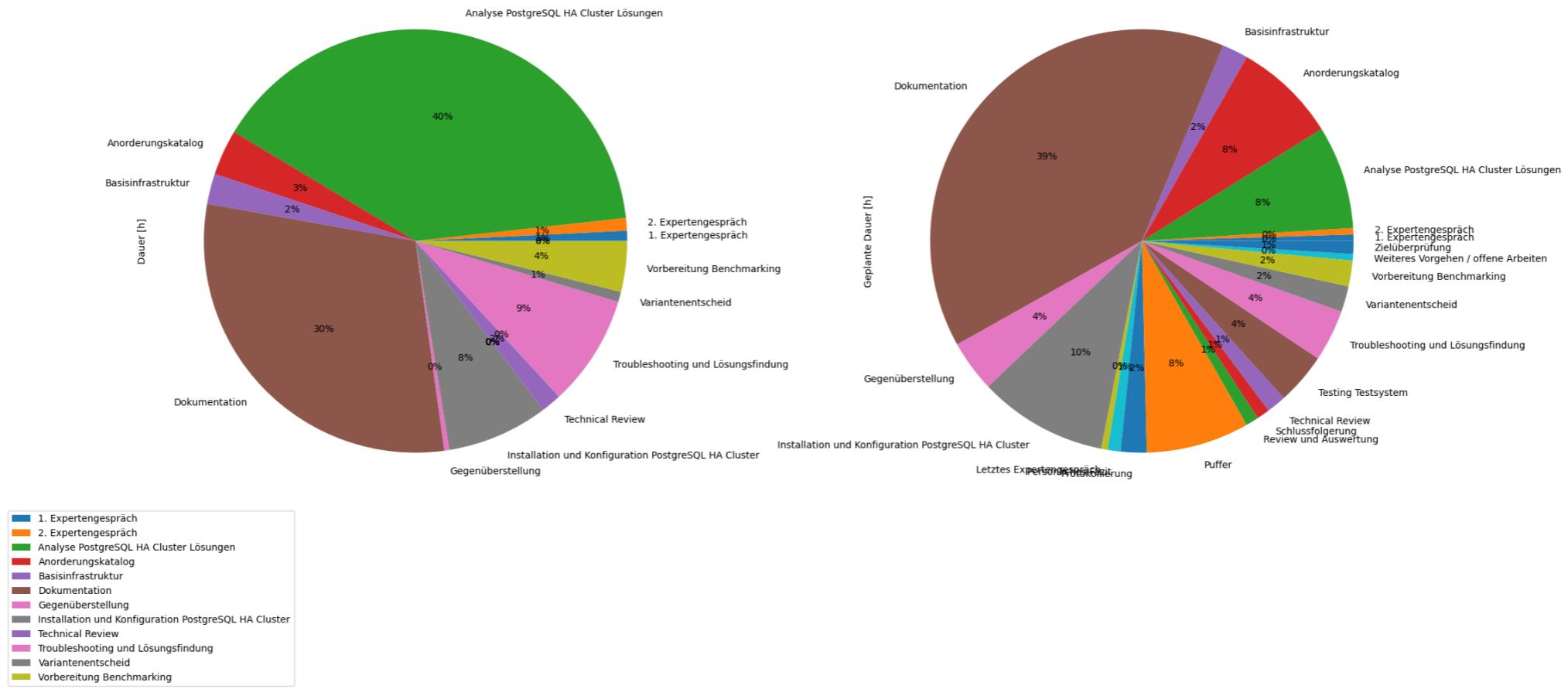
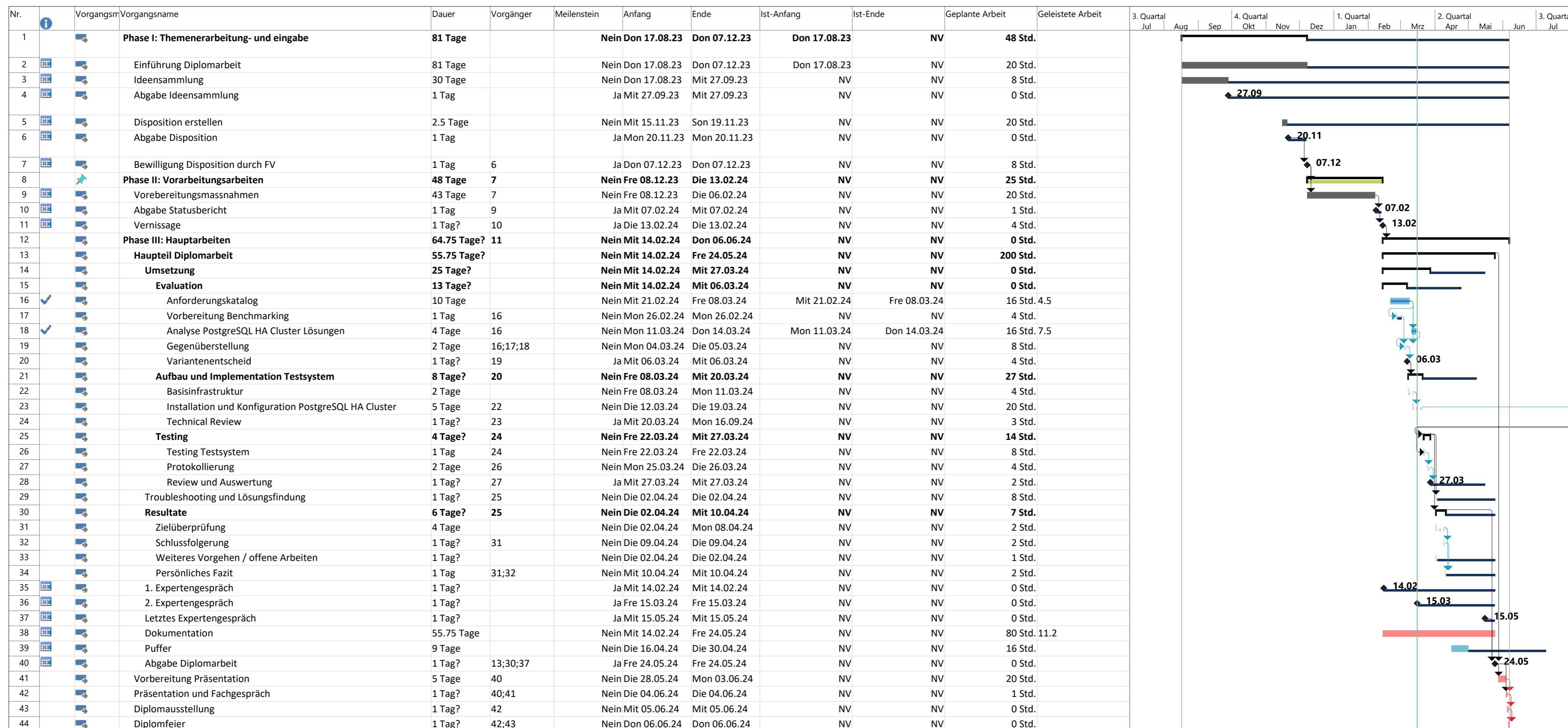


Abbildung 2.1: Projektcontrolling



© Kantonsspital Graubünden | PostgreSQL HA Cluster - Konzeption und Implementatio

Projekt: Diplomarbeit - PostgreSQL HA Cluster	Vorgang	Projektsammelvorgang
Datum: Sam 16.03.24	Unterbrechung	Inaktiver Vorgang
	Meilenstein	Inaktiver Meilenstein
	Sammelvorgang	Inaktiver Sammelvorgang

2.2 Expertengespräche

Folgende Expertengespräche fanden statt:

Fachgespräch	Datum	Fachexperte	Nebenexperte	Studenten	Bemerkungen
1	14.02.2024	Norman Süsstrunk	-	Michael Graber 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 Graber	

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.

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

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 meistens 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 mittels Replikation eine Primary/Standby-Architektur aufweisen. Man kann mittels eines SQL Proxys ein gewisses Mass an Load Balancing betreiben, hat aber immer noch das Problem das es einen Primary Node gibt auf dem beschrieben wird. Monolithische Systeme sind daher nicht Cloud Native.

Nur verteilte Systeme, sogenannte Distributed SQL wiederum sind Cloud Native

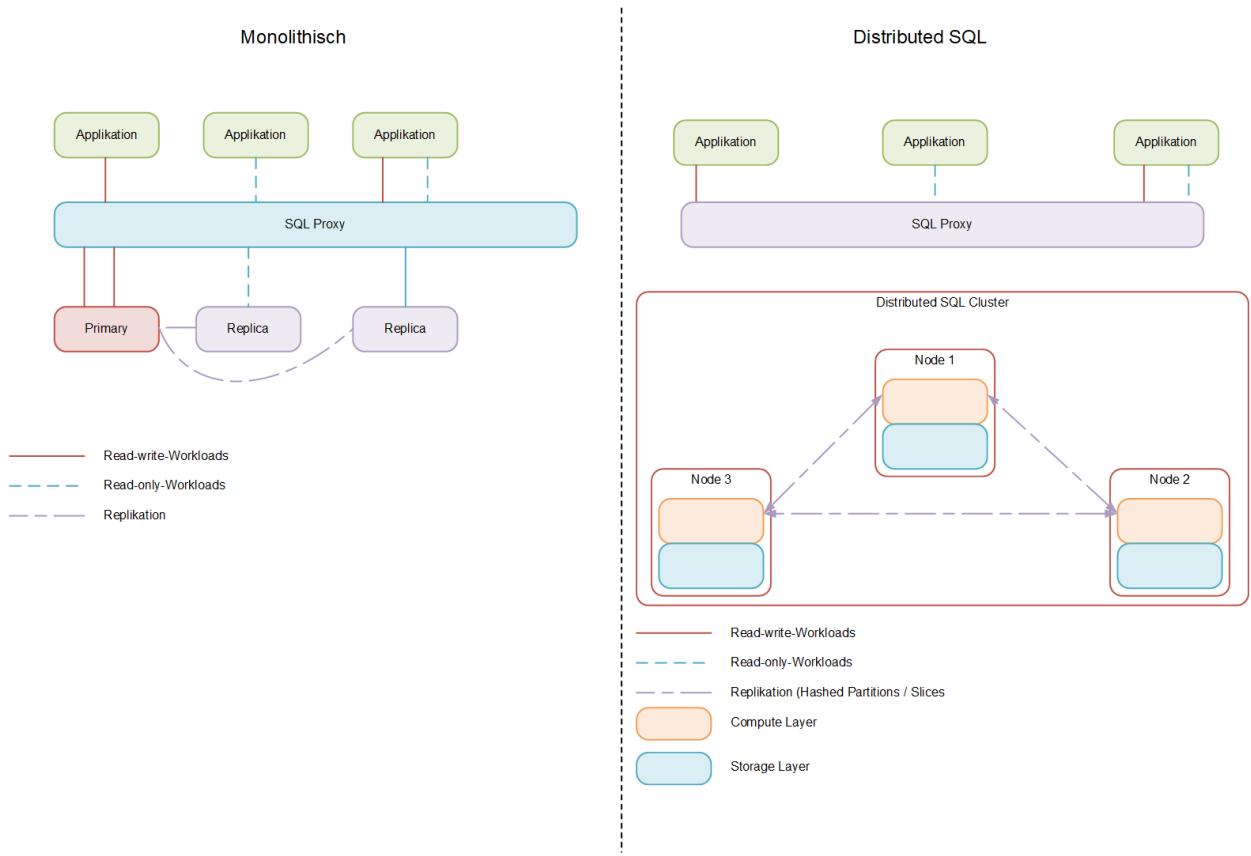


Abbildung 3.3: Monolithische vs. verteilte SQL Systeme

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 repliziert werden, dies nennt man Replikation[75]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

Synchrone Replikation

Wenn bei einer Synchroen 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 Committed ist. Bis zu diesem Moment ist die Transaktion nicht als Committed.

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 Committed 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 Commitmend 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 [73, 85, 68] beschrieben werden. Es zeigt auf, ab wie vielen Nodes die Fehlertoleranz erhöht wird und wie sich der Representative 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[57].

Konsistenz - Consistency

Die Datenbank ist konsistent, alle Clients sehen gleichzeitig die gleichen Daten unabhängig von 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.

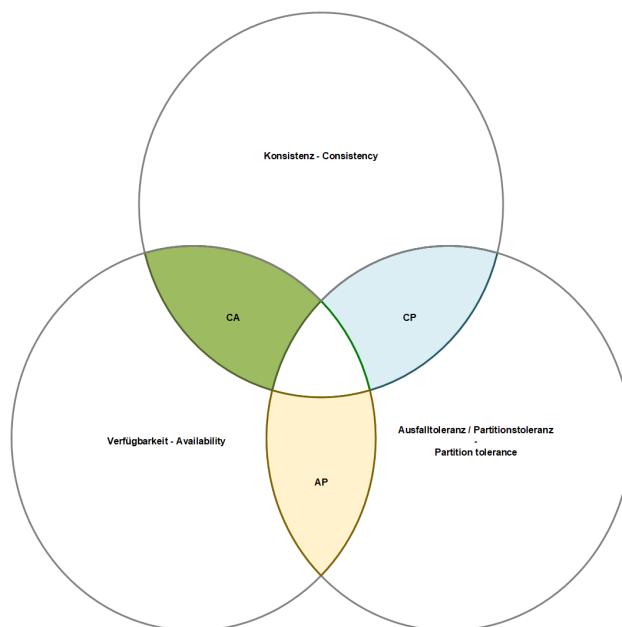


Abbildung 3.4: CAP-Theorem

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[70]:

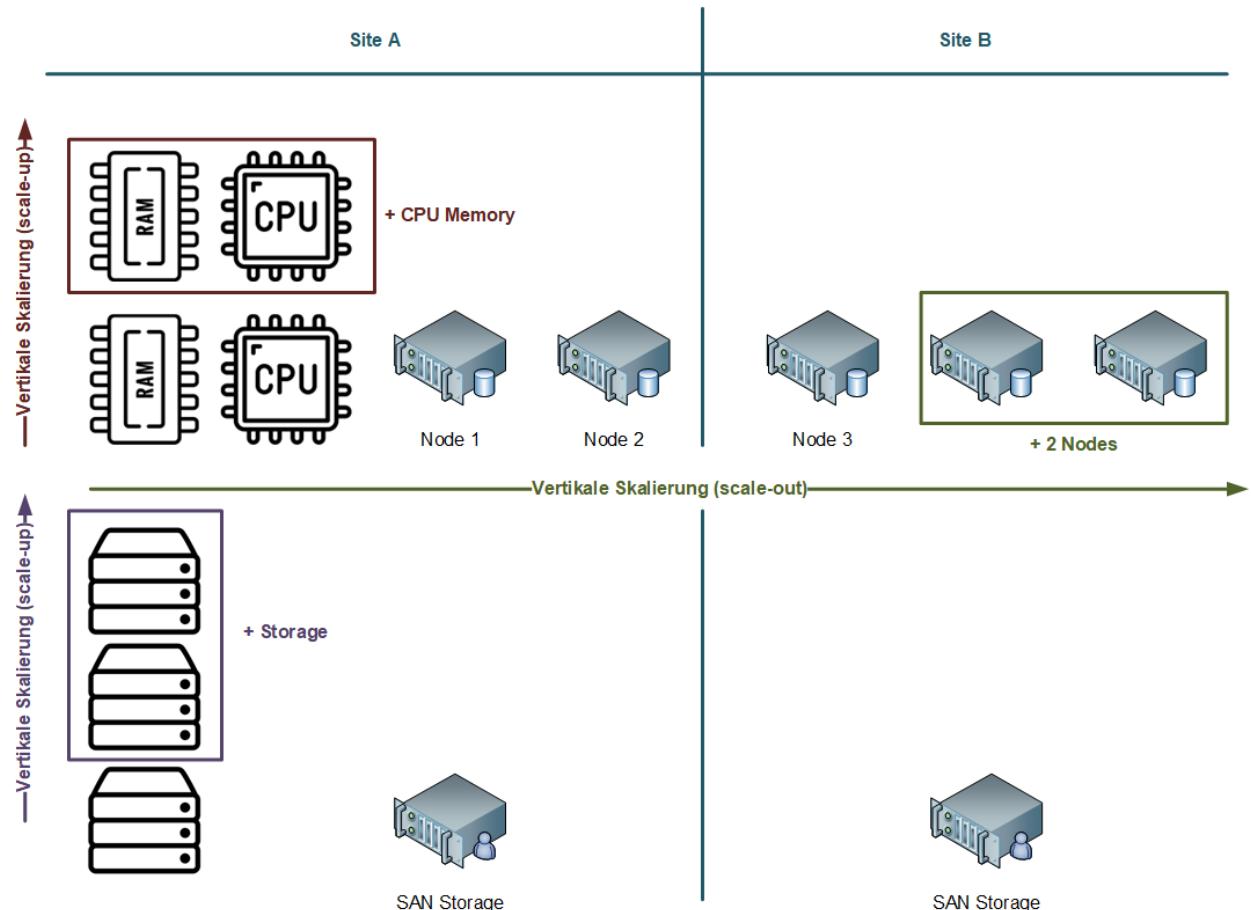


Abbildung 3.5: Datenbank skalierung

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 Monolithisches und mindestens 2 zwei Distributed SQL Cluster ermittelt werden	Beides	MUSS
2	Synergien		Skripte und APIs des Monolithisches 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 umstellt	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	Beides	MUSS
12	Restore	Geschwindigkeit	Das Wiederherstellen des Ursprungszustands muss 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	Beides	MUSS
21	Management-API	Management-API vorhanden	Das Clustersystem muss Skripte oder eine API liefern, mit dem das System zu konfigurieren, verwalten oder überwachen zu können. 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	Beides	KANN
28	Monitoring - Node Failure		Läuft ein Node auf einen Fehler, muss das Clustersystem dies erkennen und Melden resp. eine Schnittstelle liefern die abgefragt werden kann	Beides	MUSS
29	Maintenance Quality		Da die meisten PostgreSQL HA Lösungen Open-Source sind, muss sichergestellt werden, dass die gewählte Lösung auch aktiv gepflegt wird.	Beides	MUSS
30	Performance	tps - Read-Only	Als Basis dienen hier Informationen wie z.B. GitHub Insights.	Beides	MUSS
31	Performance	tps - Read-Writes	Die Transaktionsrate (transactions per second / tps) für DQL 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 mittels einer Präferenzmatrix ermittelt.

Die Grundlegende Gewichtung wurde folgendermassen vorgenommen:

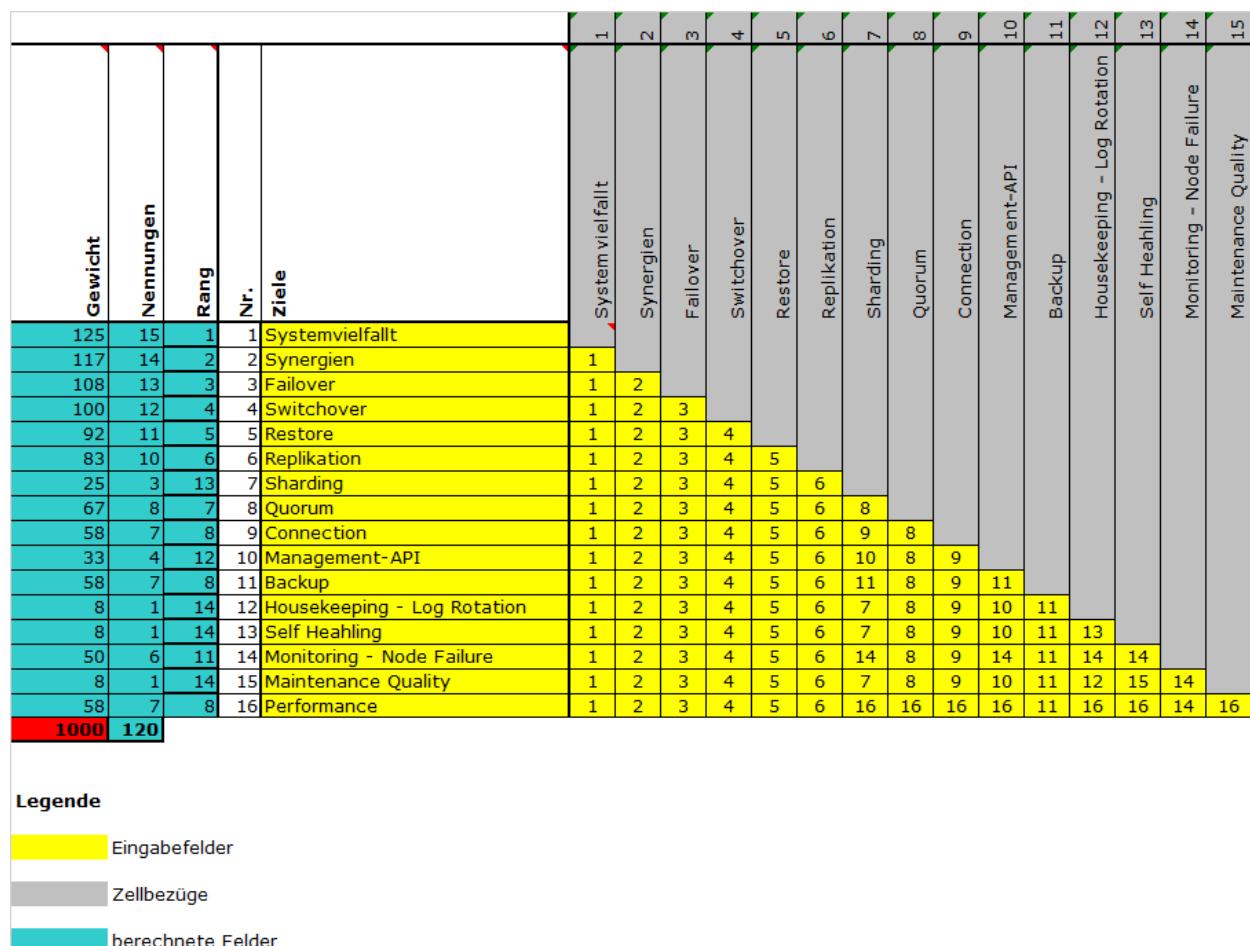


Abbildung 3.6: Präferenzmatrix

3.1.3 Testziele erarbeiten

3.1.3.1 Evaluation - Testfälle

3.1.3.1.1 Patroni

Failover

1. Der Server des Primary-Node wird manuell heruntergefahren.
2. Während dem Failover müssen Daten via SQL eingeführt und ausgelesen werden.
3. Während dem 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 mittels 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 dem 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 dem Failover müssen Daten via SQL eingeführt und ausgelesen werden.
3. Während dem 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 automatisch neu gesetzt werden, wenn notwendig

3.1.3.1.3 YugabyteDB

Failover

1. Ein k8s Node wird manuell heruntergefahren,
indem der entsprechende Server heruntergefahren wird.
2. Während dem Failover müssen Daten via SQL
eingeführt und ausgelesen werden.
3. Während dem 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:

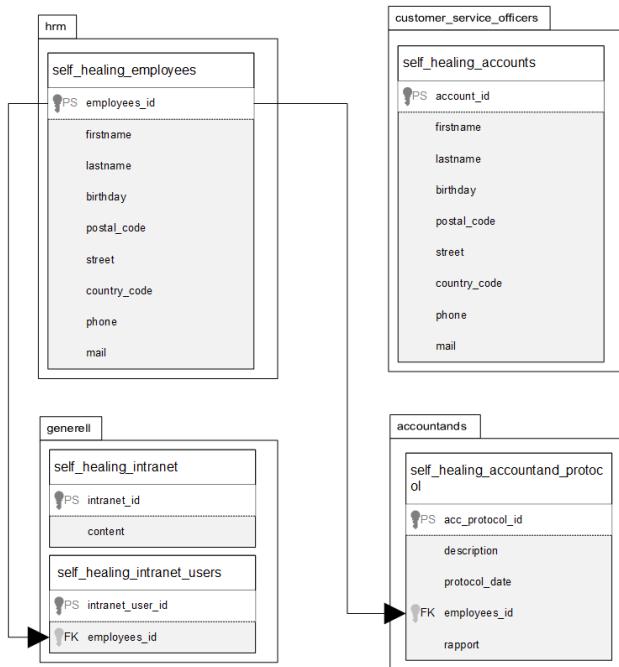


Abbildung 3.7: Testing - ERD DB `self_healing_test`

3.1.4 PostgreSQL Benchmarking

3.1.4.1 pgBench - Basis-Benchmarking

PostgreSQL bietet ein Benchmarking-Tool,[64, 1] 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.

Es gibt einiges zu beachten, wenn PostgreSQL einem Benchmarking unterzogen wird. Aus diversen Quellen [45, 42, 101, 1] sind dies folgende Anforderungen:

pgGather

Mit pgGather [72] müssen vorgängig alle Probleme analysiert und beseitigt werden.

Maintenance

Vor und nach dem Initialisieren des Benchmarks muss AUTOVACUUM gestartet werden.

Statistiken bereinigen

Um saubere Informationen mit pgGather sammeln zu können, müssen sie jeweils neu generiert werden.

Non-Default Konfiguration

Die PostgreSQL DB sollte nicht mit der Default Konfiguration betrieben werden.

Die Konfiguration sollte anhand der zu erwartenden Workloads und Sessions konfiguriert werden.

Benchmark anpassen

Der Benchmark sollte sich an die zu erwartenden Anzahl Sessions und Anzahl Transaktionen richten.

Benchmark Dauer

Die Zeit zwischen den Transaktionen und die Dauer an sich sollten über einen längeren Zeitraum stattfinden.

Nur so, kann ein echtes Verhalten gemessen werden.

Störfaktoren

Störfaktoren, etwa Netzwerklatenzen [97] usw., müssen ebenfalls bemessen werden.

Nur so kann sichergestellt werden, dass die Umgebung sauber ist.

3.1.4.2 Replication Throughput Benchmarking

Etwas Komplexer wird es beim Benchmark des Throughput. Den nebst den DB-Latenzen usw. kommt nun noch die Netzwerklatenz usw. hinzu [83].

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:

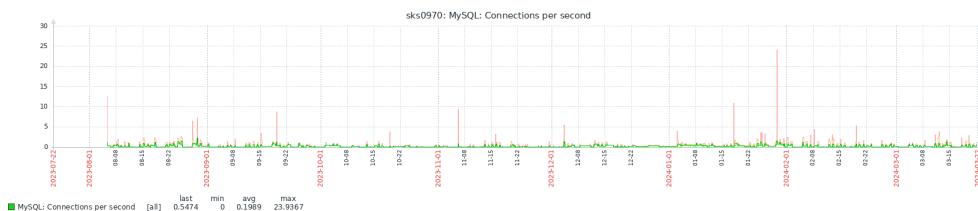


Abbildung 3.9: Benchmark Settings - Zabbix - Connections per Seconds

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

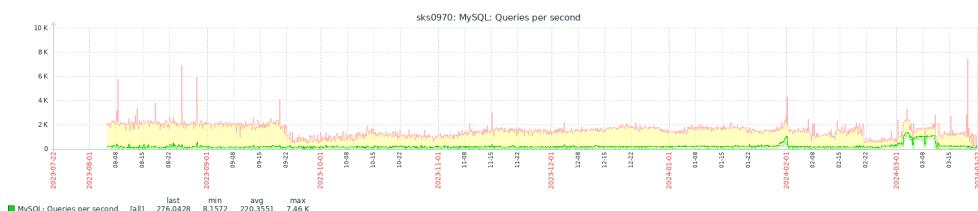


Abbildung 3.10: Benchmark Settings - Zabbix - Queries per Seconds

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

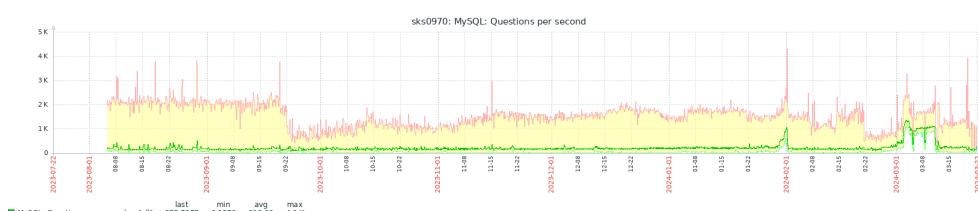


Abbildung 3.11: Benchmark Settings - Zabbix - Client Queries per Seconds

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

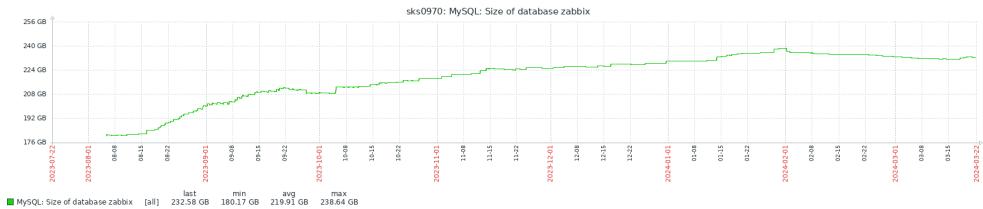


Abbildung 3.12: Benchmark Settings - Zabbix - DB Size

Nun kommen noch die restlichen, 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		ja	ja	ja	ja
dql	Select only	-S	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 einfach 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[65]:

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:

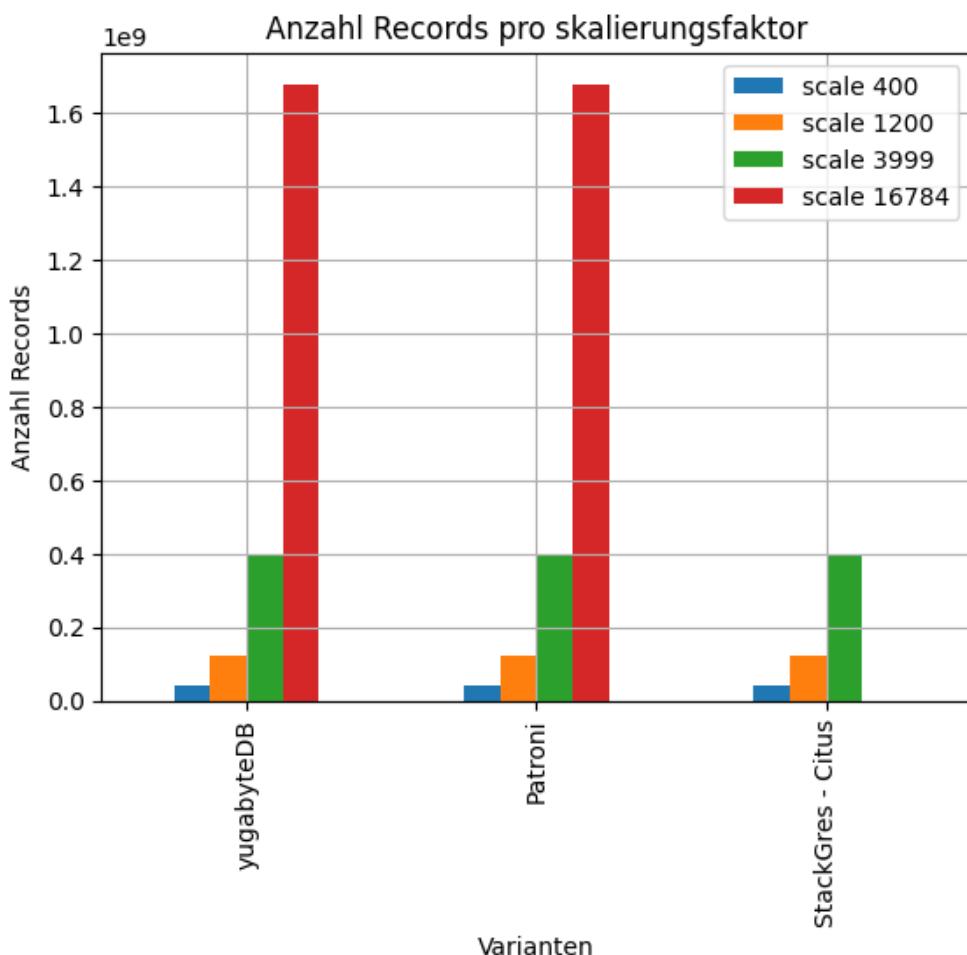


Abbildung 3.13: Benchmark Settings - Anzahl Records / Skalierungsfaktor

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 [41] aller Art für PostgreSQL an.

Percona arbeitet oft auf Basis von Patroni, bietet aber auch eigene Erweiterungen und eigene Software an[36].

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

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

3.1.5.2 EnterpriseDB

EnterpriseDB, oder EDB, ist ein führendes Software- und Servicehaus für PostgreSQL[22]. 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 mittels CloudNativePG an, hat aber auch eine eigens entwickelte Distributed SQL / Active-Active Lösung an.

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 mal wieder auf frei zugängliche Dokumentationen von EDB verwiesen werden.

3.1.5.3 KSGR Lösung

Das Kantonsspital Graubünden hat basierend auf keepalived wird geprüft ob die primäre Seite erreichbar und betriebsbereit ist. Trifft dies nicht mehr zu, wird ein Failover durchgeführt[86]. Ist die primäre Seite wieder verfügbar, wird ein Restore auf die primäre Seite gefahren.

Es wird beim Restore immer 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[62]:

- Watchdog für Automatischer Failover

- Eigener Quorum-Algorithmus
- Integrierter 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 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 aktuell gehalten, 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.

3.1.5.5.1 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

3.1.5.5.2 Replikation

pg_auto_failover bietet die Standard PostgreSQL Replikationen.

3.1.5.5.3 Proxy

pg_auto_failover benötigt einen HAProxy, um Load Balancing usw. [26]

3.1.5.5.4 API / Skripte

pg_auto_failover bietet ein eigenes CLI-Tool, pg_autoctl. Dieses bietet Commands für das einbinden neuer Nodes,
das Managen von Nodes (Maintenance resp. Switchover),
konfigurieren oder monitoren des Systems bietet[32].

3.1.5.5.5 Architektur

Die Dokumentation von pg_auto_failover [6] zeigt auf, wie der Failover funktioniert:

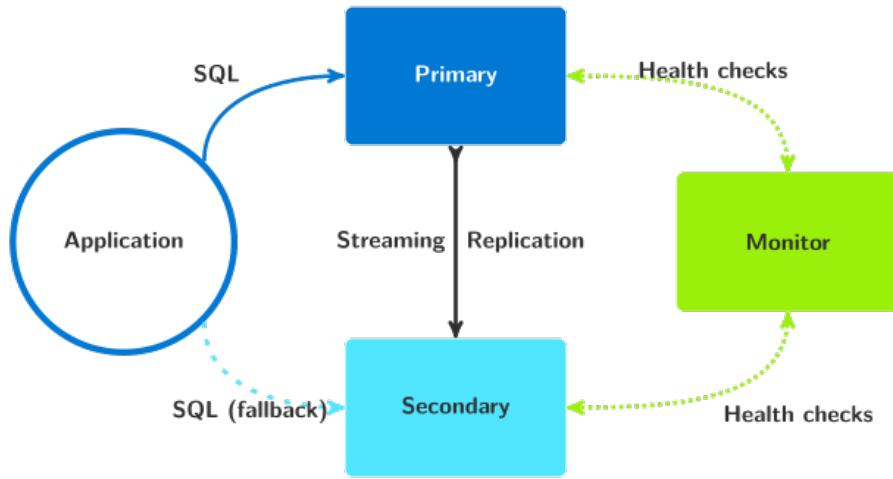


Abbildung 3.14: pg_auto_failover-Architektur - Single Standby

Aber auch Multi-Nodes können eingebunden werden[35]:

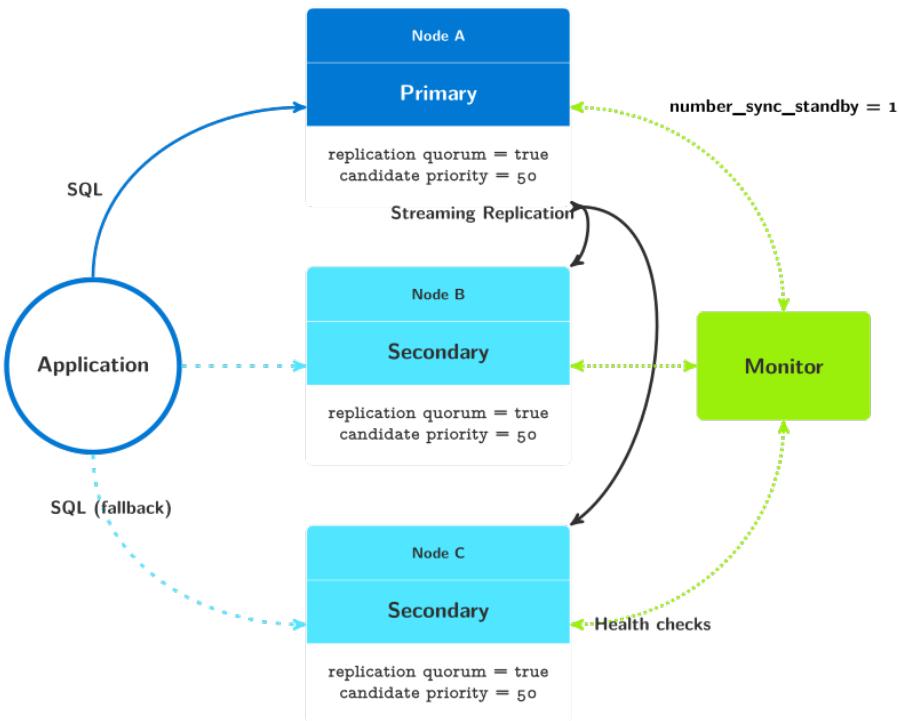


Abbildung 3.15: pg_auto_failover-Architektur - Multi-Node Standby

pg_auto_failover kann Citus einbinden[12]. Allerdings bleibt die Architektur im Kern immer Monolithisch.

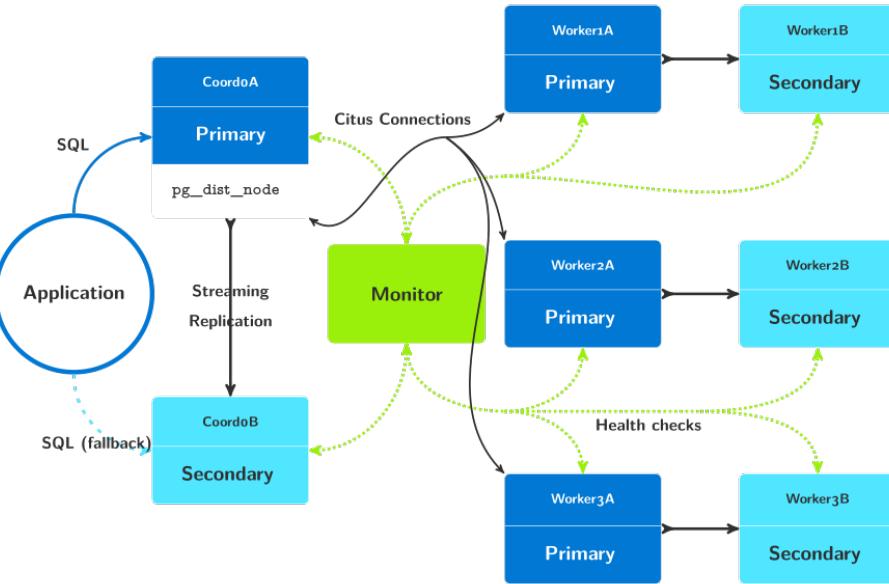


Abbildung 3.16: pg_auto_failover-Architektur - Citus

3.1.5.5.6 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[14]:

- k8s API integration
- Autoamtischer 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 Replikaionen an.

3.1.5.6.3 Proxy

CloudNativePG benötigt keinen zusätzlichen Proxy.

3.1.5.6.4 Pooling

CloudNativePG unterstützt pgBouncer als Pooler.

3.1.5.6.5 API / Skripte

CloudNativePG bietet eine API zum Monitoren und Verwalten von Backups, Clustern und dem System selbst[4].

3.1.5.6.6 Architektur

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

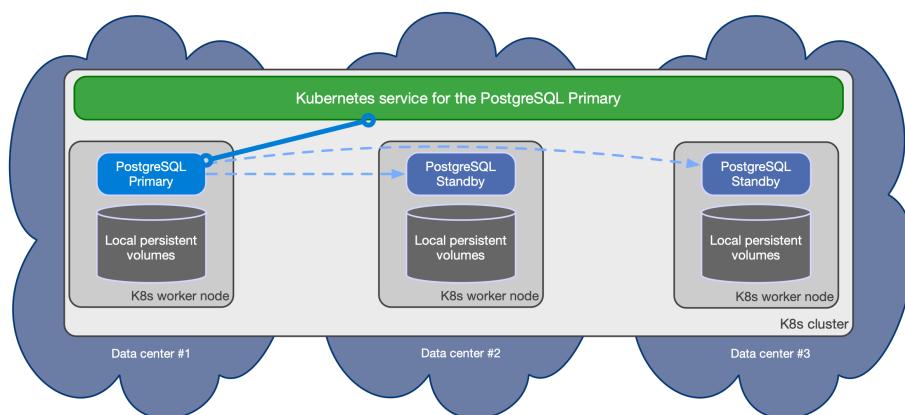


Abbildung 3.17: CloudNativePG - Kubernetes - PostgreSQL

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

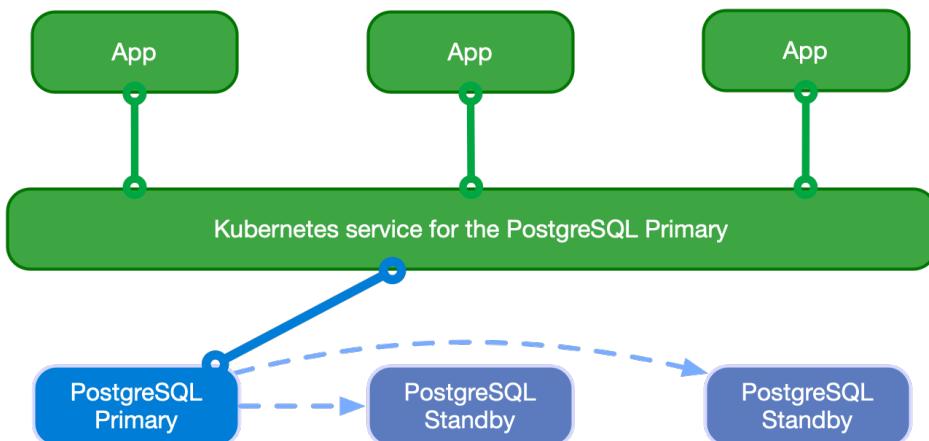


Abbildung 3.18: CloudNativePG - Kubernetes - Read-write workloads

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

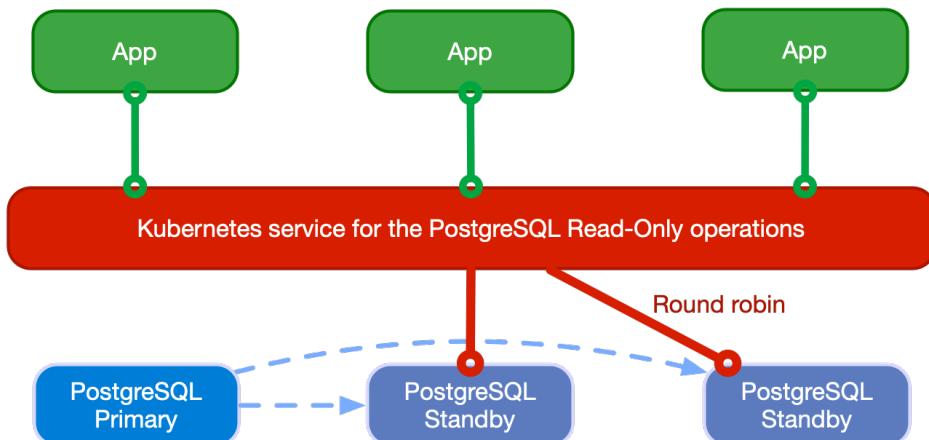


Abbildung 3.19: CloudNativePG - Kubernetes - Read-only workloads

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

[Anhang - Maintenance](#)

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

3.1.5.7.1 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 mittels 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.

Es besteht allerdings die Möglichkeit, die Synchrone Replikation von PostgreSQL selbst einzuschalten.

3.1.5.7.3 Proxy

Patroni benötigt einen HAProxy, um Load Balancing betreiben zu können[26].

3.1.5.7.4 Pooling

Patroni benötigt einen externen Pooler.
Hier wird oft PgBouncer [37] 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 bereitstellt.

3.1.5.7.6 etcd

Patroni benötigt etcd als key-value-store

3.1.5.7.7 Architektur

Das Architektur-Schaubild sieht folgendermassen aus:

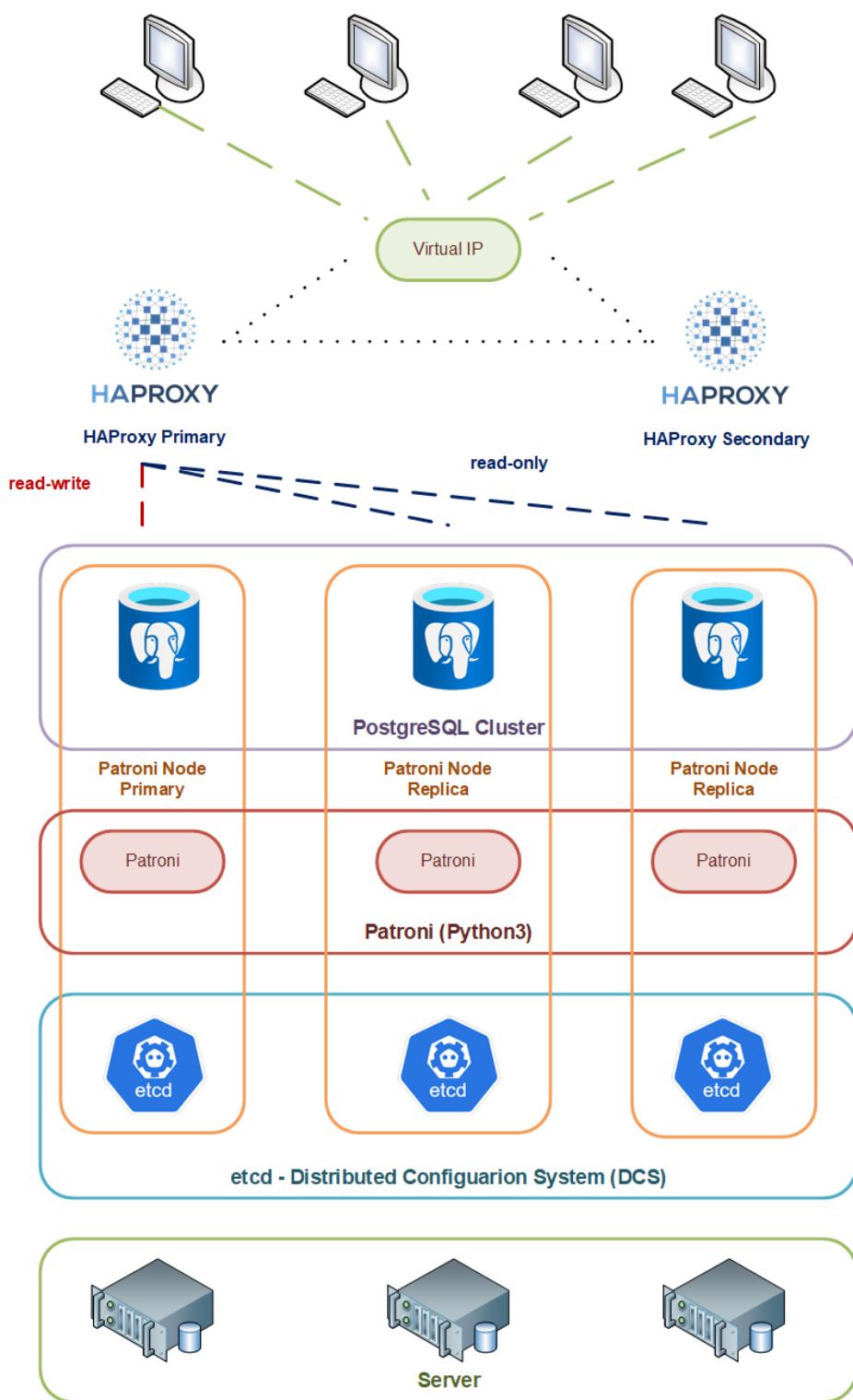


Abbildung 3.20: Patroni-Architektur

3.1.5.7.8 Maintenance

Anhang - Maintenance

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[102] 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 Installation 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[25]:

- 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[23] implementiert.

3.1.5.8.4 Pooling

PgBounder[37] ist bereits integriert, es braucht also keinen weiteren 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:

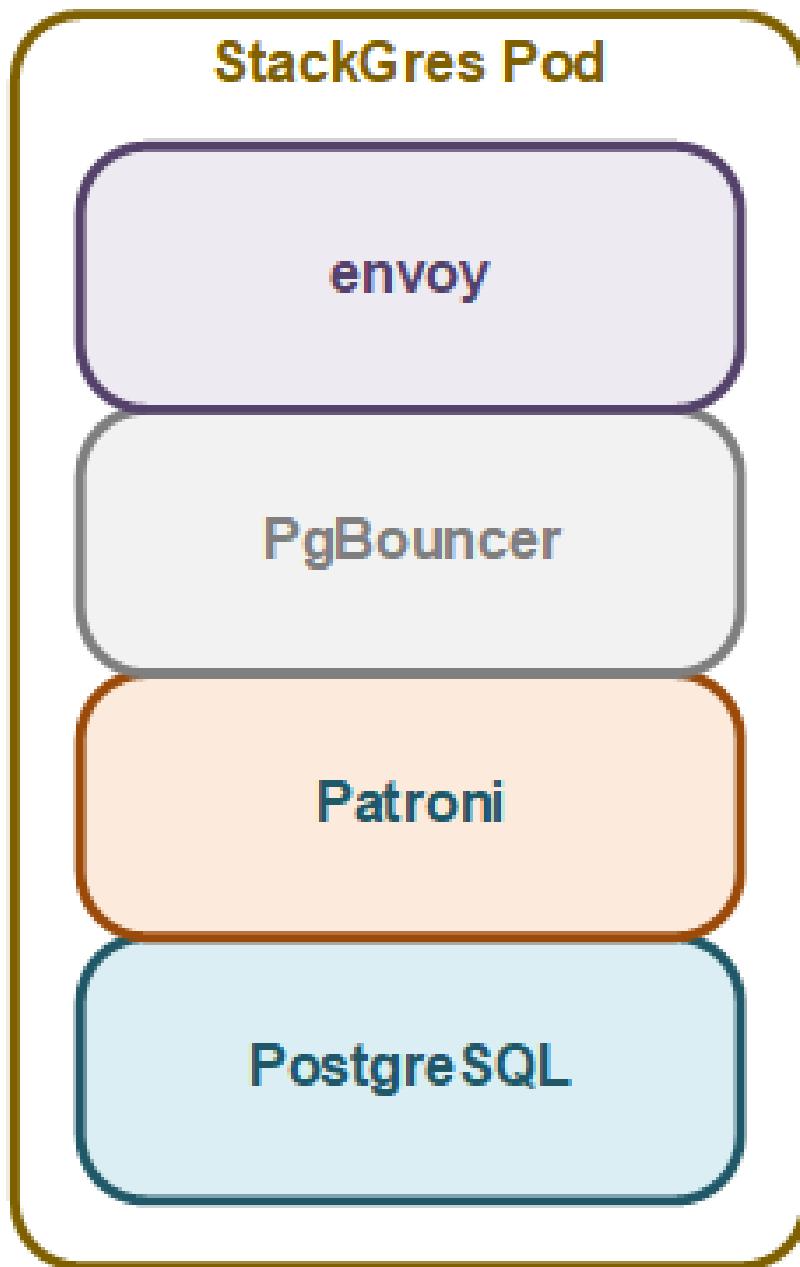


Abbildung 3.21: Stackgres - Grundarchitektur

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.

