ibW Höhere Fachschule Südostschweiz

Diplomarbeit Technik und Wirtschaftsinformatik 2023-2024

Titel der Arbeit: PostgreSQL HA Cluster - Konzeption und Implementation
Name: Graber
Vorname: Michael
Klasse: DIPL. INFORMATIKER/-IN HF - 10.0002A-2021
Firma: Kantonsspital Graubünden

Zusammenfassung

Disposition für die Diplomarbeit von Michael Graber. Ziel der Arbeit ist die Evaluation, Konzeption und Implementation eines PostgreSQL HA Clusters für das Kantonsspital Graubünden.

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 Ausgangslange . . . 6

1.1.5 Problemstellung . . . 10

1.2 Zieldefinition . . . 14

1.3 Abgrenzungen . . . 17

1.4 Abhängigkeiten . . . 19

2 Projektmanagement **21**

2.1 Risikomanagement . . . 22

2.1.1 Riskcontrolling . . . 24

2.2 Vorgehensweise und Methoden . . . 27

2.3 Projektplanung . . . 27

2.3.1 Projektcontrolling . . . 28

2.3.2 GANTT-Diagramm . . . 30

2.4 Expertengespräche . . . 32

3 Umsetzung **33**

3.1 Evaluation . . . 33

3.1.1 Exkurs Architektur . . . 33

3.1.2 Erheben und Gewichten der Anforderungen . . . 39

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 . . . 90

3.1.9 Gegenüberstellung der Lösungen . . . 92

3.1.10 Entscheid . . . 103

3.2 Aufbau und Implementation Testsystem . . . 103

3.2.1 Bereitstellen der Grundinfrastruktur . . . 103

3.2.2 Installation und Konfiguration PostgreSQL HA Cluster . . . 103

3.2.3 Technical Review der Umgebung . . . 103
3.3 Testing . . . 103
3.3.1 Testing . . . 103
3.3.2 Protokollierung . . . 103
3.3.3 Review und Auswertung . . . 104
3.4 Troubleshooting und Lösungsfindung . . . 104

4 Resultate **105**
4.1 Zielüberprüfung . . . 105
4.2 Schlussfolgerung . . . 105
4.3 Weiteres Vorgehen / offene Arbeiten . . . 105
4.4 Persönliches Fazit . . . 105

Abbildungsverzeichnis **106**

Tabellenverzeichnis **109**

Listings **110**

Literatur **114**

Abkürzungen **120**

Glossar **122**

Anhang **i**
I Arbeitsrapport . . . i
II Protokoll - Fachgespräche . . . ii
III Statusbericht . . . iii
III.I Status Report 1 . . . iii
III.II Status Report 2 . . . iii
IV Kommentare / Anmerkungen . . . iv
V Evaluation . . . vi
V.I Maintenance - CloudNativePG . . . vi
V.II Maintenance - Patroni . . . ix
V.III Maintenance - StackGres - Citus . . . xii
V.IV Maintenance - YugabyteDB . . . xix
VI Evaluationssysteme - Installation . . . xxii
VI.I rke2 . . . xxii
VI.II yugabyteDB . . . xxx
VI.III sks9016 - yugabyteDB . . . lx
VI.IV Stackgres mit Citus . . . lxi

VI.V Patroni lxxxv
VII Evaluationssysteme - Benchmarking lxxxvi
VII.I yugabyteDB lxxxvi
VII.II StackGres - Citus lxxxix
VIII Evaluationssysteme - Testing xci
VIII.I StackGres - Citus xci
VIII.II YugabyteDB xciii
IX Exkurs Architekturen - Umsysteme und Prinzipien xcvi
IX.I Raft-Konsensus xcvi
IX.II local-path-provisioner xcvi
X Python Utils xcvi
X.I zotero.py xcvi
X.II zotero_bibtex_configuration.yaml cii
X.III zotero_biblatex_keystore.yaml cii
X.IV riskmatrix.py cix
X.V riskmatrix_plotter_conf.yaml cxiii
X.VI riskmatrix_xy_axis_tuple_matrix.yaml cxvii
X.VII cost_benefit_diagram.py cxviii
X.VIII cost_benefit_diagram_plotter_conf.yaml cxx
X.IX pandas_dataframe_to_latex_table.py cxxi
X.X csv_to_latex_diplomarbeit.yaml cxxx
X.XI pandas_data_chart_plotter.py cl
X.XII pandas_data_chart_plotter_conf.yaml clvii

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[57]

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[83]

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[51]

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.

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.

Entsprechen 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.

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 Ausgangslange

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*[48] der Top-Datenbanksysteme . Allerdings werden nicht alle Datenbanksysteme berücksichtigt, entweder weil das Datenbanksystem keine Client/Server Architektur hat oder nicht im Scope der IT oder des Projekts ist.

Folgende Datenbanken sind inventarisiert:

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

Tabelle 1.1: Inventarisierte Datenbanksysteme

Folgende Datenbanksysteme sind demnach im KSGR im Einsatz:

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

Tabelle 1.2: Datenbankinventar

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

Continued on next page

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

OS	RDBMS	Appliance	Datenbanken	Instanz
	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						Abschätzung		Behandlung
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen ergreifen?
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	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja
2	A	G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung.	4	4	Ja
3	C, I, A	G0.26	Fehlfunktion von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Fehlfunktionen erleiden. Allerdings versuchen Datenbanksysteme, die Auswirkungen so gering wie möglich zu halten.	Fehlfunktionen können innerhalb von Datenbanksystemen die Datenkonsistenz verletzen, Daten können verloren gehen oder ungewollt von Dritten und unberechtigten Personen eingesehen werden. Systeme könnten nicht mehr oder nur noch eingeschränkt verfügbar werden. Daher sind sowohl Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	2	4	Ja
4	C, I, A	G0.27-1	Ressourcenmangel (personelle Ressourcen)	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. 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, das die Leute erreichbar sind, die notwendig sind	Die Auswirkungen können vielfältig sein, abhängig davon welcher Aspekt unter dem Ressourcenmangel leidet. Grundsätzlich wird aber sowohl die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	3	3	Ja
5	A	G0.27-2	Ressourcenmangel (technische Ressourcen)	Kann auftreten wenn Ressorucenwachstum zu spät bemerkt wird. So kann die CPU Usage oder das Memory Usage schnell anwachsen. Auch der Storage eines Betriebssystems kann nicht mehr ausreichend für ein System werden.	Wenn die CPU- und Memory-Usage über einen gewissen Schwellwert geht, fängt das Betriebssystem an zu Priorisieren. Dies wird primär der Endanwender in form von Performance Einbussen bemerken. Im schlimmsten Fall steht eine Anwendung nicht mehr zur Verfügung. Gefährlicher sind Storage Overflows, besonders wenn die Datenbank nicht mehr alle Informationen schreiben konnte, die sie für einen korrekten Neustart benötigte.	2	2	Ja
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 Fehlerhafter Administration und Konfiguration.	Doch die folgen bleiben nichtsdesto trotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen. Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist.	4	3	Ja
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, das in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen. Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen Wissendlich oder Unwissendlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen.	Der Wissentliche oder Unwissentliche Missbrauch von Berechtigungen kann verheerende Auswirkungen haben. Unter anderem können Daten missbräuchlich abgezogen werden, Daten manipuliert oder das ganze System komplett zerstört werden.	2	4	Ja
8	A, I	G0.45	Datenverlust	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, das alle Transaktionen die zwischen dem letzten Backup nicht mehr vorhanden sind. Hinzu kommt, das für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde. Ein weiterer Aspekt des Risikos besteht in der tatsache, das aufgrund der grossen Anzahl Datenbanken und deren Heterogenität nur wenige Backups auch wirklich regelmässig geprüft werden.	Aus dem Risiko ergeben sich zwei Auswirkungen, die aber beide ein hohes Mass an Schaden verursachen können. Erstens könnten Backups gar nicht mehr Wiederhergestellt werden, dies hätte dann einen Totalen Datenverlust zur Folge. Die zweite Ursache erwächst auf der fehlenden WAL-Archivierung, dadurch können zwar die Daten bis zu einem Zeitpunkt X Wiederhergestellt werden allerdings sind diese dann nicht zwingend Konsistent.	4	5	Ja

Tabelle 1.4: Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken

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

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

Mit diesen Massnahmen lassen sich die Risiken senden.
Die Risiken werden wie folgt gesenkt:

Abbildung 1.8: Risikomanagement PostgreSQL

1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [67, 73] 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.	Hoch
2	Failover Restore	Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den Primären Node zurückgesetzt werden. 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	Hoch
3	Filesystem Management	Die Datenmenge von Datenbanken wachsen in der Regel beständig. Die Belegung von Tablespaces 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	Hoch
4	Monitoring	Nebst den allgemeinen Metriken wie CPU / Memory Usage und der Port Verfügbarkeit gibt es noch eine reihe weiterer Aspekte die Überwacht werden müssen. Zum Beispiel ob es zu verzögerungen bei der Replikation kommt oder die Tablespaces genügend Platz haben. Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall.	Mittel
5	Statistiken / Cleanup Jobs justieren	PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. Zudem wird im Rahmen des gleichen Sheduled Tasks ein Cleanup Vorgenommen, 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 inperfomante SQL Statements ausgelesen werden und zum Teil werden auch informationen zum Tuning geliefert[46]. 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 aufgebläht 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 Bedürfnis 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.	1	Muss	
3	Filesystem Management	Hoch	Das Filesystem muss beim erreichen von 95% Usage automatisiert vergrössert werden. 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	2	Muss	
6	SQL optimierungen	Tief	Es gibt SQL Abfragen, mit dem fehlende Indizes ermittelt werden können. Diese Indizes sollen automatisiert erstellt werden. 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[46].	2	Kann	
7	Health Checks und Aktionen (Maintenance)	Hoch	Tabellen und Indizes können sich aufblähen (bloaded table / bloaded index) 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 aufzeigen, 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 evaluirt 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 evaluirt 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, bloaded 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 bloaded Indices, also aufgeblähte Indizes, automatisiert erkennen und mittels REINDEX bereinigen	Hoch
9	Housekeeping - Log Rotation	Die Log Rotation muss aktiviert werden. Die Logs müssen aber auch in das KSGR-Log Repository geschrieben werden	Hoch
10	User Management - Monitoring	Nicht verwendete User sollen einmal pro Woche automatisiert erkannt und in einem Report gemeldet werden.	Tief
11	Evaluationsziel	Am Ende der Evaluationsphase muss ein Entscheid getroffen worden sein, welche Methode verwendet wird.	Hoch
12	Installationsziel	Die Testinstallation muss Lauffähig sein und zudem alle Anforderungen und Ziele (3 und 4) erfüllen	Hoch
13	Testziele	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

Tabelle 1.7: Ziele

1.3 Abgrenzungen

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 Enterpise Linux (RHEL))	RedHat Enterprise Linux (RedHat Enterpise 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[98] folgende Abgrenzungen:

Abbildung 1.9: Systemabgrenzung

1.4 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	Ein Log Management System / SIEM muss vorhanden sein, um Logs langfristig sichern zu können.	Log Management und das SIAM werden abgelöst. Die Ausschreibung ist erfolgt	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 dass 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 Teils Sicherungen 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 offiiziellen 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

2 Projektmanagement

2.1 Risikomanagement

Aus den Abhängigkeiten heraus wurden folgende Risiken identifiziert:

Identifikation				Abschätzung		Behandlung			
						Massnahmen ergreifen?	Zielwert		Massnahme
ID	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM		WS	SM	
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	Schwächen beim Selbstmanagement und in der Selbstorganisation	Selbstmanagement und Organisation ist nicht meine Stärke	Das Projekt verzettelt sich, Zeit geht verloren. Auch eine folge könnte der Scope Verlust sein	3	3	Ja	2	2	Werkzeuge im Vorfeld definieren und bereitstellen
5	Scope verlust während des Projekts	Der Scope kann während des Projekts verloren gehen	Verzettelung und Zeitverlust bis hin zu scheitern	3	4	Ja	2	3	Ziele klar definieren
6	Scope Creep	Der Umfang kann stark steigen wenn Ziele nicht genau genug definiert wurden	Zeitverlust bis hin zu scheitern des Projekts	3	4	Ja	3	3	Ziele SMART definieren
7	SIEM / Log Plattform nicht betriebsbereit	Die öffentliche Ausschreibung für die neue / Log Plattform wurde erst am 23.10.2023 veröffentlicht. Bis zur Implementation kann noch Zeit vergehen.	Logs müssen länger auf dem System selber vorgehalten werden. Zudem müssen ggf. eigene Massnahmen zum Auslesen von Logs getroffen werden	4	1	Nein			
8	Foreman nicht betriebsbereit	Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase. Dadurch besteht das Risiko, dass sie nicht betriebsbereit zum Start der Diplomarbeit ist	Ms müssen von Hand provisioniert werden. Dies bedeutet einen massiven Mehraufwand und verzögert ggf. die Evaluationsphase und mit sicherheit die Installationsphase	3	5	Ja	3	4	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten.

Tabelle 2.1: Risiko-Matrix der Diplomarbeit

Daraus ergeben sich, mit den Massnahmen, folgende Risikomatrizen:

Abbildung 2.1: Risikomanagement Projekt

2.1.1 Riskcontrolling

2.1.1.1 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.			

Tabelle 2.2: Neu Erkannte / Erfasste Risiken

2.1.1.2 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.
2	HP-UX Ablöseprojekt	21.03.2024	5	4	sehr hoch	Ressourcen reserviert	begrenzt	ExaCC Server werden mitten während der Diplomarbeit geliefert. Von KSGR Seite fehlt eine Stellvertretung. Mithilfe notwendig.
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	21.03.2024	4	4	hoch	Externe Partner sensibilisiert	wirksam	Externe Partner können meinen Teil der Aufgaben bei Problemen abfedern. Allerdings nicht vollständig
4	Schwächen beim 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	3	3	hoch	Ziele SMART definiert	gebrenzt	Sehr viele mögliche Lösungen am Markt.
7	SIEM / Log Plattform nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		SIEM wird nicht rechtzeitig stehen. 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 2.3: Risiko-Assessment 21.03.2024

Abbildung 2.2: Riskikomatrix - Assessment 21.03.2024

2.2 Vorgehensweise und Methoden

2.3 Projektplanung

2.3.1 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	0.2	1.0	0.8
2	Aufbau und Implementation Testsystem	Basisinfrastruktur	0.0	4.0	4.0
3	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	0.0	20.0	20.0
4	Aufbau und Implementation Testsystem	Technical Review	0.0	3.0	3.0
5	Dokumentation	Dokumentation	31.0	80.0	49.0
6	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	17.8	16.0	-1.8
7	Evaluation	Anorderungskatalog	4.5	16.0	11.5
8	Evaluation	Gegenüberstellung	0.0	8.0	8.0
9	Evaluation	Variantenentscheid	0.0	4.0	4.0
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	0.0	8.0	8.0
Total			59.5	203.0	143.5