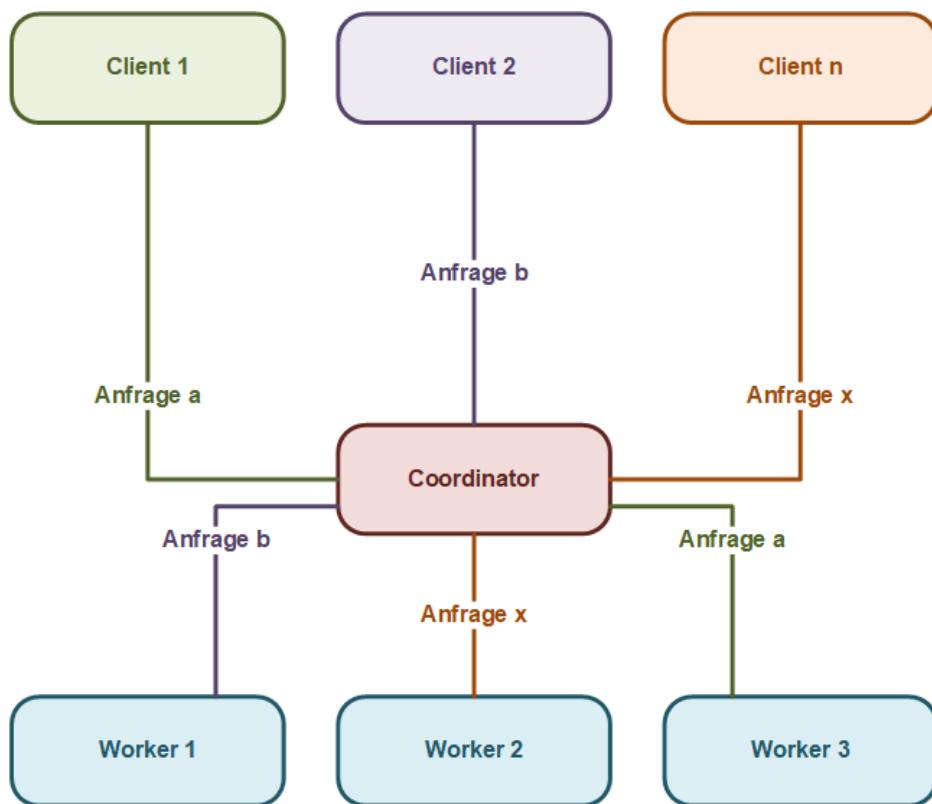


Abbildung 3.22: Citus - Coordinator und Workers

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. [17, 9]

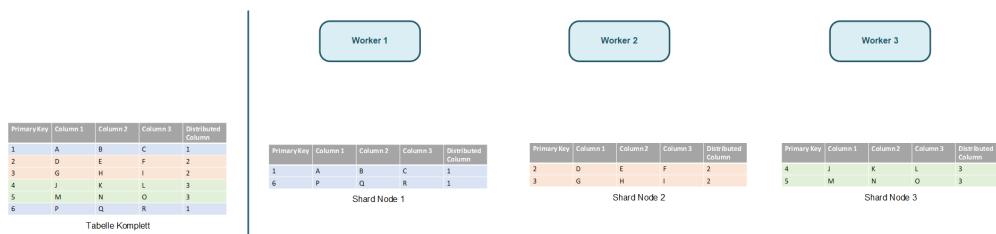


Abbildung 3.23: Citus - Row-Based-Sharding

Schema-based sharding

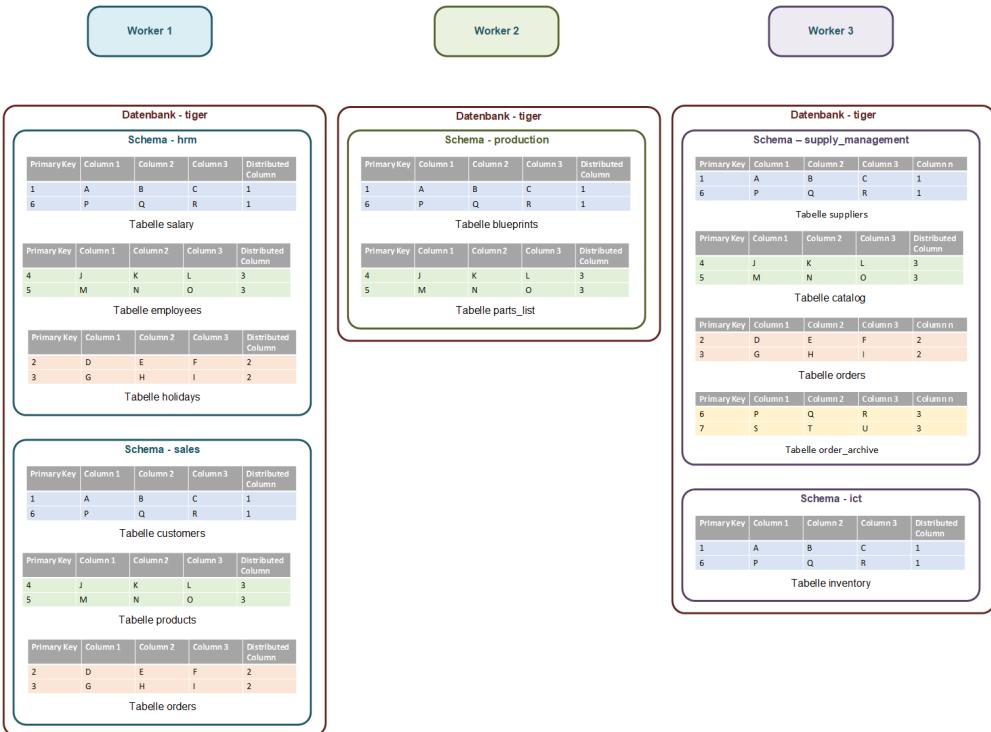


Abbildung 3.24: Citus - Schema-Based-Sharding

Schlussfolgerung Beide Sharding-Methoden haben eine grosse Schwäche. In Version 7.2 konnte noch ein Replikationsfaktor angegeben werden[16], ab Version 11 wurde auch diese Variante gestrichen und man konnte noch eine 1:1 Repliacation auf einen Worker fahren[11]. Spätestens mit Version 12 steht auch dies nicht mehr zur Verfügung, man muss eine Replication auf e Sie sind nicht vollständig ACID-Konform da Datenverlust entstehen kann, wenn ein Node wegfällt. Dies muss aber bei der evaluation mittels Tests noch bestätigt werden.

Die Shards müssen aber, stand heute, mit entsprechenden mit Replikation gesichert werden[15]. 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:

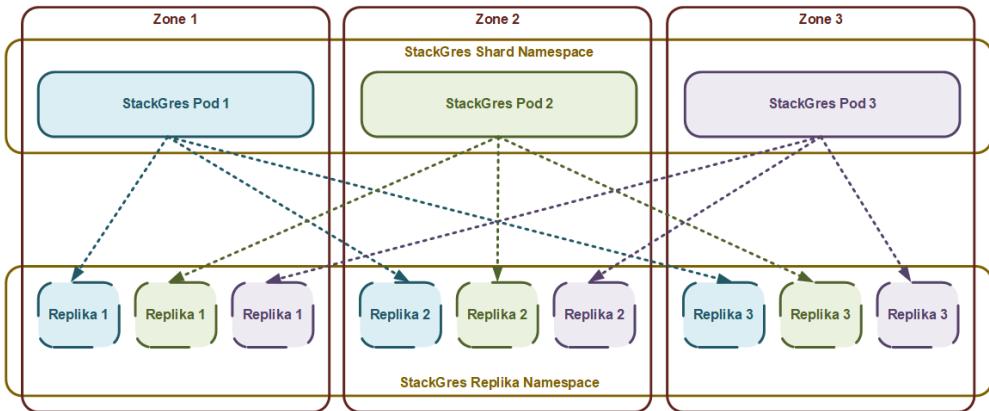


Abbildung 3.25: StackGres-Citus - Shard-Replikation

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 natürlich 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 garantiebar.

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 regelmäßig 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[98].

3.1.5.9.1 Core-Features

Die wichtigsten Features von YugabyteDB sind[8]:

- PostgreSQL Kombatibel
- Cassandra-Kompatibilität
- Horizontale Skalierbarkeit
- Global Verteilbar
- Cloud Native

3.1.5.9.2 Replikation

3.1.5.9.3 Proxy

YugabyteSQL nutzt Kubernetes und seine Core-Functions als Load Balancer. Ein zusätzlicher Proxy wird nicht benötigt.

3.1.5.9.4 Pooling

YugabyteDB hat ein Connection Pooling mit dem YSQL Connection Manager integriert[66].

3.1.5.9.5 API / Skripte

YugabyteDB bietet eigene APIs[5] und CLIs[13] für das Verwalten an.

Diese bieten auch die Möglichkeit, 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:

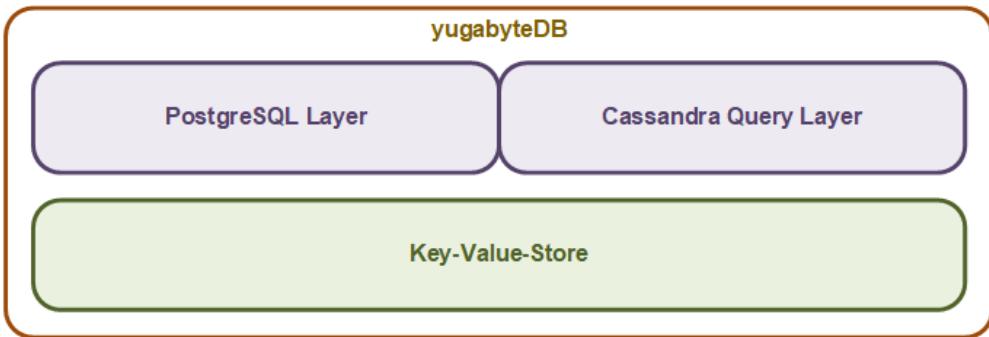


Abbildung 3.26: YugabyteDB - Grundkonzept

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

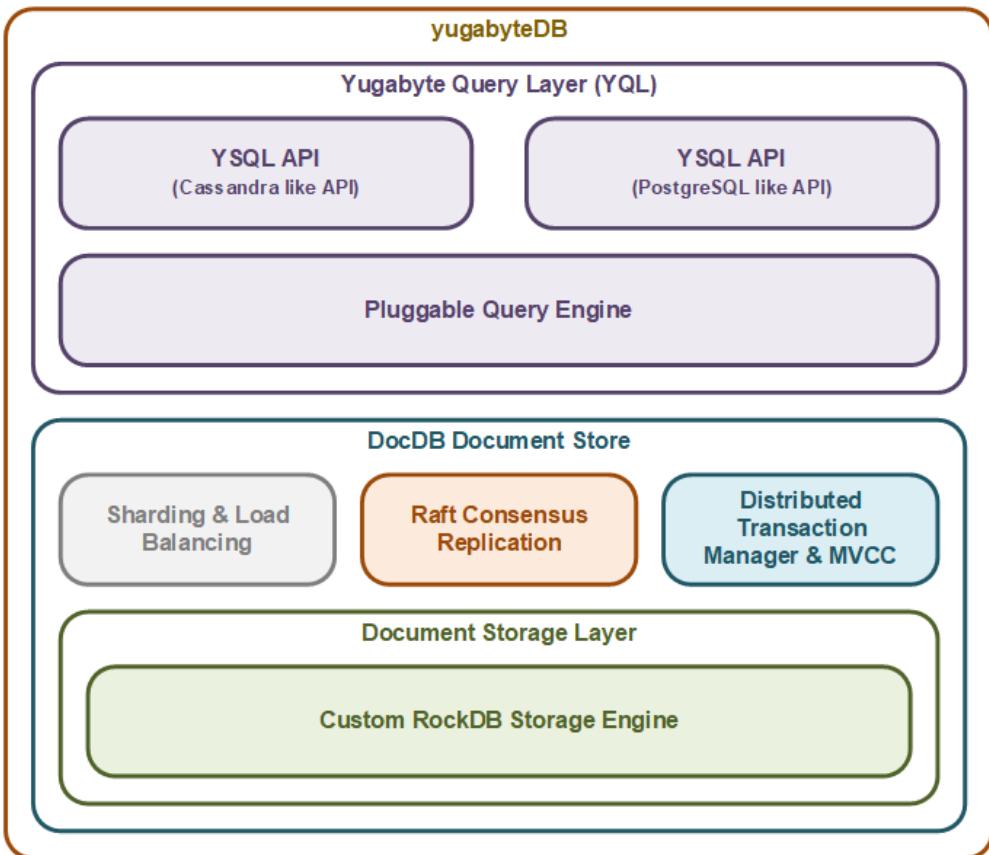


Abbildung 3.27: YugabyteDB - Architektur

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 single 'Tabelle Komplett' into three tablets ('Tablet 1', 'Tablet 2', and 'Tablet 3') distributed across three nodes.

Tabelle Komplett:

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

Tablet 1:

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

Tablet 2:

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

Tablet 3:

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

Abbildung 3.28: YugabyteDB - Sharding

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

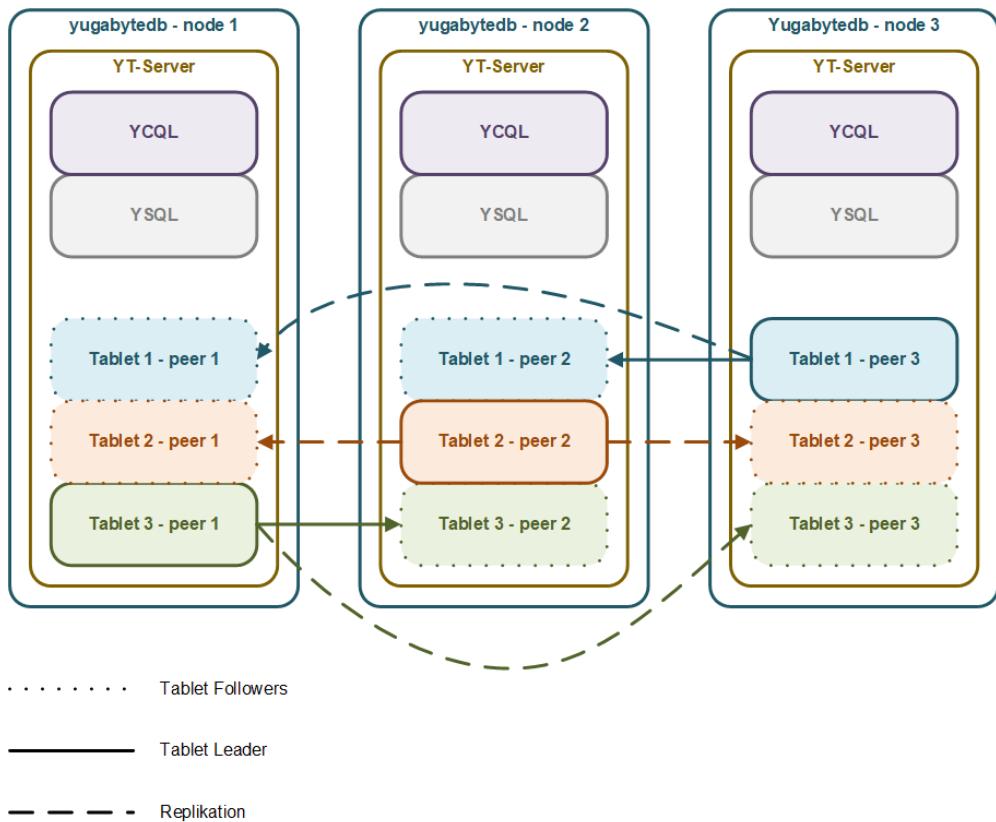


Abbildung 3.29: YugabyteDB - Tablet - Leader und Follower

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 [43] geregelt:

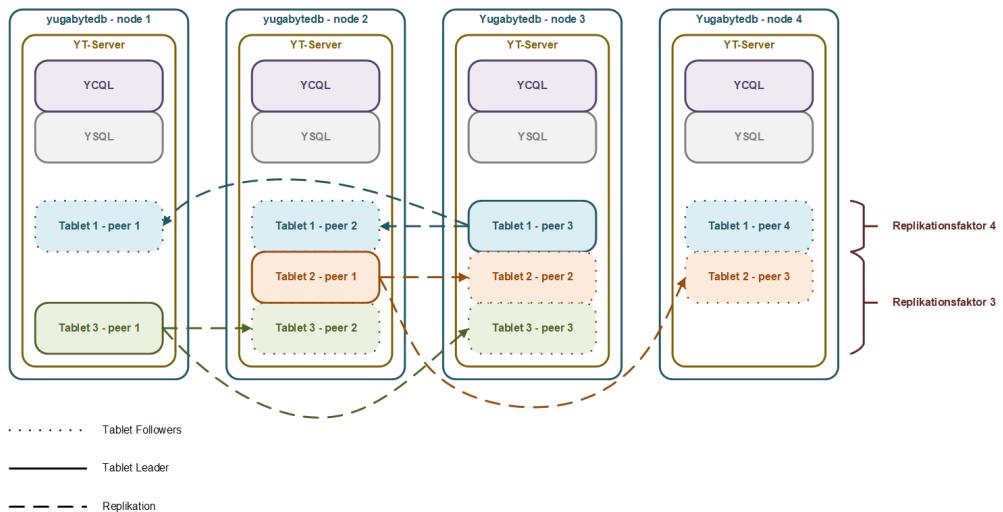


Abbildung 3.30: YugabyteDB - Tablet - Replikationsfaktor

Durch das Raft-Protokoll werden die Tablet-Leader regelmäßig gewechselt.

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

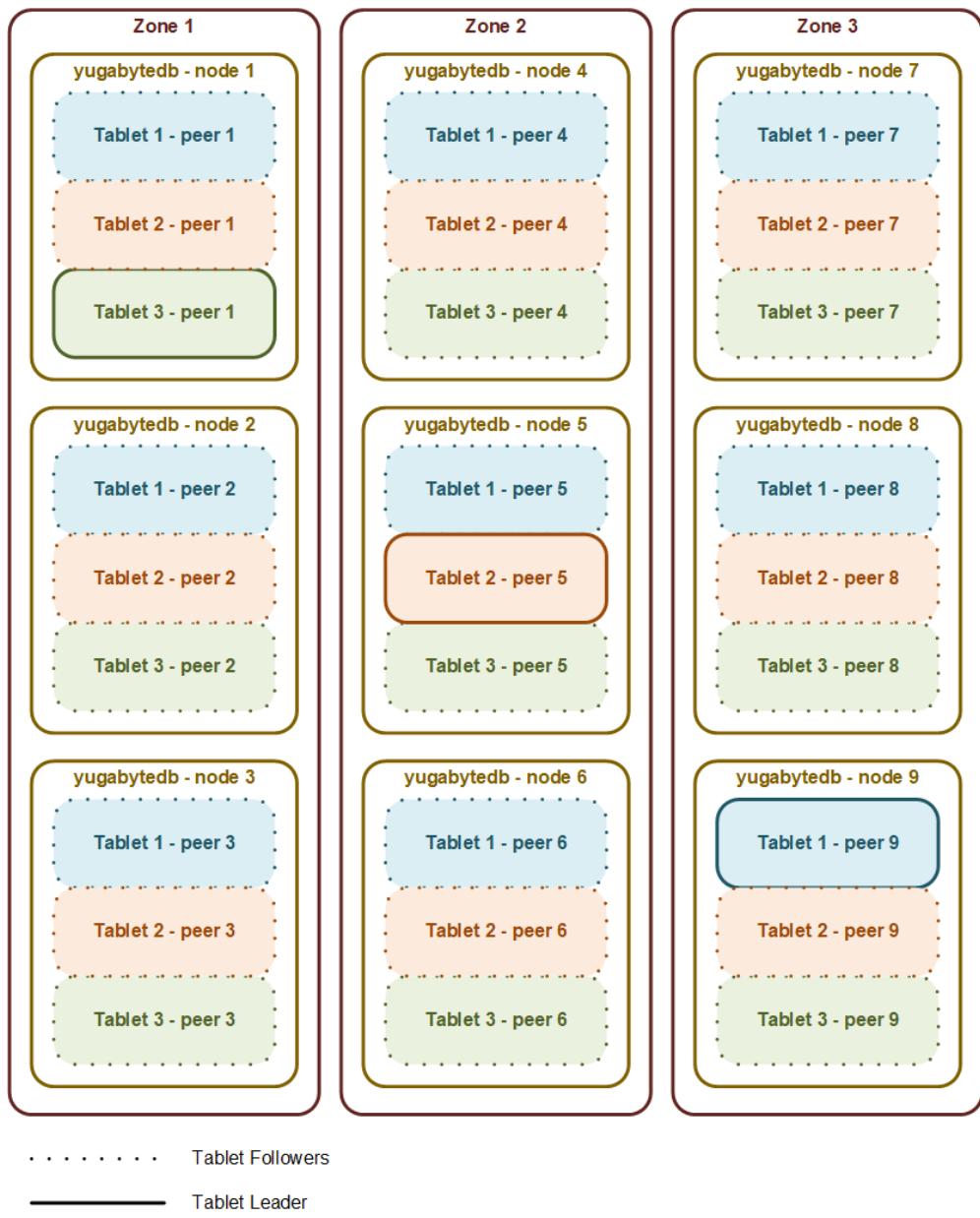


Abbildung 3.31: YugabyteDB - Zonen

Dies wird dann sinnvoll, wenn eine gewisse Ausfalltoleranz erreicht werden soll. Fällt nämlich 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. YugabyteDB nennt dies Zone outage Tolerance[40].

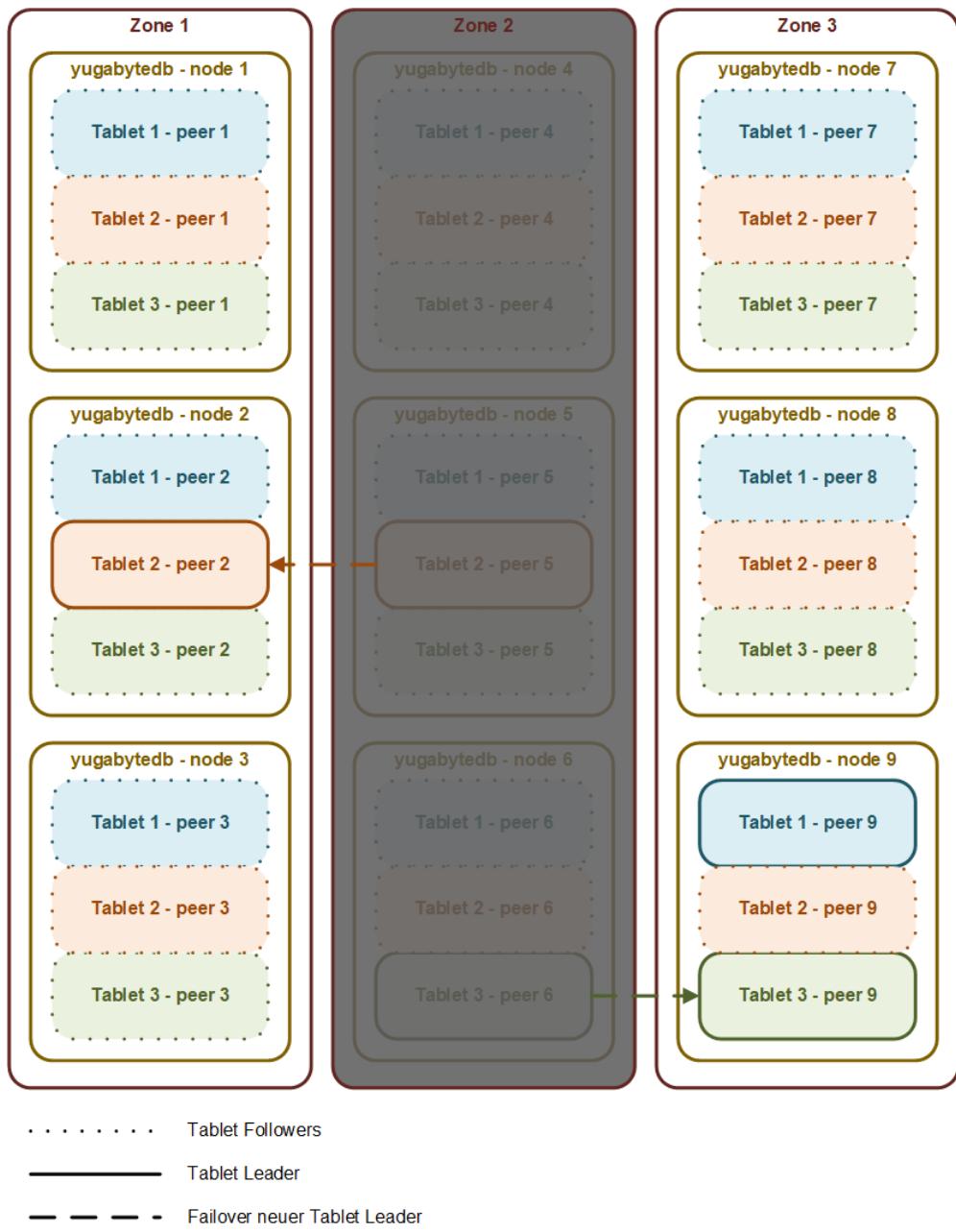


Abbildung 3.32: YugabyteDB - Zone outage Tolerance

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 Replikas. 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 Bietet eine einfache und kompakte Möglichkeit für ein Distributed SQL System.
6	Stackgres mit Citus	Evaluation	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:

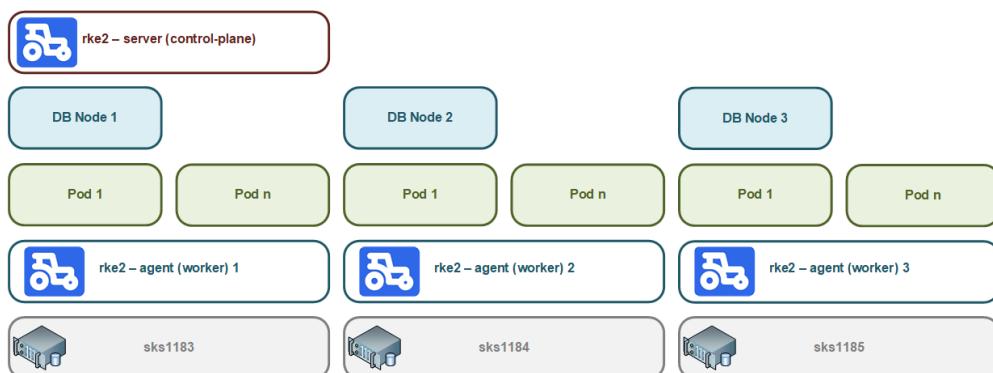


Abbildung 3.33: Evaluationssystem - Distributed SQL / Shards

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 sich die Architektur folgendermassen aussieht:

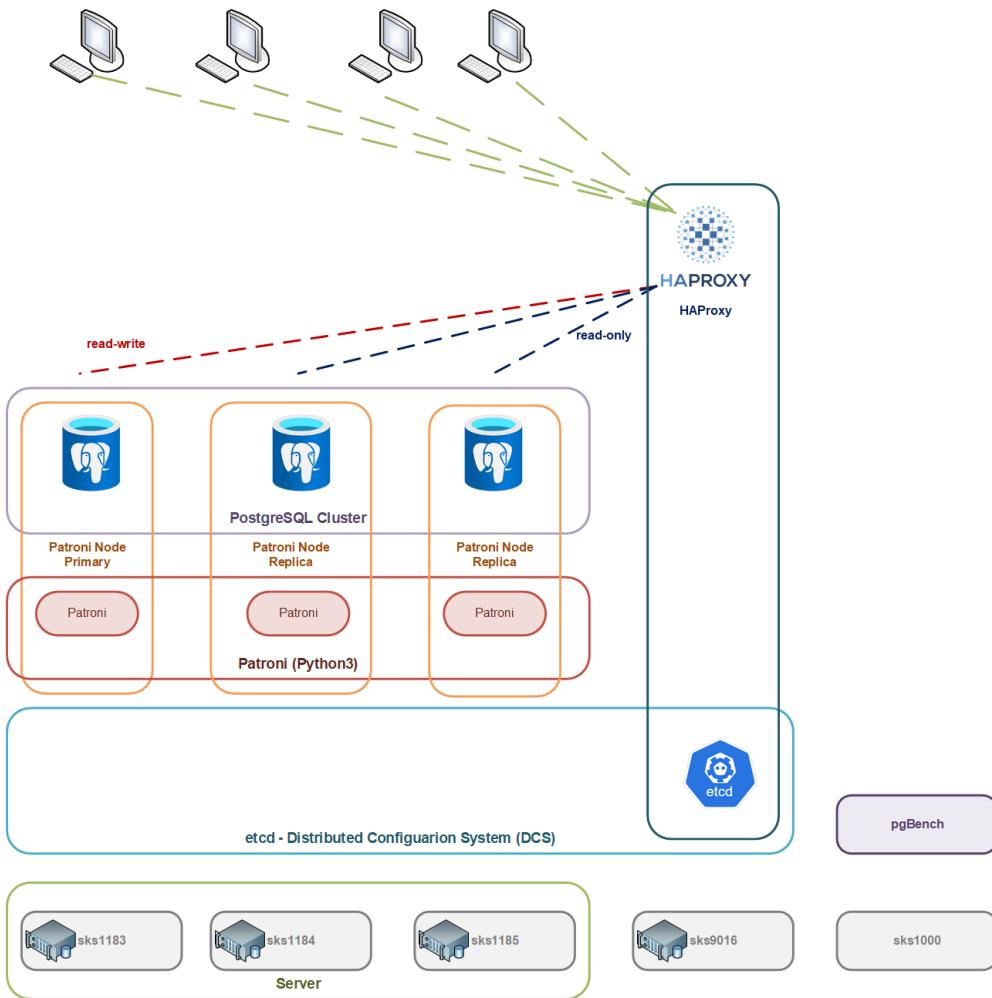


Abbildung 3.34: Patroni - Evaluationsarchitektur

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 ETCDCTL_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 jeweils 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 für den jeweiligen Hostnamen resp. die jeweilige 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 Bootstrappen, 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 Replica-Pod.

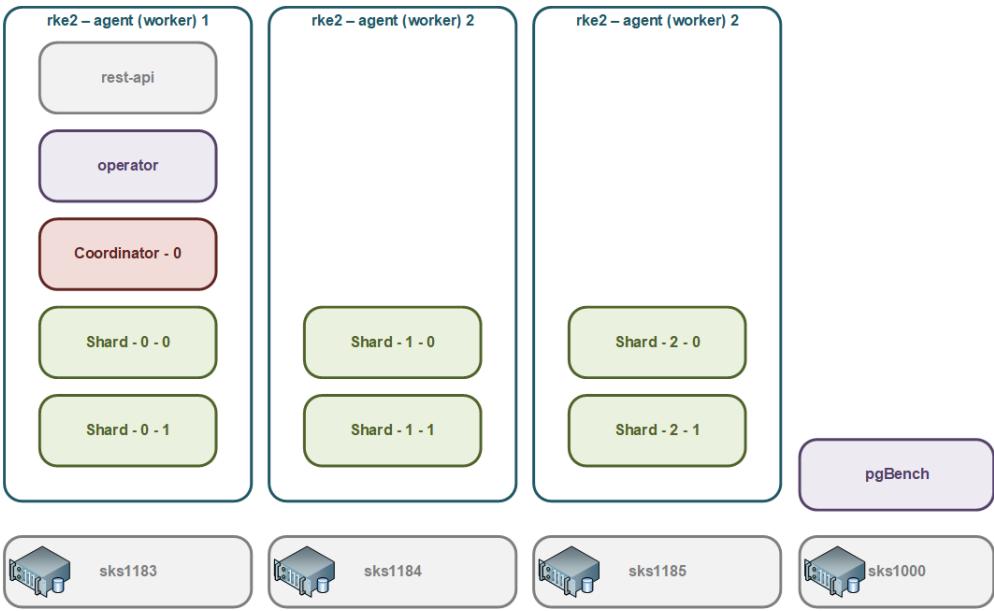


Abbildung 3.35: Stackgres - Citus - Evaluationsarchitektur Benchmarking

Für die Self Healing Tests wurde eine umfangreichere Architektur vorgenommen. Es stellt sich heraus, dass man relativ leicht die beim [Citus Sharding](#) beschriebene Lösung zum Replizieren leicht umzusetzen ist:

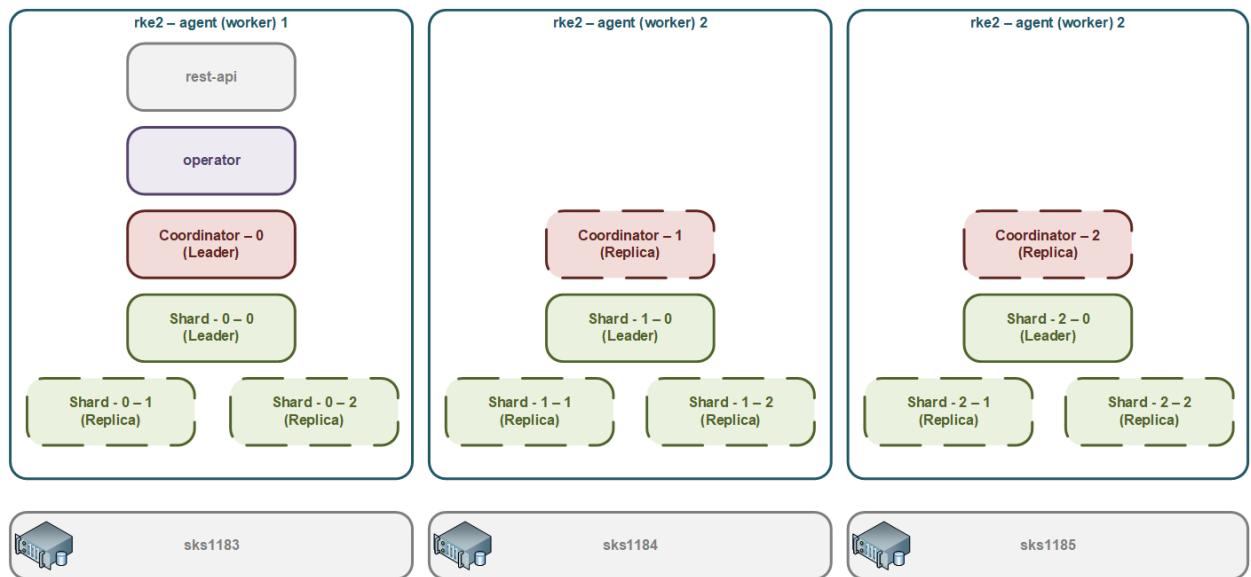


Abbildung 3.36: Stackgres - Citus - Evaluationsarchitektur Self Healing Tests

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 Worst-Case 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:

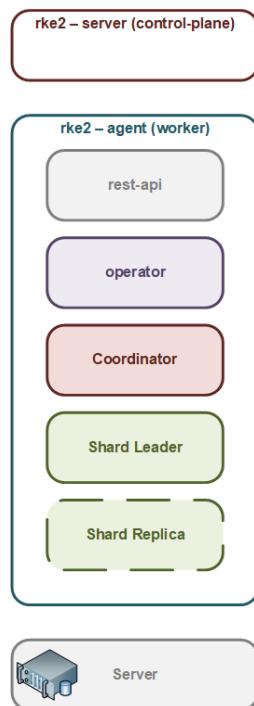


Abbildung 3.37: Stackgres - Citus - Resourcen - Stack

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:

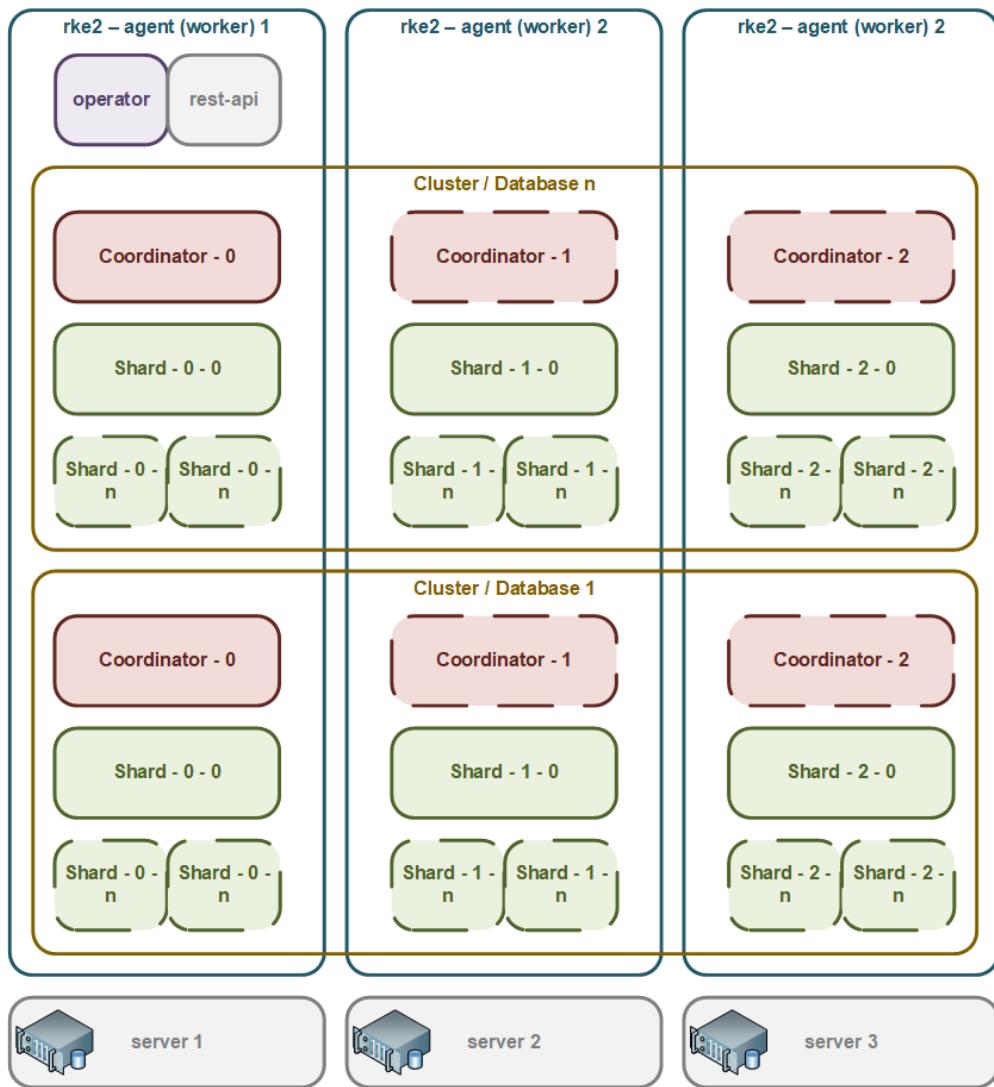


Abbildung 3.38: Stackgres - Citus - Datenbank - Cluster

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 mittels 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, welches mit `https` erreichbar sein muss.

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 mittels https aus dem Pod heraus kommuniziert werden kann, selbst wenn es mit curl möglich ist.

Damit dies möglich ist, 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%2
Fproxy.svc.first-it.ch%3A8080?skipHostnameVerification:true&setHttpScheme:
true
```

Listing 3.7: StackGres - values.yaml - Extension Proxy

Die Ursachenforschung hat viel Zeit in Anspruch genommen.

Es sind nebst dem Versuch, eine Freigabe via Pod-Affinität zu lösen, drei Tage verstrichen bis StackGres die Extensions ausführen konnte.

Anders als bei YugabyteDB, kann man das Web-GUI nicht mittels 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 mittels 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,
das 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, allerdings wird dann eine
PostgreSQL-DB mit minimalem Setting deployt.

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 mittels 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 das 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[38].

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 sich aber als schwieriges unterfangen auszeichnete da Ongres dies nicht wirklich Dokumentiert hat.

Daher wurde immer ein Testing-Profil gesetzt:

```
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.

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

3.1.7.4 YugabyteDB

3.1.7.4.1 Architektur

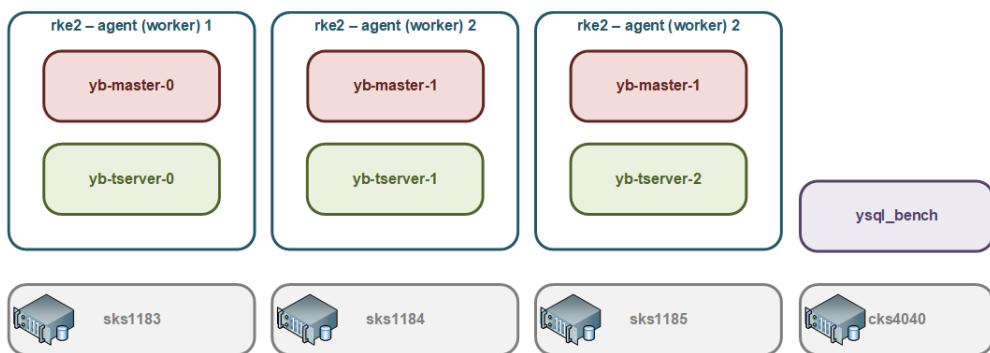


Abbildung 3.39: YugabyteDB - Evaluationsarchitektur

Die Architektur ist einfach.

Auf allen Worker-Nodes wird je ein `tmaster` und `tserver` Pod gestartet werden.

3.1.7.4.2 Installation

Während der Installation des YugabyteDB Evaluations-Enviroment wurde festgestellt, das man zwei Varianten installieren kann. YugabyteDB (Repository `yugabyte`) und YugabyteDB Anywhere (Repository `yugaware`):

```

Context: default          <C>      Copy
Cluster: default          <E>      Edit
User: default             <N>      Next Match
K9s Rev: v0.31.8 ✨v0.32.4 <Shift-N> Prev Match
K8s Rev: v1.29.0+rke2r1   <R>      Toggle Auto-Refresh
CPU: 1%                   <F>      Toggle FullScreen
MEM: 38%
Name: yw-test-yugaware-0
Optional: false
pg-init:
  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: <none>
  DownwardAPI: 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.

```

Abbildung 3.40: YugabyteDB - Subscription Yugawre

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 zuwenig Cores zur Verfügung stehen.

Bei der Konstellation rke2, Cilium und MetalLB, 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 3.13: metallb - Konfig YAML - Detail L2Advertisement

Dieses Problem ist schwer zu greifen und hat zwei Tage in Anspruch genommen, es zu lösen. 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[33], 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 3.14: YugabyteDB - Helm Chart Manifest - Detail Image

Die StorageClass muss im values.yaml gesetzt werden, einmal für den master und einmal für den 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 3.15: 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 3.16: 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 3.17: 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 3.18: YugabyteDB - Helm Chart Manifest - Detail Disable YSQL

Nun muss 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 3.19: 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 3.20: 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 3.21: 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 logischerweise dazu führt, dass zuwenig 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 3.22: 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 3.23: 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 3.24: Patroni - Testing - Reinit

Das vollständige ergebnis:

Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis	Begründung
Fallover	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.
Fallover	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.
Fallover	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 der 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 Citrus-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	Keine. StackGres setzt envoy ein. Offensichtlich nicht bei einem ganzen Cluster
Failover	3	Geschwindigkeit	Der Failover muss so schnell stattfinden, dass offene Connections nicht wegen eines Timeouts geschlossen werden.	Nicht eingetroffen	Keine. StackGres setzt envoy ein. Offensichtlich nicht bei einem ganzen Cluster
Sharding und Datenintegrität	4	Datenkonsistenz	Daten sind Konsistent und Inetgr. Eingetroffen	Eingetroffen	
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 zu finden: [Anhang - StackGres - Citus Testing](#)

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 Inetgr. 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 dem 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[87].

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 zu finden:

[Anhang - YugabyteDB Testing](#)

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 Primary die Files und die Disk lief jeweils voll.

Es wurde an verschiedenen Stellschrauben geschraubt, etwa an der Anzahl Worker, der grössen 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 3.25: Patroni - Benchmarking - Monitoring

Alle Benchmarking-Commands und SQLs zur ermittlung der grössze 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[7].

Das hat einen besonderen Grund.

Wenn die Tabellen mittels 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 3.26: Citus - Benchmarking - Distributed Table Sharding

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

pgbench erzeugt aber gleich mehrere davon:

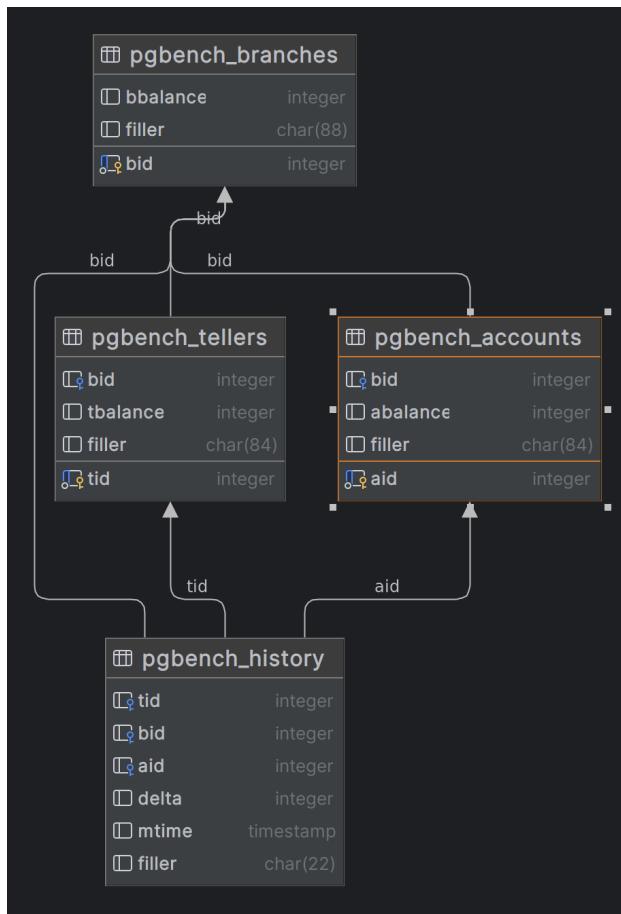


Abbildung 3.41: Benchmarking - ERD pgbench

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 3.27: 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[20].

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 gql-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), also 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, dass 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, das Log gibt allerdings nicht genug her, 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.