Tabelle 2.4: Projektcontrolling

Abbildung 2.3: Projektcontrolling

2.3.2 GANTT-Diagramm

Nr.		Vorgangsart	Vorgangsname	Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang	Ist-Ende	Geplante Arbeit	Geleistete Arbeit
1			**Phase I: Themenerarbeitung- und eingabe**	**81 Tage**		**Nein**	**Don 17.08.23**	**Don 07.12.23**	**Don 17.08.23**	**NV**	**48 Std.**	
2			Einführung Diplomarbeit	81 Tage		Nein	Don 17.08.23	Don 07.12.23	Don 17.08.23	NV	20 Std.	
3			Ideensammlung	30 Tage		Nein	Don 17.08.23	Mit 27.09.23	NV	NV	8 Std.	
4			Abgabe Ideensammlung	1 Tag		Ja	Mit 27.09.23	Mit 27.09.23	NV	NV	0 Std.	
5			Disposition erstellen	2.5 Tage		Nein	Mit 15.11.23	Son 19.11.23	NV	NV	20 Std.	
6			Abgabe Disposition	1 Tag		Ja	Mon 20.11.23	Mon 20.11.23	NV	NV	0 Std.	
7			Bewilligung Disposition durch FV	1 Tag	6	Ja	Don 07.12.23	Don 07.12.23	NV	NV	8 Std.	
8			**Phase II: Vorarbeitungsarbeiten**	**48 Tage**	**7**	**Nein**	**Fre 08.12.23**	**Die 13.02.24**	**NV**	**NV**	**25 Std.**	
9			Vorebereitungsmassnahmen	43 Tage	7	Nein	Fre 08.12.23	Die 06.02.24	NV	NV	20 Std.	
10			Abgabe Statusbericht	1 Tag	9	Ja	Mit 07.02.24	Mit 07.02.24	NV	NV	1 Std.	
11			Vernissage	1 Tag?	10	Ja	Die 13.02.24	Die 13.02.24	NV	NV	4 Std.	
12			**Phase III: Hauptarbeiten**	**64.75 Tage?**	**11**	**Nein**	**Mit 14.02.24**	**Don 06.06.24**	**NV**	**NV**	**0 Std.**	
13			**Hauptteil Diplomarbeit**	**55.75 Tage?**		**Nein**	**Mit 14.02.24**	**Fre 24.05.24**	**NV**	**NV**	**200 Std.**	
14			**Umsetzung**	**25 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 27.03.24**	**NV**	**NV**	**0 Std.**	
15			**Evaluation**	**13 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 06.03.24**	**NV**	**NV**	**0 Std.**	
16	✓		Anforderungskatalog	10 Tage		Nein	Mit 21.02.24	Fre 08.03.24	Mit 21.02.24	Fre 08.03.24	16 Std.	4.5
17			Vorbereitung Benchmarking	1 Tag	16	Nein	Mon 26.02.24	Mon 26.02.24	NV	NV	4 Std.	
18	✓		Analyse PostgreSQL HA Cluster Lösungen	4 Tage	16	Nein	Mon 11.03.24	Don 14.03.24	Mon 11.03.24	Don 14.03.24	16 Std.	7.5
19			Gegenüberstellung	2 Tage	16;17;18	Nein	Mon 04.03.24	Die 05.03.24	NV	NV	8 Std.	
20			Variantenentscheid	1 Tag?	19	Ja	Mit 06.03.24	Mit 06.03.24	NV	NV	4 Std.	
21			**Aufbau und Implementation Testsystem**	**8 Tage?**	**20**	**Nein**	**Fre 08.03.24**	**Mit 20.03.24**	**NV**	**NV**	**27 Std.**	
22			Basisinfrastruktur	2 Tage		Nein	Fre 08.03.24	Mon 11.03.24	NV	NV	4 Std.	
23			Installation und Konfiguration PostgreSQL HA Cluster	5 Tage	22	Nein	Die 12.03.24	Die 19.03.24	NV	NV	20 Std.	
24			Technical Review	1 Tag?	23	Ja	Mit 20.03.24	Mon 16.09.24	NV	NV	3 Std.	
25			**Testing**	**4 Tage?**	**24**	**Nein**	**Fre 22.03.24**	**Mit 27.03.24**	**NV**	**NV**	**14 Std.**	
26			Testing Testsystem	1 Tag	24	Nein	Fre 22.03.24	Fre 22.03.24	NV	NV	8 Std.	
27			Protokollierung	2 Tage	26	Nein	Mon 25.03.24	Die 26.03.24	NV	NV	4 Std.	
28			Review und Auswertung	1 Tag?	27	Ja	Mit 27.03.24	Mit 27.03.24	NV	NV	2 Std.	
29			Troubleshooting und Lösungsfindung	1 Tag?	25	Nein	Die 02.04.24	Die 02.04.24	NV	NV	8 Std.	
30			**Resultate**	**6 Tage?**	**25**	**Nein**	**Die 02.04.24**	**Mit 10.04.24**	**NV**	**NV**	**7 Std.**	
31			Zielüberprüfung	4 Tage		Nein	Die 02.04.24	Mon 08.04.24	NV	NV	2 Std.	
32			Schlussfolgerung	1 Tag?	31	Nein	Die 09.04.24	Die 09.04.24	NV	NV	2 Std.	
33			Weiteres Vorgehen / offene Arbeiten	1 Tag?		Nein	Die 02.04.24	Die 02.04.24	NV	NV	1 Std.	
34			Persönliches Fazit	1 Tag	31;32	Nein	Mit 10.04.24	Mit 10.04.24	NV	NV	2 Std.	
35			1. Expertengespräch	1 Tag?		Ja	Mit 14.02.24	Mit 14.02.24	NV	NV	0 Std.	
36			2. Expertengespräch	1 Tag?		Ja	Fre 15.03.24	Fre 15.03.24	NV	NV	0 Std.	
37			Letztes Expertengespräch	1 Tag?		Ja	Mit 15.05.24	Mit 15.05.24	NV	NV	0 Std.	
38			Dokumentation	55.75 Tage		Nein	Mit 14.02.24	Fre 24.05.24	NV	NV	80 Std.	11.2
39			Puffer	9 Tage		Nein	Die 16.04.24	Die 30.04.24	NV	NV	16 Std.	
40			Abgabe Diplomarbeit	1 Tag?	13;30;37	Ja	Fre 24.05.24	Fre 24.05.24	NV	NV	0 Std.	
41			Vorbereitung Präsentation	5 Tage	40	Nein	Die 28.05.24	Mon 03.06.24	NV	NV	20 Std.	
42			Präsentation und Fachgespräch	1 Tag?	40;41	Nein	Die 04.06.24	Die 04.06.24	NV	NV	1 Std.	
43			Diplomausstellung	1 Tag?	42	Nein	Mit 05.06.24	Mit 05.06.24	NV	NV	0 Std.	
44			Diplomfeier	1 Tag?	42;43	Nein	Don 06.06.24	Don 06.06.24	NV	NV	0 Std.	

Projekt: Diplomarbeit - Postgre
Datum: Sam 16.03.24

Vorgang		Projektsammelvorgang		Manueller Vorgang		Nur Anfang		Kritisch		Geplanter Meilenstein		Puffer	
Unterbrechung		Inaktiver Vorgang		Nur Dauer		Nur Ende		Kritische Unterbrechung		Geplanter Sammelvorgang			
Meilenstein		Inaktiver Meilenstein		Manueller Sammelvollup		Externe Vorgänge		Verspätet		In Arbeit			
Sammelvorgang		Inaktiver Sammelvorgang		Manueller Sammelvorgang		Stichtag		Geplant		Manueller Fortschritt			

2.4 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 2.5: 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[71]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

Synchrone Replikation
Wenn bei einer Synchronen Replikation eine Transaktion abgesetzt wird, wird der Commit auf der primären Seite erst gesetzt, wenn die Änderung auf der sekundären Seite oder den sekundären Seiten ebenfalls eingetragen und 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, das 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-brainSituation 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 [69, 80, 64] 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, das nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[53].

Konsistenz - Consistency
Die Datenbank ist Konsistent, alle Clients seher gleichzeitig die gleichen Daten unabhängig auf welchem Node Zugegriffen wird. Hierzu muss eine Replikation der Daten an alle Nodes stattfinden und der Commit zurückgegeben werden, also eine Synchrone Replikation stattfinden.

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

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[66]:

Abbildung 3.5: Datenbankskalierung

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	Systemvielfallt		Es muss mindestens eine Monolitisches 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 ausgefühgrt 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 geordeten 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 ausgefühgrt 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 Restoren 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	Monolitisch	MUSS
14	Replikation	Failover / Switchover Garantie	Die Replikation muss sicherstellen, das es bei einem Failover/Switchover zu keinem Fehler kommt	Monolitisch	MUSS
15	Replikation	Throughput	Beschreibt, wie viele Transaktionen pro Zeiteinheit vom Primary an die Replikas gesendet und Commited werden. Dieser Wert ist bei Synchroner 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	Robhustheit	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 Standanrds	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 Heahling		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. Als Basis dienen hier Informationen wie z.B. GitHub Insights.	Beides	MUSS
30	Performance	tps - Read-Only	Die Transaktionsrate (transactions per second / tps) für DQL Transaktionen	Beides	MUSS
31	Performance	tps - Read-Writes	Die Transaktionsrate (transactions per second / tps) für DML Transaktionen	Beides	MUSS
32	Performance	Ø Latenz - Read-Only	Die Latenzzeit bei DQL Transaktionen	Beides	MUSS
33	Performance	Ø Latenz - Read-Write	Die Latenzzeit bei DML Transaktionen	Beides	MUSS

Tabelle 3.2: Anforderungskatalog

3.1.2.2 Stakeholder

Rolle	Funktion	Departement	Bereich	Abteilung
Zabbix Stakeholder	Abteilungsleiter	D10 ICT	Infrastrukturmanagement	ICT Netzwerk, Security und Comm.
Staekholder 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.
Dabei wurden folgende Anforderungen aus übersichtsgründe in Sub-Matrizen aufgeteilt:

- Failover
- Switchover
- Restore
- Replikation
- Sharding
- Quorum
- Management-IP
- Backup
- Performance

Die Grundlegende Gewichtung wurde folgendermassen vorgenommen:

Gewicht	Nennungen	Rang	Nr.	Ziele	1 Systemvielfallt	2 Synergien	3 Failover	4 Switchover	5 Restore	6 Replikation	7 Sharding	8 Quorum	9 Connection	10 Management-API	11 Backup	12 Housekeeping - Log Rotation	13 Self Heahling	14 Monitoring - Node Failure	15 Maintenance Quality
125	15	1	1	Systemvielfallt															
117	14	2	2	Synergien	1														
108	13	3	3	Failover	1	2													
100	12	4	4	Switchover	1	2	3												
92	11	5	5	Restore	1	2	3	4											
83	10	6	6	Replikation	1	2	3	4	5										
25	3	13	7	Sharding	1	2	3	4	5	6									
67	8	7	8	Quorum	1	2	3	4	5	6	8								
58	7	8	9	Connection	1	2	3	4	5	6	9	8							
33	4	12	10	Management-API	1	2	3	4	5	6	10	8	9						
58	7	8	11	Backup	1	2	3	4	5	6	11	8	9	11					
8	1	14	12	Housekeeping - Log Rotation	1	2	3	4	5	6	7	8	9	10	11				
8	1	14	13	Self Heahling	1	2	3	4	5	6	7	8	9	10	11	13			
50	6	11	14	Monitoring - Node Failure	1	2	3	4	5	6	14	8	9	14	11	14	14		
8	1	14	15	Maintenance Quality	1	2	3	4	5	6	7	8	9	10	11	12	15	14	
58	7	8	16	Performance	1	2	3	4	5	6	16	16	16	16	11	16	16	14	16
1000	120																		

Legende

Eingabefelder

Zellbezüge

berechnete Felder

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`-Commandund 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`-Commandund 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-Cooordinator-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,[60, 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 [41, 38, 96, 1] sind dies folgende Anforderungen:

pgGather
Mit pgGather [68] 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 [92] 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 [79].

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[61]:

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 [37] aller Art für PostgreSQL an.

Percona arbeitet oft auf Basis von Patroni,
bietet aber auch eigene Erweiterungen und eigene Software an[33].

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[20].
Nebst Dienstleistungen für das Management von PostgreSQL-Umgebungen, Migrationen aus Oracle-Umgebungen und anderem,
bietet EDB auch eine vielwahl 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 aufgekaufte 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[81]. 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[58]:

- 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 intergrierten 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. [24]

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[29].

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[32]:

Abbildung 3.15: pg_auto_failover-Architektur - Multi-Node Standby

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

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 Sheduled 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[24].

3.1.5.7.4 Pooling

Patroni benötigt einen externen Pooler.
Hier wird oft PgBouncer [34] 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`[97] 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[23]:

- k8s integration
- Deklaratives k8s CRD
- Autoamtische 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[21] implementiert.

3.1.5.8.4 Pooling

PgBounder[34] 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

Anhang - Maintenance

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[93].

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[62].

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:

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

Tabelle Komplett

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

Tablet 1

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

Tablet 2

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

Tablet 3

Abbildung 3.28: yugabyteDB - Sharding

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

Abbildung 3.29: yugabyteDB - Tablet - Leader und Follower

Mit dem Replikationsfaktor kann angegeben werden, auf wie vielen Nodes ein Tablet repliziert werden soll. Bei einem 4-Node System können z.B. einige Tablets einen Faktor 3 haben, dass

heisst, dass die Daten nur auf 3 Nodes repliziert werden. Bei einem Replikationsfaktor 4 werden die Daten auf alle Nodes repliziert. Dies wird mit einem eigenen Service, dem YB-TServer service [39] 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. YuganyteDB nennt dies `Zone outage Tolerance`[36].

Abbildung 3.32: yugabyteDB - Zone outage Tolerance

3.1.5.9.7 Maintenance

Anhang - Maintenance

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 evaluirt, 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 vernüftigen 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 Hocheffizient wenn es um Ressourcenverteilung geht und zudem echtes Sharding. Allerdings setzt es anpassungen an den Tabellen der Applikationen voraus. Das KSGR ist allerdings kein Softwarehaus und kann keine Forks durchführen, auch weil viele Applikationen zertifiziert sein müssen. Scheitert daher an der Machbarkeit

Tabelle 3.9: Vorauswahl - Ausgeschieden

Entsprechend werden nur noch nachfolgende Lösungen genauer betrachtet:

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

Tabelle 3.10: Vorauswahl - Evaluation

3.1.7 Installation verschiedener Lösungen

Entsprechend wurden folgende Server bereitgestellt:

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

Tabelle 3.11: Evaluationssyssteme

3.1.7.1 rke2 - Evaluationsplattform

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

Abbildung 3.33: Evaluationssystem - Distributed SQL / Shards

Die Konfiguration der rke2-Nodes sieht folgendermassen aus:

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

Tabelle 3.12: Evaluationssysstem - Distributed SQL / Sharding

3.1.7.2 Patroni

3.1.7.2.1 Architektur

Abbildung 3.34: Patroni - Evaluationsarchitektur

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 stellte 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 Architekturschemen 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:

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

Listing 3.1: StackGres - values.yaml - Extension proxyUrl

Proxy Schema meint dabei `http` oder `https` Für den KSGR-Proxy sieht der gesamte String entsprechend so aus:

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

Listing 3.2: 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:

```
  postgresServices:
    coordinator:
      primary:
        type: LoadBalancer
      any:
        type: LoadBalancer
    shards:
      primaries:
        type: LoadBalancer
  metadata:
    annotations:
      primaryService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.106
      replicasService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.153
        externalTrafficPolicy: "Cluster"
```

Listing 3.3: 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`):

```
...
  coordinator:
    ...
    pods:
      persistentVolume:
        size: '<öGrsse>Gi'
        storageClass: "stackgres-storage"
...
  shards:
    ...
    pods:
      persistentVolume:
        size: 'öGrsseGi'
        storageClass: "stackgres-storage"
```

Listing 3.4: StackGres-Citus - StorageClass -PVC Binding

Die Instanz-Profile lassen sich wie folgt zuweisen:

```
  ...
  coordinator:
    instances: 1
    ...
    sgInstanceProfile: "<Instanz-Profil Coordinator>"
  ...
  shards:
    ...
```

```
9     sgInstanceProfile: "<Instanz-Profil Shard>"
```

Listing 3.5: 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[35].
Für das Benchmarking wurde dies dann auch umgesetzt.
Dies wird folgendermassen gemacht:

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

Listing 3.6: 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 3.7: 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 Installation

Wähend der Installation des YugabyteDB Evaluations-Enviroment wurde festgestellt, das man zwei Varianten Installieren kann. YugabyteDB (Repository yugabyte) und YugabyteDB Anywhere (Repository yugawre):

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     1%
MEM:     38%
<c>       Copy
<e>       Edit
<n>       Next Match
<shift-n> Prev Match
<r>       Toggle Auto-Refresh
<f>       Toggle FullScreen
Describe(yb-platform/yw-test-yugaware-0)
    Name:      yw-test-yugaware-pg-upgrade
    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-api-access-rgtwb:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    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 ima
ge "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/yugaby
te/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 t
he image may not succeed.
<namespace> <pod> <describe>
```

Abbildung 3.39: yugabyteDB - Susbsription 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 konstelation 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:

```yaml
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
```

```yaml
metadata:
  name: l2adv
  namespace: metallb-system
spec:
  ipAddressPools:
  - distributed-sql

```

Listing 3.8: 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 deakivieren von `kube-proxy` bis hin zu einer Migration zum Cilium-Loadb-Balancers.
Mit diesem funktionierte dann nicht einmal mehr die Installation von yugabyteDB.
Lösung brachte nur ein `GitHub`-Eintrag[30], wo oben genannter Ansatz empfohlen wurde.

3.1.7.4.2 Konfiguration

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

```yaml
...
Image:
  repository: "yugabytedb/yugabyte"
  tag: 2.20.2.1-b3
  pullPolicy: IfNotPresent
  pullSecretName: ""
...

```

Listing 3.9: yugabyteDB - Helm Chart Manifest - Detail Image

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

```yaml
...
storage:
  ephemeral: false  # will not allocate PVs when true
  master:
    count: 1
    size: 3Gi
    storageClass: "yb-storage"
  tserver:
    count: 1
    size: 3Gi
    storageClass: "yb-storage"
...

```

Listing 3.10: yugabyteDB - Helm Chart Manifest - Detail StorageClass

Dem node werden je 4 Cores zur verfügung gestellt. Zei für den `master` und zwei für den `tserver`. Beide erhalten 4GiB Memory:

```
...
resource:
  master:
    requests:
      cpu: "1"
      memory: 2Gi
    limits:
      cpu: "1"
      ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
      ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
      ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
      ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
      memory: 2Gi
  tserver:
    requests:
      cpu: "1"
      memory: 4Gi
    limits:
      cpu: "1"
      ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
      ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
      ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
      ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
      memory: 4Gi
...

```

Listing 3.11: yugabyteDB - Helm Chart Manifest - Detail Resources

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

```
...
replicas:
  master: 3
  tserver: 3
  ## Used to set replication factor when isMultiAz is set to true
  totalMasters: 3
...
```

```

```

Listing 3.12: 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:

```
...
# Disable the YSQL
disableYsql: false
...

```

Listing 3.13: 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.