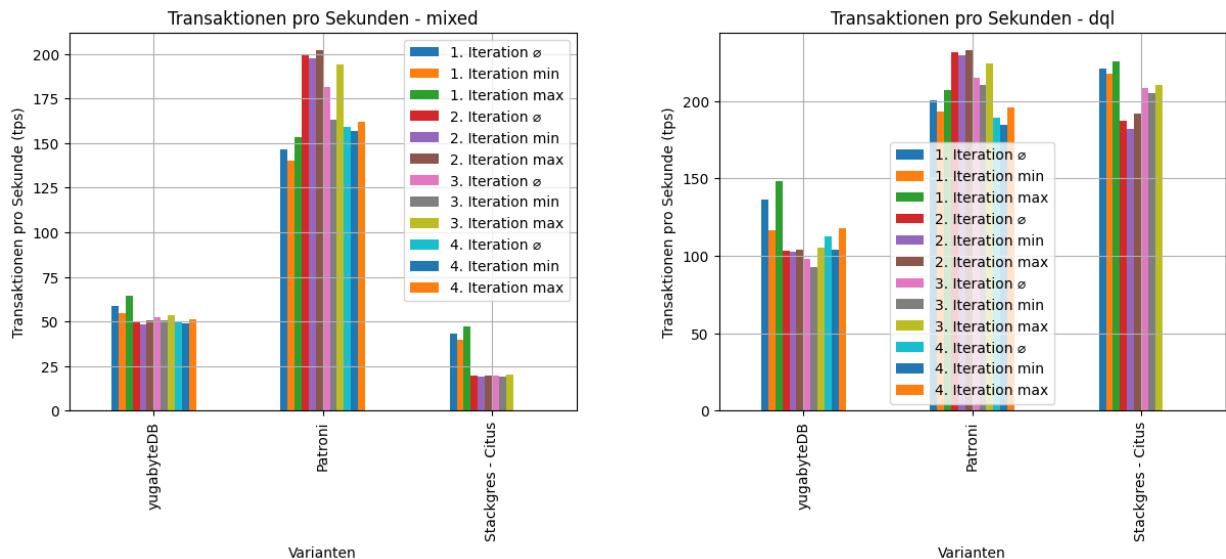


Abbildung 3.42: Benchmarks - tps

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

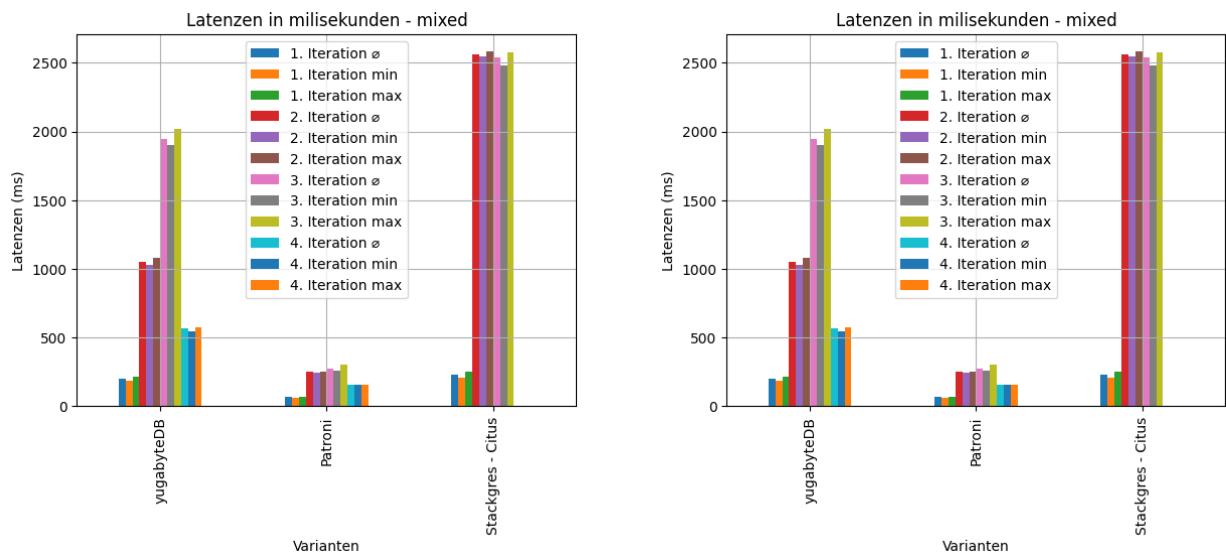


Abbildung 3.43: Benchmarks - latency

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:

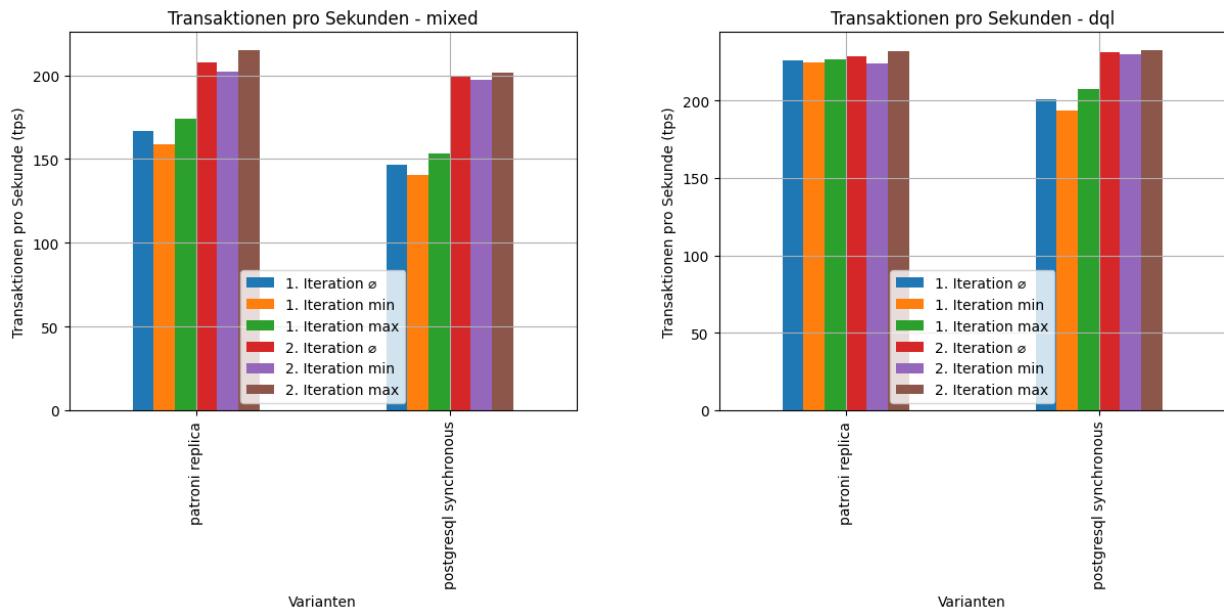


Abbildung 3.44: Benchmarks - tps Patroni Replica

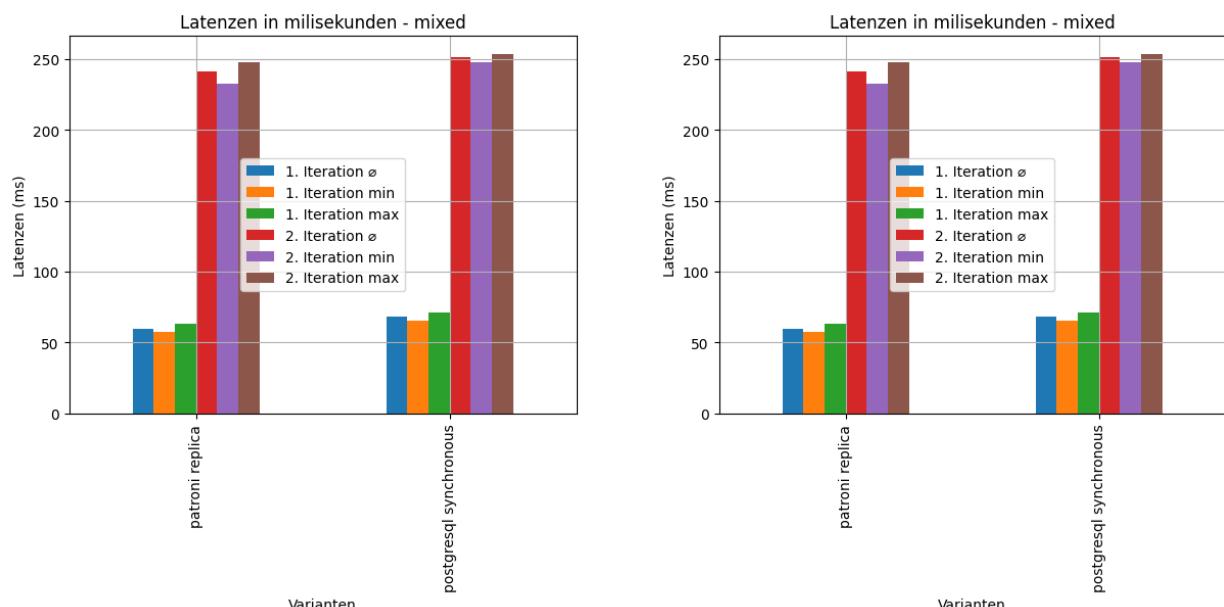


Abbildung 3.45: Benchmarks - latency Patroni Replica

Die Asynchrone Replikation ist dabei ein klein wenig schneller als die Sychrone Replikation.

Ein weiterer Benchmark sind die Fehler, die bei den DML-Transktionen beim mixed-Benchmark

aufreten können.

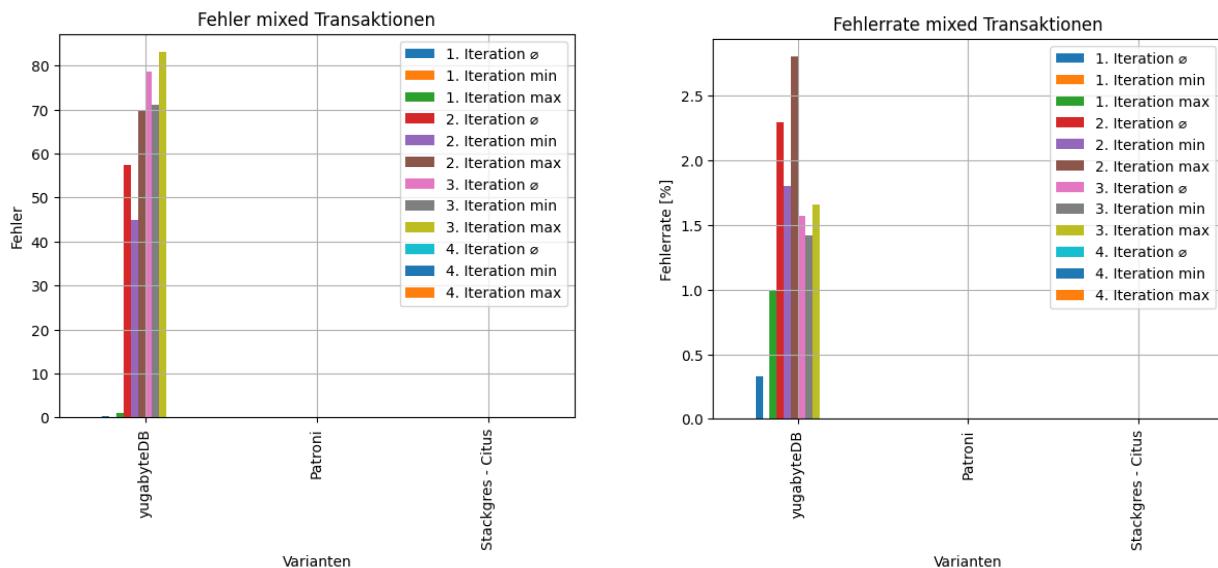


Abbildung 3.46: Benchmarks - Fehler bei mixed-Transaktionen

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

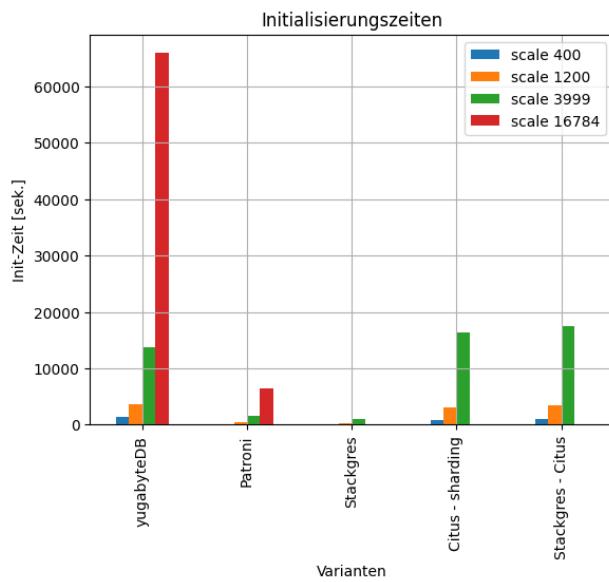


Abbildung 3.47: Benchmarks - Initialisierungszeit - sekunden

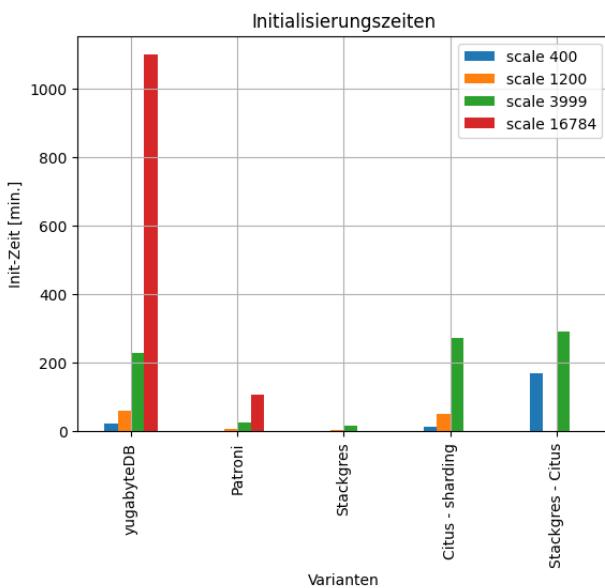


Abbildung 3.48: Benchmarks - Initialisierungszeit - Minuten

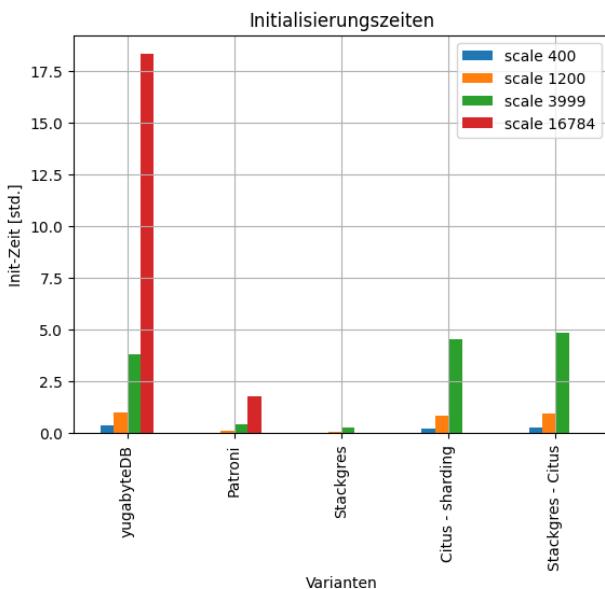


Abbildung 3.49: Benchmarks - Initialisierungszeit - Stunden

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.

Es wurden also folgende Annahmen getroffen resp. folgende Parameter sind gegeben:

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:

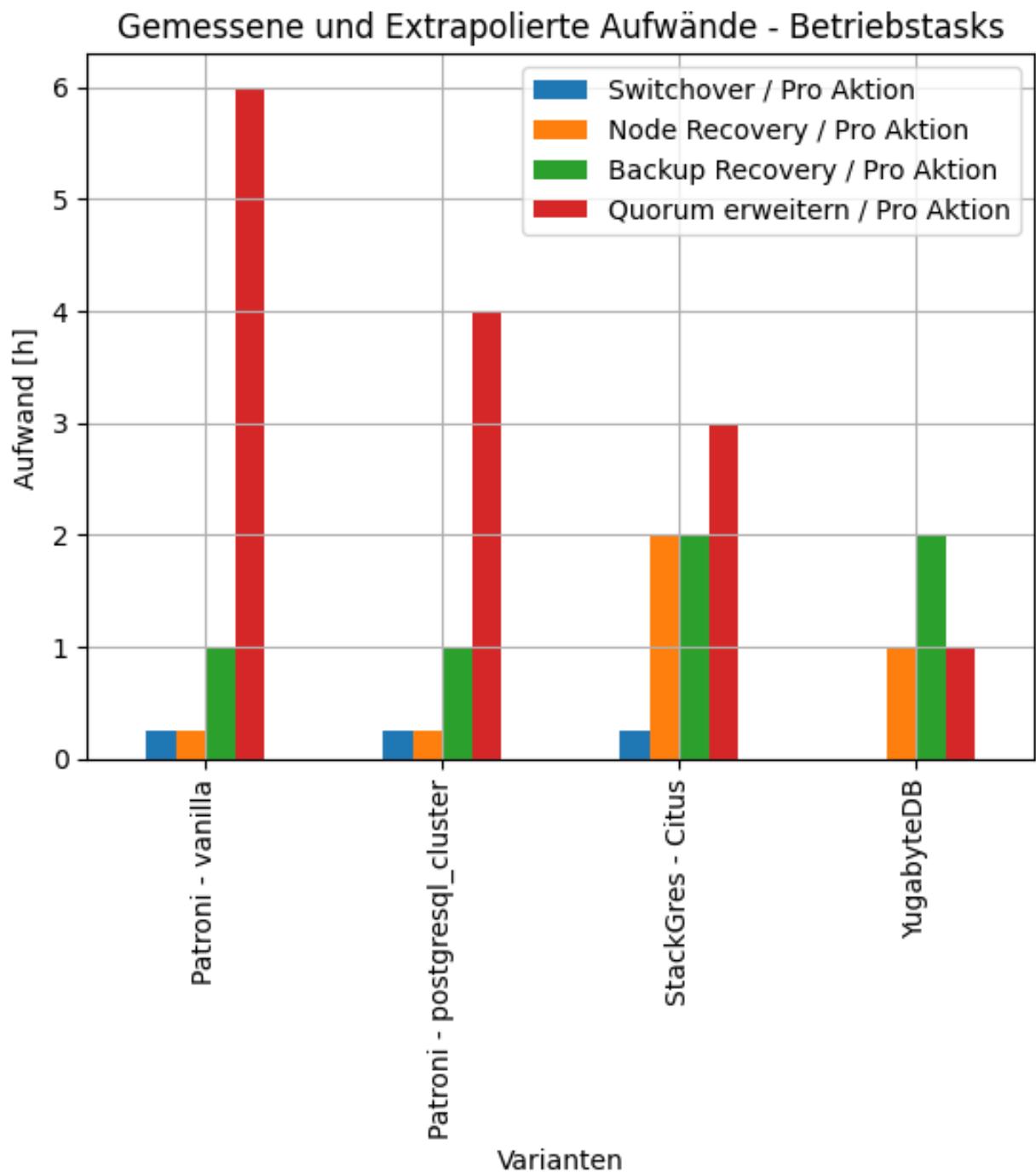


Abbildung 3.50: Zeitaufwände pro Betriebstask

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

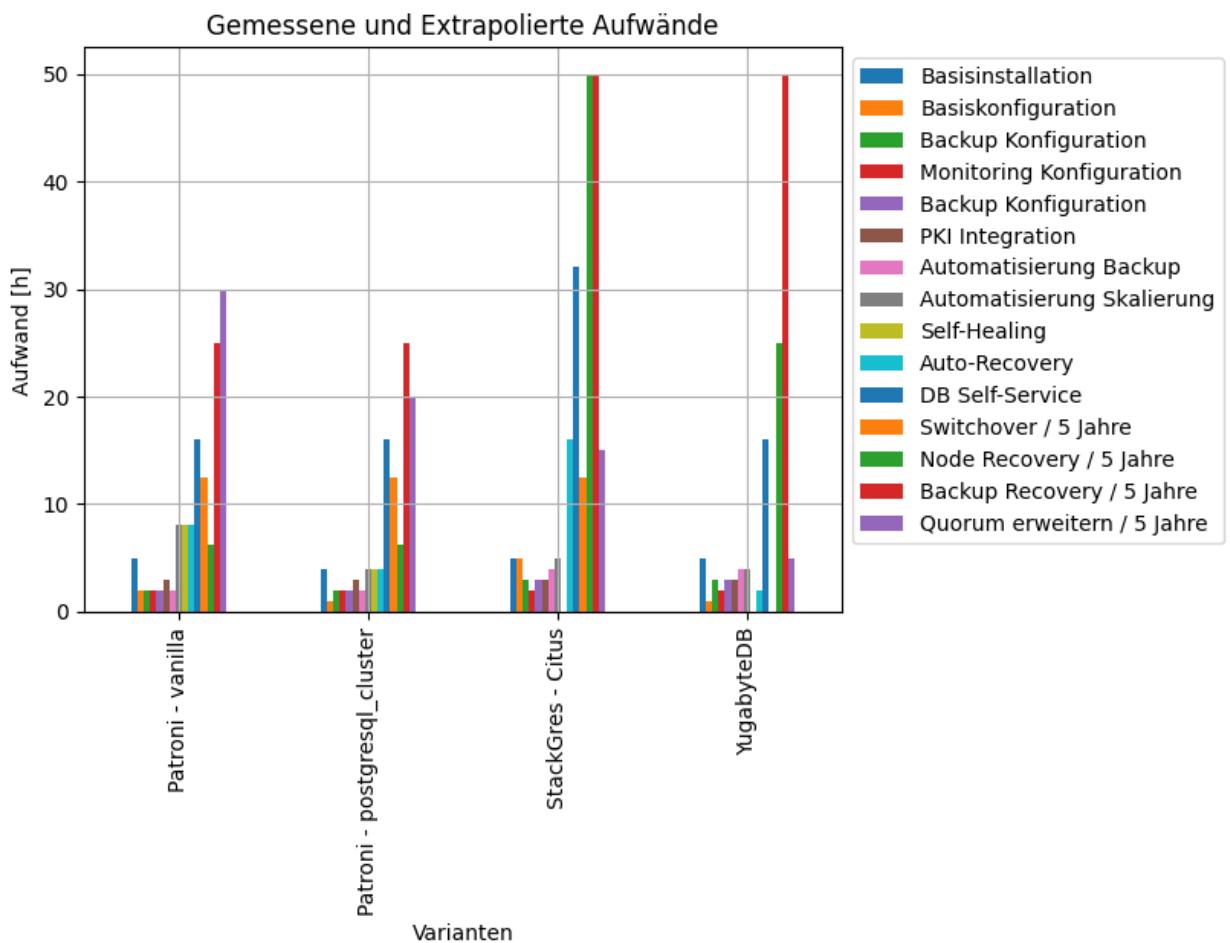


Abbildung 3.51: Zeitaufwände

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

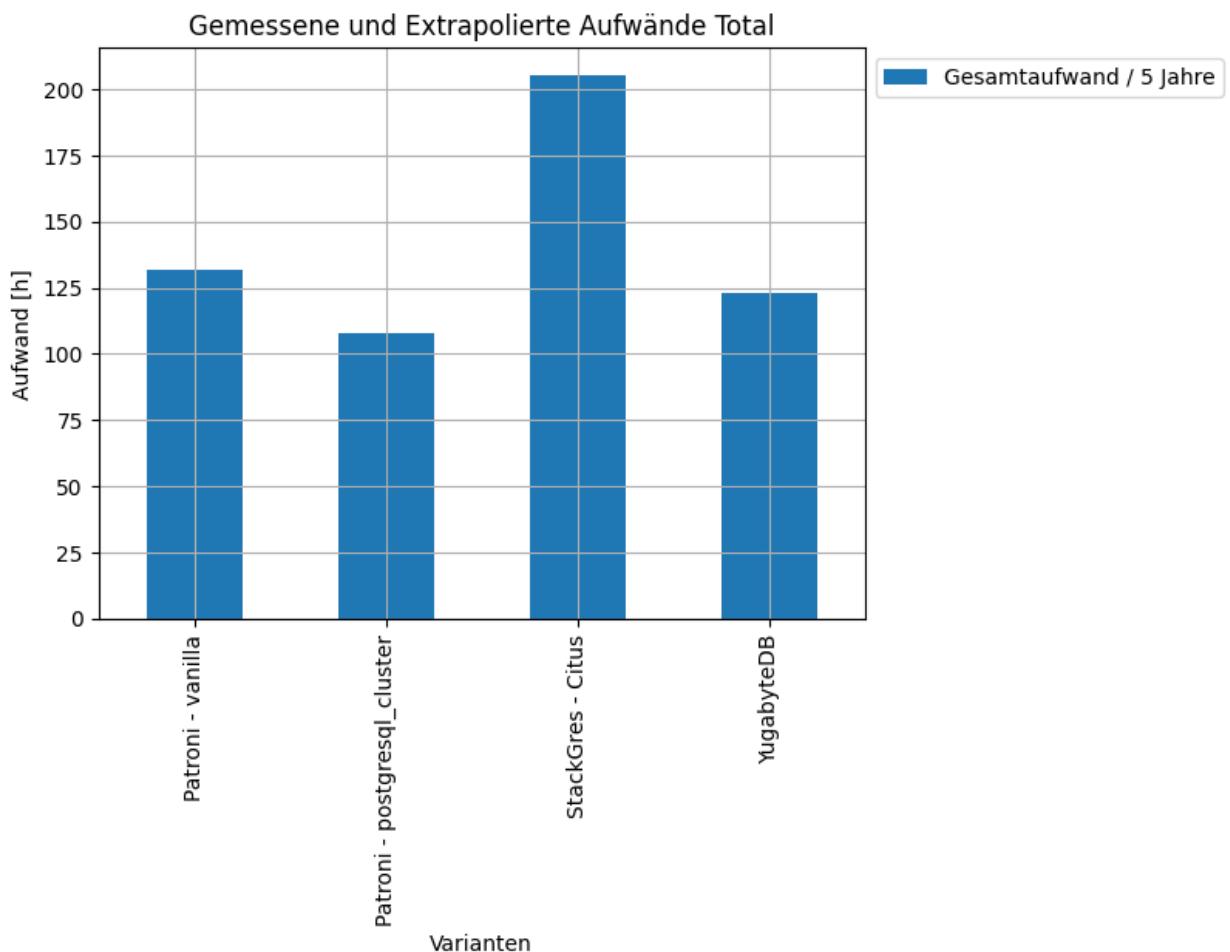


Abbildung 3.52: Zeitaufwände summiert

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:

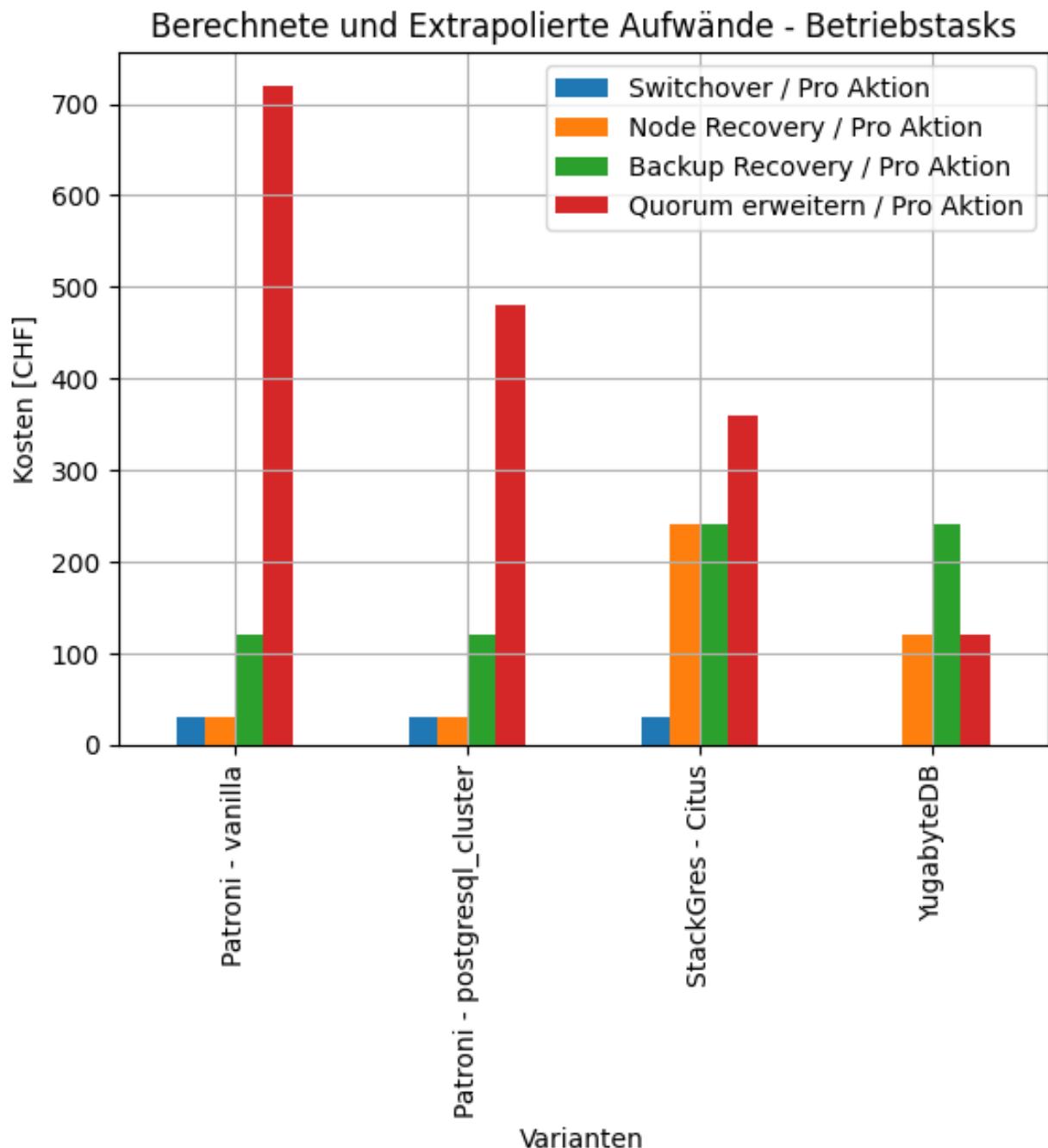


Abbildung 3.53: Kostenaufwände pro Betriebstask

Entsprechend entstehen ist auf fünf Jahre verteilt, mit folgenden Kosten zu rechnen:

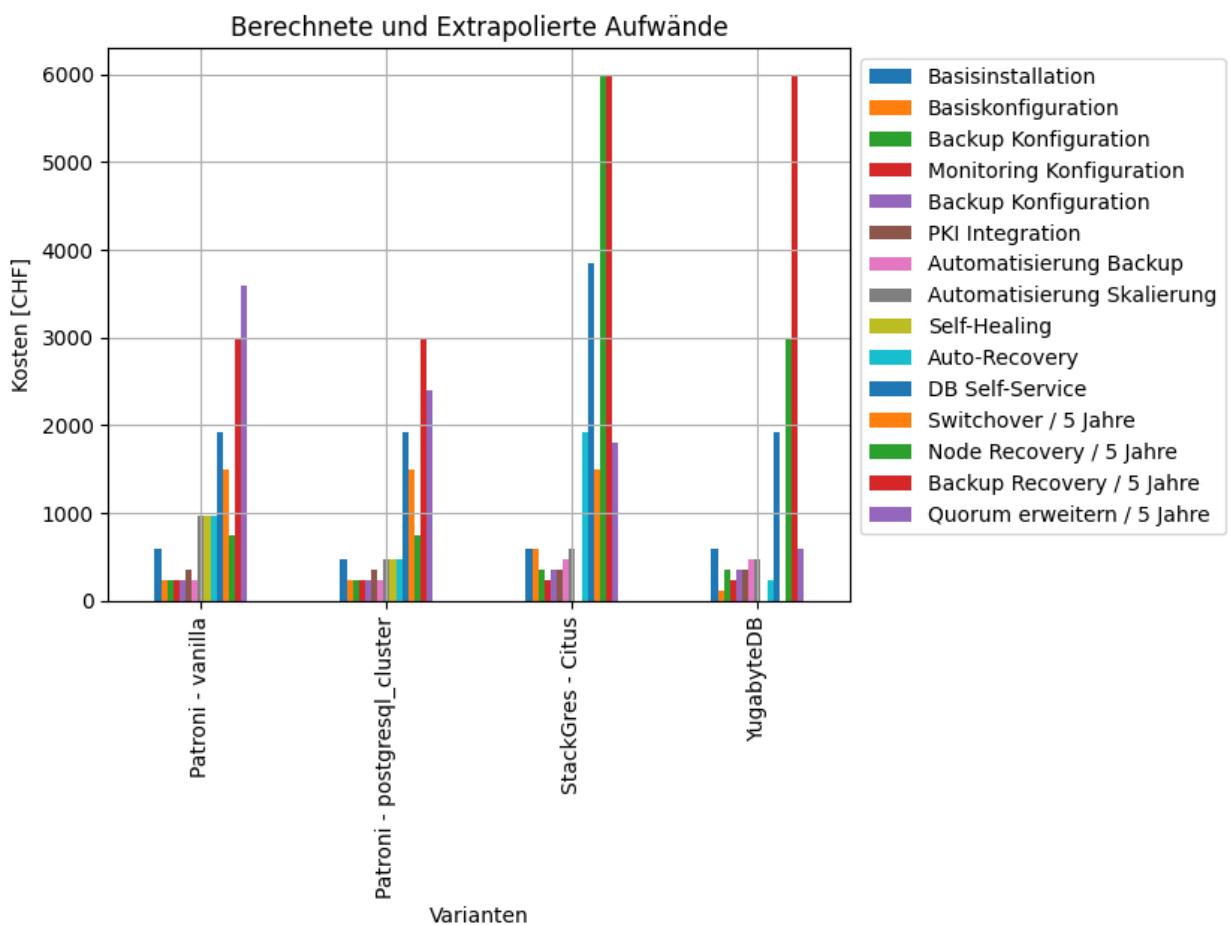


Abbildung 3.54: Kostenaufwände

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

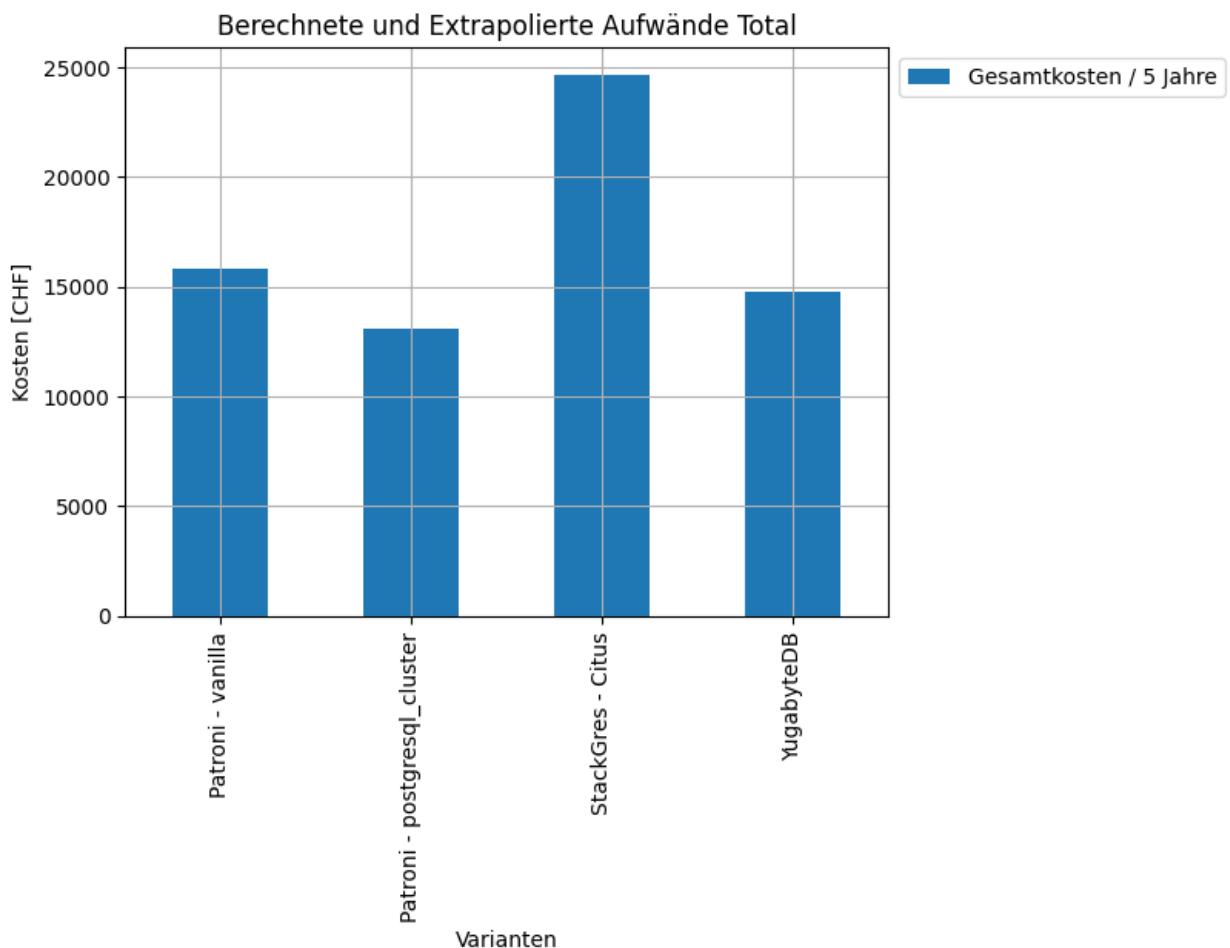


Abbildung 3.55: Kostenaufwände Total

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		3	75	Nur mit Zweck-entfremdung möglich	2	50	Kein Sharding möglich	0	0
8 Quorum	67		3	200		3	200		2	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 Klassisches Veeam eher schwierig	1	58		3	175
12 Housekeeping - Log Rotation	8		3	25		3	25		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	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 relativ 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 Diagramm:

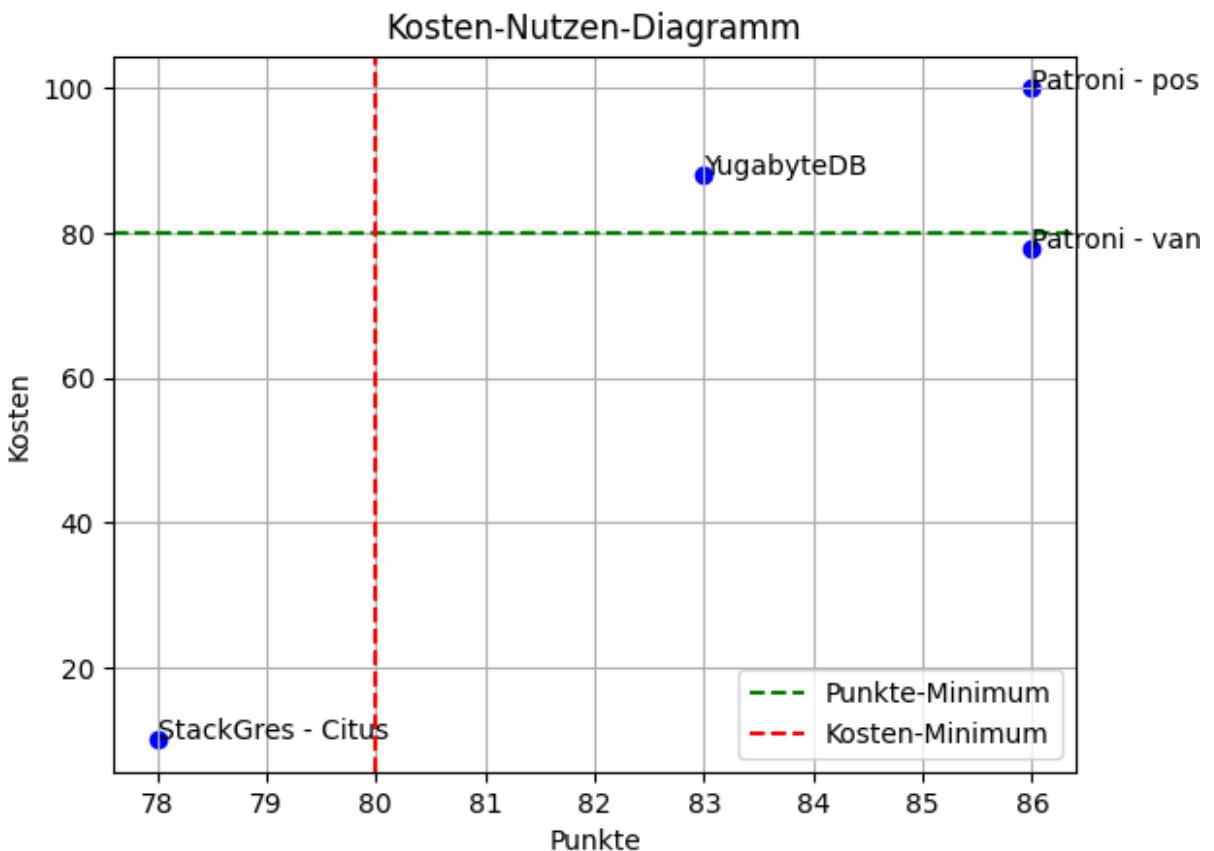


Abbildung 3.58: Kosten-Nutzen-Diagramm

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:

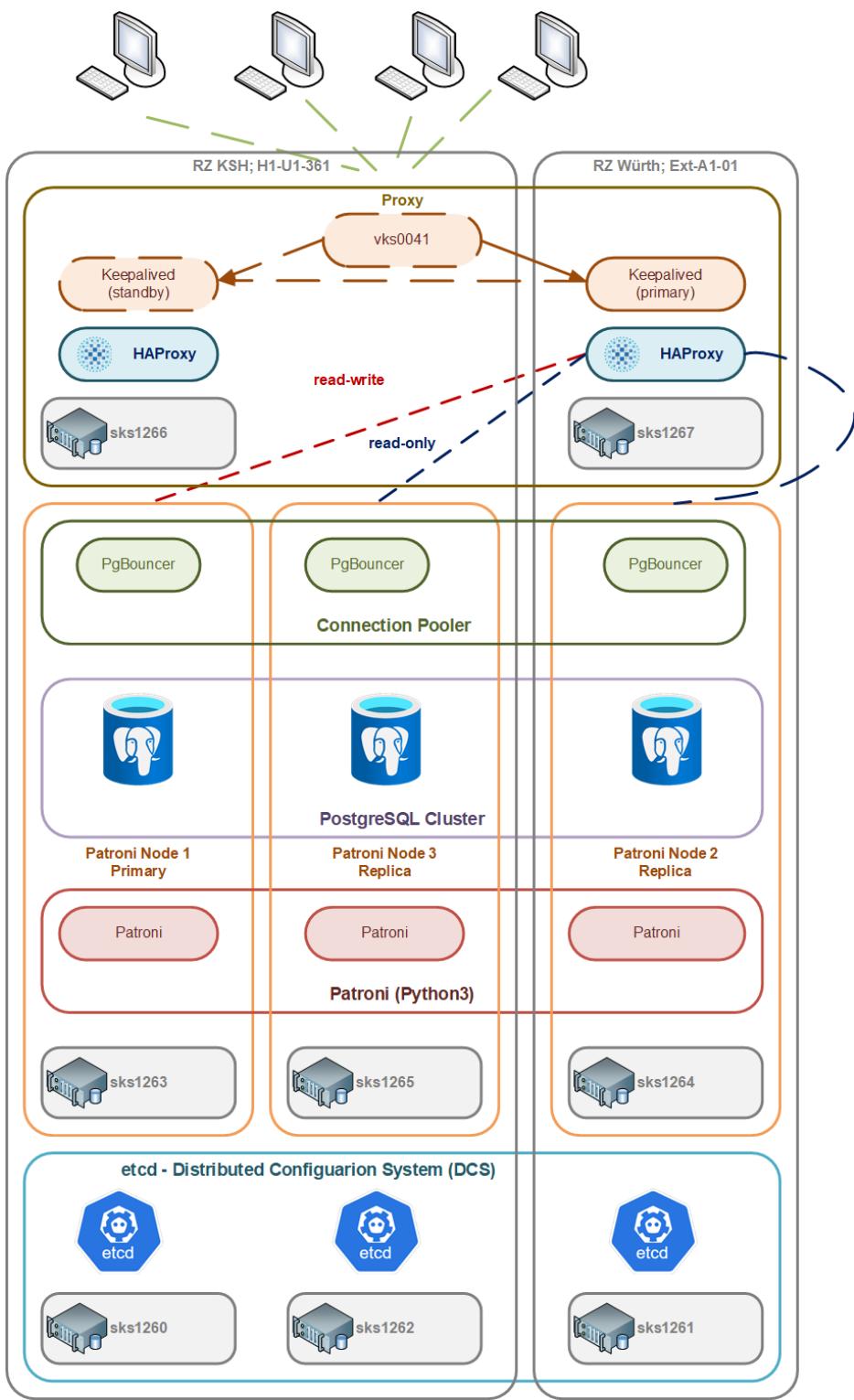


Abbildung 3.59: Testsystem - Architektur

vitabcks / 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 Timeouts 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 mittels 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-API 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 gng 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 verwendet, wird folgendens eingetragen:

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

Listing 3.28: 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 3.29: 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 3.30: 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 3.31: 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 3.32: 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 verstand.

Zum Testen wurden nachträglich folgende Werte hinzugefügt:

- Der Proxy wurde gesetzt
- Der REST-API wurde die Patroni-Adresse mittels `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 3.33: HAProxy Node erweitern - Inventory

Zum Schluss muss nur noch das Playbook ausgeführt werden:

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

Listing 3.34: 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 3.35: Patroni Node erweitern - Inventory

Das Deployment erfolgt nun ebenfalls via Playbook:

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

Listing 3.36: Patroni Node erweitern - add_pgnode.yml

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

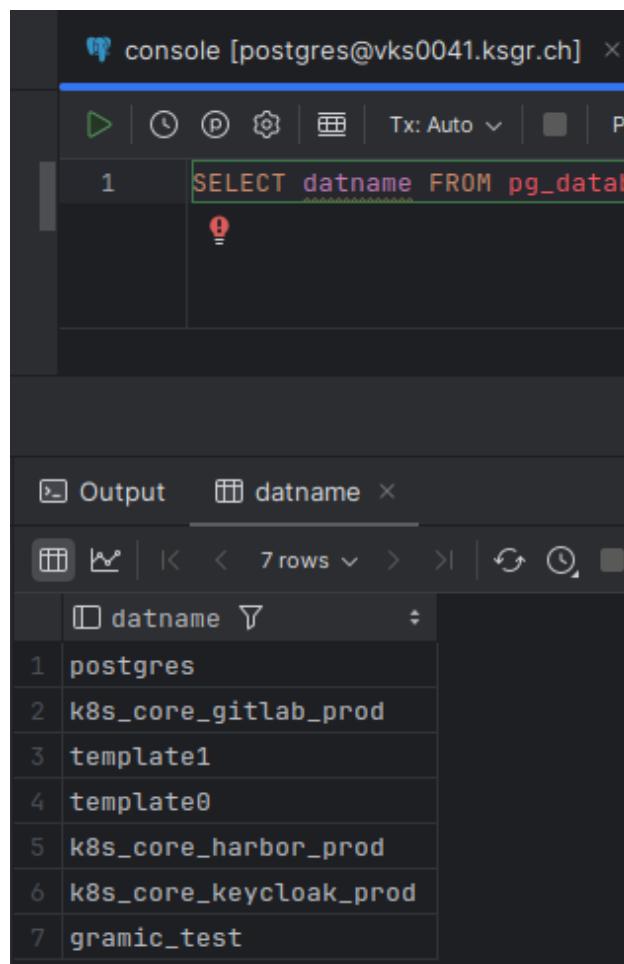
3.2.4 Technical Review der Umgebung

Die beiden Patroni-Nodes sks1264 und sks1265 sind Synchrone Replicas 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:

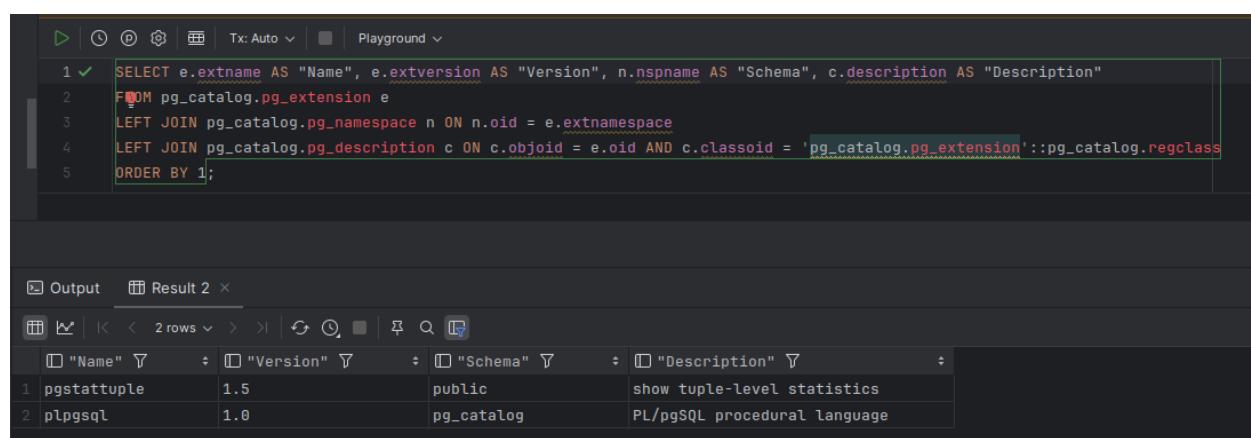


The screenshot shows a PostgreSQL terminal window with the title "console [postgres@vks0041.ksgr.ch]". In the main pane, a query is being typed: "SELECT datname FROM pg_database". Below the query, the results are displayed in a table titled "Output" with the column "datname". The results show seven databases: postgres, k8s_core_gitlab_prod, template1, template0, k8s_core_harbor_prod, k8s_core_keycloak_prod, and gramic_test.

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 with the title "Playground". A query is being run to list installed extensions from the pg_catalog.pg_extension catalog. The results are shown in a table titled "Result 2" with columns "Name", "Version", "Schema", and "Description". Two extensions are listed: pgstattuple (version 1.5, schema public, description "show tuple-level statistics") and plpgsql (version 1.0, schema pg_catalog, description "PL/pgSQL procedural language").

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:

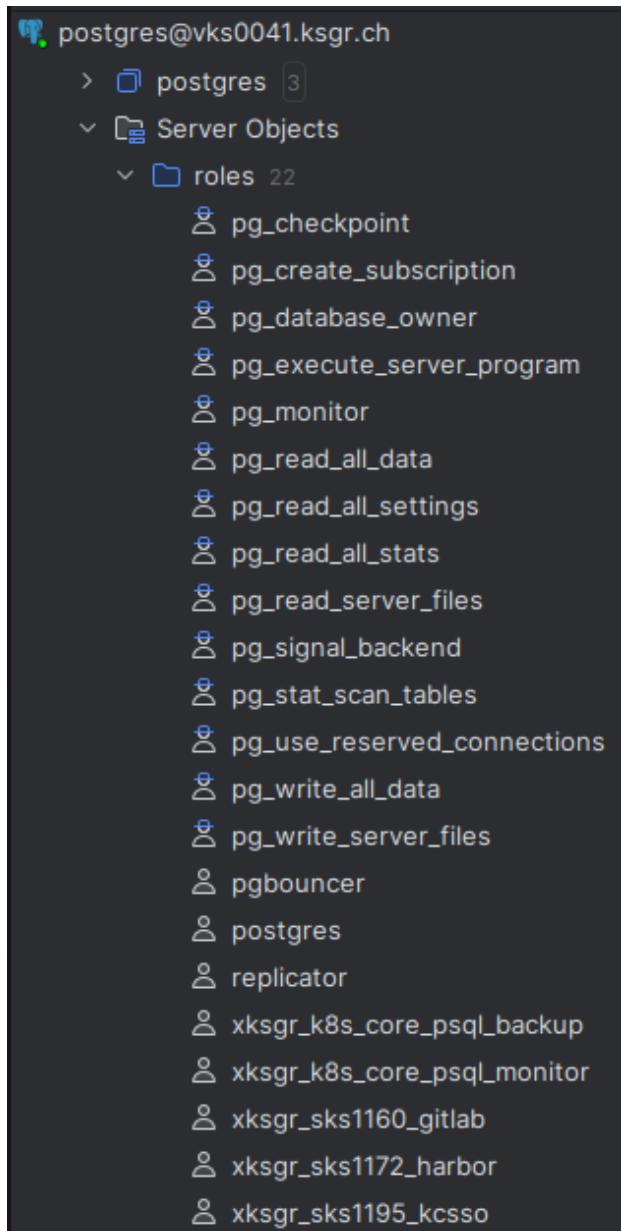


Abbildung 3.63: Technical Review - PostgreSQL User und Rollen

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, rlimit = 1, rlimitread = 4)
system id: 0x12345678
system maxmem = unlimited, ulimit = 200000
maxconn = 20000, maxconn1 = 20000, maxpipes = 0
current conn = 4, current pipes = 0
conn rate = 0/sec, per sec = 0.719 kbps
Running tasks: 0/0, idle: 0/0, iowait: 0
```

STATS															Display option:										External resources:															
Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			Server		Weight			Act		Bck		Chk		Dom		Down		Throttle	
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Resp	Conn	Rerr	Retr	Redis	Status	Lauchk	Wght	Act	Bck	Chk	Dom	Down	Throttle										
Frontend	0	2	-	2	3	-	2	3	100 000	3	1 157	134 395	0	0	0	1	0	0	0	0	0	OPEN		0/0	0	0	0	0	0	0	0	0								
Backend	0	0	-	0	0	-	0	0	10 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		0/0	0	0	0	0	0	0	0	0									
sk1263	0	0	-	0	1	-	0	1	-	4	2h15m	40 652	95 279	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	0	0	0	0	0	0								
sk1264	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h10m DOWN		LT0K200 in 2hrs	1/1	Y	-	1	1	0	0	0	0								
sk1265	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h10m DOWN		LT0K200 in 2hrs	0/1	Y	-	2	2	0	0	0	0								
Backend	0	0	-	0	1	-	0	1	1 000	4	2h15m	40 652	95 279	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	0	0	0	0	0	0								
sk1266	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	1	1	0	0	0	0									
Backend	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		2/2	Z	-	2	0	0	0	0	0									
sk1267	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		2/2	Z	-	1	1	0	0	0	0									

Abbildung 3.64: Technical Review - HAProxy sks1266



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

Statistics Report for pid 58804

> General process information

```
pid 58804 (process #1, rlimit = 4)
system id: 0x12345678
system maxmem = unlimited, ulimit = 200000
maxconn = 20000, maxconn1 = 20000, maxpipes = 0
current conn = 4, current pipes = 0
conn rate = 0/sec, per sec = 0.000 kbps
Running tasks: 0/0, idle: 0/0, iowait: 0
```

STATS															Display option:										External resources:															
Queue			Session rate			Sessions			Bytes			Denied			Errors			Warnings			Status			Server		Weight			Act		Bck		Chk		Dom		Down		Throttle	
Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Resp	Conn	Rerr	Retr	Redis	Status	Lauchk	Wght	Act	Bck	Chk	Dom	Down	Throttle										
Frontend	0	2	-	2	3	-	2	3	10 000	3	423	44 656	0	0	1	0	0	0	0	0	0	OPEN		0/0	0	0	0	0	0	0	0	0								
Backend	0	0	-	0	0	-	0	0	10 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		0/0	0	0	0	0	0	0	0	0									
sk1263	0	0	-	0	1	-	0	1	-	4	2h15m	11 405	11 069	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	0	0	0	0	0	0								
sk1264	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h10m DOWN		LT0K200 in 2hrs	1/1	Y	-	1	1	0	0	0	0								
sk1265	0	0	-	0	0	-	0	0	-	0	0	0	0	0	0	0	0	0	0	0	2h10m DOWN		LT0K200 in 2hrs	0/1	Y	-	2	2	0	0	0	0								
Backend	0	0	-	0	1	-	0	1	1 000	7	11m59s	11 405	11 069	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	0	0	0	0	0	0								
sk1266	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		1/1	Y	-	1	1	0	0	0	0									
Backend	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		2/2	Z	-	2	0	0	0	0	0									
sk1267	0	0	-	0	0	-	0	0	1 000	0	0	0	0	0	0	0	0	0	0	0	2h20m UP		2/2	Z	-	1	1	0	0	0	0									

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 3.37: PgBouncer - Connect

Die Abfrage der Tabellen wird dann mittels folgendem Command gemacht:

```
1 show databases;
```

Listing 3.38: 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, wird 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 Primary-Node wird manuell rebooted.

Switchover

5. Mit dem `patronictl`-Command wird der Switchover 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 der Node wiederhergestellt und abschliessend mittels Switchover 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 ein 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 Beim Restore des Primary-Nodes dürfen keine Daten, die seit dem Failover geschrieben wurden, darf es zu keinem Datenverlust kommen	Eingetroffen	
Restore	9	Datensicherheit		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

3.4 Maintenance-Tool

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:

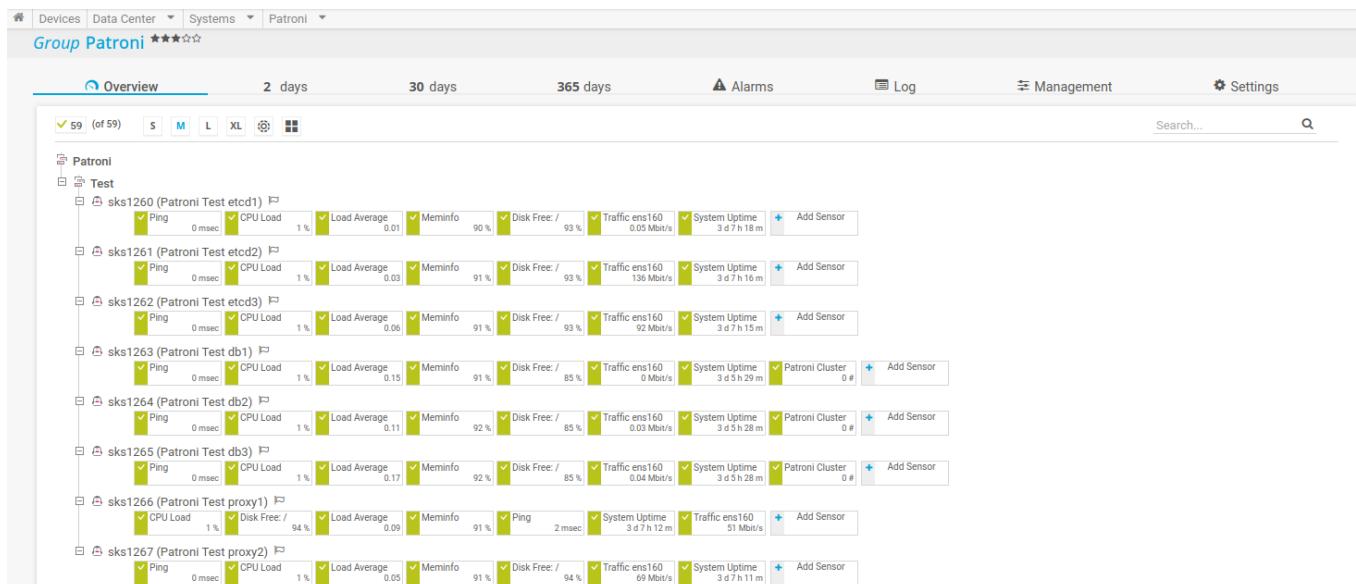


Abbildung 3.69: PRTG - Patroni Monitoring

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

Cluster Unlock

Failsafe Mode Activity

Replication Lag

Replication State

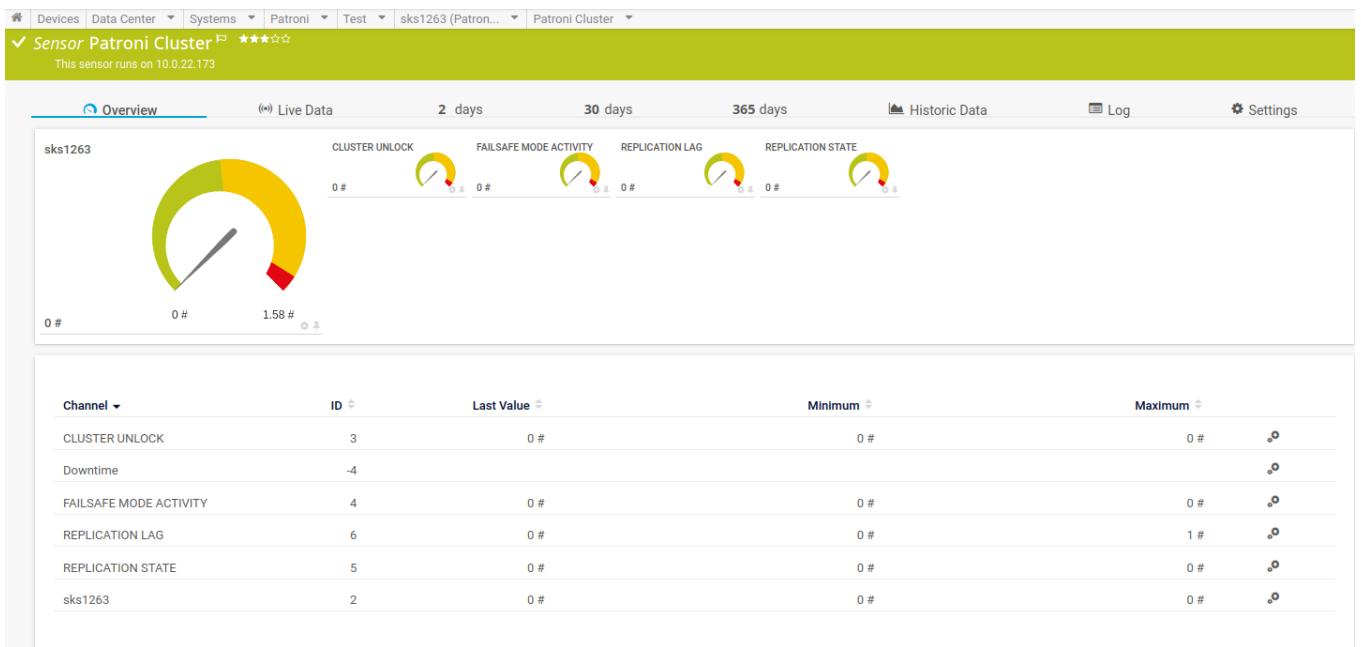


Abbildung 3.70: PRTG - Patroni Cluster Sensor

3.6 Troubleshooting und Lösungsfindung

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

Evaluation - Yugaware

Erst wurde das Lizenzpflichtige 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 Stanbdalone 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 Loead 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 containiderd-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 mittels http erzwungen wird.

Da es sich nur um eine Evaluationsplattform handelte, kam letztere Lösung zum Einsatz.

Evaluation - Benchmarking automatisation

Ursprünglich war der Plan, dass die Benchmarks mittels pgbench resp. mysql_bench mittels 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 gesetzt 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**
- 4.2 Schlussfolgerung**
- 4.3 Weiteres Vorgehen / offene Arbeiten**
- 4.4 Persönliches Fazit**

Abbildungsverzeichnis

1.1	Spitalregionen Kanton Graubünden[61]	1
1.2	Wahlkreise Kanton St. Gallen[88]	2
1.3	Spitalregionen / Spitalstrategie Kanton St. Gallen[55]	3
1.4	Organigramm Kantonsspital Graubünden	4
1.5	Organigramm Departement 10 - ICT	5
1.6	Datenbanken - Aufgeschlüsselt nach RDBMS	8
1.7	Datenbanken - Aufgeschlüsselt nach Betriebssystem	9
1.8	Risikomanagement PostgreSQL	13
1.9	Systemabgrenzung	18
1.10	Risikomanagement Projekt	24
1.11	Riskikomatrix - Assessment 21.03.2024	26
1.12	Riskikomatrix - Assessment 29.04.2024	28
2.1	Projektcontrolling	31
3.1	Sharding - Vertikale Partitionierung	34
3.2	Sharding - Horizontales Partitionierung	35
3.3	Monolithische vs. verteilte SQL Systeme	37
3.4	CAP-Theorem	39
3.5	Datenbankskalierung	40
3.6	Präferenzmatrix	44
3.7	Testing - ERD DB self_healing_test	47
3.8	Benchmark Settings - Zabbix - Systeminformationen	48
3.9	Benchmark Settings - Zabbix - Connections per Seconds	49
3.10	Benchmark Settings - Zabbix - Queries per Seconds	49
3.11	Benchmark Settings - Zabbix - Client Queries per Seconds	49
3.12	Benchmark Settings - Zabbix - DB Size	50
3.13	Benchmark Settings - Anzahl Records / Skalierungsfaktor	52
3.14	pg_auto_failover-Architektur - Single Standby	56
3.15	pg_auto_failover-Architektur - Multi-Node Standby	56
3.16	pg_auto_failover-Architektur - Citus	57
3.17	CloudNativePG - Kubernetes - PostgreSQL	58
3.18	CloudNativePG - Kubernetes - Read-write workloads	59
3.19	CloudNativePG - Kubernetes - Read-only workloads	59
3.20	Patroni-Architektur	62
3.21	Stackgres - Grundarchitektur	65
3.22	Citus - Coordinator und Workers	66
3.23	Citus - Row-Based-Sharding	66
3.24	Citus - Schema-Based-Sharding	67
3.25	StackGres-Citus - Shard-Replikation	68

3.26 YugabyteDB - Grundkonzept	70
3.27 YugabyteDB - Architektur	70
3.28 YugabyteDB - Sharding	71
3.29 YugabyteDB - Tablet - Leader und Follower	71
3.30 YugabyteDB - Tablet - Replikationsfaktor	72
3.31 YugabyteDB - Zonen	73
3.32 YugabyteDB - Zone outage Tolerance	74
3.33 Evaluationssystem - Distributed SQL / Shards	76
3.34 Patroni - Evaluationsarchitektur	77
3.35 Stackgres - Citus - Evaluationsarchitektur Benchmarking	80
3.36 Stackgres - Citus - Evaluationsarchitektur Self Healing Tests	80
3.37 Stackgres - Citus - Ressourcen - Stack	81
3.38 Stackgres - Citus - Datenbank - Cluster	82
3.39 YugabyteDB - Evaluationsarchitektur	87
3.40 YugabyteDB - Subscription Yugaware	88
3.41 Benchmarking - ERD pgbench	98
3.42 Benchmarks - tps	100
3.43 Benchmarks - latency	100
3.44 Benchmarks - tps Patroni Replica	101
3.45 Benchmarks - latency Patroni Replica	101
3.46 Benchmarks - Fehler bei mixed-Transaktionen	102
3.47 Benchmarks - Initialisierungszeit - sekunden	102
3.48 Benchmarks - Initialisierungszeit - Minuten	103
3.49 Benchmarks - Initialisierungszeit - Stunden	103
3.50 Zeitaufwände pro Betriebstask	106
3.51 Zeitaufwände	107
3.52 Zeitaufwände summiert	108
3.53 Kostenaufwände pro Betriebstask	109
3.54 Kostenaufwände	110
3.55 Kostenaufwände Total	111
3.56 Kosten-Nutzen-Analyse	112
3.57 Kosten-Nutzen-Ranking	112
3.58 Kosten-Nutzen-Diagramm	113
3.59 Testsystem - Architektur	115
3.60 Technical Review - Patroni Nodes	121
3.61 Technical Review - Datenbanken	122
3.62 Technical Review - PostgreSQL Extensions	122
3.63 Technical Review - PostgreSQL User und Rollen	123
3.64 Technical Review - HAProxy sks1266	124
3.65 Technical Review - HAProxy sks1267	124
3.66 Technical Review - PgBouncer - Datenbanken	125
3.67 Technical Review - PostgreSQL - Log Rotation	125
3.68 Testing - Patroni Node hinzufügen - add_pgnode.yml	129
3.69 PRTG - Patroni Monitoring	129

3.70 PRTG - Patroni Cluster Sensor	130
I CloudNativePG - Pulse	xix
II CloudNativePG - Code Frequency	xix
III CloudNativePG - Community Standards	xx
IV CloudNativePG - Contributors Commits	xx
V CloudNativePG - Contributors Deletations	xxi
VI CloudNativePG - Contributors Additions	xxi
VII CloudNativePG - Commit Activity	xxi
VIII CloudNativePG - Network Graph	xxii
IX Patroni - Pulse	xxii
X Patroni - Code Frequency	xxiii
XI Patroni - Community Standards	xxiii
XII Patroni - Contributors Commits	xxiv
XIII Patroni - Contributors Deletations	xxiv
XIV Patroni - Contributors Additions	xxiv
XV Patroni - Commit Activity	xxv
XVI Patroni - Network Graph	xxv
XVII Stackgres - Pulse	xxvi
XVIII Citus - Pulse	xxvi
XIX Stackgres - Code Frequency	xxvi
XX Citus - Code Frequency	xxvii
XXI Stackgres - Community Standards	xxvii
XXII Citus - Community Standards	xxviii
XXIII Stackgres - Contributors Commits	xxviii
XXIV Stackgres - Contributors Deletations	xxviii
XXV Stackgres - Contributors Additions	xxix
XXVI Citus - Contributors Commits	xxix
XXVII Citus - Contributors Deletations	xxix
XXVIII Citus - Contributors Additions	xxix
XXIX Stackgres - Commit Activity	xxx
XXX Citus - Commit Activity	xxx
XXXI Stackgres - Network Graph	xxxi
XXXII Citus - Network Graph	xxxi
XXXIII YugabyteDB - Pulse	xxxii
XXXIV YugabyteDB - Code Frequency	xxxii
XXXV YugabyteDB - Community Standards	xxxiii
XXXVI YugabyteDB - Contributors	xxxiii
XXXVII YugabyteDB - Commit Activity	xxxiv
XXXVIII YugabyteDB - Network Graph	xxxiv
XXXIX Aproxy - Web-GUI	lxxvii
XL StackGres Testing - Node sks1184 down	cx
XLI StackGres Testing - Pods Down	cx
XLII StackGres Testing - Patroni Übersicht	cx

XLIII StackGres Testing - DB Zugriff	cxii
XLIV StackGres Testing - Connection Timeout	cxiii
XLV YugabyteDB - Too big clock skew is detected	cxiv
XLVI YugabyteDB - Tablet Leader - No Lease	cxv
XLVII YugabyteDB - CrashLoopBackOff	cxv
XLVII YugabyteDB - Too big clock skew is detected - tmaster	cxvi
XLIX YugabyteDB - Too big clock skew is detected - tserver	cxvii

Tabellenverzeichnis

1.1 Inventarisierte Datenbanksysteme	7
1.2 Datenbankinventar	8
1.3 Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt	9
1.4 Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken	12
1.5 Administrative Aufgaben	14
1.6 Automatisierung Administrativer Aufgaben	15
1.7 Ziele	16
1.8 Gegebene Systeme	17
1.9 Abhängigkeiten	20
1.10 Risiko-Matrix der Diplomarbeit	22
1.11 Neu Erkannte / Erfasste Risiken	25
1.12 Risiko-Assessment 21.03.2024	25
1.13 Risiko-Assessment 29.04.2024	27
2.1 Projektcontrolling	30
2.2 Fachgespräche	33
3.1 Quorum Beispiele	38
3.2 Anforderungskatalog	43
3.3 Stakeholder	44
3.4 Benchmark Settings - Mixed Transaktionen	50
3.5 Benchmark Settings - DQL Transaktionen	50
3.6 Benchmark Settings - Skalierungsfaktoren	51
3.7 Benchmark Settings - Datenbankgrößen / Skalierungsfaktor	51
3.8 Benchmark Settings - Tabellengrößen / Skalierungsfaktor	51
3.9 Vorauswahl - Ausgeschieden	75
3.10 Vorauswahl - Evaluation	75
3.11 Evaluationssystems	76
3.12 Evaluationssystem - Distributed SQL / Sharding	76
3.13 Testresultate Evaluation Patroni	94
3.14 Testresultate Evaluation StackGres - Citus	95
3.15 Testresultate Evaluation YugabyteDB	95
3.16 Kostenberechnung - Annahmen	104
3.17 Gemessene und Extrapolierte Aufwände Bsp.	105
3.18 Testsystem - Komponenten	114
3.19 Testsystem - Inventar	116
3.20 Testsystem - Auxiliar Inventar	116
3.21 Testsystem - Benötigte Ports	118
3.22 Testresultate Testsystem	128

I	Arbeitsrapport	ii
II	Kommentare - Anmerkung	xvii

Listings

3.1 Patroni - etcd API V2 Error	78
3.2 Patroni - etcd API V2 Enable	78
3.3 Patroni - etcd3 Flag	78
3.4 Patroni - Passwörter	78
3.5 Patroni - Synchrone Replikation setzen	79
3.6 StackGres - values.yaml - Extension proxyUrl	83
3.7 StackGres - values.yaml - Extension Proxy	83
3.8 StackGres-Citus - LoadBalancer -Annotation	85
3.9 StackGres-Citus - StorageClass -PVC Binding	85
3.10 StackGres-Citus - Instanz-Profile	85
3.11 StackGres-Citus - StorageClass -PVC Binding	86
3.12 StackGres-Citus - Cluster Profil	86
3.13 metallb - Konfig YAML - Detail L2Advertisement	88
3.14 YugabyteDB - Helm Chart Manifest - Detail Image	89
3.15 YugabyteDB - Helm Chart Manifest - Detail StorageClass	89
3.16 YugabyteDB - Helm Chart Manifest - Detail Resources	89
3.17 YugabyteDB - Helm Chart Manifest - Detail Replika	90
3.18 YugabyteDB - Helm Chart Manifest - Detail Disable YSQL	90
3.19 YugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints	91
3.20 local-path-provisioner nodePathMap	91
3.21 local-path-provisioner nodePathMap Beispiel	92
3.22 YugabyteDB - StorageClass nodeAffinity	92
3.23 Patroni - Testing - Switchover	93
3.24 Patroni - Testing - Reinit	94
3.25 Patroni - Benchmarking - Monitoring	96
3.26 Citus - Benchmarking - Distributed Table Sharding	97
3.27 Citus - Benchmarking - Reference Table Sharding	98
3.28 main.xyml - No Proxy	119
3.29 main.xyml - Gruppenproxy	119
3.30 main.xyml - SProxy	119
3.31 Deploy - deploy_pgcluster.yml	119
3.32 Maintenance - config_pgcluster.yml	120
3.33 HAProxy Node erweitern - Inventory	120
3.34 HAProxy Node erweitern - add_balancer.yml	121
3.35 Patroni Node erweitern - Inventory	121
3.36 Patroni Node erweitern - add_pgnode.yml	121
3.37 PgBouncer - Connect	125
3.38 PgBouncer - Databases	125
1 Proxy Settings	xxxiv

2	rke2 server - Verzeichnis erstellen	xxxv
3	rke2 server - config.yaml	xxxv
4	rke2 server - cilium-config.yaml	xxxv
5	rke2 server installieren	xxxv
6	rke2 agenten installieren	xxxvi
7	rke2 agent - config.yaml	xxxvi
8	-rke2 agent service restart	xxxvi
9	rke2 server proxy	xxxvi
10	rke2 server proxy kopieren	xxxvi
11	rke2 server cilium installieren	xxxvi
12	rke2 server cilium aktivieren	xxxvi
13	rke2 server starten	xxxvi
14	iptables entries server	xxxvii
15	rke2 server token	xxxvii
16	local-path-storage auf Linux Bereitstellen	xxxvii
17	local-path-provisioner definieren	xxxvii
18	local-path-storage aktualisieren	xxxviii
19	MetallB installieren	xxxviii
20	MetallB konfigurieren	xxxviii
21	MetallB Konfiguration einspielen	xxxix
22	rke2 - 250GiB Disk mount	xxxix
23	local-path-storage 250GiB auf Linux Bereitstellen	xxxix
24	local-path-provisioner Grosse Volumes	xl
25	local-path-storage 250GiB aktualisieren	xli
26	yugabyteDB - StorageClass setzen	xli
27	yugabyteDB - StorageClass / PersistentVolume aktivieren	xli
28	yugabyteDB - Namespace	xlii
29	yugabyteDB - Helm Chart Manifest	xlii
30	yugabyteDB - Installation	lii
31	yugabyteDB - Deinstallieren	lii
32	yugabyteDB - StorageClass setzen	lii
33	yugabyteDB - StorageClass / PersistentVolume Grosse Volumes aktivieren	liii
34	yugabyteDB - Namespace 250GiB	liii
35	yugabyteDB - Helm Chart Manifest 250GiB	liii
36	yugabyteDB - Cloud - Region - Zone	lxii
37	yugabyteDB - Benchmarking - DB erstellen	lxii
38	yugabyteDB - Benchmarking - Table Size	lxii
39	YugabyteDB - Self Healing Tests - CREATE-SQL	lxii
40	YugabyteDB - Self Healing Tests - Init Data	lxiv
41	YugabyteDB - Self Healing Tests - Failover Data	lxv
42	YugabyteDB - Self Healing Tests - Recovery Data	lxv
43	sks9016 - Download YugabyteDB On-Premise	lxvi
44	sks9016 - Installation YugabyteDB On-Premise	lxvi
45	sks9016 - Check YugabyteDB On-Premise	lxvi

46	Patroni - Firewall Settings	lxvii
47	Patroni - Proxy Settings	lxvii
48	Patroni - apt-Proxy Settings	lxvii
49	Patroni - PostgreSQL einbinden	lxviii
50	Patroni - Prerequisites installieren	lxviii
51	Patroni - Stop Patroni und PostgreSQL	lxviii
52	Patroni - Symlink binaries	lxviii
53	Patroni - Checks	lxviii
54	etcd - Installation	lxviii
55	etcd - Konfiguration	lxix
56	etcd - restart	lxix
57	etcd - member list	lxix
58	Patroni Bootstrap - Konfiguration bereinigen	lxix
59	Patroni - Konfiguration - sks1232	lxix
60	Patroni - Konfiguration - sks1233	lxxi
61	Patroni - Konfiguration - sks1234	lxxiii
62	Patroni Bootstrap - pg_hba	lxxiv
63	Patroni Bootstrap - Patroni-Verzeichnis	lxxv
64	Patroni Bootstrap - Neu starten	lxxv
65	Patroni Bootstrap - Disable PostgreSQL	lxxv
66	HAproxy - Hostliste	lxxv
67	HAproxy - Installation	lxxv
68	HAproxy - Safe Alte Config	lxxv
69	HAproxy - Konfiguration	lxxv
70	HAproxy - Restart	lxxvi
71	Patroni - 250GiB Disk mount	lxxvii
72	Patroni - 250GiB Verzeichnisse	lxxvii
73	Patroni - 250GiB Cluster Pause	lxxviii
74	Patroni - 250GiB PostgreSQL stoppen	lxxviii
75	Patroni - 250GiB move pg_wal	lxxviii
76	Patroni - 250GiB chmod - chown pg_wal	lxxviii
77	Patroni - 250GiB PostgreSQL - Patroni resume	lxxviii
78	Patroni - 250GiB Finaler Check	lxxviii
79	Patroni - 250GiB set Parameter	lxxxi
80	Patroni - Benchmarking - DB erstellen	lxxxii
81	Patroni - Benchmarking - DB Cleanup	lxxxii
82	Patroni - Benchmarking - Tablespaces erneut erstellen	lxxxii
83	Patroni - Self Healing Tests - CREATE-SQL	lxxxii
84	Patroni - Self Healing Tests - Init Data	lxxxv
85	Patroni - Self Healing Tests - Failover Data	lxxxv
86	Patroni - Self Healing Tests - Recovery Data	lxxxvi
87	StackGres-Citus - StorageClass setzen	lxxxvi
88	StackGres-Citus - StorageClass / PersistentVolume aktivieren	lxxxvii
89	StackGres-Citus - Namespace	lxxxvii

90	StackGres-Citus - Helm Chart Manifest	lxxxvii
91	StackGres-Citus - Installation	xciv
92	StackGres-Citus - Post-Installation	xciv
93	StackGres-Citus - System Username	xciv
94	StackGres-Citus - System Passwort	xciv
95	StackGres-Citus - System Passwort Cleanup	xcv
96	StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator	xcv
97	StackGres-Citus - Benchmarking - SGInstanceProfile Shard	xcv
98	StackGres-Citus - Benchmarking - Instanz-Profil Deploy	xcv
99	StackGres-Citus - Benchmarking - SGShardedCluster	xcv
100	StackGres-Citus - Benchmark - Cluster Deploy	xcvi
101	StackGres-Citus - Benchmark DB Passwort	xcvi
102	StackGres-Citus - Self Healing Testing - SGInstanceProfile Coordinator	xcvi
103	StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard	xcvii
104	StackGres-Citus - Self Healing Testing - Instanz-Profil Deploy	xcvii
105	StackGres-Citus - Self Healing Testing - SGShardedCluster	xcvii
106	StackGres-Citus - Self Healing Testing - Cluster Deploy	xcviii
107	StackGres-Citus - Self Healing Testing DB Passwort	xcviii
108	StackGres-Citus - Deinstallieren	xcviii
109	StackGres-Citus - StorageClass setzen	xcviii
110	StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren	xcix
111	StackGres-Citus - Namespace 250GiB	xcix
112	StackGres-Citus - Installation 250GiB	xcix
113	StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB	xcix
114	StackGres-Citus - Benchmarking - SGInstanceProfile Shard 250GiB	xcix
115	StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB	c
116	StackGres-Citus - Benchmarking - SGShardedCluster 250GiB	c
117	StackGres-Citus - Benchmark - Cluster Deploy 250GiB	ci
118	StackGres-Citus - Self Healing Tests - CREATE-SQL	ci
119	StackGres-Citus - Self Healing Tests - Init Data	ciii
120	StackGres-Citus - Self Healing Tests - Failover Data	civ
121	StackGres-Citus - Self Healing Tests - Recovery Data	civ
122	YugabyteDB - Benchmarking-Commands	cv
123	yugabyteDB - Benchmarking - Table Size SQL	cvi
124	Patroni - Benchmarking-Commands	cvi
125	Patroni - Benchmarking - Table Size SQL	cvi
126	StackGres-Citus - Benchmarking-Commands	cvi
127	StackGres-Citus - Benchmarking - Table Size SQL	cix
128	Ansible - Installation	cxvii
129	Ansible - Repository Clone	cxviii
130	Ansible - cd Repository	cxviii
131	Testsystem - Firewall Settings	cxviii
132	Testsystem - Proxy Settings	cxix
133	Testsystem - apt-Proxy Settings	cxix

134 Testsystem - Deployment - inventory	cixx
135 Testsystem - Deployment - main.yml	cxx
136 Deploy - Anhang - Ansible Ping	cxxxii
137 Testsystem - Maintenance - main.yml	cxxxii
138 Python LaTex - zotero.py - Zotero BibLaTex Importer	cxxxii
139 Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer	cxxxvii
140 Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer	cxxxvii
141 Python LaTex - riskmatrix.py - Risikomatrizen	cxliii
142 Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risikomatrizen	cxlvii
143 Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples	cl
144 Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm	cli
145 Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm	cliii
146 Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle	cliii
147 Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle	clix
148 Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm	clxxvii
149 Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme	clxxxii

Literatur

- [1] About pgbench-tools. <https://github.com/greg5104/pgbench-tools>. original-date: 2010-02-17T13:33:28Z 2023.
- [2] Satyadeep Ashwathnarayana und Inc. Netdata. *How to monitor and fix Database bloats in PostgreSQL? / Netdata Blog*. <https://blog.netdata.cloud/postgresql-database-bloat/>. 2022.
- [3] unknown author. #1 Backup-Lösung für Kubernetes. <https://www.veeam.com/de/kubernetes-native-backup.html?ck=1697900263871>.
- [4] unknown author. API Reference - CloudNativePG. <https://cloudnative-pg.io/documentation/1.22/cloudnative-pg.v1/>.
- [5] unknown author. API reference (for YSQL and YCQL). <https://docs.yugabyte.com/preview/api/>.
- [6] unknown author. Architecture Basics — pg_auto_failover 2.0 documentation. <https://pg-auto-failover.readthedocs.io/en/main/architecture.html>.
- [7] unknown author. Benchmark Setup with Citus and pgbench - Citus 12.1 documentation. https://docs.citusdata.com/en/stable/extra/write_throughput_benchmark.html.
- [8] unknown author. Benefits of using YugabyteDB. <https://docs.yugabyte.com/preview/features/>. Section: preview.
- [9] unknown author. Choosing Distribution Column - Citus 12.1 documentation. https://docs.citusdata.com/en/v12.1/sharding/data_modeling.html#distributed-data-modeling.
- [10] unknown author. Cilium - Cloud Native, eBPF-based Networking, Observability, and Security. <https://cilium.io>.
- [11] unknown author. Citus Replication Model: Today and Tomorrow - Replication Groups. <https://www.citusdata.com/blog/2016/12/15/citus-replication-model-today-and-tomorrow/>.
- [12] unknown author. Citus Support — pg_auto_failover 2.0 documentation. <https://pg-auto-failover.readthedocs.io/en/main/citus.html>.
- [13] unknown author. CLIs and command line tools. <https://docs.yugabyte.com/preview/admin/>.
- [14] unknown author. CloudNativePG - Main Features. <https://cloudnative-pg.io/documentation/1.22/#main-features>.
- [15] 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.
- [16] unknown author. Cluster Management — Citus Docs 7.2 documentation. https://docs.citusdata.com/en/v7.2/admin_guide/cluster_management.html.
- [17] unknown author. Concepts - Citus 12.1 documentation - row-based-sharding. https://docs.citusdata.com/en/v12.1/get_started/concepts.html#row-based-sharding.
- [18] unknown author. Consensus Protocol | Raft | Consul | HashiCorp Developer. <https://developer.hashicorp.com/consul/docs/architecture/consensus>.

- [19] unknown author. *Consul Architecture* | Consul | HashiCorp Developer. <https://developer.hashicorp.com/consul/docs/architecture>.
- [20] 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.
- [21] unknown author. *Dynamic Configuration Settings* — Patroni 3.2.2 documentation. https://patroni.readthedocs.io/en/latest/dynamic_configuration.html.
- [22] unknown author. *EDB-Home*. <https://enterprisedb.com/>.
- [23] unknown author. *Envoy proxy - home*. <https://www.envoyproxy.io/>.
- [24] unknown author. *etcd*. <https://etcd.io/>.
- [25] unknown author. *Features* - StackGres Documentation. <https://stackgres.io/doc/latest/features/>.
- [26] unknown author. *HAProxy Documentation Converter*. <https://docs.haproxy.org/>.
- [27] unknown author. *HAProxy version 2.9.6 - Starter Guide*. <https://docs.haproxy.org/2.9/intro.html#3.2>.
- [28] unknown author. *Helm*. <https://helm.sh/>.
- [29] unknown author. *Introduction | RKE2*. <https://docs.rke2.io/>. 2024.
- [30] unknown author. *Introduction to Cilium & Hubble* — Cilium 1.15.3 documentation. <https://docs.cilium.io/en/stable/overview/intro/#what-is-cilium>.
- [31] unknown author. *Keycloak*. <https://www.keycloak.org/>.
- [32] unknown author. *Manual Pages* — pg_auto_failover 2.0 documentation. <https://pg-auto-failover.readthedocs.io/en/main/ref/manual.html>.
- [33] 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>.
- [34] unknown author. *MetalLB, bare metal load-balancer for Kubernetes*. <https://metallb.universe.tf/>.
- [35] unknown author. *Multi-node Architectures* — pg_auto_failover 2.0 documentation. <https://pg-auto-failover.readthedocs.io/en/main/architecture-multi-standby.html>.
- [36] unknown author. *Percona Software for PostgreSQL*. <https://www.percona.com/postgresql/software>.
- [37] unknown author. *PgBouncer - lightweight connection pooler for PostgreSQL*. <https://www.pgbouncer.org/>.
- [38] unknown author. *Problem with 08P01: server conn crashed?* · Issue #714 · pgbouncer/pgbouncer. <https://github.com/pgbouncer/pgbouncer/issues/714>.
- [39] unknown author. *Red Hat Ansible Automation Platform automation controller*. <https://www.redhat.com/en/technologies/management/ansible/automation-controller>.
- [40] unknown author. *Replication in DocDB - Zone Fault Tolerance*. <https://docs.yugabyte.com/preview/architecture/docdb-replication/replication/>. Section: preview.
- [41] unknown author. *Support and Services for PostgreSQL*. <https://www.percona.com/postgresql/support-and-services>.

- [42] unknown author. *Tuning PostgreSQL with pgbench*. <https://www.cloudbees.com/blog/tuning-postgresql-with-pgbench>. 2017.
- [43] unknown author. *YB-TServer service*. <https://docs.yugabyte.com/preview/architecture/concepts/yb-tserver/>. Section: preview.
- [44] GitLab B.V. und GitLab Inc. *The DevSecOps Platform | GitLab*. <https://about.gitlab.com/>.
- [45] Fernando Laudares Camargos Avinash Vallarapu. *Tuning PostgreSQL for sysbench-tpcc*. <https://www.percona.com/blog/tuning-postgresql-for-sysbench-tpcc/>. 2018.
- [46] Alexandre Cassen und Read the Docs. *Introduction — Keepalived 1.2.15 documentation*. <https://keepalived.readthedocs.io/en/latest/introduction.html>. 2017.
- [47] Microsoft Corporation. *Azure SQL-Datenbank – ein verwalteter Clouddatenbankdienst | Microsoft Azure*. <https://azure.microsoft.com/de-de/products/azure-sql/database>. 2023.
- [48] Microsoft Corporation. *Datenbank-Software und Datenbankanwendungen | Microsoft Access*. <https://www.microsoft.com/de-de/microsoft-365/access>. 2023.
- [49] Microsoft Corporation. *Microsoft Data Platform | Microsoft*. <https://www.microsoft.com/de-ch/sql-server>.
- [50] Varun Dhawan und data-nerd.blog. *PostgreSQL-Diagnostic-Queries – data-nerd.blog*. <https://data-nerd.blog/2018/12/30/postgresql-diagnostic-queries/>.
- [51] Elektronik-Kompendium.de und Schnabel Schnabel. *SAN - Storage Area Network*. <https://www.elektronik-kompendium.de/sites/net/0906071.htm>. 2023.
- [52] DB-Engines und solidIT consulting & software development gmbh. *DB-Engines Ranking*. <https://db-engines.com/en/ranking>.
- [53] DB-Engines und solidIT consulting & software development gmbh. *relationale Datenbanken - DB-Engines Enzyklopädie*. <https://db-engines.com/de/article/relationale+Datenbanken?ref=RDBMS>.
- [54] The Linux Foundation. *Harbor*. <https://goharbor.io/>. 2023.
- [55] 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/spitaelespitalzukunft.html>.
- [56] Git. *About - Git*. <https://git-scm.com/about>.
- [57] IBM Deutschland GmbH. *Was ist das CAP-Theorem? | IBM*. <https://www.ibm.com/de-de/topics/cap-theorem>. 2023.
- [58] IBM Deutschland GmbH. *Was ist OLAP? | IBM*. <https://www.ibm.com/de-de/topics/olap>.
- [59] Jedox GmbH. *Was ist OLAP? Online Analytical Processing im Überblick*. <https://www.jedox.com/de/blog/was-ist-olap/>. Section: Knowledge.
- [60] 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>.
- [61] 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/InstitutionenGesundeitswesens/Spitaeler/Dok%20Spitler/Kenndaten%202016%20Spit%C3%A4ler.pdf.
- [62] The Pgpool Global Development Group. *What is Pgpool-II?* <https://www.pgpool.net/docs/44/en/html/intro-whatis.html>. 2023.
- [63] The PostgreSQL Global Development Group. *25.1. Routine Vacuuming*. <https://www.postgresql.org/docs/16/routine-vacuuming.html>. 2023.
- [64] The PostgreSQL Global Development Group. *pgbench*. <https://www.postgresql.org/docs/16/pgbench.html>. 2023.
- [65] 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.
- [66] Michael Haag. *Built-in Connection Manager Turns Key PostgreSQL Weakness into a Strength*. <https://www.yugabyte.com/blog/connection-pooling-management/>. 2023.
- [67] Inc. HashiCorp. *Terraform by HashiCorp*. <https://www.terraform.io/>.
- [68] Patrick Hunt, Mahadev Konar, Flavio P Junqueira und Benjamin Reed. „ZooKeeper: Wait-free coordination for Internet-scale systems“. In: (2010).
- [69] Splunk Inc. *Splunk / Der Schlüssel zu einem resilienten Unternehmen*. https://www.splunk.com/de_de. 2023.
- [70] Sebastian Insausti. *Scaling PostgreSQL for Large Amounts of Data*. <https://severalnines.com/blog/scaling-postgresql-large-amounts-data/>. 2019.
- [71] Shiv Iyer und MinervaDB. *PostgreSQL DBA Daily Checklist*. <https://minervadb.xyz/postgresql-dba-daily-checklist>. 2020.
- [72] jobinau/pg_gather. https://github.com/jobinau/pg_gather. original-date: 2021-01-19T08:12:07Z. 2024.
- [73] Unmesh Joshi. *Quorum*. <https://martinfowler.com/articles/patterns-of-distributed-systems/quorum.html>. 2020.
- [74] Martin Keen und IBM Deutschland GmbH. *IBM Db2*. <https://www.ibm.com/de-de/products/db2>.
- [75] Pasha Kostohrys. *Database replication — an overview*. <https://medium.com/@pkostohrys/database-replication-101>. 2020.
- [76] Anatoli Kreyman. *Was ist eigentlich Splunk?* <https://www.kreyman.de/index.php/splunk/76-was-ist-eigentlich-splunk>.
- [77] 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>.
- [78] Red Hat Limited. *Was ist Ansible?* <https://www.redhat.com/de/technologies/management/ansible/what-is-ansible>.
- [79] 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>.
- [80] Switzerland Linuxfabrik GmbH Zurich. *Keepalived — Open Source Admin-Handbuch der Linuxfabrik*. <https://docs.linuxfabrik.ch/software/keepalived.html>. 2023.

- [81] Nico Litzel, Stefan Luber und Vogel IT-Medien GmbH. *Was ist Elasticsearch?* <https://www.bigdata-insider.de/was-ist-elasticsearch-a-939625/>. 2020.
- [82] Hewlett Packard Enterprise Development LP. *Was ist SAN-Speicher? | Glossar*. <https://www.hpe.com/ch/de/what-is/san-storage.html>.
- [83] Julian Markwort. „Benchmarking four Different Replication Solutions“. In: ().
- [84] Sujoy Nath. *Database Connection Pool*. <https://medium.com/@sujoy.swe/database-connection-pool-647>. 2023.
- [85] Diego Ongaro. „Consensus: Bridging Theory and Practice“. In: (2014).
- [86] Bruno Queirós und LinkedIn Ireland Unlimited Company. *Postgresql replication with automatic failover*. <https://www.linkedin.com/pulse/postgresql-replication-automatic-failover-bruno-queir%C3%B3s>. 2020.
- [87] Karthik Ranganathan. *Evolving Clock Sync in Distributed Databases | YugabyteDB*. <https://www.yugabyte.com/blog/evolving-clock-sync-for-distributed-databases/>. 2022.
- [88] 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>.
- [89] Ed Reckers und SnapLogic Inc. *Was ist die Snowflake-Datenplattform?* <https://www.snaplogic.com/de/blog/snowflake-data-platform>. 2023.
- [90] IONOS SE. *Apache Cassandra: Verteilte Verwaltung großer Datenbanken*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/apache-cassandra-vorgestellt/>. 2021.
- [91] IONOS SE. *Datenbankmanagementsystem (DBMS) erklärt*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/>. 2020.
- [92] IONOS SE. *MongoDB – die flexible und skalierbare NoSQL-Datenbank*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysql/>. 2019.
- [93] IONOS SE. *SQLite: Die bekannte Programmiersbibliothek im Detail vorgestellt*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/sqlite/>. 2023.
- [94] IONOS SE. *Terraform*. <https://www.ionos.de/digitalguide/server/tools/was-ist-terraform/>. 2020.
- [95] IONOS SE. *Was ist Redis? Die Datenbank vorgestellt*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-redis/>. 2020.
- [96] IONOS SE. *Was ist SIEM (Security Information and Event Management)?* <https://www.ionos.de/digitalguide/server/sicherheit/was-ist-siem/>. 2020.
- [97] Naveed Shaikh. *It's All About Replication Lag in PostgreSQL*. <https://www.percona.com/blogreplication-lag-in-postgresql/>. 2023.
- [98] Sami Ahmed Siddiqui. *Distributed SQL 101*. <https://www.yugabyte.com/distributed-sql/>.
- [99] Inc. Snowflake. *Datenbanken, Tabellen und Ansichten – Überblick | Snowflake Documentation*. <https://docs.snowflake.com/de/guides-overview-db>.
- [100] Thomas-Krenn.AG. *Git Grundlagen – Thomas-Krenn-Wiki*. https://www.thomas-krenn.com/de/wiki/Git_Grundlagen.

- [101] Mahmut Can Uçanefe. *Pgbench Load Test*. <https://medium.com/@c.ucanefe/pgbench-load-test-166bdfb> 2023.
- [102] vitabaks/postgresql_cluster. https://github.com/vitabaks/postgresql_cluster. original-date: 2019-06-04T13:26:17Z. 2024.
- [103] 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	Private 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 auf die Zielgeräte und führt dort die Hinterlegten Module aus. Oft werden die verschieden Aufgaben in einem Skript, in einem sogenannten Playbook geschrieben werden[78].. 17, 104, 116, 124, 126, 131, 156, cxvii

AUTOVACUUM Der AUTOVACUUM Job räumt die Tablespaces und Data Files innerhalb von PostgreSQL sowie auf dem Filesystem nach Lösch- und Manipulations-Transaktionen auf, aktualisiert Datenbank interne Statistiken und verhindert Datenverlust von selten genutzten Datensätzen[63].. 15, 16, 47, lxxx

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[90] wurde.. 7, 69, 75

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[79].. 150, 156

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[10, 30].. 88

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 Seiten DB reduziert[84].. 114, 115, 116, 117

Consul Consul ist ein Protokoll, welches zur Konsensfindung dient[19, 18].. 117

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[91].. 150

DCS Der DSC ist eine Kernkomponente von Patroni [21]. Realisiert wird der DCS bei Patroni mit Etcd.. 114, 117, 150

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.. 17, 119

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[81].. 7

etcd etcd ist [24]. 61, 77, 78, 114, 117, 118, 152, ii, lxviii

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 16, 37, 38, 53, 116, 153

Foreman Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 17, 22, 116, 119, lxvii

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[56, 100].. 153

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].. 104, 111, 116, 119, 131, ii, cxviii

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[44].. 16, 53, 117

HAProxy HAProxy [27]. 55, 60, 77, 116, 117, 118, 120, 121, 126, 128

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[54]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 16, 53, 117

helm helm bietet mit seinen helm charts Paketressourcen, die das deployment von Kubernetes-Ressourcen erleichtert[28].. 82, xcix

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, 9, 22, 150

IBM DB2 IBM DB2 ist eine Relationale Datenbank[74] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 39

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[80, 46].. 53

Key-Value-orientierte Siehe Key-Value-Datenbank. 156

Key-Value-Datenbank Eine Key-Value-Datenbank ist ein Typ derNoSQL 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. 153, 154, 155

Keycloak Keycloak ist ein Identity und Access Management Tool[31], welches vom KSGR für SSO resp. Authentifizierung und das Management von Secrets für Kubernetes, und Applikationen die in Kubernetes betrieben werden, verwendet wird.. 117

Key-Value-Store Siehe Key-Value-Datenbank. 51, 69

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.. 9, 17, 95, 113, 131, 150, 153

Linux Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und löse vom UNIX Derivat MINIX inspiert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 154

local-path-provisioner local-path-provisioner ist ein leichtgewichtiger Storage-Provider von Rancher. Er bietet den Applikationen einen persistenten Storage an.. 91, 93, 131

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.. 5, 7, 9, 10, 52

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 dass Pods direkt von Routern angesteuert werden können, ohne via Host gehen zu müssen[34].. 84, 88, 130

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[47].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[48].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[49]. 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, 155

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zum ersten Mal 2007 veröffentlicht wurde[92].. 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 Microsystem 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.. 5, 7, 9, 52

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, 152, 153, 154, 156

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[59, 58]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 7, 150

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.. 17

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 erschienenn 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 Perforamce oder grosse Datenmengen geht.. 5, 7, 9, 39, 155

PKI . 113, 150

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.. 5, 7, 9, 10, 14, 39, 53, 69

PostgreSQL HA Cluster Der HA Cluster des PostgreSQL Clusters. 16

PostgreSQL Cluster Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wei Oracle.. 15, 16, 53, 155, Ixviii, Ixxv, Ixxx

PRTG Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum erstmals 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.. 5, 15, 17, 129, 130, 134, 135, 150

Quorum In verteilten Systemen resp. Cluster muss sichergestellt werden, das bei einem Ausfall oder ein Netzwerktrennung 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 die 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 [Unterunterabschnitt 3.1.1.4](#) wird genauer auf die Mechanik eines Quorums eingegangen. . 54, 153

RDBMS Ein RDBMS ist ein Datenbankmanagementsystem für eine Relationale Datenbank. Relationale Datenbanken sind Tabellenorganisierte Datenmodelle die auf Relationen aufbauen, deren Schemata sich Normalisieren lassen. Dabei müssen Relationale Datenbanken dabei auch Mengenoperationen, Selektion, Projektion und Joins erfüllen um als Relationale Datenbanken zu gelten[53].. 69, 150

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[39].. 116

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.. 17

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[95]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 153

rke2 rke2 ist eine Leichtgewichtige Kubernetes Distribution, die alles notwendige mitbringt, um einen k8s Cluster zu betreiben[29].. 76, 83, 88, 96, 130, 131

Rocky Linux Rocky Linux basierte auf der offen zugänglichen Linux Distribution CentOS welche RHEL Binärkompatibel war und gilt als inoffizieller Nachfolger von CentOS.. 17

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[51, 82, 60]. 5, 17, 22, 99, 150

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[96].. 17, 113, 125, 150

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[99, 89]. 7

Split-brain Im Kapitel ?? werden die Ursachen und Folgenden eines Split-brains genauer besprochen. . 38, 155

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[69, 76]. . 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[93].. 7

Switchover In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geswitched.. 16, 116

SWOT-Analyse Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzueigen. Anhand einer SWOT-Analyse werden i.d.R. anschliessend Strategien abgeleitet um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder anzumildern.. 150

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[94, 67].. 17

Transaktion Eine Transaktion ist beinhaltet Schreib-, Lese-, Mutations- oder Löschoperationen auf Daten.. 48, 50

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.. 150

VMware vSphere Die vSphere® ist ein Typ-1 Hypervisor der Firma VMware® der eine Reihe leistungsfester Tools und Funktionen mitbringt.. ii, xxxix, lxxvii

VRRP VRRP . 150, 153

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 heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert werden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge abschliessen.. 10, 17

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.

|

Arbeitsrapport

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	Aufbau analog ursprüngliche Evaluationsplattform von Patroni.	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	Architektur entsprechend zeichnen.	etcd-Server bereitet Probleme	Vereinfachen soweit möglich
19.04.2024	20:00	21:00	1.0	Aufbau und Implementation Testsystem	Basisinfrastruktur	Patroni Test Server DMT / Auftrag	DMT Einträge für Patroni Testserver erstellt.		
20.04.2024	10:00	12:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking	Ticket an Niculin Fürst für Foreman Job.		
20.04.2024	14:00	16:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking / grosse Volumes	Auf erweiterte Disk umstellen und letzten Benchmark fahren		
22.04.2024	09:00	09:15	0.2	Evaluation	Gegenüberstellung				
22.04.2024	09:30	11:00	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
23.04.2024	13:45	14:00	0.2	Evaluation	Gegenüberstellung				
24.04.2024	15:00	17:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern			
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	Prüfeneries umgesetzt.		
29.04.2024	07:00	09:00	2.0	Aufbau und Implementation Testsystem	Basisinfrastruktur	Anpassungen / Passwörter	Snapshots auf VMware vSphere gesetzt.		

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 D D	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</p> <p>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)

III.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 veratl. 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)	LO-001 Anforderungskatalog LO-002 Vorbereiten Benchmarking LO-003 LO-004 LO-005	Status	Erliebigungsgrad
		in arbeit	<div style="width: 60%;">60%</div>
			<div style="width: 0%;">0%</div>
# Risiken		Auswirkungsgrad	Massnahmen
R-001	Fehlende Ressourcen	orange	Organisation und Selbstmanagement
R-002	HP-UX Ablöseprojekt	rot	Ressourcen reservieren
R-003	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	rot	Monitoring vorgängig ausbauen und Massnahmen definieren
R-004	Schwächen beim Selbstmanagement und in der Selbstorganisation	orange	Werkzeuge im Vorfeld definieren und bereitstellen
Kostenübersicht		Abhängigkeiten zu anderen Projekten	
Verfügbare Finanzen bis Ende Projekt: $\frac{100 \text{ CHF}}{\text{A}} + 2000 = 24.000 \text{ CHF}$		Massnahmen	
		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)			

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 13.02.2024	
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	PMA
ICT veratl. 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	
LO-001 Anforderungskatalog		Erliebigungsgrad	
LO-002 Vorbereiten Benchmarking		Soll Datum	
LO-003		60% 23.02.2024	
LO-004		0% 26.02.2024	
LO-005			
# Risiken		Auswirkungsgrad	
R-001 Fehlende Ressourcen		Massnahmen	
R-002 HP-UX Ablöseprojekt		Organisation und Selbstmanagement	
R-003 Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden		Ressourcen reservieren	
R-004 Schwächen beim Selbstmanagement und in der Selbstorganisation		Monitoring vorgängig ausbauen und Massnahmen definieren	
Werkzeuge im Vorfeld definieren und bereitstellen			
Kostenübersicht		Abhängigkeiten zu anderen Projekten	
Verfügbare Finanzen bis Ende Projekt: 100 CHF + 2000 = 24.000 CHF		Massnahmen	
Anträge		Ressourcen reservieren	
Bemerkungen / Informationen		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Eingereicht		Bemerkungen/Auftrag PMO	
PL:	Geprüft		
Datum:	PMO:		
# erledigte Lieferobjekte (inkl. allfällige Links)			

III.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	-	
			
Gesamtprojekt	Ampel	Tendenz	Begründung
Zeitplanung			In verzögert. 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.
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	
# Risiken	Auswirkungsgrad	Massnahmen	Verantw.
		Organisation und Selbstmanagement	
R-001		Ressourcen reservieren	
R-002		Monitoring vorgängig ausbauen und Massnahmen definieren	
R-003		Werkzeuge im Vorfeld definieren und bereitstellen	
R-004		Ziele klar definieren	
R-005		Ziele SMART definieren	
R-006		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
R-007			
R-008			
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	
Bemerkungen / Informationen		Massnahmen	
		Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Eingereicht	Geprüft	Bemerkungen/Auftrag PMO	
PL:	PMO:		
Datum:	Datum:		
# erledigte Lieferobjekte (inkl. allfällige Links)			
LO-001 Anforderungskatalog			

Table 1: Zweiter Statusbericht

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 18.03.2024	
Projektbeschreibung Evaluation und Implementation PostgreSQL HA Cluster		Priorität	PMA -
ICT verantwort. Person Michael Graber		Heute	
			
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 Spur Datum
LO-002 Vorbereiten Benchmarking		offen	
LO-003 Analyse PostgreSQL HA Cluster Lösungen		in Arbeit	
LO-004 Gegenüberstellung		offen	
LO-005 Variantenentscheid		offen	
LO-006 Aufbau Basisinfrastruktur Testsystem		offen	
# Risiken		Auswirkungsgrad	Massnahmen Verantw.
R-001 Fehlende Ressourcen		orange	Organisation und Selbstmanagement
R-002 HP-UX Ablöseprojekt		rot	Ressourcen reservieren
R-003 Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden		rot	Monitoring vorgängig ausbauen und Massnahmen definieren
R-004 Schwächen beim Selbstmanagement und in der Selbstorganisation		orange	Werkzeuge im Vorfeld definieren und bereitstellen
R-005 Scope verlust während des Projekts		orange	Ziele klar definieren
R-006 Scope Creep		orange	Ziele SMART definieren
R-007 SIEM / Log Plattform nicht betriebsbereit		orange	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten
R-008 Foreman nicht betriebsbereit		orange	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 Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) => Foreman Umgebung	
Bemerkungen / Informationen		Massnahmen Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Eingereicht	Geprüft	Bemerkungen/Auftrag PMO	
PL:	PMO:		
Datum:	Datum:		
# erledigte Lieferobjekte (inkl. allfällige Links)			
LO-001 Anforderungskatalog			

Table 1: Zweiter Statusbericht

III.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		<ul style="list-style-type: none"> - 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	
LO-007	Installation Patroni PostgreSQL Cluster	offen	
LO-008	Technical Review Testsystem	offen	
LO-009	Testing	offen	
LO-010	Resultate	offen	
# 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	
R-005		Ziele klar definieren	
R-006		Ziele SMART definieren	
R-007			
R-008		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	
Kostenübersicht			
Verfügbare Finanzen bis Ende Projekt: $100 \text{ CHF h}^{-1} \cdot 200 = 21\,000 \text{ CHF}$		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		

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht 29.04.2024	
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	-
ICT vertrw. Person	Michael Gruber	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		<ul style="list-style-type: none"> - 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	
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: $100 \text{ CHF h}^{-1} \cdot 200 = 21\,000 \text{ CHF}$		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		

IV

Kommentare / Anmerkungen

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 Evaluationsinfrastruktur</p> <ul style="list-style-type: none"> - Das Problem mit dem «private container registry» rührte daher,
KW13	<p>dass das YugabyteDB Anywhere (Repository <code>yugaware</code>) verwendet wurde.</p> <p>Kurz ein Schock, dass YugabyteDB ausgeschieden ist.</p>
KW13	<p>Später bemerkte ich, dass man das Repo <code>yugabytedb</code> auswählen muss.</p>
KW13	<ul style="list-style-type: none"> - MetalLB benötigt zwingend L2Advertisement, <p>damit Linux die Kommunikation von aussen nach innen leiten kann.</p> <ul style="list-style-type: none"> - Bereits jetzt viel über Kubernetes, Ranger (Ike2) und Helm gelernt.

V

Evaluation

V.I

Maintenance - CloudNativePG

Das Projekt hat eine relativ hohe Anzahl an aktiven Issues, wobei viele neue dazugekommen sinnen:

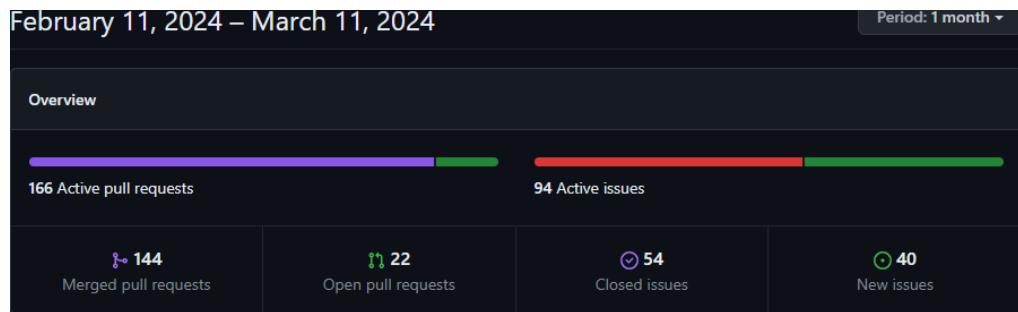


Abbildung I: CloudNativePG - Pulse

Der Code ist aber recht gut gepflegt, Code wird nicht nur regelmässig hinzugefügt, sondern auch entfernt. Auffällig ist, das im April 2022 eine grosse Menge Code entfernt wurde:

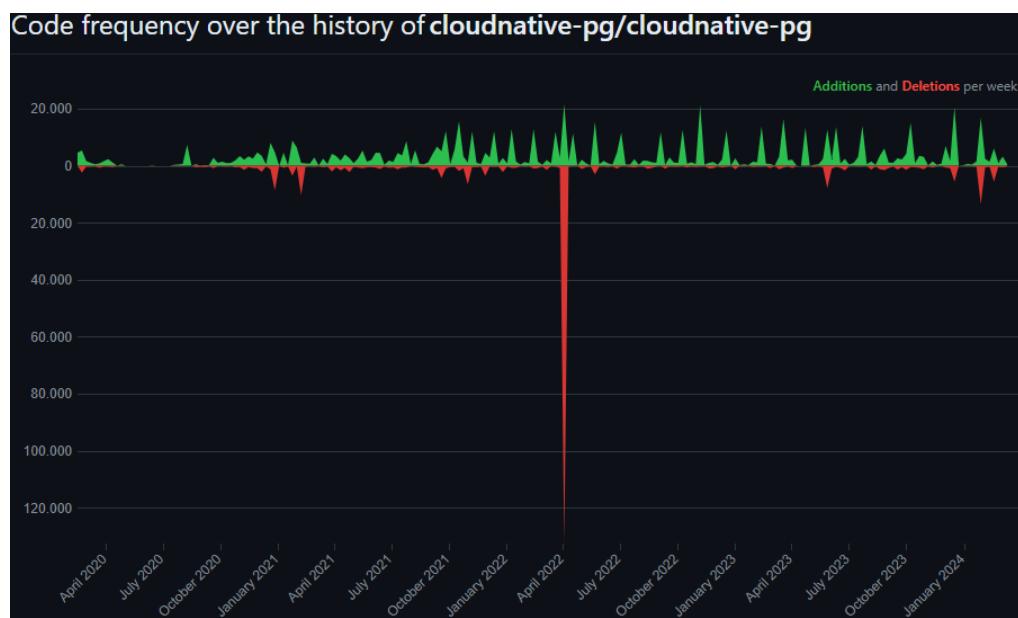


Abbildung II: CloudNativePG - Code Frequency

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 dann zu grösseren Aufräumarbeiten kommt wie im April 2022.

Es kann der Eindruck gewonnen werden, dass der Code wenig aufgeräumt wird und viel Balast mit sich schleppt,

was ein Sicherheitsrisiko darstellen kann:

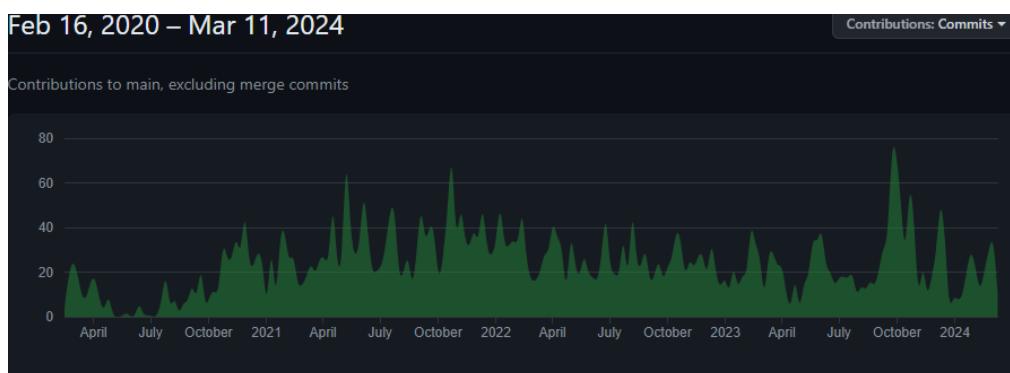


Abbildung IV: CloudNativePG - Contributors Commits



Abbildung V: CloudNativePG - Contributors Deletations

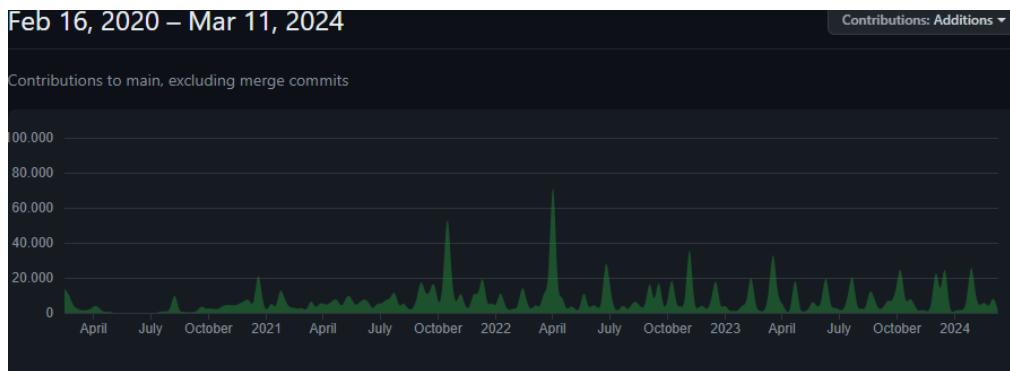


Abbildung VI: CloudNativePG - Contributors Additions

Commits werden regelmässig abgesetzt, allerdings gibt es immer wieder gehäufte Commits.
Oft um die Monatswechsel herum:

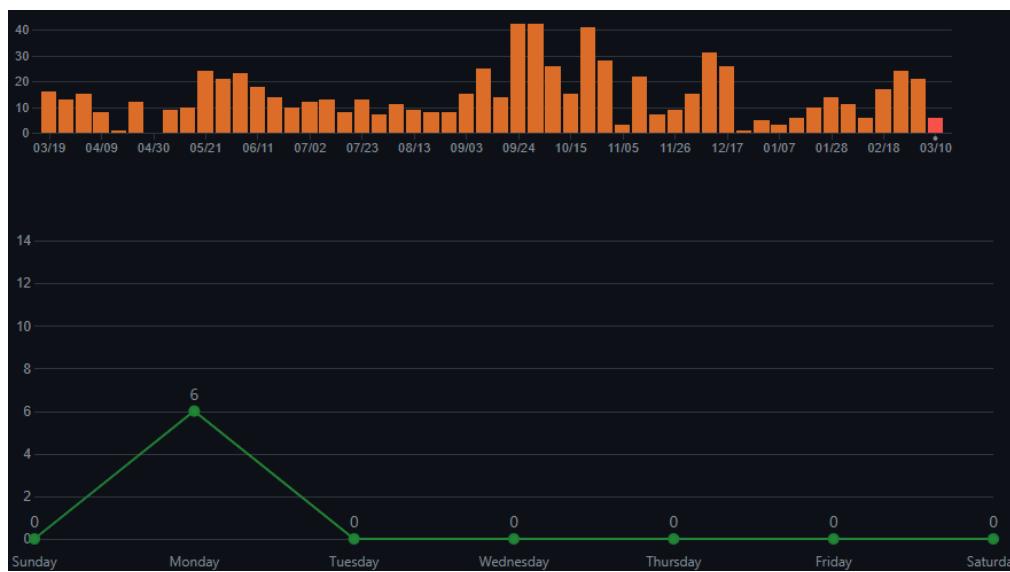


Abbildung VII: CloudNativePG - Commit Activity

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.

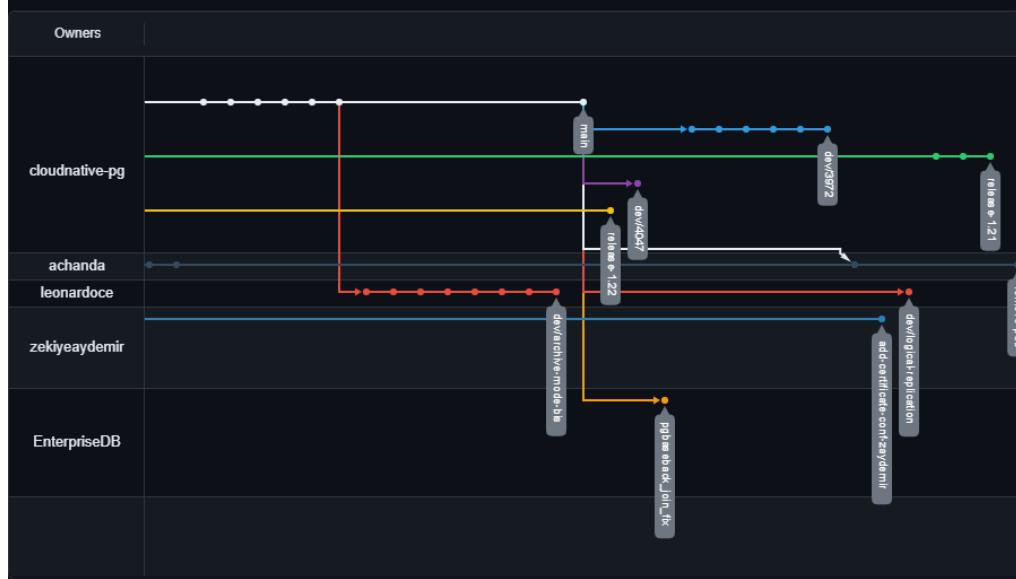


Abbildung VIII: CloudNativePG - Network Graph

V.II Maintenance - Patroni

Patroni wird von Zalando regelmäßig gepflegt. Das Projekt hat eine überschaubare Anzahl an Issues, wird aber Regelmässig

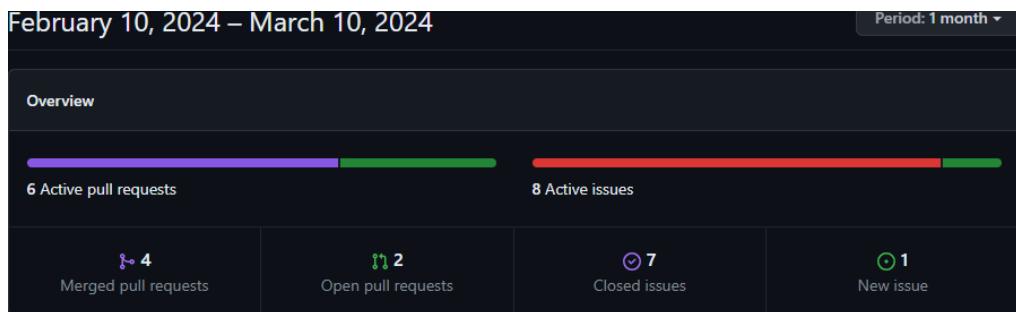


Abbildung IX: Patroni - Pulse

Code wird Regelmässig hinzugefügt und entfernt:

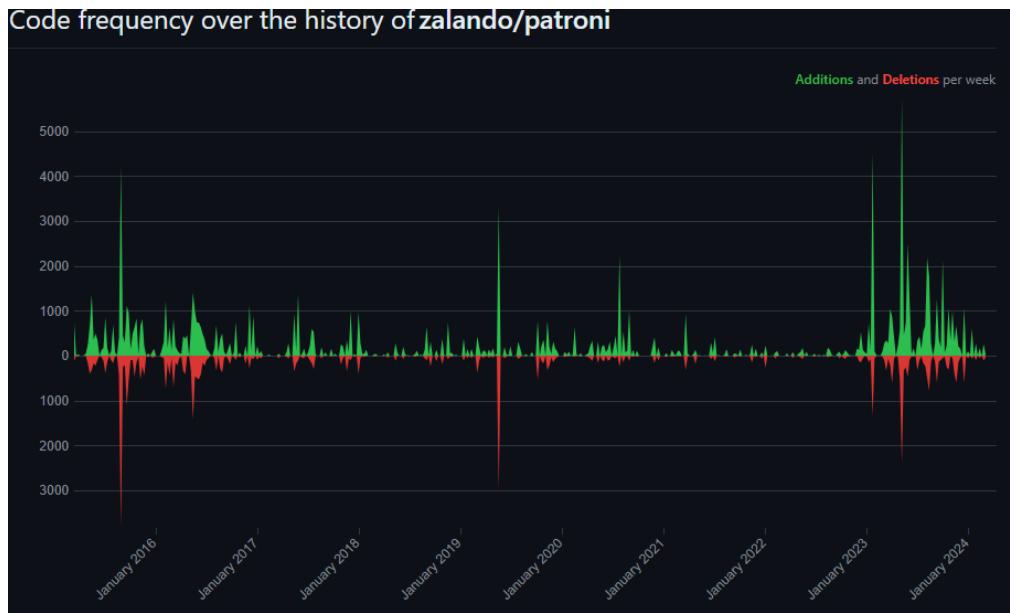


Abbildung X: Patroni - Code Frequency

Das Projekt hält auch die gängigen Standards auf Github ein:

Abbildung XI: Patroni - Community Standards

Die Contributors commiten, löschen und erweitern Patroni Regelmässig:

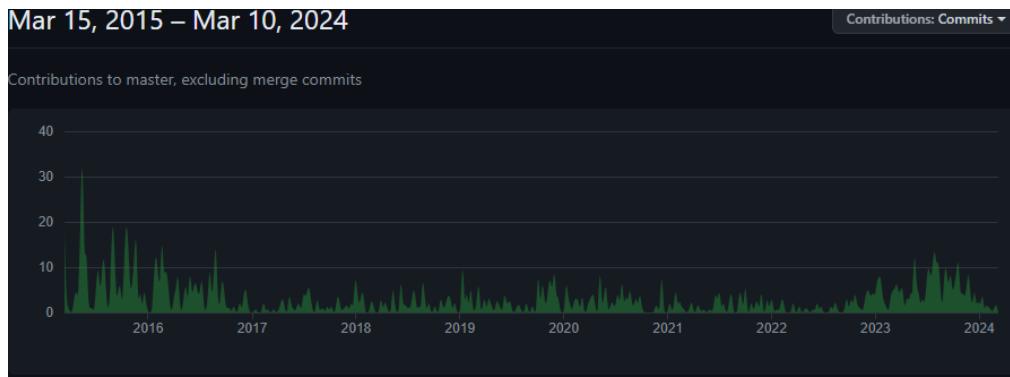


Abbildung XII: Patroni - Contributors Commits



Abbildung XIII: Patroni - Contributors Deletations

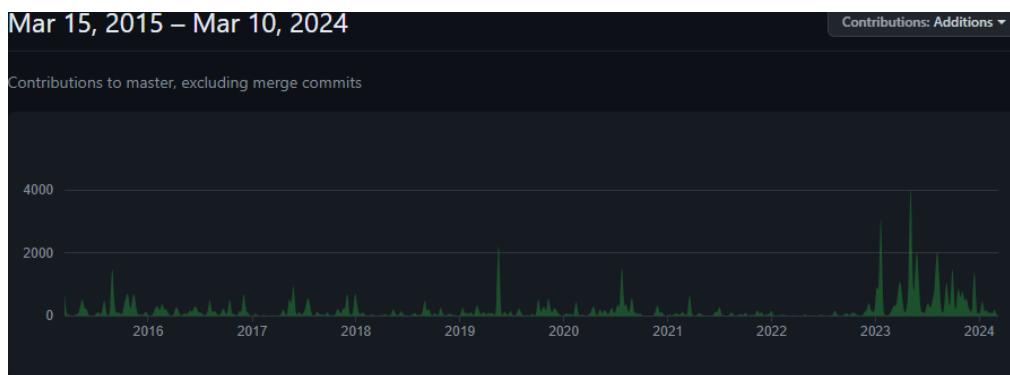


Abbildung XIV: Patroni - Contributors Additions

Commits werden nach wie vor immer noch Regelmässig eingespielt, auch wenn die Frequenz etwas nachgelassen hat:

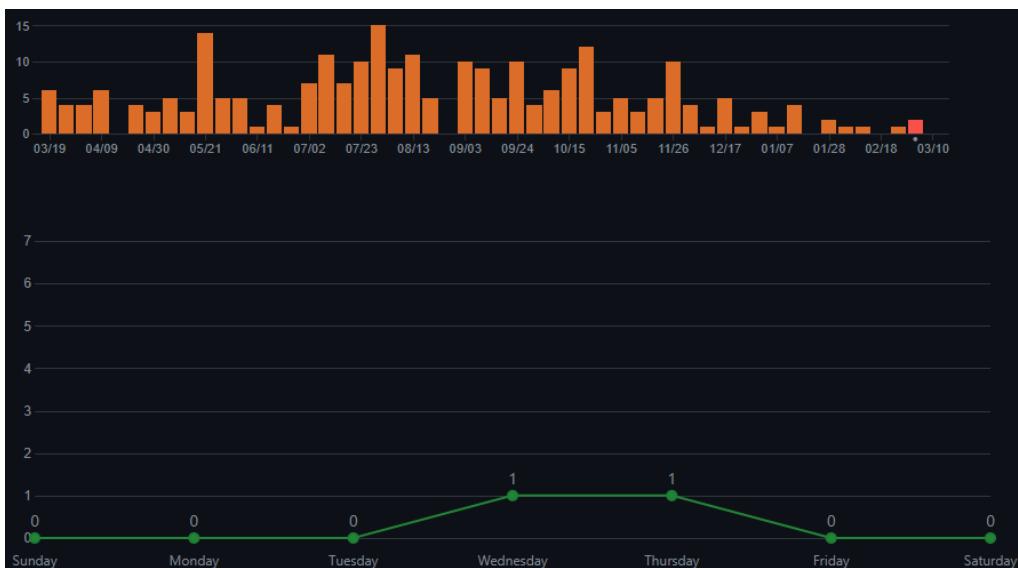


Abbildung XV: Patroni - Commit Activity

Nebst Zalando selbst hat auch EnterpriseDB [[LNF967SI](#)] ein grösseres Repository eingebunden. Dies weil EnterpriseDB stark auf Patroni setzt.

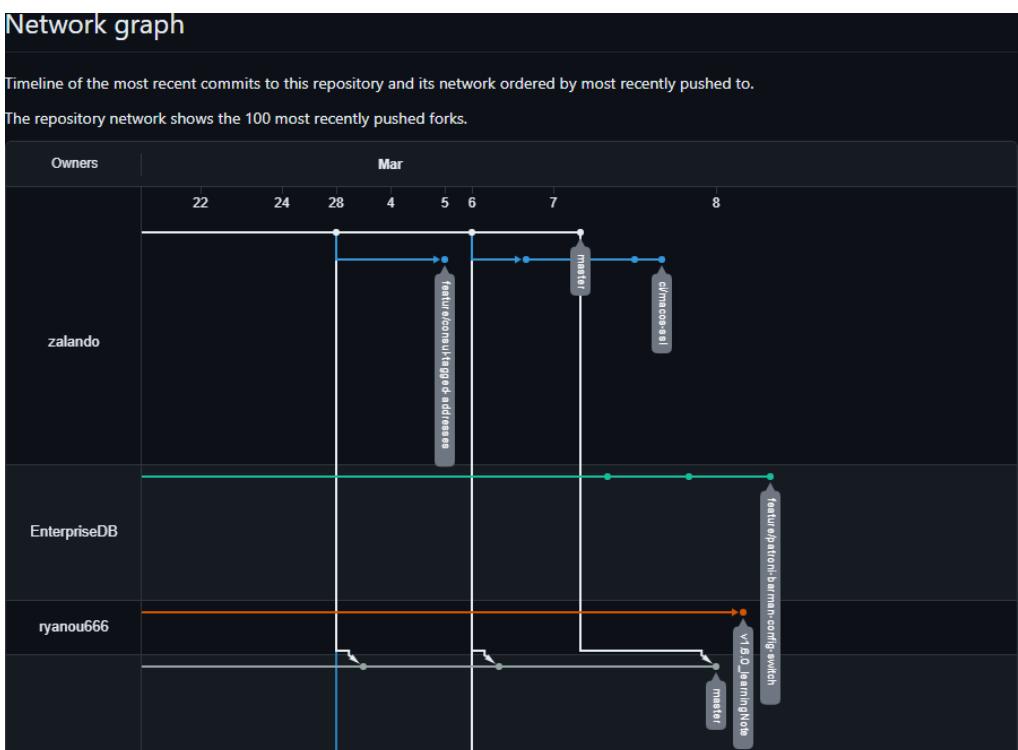


Abbildung XVI: Patroni - Network Graph

V.III Maintenance - StackGres - Citus

Bei StackGres gab es im letzten Monat keine wirkliche Bewegung:

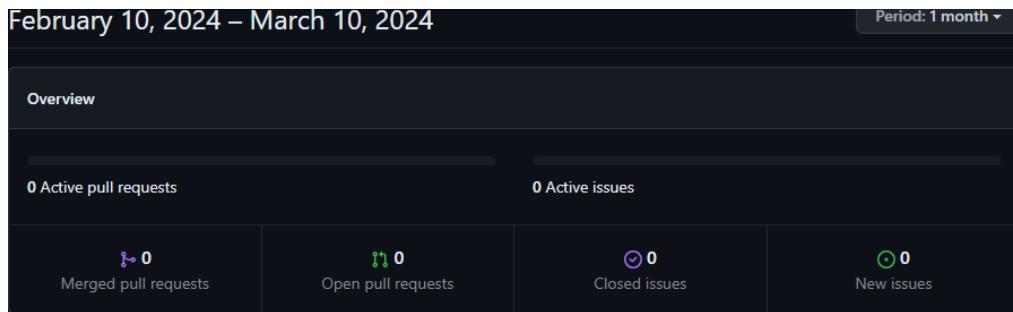


Abbildung XVII: Stackgres - Pulse

Anders sieht es bei Citus aus, die Firma die mittlerweile zu Microsoft gehört, schliesst Issues rasch und hat eine verhältnismässig hohe Requistrate:

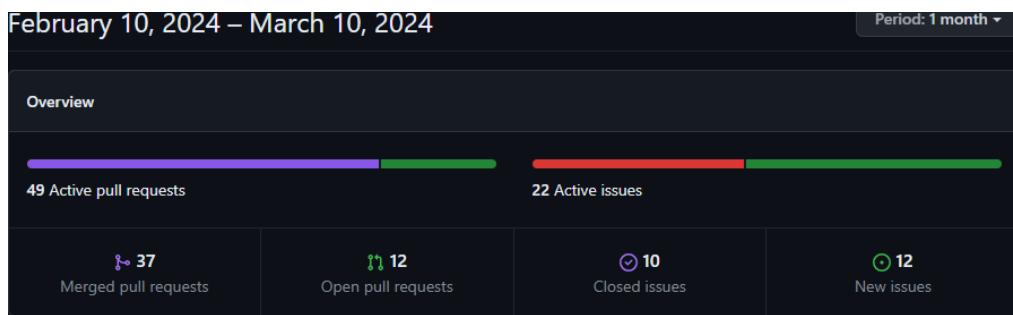


Abbildung XVIII: Citus - Pulse

Bei Stackgres wird sehr viel Code hinzugefügt oder gelöscht, beim älteren Citus wurden weniger änderungen verzeichnet:

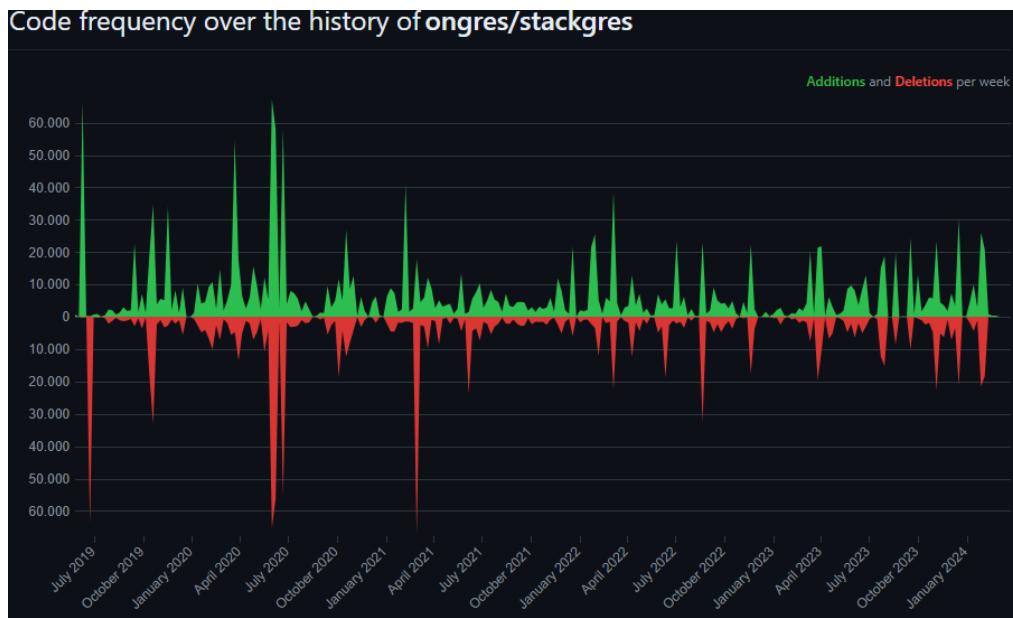


Abbildung XIX: Stackgres - Code Frequency

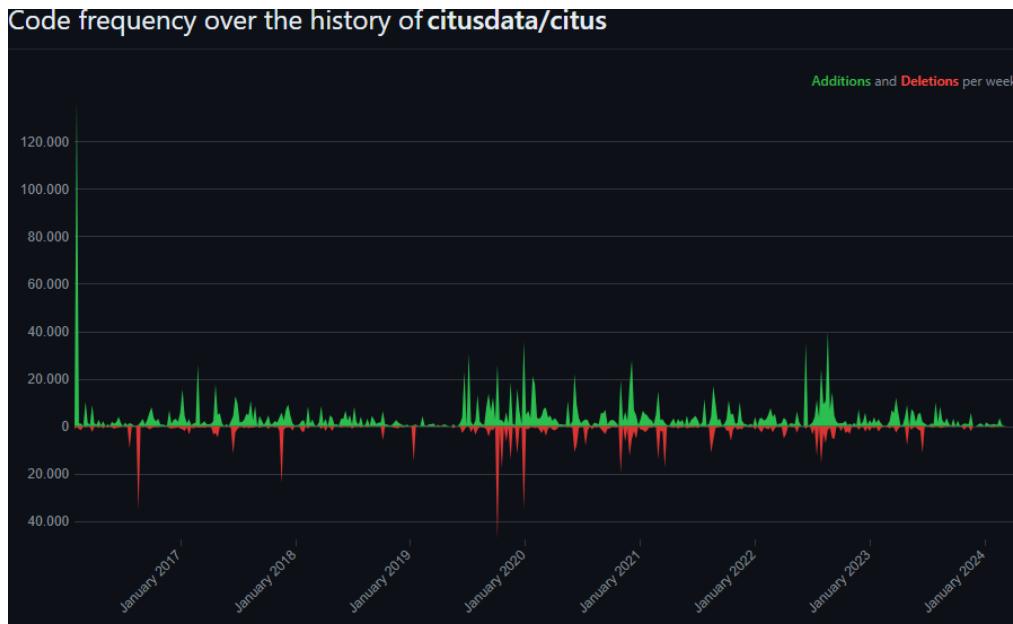


Abbildung XX: Citus - Code Frequency

Citus legt einen hohen Stellenwert auf die Community-Standards, Stackgres selbst schneidet hier nur Mittelmässig ab:

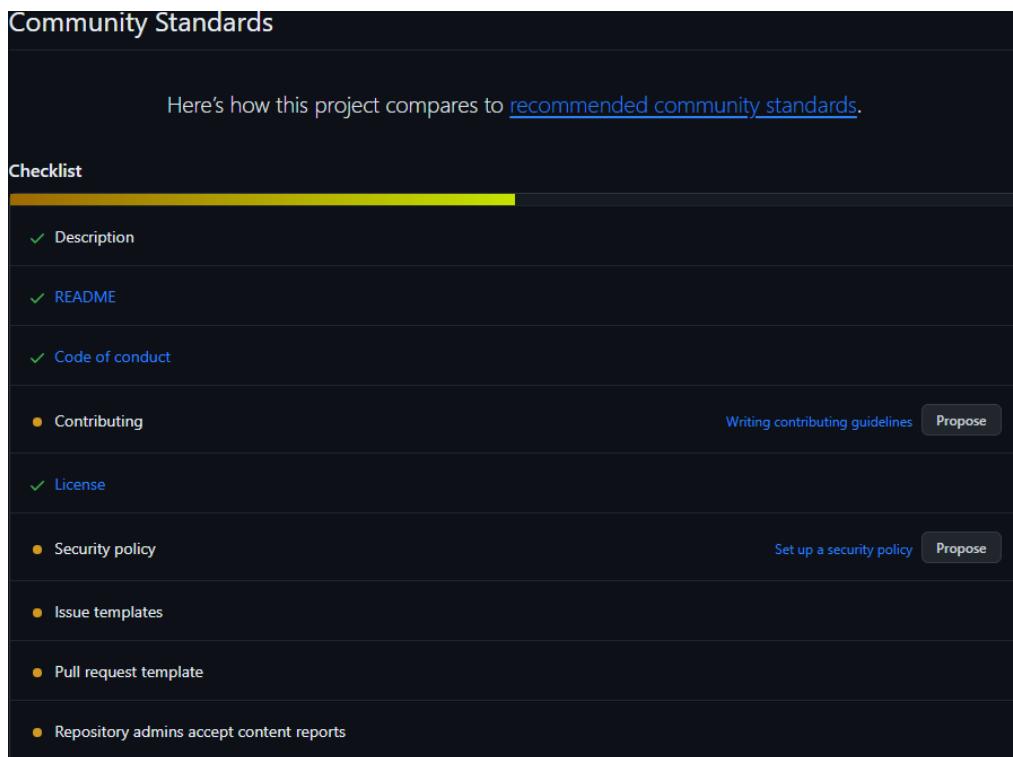


Abbildung XXI: Stackgres - Community Standards

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.

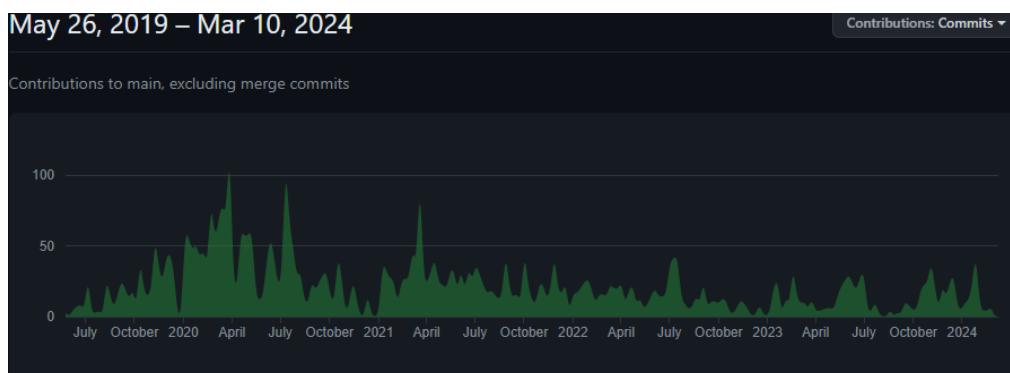


Abbildung XXIII: Stackgres - Contributors Commits

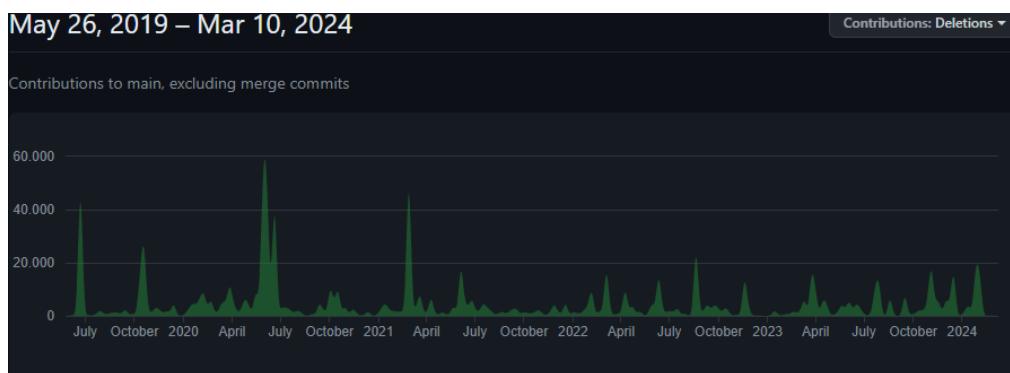


Abbildung XXIV: Stackgres - Contributors Deletations

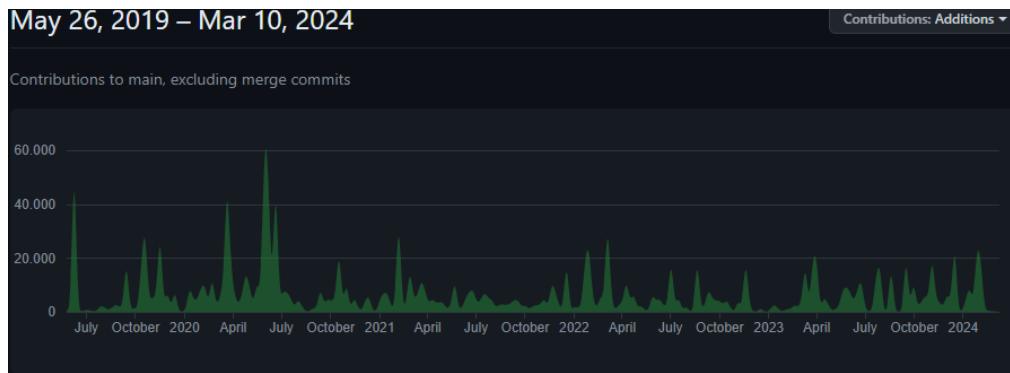


Abbildung XXV: Stackgres - Contributors Additions

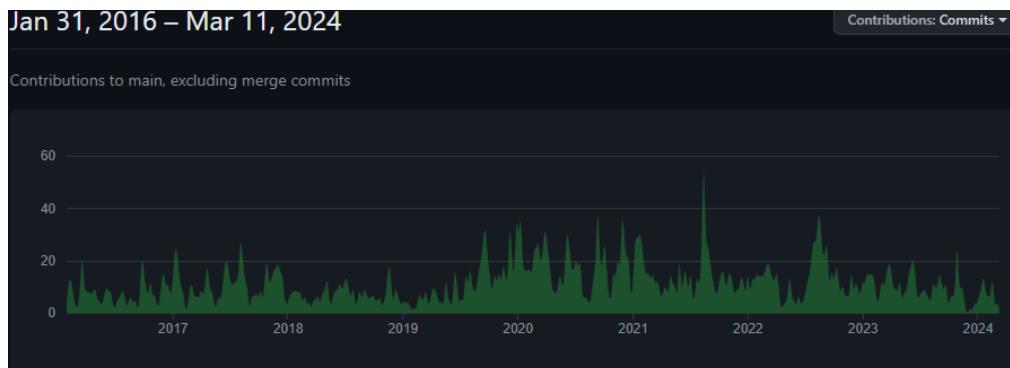


Abbildung XXVI: Citus - Contributors Commits

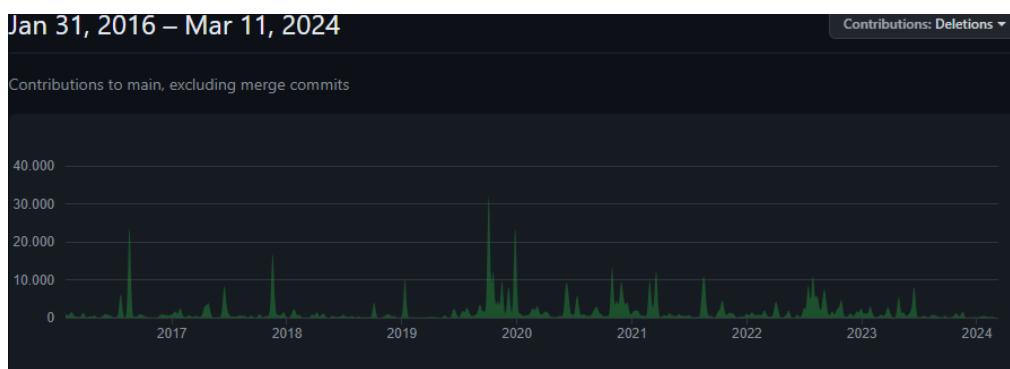


Abbildung XXVII: Citus - Contributors Deletations

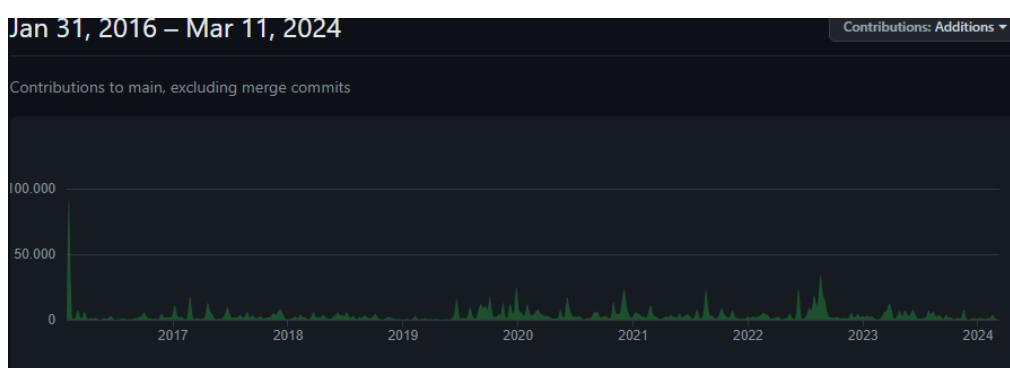


Abbildung XXVIII: Citus - Contributors Additions

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:

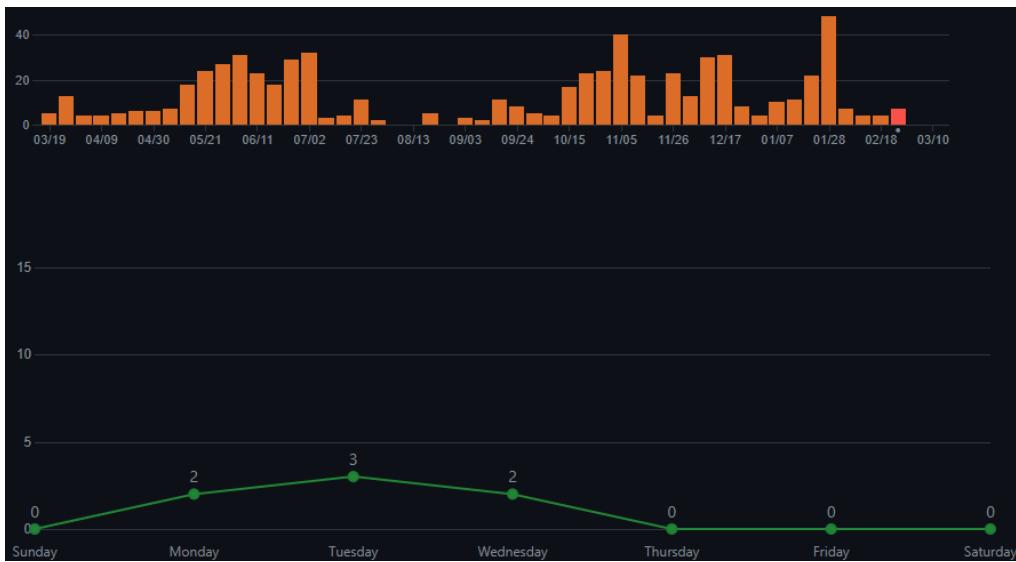


Abbildung XXIX: Stackgres - Commit Activity



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.

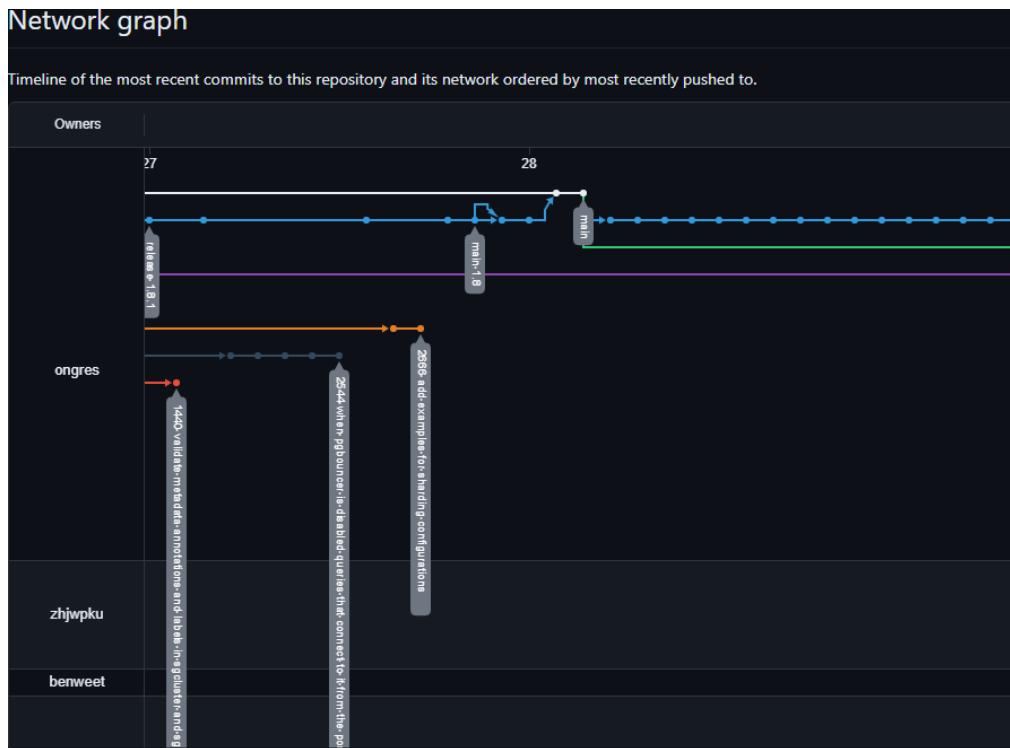


Abbildung XXXI: Stackgres - Network Graph

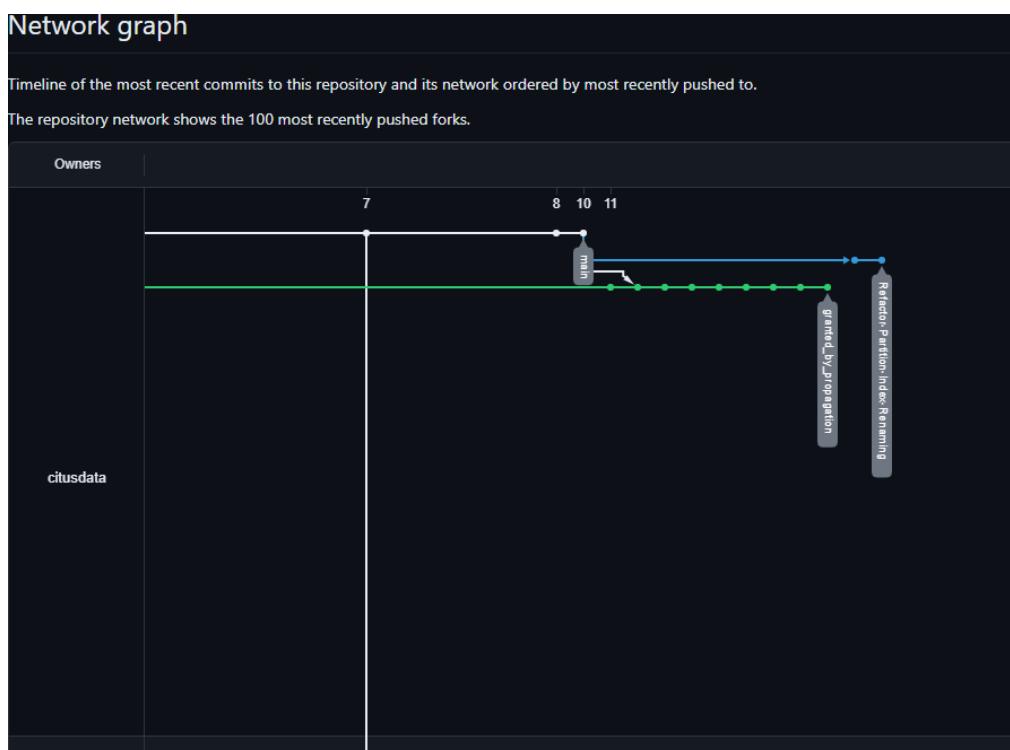


Abbildung XXXII: Citus - Network Graph

V.IV

Maintenance - YugabyteDB

Das Projekt hat eine sehr hohe Anzahl an aktiven Issues, wobei viele neue dazugekommen sinnen:

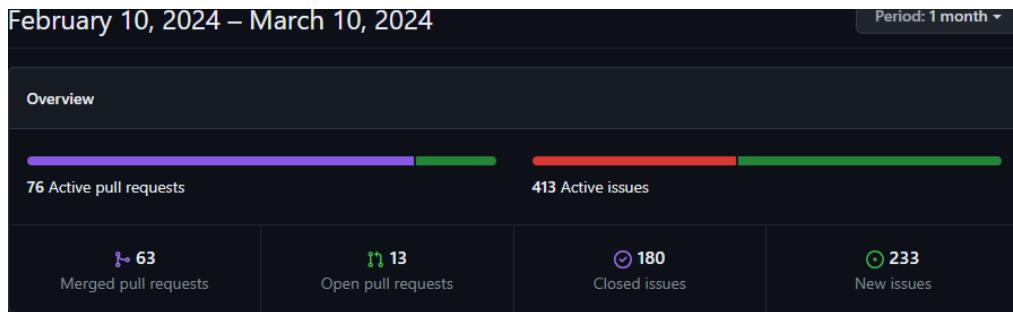


Abbildung XXXIII: YugabyteDB - Pulse

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



✓ 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:

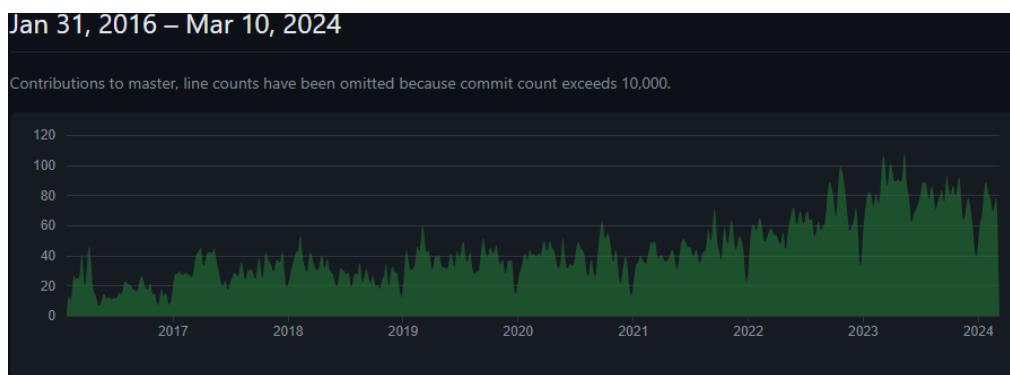


Abbildung XXXVI: YugabyteDB - Contributors

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:

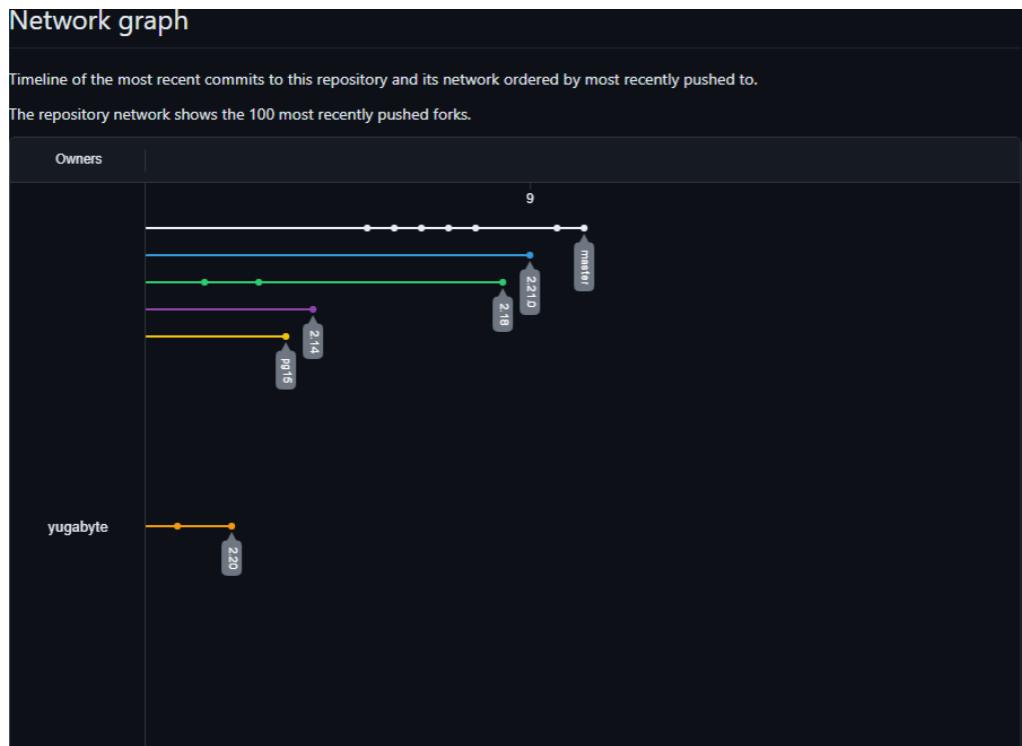


Abbildung XXXVIII: YugabyteDB - Network Graph

VI Evaluationssysteme - Installation

VI.I rke2

VI.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

VI.I.II Installation

VI.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

VI.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 restartet werden