```
...
domainName: "cluster.local"

serviceEndpoints:
  - name: "yb-master-ui"
    type: LoadBalancer
    annotations: {}
    clusterIP: ""
    ## Sets the Service's externalTrafficPolicy
    externalTrafficPolicy: ""
    app: "yb-master"
    loadBalancerIP: ""
    ports:
      http-ui: "7000"

  - name: "yb-tserver-service"
    type: LoadBalancer
    annotations:
      metallb.universe.tf/loadBalancerIPs: 10.0.20.106
    clusterIP: ""
    ## Sets the Service's externalTrafficPolicy
    externalTrafficPolicy: ""
    app: "yb-tserver"
    loadBalancerIP: ""
    ports:
      tcp-yql-port: "9042"
      tcp-yedis-port: "6379"
      tcp-ysql-port: "5433"
...
```

```
30
```

Listing 3.14: 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 3.15: 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 3.16: 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`:

```
nodeAffinity:
  required:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - sks1183
        - sks1184
        - sks1185
```

Listing 3.17: 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.

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

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 ber einen 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 ber einen ganzen Cluster
Sharding und Datenintegrität	4	Datenkonsistenz	Daten sind Konsistent und Inetger.	Eingetroffen	
Sharding	5	Schutz vor Datenverlust	Die Daten müssen Konsistent und schnell auf die Shards verteilt werden	Eingtetroffen	
Self Healing	6	Node stellt sich selber wieder her	Shard Node wird automatisch synchronisiert	Eingtetroffen	
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 Daten sind Konsistent und Inetger.	Datenkonsistenz 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 `sks1184`), 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 zutritt, ist der Server Node nicht mehr arbeitsfähig da die Zeit für die Synchronisation der Daten benötigt wird[82].
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 Tablespaces erzeugt werden, die genauen Schritte sind im Anhang - YugabyteDB Benchmark SQL zu finden.
Anschliessend 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.

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` die Tabellen einem Sharding unterzogen werden sollen, kommt es zu einer Fehlermeldung.
Die Shards würden mit folgenden SQL-Statements erzeugt:

```
SELECT create_distributed_table('pgbench_branches', 'bid');
SELECT create_distributed_table('pgbench_tellers', 'tid');
SELECT create_distributed_table('pgbench_accounts', 'aid');
SELECT create_distributed_table('pgbench_history', 'aid');
```

Listing 3.18: Citus - Benchmarking - Distributed Table Sharding

Ursache ist, dass eine Distributed Table offenbar keine Foreign Key Constraints erlaubt.
`pgbench` erzeugt aber gleich mehrere davon:

Abbildung 3.40: 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:

```
SELECT create_reference_table('pgbench_branches');
SELECT create_reference_table('pgbench_tellers');
SELECT create_reference_table('pgbench_accounts');
SELECT create_reference_table('pgbench_history');
```

Listing 3.19: 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[18].

Aber auch in diesem Fall muss das Sharding im Nachgang des `pgbench`-Inits gemacht werden. Bei den kleinen Tabellen geht das relativ flot, doch gerade bei der Tabelle `pgbench_accounts` dauert es sehr lange.
Lang genug, dass eine eigene betrachtung beim Benchmarking angezeigt wurde.
Leider zeigte sich auch bei den mixed-Benchmarks, anders als bei den dql-Benchmarks, dass diese Art des Sharding nicht sehr performant ist.

Wie bei Patroni und YugabyteDB auch, musste für den letzten Benchmark die StorageClass auf die neue Disk verlegt werden.
Aber anders als bei YugabyteDB reichten 250GiB nicht mehr, wie bei Patroni lag die Ursache beim Generieren der Primary- und Foreign-Keys.

Es zeigte sich aber auch rasch, dass der Coordinator die gleiche Grösse annahm, wie die Shard-Pods.
Das führte dazu, dass beim letzten Benchmark nur noch 2 Shard-Instanzen deployt werden konnten,
da sonst bei jeder Disk nochmals mindestens 350GiB hinzugefügt hätte werden müssen (da sich der Coordinator den Node mit einem Shard geteilt hätte und der Coordinator auf jedem Node erscheinen könnte).
Es hätten also für die Evaluation 3 x 700GiB (plus noch 3 x 50GiB für den Rest), 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.

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 3.41: Benchmarks - tps

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

Abbildung 3.42: 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 3.43: Benchmarks - tps Patroni Replica

Abbildung 3.44: Benchmarks - latency Patroni Replica

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

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

Abbildung 3.45: Benchmarks - Fehler bei mixed-Transaktionen

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

Abbildung 3.46: Benchmarks - Initialisierungszeit - sekunden

Abbildung 3.47: Benchmarks - Initialisierungszeit - minuten

Abbildung 3.48: 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.

3.1.9.3 Gemessene Zeiten und Kosten

3.1.9.3.1 Gegebene Parameter und Annahmen

3.1.9.3.2 Gemessene Zeiten

3.1.9.3.3 Zeitvergleiche

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
Secuirty 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 3.16: Gemessene und Extrapolierte Aufwände Bsp.

Abbildung 3.49: Zeitaufwände

3.1.9.3.4 Kostenvergleiche

Abbildung 3.50: Kostenaufwände

3.1.9.4 Kosten-Nutzen

Abbildung 3.51: Kosten-Nutzen-Analyse

3.1.10 Entscheid

3.2 Aufbau und Implementation Testsystem

3.2.1 Bereitstellen der Grundinfrastruktur

3.2.2 Installation und Konfiguration PostgreSQL HA Cluster

3.2.3 Technical Review der Umgebung

3.3 Testing

3.3.1 Testing

3.3.2 Protokollierung

3.3.3 Review und Auswertung

3.4 Troubleshooting und Lösungsfindung

4 Resultate

4.1 Zielüberprüfung

4.2 Schlussfolgerung

4.3 Weiteres Vorgehen / offene Arbeiten

4.4 Persönliches Fazit

Abbildungsverzeichnis

1.1 Spitalregionen Kanton Graubünden[57] . . . 1
1.2 Wahlkreise Kanton St. Gallen[83] . . . 2
1.3 Spitalregionen / Spitalstrategie Kanton St. Gallen[51] . . . 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

2.1 Risikomanagement Projekt . . . 23
2.2 Riskikomatrix - Assessment 21.03.2024 . . . 27
2.3 Projektcontrolling . . . 29

3.1 Sharding - Vertikale Partitionierung . . . 33
3.2 Sharding - Horizontales Partitionierung . . . 34
3.3 Monolithische vs. verteilte SQL Systeme . . . 36
3.4 CAP-Theorem . . . 38
3.5 Datenbankskalierung . . . 39
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 . . . 69
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 . . . 78
3.36 Stackgres - Citus - Evaluationsarchitektur Self Healing Tests . . . 78
3.37 Stackgres - Citus - Resourcen - Stack . . . 79
3.38 Stackgres - Citus - Datenbank - Cluster . . . 80
3.39 yugabyteDB - Susbsription yugawre . . . 85
3.40 Benchmarking - ERD pgbench . . . 94
3.41 Benchmarks - tps . . . 96
3.42 Benchmarks - latency . . . 96
3.43 Benchmarks - tps Patroni Replica . . . 97
3.44 Benchmarks - latency Patroni Replica . . . 97
3.45 Benchmarks - Fehler bei mixed-Transaktionen . . . 98
3.46 Benchmarks - Initialisierungszeit - sekunden . . . 98
3.47 Benchmarks - Initialisierungszeit - minuten . . . 99
3.48 Benchmarks - Initialisierungszeit - stunden . . . 99
3.49 Zeitaufwände . . . 101
3.50 Kostenaufwände . . . 102
3.51 Kosten-Nutzen-Analyse . . . 103

I CloudNativePG - Pulse . . . vi
II CloudNativePG - Code Frequency . . . vi
III CloudNativePG - Community Standards . . . vii
IV CloudNativePG - Contributors Commits . . . vii
V CloudNativePG - Contributors Deletations . . . viii
VI CloudNativePG - Contributors Additions . . . viii
VII CloudNativePG - Commit Activity . . . viii
VIII CloudNativePG - Network Graph . . . ix
IX Patroni - Pulse . . . ix

X Patroni - Code Frequency x
XI Patroni - Community Standards x
XII Patroni - Contributors Commits xi
XIII Patroni - Contributors Deletations xi
XIV Patroni - Contributors Additions xi
XV Patroni - Commit Activity xii
XVI Patroni - Network Graph xii
XVII Stackgres - Pulse xiii
XVIII Citus - Pulse xiii
XIX Stackgres - Code Frequency xiii
XX Citus - Code Frequency xiv
XXI Stackgres - Community Standards xiv
XXII Citus - Community Standards xv
XXIII Stackgres - Contributors Commits xv
XXIV Stackgres - Contributors Deletations xvi
XXV Stackgres - Contributors Additions xvi
XXVI Citus - Contributors Commits xvi
XXVII Citus - Contributors Deletations xvii
XXVIII Citus - Contributors Additions xvii
XXIX Stackgres - Commit Activity xvii
XXX Citus - Commit Activity xviii
XXXI Stackgres - Network Graph xviii
XXXII Citus - Network Graph xix
XXXIII YugabyteDB - Pulse xix
XXXIV YugabyteDB - Code Frequency xx
XXXV YugabyteDB - Community Standards xx
XXXVI YugabyteDB - Contributors xxi
XXXVII YugabyteDB - Commit Activity xxi
XXXVIII YugabyteDB - Network Graph xxii
XXXIX StackGres Testing - Node sks1184 down xci
XL StackGres Testing - Pods Down xci
XLI StackGres Testing - Patroni Übersicht xcii
XLII StackGres Testing - DB Zugriff xciii
XLIII StackGres Testing - Connection Timeout xciii
XLIV YugabyteDB - Too big clock skew is detected xciv
XLV YugabyteDB - Tablet Leader - No Lease xciv
XLVI YugabyteDB - CrashLoopBackOff xciv
XLVII YugabyteDB - Too big clock skew is detected - tmaster xcv
XLVIII YugabyteDB - Too big clock skew is detected - tserver xcv

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 . . . 19

2.1 Risiko-Matrix der Diplomarbeit . . . 22
2.2 Neu Erkannte / Erfasste Risiken . . . 24
2.3 Risiko-Assessment 21.03.2024 . . . 26
2.4 Projektcontrolling . . . 28
2.5 Fachgespräche . . . 32

3.1 Quorum Beispiele . . . 37
3.2 Anforderungskatalog . . . 42
3.3 Stakeholder . . . 43
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 Evaluationssyssteme . . . 76
3.12 Evaluationssysstem - Distributed SQL / Sharding . . . 76
3.13 Testresultate Evaluation Patroni . . . 91
3.14 Testresultate Evaluation StackGres - Citus . . . 91
3.15 Testresultate Evaluation YugabyteDB . . . 92
3.16 Gemessene und Extrapolierte Aufwände Bsp. . . . 100

I Arbeitsrapport . . . i
II Fachgespräche - Protokoll . . . ii
III Kommentare - Anmerkung . . . v

Listings

3.1 StackGres - values.yaml - Extension proxyUrl . . . 81
3.2 StackGres - values.yaml - Extension Proxy . . . 81
3.3 StackGres-Citus - LoadBalancer -Annotation . . . 83
3.4 StackGres-Citus - StorageClass -PVC Binding . . . 83
3.5 StackGres-Citus - Instanz-Profile . . . 83
3.6 StackGres-Citus - StorageClass -PVC Binding . . . 84
3.7 StackGres-Citus - Cluster Profil . . . 84
3.8 metallb - Konfig YAML - Detail L2Advertisement . . . 85
3.9 yugabyteDB - Helm Chart Manifest - Detail Image . . . 86
3.10 yugabyteDB - Helm Chart Manifest - Detail StorageClass . . . 86
3.11 yugabyteDB - Helm Chart Manifest - Detail Resources . . . 87
3.12 yugabyteDB - Helm Chart Manifest - Detail Replika . . . 87
3.13 yugabyteDB - Helm Chart Manifest - Detail Disable YSQL . . . 88
3.14 yugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints . 88
3.15 local-path-provisioner nodePathMap . . . 89
3.16 local-path-provisioner nodePathMap Beispiel . . . 89
3.17 yugabyteDB - StorageClass nodeAffinity . . . 90
3.18 Citus - Benchmarking - Distributed Table Sharding . . . 93
3.19 Citus - Benchmarking - Reference Table Sharding . . . 94

1 Proxy Settings . . . xxii
2 rke2 server - Verzeichnis erstellen . . . xxiii
3 rke2 server - config.yaml . . . xxiii
4 rke2 server - cilium-config.yaml . . . xxiii
5 rke2 server installieren . . . xxiii
6 rke2 agenten installieren . . . xxiii
7 rke2 agent - config.yaml . . . xxiv
8 -rke2 agent service restart . . . xxiv
9 rke2 server proxy . . . xxiv
10 rke2 server proxy kopieren . . . xxiv
11 rke2 server cilium installieren . . . xxiv
12 rke2 server cilium aktivieren . . . xxiv
13 rke2 server starten . . . xxiv
14 iptables entries server . . . xxv
15 rke2 server token . . . xxv

16 local-path-storage auf Linux Bereitstellen . . . xxvi
17 local-path-provisioner definieren . . . xxvi
18 local-path-storage aktualisieren . . . xxvii
19 MetallB installieren . . . xxvii
20 MetallB konfigurieren . . . xxvii
21 MetallB Konfiguration einspielen . . . xxvii
22 rke2 - 250GiB Disk mount . . . xxviii
23 local-path-storage 250GiB auf Linux Bereitstellen . . . xxviii
24 local-path-provisioner Grosse Volumes . . . xxviii
25 local-path-storage 250GiB aktualisieren . . . xxix
26 yugabyteDB - StorageClass setzen . . . xxx
27 yugabyteDB - StorageClass / PersistentVolume aktivieren . . . xxx
28 yugabyteDB - Namespace . . . xxxi
29 yugabyteDB - Helm Chart Manifest . . . xxxi
30 yugabyteDB - Installation . . . xliii
31 yugabyteDB - Deinstallieren . . . xliii
32 yugabyteDB - StorageClass setzen . . . xliv
33 yugabyteDB - StorageClass / PersistentVolume Grosse Volumes aktivieren . . . xliv
34 yugabyteDB - Namespace 250GiB . . . xlv
35 yugabyteDB - Helm Chart Manifest 250GiB . . . xlv
36 yugabyteDB - Cloud - Region - Zone . . . liv
37 yugabyteDB - Benchmarking - DB erstellen . . . liv
38 yugabyteDB - Benchmarking - Table Size . . . lv
39 YugabyteDB - Self Healing Tests - CREATE-SQL . . . lv
40 YugabyteDB - Self Healing Tests - Init Data . . . lviii
41 YugabyteDB - Self Healing Tests - Failover Data . . . lix
42 YugabyteDB - Self Healing Tests - Recovery Data . . . lx
43 sks9016 - Download yugabyteDB On-Premise . . . lxi
44 sks9016 - Installation yugabyteDB On-Premise . . . lxi
45 sks9016 - Check yugabyteDB On-Premise . . . lxi
46 StackGres-Citus - StorageClass setzen . . . lxi
47 StackGres-Citus - StorageClass / PersistentVolume aktivieren . . . lxii
48 StackGres-Citus - Namespace . . . lxiii
49 StackGres-Citus - Helm Chart Manifest . . . lxiii
50 StackGres-Citus - Installation . . . lxxii
51 StackGres-Citus - Post-Installation . . . lxxii
52 StackGres-Citus - System Username . . . lxxii
53 StackGres-Citus - System Passwort . . . lxxii
54 StackGres-Citus - System Passwort Cleanup . . . lxxiii

55 StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator lxxiii
56 StackGres-Citus - Benchmarking - SGInstanceProfile Shard lxxiii
57 StackGres-Citus - Benchmarking - Instanz-Profil Deploy lxxiii
58 StackGres-Citus - Benchmarking - SGShardedCluster lxxiii
59 StackGres-Citus - Benchmark - Cluster Deploy lxxiv
60 StackGres-Citus - Benchmark DB Passwort lxxiv
61 StackGres-Citus - Self Healing Testing - SGInstanceProfile Coordinator lxxv
62 StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard lxxv
63 StackGres-Citus - Self Healing Testing - Instanz-Profil Deploy lxxv
64 StackGres-Citus - Self Healing Testing - SGShardedCluster lxxv
65 StackGres-Citus - Self Healing Testing - Cluster Deploy lxxvi
66 StackGres-Citus - Self Healing Testing DB Passwort lxxvi
67 StackGres-Citus - Deinstallieren lxxvii
68 StackGres-Citus - StorageClass setzen lxxvii
69 StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren . . lxxviii
70 StackGres-Citus - Namespace 250GiB lxxviii
71 StackGres-Citus - Installation 250GiB lxxviii
72 StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB lxxviii
73 StackGres-Citus - Benchmarking - SGInstanceProfile Shard 250GiB lxxviii
74 StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB lxxix
75 StackGres-Citus - Benchmarking - SGShardedCluster 250GiB lxxix
76 StackGres-Citus - Benchmark - Cluster Deploy 250GiB lxxx
77 StackGres-Citus - Self Healing Tests - CREATE-SQL lxxx
78 StackGres-Citus - Self Healing Tests - Init Data lxxxiii
79 StackGres-Citus - Self Healing Tests - Failover Data lxxxiv
80 StackGres-Citus - Self Healing Tests - Recovery Data lxxxv
81 Patroni - Proxy Settings lxxxv
82 Patroni - apt-Proxy Settings lxxxv
83 Patroni - PostgreSQL einbinden lxxxvi
84 YugabyteDB - Benchmarking-Commands lxxxvii
85 yugabyteDB - Benchmarking - Table Size SQL lxxxix
86 StackGres-Citus - Benchmarking-Commands lxxxix
87 StackGres-Citus - Benchmarking - Table Size SQL xci
88 Python LaTex - zotero.py - Zotero BibLaTex Importer xcvi
89 Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer cii
90 Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer cii
91 Python LaTex - riskmatrix.py - Risxikomatrizen cix

92 Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen cxiii
93 Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples cxvii
94 Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm cxviii
95 Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm cxx
96 Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle cxxi
97 Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle cxxx
98 Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm cl
99 Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme clvii

Literatur

[1] *About pgbench-tools*. https://github.com/gregs1104/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-ba 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 - worder-node-failure*. https://docs.citusdata.com/en/v12.1/admin_guide/cluster_management.html#worker-node-failu

[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. *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.

[19] unknown author. *Dynamic Configuration Settings — Patroni 3.2.2 documentation*. https://patroni.readthedocs.io/en/latest/dynamic_configuration.html.

[20] unknown author. *EDB-Home*. https://enterprisedb.com/.

[21] unknown author. *Envoy proxy - home*. https://www.envoyproxy.io/.

[22] unknown author. *etcd*. https://etcd.io/.

[23] unknown author. *Features - StackGres Documentation*. https://stackgres.io/doc/latest/features/.

[24] unknown author. *HAProxy Documentation Converter*. https://docs.haproxy.org/.

[25] unknown author. *HAProxy version 2.9.6 - Starter Guide*. https://docs.haproxy.org/2.9/intro.html#3.2.

[26] unknown author. *Helm*. https://helm.sh/.

[27] unknown author. *Introduction | RKE2*. https://docs.rke2.io/. 2024.

[28] unknown author. *Introduction to Cilium & Hubble — Cilium 1.15.3 documentation*. https://docs.cilium.io/en/stable/overview/intro/#what-is-cilium.

[29] unknown author. *Manual Pages — pg_auto_failover 2.0 documentation*. https://pg-auto-failover.readthedocs.io/en/main/ref/manual.html.

[30] 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.

[31] unknown author. *MetalLB, bare metal load-balancer for Kubernetes*. https://metallb.universe.tf/.

[32] unknown author. *Multi-node Architectures — pg_auto_failover 2.0 documentation*. https://pg-auto-failover.readthedocs.io/en/main/architecture-multi-standby.html.

[33] unknown author. *Percona Software for PostgreSQL*. https://www.percona.com/postgresql/software.

[34] unknown author. *PgBouncer - lightweight connection pooler for PostgreSQL*. https://www.pgbouncer.org/.

[35] unknown author. *Problem with 08P01: server conn crashed? · Issue #714 · pgbouncer/pgbouncer*. https://github.com/pgbouncer/pgbouncer/issues/714.

[36] unknown author. *Replication in DocDB - Zone Fault Tolerance*. https://docs.yugabyte.com/preview/architecture/docdb-replication/replication/. Section: preview.

[37] unknown author. *Support and Services for PostgreSQL*. https://www.percona.com/postgresql/support-and-services.

[38] unknown author. *Tuning PostgreSQL with pgbench*. https://www.cloudbees.com/blog/tuning-postgresql-with-pgbench. 2017.

[39] unknown author. *YB-TServer service*. https://docs.yugabyte.com/preview/architecture/concepts/yb-tserver/. Section: preview.

[40] GitLab B.V. und GitLab Inc. *The DevSecOps Platform | GitLab*. https://about.gitlab.com/.

[41] Fernando Laudares Camargos Avinash Vallarapu. *Tuning PostgreSQL for sysbench-tpcc*. https://www.percona.com/blog/tuning-postgresql-for-sysbench-tpcc/. 2018.

[42] Alexandre Cassen und Read the Docs. *Introduction — Keepalived 1.2.15 documentation*. https://keepalived.readthedocs.io/en/latest/introduction.html. 2017.

[43] Microsoft Corporation. *Azure SQL-Datenbank – ein verwalteter Clouddatenbankdienst | Microsoft Azure*. https://azure.microsoft.com/de-de/products/azure-sql/database. 2023.

[44] Microsoft Corporation. *Datenbank-Software und Datenbankanwendungen | Microsoft Access*. https://www.microsoft.com/de-de/microsoft-365/access. 2023.

[45] Microsoft Corporation. *Microsoft Data Platform | Microsoft*. https://www.microsoft.com/de-ch/sql-server.

[46] Varun Dhawan und data-nerd.blog. *PostgreSQL-Diagnostic-Queries – data-nerd.blog*. https://data-nerd.blog/2018/12/30/postgresql-diagnostic-queries/.

[47] Elektronik-Kompendium.de und Schnabel Schnabel. *SAN - Storage Area Network*. https://www.elektronik-kompendium.de/sites/net/0906071.htm. 2023.

[48] DB-Engines und solidIT consulting & software development gmbh. *DB-Engines Ranking*. https://db-engines.com/en/ranking.

[49] DB-Engines und solidIT consulting & software development gmbh. *relationale Datenbanken - DB-Engines Enzyklopädie*. https://db-engines.com/de/article/relationale+Datenbanken?ref=RDBMS.

[50] The Linux Foundation. *Harbor*. https://goharbor.io/. 2023.

[51] 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-s spitaeler-kliniken/spitalzukunft.html.

[52] Git. *About - Git*. https://git-scm.com/about.

[53] IBM Deutschland GmbH. *Was ist das CAP-Theorem? | IBM*. https://www.ibm.com/de-de/topics/cap-theorem. 2023.

[54] IBM Deutschland GmbH. *Was ist OLAP? | IBM*. https://www.ibm.com/de-de/topics/olap.

[55] Jedox GmbH. *Was ist OLAP? Online Analytical Processing im Überblick*. https://www.jedox.com/de/blog/was-ist-olap/. Section: Knowledge.

[56] 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.

[57] 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.

[58] The Pgpool Global Development Group. *What is Pgpool-II?* https://www.pgpool.net/docs/44/en/html/intro-whatis.html. 2023.

[59] The PostgreSQL Global Development Group. *25.1. Routine Vacuuming*. https://www.postgresql.org/docs/16/routine-vacuuming.html. 2023.

[60] The PostgreSQL Global Development Group. *pgbench*. https://www.postgresql.org/docs/16/pgbench.html. 2023.

[61] 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-targe 2018.

[62] Michael Haag. *Built-in Connection Manager Turns Key PostgreSQL Weakness into a Strength*. https://www.yugabyte.com/blog/connection-pooling-management/. 2023.