```
1 systemctl start rke2-agent.service
```

Listing 8: -rke2 agent service restart

VI.I.III.III Cluster Konfiguration

VI.I.III.II server

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

Dieses File muss entsprechend in das Homeverzeichnis gespeichert werden:

Listing 10: rke2 server proxy kopieren

Für den Netzwerkeil muss nun Cilium installiert werden:

Listing 11: rke2 server cilium installieren

Cilium muss nun aktiviert werden:

Listing 12: rke2 server cilium aktivieren

Der rke2-Server muss nun mit der entsprechenden Config gestartet werden, anschliessend muss Cilium noch in die Conig und diese mittels Service reboot aktiviert werden:

Listing 13: rke2 server starten

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]:OUTPUT ACCEPT [0:0]
7 -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
8 -A INPUT -p udp -m udp --sport 53 -j ACCEPT
9 -A INPUT -p icmp -j ACCEPT
10 -A INPUT -i lo -j ACCEPT
11 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
12 -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
13 -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
14 -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
15 -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
16 -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
17 -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
18 -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
19 -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 14: iptables entries server

Für den Connect der Agents muss noch ein Token generiert werden:

Listing 15: rke2 server token

VI.I.II.IV agents

VI.I.II.V 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 16: 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 17: 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 18: local-path-storage aktualisieren

VI.I.II.VI 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 19: 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 20: MetallB konfigurieren

Das Manifest musste danach eingespielt werden:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/rke2/metallb-values.yaml
```

Listing 21: MetallB Konfiguration einspielen

VI.I.II.VII 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 22: 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 23: 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           "node": "sks1183",  
16           "paths": ["/srv/data/local-path-provisioner"]  
17         },  
18         {  
19           "node": "sks1184",  
20           "paths": ["/srv/data/local-path-provisioner"]  
21         },  
22         {  
23           "node": "sks1185",  
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 24: 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 25: local-path-storage 250GiB aktualisieren

VI.II **yugabyteDB**

VI.II.I **Prerequisites**

VI.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 26: yugabyteDB - StorageClass setzen

Die Storage Class 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 27: yugabyteDB - StorageClass / PersistentVolume aktivieren

VI.II.II Installation

Zuerst muss ein Namespace erstellt werden:

```
1 kubectl create namespace yb-platform
```

Listing 28: YugabyteDB - Namespace

```
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: 3Gi
20     storageClass: "yb-storage"
21   tserver:
22     count: 1
23     size: 3Gi
24     storageClass: "yb-storage"
25
26 resource:
27   master:
28     requests:
29       cpu: "1"
30       memory: 2Gi
31     limits:
32       cpu: "1"
33       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
34       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
35       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
36       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
37       memory: 2Gi
38   tserver:
39     requests:
40       cpu: "1"
41       memory: 4Gi
42     limits:
43       cpu: "1"
44       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
45       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
46       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
47       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
```

```

48     memory: 4Gi
49
50 replicas:
51   master: 3
52   tserver: 3
53   ## Used to set replication factor when isMultiAz is set to true
54 totalMasters: 3
55
56 partition:
57   master: 0
58   tserver: 0
59
60 updateStrategy:
61   type: RollingUpdate
62
63 # Used in Multi-AZ setup
64 masterAddresses: ""
65
66 isMultiAz: false
67 AZ: ""
68
69 # 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: ""
107   LS0tLS1CRUdJTiBDRVJUSUZJQOFURS0tCk1JSUM2VENDQWRHZ0F3SUJBZ01CQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJRp0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakFlRncweE9UQX1NRGd3TURRd01qSmFGdzB5T1RBeU1EVXdNRFF3TWpK
108   =
109   key: ""
110   LS0tLS1CRUdJTiBSUOEgUFJJVkJURSBLRVktLS0tLQpNSU1FcEFJQkFBSONBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBx0XJpWDZqRXJ1YnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQLOJ1M2RxMW4yb0RDZkZUTDB4aTI0V01kTFFyckEy
111   =
112 ## When tls.certManager.enabled=false
113 ## nodeCert and clientCert will be used only when rootCA.key is empty.
114 ## Will be ignored and genSignedCert will be used to generate
115 ## node and client certs if rootCA.key is provided.
116 ## cert and key are base64 encoded content of certificate and key.
117 nodeCert:
118   cert: ""
119   key: ""
120 clientCert:
121   cert: ""
122   key: ""
123
124 gflags:
125   master:
126     default_memory_limit_to_ram_ratio: 0.85
127   tserver: {}
128 #   use_cassandra_authentication: false
129
130 PodManagementPolicy: Parallel
131
132 enableLoadBalancer: true
133
134 ybc:
135   enabled: false
136   ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#resource-requests-and-limits-of-pod-and-container
137   ## Use the above link to learn more about Kubernetes resources configuration.
138   # resources:
139   #   requests:
140   #     cpu: "1"
141   #     memory: 1Gi
142   #   limits:
143   #     cpu: "1"
144   #     memory: 1Gi
145
146 ybCleanup: {}
147   ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#resource-requests-and-limits-of-pod-and-container
148   ## Use the above link to learn more about Kubernetes resources configuration.
149   # resources:
150   #   requests:
151   #     cpu: "1"
152   #     memory: 1Gi
153   #   limits:
154   #     cpu: "1"
155   #     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: "multicluster.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     #             values:
339     #               - "yb-master"
340     #     topologyKey: kubernetes.io/hostname
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 ## extraVolumesMounts 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 29: 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 30: YugabyteDB - Installation

VI.II.III Rekonfiguration mit 250GiB Storage

VI.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 31: YugabyteDB - Deinstallieren

VI.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
36             - sks1185

```

Listing 32: YugabyteDB - StorageClass setzen

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/yugabytedb/yugabytedb/storageclass_250gib.yaml
```

Listing 33: YugabyteDB - StorageClass / PersistentVolume Große Volumes aktivieren

VI.II.III.III Installation - 250GiB

Zuerst muss wieder der Namespace erstellt werden:

```
1 kubectl create namespace yb-platform
```

Listing 34: 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: "
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ01CQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJRD0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakFlRncweE9UQX1NRGd3TURRd01qSmFGdzB5T1RBeU1EVXdNRFF3TWpK
="
91   key: "
LS0tLS1CRUdJTiBSU0EgUFJJVkfURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBx0XJpWDZqRXJlYnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQLOJ1M2RxMW4yb0RDZkZUTDB4aTiOV01kTFFyckEy
="
92 nodeCert:
93   cert: ""
94   key: ""
95 clientCert:
96   cert: ""
97   key: ""

98 gflags:
99 master:
100   default_memory_limit_to_ram_ratio: 0.85
101 tserver:
102   ysql_beta_features: true          # gramic, wird über vacuum obentigt
103   follower_unavailable_considered_sec: 300 # gramic, autobalancing auf 5 Minuten heruntersetzen
104
105 PodManagementPolicy: Parallel
106
107 enableLoadBalancer: true
108
109
110 ybc:
111   enabled: false
112
113
114 ybCleanup:
115   resources:
116     requests:

```

```

117   cpu: "1"
118   memory: 0.5Gi
119   limits:
120     cpu: "1"
121     memory: 0.5Gi
122
123 domainName: "cluster.local"
124
125 serviceEndpoints:
126   - name: "yb-master-ui"
127     type: LoadBalancer
128     annotations:
129       metallb.universe.tf/loadBalancerIPs: 10.0.20.151
130     clusterIP: ""
131     ## Sets the Service's externalTrafficPolicy
132     externalTrafficPolicy: "Cluster"
133     app: "yb-master"
134     loadBalancerIP: ""
135     ports:
136       http-ui: "7000"
137
138   - name: "yb-tserver-service"
139     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: "multicluster.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 35: YugabyteDB - Helm Chart Manifest 250GiB

VI.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 36: 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 -- Genereller Tablespace erstellen
9 drop tablespace if exists eval_data_tablespace;
10 CREATE TABLESPACE eval_data_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');

```

Listing 37: 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 38: YugabyteDB - Benchmarking - Table Size

VI.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_serviceOfficers;create role customer_serviceOfficers;
22 drop role if exists legal_affairs;
23 create role legal_affairs;
24
25 -- User erstellen
26 drop user if exists hrm_1;
27 drop user if exists hrm_2;
28 create user hrm_1 with password 'hrm1' role hrm;
29 create user hrm_2 with password 'hrm2' role hrm;
30
31 drop user if exists cso_1;
32 drop user if exists cso_2;
33 create user cso_1 with password 'cso1' role customer_serviceOfficers;
34 create user cso_2 with password 'cso2' role customer_serviceOfficers;
35
36 drop user if exists la_1;
37 drop user if exists la_2;
38 create user la_1 with password 'la1' role legal_affairs;
39 create user la_2 with password 'la2' role legal_affairs;
40
41 -- Schemas erstellen
42 drop schema if exists hrm;
43 create schema hrm authorization hrm;
44 drop schema if exists accountands;
45 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 -- self_healing_employees für Schema hrm drop table if exists hrm.self_healing_employees;
76 create table hrm.self_healing_employees (
77     employees_id int primary key,
78     firstname varchar(255) not null,
79     lastname varchar(255) not null,
80     birthday date not null,
81     postal_code varchar(50),
82     street varchar(255),
83     country_code varchar(2),
84     phone varchar(25),
85     mail varchar(255) check (mail like '%@%')
86 ) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
87 create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;
88
89 -- self_healing_accountand_protocol für Schema accountands
90 drop table if exists accountands.self_healing_accountand_protocol;
91 create table accountands.self_healing_accountand_protocol (
92     acc_protocol_id int primary key,
93     description varchar(100) not null,
94     protocol_date date not null,
95     employees_id int not null,
96     rapport TEXT,
97     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
98 ) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;
99
100 -- self_healing_intranet für public Schema
101 drop table if exists generell.self_healing_intranet;
102 create table generell.self_healing_intranet (
103     intranet_id int primary key,
104     content text
105 ) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;
106
107 -- self_healing_intranet für public Schema
108 drop table if exists generell.self_healing_intranet_users;
109 create table generell.self_healing_intranet_users (
110     intranet_user_id int primary key,
111     employees_id int not null,
112     foreign key (employees_id) references hrm.self_healing_employees(employees_id) on update restrict on delete restrict
113 );
114 create unique index intranet_unique_combi on generell.self_healing_intranet_users(intranet_user_id, employees_id);

```

Listing 39: 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 );
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 40: YugabyteDB - Self Healing Tests - Init Data

Während dem 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 41: 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_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 42: YugabyteDB - Self Healing Tests - Recovery Data

VI.III sks9016 - YugabyteDB

VI.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
2

```

Listing 43: sks9016 - Download YugabyteDB On-Premise

Im nächsten Schritt wird es im /opt entpackt und das post_install.sh-Skript ausgeführt:

```

1 root@sk9016:/opt# tar xvfz yugabyte-2.21.0.0-b545-linux-x86_64.tar.gz && cd yugabyte-2.21.0.0/
2 ...
3 root@sk9016:/opt/yugabyte-2.21.0.0# ./bin/post_install.sh
4

```

Listing 44: sks9016 - Installation YugabyteDB On-Premise

Um nun zu Testen, ob das ganze funktioniert, kann eine Verbindung zum Evaluationssystem hergestellt werden:

```

1 root@sk9016:/opt/yugabyte-2.21.0.0# cd /opt/yugabyte-2.21.0.0/postgres/bin/
2 root@sk9016:/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 45: sks9016 - Check YugabyteDB On-Premise

Damit ist der Benchmarking-Server ready.

VI.IV Patroni

VI.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:INPUT ACCEPT [0:0]
4 :FORWARD ACCEPT [0:0]
5 :OUTPUT ACCEPT [0:0]
6 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
7 -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
8 -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
9 -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
10 -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
11 -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
12 -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
13 -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
14 -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
15 -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 # postgres
20 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
21 # patroni
22 -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 46: 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 47: 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.svc.first-it.ch:8080";
5 Acquire::https::Proxy "http://sproxy.svc.first-it.ch:8080";
6 Acquire::http::proxy::foreman.ksgr.ch "DIRECT";
```

Listing 48: 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 einen 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 49: 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 50: Patroni - Prerequisites installieren

Im nächsten Schritt müssen Patroni und der PostgreSQL Cluster gestoppt werden:

```
1 systemctl stop postgresql patroni
```

Listing 51: 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 52: 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 53: Patroni - Checks

Damit kann zum etcd übergegangen werden.

VI.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 54: 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,ETCD_ADVERTISE_CLIENT_URLS=http://10.0.28.16:2379"
6 ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
7 ETCD_INITIAL_CLUSTER_STATE="new"

```

Listing 55: 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 56: etcd - restart

Es sollte nun ein Member gelistet werden:

```
1 etcdctl member list
```

Listing 57: etcd - member list

Damit kann nun das Bootstrapping gestartet werden.

VI.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 58: 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 59: 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 60: 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 61: 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 62: 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 63: 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 64: 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 65: Patroni Bootstrap - Disable PostgreSQL

VI.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 66: HAProxy - Hostliste

Nun müssen die Repositories aktualisiert werden und HAproxy installiert werden:

```
1 apt update
2 apt install haproxy
```

Listing 67: 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 68: 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      # Global log configuration
11   log global
12   # Set mode to TCP
13   mode tcp
14   # Number of retries
15   retries 2
16   # Client timeout
17   timeout client 30m
18   # Connect timeout
19   timeout connect 4s
20   # Server timeout
21   timeout server 30m
22   # Check timeout
23   timeout check 5s
24
25 # Stats configuration
26 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 69: HAProxy - Konfiguration

Nun kann HAProxy neu gestartet werden:

```

1 systemctl restart haproxy
2 systemctl status haproxy

```

Listing 70: HAProxy - Restart

Nun kann HAProxy geöffnet werden: <http://10.0.28.16:8080/>

Es sollte nun so aussehen:

Statistics Report for pid 138787

> General process information

```
pid = 138787 (process #1, nbproc = 1, nbthread = 4)
uptime = 10d 19h 3m 35s
system limits: memmax = unlimited; ulimit-n = 252
maxsock = 252; maxconn = 100; maxpipes = 0
current connn = 2; current pipes = 0/0; conn rate = 2/sec; bit rate = 0.543 kbps
Running tasks: 0/23; idle = 100 %
```

active UP
 active UP, going down
 active DOWN, going up
 active or backup DOWN, going up
 not checked
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance
 Note: "NOLBY"|"DRAIN" = UP with load-balancing disabled.

Display option: External resources:
 • Scope :
 • Primary site
 • Hide DOWN servers
 • Refresh now
 • CSV export
 • JSON export (schema)

stats																			Server												
	Queue			Session rate			Sessions			Bytes			Denied		Errors		Warnings		Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme	Thrtle				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis										
Frontend	-	-	-	2	2	-	2	2	100	8	-	1 716	85 188	0	0	3	-	-	-	-	-	OPEN	-	-	-	-	-	-	-		
Backend	0	0	0	0	0	0	0	0	10	0	0s	1 716	85 188	0	0	0	0	0	0	0	0	10d19h UP	-	0	0	0	0	0	0	0	
postgres																			Server												
	Queue			Session rate			Sessions			Bytes			Denied		Errors		Warnings		Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtme	Thrtle				
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis										
Frontend	-	-	-	0	225	-	0	44	100	80 717	-	278 082 821 991	105 305 084	0	0	0	-	-	-	-	-	OPEN	-	-	-	-	-	-	-	-	
postgres01	0	0	-	0	2	-	0	4	100	19	1d3h	41 244 767 171	8 621 308	0	0	1	0	0	0	0	0	7d3h UP	L7OK/200 in 4ms	1/1	Y	-	20	6	2d15h	-	
postgres02	0	0	-	0	225	-	0	44	100	80 697	80 697	8d2h	236 838 054 820	95 683 776	0	0	0	0	0	0	0	0	7d3h DOWN	L7STS/503 in 3ms	1/1	Y	-	18	5	8d4h	-
postgres03	0	0	-	0	0	-	0	0	100	0	0	? ?	0	0	0	0	0	0	0	0	0	10c19h DOWN	L7STS/503 in 3ms	1/1	Y	-	1	1	10d19h	-	
Backend	0	0	0	0	225	-	0	44	10	80 717	80 718	1d3h	278 082 821 991	105 305 084	0	0	1	1	0	0	0	0	7d3h UP	-	1/1	1	0	0	9	2d22s	-

Abbildung XXXIX: HAProxy - Web-GUI

VI.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 71: 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 chmod 700 /srv/data/self_healing_test_data_tablespace
15
16 mkdir -p /srv/data/pg_wal
17 chown -R postgres:postgres /srv/data/pg_wal
18 chmod 700 /srv/data/pg_wal
```

Listing 72: 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 73: 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 74: 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 75: 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 76: 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 77: 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 78: 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 1GiB reduziert.

Das zwingt den Primary dazu, die WAL-Files nur bis zu Maximal 1GiB 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 16MiB erhöht.

wal_writer_delay

Wenn der WAL-Writer einen Flush durchgeführt hat, geht er in ein Timeout.

Dies wurde auf 20ms 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 1MiB 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 1GiB 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 5GiB begrenzt.

commit_delay

Maximal 20ms soll gewartet werden, bevor ein Transaktions-Commit geflushed wird.

commit_siblings

Maximal 10 Transktionen sollen offen bleiben, 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 3min 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 workers (8) für das parallelisieren verwendet werden.

shared_buffers

Der gesamte Server hat 16GiB Memory, $\frac{1}{4}$ davon soll als Buffer dienen, also 4GiB.

maintenance_work_mem

1GiB soll zur Verfügung stehen für den AUTOVACUUM-Job oder Maintenance-Jobs wie das indexieren usw.

work_mem

Der ganze PostgreSQL Cluster soll 12GiB 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 200GiB Diskspace allokiert werden.

vacuum_cost_delay

Der AUTOVACUUM-Job soll maximal 2ms ruhen.

vacuum_cost_limit

Der gesamte Prozess darf maximal zehn sekunden schlafen.

bgwriter_delay

Der Hintergrundprozess soll nur 10ms 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 mittels 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                "max_connections": 1100,
12                "superuser_reserved_connections":10,
13                "max_worker_processes": 16,
14                "wal_log_hints": "on",
15                "max_wal_senders": 32,
16                "max_replication_slots": 32,
17                "wal_keep_size": "1GB",
18                "wal_level": "logical",
19                "wal_buffers": "16MB",
20                "wal_writer_delay": "20ms",
21                "wal_writer_flush_after": "1MB",
22                "min_wal_size": "1GB",
23                "max_wal_size": "5GB",
24                "commit_delay": 20,
25                "commit_siblings": 10,
26                "checkpoint_timeout": "5min",
27                "checkpoint_completion_target": 0.95,
28                "archive_mode": "off",
29                "max_standby_archive_delay": "10min",
30                "max_standby_streaming_delay": "3min",
31                "wal_receiver_status_interval": "1s",
32                "hot_standby_feedback": "on",
33                "wal_receiver_timeout": "60s",
34                "max_logical_replication_workers": 8,
35                "max_sync_workers_per_subscription": 8,
36                "shared_buffers": "4GB",
37                "maintenance_work_mem": "1GB",
38                "work_mem": "12GB",
39                "temp_file_limit": "200GB",
40                "vacuum_cost_delay": "2ms",
41                "vacuum_cost_limit": 10000,
42                "bgwriter_delay": "10ms",
43                "bgwriter_lru_maxpages": 800,
44                "bgwriter_lru_multiplier": "5.0"
45            }
46        }
47    }
48 JSON
```

Listing 79: Patroni - 250GiB set Parameter

VI.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;
11 CREATE TABLESPACE eval_data_tablespace owner postgres location '/var/lib/patroni/eval_data_tablespace';
```

Listing 80: 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 81: 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 82: Patroni - Benchmarking - Tablespaces erneut erstellen

VI.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 -- Genereller Tablespace erstellen
9 drop tablespace if exists self_healing_datas_tablespace;
10 CREATE TABLESPACE self_healing_datas_tablespace owner postgres location '/srv/data/self_healing_test_data_tablespace';
11
12 -- Rollen erstellen
13 drop role if exists hrm;
14 create role hrm;
15 drop role if exists accountands;
16 create role accountands;
17 drop role if exists customer_service_officers;
18 create role customer_service_officers;
19 drop role if exists legal_affairs;
20 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_service_officers;
31 create user cso_2 with password 'cso2' role customer_service_officers;
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_service_officers;
44 create schema customer_service_officers authorization customer_service_officers;
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_service_officers 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_service_officers 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;create table customer_serviceOfficers.self_healing_accounts (
60   account_id int primary key,
61   firstname varchar(255) not null,
62   lastname varchar(255) not null,
63   birthday date not null,
64   postal_code varchar(50),
65   street varchar(255),
66   country_code varchar(2),
67   phone varchar(25),
68   mail varchar(255) check (mail like '%@%')
69 ) tablespace self_healing_datas_tablespace;
70 create unique index accounts_personal_mark on customer_serviceOfficers.self_healing_accounts(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;
71
72 -- self_healing_employees für Schema hrm
73 drop table if exists hrm.self_healing_employees;
74 create table hrm.self_healing_employees (
75   employees_id int primary key,
76   firstname varchar(255) not null,
77   lastname varchar(255) not null,
78   birthday date not null,
79   postal_code varchar(50),
80   street varchar(255),
81   country_code varchar(2),
82   phone varchar(25),
83   mail varchar(255) check (mail like '%@%')
84 ) tablespace self_healing_datas_tablespace;
85 create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday) tablespace self_healing_indices_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 83: 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 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 84: Patroni - 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 select * from customer_serviceOfficers.self_healing_accounts;select * from hrm.self_healing_employees;
21 select * from accountands.self_healing_accountand_protocol;
22 select * from generell.self_healing_intranet;
23 select * from generell.self_healing_intranet_users;

```

Listing 85: 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_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 86: Patroni - Self Healing Tests - Recovery Data

VI.V Stackgres mit Citus

VI.V.I Prerequisites

VI.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 87: StackGres-Citus - StorageClass setzen

Die Storage Class und das PercistenVolume 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 88: StackGres-Citus - StorageClass / PersistentVolume aktivieren

VI.V.II Installation

```
1 kubectl create namespace sg-platform
```

Listing 89: 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 # Section to configure Prometheus integration.
231 prometheus:
232
233 allowAutobind: true

```

```

232 # Section to configure Grafana integration
233 grafana:
234   # -- When set to 'true' embed automatically Grafana into the Web Console by creating the
235   # StackGres dashboards and the read-only role used to read it from the Web Console
236   autoEmbed: false
237   # -- The schema to access Grafana. By default http. (used to embed manually and
238   # automatically grafana)
239   schema: http
240   # -- (string) The service host name to access grafana (used to embed manually and
241   # automatically Grafana).
242   webHost:
243     # -- The datasource name used to create the StackGres Dashboards into Grafana
244   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.svc.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 90: 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 91: 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 92: 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 93: 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 94: 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 95: StackGres-Citus - System Passwort Cleanup

VI.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 96: 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 97: StackGres-Citus - Benchmarking - SGInstanceProfile Shard

Deployt 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 98: StackGres-Citus - Benchmarking - Instanz-Profil Deploy

Auf ein Deployment einer SGPostgresConfig wurde verzichtet, da für Patroni in den ersten drei Benchmarks keine spezifischen Anpassungen erfuhrt.

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 99: StackGres-Citus - Benchmarking - SGShardedCluster

Der Deploy:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGShardedCluster_pgbench.yaml
```

Listing 100: 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 101: StackGres-Citus - Benchmark DB Passwort

VI.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 102: 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 103: StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard

Deployt wird ebenfalls via kubectl:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_self_healing_coord.yaml
2 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGInstanceProfile_self_healing_shard.yaml
```

Listing 104: 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 105: 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 106: 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 107: StackGres-Citus - Self Healing Testing DB Passwort

VI.V.V Rekonfiguration mit 250GiB Storage

VI.V.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 108: StackGres-Citus - Deinstallieren

VI.V.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 109: StackGres-Citus - StorageClass setzen

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/storageclass_big.yaml
```

Listing 110: StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren

VI.V.V.III Installation

```
1 kubectl create namespace sg-platform
```

Listing 111: StackGres-Citus - Namespace 250GiB

Natürlich muss auch wieder das helm-Manifest deployt 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 112: StackGres-Citus - Installation 250GiB

VI.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 113: 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 114: 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 115: StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB

Das Manifest für den Benchmark-Cluster wurde die Shard-Cluster Anzahl auf 2 reduziert und die PVC entsprechend dimensioniert:

```
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  
18     disableConnectionPooling: true  
19     sgInstanceProfile: "sg-pgbench-coordinator"  
20   shards:  
21     clusters: 2 # gramic, 19.04.2024: 250GiB Tabelle  
22     instancesPerCluster: 1  
23     pods:  
24       persistentVolume:  
25         size: '230Gi'  
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 116: StackGres-Citus - Benchmarking - SGShardedCluster 250GiB

Der Deploy:

```
1 kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus/SGShardedCluster_pgbench.yaml
```

Listing 117: StackGres-Citus - Benchmark - Cluster Deploy 250GiB

VI.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.

VI.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

```

```

27 -- Schemas erstellen
28 drop schema if exists hrm;
29 create schema hrm authorization hrm;drop schema if exists accountands;
30 create schema accountands authorization accountands;
31 drop schema if exists customer_serviceOfficers;
32 create schema customer_serviceOfficers authorization customer_serviceOfficers;
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_serviceOfficers 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_serviceOfficers to postgres;
44 grant all on all tables in schema generell to postgres;
45
46 -- self_healing_accounts für Schema customer_serviceOfficers
47 drop table if exists customer_serviceOfficers.self_healing_accounts;
48 create table customer_serviceOfficers.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_serviceOfficers.self_healing_accounts(firstname, lastname, birthday);
60 SELECT create_reference_table('customer_serviceOfficers.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
79 drop table if exists accountands.self_healing_accountand_protocol;
80 create table accountands.self_healing_accountand_protocol (

```

```

81 acc_protocol_id int primary key,
82 description varchar(100) not null,      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 118: 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
12 insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
13 insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
14 insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');
15
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 119: 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 );
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 120: 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 121: StackGres-Citus - Self Healing Tests - Recovery Data

VII Evaluationssysteme - Benchmarking

VII.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_4_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
33 # Benchmarking dql
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_1_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_2_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_3_yugabytedb_dql_benchmark.txt
37 ./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
38
39 #####
40 # 3. Lauf      #
41 # ca. 50GiB    #
42 #####
43 ./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
44
45 # Benchmarking mixed
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_1_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_2_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_3_yugabytedb_mixed_benchmark.txt
```

```

49 ./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
50 # Benchmarking dql ./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
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_2_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_3_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_4_yugabytedb_dql_benchmark.txt
54
55
56 ######
57 # 4. Lauf #
58 # ca. 250GiB #
59 #####
60 ./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
61
62 # Benchmarking mixed
63 ./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
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_2_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_3_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_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 122: YugabyteDB - Benchmarking-Commands

Die grössere 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 123: YugabyteDB - Benchmarking - Table Size SQL

VII.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
14 # Benchmarking dql
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_1_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_2_patroni_dql_benchmark.txt
17 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
18 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
19 #####
20 # 2. Lauf #
21 # ca. 15GiB #
22 #####
23 # Init
24 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
25
26 # Benchmarking mixed
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_1_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_2_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_3_patroni_mixed_benchmark.txt
30 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
31 # Benchmarking dql
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_1_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_2_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_3_patroni_dql_benchmark.txt
35 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
36
37 #####
38 # 3. Lauf #
39 # ca. 50GiB #
40 #####
41 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
42
43 # Benchmarking mixed
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_1_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_2_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_3_patroni_mixed_benchmark.txt
47 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
48 # Benchmarking dql
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_1_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_2_patroni_dql_benchmark.txt
51 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
52 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
53
54
55 #####
56 # 4. Lauf #
57 # ca. 250GiB #
58 #####

```

```

59 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
60 # Benchmarking mixedpgbench -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 124: Patroni - Benchmarking-Commands

Die grössere der Tabellen lässt sich wie folgt auslesen:

```
1 SELECT pg_size.pretty( pg_database_size('pgbench_eval_bench') );
```

Listing 125: Patroni - Benchmarking - Table Size SQL

VII.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      #
3 #    ca. 5GiB     #
4 #####
5 # Init
6 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=400 --foreign-keys --fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
7
8 # Benchmarking mixed
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_1_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_2_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_3_stackgresmixed_benchmark.txt
12 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
13
14 # Benchmarking dql
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_1_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_2_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_3_stackgresdql_benchmark.txt
18 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
19
20 #####
21 #    2. Lauf      #
22 #    ca. 15GiB    #
23 #####
24 # Init
25 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=1200 --foreign-keys --fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
26

```

```

27 # Benchmarking mixed
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_1_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_2_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_3_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 #
39 # ca. 50GiB #
40 #####
41 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=3999 --foreign-keys --fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
42
43 # Benchmarking mixed
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_1_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_2_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_3_stackgres_mixed_benchmark.txt
47 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
48 # Benchmarking dql
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_1_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_2_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_3_stackgres_dql_benchmark.txt
52 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
53
54
55 #####
56 # 4. Lauf #
57 # ca. 250GiB #
58 #####
59 pgbench --host=10.0.20.106 --port=5432 --initialize --scale=16784 --foreign-keys --fillfactor=100 --username=dtgvpf --username=postgres pgbench_eval_bench
60
61 # Benchmarking mixed
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_1_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_2_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_3_stackgresmixed_benchmark.txt
65 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
66 # Benchmarking dql
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_1_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_2_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_3_stackgresdql_benchmark.txt
70 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 126: StackGres-Citus - Benchmarking-Commands

Die grössere der Tabellen lässt sich wie folgt auslesen:

```
1 SELECT * FROM citus_tables;
```

Listing 127: 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.

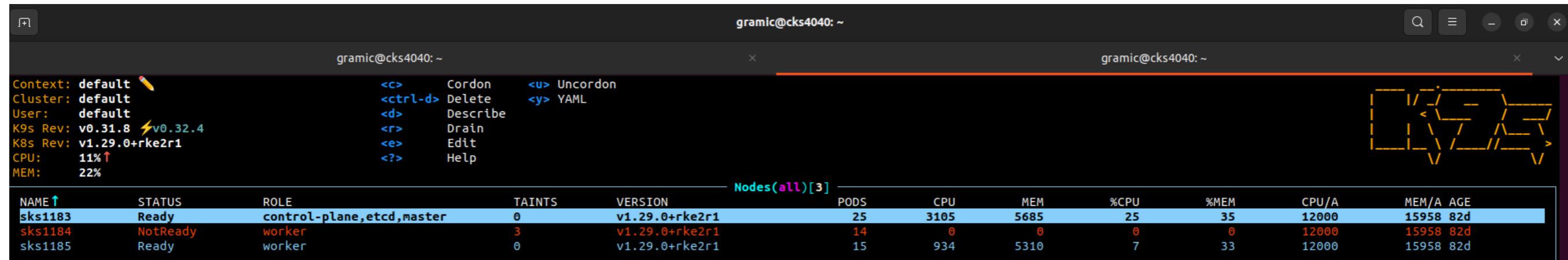
Die korrekte Grösse muss wie folgt kalkuliert werden:

$$realSize = \frac{size}{[coordinator_{syncInstances} + (shard_{clusters} \times shard_{instancesPerCluster})]}$$

VIII Evaluationssysteme - Testing

VIII.I StackGres - Citus

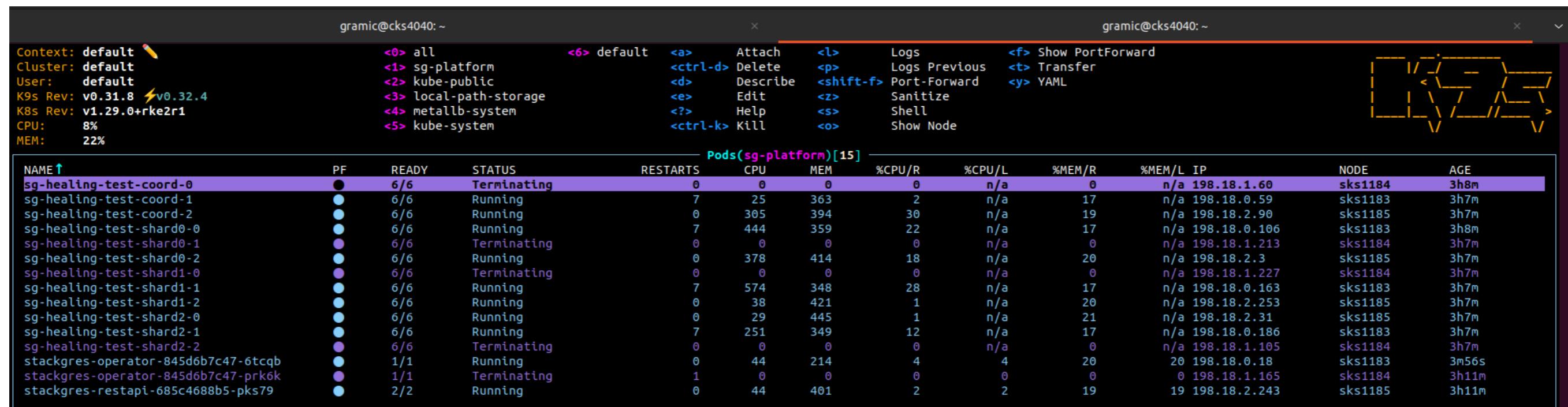
Der Node ging down, als der Server sks1184 heruntergefahren wurde:



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	3	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	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-845d6b7c47-6tcqb	●	1/1	Running	0	44	214	4	4	20	20	198.18.0.18	sks1183	3m56s
stackgres-operator-845d6b7c47-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:

```
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
```

	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

VIII.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:

```

Context: default <0> tail <6> 1h <shift-c> Clear <t> Toggle Timestamp
Cluster: default <1> head <cc> Copy <w> Toggle Wrap
User: default <2> 1m <m> Mark
K9s Rev: v0.31.8 <3> 5m <ctrl-s> Save
K8s Rev: v1.29.0+rke2r1 <4> 15m <s> Toggle AutoScroll
CPU: 1% <5> 30m <f> Toggle FullScreen
MEM: 36%
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 @ 0x55e8dcc46377 google::LogMessage::SendToLog()
yb-tserver @ 0x55e8dcc472dd google::LogMessage::Flush()
yb-tserver @ 0x55e8dcc47959 google::LogMessageFatal::~LogMessageFatal()
yb-tserver @ 0x55e8dde9756c yb::server::HybridClock::NowWithError()
yb-tserver @ 0x55e8dde95661 yb::server::HybridClock::NowRange()
yb-tserver @ 0x55e8ddf8e39 yb::tablet::TransactionCoordinator::Impl::Poll()
yb-tserver @ 0x55e8dddada3c yb::rpc::Poller::Poll()
yb-tserver @ 0x55e8ddd5d6e _ZN5boost4asio6detail_completion_handlerIZN2yb3rpc9Scheduler4Impl11HandleTimerERKNS_6system10error_codeEEUlxE_NS0_10io_context19basic_executor_typeINSt3__19alloc
yb-tserver @ 0x55e8ddd93a34 boost::asio::detail::scheduler::run()
yb-tserver @ 0x55e8ddd92f57 yb::rpc::IoThreadPool::Impl::Execute()
yb-tserver @ 0x55e8de61acb4 yb::Thread::SuperviseThread()
yb-tserver @ 0x7f1de2421694 start_thread
yb-tserver @ 0x7f1de271e41d __clone

```

Abbildung XLV: YugabyteDB - Too big clock skew is detected

Eine Folge ist, dass kein neuer Leader bestimmt werden kann:

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
a6609fe063354432bcb38b435fe1413d	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:

Screenshot captured													
You can paste the image from the clipboard.													
<f> Show PortForward <t> Transfer <y> YAML													
NAME	PF	READY	STATUS	RESTARTS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE	AGE
yb-master-0	(F)	2/2	Running	0	11	238	0	n/a	23	n/a	198.18.1.68	sks1184	34h
yb-master-1	(●)	2/2	Running	0	5	154	0	n/a	15	n/a	198.18.0.31	sks1183	34h
yb-master-2	(●)	2/2	Running	0	4	210	0	n/a	20	n/a	198.18.2.180	sks1185	7m54s
yb-tserver-0	(●)	2/2	Running	0	8	1101	0	n/a	8	n/a	198.18.2.207	sks1185	7m54s
yb-tserver-1	(●)	1/2	CrashLoopBackOff	5	1	1	0	n/a	0	n/a	198.18.1.77	sks1184	34h
yb-tserver-2	(●)	2/2	Running	0	27	7591	1	n/a	61	n/a	198.18.0.136	sks1183	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:

```
Logs(yb-platform/yb-master-0)[tail]
Autoscroll:On   FullScreen:Off   Timestamps:Off   Wrap:Off
yb-master E0423 12:16:45.676072 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.550s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:46.785388 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.909s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:47.802490 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.708s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:49.147837 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.841s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:50.516947 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.875s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:51.640313 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.517s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:52.773218 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.550s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:53.887251 178 hybrid_clock.cc:181] Too big clock skew is detected: 1.006s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:55.343684 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.932s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:56.360834 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.948s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:57.771939 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.665s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:59.056317 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.954s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:00.150394 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:01.252478 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.718s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:02.274430 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.789s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:03.282796 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.895s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:04.575225 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.770s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:06.040117 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.964s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:07.525339 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.512s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:08.979657 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.983s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:10.274869 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.726s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:11.373963 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.682s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:12.413090 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.760s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:13.464146 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.618s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:14.517447 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.782s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:15.871708 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.638s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:17.392766 178 hybrid_clock.cc:181] Too big clock skew is detected: 1.000s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:18.841015 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.661s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:19.903977 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.629s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:21.329376 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.832s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:22.704488 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.669s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:23.861863 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.961s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:24.931900 105 hybrid_clock.cc:181] Too big clock skew is detected: 0.827s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:26.376091 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.785s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:27.492267 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.568s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:28.507580 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.675s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:29.576633 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.558s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:30.750717 178 hybrid_clock.cc:181] Too big clock skew is detected: 1.039s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:31.845826 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.857s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:32.951920 178 hybrid_clock.cc:181] Too big clock skew is detected: 0.864s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
```

Abbildung XLVIII: YugabyteDB - Too big clock skew is detected - tmaster

```

Context: default
Cluster: default
User: default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU: 0%
MEM: 22%  

Logs(yb-platform/yb-tserver-1)[tail]  

Autoscroll:On FullScreen:Off Timestamps:Off Wrap:Off  

yb-tserver E0423 11:44:56.039235 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:44:57.039614 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.798s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:44:58.039772 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.779s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:44:59.039928 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.718s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:00.040057 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.781s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:01.040220 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.672s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:02.040351 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.777s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:03.040545 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.759s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:04.040710 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.759s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:05.040827 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.810s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:06.040973 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.810s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:07.041210 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.800s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:08.041371 139 hybrid_clock.cc:181] Too big clock skew is detected: 0.786s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:09.041580 139 hybrid_clock.cc:181] Too big clock skew is detected: 0.799s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:10.041759 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.811s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:11.041956 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.794s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:12.042100 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.815s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:13.042212 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.818s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:14.042308 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.708s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:15.042454 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.776s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:16.042851 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.807s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:17.043002 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.804s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:17.043002 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.804s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:18.043179 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.803s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:19.043371 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.791s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:20.043519 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.652s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:21.043641 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.717s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:22.043789 139 hybrid_clock.cc:181] Too big clock skew is detected: 0.820s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:23.043915 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.745s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:24.044063 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.822s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:25.044202 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.824s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:26.044380 142 hybrid_clock.cc:181] Too big clock skew is detected: 0.827s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:27.044510 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.823s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:28.044632 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.792s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:29.044777 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:30.044929 141 hybrid_clock.cc:181] Too big clock skew is detected: 0.818s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:31.045047 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.829s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:32.045161 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.773s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:33.045315 140 hybrid_clock.cc:181] Too big clock skew is detected: 0.822s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000  

yb-tserver E0423 11:45:34.045454 139 hybrid_clock.cc:181] Too big clock skew is detected: 0.781s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000

```

Abbildung XLIX: YugabyteDB - Too big clock skew is detected - tserver

YugabyteDB erlaubt in so einem Fall keine Zugriffe mehr auf den Cluster.

So wird verhindert, dass der Cluster korrumptiert wird.

IX Testsystem - Installation

IX..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 128: 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 129: Ansible - Repository Clone

Das Verzeichnis ist postgresql_cluster beinhaltet die Playbooks und Ressourcen.

```
1 cd postgresql_cluster/
2
```

Listing 130: 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

IX.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
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 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8080 -j ACCEPT
21 # postgres
22 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
23 # pgbouncer
24 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6432 -j ACCEPT# 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 131: 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 132: 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.sivc.first-it.ch:9090";
4 Acquire::http::proxy::foreman.ksgr.ch "DIRECT";
5

```

Listing 133: Testsystem - apt-Proxy Settings

IX.II Deployment

Zuerst muss das Inventory angepasst hat 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 # 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
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
30 # Connection settings
31 [all:vars]
32 ansible_connection='ssh',
33 ansible_ssh_port='22',
34 ansible_user='itgramic',
35 ansible_ssh_pass='<Secret ;-)>' # "sshpass" package is required for use "ansible_ssh_pass"
36 #ansible_ssh_private_key_file=
37 ansible_python_interpreter='/usr/bin/env python3'
38
39 [pgbackrest:vars]
40
41

```

Listing 134: 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
4 proxy_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: "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090"
9
10 # Cluster variables
11 #cluster_vip: "" # IP address for client access to the databases in the cluster (optional).
12 cluster_vip: "10.0.22.178" # IP address for client access to the databases in the cluster (optional).
13 vip_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)
17 patroni_cluster_name: "k8s-core-psql" # the cluster name (must be unique for each cluster)
18 patroni_install_version: "3.3.0" # or 'latest'
19
20 patroni_superuser_username: "postgres"
21 patroni_superuser_password: "<Password Secure / Safe>"
22 patroni_replication_username: "replicator"
23 patroni_replication_password: "<Password Secure / Safe>"
24
25 #synchronous_mode: false # or 'true' for enable synchronous database replication
26 synchronous_mode: true # or 'true' for enable synchronous database replication
27 synchronous_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
29 synchronous_node_count: 2 # number of synchronous standby databases
30
31 # Load Balancing
32 with_haproxy_load_balancing: true # or 'true' if you want to install and configure the load-balancing
33 haproxy_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
42 haproxy_maxconn:
43   global: 100000
44   master: 10000
45   replica: 10000
46 haproxy_timeout:
47   client: "60m"
48   server: "60m"
49
50 # keepalived (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'true')
51 keepalived_virtual_router_id: "{{ cluster_vip.split('.')[3] | int }}" # The last octet of 'cluster_vip' IP address is used by default.
52 # virtual_router_id - must be unique in the network (available values are 0..255).
53
54 # vip-manager (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'false')
55 vip_manager_version: "2.4.0" # version to install
56 vip_manager_conf: "/etc/patroni/vip-manager.yml"
57 vip_manager_interval: "1000" # time (in milliseconds) after which vip-manager wakes up and checks if it needs to register or release ip addresses.
58 vip_manager_iface: "{{ vip_interface }}" # interface to which the virtual ip will be added
59 vip_manager_ip: "{{ cluster_vip }}" # the virtual ip address to manage
60 vip_manager_mask: "24" # netmask for the virtual ip
61
62
63 # DCS (Distributed Consensus Store)
64 dcs_exists: false # or 'true' if you don't want to deploy a new etcd cluster
65 dcs_type: "etcd" # or 'consul'
66
67 # if dcs_type: "etcd" and dcs_exists: false
68 etcd_version: "3.5.11" # version for deploy etcd cluster

```

```

69 etcd_data_dir: "/var/lib/etcd"
70 etcd_cluster_name: "etcd-{{ patroni_cluster_name }}" # ETCD_INITIAL_CLUSTER_TOKEN
71
72 # if dcs_type: "etcd" and dcs_exists: true
73 patroni_etcd_hosts: [] # list of servers of an existing etcd cluster
74
75 patroni_etcd_namespace: "service" # (optional) etcd namespace (prefix)
76 patroni_etcd_username: "" # (optional) username for etcd authentication
77 patroni_etcd_password: "" # (optional) password for etcd authentication
78 patroni_etcd_protocol: "" # (optional) http or https, if not specified http is used
79
80 # if dcs_type: "consul"
81 consul_version: "1.15.8"
82 consul_config_path: "/etc/consul"
83 consul_configd_path: "{{ consul_config_path }}/conf.d"
84 consul_data_path: "/var/lib/consul"
85 consul_domain: "consul" # Consul domain name
86 consul_datacenter: "dc1" # Datacenter label (can be specified for each host in the inventory)
87 consul_disable_update_check: true # Disables automatic checking for security bulletins and new version releases
88 consul_enable_script_checks: true # This controls whether health checks that execute scripts are enabled on this agent
89 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 pgbouncer_install: false
112     checks:
113       - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/primary", interval: "2s" }
114       - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" } # comment out this check if pgbouncer_install: false
115   - name: "{{ patroni_cluster_name }}"
116     id: "{{ patroni_cluster_name }}-replica"
117     tags: ['replica']
118     port: "{{ pgbouncer_listen_port }}"
119     checks:
120       - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/replica?lag={{ patroni_maximum_lag_on_replica }}", interval: "2s" }
121       - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }
122

```

```

123 # PostgreSQL variables
124 postgresql_version: "16"
125 # postgresql_data_dir: see vars/Debian.yml or vars/RedHat.yml
126 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.
127 postgresql_port: "5432"
128 postgresql_encoding: "UTF8" # for bootstrap only (initdb)
129 postgresql_locale: "en_US.UTF-8" # for bootstrap only (initdb)
130 postgresql_data_checksums: true # for bootstrap only (initdb)
131 postgresql_password_encryption_algorithm: "scram-sha-256" # or "md5" if your clients do not work with passwords encrypted with SCRAM-SHA-256
132
133 # (optional) list of users to be created (if not already exists)
134 postgresql_users:
135   - { name: "{{ pgbouncer_auth_username }}", password: "{{ pgbouncer_auth_password }}", flags: "LOGIN", role: "" }
136   - { name: "xksgr_sks1160_gitlab", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
137   - { name: "xksgr_sks1172_harbor", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
138   - { name: "xksgr_sks1195_kcsso", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
139   - { name: "xksgr_k8s_core_psql_monitor", password: "<Password Secure / Safe>", flags: "LOGIN", role: "pg_monitor" }
140   - { name: "xksgr_k8s_core_psql_backup", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
141
142 # (optional) list of databases to be created (if not already exists)
143 #postgresql_databases: []
144 postgresql_databases:
145   - { db: "k8s_core_gitlab_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1160_gitlab" }
146   - { db: "k8s_core_harbor_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1172_harbor" }
147   - { db: "k8s_core_keycloak_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1195_kcsso" }
148   - { db: "gramic_test", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "postgres" }
149
150 # (optional) list of schemas to be created (if not already exists)
151 postgresql_schemas: []
152 #   - { schema: "myschema", db: "mydatabase", owner: "mydb-user" }
153
154 # (optional) list of database extensions to be created (if not already exists)
155 #postgresql_extensions: []
156 postgresql_extensions:
157   - { ext: "pgstattuple", db: "postgres" }
158   - { ext: "pgstattuple", db: "k8s_core_gitlab_prod" }
159   - { ext: "pgstattuple", db: "k8s_core_harbor_prod" }
160   - { ext: "pgstattuple", db: "k8s_core_keycloak_prod" }
161   - { ext: "pgstattuple", db: "gramic_test" }
162
163 # postgresql parameters to bootstrap dcs (are parameters for example)
164 postgresql_parameters:
165   - { option: "max_connections", value: "500" }
166   - { option: "superuser_reserved_connections", value: "5" }
167   - { option: "password_encryption", value: "{{ postgresql_password_encryption_algorithm }}" }
168   - { option: "max_locks_per_transaction", value: "512" }
169   - { option: "max_prepared_transactions", value: "0" }
170   - { option: "huge_pages", value: "try" } # or "on" if you set "vm_nr_hugepages" in kernel parameters
171   - { option: "shared_buffers", value: "{{ (ansible_memtotal_mb * 0.25) | int }}MB" } # by default, 25% of RAM
172   - { option: "effective_cache_size", value: "{{ (ansible_memtotal_mb * 0.75) | int }}MB" } # by default, 75% of RAM
173   - { option: "work_mem", value: "128MB" } # please change this value
174   - { 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 pgbounce_conf_dir: "/etc/pgbounce"
286 pgbounce_log_dir: "/var/log/pgbounce"
287 pgbounce_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
288 pgbounce_listen_port: 6432
289 pgbounce_max_client_conn: 10000
290 pgbounce_max_db_connections: 1000
291 pgbounce_max_prepared_statements: 1024
292 pgbounce_default_pool_size: 20
293 pgbounce_query_wait_timeout: 120
294 pgbounce_default_pool_mode: "session"
295 pgbounce_admin_users: "{{ patroni_superuser_username }}" # comma-separated list of users, who are allowed to change settings
296 pgbounce_stats_users: "{{ patroni_superuser_username }}" # comma-separated list of users who are just allowed to use SHOW command
297 pgbounce_ignore_startup_parameters: "extra_float_digits,geqo,search_path"
298 pgbounce_auth_type: "{{ postgresql_password_encryption_algorithm }}"
299 pgbounce_auth_user: true # or 'false' if you want to manage the list of users for authentication in the database via userlist.txt
300 pgbounce_auth_username: pgbounce # user who can query the database via the user_search function
301 pgbounce_auth_password: "<Password Secure / Safe>"
302 pgbounce_auth_dbname: "postgres"
303 pgbounce_client_tls_sslmode: "disable"
304 pgbounce_client_tls_key_file: ""
305 pgbounce_client_tls_cert_file: ""
306 pgbounce_client_tls_ca_file: ""
307 pgbounce_client_tls_protocols: "secure" # allowed values: tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3, all, secure (tlsv1.2,tlsv1.3)
308 pgbounce_client_tls_ciphers: "default" # allowed values: default, secure, fast, normal, all (not recommended)
309
310 pgbounce_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 concatenating.
404 pg_probackup_command_parts:
405   - "pg_probackup-{{ pg_probackup_version }}"
406   - " restore -B {{ pg_probackup_dir }}"
407   - " --instance {{ pg_probackup_instance }}"
408   - " -j {{ pg_probackup_threads }}"
409   - " {{ pg_probackup_add_keys }}"
410 pg_probackup_restore_command: "{{ pg_probackup_command_parts | join('') }}"
411 pg_probackup_patroni_cluster_bootstrap_command: "{{ pg_probackup_command_parts | join('') }}"
412
413 # WAL-G
414 wal_g_install: false # or 'true'
415 wal_g_version: "3.0.0"
416 wal_g_json: # config https://github.com/wal-g/wal-g#configuration
417   - { option: "AWS_ACCESS_KEY_ID", value: "{{ AWS_ACCESS_KEY_ID | default('') }}" } # define values or pass via --extra-vars
418   - { option: "AWS_SECRET_ACCESS_KEY", value: "{{ AWS_SECRET_ACCESS_KEY | default('') }}" } # define values or pass via --extra-vars
419   - { option: "WALG_S3_PREFIX", value: "{{ WALG_S3_PREFIX | default('') }}" } # define values or pass via --extra-vars
420   - { option: "WALG_COMPRESSION_METHOD", value: "brotli" } # or "lz4", "lzma", "zstd"
421   - { option: "WALG_DELTA_MAX_STEPS", value: "6" } # determines how many delta backups can be between full backups
422   - { option: "PGDATA", value: "{{ postgresql_data_dir }}" }
423   - { option: "PGHOST", value: "{{ postgresql_unix_socket_dir }}" }
424   - { option: "PGPORT", value: "{{ postgresql_port }}" }
425   - { option: "PGUSER", value: "{{ patroni_superuser_username }}" }
426 #   - { option: "AWS_S3_FORCE_PATH_STYLE", value: "true" } # to use Minio.io S3-compatible storage
427 #   - { option: "AWS_ENDPOINT", value: "http://minio:9000" } # to use Minio.io S3-compatible storage
428 #   - { option: "", value: "" }
429 wal_g_archive_command: "wal-g wal-push %p"
430 wal_g_patroni_cluster_bootstrap_command: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST"
431
432 # Define job_parts outside of wal_g_cron_jobs
433 # Ensure there is a space at the beginning of each part to prevent commands from concatenating.
434 wal_g_backup_command:
435   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
436   - " && wal-g backup-push {{ postgresql_data_dir }} > {{ postgresql_log_dir }}/walg_backup.log 2>&1"
437 wal_g_delete_command:
438   - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
439   - " && 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 company's backup retention policy)
452 user: "postgres"
453 file: /etc/cron.d/walg
454 minute: "30"
455 hour: "6"
456 day: "*"
457 month: "*"
458 weekday: "*"
459 job: "{{ wal_g_delete_command | join('') }}"
460
461 # pgBackRest
462 pgbackrest_install: false # or 'true'
463 pgbackrest_install_from_pgdg_repo: false # or 'false'
464 pgbackrest_stanza: "{{ patroni_cluster_name }}" # specify your --stanza
465 pgbackrest_repo_type: "posix" # or "s3", "gcs", "azure"
466 pgbackrest_repo_host: "" # dedicated repository host (optional)
467 pgbackrest_repo_user: "postgres"
468 pgbackrest_conf_file: "/etc/pgbackrest/pgbackrest.conf"
469 # config https://pgbackrest.org/configuration.html
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 (details: https://pgbackrest.org/user-guide.html#async-archiving)
484     - { option: "archive-get-queue-max", value: "1GiB" }
485 stanza: # [stanza_name] section
486   - { option: "process-max", value: "4" }
487   - { option: "log-level-console", value: "info" }
488   - { option: "recovery-option", value: "recovery_target_action=promote" }
489   - { option: "pg1-socket-path", value: "{{ postgresql_unix_socket_dir }}" }
490   - { 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 --type=diff --stanza={{ pgbackrest_stanza }} backup; fi"
543
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 "pgbackrest" --delta restore).
546 # 2) And also the patroni cluster "{{ patroni_cluster_name }}" will be removed from the DCS (if exist) before recovery.
547
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 135: Testsystem - Deployment - main.yml

Jetzt muss geprüft werden, ob die Hosts für Ansible erreichbar sind:

```

1 root@sks1000:~/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 }
34

```

Listing 136: Deploy - Anhang - Ansible Ping

Deployt wird wie folgt: Nach dem deployment sollte folgende Ausgabe zu sehen sein:

IX.III Maintenance

Das Inventory bleibt gleich.

Geändert wurde nur das main.yml:

1
2

Listing 137: Testsystem - Maintenance - main.yml

IX.IV	Prequenteries - Erweiterungstests
IX.V	Haproxy erweitern
IX.VI	Patroni Node
X	Testsystem - Testing
XI	Exkurs Architekturen - Umsysteme und Prinzipien
XI.I	Raft-Konsensus
XI.II	local-path-provisioner
XII	Python Utils
XII.I	zotero.py

```
1 import json
2 import pybtex
3 import requests
4 import os
5 from pybtex.database import BibliographyData, Entry, Person
6 from dateutil.parser import parse
7 import math
8 import yaml
9
10 # Load the Configurations
11 def load_configuration(zotero_conf_filename):
12     zotero_bibtex_config = dict()
13     zotero_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')
14
15     yaml_path = os.path.join(zotero_conf_dir, zotero_conf_filename)
16
17     with open(yaml_path, "r") as file:
18         zotero_bibtex_config = yaml.load(file, Loader=yaml.FullLoader)
19
20     return zotero_bibtex_config
21 def download_zotero_data(URL, API_KEY):
22     zotero_result = list()
23     response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
24     response = response.json()
25     zotero_raw = json.dumps(response, ensure_ascii=False) # json.loads(response)
26     zotero_result = json.loads(zotero_raw)
27     return zotero_result
```

```

28
29 # Get the bibtex Data from Zotero
30 def get_data(zotero_bibtex_config):
31     result_limit = int(zotero_bibtex_config.get('result_limit'))
32     access_type = zotero_bibtex_config.get('access_type')      zotero_access_id = zotero_bibtex_config.get('zotero_access_id')
33     collection_id = zotero_bibtex_config.get('collection_id')
34     API_KEY = zotero_bibtex_config.get('api_key')
35     zotero_data = list()
36     URL = 'https://api.zotero.org/' + str(access_type) + '/' + str(zotero_access_id) + '/collections/' + str(
37         collection_id) + '/items?limit=1?format=json?sort=dateAdded?direction=asc'
38
39     response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
40
41     header_dict = response.headers
42     total_elemets = int(header_dict.get('Total-Results'), 0)
43
44     if total_elemets < result_limit:
45         URL_ALL_ITEMS = 'https://api.zotero.org/' + str(access_type) + '/' + str(
46             zotero_access_id) + '/collections/' + str(collection_id) + '/items?limit=' + str(
47                 result_limit) + '?format=json?sort=dateAdded?direction=asc',
48         zotero_result = downlaod_zotero_datas(URL_ALL_ITEMS, API_KEY)
49
50         zotero_data.extend(zotero_result)
51     else:
52         runs = int(math.ceil(total_elemets / result_limit))
53         index = 0
54         start_index = 0
55         while index < runs:
56             URL_Separated = 'https://api.zotero.org/' + str(access_type) + '/' + str(
57                 zotero_access_id) + '/collections/' + str(collection_id) + '/items?limit=' + str(
58                     result_limit) + '?format=json?sort=dateAdded?direction=asc' + '&start=' + str(start_index)
59             zotero_result = downlaod_zotero_datas(URL_Separated, API_KEY)
60
61             zotero_data.extend(zotero_result)
62
63             start_index += result_limit
64             index += 1
65
66     return zotero_data
67
68 # Convert String to Datetime
69 def convert_to_datetime(input_str, parserinfo=None):
70     return parse(input_str, parserinfo=parserinfo)
71
72 # Get Dates from Datetime
73 def get_dates(date, bibtex_item_type, bibtex_month_attributes):
74     dated_date = convert_to_datetime(date)
75     return_value = dict()
76     if bibtex_item_type in bibtex_month_attributes:
77         year = dated_date.year
78         month = dated_date.month
79         return_value = {'year': year, 'month': month}
80     else:
81         year = dated_date.year

```

```

82     return_value = {'year': year}
83 
84 # Split Creators into biblatex Creators
85 def split_creators(creators):
86     if creators != []:
87 
88         creatorlist = ''
89         for index, creator in enumerate(creators):
90             type = creator.get('creatorType')
91             firstname = creator.get('firstName')
92             lastname = creator.get('lastName')
93             name = creator.get('name')
94             if type == 'author':
95 
96                 if name and not (firstname or lastname):
97                     creatorlist = creatorlist + name
98                     if index != len(creators) - 1:
99                         creatorlist = creatorlist + ' and '
100                else:
101                    creatorlist = creatorlist + lastname + ',' + firstname
102                    if index != len(creators) - 1:
103                        creatorlist = creatorlist + ' and '
104            else:
105                creatorlist = 'unknown author'
106 
107 bib_entry = 'author=' + '"' + creatorlist + '"'
108 
109 return bib_entry
110 
111 # Write the *.bib File
112 def write_bibliography(zotero_data, zotero_bibtex_config):
113     keystore_file = zotero_bibtex_config.get('keystore_file')
114     keystore_path = zotero_bibtex_config.get('keystore_filepath')
115     tex_dir = os.path.join(os.path.dirname(os.getcwd()), keystore_path)
116 
117 yaml_path = os.path.join(tex_dir, keystore_file)
118 
119 with open(yaml_path, "r") as file:
120     zotero_bibtex_keys = yaml.load(file, Loader=yaml.FullLoader)
121 
122 zotero_bibtex_keys_specials = {
123     'thesis': {'phdthesis': ['dissertation', 'phd', 'doctorial', 'doctor', 'doktor', 'doktorarbeit'],
124                'masterthesis': ['ma', 'master', 'masters']}
125 }
126 zotero_bibtex_attributes_special = {
127     'date': 'get_dates',
128     'creators': 'split_creators'
129 }
130 bibtex_month_attributes = ['booklet', 'masterthesis', 'phdthesis', 'techreport']
131 # Bibliography
132 # tex_dir = os.path.join(os.path.dirname(os.getcwd()), 'source')
133 bibtex_path = zotero_bibtex_config.get('bibtex_filepath')
134 tex_dir = os.path.join(os.path.dirname(os.getcwd()), bibtex_path)
135 # tex_dir = os.path.join(os.getcwd(), 'src', 'content')

```

```

136 # file_name = 'Datenbank_Projektauftrag_Michael_Gruber.bib'
137 file_name = zotero_bibtex_config.get('bibtex_filename')
138 file_path = os.path.join(tex_dir, file_name)
139
140 # bib_datas = BibliographyData()
141 listKeys = list()
142 bib_data = ''
143 for zotero_items in zotero_data:
144     biblio_item = zotero_items.get('data')
145     itemkeys = biblio_item.keys()
146     listKeys.extend(biblio_item.keys())
147     zotero_item_key = biblio_item.get('key')
148     zotero_item_title = biblio_item.get('title')
149     zotero_item_nameofact = biblio_item.get('nameOfAct')
150     zotero_item_nameofcase = biblio_item.get('caseName')
151     zotero_item_subject = biblio_item.get('subject')
152     zotero_item_type = biblio_item.get('itemType')
153
154     # some item types have no titles
155     # set the special names instead of the title
156     if zotero_item_title:
157         bibtex_item_titel = zotero_item_title
158     else:
159         if zotero_item_type == 'statute':
160             biblio_item['title'] = zotero_item_nameofact
161             bibtex_item_titel = zotero_item_nameofact
162         elif zotero_item_type == 'case':
163             biblio_item['title'] = zotero_item_nameofcase
164             bibtex_item_titel = zotero_item_nameofcase
165         elif zotero_item_type == 'email':
166             biblio_item['title'] = zotero_item_subject
167             bibtex_item_titel = zotero_item_subject
168
169     if zotero_item_type == 'thesis':
170         master_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('masterthesis')
171         phd_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('phdthesis')
172
173     # First Master thesis
174     if any(item in bibtex_item_titel for item in master_list):
175         bibtex_item_key = 'masterthesis'
176     # Second PHD Thesis
177     elif any(item in bibtex_item_titel for item in phd_list):
178         bibtex_item_key = 'phdthesis'
179     else:
180         bibtex_item_key = 'masterthesis'
181     else:
182         if zotero_bibtex_keys.get(zotero_item_type).get('key'):
183             bibtex_item_key = zotero_bibtex_keys.get(zotero_item_type).get('key')
184         else:
185             bibtex_item_key = 'misc'
186
187     # get all Keys for the zotero item type
188     entryset = '\n'
189     entry = '',

```

```

190
191 zotero_item_attributes = zotero_bibtex_keys.get(zotero_item_type).get('attributes').keys()           item_attributes = sorted(zotero_item_attributes, reverse=True)
192
193 for index, item_attribute in enumerate(item_attributes):
194     bibtex_item_attribute = zotero_bibtex_keys.get(zotero_item_type).get('attributes').get(item_attribute)
195     zotero_item_value = biblio_item.get(item_attribute)
196     zotero_item_value_extra = ''
197     bibtex_item_attribute_extra = ''
198
199 # Special Cases
200 if bibtex_item_attribute == 'SPECIALCHECK' and zotero_item_value not in ['', None]:
201     bibtex_special_attribute = zotero_bibtex_attributes_special.get(item_attribute)
202
203     match bibtex_special_attribute:
204         case 'get_dates':
205             zotero_item_value = get_dates(zotero_item_value, bibtex_item_key, bibtex_month_attributes)
206             if zotero_item_value.get('month'):
207                 zotero_item_value_extra = zotero_item_value.get('month')
208                 bibtex_item_attribute_extra = 'month'
209
210             zotero_item_value = zotero_item_value.get('year')
211             bibtex_item_attribute = 'year'
212         case 'split_creators':
213             authors = split_creators(zotero_item_value)
214             entryset = entryset + authors
215     elif bibtex_item_attribute == 'howpublished':
216         if zotero_item_value not in ['', None, []]:
217             zotero_item_value = '\\url{' + zotero_item_value + '}'
218
219 if bibtex_item_attribute not in ['', 'None', 'author', 'SPECIALCHECK'] and zotero_item_value not in ['', None, []]:
220     if zotero_item_value_extra:
221
222         if type(zotero_item_value_extra) == "string":
223             entryset = entryset + str(bibtex_item_attribute_extra) + '=' + str(zotero_item_value_extra) + '\n'
224         else:
225             entryset = entryset + str(bibtex_item_attribute_extra) + '=' + str(zotero_item_value_extra)
226
227         if index != len(item_attributes) - 1:
228             entryset = entryset + ',\n'
229         else:
230             entryset = entryset + '\n'
231
232         if type(zotero_item_value) == str and not zotero_item_value.isnumeric():
233             entryset = entryset + str(bibtex_item_attribute) + '=' + str(zotero_item_value) + '\n'
234         else:
235             entryset = entryset + str(bibtex_item_attribute) + '=' + str(zotero_item_value)
236
237         if index != len(item_attributes) - 1:
238             entryset = entryset + ',\n'
239         else:
240             entryset = entryset + '\n'
241
242 # create the Entry
243 entry = '@' + bibtex_item_key + '{' + zotero_item_key + ',\n'

```

```

244     entry = entry + entryset + '}'
245     bib_data = bib_data + '\n' + entry
246 # parse String to pybtex.database Object
247 # bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex", encoding='ISO-8859-1')
248 bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex", encoding='Utf-8')
249 # Save pybtex.database to file
250 # BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding='ISO-8859-1')
251 BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding='utf-8')

252

253

254 zotero_bibtex_config = load_configuration('zotero_bibtex_configuration.yaml')
255 zotero_data = get_data(zotero_bibtex_config)
256 write_bibliography(zotero_data, zotero_bibtex_config)

```

Listing 138: Python LaTex - zotero.py - Zotero BibLaTex Importer

XII.II zotero_bibtex_configuration.yaml

```

1 result_limit: 100
2 access_type: "groups"
3 zotero_access_id: "5222465"
4 collection_id: "PC3BW6EP"
5 api_key: "6Xgb3XhGjQXwA8NuZgu3bw3s"
6 keystore_file: "zotero_biblatex_keystore.yaml"
7 keystore_filepath: "source/configuration"
8 bibtex_filepath: "source"
9 bibtex_filename: "Diplomarbeit_Michael_Graber.bib"

```

Listing 139: Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer

XII.III zotero_biblatex_keystore.yaml

```

1 --
2 artwork:
3   key: misc
4   attributes:
5     title: title
6     date: SPECIALCHECK
7     creators: SPECIALCHECK
8     url: howpublished
9     extra: note
10 audioRecording:
11   key: misc
12   attributes:
13     title: title
14     date: SPECIALCHECK
15     creators: SPECIALCHECK
16 bill:
17   key: misc
18   attributes:
19     title: title
20     date: SPECIALCHECK

```

```
21   creators: SPECIALCHECK
22   url: howpublished
23   extra: note
24 blogPost:
25   key: misc
26   attributes:
27     title: title
28     date: SPECIALCHECK
29     creators: SPECIALCHECK
30     url: howpublished
31     extra: note
32 book:
33   key: book
34   attributes:
35     title: title
36     date: SPECIALCHECK
37     creators: SPECIALCHECK
38     publisher: publisher
39     place: address
40 bookSection:
41   key: inbook
42   attributes:
43     title: title
44     date: SPECIALCHECK
45     creators: SPECIALCHECK
46     pages: pages
47     publisher: publisher
48     place: address
49     bookTitle: booktitle
50 case:
51   key: misc
52   attributes:
53     title: title
54     date: SPECIALCHECK
55     creators: SPECIALCHECK
56     url: howpublished
57     extra: note
58 conferencePaper:
59   key: inproceedings
60   attributes:
61     title: title
62     date: SPECIALCHECK
63     creators: SPECIALCHECK
64     series: series
65     proceedingsTitle: booktitle
66     publisher: publisher
67     pages: pages
68     place: address
69 dictionaryEntry:
70   key: misc
71   attributes:
72     title: title
73     date: SPECIALCHECK
74     creators: SPECIALCHECK
```

```
75     url: howpublished
76     extra: note
77 document:
78     key: misc
79     attributes:
80         title: title
81         date: SPECIALCHECK
82         creators: SPECIALCHECK
83         url: howpublished
84         extra: note
85 email:
86     key: misc
87     attributes:
88         title: title
89         date: SPECIALCHECK
90         creators: SPECIALCHECK
91         url: howpublished
92         extra: note
93 encyclopediaArticle:
94     key: misc
95     attributes:
96         title: title
97         date: SPECIALCHECK
98         creators: SPECIALCHECK
99         url: howpublished
100        extra: note
101 film:
102     key: misc
103     attributes:
104         title: title
105         date: SPECIALCHECK
106         creators: SPECIALCHECK
107         url: howpublished
108         extra: note
109 forumPost:
110     key: misc
111     attributes:
112         title: title
113         date: SPECIALCHECK
114         creators: SPECIALCHECK
115         url: howpublished
116         extra: note
117 hearing:
118     key: misc
119     attributes:
120         title: title
121         date: SPECIALCHECK
122         creators: SPECIALCHECK
123         url: howpublished
124         extra: note
125 instantMessage:
126     key: misc
127     attributes:
128         title: title
```

```
129 date: SPECIALCHECK
130 creators: SPECIALCHECK
131 url: howpublished
132 extra: note
133 interview:
134   key: misc
135   attributes:
136     title: title
137     date: SPECIALCHECK
138     creators: SPECIALCHECK
139     url: howpublished
140     extra: note
141 journalArticle:
142   key: article
143   attributes:
144     title: title
145     date: SPECIALCHECK
146     creators: SPECIALCHECK
147     volume: volume
148     pages: pages
149     seriesNumber: number
150     seriesTitle: journal
151     url: url
152 letter:
153   key: misc
154   attributes:
155     title: title
156     date: SPECIALCHECK
157     creators: SPECIALCHECK
158     url: howpublished
159     extra: note
160 magazineArticle:
161   key: article
162   attributes:
163     title: title
164     date: SPECIALCHECK
165     creators: SPECIALCHECK
166     volume: volume
167     pages: pages
168     seriesNumber: number
169     seriesTitle: journal
170     url: url
171 manuscript:
172   key: unpublished
173   attributes:
174     title: title
175     date: SPECIALCHECK
176     creators: SPECIALCHECK
177 map:
178   key: misc
179   attributes:
180     title: title
181     date: SPECIALCHECK
182     creators: SPECIALCHECK
```

```
183 url: howpublished
184 extra: note
185 newspaperArticle:
186   key: article
187   attributes:
188     title: title
189     date: SPECIALCHECK
190     creators: SPECIALCHECK
191     volume: volume
192     pages: pages
193     seriesNumber: number
194     seriesTitle: journal
195     url: url
196 patent:
197   key: misc
198   attributes:
199     title: title
200     date: SPECIALCHECK
201     creators: SPECIALCHECK
202     url: howpublished
203     extra: note
204 podcast:
205   key: misc
206   attributes:
207     title: title
208     date: SPECIALCHECK
209     creators: SPECIALCHECK
210     url: howpublished
211     extra: note
212 presentation:
213   key: misc
214   attributes:
215     title: title
216     date: SPECIALCHECK
217     creators: SPECIALCHECK
218     url: howpublished
219     extra: note
220 radioBroadcast:
221   key: misc
222   attributes:
223     title: title
224     date: SPECIALCHECK
225     creators: SPECIALCHECK
226     url: howpublished
227     extra: note
228 report:
229   techreport: misc
230   attributes:
231     title: title
232     date: SPECIALCHECK
233     creators: SPECIALCHECK
234     url: howpublished
235     extra: note
236 software:
```

```
237 key: misc
238 attributes:
239   title: title
240   date: SPECIALCHECK
241   creators: SPECIALCHECK
242   url: howpublished
243   extra: note
244 computerProgram:
245   key: misc
246   attributes:
247     title: title
248     date: SPECIALCHECK
249     creators: SPECIALCHECK
250     url: howpublished
251     extra: note
252 statute:
253   key: misc
254   attributes:
255     title: title
256     date: SPECIALCHECK
257     creators: SPECIALCHECK
258     url: howpublished
259     extra: note
260 tvBroadcast:
261   key: misc
262   attributes:
263     title: title
264     date: SPECIALCHECK
265     creators: SPECIALCHECK
266     url: howpublished
267     extra: note
268 videoRecording:
269   key: misc
270   attributes:
271     title: title
272     date: SPECIALCHECK
273     creators: SPECIALCHECK
274     url: howpublished
275     extra: note
276 webpage:
277   key: misc
278   attributes:
279     title: title
280     date: SPECIALCHECK
281     creators: SPECIALCHECK
282     url: howpublished
283     extra: note
284 attachment:
285   key: misc
286   attributes:
287     title: title
288     date: SPECIALCHECK
289     creators: SPECIALCHECK
290     url: howpublished
```

```

291     extra: note
292 note:
293     key: misc
294     attributes:
295         title: title
296         date: SPECIALCHECK
297         creators: SPECIALCHECK
298         url: howpublished
299         extra: note
300 standard:
301     key: misc
302     attributes:
303         title: title
304         date: SPECIALCHECK
305         creators: SPECIALCHECK
306         url: howpublished
307         extra: note
308 preprint:
309     key: misc
310     attributes:
311         title: title
312         date: SPECIALCHECK
313         creators: SPECIALCHECK
314         url: howpublished
315         extra: note
316 dataset:
317     key: misc
318     attributes:
319         title: title
320         date: SPECIALCHECK
321         creators: SPECIALCHECK
322         url: howpublished
323         extra: note
324 thesis:
325     key: thesis
326     attributes:
327         title: title
328         date: SPECIALCHECK
329         creators: SPECIALCHECK
330         place: address
331         university: school

```

Listing 140: Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTeX Importer

XII.IV riskmatrix.py

```

1 import os
2 import matplotlib.pyplot as plt
3 import yaml
4
5 # Load Configurations
6 def load_configuration(riskmatrix_conf_filename):
7     riskmatrix_config = dict()

```

```

8
9     riskmatrix_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')    yaml_path = os.path.join(riskmatrix_conf_dir, riskmatrix_conf_filename)
10
11    with open(yaml_path, "r") as file:
12        riskmatrix_config = yaml.load(file, Loader=yaml.FullLoader)
13
14    return riskmatrix_config
15
16 # Load x-y axis tuples
17 def load_xy_axis_tuples(riskmatrix_config):
18     startpath = riskmatrix_config.get('riskmatrix').get('startpath')
19     riskmatrix_xy_axis_tuples_dir = riskmatrix_config.get('riskmatrix').get('configfile_path')
20     riskmatrix_xy_axis_tuples_config = riskmatrix_config.get('riskmatrix').get('configfile_name')
21
22     if startpath == 'omedir':
23         directory = os.path.join(os.getcwd(), riskmatrix_xy_axis_tuples_dir)
24     else: # parentdir
25         directory = os.path.join(os.path.dirname(os.getcwd()), riskmatrix_xy_axis_tuples_dir)
26
27     riskmatrix_xy_axis_tuples_path = os.path.join(directory, riskmatrix_xy_axis_tuples_config)
28     riskmatrix_xy_axis_tuples = dict()
29     riskmatrix_xy_axis_tuples_aux = dict()
30
31     with open(riskmatrix_xy_axis_tuples_path, "r") as file:
32         riskmatrix_xy_axis_tuples_aux = yaml.load(file, Loader=yaml.FullLoader)
33
34     for string_key in riskmatrix_xy_axis_tuples_aux:
35         value = riskmatrix_xy_axis_tuples_aux.get(string_key)
36         int_key = eval(string_key)
37         riskmatrix_xy_axis_tuples.update({int_key:value})
38     return riskmatrix_xy_axis_tuples
39
40 # Load Data from csv
41 def get_data(data_path):
42
43     with open(data_path) as f:
44         csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]
45
46     (_, *header), *data = csv_list
47     datas = {}
48     for row in data:
49         key, *values = row
50         datas[key] = {key: value for key, value in zip(header, values)}
51
52     return datas
53
54 # Generate Riskmatrix Image
55 #def riskmatrix(risk, conf, matrix):
56 def riskmatrix(conf, matrix):
57     risks = conf.get('risk_inventory')
58     for risk_conf in risks:
59         # get the risk config datas
60         startpath = conf.get('risks').get(risk_conf).get('startpath')
61         destination = conf.get('risks').get(risk_conf).get('destination_path')

```

```

62 imagename = conf.get('risks').get(risk_conf).get('imagename')
63 datafilename = conf.get('risks').get(risk_conf).get('datafile')           itemname = conf.get('risks').get(risk_conf).get('itemname')
64 x_axis_title = conf.get('risks').get(risk_conf).get('x-axis-title')
65 y_axis_title = conf.get('risks').get(risk_conf).get('y-axis-title')
66 title = conf.get('risks').get(risk_conf).get('title')
67 bubble_standard_size = conf.get('risks').get(risk_conf).get('bubble-standard-size')
68
69 # Identify the index of the axes
70 green = conf.get('risks').get(risk_conf).get('settings').get('green-boxes')
71 yellow = conf.get('risks').get(risk_conf).get('settings').get('yellow-boxes')
72 orange = conf.get('risks').get(risk_conf).get('settings').get('orange-boxes')
73 red = conf.get('risks').get(risk_conf).get('settings').get('red-boxes')
74
75 if startpath == 'omedir':
76     directory = os.path.join(os.getcwd(), destination)
77 else: # parentdir
78     directory = os.path.join(os.path.dirname(os.getcwd()), destination)
79
80 data_path = os.path.join(directory, datafilename)
81 image_path = os.path.join(directory, imagename)
82
83 # get the Data as direct
84 datas = get_data(data_path)
85
86 fig = plt.figure()
87 plt.subplots_adjust(wspace=0, hspace=0)
88 plt.xticks([])
89 plt.yticks([])
90 plt.xlim(0, 5)
91 plt.ylim(0, 5)
92 plt.xlabel(x_axis_title)
93 plt.ylabel(y_axis_title)
94 plt.title(title)
95
96 # This example is for a 5 * 5 matrix
97 nrows = 5
98 ncols = 5
99 axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0, nrows) for c in range(0, ncols)]
100
101 # remove the x and y ticks
102 for ax in axes:
103     ax.set_xticks([])
104     ax.set_yticks([])
105     ax.set_xlim(0, 5)
106     ax.set_ylim(0, 5)
107
108 # Add background colors
109 # This has been done manually for more fine-grained control
110 # Run the loop below to identify the indice of the axes
111 for _ in green:
112     axes[_].set_facecolor('green')
113
114 for _ in yellow:
115     axes[_].set_facecolor('yellow')

```

```

116
117     for _ in orange:           axes[_].set_facecolor('orange')
118
119     for _ in red:             axes[_].set_facecolor('red')
120
121
122     # run through datas and generate axis datas
123     dict_bubble_axis = dict()
124     bubble_axis = list()
125
126     for datasets in datas:
127         # get the datas
128         riskid = datas.get(datasets).get('risk-id')
129         x_axis = int(datas.get(datasets).get('x-axis'))
130         y_axis = int(datas.get(datasets).get('y-axis'))
131         axis_point = matrix.get((x_axis, y_axis))
132         x_axis_text = float(datas.get(datasets).get('x-axis-text'))
133         y_axis_text = float(datas.get(datasets).get('y-axis-text'))
134         x_axis_bubble = float(datas.get(datasets).get('x-axis-bubble'))
135         y_axis_bubble = float(datas.get(datasets).get('y-axis-bubble'))
136         bubble_axis.append(axis_point)
137
138         # merge riks if two or more risks share the same axispoint
139         if dict_bubble_axis.get(axis_point):
140             risktag = dict_bubble_axis.get(axis_point).get('risk')
141             risktag = risktag + '/' + riskid
142             x_axis_text = x_axis_text + 0.25
143             y_axis_text = y_axis_text - 0.5
144             bubble_size = bubble_standard_size * 2
145         else:
146             risktag = itemname + riskid
147             bubble_size = bubble_standard_size
148             dict_axis_value = dict()
149
150             dict_axis_value['risk'] = risktag
151             dict_axis_value['x-axis-text'] = x_axis_text
152             dict_axis_value['y-axis-text'] = y_axis_text
153             dict_axis_value['x-axis-bubble'] = x_axis_bubble
154             dict_axis_value['y-axis-bubble'] = y_axis_bubble
155             dict_axis_value['size'] = bubble_size
156             dict_bubble_axis[axis_point] = dict_axis_value
157
158         # cleanup the list, remove duplicated entries
159         bubble_axis = set(bubble_axis)
160
161         # plot the bubbles and texts in the bubbles
162         for axispoint in bubble_axis:
163             axes[axispoint].scatter(dict_bubble_axis[axispoint]['x-axis-bubble'],
164                                    dict_bubble_axis[axispoint]['y-axis-bubble'],
165                                    dict_bubble_axis[axispoint]['size'], alpha=1)
166             axes[axispoint].text(dict_bubble_axis[axispoint]['x-axis-text'],
167                                 dict_bubble_axis[axispoint]['y-axis-text'], s=dict_bubble_axis[axispoint]['risk'],
168                                 va='bottom', ha='center')
169
# save the plot as image

```

```

170     plt.savefig(image_path)
171 riskmatrix_config = load_configuration('riskmatrix_plotter_conf.yaml')riskmatrix_xy_axis_tuples = load_xy_axis_tuples(riskmatrix_config)
172 riskmatrix(riskmatrix_config, riskmatrix_xy_axis_tuples)

```

Listing 141: Python LaTex - riskmatrix.py - Risikomatrizen

XII.V riskmatrix_plotter_conf.yaml

```

1 risk_inventory:
2   - "postgresql"
3   - "project"
4   - "Postgresql-massnahme"
5   - "Project-massnahme"
6 riskmatrix:
7   startpath: "parentdir"
8   configfile_path: "source/configuration"
9   configfile_name: "riskmatrix_xy_axis_tuple_matrix.yaml"
10 risks:
11   postgresql:
12     riskid: "postgresql"
13     startpath: "parentdir"
14     destination_path: "source/riskmatrix"
15     imagename: "riskmatrixproblem.png"
16     datafile_path: "source/tables"
17     datafile: "riskmatrixproblem.csv"
18     itemname: "R"
19     x-axis-title: "Schadensausmass (SM)"
20     y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
21     title: "Risiko Cockpit PostgreSQL Datenbanken KSGR"
22     bubble-standard-size: 1000
23     settings:
24       green-boxes:
25         - 10
26         - 15
27         - 16
28         - 20
29         - 21
30       yellow-boxes:
31         - 0
32         - 5
33         - 6
34         - 11
35         - 17
36         - 22
37         - 23
38       orange-boxes:
39         - 1
40         - 2
41         - 7
42         - 12
43         - 13
44         - 18
45         - 19

```

```

46      - 24
47  red-boxes:
48    - 3
49    - 4
50    - 8
51    - 9
52    - 14
53 project:
54   riskid: "project"
55   startpath: "parentdir"
56   destination_path: "source/riskmatrix"
57   imagename: "riskmatrix-project.png"
58   datafile_path: "source/tables"
59   datafile: "riskmatrix-project.csv"
60   itemname: "R"
61   x-axis-title: "Schadensausmass (SM)"
62   y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
63   title: "Risiko Cockpit Projekt"
64   bubble-standard-size: 1000
65 settings:
66   green-boxes:
67     - 10
68     - 15
69     - 16
70     - 20
71     - 21
72   yellow-boxes:
73     - 0
74     - 5
75     - 6
76     - 11
77     - 17
78     - 22
79     - 23
80   orange-boxes:
81     - 1
82     - 2
83     - 7
84     - 12
85     - 13
86     - 18
87     - 19
88     - 24
89   red-boxes:
90     - 3
91     - 4
92     - 8
93     - 9
94     - 14
95 Postgresql-massnahme:
96   riskid: "Postgresql-massnahme"
97   startpath: "parentdir"
98   destination_path: "source/riskmatrix"
99   imagename: "Riskmatrixproblem-massnahmen.png"

```

```

100 datafile_path: "source/tables"
101 datafile: "riskmatrixproblem-massnahmen.csv"
102 itemname: "R"
103 x-axis-title: "Schadensausmass (SM)"
104 y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
105 title: "Risiko Cockpit PostgreSQL Datenbanken KSGR - Massnahme"
106 bubble-standard-size: 1000
107 settings:
108   green-boxes:
109     - 10
110     - 15
111     - 16
112     - 20
113     - 21
114   yellow-boxes:
115     - 0
116     - 5
117     - 6
118     - 11
119     - 17
120     - 22
121     - 23
122   orange-boxes:
123     - 1
124     - 2
125     - 7
126     - 12
127     - 13
128     - 18
129     - 19
130     - 24
131   red-boxes:
132     - 3
133     - 4
134     - 8
135     - 9
136     - 14
137 Project-massnahme:
138   riskid: "Project-massnahme"
139   startpath: "parentdir"
140   destination_path: "source/riskmatrix"
141   imagename: "Riskmatrix-project-massnahmen.png"
142   datafile_path: "source/tables"
143   datafile: "riskmatrix-project-massnahmen.csv"
144   itemname: "R"
145   x-axis-title: "Schadensausmass (SM)"
146   y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
147   title: "Risiko Cockpit Projekt - Massnahme"
148   bubble-standard-size: 1000
149   settings:
150     green-boxes:
151       - 10
152       - 15
153       - 16

```

```

154     - 20
155     - 21
156 yellow-boxes:
157     - 0
158     - 5
159     - 6
160     - 11
161     - 17
162     - 22
163     - 23
164 orange-boxes:
165     - 1
166     - 2
167     - 7
168     - 12
169     - 13
170     - 18
171     - 19
172     - 24
173 red-boxes:
174     - 3
175     - 4
176     - 8
177     - 9
178     - 14

```

Listing 142: Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risikomatrizen

XII.VI riskmatrix_xy_axis_tuple_matrix.yaml

```

1 #Matrix
2 #This Matrix translate the x/y axis from a given risk matrix csv to the axispoint.
3 #
4 #The key of each axispoint is an integer tupel (x, y)
5 #So, you can access the axis point this way:
6 #<axispoint> = matrix.get((<x_axis>, <y_axis>))
7 (1, 1): 20
8 (1, 2): 15
9 (1, 3): 10
10 (1, 4): 5
11 (1, 5): 0
12 (2, 1): 21
13 (2, 2): 16
14 (2, 3): 11
15 (2, 4): 6
16 (2, 5): 1
17 (3, 1): 22
18 (3, 2): 17
19 (3, 3): 12
20 (3, 4): 7
21 (3, 5): 2
22 (4, 1): 23
23 (4, 2): 18

```

```

24 (4, 3): 13
25 (4, 4): 8
26 (4, 5): 3
27 (5, 1): 24
28 (5, 2): 19
29 (5, 3): 14
30 (5, 4): 9
31 (5, 5): 4

```

Listing 143: Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples

XII.VII cost_benefit_diagram.py

```

1 import os
2 import matplotlib.pyplot as plt
3 import yaml
4
5 # Get the Configuration
6 def load_configuration():
7     cost_benefit_config = dict()
8     cbd_conf_filename = 'scatter_plotter_conf.yaml'
9     cbd_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')
10    yaml_path = os.path.join(cbd_conf_dir, cbd_conf_filename)
11
12    with open(yaml_path, "r") as file:
13        cost_benefit_config = yaml.load(file, Loader=yaml.FullLoader)
14
15    return cost_benefit_config
16 # Get the Datas
17 def get_data(cost_benefit_config):
18     # Config Variables
19     startpath = cost_benefit_config.get('startpath')
20     destination = cost_benefit_config.get('desitination_path')
21     datafilename = cost_benefit_config.get('datafile')
22
23     if startpath == 'homedir':
24         directory = os.path.join(os.getcwd(), destination)
25     else: # parentdir
26         directory = os.path.join(os.path.dirname(os.getcwd()), destination)
27
28     # get the Datas as dirct
29     data_path = os.path.join(directory, datafilename)
30
31     # load datas from csv into dict
32     with open(data_path) as f:
33         csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]
34
35     (_, *header), *data = csv_list
36     datas = {}
37     for row in data:
38         key, *values = row
39         datas[key] = {key: value for key, value in zip(header, values)}
40

```

```

41 cost_benefit_data = {}
42 for key, value in datas.items(): variant_name = value['variant_name']
43     x_axis = int(value['x-axis'])
44     y_axis = int(value['y-axis'])
45     cost_benefit_data[variant_name] = (x_axis, y_axis)
46
47 return cost_benefit_data
48
49 # Plot the Data
50 def cost_benefit_diagram(cost_benefit_config, cost_benefit_data):
51     # Config Variables
52     startpath = cost_benefit_config.get('startpath')
53     destination = cost_benefit_config.get('desitination_path')
54     imagename = cost_benefit_config.get('imagineame')
55
56     if startpath == 'homendir':
57         directory = os.path.join(os.getcwd(), destination)
58     else: # parentdir
59         directory = os.path.join(os.path.dirname(os.getcwd()), destination)
60
61     # get the Data as direct
62     data_path = os.path.join(directory, imagename)
63
64     # Extract the Data
65     labels, values = zip(*cost_benefit_data.items())
66     x, y = zip(*values)
67
68     # Create Scatter-Diagram
69     plt.scatter(x, y, color=cost_benefit_config.get('scatter-point-color'))
70
71     # X-Lines
72     plt.axhline(y=cost_benefit_config.get('y-axis-line-pos'), color=cost_benefit_config.get('y-axis-line-color'), linestyle=cost_benefit_config.get('y-axis-line-type'), label=cost_benefit_config.get('y-axis-line-label'))
73
74     # Y-Lines
75     plt.axvline(x=cost_benefit_config.get('x-axis-line-pos'), color=cost_benefit_config.get('x-axis-line-color'), linestyle=cost_benefit_config.get('x-axis-line-type'), label=cost_benefit_config.get('x-axis-line-label'))
76
77     # Add Labels
78     plt.xlabel(cost_benefit_config.get('x-axis-title'))
79     plt.ylabel(cost_benefit_config.get('y-axis-title'))
80     plt.title(cost_benefit_config.get('title'))
81
82     # Labling Data Points
83     for label, x_point, y_point in zip(labels, x, y):
84         plt.text(x_point, y_point, label)
85
86     # Show Legends
87     plt.legend()
88
89     # Show Grid
90     plt.grid(True)
91
92     # Save Diagram as PNG

```

```

93     plt.savefig(data_path)
94 cost_benefit_config = load_configuration()cost_benefit_data = get_data(cost_benefit_config)
95 cost_benefit_diagram(cost_benefit_config, cost_benefit_data)

```

Listing 144: Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm

XII.VIII cost_benefit_diagram_plotter_conf.yaml

```

1 startpath: "parentdir"
2 desitination_path: "source/cost_benefit_diagram"
3 datafile: "cost_benefit_diagram.csv"
4 imagename: "cost_benefit_diagram.png"
5 scatter-point-color: "blue"
6 x-axis-title: "Punkte"
7 x-axis-line-pos: 80
8 x-axis-line-label: "Kosten-Minimum"
9 x-axis-line-type: "--"
10 x-axis-line-color: "red"
11 y-axis-title: "Kosten"
12 y-axis-line-pos: 80
13 y-axis-line-label: "Punkte-Minimum"
14 y-axis-line-type: "--"
15 y-axis-line-color: "green"
16 title: "Kosten-Nutzen-Diagramm Beispiel"

```

Listing 145: Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm

XII.IX pandas_dataframe_to_latex_table.py

```

1 import os
2 import pandas as pd
3 import yaml
4 from pathlib import Path
5 import chardet
6
7 import csv
8
9 # Get the Configuration
10 def load_configuration(plt_conf_filename):
11     panda_latex_tables_config = dict()
12     plt_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')
13     yaml_path = os.path.join(plt_conf_dir, plt_conf_filename)
14
15     with open(yaml_path, "r") as file:
16         panda_latex_tables_config = yaml.load(file, Loader=yaml.FullLoader)
17
18     return panda_latex_tables_config
19
20
21 def get_data(startpath, destination, tablefilename, datafile_path, datafile, alternative_csv_load, separator, decimal):
22     # Config Variables
23     if startpath == 'omedir':

```

```

24     directory = os.path.join(os.getcwd(), datafile_path)
25 else: # parentdir
26     directory = os.path.join(os.path.dirname(os.getcwd()), datafile_path)
27 # get the Data as direct
28 data_path = os.path.join(directory, datafile)
29
30 # load datas from csv into dict
31 detected = chardet.detect(Path(data_path).read_bytes())
32 encoding = detected.get("encoding")
33
34 # if alternative_csv_load:
35 #     with open(data_path, 'r', encoding=encoding) as file:
36 #         reader = csv.reader(file)
37 #         data = list(reader)
38 #
39 #         # panda_table_data = pd.DataFrame(data, columns=data[0])
40 #         # panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding, lineterminator='\n', engine='python')
41 #         panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding, lineterminator='\n')
42 #         df_dtype = {
43 #             "Nr.": int,
44 #             "Anforderung": str,
45 #             "Beschreibung": str,
46 #             "System": str,
47 #             "Muss / Kann": str
48 #         }
49 #         # panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, lineterminator='\n', dtype=df_dtype)
50 #         # panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding)
51 #     else:
52 #         panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
53 #         panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, low_memory=False, engine='python')
54 #         panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, engine='python', dtype='unicode')
55 #         readed = open(data_path, 'r', encoding=encoding)
56 #         panda_table_data = pd.read_csv(open(data_path, 'r', encoding=encoding), sep=",", decimal=". ", encoding=encoding)
57 #         panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding="ISO-8859-1")
58 #         panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, chunksize=10)
59
60 # for chunk in pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, chunksize=5):
61 #     print(chunk)
62 panda_table_data = pd.DataFrame()
63 temp = pd.read_csv(data_path, iterator=True, sep=",", decimal=". ", encoding=encoding, chunksize=1000)
64 panda_table_data = pd.concat(temp, ignore_index=True)
65
66 df_dtype = {
67     "Nr.": int,
68     "Anforderung": str,
69     "Beschreibung": str,
70     "System": str,
71     "Muss / Kann": str
72 }
73 panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, engine='python', dtype=df_dtype)
74 panda_table_data = pd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding, dtype=df_dtype)
75
76 # import dask.dataframe as dd
77 # df = dd.read_csv(data_path, sep=",", decimal=". ", encoding=encoding)

```

```

78 # panda_table_data = df
79 print(encoding)    panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
80 # return data
81 return panda_table_data
82
83
84 def create_latex_tables(panda_latex_tables_config):
85     plt_tables = panda_latex_tables_config.get('tables_inventory')
86     for table_item in plt_tables:
87         # id and filesystem informations
88         table_id = panda_latex_tables_config.get('tables').get(table_item).get('id')
89         isbigfile = panda_latex_tables_config.get('tables').get(table_item).get('isbigfile')
90         has_longtexts = panda_latex_tables_config.get('tables').get(table_item).get('has_longtexts')
91         if isbigfile or has_longtexts:
92             alternative_cvs_load = True
93         else:
94             alternative_cvs_load = False
95         startpath = panda_latex_tables_config.get('tables').get(table_item).get('startpath')
96         destination = panda_latex_tables_config.get('tables').get(table_item).get('destination_path')
97         tablefilename = panda_latex_tables_config.get('tables').get(table_item).get('tablefilename')
98         datafile_path = panda_latex_tables_config.get('tables').get(table_item).get('datafile_path')
99         datafile = panda_latex_tables_config.get('tables').get(table_item).get('datafile')
100        if startpath == 'homedir':
101            directory = os.path.join(os.getcwd(), destination)
102        else: # parentdir
103            directory = os.path.join(os.path.dirname(os.getcwd()), destination)
104        tablefile = os.path.join(directory, tablefilename)
105        separator = panda_latex_tables_config.get('tables').get(table_item).get('separator')
106        decimal = panda_latex_tables_config.get('tables').get(table_item).get('decimal')
107
108        # column operations
109        column_operations = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('datas')
110
111        # group by / aggregation
112        groupby_values = panda_latex_tables_config.get('tables').get(table_item).get('group_by')
113        group_by_function = panda_latex_tables_config.get('tables').get(table_item).get('group_by_function')
114        # selected_rows = panda_latex_tables_config.get('tables').get(table_item).get('selected_rows')
115        agg_funtion = panda_latex_tables_config.get('tables').get(table_item).get('agg_funtion')
116        agg_colums = panda_latex_tables_config.get('tables').get(table_item).get('agg_colums')
117        # dropping and renaming columns
118        drop_columns = panda_latex_tables_config.get('tables').get(table_item).get('drop_columns')
119        rename_columns = panda_latex_tables_config.get('tables').get(table_item).get('rename_columns')
120
121        # table filtering and sorting
122        where_clausel = panda_latex_tables_config.get('tables').get(table_item).get('where_clausel')
123        order_by = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('order_by')
124        sort_acending = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('sort_acending')
125        sort_inplace = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('sort_inplace')
126
127        # pivot settings
128        pivot = panda_latex_tables_config.get('tables').get(table_item).get('pivot')
129        pivot_column = panda_latex_tables_config.get('tables').get(table_item).get('pivot_columns')
130        pivot_value = panda_latex_tables_config.get('tables').get(table_item).get('pivot_values')
131

```

```

132     # pivot_table settings
133     pivot_table = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table')           pivot_table_column = panda_latex_tables_config.get('tables').get(table_item)
134     .get('pivot_table').get(                                         .get('pivot_table_column')
135         'pivot_columns')
136     pivot_table_value = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
137         'pivot_values')
138     pivot_table_agg_function = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
139         'pivot_agg_func')
140     pivot_table_indexes = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
141         'pivot_index').get('pivot_indexes')
142     pivot_table_indexes_visible = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
143         'pivot_index').get('pivot_indexes_visible')
144     pivot_table_rename_indexes = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
145         'pivot_index').get('pivot_rename_indexes')

146     # margins (subtotals)
147     margin = panda_latex_tables_config.get('tables').get(table_item).get('margins').get('margin')
148     margin_name = panda_latex_tables_config.get('tables').get(table_item).get('margins').get('margin_name')

149     # table settings
150     table_caption = panda_latex_tables_config.get('tables').get(table_item).get('caption')
151     table_label = panda_latex_tables_config.get('tables').get(table_item).get('label')
152     table_style = panda_latex_tables_config.get('tables').get(table_item).get('table_styles')
153     sparse_columns = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get(
154         'sparse_columns')
155     table_caption_position = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get(
156         'props').get('caption-side')
157     table_position = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
158         'position')
159     longtable = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
160         'longtable')
161     linebreak_columns = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
162         'linebreak_columns')
163     resize_textwidth = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
164         'resize_textwidth')

165     # get the pandas (panda data)
166     panda_table_data = get_data(startpath, destination, tablefilename, datafile_path, datafile, alternative_csv_load, separator, decimal)

167     # filter by where clause
168     if where_clause:
169         panda_table_data = panda_table_data.query(where_clause)

170     # Drop unused columns
171     if drop_columns:
172         panda_table_data = panda_table_data.drop(columns=drop_columns)

173     # set aggregation functions
174     # if groupby_values and not agg_funtion and not pivot_column and not pivot_table_column:
175     if groupby_values and not (pivot_column or (pivot_table_column or pivot_table_value or pivot_table_indexes)):
176         match group_by_function:
177             case 'max':
178                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).max()
179             case 'min':
180

```

```

185     panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).min()
186     case 'head':
187         panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).head()
188     case 'sum':
189         panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).sum()
190     case 'mean':
191         panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).mean()
192 else:
193     panda_table_data = panda_table_data
194
195 # pivot if pivot is selected
196 if pivot_table_column or pivot_table_value or pivot_table_indizes:
197     if type(pivot_table_agg_function) is list:
198         agg_tuple = tuple(pivot_table_agg_function)
199         panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
200                                         columns=pivot_table_column, values=pivot_table_value,
201                                         aggfunc=agg_tuple, margins=margin, margins_name=margin_name)
202     elif type(pivot_table_agg_function) is dict:
203         panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
204                                         columns=pivot_table_column,
205                                         values=pivot_table_value, aggfunc=pivot_table_agg_function,
206                                         margins=margin, margins_name=margin_name)
207     else:
208         panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
209                                         columns=pivot_table_column, values=pivot_table_value,
210                                         aggfunc=pivot_table_agg_function, margins=margin,
211                                         margins_name=margin_name)
212
213 # set column operations
214 if column_operations:
215     for column_ops in column_operations:
216         operation_function = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('operation_function')
217         operation_columns = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('columns')
218         operation_axis = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('axis_number')
219         match operation_function:
220             case 'max':
221                 panda_table_data[column_ops] = panda_table_data[operation_columns].max()
222             case 'min':
223                 panda_table_data[column_ops] = panda_table_data[operation_columns].min()
224             case 'head':
225                 panda_table_data[column_ops] = panda_table_data[operation_columns].head()
226             case 'sum':
227                 panda_table_data[column_ops] = panda_table_data[operation_columns].sum(axis=operation_axis)
228             case 'mean':
229                 panda_table_data[column_ops] = panda_table_data[operation_columns].mean()
230             case 'diff':
231                 panda_table_data[column_ops] = panda_table_data[operation_columns[1]] - panda_table_data[operation_columns[0]]
232
233 # order by
234 if order_by:
235     panda_table_data.sort_values(by=order_by, inplace=sort_inplace, ascending=sort_acending)
236
237 # rename columns
238 if rename_columns:

```

```

239     panda_table_data = panda_table_data.rename(columns=rename_columns)
240 # rename indices      if pivot_table_rename_indizes:
241     panda_table_data = panda_table_data.rename_axis(index=pivot_table_rename_indizes)
242
243 # frame carriage return columns in subtable
244 if linebreak_columns:
245     for lbr_column in linebreak_columns:
246         panda_table_data[lbr_column] = "\\\begin{tabular}[c]{@{}l@{}}" + panda_table_data[lbr_column].astype(str) + "\\end{tabular}"
247 # convert python panda to latex table
248 latex_table = panda_table_data.to_latex(header=True, bold_rows=False, longtable=longtable,
249                                         sparsify=sparse_columns, label=table_label, caption=table_caption,
250                                         position=table_position, na_rep=' ', index=pivot_table_indizes_visible)
251
252 # textwidth resize
253 if resize_textwidth:
254     with open(tablefile, 'w') as wrlt:
255         wrlt.write(latex_table)
256
257     with open(tablefile) as file:
258         lines = file.readlines()
259
260     # replace table with resize
261     resize_line_nr = 0
262     resize_line = ""
263     if longtable:
264         table_type = '\\begin{longtable}',
265     else:
266         table_type = '\\begin{table}'
267
268     for number, line in enumerate(lines, 1):
269     # for number, line in latex_table.splitlines():
270     # for number, line in latex_table.readlines():
271     # for number, line in latex_table.splitlines('\n'):
272     # for number, line in lines.split('\n'):
273
274         # Condition true if the key exists in the line
275         # If true then display the line number
276         if table_type in line:
277             # print(f'{key} is at line {number}')
278             resize_line_nr = number
279             resize_line = line
280
281     line_table_resize = resize_line + "\n" + "\\resizebox{\\columnwidth}{!}{"
282     latex_table = latex_table.replace(resize_line, line_table_resize)
283
284     # replace table end with bracket
285     resize_line_nr = 0
286     resize_line = ""
287     if longtable:
288         table_type = '\\end{longtable}',
289     else:
290         table_type = '\\end{table}'
291
292     for number, line in enumerate(lines, 1):

```

```

293
294     # Condition true if the key exists in the line                      # If true then display the line number
295     if table_type in line:
296         # print(f'{key} is at line {number}')
297         resize_line_nr = number
298         resize_line = line
299
300     line_table_resize = "}" + "\n" + resize_line
301     latex_table = latex_table.replace(resize_line, line_table_resize)
302
303     # caption below is not supported yet (pandas 2.2)
304     # replace caption and replace table end with the caption line and table end
305     if table_caption_position == 'below':
306         caption_label = "\caption{" + table_caption + "}" + "\label{" + table_label + "}" + "\\\""
307         caption_label_nbr = "\caption{" + table_caption + "}" + "\label{" + table_label + "}" + "\\" + "\\" + "\"
308         caption_only = "\caption{" + table_caption + "}" + "\\\""
309         caption_only_nbr = "\caption{" + table_caption + "}" + "\\" + "\"
310         label_only = "\label{" + table_label + "}" + "\\\""
311         label_only_nbr = "\label{" + table_label + "}" + "\\" + "\"
312         latex_table = latex_table.replace(caption_label, '')
313         latex_table = latex_table.replace(caption_only, '')
314         latex_table = latex_table.replace(label_only, '')
315         latex_table = latex_table.replace(caption_label_nbr, '')
316         latex_table = latex_table.replace(caption_only_nbr, '')
317         latex_table = latex_table.replace(label_only_nbr, '')
318
319     if longtable:
320         table_string = '\end{longtable}'
321         new_caption = caption_label_nbr + "\n" + table_string
322         latex_table = latex_table.replace(table_string, new_caption)
323     else:
324         table_string = '\end{table}'
325         new_caption = caption_label_nbr + "\n" + table_string
326         latex_table = latex_table.replace(table_string, new_caption)
327
328
329     # write latex table to filesystem
330     with open(tablefile, 'w') as wrlt:
331         wrlt.write(latex_table)
332
333
334 # run the methods / functions
335 panda_latex_tables_config = load_configuration('csv_to_latex_diplomarbeit.yaml')
336 create_latex_tables(panda_latex_tables_config)

```

Listing 146: Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle

XII.X csv_to_latex_diplomarbeit.yaml

```

1 tables_inventory:
2 - "db_inventory"
3 - "db_inventory_per_rdbms"
4 - "db_inventory_per_os"

```

```

5 - "anforderungskatalog"
6 - "arbeitsrapport"
7 - "projektcontrolling"
8 - "evaluation_inventory"
9 - "dependencis"
10 - "predecision_out"
11 - "predecision_in"
12 - "project_comments"
13 - "evaluation_distributed_sql"
14 - "expert_discussions_overview"
15 - "expert_discussions_full_list"
16 - "stakeholder"
17 tables:
18 db_inventory:
19   id: "db_inventory"
20   isbigfile:
21   has_longtexts: False
22   separator: ","
23   decimal: "."
24   caption: "Datenbankinventar - Roh"
25   label: "db_inventory"
26   startpath: "parentdir"
27   destination_path: "content/latex_tables"
28   datafile_path: "source/tables"
29   datafile: "inventory.csv"
30   tablefilename: "db_inventory.tex"
31   decimal_format:
32   group_by:
33   group_by_function:
34   agg_funtion:
35   agg_colums:
36   drop_columns:
37   - "comment"
38   - "eol"
39   - "eol_since"
40   - "releasedate"
41 column_operations:
42   datas:
43   operations:
44     dauer_summe:
45       operation_function:
46       axis_number:
47       columns:
48 pivot:
49   pivot_columns:
50   pivot_values:
51 pivot_table:
52   pivot_index:
53   pivot_indizes_visible:
54   pivot_rename_indizes:
55   pivot_columns:
56   pivot_values:
57   pivot_agg_func:
58   rename_columns:

```

```

59 server: "Server - Hostname"
60 os: "OS"
61 rdbms: "RDBMS"
62 instance: "Instanz"
63 databases: "Datenbanken"
64 appliance: "Appliance"
65 comment: "Kommentar"
66 version: "Version"
67 releasedate: "Version - Releasedatum"
68 eol: "EoL"
69 age: "Version - Alter"
70 eol_since: "EoL seit"
71 where_clause:
72 sorting:
73 order_by:
74   - "server"
75   - "rdbms"
76 sort_acending: True
77 sort_inplace: True
78 margins:
79   margin: False
80   margin_name:
81 table_styles:
82   selector: "caption"
83 props:
84   caption-side: "below"
85   position: "H"
86   sparse_columns: True
87   longtable: True
88   resize_textwidth: False
89   linebreak_columns:
90   table_header: True
91 db_inventory_per_rdbms:
92   id: "db_inventory_per_rdbms"
93   isbigfile:
94   has_longtexts: False
95   separator: ","
96   decimal: "."
97   caption: "Datenbankinventar"
98   label: "db_inventory_per_rdbms"
99   startpath: "parentdir"
100  destination_path: "content/latex_tables"
101  datafile_path: "source/tables"
102  datafile: "inventory.csv"
103  tablefilename: "db_inventory_per_rdbms.tex"
104  decimal_format:
105  group_by:
106    - "rdbms"
107  group_by_function: "sum"
108  agg_funtion:
109  agg_columns:
110    - "rdbms"
111  drop_columns:
112    - "server"

```

```

113 - "os"
114 - "version"
115 - "releasedate"
116 - "eol"
117 - "age"
118 - "eol_since"
119 - "comment"
120 column_operations:
121   datas:
122     operations:
123       dauer_summe:
124         operation_function:
125         axis_number:
126         columns:
127       pivot:
128         pivot_columns:
129         pivot_values:
130       pivot_table:
131         pivot_index:
132           pivot_indexes_visible:
133           pivot_rename_indexes:
134         pivot_columns:
135         pivot_values:
136         pivot_agg_func:
137       rename_columns:
138         rdbms: "RDBMS"
139         instance : "Instanz"
140         databases : "Datenbanken"
141         appliance: "Appliance"
142       where_clause:
143       sorting:
144         order_by:
145           - "rdbms"
146         sortAscending: True
147         sortInplace: True
148       margins:
149         margin: True
150         margin_name: "Gesamtergebnis"
151       table_styles:
152         selector: "caption"
153       props:
154         caption_side: "below"
155         position: "H"
156         sparse_columns: True
157         longtable: False
158         resize_textwidth: False
159         linebreak_columns:
160           table_header: True
161       db_inventory_per_os:
162         id: "db_inventory_per_os"
163         isbigfile:
164         has_longtexts: False
165         separator: ","
166         decimal: "."

```

```

167 caption: "Datenbankinventor - Nach Betriebssystemen üaufgeschlüsselt"
168 label: "db_inventory_per_os"
169 startpath: "parentdir"
170 destination_path: "content/latex_tables"
171 datafile_path: "source/tables"
172 datafile: "inventory.csv"
173 tablefilename: "db_inventory_per_os.tex"
174 decimal_format:
175 group_by:
176   - "rdbms"
177   - "os"
178 group_by_function: "sum"
179 agg_funtion:
180 agg_colums:
181   - "os"
182 drop_columns:
183   - "server"
184   - "version"
185   - "releasedate"
186   - "eol"
187   - "age"
188   - "eol_since"
189   - "comment"
190 #   - "appliance"
191 column_operations:
192   datas:
193     operations:
194       dauer_summe:
195         operation_function:
196         axis_number:
197         columns:
198       pivot:
199         pivot_columns:
200         pivot_values:
201       pivot_table:
202         pivot_index:
203           pivot_indizes:
204             - "os"
205             - "rdbms"
206         pivot_indizes_visible: True
207         pivot_rename_indizes:
208           os: "OS"
209           rdbms: "RDBMS"
210       pivot_columns:
211         pivot_values:
212         pivot_agg_func:
213           instance: "sum"
214           databases: "sum"
215           appliance: "sum"
216         transpose: True
217       rename_columns:
218         rdbms: "RDBMS"
219         instance : "Instanz"
220         databases : "Datenbanken"

```

```
221 os : "OS"
222 appliance: "Appliance"
223 where_clause:
224 sorting:
225 order_by:
226 sort_ascending: False
227 sort_inplace: True
228 margins:
229 margin: True
230 margin_name: "Gesamtergebnis"
231 table_styles:
232 selector: "caption"
233 props:
234 caption-side: "below"
235 position: "H"
236 sparse_columns: True
237 longtable: True
238 resize_textwidth: False
239 linebreak_columns:
240 table_header: True
241 anforderungskatalog:
242 id: "anforderungskatalog"
243 isbigfile:
244 has_longtexts: True
245 separator: ";"
246 decimal: "."
247 caption: "Anforderungskatalog"
248 label: "anforderungskatalog"
249 startpath: "parentdir"
250 destination_path: "content/latex_tables"
251 datafile_path: "source/tables"
252 datafile: "anforderungskatalog.CSV"
253 tablefilename: "anforderungskatalog.tex"
254 decimal_format:
255 group_by:
256 group_by_function:
257 agg_funtion:
258 agg_colums:
259 drop_columns:
260 column_operations:
261 datas:
262 operations:
263 dauer_summe:
264 operation_function:
265 axis_number:
266 columns:
267 pivot:
268 pivot_columns:
269 pivot_values:
270 pivot_table:
271 pivot_index:
272 pivot_indexes_visible: False
273 pivot_rename_indexes:
274 pivot_columns:
```

```

275 pivot_values:
276 pivot_agg_func:
277 rename_columns:
278 where_clausel:
279 sorting:
300 order_by:
301 - "Nr."
302 sort_acending: True
303 sort_inplace: True
304 margins:
305 margin: False
306 margin_name:
307 table_styles:
308 selector: "caption"
309 props:
310 caption-side: "below"
311 position: "H"
312 sparse_columns: False
313 longtable: False
314 resize_textwidth: True
315 linebreak_columns:
316 - "Beschreibung"
317 table_header: True
318 arbeitsrapport:
319 id: "arbeitsrapport"
320 isbigfile:
321 has_longtexts: False
322 separator: ";"
323 decimal: "."
324 caption: "Arbeitsrapport"
325 label: "arbeitsrapport"
326 startpath: "parentdir"
327 destination_path: "content/latex_tables"
328 datafile_path: "source/tables"
329 datafile: "arbeitsrapport.CSV"
330 tablefilename: "arbeitsrapport.tex"
331 decimal_format: "{:0.1f}"
332 group_by:
333 group_by_function:
334 agg_funtion:
335 agg_colums:
336 drop_columns:
337 - "Hide"
338 - "Geplante Dauer [h]"
339 - "dauer_summe"
340 column_operations:
341 datas:
342 operations:
343 dauer_summe:
344 operation_function:
345 axis_number:
346 columns:
347 pivot:
348 pivot_columns:

```

```

329 pivot_values:
330 pivot_table:
331 pivot_index:
332 pivot_indexes_visible: False
333 pivot_rename_indexes:
334 pivot_columns:
335 pivot_values:
336 pivot_agg_func:
337 rename_columns:
338 where_clause: "Hide == 0"
339 sorting:
340 order_by:
341 - "Datum"
342 - "Von"
343 sort_ascending: False
344 sort_inplace: False
345 margins:
346 margin: False
347 margin_name:
348 table_styles:
349 selector: "caption"
350 props:
351 caption-side: "below"
352 position: "H"
353 sparse_columns: True
354 longtable: False
355 resize_textwidth: True
356 linebreak_columns:
357 - "äTtigkeit"
358 - "Bemerkung"
359 - "Schwierigkeit"
360 - "öLsungen"
361 table_header: True
362 projektcontrolling:
363 id: "projektcontrolling"
364 isbigfile:
365 has_longtexts: False
366 separator: ";"
367 decimal: "."
368 caption: "Projektcontrolling"
369 label: "projektcontrolling"
370 startpath: "parentdir"
371 destination_path: "content/latex_tables"
372 datafile_path: "source/tables"
373 datafile: "arbeitsrapport.CSV"
374 tablefilename: "projektcontrolling.tex"
375 decimal_format: "{:0.1f}"
376 group_by:
377 - "Phase"
378 - "Subphase"
379 group_by_function: "sum"
380 agg_funtion:
381 agg_colums:
382 - "Dauer [h]"

```

```

383     - "Geplante Dauer [h]"
384     - "dauer_summe"
385 drop_columns:
386     - "Datum"
387     - "Von"
388     - "Bis"
389     - "Hide"
390     - "äTtigkeit"
391     - "Bemerkung"
392     - "Schwierigkeit"
393     - "öLsungen"
394 column_operations:
395     datas:
396         - "dauer_summe"
397 operations:
398     dauer_summe:
399         operation_function: "diff"
400         axis_number: 1
401     columns:
402         - "Dauer [h]"
403         - "Geplante Dauer [h]"
404 pivot:
405     pivot_columns:
406     pivot_values:
407 pivot_table:
408     pivot_index:
409         pivot_indizes_visible:
410         pivot_rename_indizes:
411     pivot_columns:
412     pivot_values:
413     pivot_agg_func:
414 rename_columns:
415     dauer_summe: "Verbleibende Zeit [h]"
416 where_clause:
417 sorting:
418     order_by:
419         - "Phase"
420         - "Subphase"
421     sortAscending: True
422     sortInplace: True
423 margins:
424     margin: True
425     margin_name: "Total"
426 table_styles:
427     selector: "caption"
428 props:
429     caption_side: "below"
430     position: "H"
431     sparse_columns: True
432     longtable: False
433     resize_textwidth: True
434     linebreak_columns:
435     table_header: True
436 evaluation_inventory:

```

```
437 id: "evaluation_inventory"
438 isbigfile:
439 has_longtexts: False
440 separator: ";"
441 decimal: "."
442 caption: "Evaluationssysteme"
443 label: "evaluation_inventory"
444 startpath: "parentdir"
445 destination_path: "content/latex_tables"
446 datafile_path: "source/tables"
447 datafile: "evaluation_platform_serverlist.csv"
448 tablefilename: "evaluation_inventory.tex"
449 decimal_format:
450 group_by:
451 group_by_function:
452 agg_funtion:
453 agg_colums:
454 drop_columns:
455 column_operations:
456 datas:
457 operations:
458 dauer_summe:
459 operation_function:
460 axis_number:
461 columns:
462 pivot:
463 pivot_columns:
464 pivot_values:
465 pivot_table:
466 pivot_index:
467 pivot_indizes_visible: False
468 pivot_rename_indizes:
469 pivot_columns:
470 pivot_values:
471 pivot_agg_func:
472 rename_columns:
473 where_clausel:
474 sorting:
475 order_by:
476 - "Server"
477 - "Typ"
478 sort_ascending: True
479 sort_inplace: True
480 margins:
481 margin: False
482 margin_name:
483 table_styles:
484 selector: "caption"
485 props:
486 caption-side: "below"
487 position: "H"
488 sparse_columns: True
489 longtable: True
490 resize_textwidth: False
```

```
491 linebreak_columns:  
492     table_header: True  
493 dependencis:  
494     id: "dependencis"  
495     isbigfile:  
496     has_longtexts: False  
497     separator: ";"  
498     decimal: "."  
499     caption: "ÄAbhngigkeiten"  
500     label: "dependencis"  
501     startpath: "parentdir"  
502     destination_path: "content/latex_tables"  
503     datafile_path: "source/tables"  
504     datafile: "dependencis.csv"  
505     tablefilename: "dependencis.tex"  
506     decimal_format:  
507     group_by:  
508     group_by_function:  
509     agg_funtion:  
510     agg_colums:  
511     drop_columns:  
512     column_operations:  
513     datas:  
514     operations:  
515     dauer_summe:  
516     operation_function:  
517     axis_number:  
518     columns:  
519 pivot:  
520     pivot_columns:  
521     pivot_values:  
522 pivot_table:  
523     pivot_index:  
524     pivot_indizes_visible: False  
525     pivot_rename_indizes:  
526 pivot_columns:  
527     pivot_values:  
528     pivot_agg_func:  
529 rename_columns:  
530 where_clausel:  
531 sorting:  
532     order_by:  
533     - "Nr."  
534     sort_acending: True  
535     sort_inplace: True  
536 margins:  
537     margin: False  
538     margin_name:  
539 table_styles:  
540     selector: "caption"  
541 props:  
542     caption-side: "below"  
543     position: "H"  
544     sparse_columns: True
```

```

545 longtable: False
546 resize_textwidth: True
547 linebreak_columns:
548 - "Abhngigkeit"
549 - "Beschreibung"
550 - "Status"
551 - "Risiko"
552 - "Impact"
553 table_header: True
554 predecision_out:
555 id: "predecision_out"
556 isbigfile:
557 has_longtexts: False
558 separator: ";"
559 decimal: "."
560 caption: "Vorauswahl - Ausgeschieden"
561 label: "predecision_out"
562 startpath: "parentdir"
563 destination_path: "content/latex_tables"
564 datafile_path: "source/tables"
565 datafile: "pre-decision.csv"
566 tablefilename: "pre-decision-out.tex"
567 decimal_format:
568 group_by:
569 group_by_function:
570 agg_funtion:
571 agg_colums:
572 drop_columns:
573 - "hide_state"
574 column_operations:
575 datas:
576 operations:
577 dauer_summe:
578 operation_function:
579 axis_number:
580 columns:
581 pivot:
582 pivot_columns:
583 pivot_values:
584 pivot_table:
585 pivot_index:
586 pivot_indizes_visible: False
587 pivot_rename_indizes:
588 pivot_columns:
589 pivot_values:
590 pivot_agg_func:
591 rename_columns:
592 where_clause1: "hide_state == 1"
593 sorting:
594 order_by:
595 - "Nr."
596 sort_acending: True
597 sort_inplace: True
598 margins:

```

```
599 margin: False
600 margin_name:
601 table_styles:
602 selector: "caption"
603 props:
604   caption-side: "below"
605   position: "H"
606   sparse_columns: True
607   longtable: False
608   resize_textwidth: True
609   linebreak_columns:
610   - "ÜBegründung"
611   table_header: True
612 predecision_in:
613 id: "predecision_in"
614 isbigfile:
615 has_longtexts: False
616 separator: ";"
617 decimal: "."
618 caption: "Vorauswahl - Evaluation"
619 label: "predecision_in"
620 startpath: "parentdir"
621 destination_path: "content/latex_tables"
622 datafile_path: "source/tables"
623 datafile: "pre-decision.csv"
624 tablefilename: "pre-decision-in.tex"
625 decimal_format:
626 group_by:
627 group_by_function:
628 agg_funtion:
629 agg_colums:
630 drop_columns:
631 - "hide_state"
632 column_operations:
633 datas:
634 operations:
635 dauer_summe:
636 operation_function:
637 axis_number:
638 columns:
639 pivot:
640 pivot_columns:
641 pivot_values:
642 pivot_table:
643 pivot_index:
644   pivot_indizes_visible: False
645   pivot_rename_indizes:
646 pivot_columns:
647 pivot_values:
648 pivot_agg_func:
649 rename_columns:
650 where_clause1: "hide_state == 2"
651 sorting:
652 order_by:
```

```
653 - "Nr."
654 sort_acending: True
655 sort_inplace: True
656 margins:
657 margin: False
658 margin_name:
659 table_styles:
660 selector: "caption"
661 props:
662 caption-side: "below"
663 position: "H"
664 sparse_columns: True
665 longtable: False
666 resize_textwidth: True
667 linebreak_columns:
668 - "ÜBegrndung"
669 table_header: True
670 project_comments:
671 id: "project_comments"
672 isbigfile:
673 has_longtexts: False
674 separator: ";"
675 decimal: "."
676 caption: "Kommentare - Anmerkung"
677 label: "project_comments"
678 startpath: "parentdir"
679 destination_path: "content/latex_tables"
680 datafile_path: "source/tables"
681 datafile: "pre-fazit.csv"
682 tablefilename: "pre-fazit.tex"
683 decimal_format:
684 group_by:
685 group_by_function:
686 agg_funtion:
687 agg_colums:
688 drop_columns:
689 column_operations:
690 datas:
691 operations:
692 dauer_summe:
693 operation_function:
694 axis_number:
695 columns:
696 pivot:
697 pivot_columns:
698 pivot_values:
699 pivot_table:
700 pivot_index:
701 pivot_indizes_visible: False
702 pivot_rename_indizes:
703 pivot_columns:
704 pivot_values:
705 pivot_agg_func:
706 rename_columns:
```

```

707 where_clause:
708 sorting:
709   order_by:
710     - "Woche"
711   sort_ascending: True
712   sort_inplace: True
713 margins:
714   margin: False
715   margin_name:
716 table_styles:
717   selector: "caption"
718 props:
719   caption-side: "below"
720   position: "H"
721   sparse_columns: True
722   longtable: False
723   resize_textwidth: True
724   linebreak_columns:
725     - "Beschreibung / Event / Problem"
726   table_header: True
727 evaluation_distributed_sql:
728   id: "evaluation_distributed_sql"
729   isbigfile:
730   has_longtexts: False
731   separator: ";"
732   decimal: "."
733   caption: "Evaluationssystem - Distributed SQL / Sharding"
734   label: "evaluation_distributed_sql"
735   startpath: "parentdir"
736   destination_path: "content/latex_tables"
737   datafile_path: "source/tables"
738   datafile: "evaluation_platform_distributed_sql.csv"
739   tablefilename: "evaluation_platform_distributed_sql.tex"
740   decimal_format:
741   group_by:
742   group_by_function:
743   agg_function:
744   agg_columns:
745   drop_columns:
746   column_operations:
747   datas:
748   operations:
749     dauer_summe:
750     operation_function:
751     axis_number:
752     columns:
753   pivot:
754     pivot_columns:
755     pivot_values:
756   pivot_table:
757     pivot_index:
758     pivot_indexes_visible: False
759     pivot_rename_indexes:
760     pivot_columns:

```

```

761 pivot_values:
762 pivot_agg_func:
763 rename_columns:
764 where_clause:
765 sorting:
766 order_by:
767 sort_ascending: False
768 sort_inplace: False
769 margins:
770 margin: False
771 margin_name:
772 table_styles:
773 selector: "caption"
774 props:
775 caption-side: "below"
776 position: "H"
777 sparse_columns: True
778 longtable: False
779 resize_textwidth: False
780 linebreak_columns:
781 table_header: False
782 expert_discussions_overview:
783 id: "expert_discussions_overview"
784 isbigfile:
785 has_longtexts: False
786 separator: ";"
787 decimal: "."
788 caption: "Fachgespräche"
789 label: "expert_discussions_overview"
790 startpath: "parentdir"
791 destination_path: "content/latex_tables"
792 datafile_path: "source/tables"
793 datafile: "expert_discussions.csv"
794 tablefilename: "expert_discussions_overview.tex"
795 decimal_format:
796 group_by:
797 group_by_function:
798 agg_function:
799 agg_columns:
800 drop_columns:
801 - "Fragen"
802 - "Antworten"
803 - "Sonstige Themen"
804 column_operations:
805 datas:
806 operations:
807 dauer_summe:
808 operation_function:
809 axis_number:
810 columns:
811 pivot:
812 pivot_columns:
813 pivot_values:
814 pivot_table:

```

```
815 pivot_index:  
816     pivot_indexes_visible: False  
817 pivot_rename_indexes:  
818 pivot_columns:  
819 pivot_values:  
820 pivot_agg_func:  
821 rename_columns:  
822 where_clausel:  
823 sorting:  
824     order_by:  
825         - "äFachgesprch"  
826 sort_acending: True  
827 sort_inplace: True  
828 margins:  
829     margin: False  
830     margin_name:  
831 table_styles:  
832     selector: "caption"  
833 props:  
834     caption-side: "below"  
835     position: "H"  
836     sparse_columns: True  
837     longtable: False  
838     resize_textwidth: True  
839     linebreak_columns:  
840         - "Studenten"  
841         - "Bemerkungen"  
842     table_header: True  
843 expert_discussions_full_list:  
844     id: "expert_discussions_full_list"  
845     isbigfile:  
846     has_longtexts: False  
847     separator: ";"  
848     decimal: "."  
849     caption: "äFachgesprche - Protokoll"  
850     label: "expert_discussions_full_list"  
851     startpath: "parentdir"  
852     destination_path: "content/latex_tables"  
853     datafile_path: "source/tables"  
854     datafile: "expert_discussions.csv"  
855     tablefilename: "expert_discussions_full_list.tex"  
856     decimal_format:  
857     group_by:  
858     group_by_function:  
859     agg_funtion:  
860     agg_colums:  
861     drop_columns:  
862     column_operations:  
863     datas:  
864     operations:  
865     dauer_summe:  
866     operation_function:  
867     axis_number:  
868     columns:
```

```
869 pivot:  
870   pivot_columns:  
871   pivot_values:  
872   pivot_table:  
873     pivot_index:  
874       pivot_indexes_visible: False  
875       pivot_rename_indexes:  
876     pivot_columns:  
877     pivot_values:  
878     pivot_agg_func:  
879     rename_columns:  
880     where_clause:  
881     sorting:  
882       order_by:  
883         - "äFachgespräch"  
884       sort_ascending: True  
885       sort_inplace: True  
886     margins:  
887       margin: False  
888       margin_name:  
889     table_styles:  
890       selector: "caption"  
891     props:  
892       caption_side: "below"  
893       position: "H"  
894       sparse_columns: True  
895       longtable: False  
896       resize_textwidth: True  
897     linebreak_columns:  
898       - "Studenten"  
899       - "Fragen"  
900       - "Antworten"  
901       - "Sonstige Themen"  
902       - "Bemerkungen"  
903     table_header: True  
904 stakeholder:  
905   id: "stakeholder"  
906   isbigfile:  
907   has_longtexts: False  
908   separator: ";"  
909   decimal: "."  
910   caption: "Stakeholder"  
911   label: "stakeholder"  
912   startpath: "parentdir"  
913   destination_path: "content/latex_tables"  
914   datafile_path: "source/tables"  
915   datafile: "stakeholder.csv"  
916   tablefilename: "stakeholder.tex"  
917   decimal_format:  
918   group_by:  
919   group_by_function:  
920   agg_function:  
921   agg_columns:  
922   drop_columns:
```

```

923 column_operations:
924   datas:
925   operations:
926     dauer_summe:
927       operation_function:
928       axis_number:
929       columns:
930   pivot:
931     pivot_columns:
932     pivot_values:
933   pivot_table:
934     pivot_index:
935     pivot_indizes_visible: False
936     pivot_rename_indizes:
937   pivot_columns:
938   pivot_values:
939   pivot_agg_func:
940   rename_columns:
941   where_clausel:
942   sorting:
943     order_by:
944     sort_acending: True
945     sort_inplace: True
946   margins:
947     margin: False
948     margin_name:
949   table_styles:
950     selector: "caption"
951   props:
952     caption-side: "below"
953     position: "H"
954     sparse_columns: True
955     longtable: False
956     resize_textwidth: True
957     linebreak_columns:
958     table_header: True

```

Listing 147: Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle

XII.XI pandas_data_chart_plotter.py

```

1 import os
2 from pathlib import Path
3 import chardet
4 import pandas as pd
5 import yaml
6
7
8 def load_configuration(panda_diagram_plotter_conf_filename):
9   panda_diagram_plotter_config = dict()
10
11   riskmatrix_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')
12   yaml_path = os.path.join(riskmatrix_conf_dir, panda_diagram_plotter_conf_filename)

```

```

13
14     with open(yaml_path, "r") as file:      panda_diagram_plotter_config = yaml.load(file, Loader=yaml.FullLoader)
15
16     return panda_diagram_plotter_config
17
18 def get_data(startpath, destination, tablefilename, datafile_path, datafile, separator, decimal):
19     # Config Variables
20     if startpath == 'homedir':
21         directory = os.path.join(os.getcwd(), datafile_path)
22     else: # parentdir
23         directory = os.path.join(os.path.dirname(os.getcwd()), datafile_path)
24
25     # get the Data as dict
26     data_path = os.path.join(directory, datafile)
27
28     # load datas from csv into dict
29     detected = chardet.detect(Path(data_path).read_bytes())
30     encoding = detected.get("encoding")
31
32     print(datafile, ':', encoding)
33     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
34     # return data
35     return panda_table_data
36 def create_panda_diagram_plotter(panda_diagram_plotter_config):
37     pdp_tables = panda_diagram_plotter_config.get('diagram_inventory')
38     for table_item in pdp_tables:
39         print(table_item)
40         startpath = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('startpath')
41         destination = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('destination_path')
42         imagename = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('imagename')
43         datafile_path = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('datafile_path')
44         datafile = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('datafile')
45         if startpath == 'homedir':
46             directory = os.path.join(os.getcwd(), destination)
47         else: # parentdir
48             directory = os.path.join(os.path.dirname(os.getcwd()), destination)
49         image_path = os.path.join(directory, imagename)
50         separator = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('separator')
51         decimal = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('decimal')
52
53         # column operations
54         column_operations = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('datas')
55
56         # group by / aggregation
57         groupby_values = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('group_by')
58         group_by_function = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('group_by_function')
59
60         agg_funtion = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('agg_funtion')
61         agg_colums = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('agg_colums')
62         # dropping and renaming columns
63         drop_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('drop_columns')
64         rename_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('rename_columns')
65
66         # table filtering and sorting

```

```

67     where_clause1 = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('where_clause1')
68     order_by = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('sorting').get('order_by')           sort_acending = panda_diagram_plotter_config.get
69     ('panda_diagram_plotter').get(table_item).get('sorting').get('sort_acending')
70     sort_inplace = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('sorting').get('sort_inplace')
71
72     # pivot settings
73     pivot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot')
74     pivot_column = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_columns')
75     pivot_value = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_values')
76
77     # pivot_table settings
78     pivot_table = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table')
79     pivot_table_column = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
80         'pivot_columns')
81     pivot_table_value = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
82         'pivot_values')
83     pivot_table_agg_function = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
84         'pivot_agg_func')
85     pivot_table_indexes = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
86         'pivot_index').get('pivot_indexes')
87     pivot_table_indexes_visible = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
88         'pivot_index').get('pivot_indexes_visible')
89     pivot_table_rename_indexes = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
90         'pivot_index').get('pivot_rename_indexes')
91
92     # margins (subtotals)
93     margin = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('margins').get('margin')
94     margin_name = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('margins').get('margin_name')
95
96     # chart settings
97     chart = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-kind')
98     title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('title')
99     x_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('x-axis-title')
100    y_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('y-axis-title')
101    x_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('x-axis-columns')
102    y_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('y-axis-columns')
103    index = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-index')
104
105    # chart styles
106    grid = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('grid')
107    legend = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('legend')
108    rot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('rot')
109    fontsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('fontsize')
110    figsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('figsize')
111    stacked = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('stacked')
112    secondary_y = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('secondary_y')
113    stylelist = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('stylelist')
114    subplots = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('subplots')
115    autopct = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('autopct')
116    loc = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('loc')
117    bbox_to_anchor = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('bbox_to_anchor')
118
119    # get the pandas (panda data)
120    panda_table_data = get_data(startpath, destination, imagename, datafile_path, datafile, separator, decimal)

```

```

120
121     # filter by where clause           if where_clause:
122         panda_table_data = panda_table_data.query(where_clause)
123
124     # Drop unused columns
125     if drop_columns:
126         panda_table_data = panda_table_data.drop(columns=drop_columns)
127
128     # set aggregation functions
129     # if groupby_values and not agg_funtion and not pivot_column and not pivot_table_column:
130     if groupby_values and not (pivot_column or (pivot_table_column or pivot_table_value or pivot_table_indexes)):
131         match group_by_function:
132             case 'max':
133                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).max()
134             case 'min':
135                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).min()
136             case 'head':
137                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).head()
138             case 'sum':
139                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).sum()
140             case 'mean':
141                 panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).mean()
142     else:
143         panda_table_data = panda_table_data
144
145     # pivot if pivot is selected
146     if pivot_table_column or pivot_table_value or pivot_table_indexes:
147         if type(pivot_table_agg_function) is list:
148             agg_tuple = tuple(pivot_table_agg_function)
149             panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indexes,
150                                              columns=pivot_table_column, values=pivot_table_value,
151                                              aggfunc=agg_tuple, margins=margin, margins_name=margin_name)
152         elif type(pivot_table_agg_function) is dict:
153             panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indexes,
154                                              columns=pivot_table_column,
155                                              values=pivot_table_value, aggfunc=pivot_table_agg_function,
156                                              margins=margin, margins_name=margin_name)
157         else:
158             panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indexes,
159                                              columns=pivot_table_column, values=pivot_table_value,
160                                              aggfunc=pivot_table_agg_function, margins=margin,
161                                              margins_name=margin_name)
162
163     # set column operations
164     if column_operations:
165         for column_ops in column_operations:
166             operation_function = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('operations').get(column_ops).get(
167             operation_function)
168             operation_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('operations').get(column_ops).get(
169             columns)
170             operation_axis = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('operations').get(column_ops).get(
171             axis_number)
172             match operation_function:
173                 case 'max':

```

```

171     panda_table_data[column_ops] = panda_table_data[operation_columns].max()
172     case 'min':
173         panda_table_data[column_ops] = panda_table_data[operation_columns].min()
174     case 'head':
175         panda_table_data[column_ops] = panda_table_data[operation_columns].head()
176     case 'sum':
177         panda_table_data[column_ops] = panda_table_data[operation_columns].sum(axis=operation_axis)
178     case 'mean':
179         panda_table_data[column_ops] = panda_table_data[operation_columns].mean()
180     case 'diff':
181         panda_table_data[column_ops] = panda_table_data[operation_columns[1]] - panda_table_data[operation_columns[0]]
182
183     # order by
184     if order_by:
185         panda_table_data.sort_values(by=order_by, inplace=sort_inplace, ascending=sort_acending)
186
187     # rename columns
188     if rename_columns:
189         panda_table_data = panda_table_data.rename(columns=rename_columns)
190
191     # set indices
192     if index:
193         index_values = panda_table_data.get(index)
194         #panda_table_data.set_index(index_values)
195         panda_table_data = panda_table_data.set_index(index)
196
197     # rename indices
198     if pivot_table_rename_indizes:
199         panda_table_data = panda_table_data.rename_axis(index=pivot_table_rename_indizes)
200
201     # Plotter
202     # Plotter Process starts here!
203     if autopct:
204         panda_chart_plot = panda_table_data.plot(kind=chart, title=title, y=y_axis_columns, x=x_axis_columns, xlabel=x_axis_title,
205                                                 ylabel=y_axis_title, grid=grid, stacked=stacked, legend=legend,
206                                                 secondary_y=secondary_y, subplots=subplots, rot=rot, fontsize=fontsize,
207                                                 figsize=figsize, autopct=autopct)
208     else:
209         panda_chart_plot = panda_table_data.plot(kind=chart, title=title, y=y_axis_columns, x=x_axis_columns, xlabel=x_axis_title,
210                                                 ylabel=y_axis_title, grid=grid, stacked=stacked, legend=legend,
211                                                 secondary_y=secondary_y, subplots=subplots, rot=rot, fontsize=fontsize,
212                                                 figsize=figsize)
213
214     match chart:
215         case 'pie':
216             panda_chart_plot[0].legend(loc=loc, bbox_to_anchor=bbox_to_anchor)
217             plt = panda_chart_plot[0].get_figure()
218             plt.savefig(image_path, bbox_inches='tight')
219         case _:
220             plt = panda_chart_plot.get_figure()
221             plt.savefig(image_path, bbox_inches='tight')
222
223
224     return "blade runner"

```

```
225 panda_diagram_plotter_config = load_configuration('pandas_data_chart_plotter_conf.yaml')
226 create_panda_diagram_plotter(panda_diagram_plotter_config)
```

Listing 148: Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm

XII.XII pandas_data_chart_plotter_conf.yaml

```
1 diagram_inventory:
2   - "tps_mixed"
3   - "db_inventory_per_rdbms"
4   - "db_inventory_per_os"
5 panda_diagram_plotter:
6   tps_mixed:
7     id: "tps_mixed"
8     startpath: "parentdir"
9     destination_path: "source/pandas_data_chart_plotter"
10    imagename: "tps_mixed.png"
11    datafile_path: "source/pandas_data_chart_plotter"
12    datafile: "tps_evaluation.csv"
13    separator: ","
14    decimal: "."
15    x-axis-columns: "Varianten"
16    y-axis-columns:
17      - "2. Iteration"
18      - "3. Iteration"
19      - "4. Iteration"
20    x-axis-title: "Varianten"
21    y-axis-title: "Transaktionen pro Sekunde (tps) Bsp."
22    title: "Transaktionen pro Sekunden - mixed"
23    chart-index:
24      chart-kind: "bar"
25      chart-designs:
26        subplots: False
27        grid: True
28        legend: True
29        rot:
30        fontsize:
31        stacked: False
32        secondary_y: False
33        stylelist:
34        figsize:
35        autopct:
36        loc:
37        bbox_to_anchor:
38      group_by:
39      group_by_function:
40      agg_funtion:
41      agg_columns:
42      drop_columns:
43        - "tps_1_iteration"
44        - "tps_typ"
45      column_operations:
46        datas:
```

```

47 pivot:
48   pivot_columns:
49   pivot_values:
50 pivot_table:
51   pivot_index:
52     pivot_indexizes_visible: False
53   pivot_rename_indexizes:
54   pivot_columns:
55   pivot_values:
56   pivot_agg_func:
57 rename_columns:
58   variante: "Varianten"
59   tps_2_iteration: "2. Iteration"
60   tps_3_iteration: "3. Iteration"
61   tps_4_iteration: "4. Iteration"
62 where_clause1: "tps_typ == 'mixed'"
63 sorting:
64   order_by:
65     sort_ascending: True
66     sort_inplace: True
67 margins:
68   margin: False
69   margin_name:
70 db_inventory_per_rdbms:
71   id: "db_inventory_per_rdbms"
72   startpath: "parentdir"
73   destination_path: "source/pandas_data_chart_plotter"
74   imagename: "db_inventory_per_rdbms.png"
75   datafile_path: "source/tables"
76   datafile: "inventory.csv"
77   separator: ","
78   decimal: "."
79   x-axis-columns: "RDBMS"
80   y-axis-columns:
81   x-axis-title:
82   y-axis-title:
83   title: "Datenbankinventor - Pro RDBMS"
84   chart-index: "RDBMS"
85   chart-kind: "pie"
86   chart-designs:
87     subplots: True
88     grid: False
89     legend: True
90     rot:
91     fontsize:
92     stacked: False
93     secondary_y: False
94     stylelist:
95     figsize: !!python/tuple [25,10]
96     autopct: '%1.0f%'
97     loc: "best"
98     bbox_to_anchor:
99 group_by:
100  - "rdbms"

```

```

101 group_by_function: "sum"
102 agg_funtion:
103 agg_columns:
104   - "rdbms"
105 drop_columns:
106   - "server"
107   - "os"
108   - "version"
109   - "releasedate"
110   - "eol"
111   - "age"
112   - "eol_since"
113   - "comment"
114   - "appliance"
115 column_operations:
116   datas:
117 pivot:
118   pivot_columns:
119   pivot_values:
120 pivot_table:
121   pivot_index:
122   pivot_indexizes_visible: False
123   pivot_rename_indexizes:
124   pivot_columns:
125   pivot_values:
126   pivot_agg_func:
127 rename_columns:
128   rdbms: "RDBMS"
129   instance : "Instanz"
130   databases : "Datenbanken"
131   appliance: "Appliance"
132 where_clause:
133 sorting:
134   order_by:
135     - "rdbms"
136   sort_acending: True
137   sort_inplace: True
138 margins:
139   margin: False
140   margin_name:
141 db_inventory_per_os:
142   id: "db_inventory_per_os"
143   separator: ","
144   decimal: "."
145   startpath: "parentdir"
146   destination_path: "source/pandas_data_chart_plotter"
147   datafile_path: "source/tables"
148   datafile: "inventory.csv"
149   imagename: "db_inventory_per_os.png"
150   decimal_format:
151   x-axis-columns: "RDBMS"
152   y-axis-columns:
153   x-axis-title:
154   y-axis-title:

```

```

155 title: "Datenbankinventor - Pro OS"
156 chart-index:
157 chart-kind: "pie"
158 chart-designs:
159     subplots: True
160     grid: False
161     legend: False
162     rot:
163     fontsize:
164     stacked: False
165     secondary_y: False
166     stylelist:
167     figsize: !!python/tuple [25,10]
168     autopct: '%1.0f%%'
169     loc: "upper center"
170     bbox_to_anchor: !!python/tuple [0,0]
171 group_by:
172     - "rdbms"
173     - "os"
174 group_by_function: "sum"
175 agg_funtion:
176 agg_columns:
177     - "os"
178 drop_columns:
179     - "server"
180     - "version"
181     - "releasedate"
182     - "eol"
183     - "age"
184     - "eol_since"
185     - "comment"
186 #     - "appliance"
187 column_operations:
188     datas:
189     operations:
190         dauer_summe:
191             operation_function:
192             axis_number:
193             columns:
194         pivot:
195             pivot_columns:
196             pivot_values:
197             pivot_table:
198             pivot_index:
199             pivot_indizes:
200                 - "os"
201                 - "rdbms"
202             pivot_indizes_visible: True
203             pivot_rename_indizes:
204                 os: "OS"
205                 rdbms: "RDBMS"
206             pivot_columns:
207             pivot_values:
208             pivot_agg_func:

```

```
209     instance: "sum"
210     databases: "sum"
211     appliance: "sum"
212     transpose: True
213     rename_columns:
214       rdbms: "RDBMS"
215       instance : "Instanz"
216       databases : "Datenbanken"
217       os : "OS"
218       appliance: "Appliance"
219     where_clause:
220     sorting:
221       order_by:
222         sort_acending: False
223         sort_inplace: True
224     margins:
225       margin: False
226       margin_name:
227     table_styles:
228       selector: "caption"
229     props:
230       caption-side: "below"
231       position: "H"
232       sparse_columns: True
233       longtable: True
234       resize_textwidth: False
235       linebreak_columns:
236       table_header: True
```

Listing 149: Python LaTex - pandas_data_chart_plotter_conf.yaml