[63] Inc. HashiCorp. *Terraform by HashiCorp*. https://www.terraform.io/.

[64] Patrick Hunt, Mahadev Konar, Flavio P Junqueira und Benjamin Reed. „ZooKeeper: Wait-free coordination for Internet-scale systems". In: (2010).

[65] Splunk Inc. *Splunk | Der Schlüssel zu einem resilienten Unternehmen*. https://www.splunk.com/de_de. 2023.

[66] Sebastian Insausti. *Scaling PostgreSQL for Large Amounts of Data*. https://severalnines.com/blog/scaling-postgresql-large-amounts-data/. 2019.

[67] Shiv Iyer und MinervaDB. *PostgreSQL DBA Daily Checklist*. https://minervadb.xyz/postgresql-dba-d 2020.

[68] *jobinau/pg_gather*. https://github.com/jobinau/pg_gather. original-date: 2021-01-19T08:12:07Z. 2024.

[69] Unmesh Joshi. *Quorum*. https://martinfowler.com/articles/patterns-of-distributed-systems/quorum.html. 2020.

[70] Martin Keen und IBM Deutschland GmbH. *IBM Db2*. https://www.ibm.com/de-de/products/db2.

[71] Pasha Kostohrys. *Database replication — an overview*. https://medium.com/@pkostohrys/database-replication-an-overview-f7ade110477. 2020.

[72] Anatoli Kreyman. *Was ist eigentlich Splunk?* https://www.kreyman.de/index.php/splunk/76-was-ist-eigentlich-splunk-big-data-platform-monitoring-security.

[73] 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.

[74] Red Hat Limited. *Was ist Ansible?* https://www.redhat.com/de/technologies/management/ansible/what-is-ansible.

[75] 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.

[76] Switzerland Linuxfabrik GmbH Zurich. *Keepalived — Open Source Admin-Handbuch der Linuxfabrik*. https://docs.linuxfabrik.ch/software/keepalived.html. 2023.

[77] Nico Litzel, Stefan Luber und Vogel IT-Medien GmbH. *Was ist Elasticsearch?* https://www.bigdata-insider.de/was-ist-elasticsearch-a-939625/. 2020.

[78] Hewlett Packard Enterprise Development LP. *Was ist SAN-Speicher? | Glossar*. https://www.hpe.com/ch/de/what-is/san-storage.html.

[79] Julian Markwort. „Benchmarking four Different Replication Solutions“. In: ().

[80] Diego Ongaro. „Consensus: Bridging Theory and Practice“. In: (2014).

[81] Bruno Queirós und LinkedIn Ireland Unlimited Company. *Postgresql replication with automatic failover*. https://www.linkedin.com/pulse/postgresql-replication-automatic-failover-brun C3%B3s. 2020.

[82] Karthik Ranganathan. *Evolving Clock Sync in Distributed Databases | YugabyteDB*. https://www.yugabyte.com/blog/evolving-clock-sync-for-distributed-databases/. 2022.

[83] 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.

[84] Ed Reckers und SnapLogic Inc. *Was ist die Snowflake-Datenplattform?* https://www.snaplogic.com/de/blog/snowflake-data-platform. 2023.

[85] IONOS SE. *Apache Cassandra: Verteilte Verwaltung großer Datenbanken*. https://www.ionos.de/digitalguide/hosting/hosting-technik/apache-cassandra-vorgestellt/. 2021.

[86] IONOS SE. *Datenbankmanagementsystem (DBMS) erklärt*. https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklaert/. 2020.

[87] IONOS SE. *MongoDB – die flexible und skalierbare NoSQL-Datenbank*. https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysc 2019.

[88] IONOS SE. *SQLite: Die bekannte Programmbibliothek im Detail vorgestellt*. https://www.ionos.de/digitalguide/websites/web-entwicklung/sqlite/. 2023.

[89] IONOS SE. *Terraform*. https://www.ionos.de/digitalguide/server/tools/was-ist-terraform/. 2020.

[90] IONOS SE. *Was ist Redis? Die Datenbank vorgestellt*. https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-redis/. 2020.

[91] IONOS SE. *Was ist SIEM (Security Information and Event Management)?* https://www.ionos.de/digitalguide/server/sicherheit/was-ist-siem/. 2020.

[92] Naveed Shaikh. *It's All About Replication Lag in PostgreSQL*. https://www.percona.com/blog/replication-lag-in-postgresql/. 2023.

[93] Sami Ahmed Siddiqui. *Distributed SQL 101*. https://www.yugabyte.com/distributed-sql/.

[94] Inc. Snowflake. *Datenbanken, Tabellen und Ansichten – Überblick | Snowflake Documentation*. https://docs.snowflake.com/de/guides-overview-db.

[95] Thomas-Krenn.AG. *Git Grundlagen – Thomas-Krenn-Wiki*. https://www.thomas-krenn.com/de/wiki/Git_Grundlagen.

[96] Mahmut Can Uçanefe. *Pgbench Load Test*. https://medium.com/@c.ucanefe/pgbench-load-test-166b 2023.

[97] *vitabaks/postgresql_cluster*. https://github.com/vitabaks/postgresql_cluster. original-date: 2019-06-04T13:26:17Z. 2024.

[98] 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 Mananagement System
k8s	Kubernetes
HPE	Hewlett Packard Enterprise
HP-UX	Hewlett Packard UNIX
SAP	Systemanalyse Programmentwicklung
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[74].. 17

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[59].. 15, 16, 47

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[85] wurde.. 7, 69, 75

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[75].. 120

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-Policises, Load-Balancer und andere Features[10, 28].. 85, 86

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[86].. 120

DCS Der DSC ist eine Kernkomponente von Patroni [19]. Realisiert wird der DCS bei Patroni mit Etcd.. 121

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 beliebstesten Distributionen wie Ubuntu hervor.. 17

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[77].. 7

etcd etcd ist [22]. 61, 122, lxxxvi

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 16, 36, 37, 53, 123

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, lxxxv

Git Git ist eine Versionierungssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral sondern dezentral organsiert und arbeiten daher mit Working Copies von Repositories[52, 95].. 123

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**].. lxxxvi

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[40].. 16, 53

HAProxy HAProxy [25]. 55, 60

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[50]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 16, 53

helm helm bietet mit seinen helm charts Paketressourcen, die das deployment von Kubernetes-Ressourcen erleichtert[26].. 80, lxxviii

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, 120

IBM DB2 IBM DB2 ist eine Relationale Datenbank[70] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 38

keepalived keepalived nutzt VRRP um eine leichtgewichtige Lösung für ein HA-Failover zu realsieren. 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[76, 42].. 53

Key-Value-orientierte Siehe Key-Value-Datenbank. 126

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. 123, 124, 125

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, 91, 120, 123

Linux Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und lose 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, 124

local-path-provisioner local-path-provisioner ist ein leichtgewichtiger Storage-Provider von Rancher. Er bietet den Applikationen einen persistenten Storage an.. 89, 90

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 eder Zeit immer mehr voneinander undf sind nicht mehr in jdem Fall kompatibel oder beliebig austauschbar. Auf den Linux Distributionen tratt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 9, 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[31].. 82, 85

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[43].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[44].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[45]. 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, 125

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zurm ersten Mal 2007 veröffentlicht wurde[87].. 7

MySQL Die Datenbank MySQL wurde Ursprünglich als reine Relationale Open-Source Datenbank von Firma MySQL AB 1994 Entwickelt. Der Name My leitet sich vom Namen My der Tochter des Mitbegründers Michael Widenius ab. Als Sun Microsystem 2008 MySQL übernahm, hielt sich die Option frei, bei einem Kauf von Sun Microsyszem durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 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, 122, 123, 124, 126, 127

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[55, 54]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 7, 120

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 komnt auf den Oracle Eigenenen 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 Vergangheit ermöglichen) konnte Oracle sich stark von der Konkurenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Perforamce oder grosse Datenmengen geht.. 5, 7, 9, 38, 125

PKI . 121

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

beliebstesten OpenSource Datenbanken. Zudem besteht in vielen bereichen eine gewisse ähnlichkeit zu Oracles Oracle Database.. 5, 7, 9, 10, 14, 38, 53, 68

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, 126

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, 120

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 Konstelationen 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, 123

RDBMS Ein RDBMS ist ein Datenbankmanagementsystem für eine Relationale Datenbank. Relationale Datenbanken sind Tabellenorgansierte Datenmodelle die auf Relationen aufbauen, deren Schematas sich Normalisieren lassen. Dabei müssen Relationale Datenbanken müssen dabei auch Mengenoperationen, Selektion, Projektion und Joins erfüllen um als Relationale Datenbanken zu gelten[49].. 69, 120

RedHat Enterpise Linux (RHEL) RHEL wurde in seiner Ursprüglichen 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 aderen 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[90]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 123

rke2 rke2 ist eine Leichgewichtige Kubernetes Distribution, die alles notwendige mitbringt, um einen k8s Cluster zu betreiben[27].. 76, 81, 85, 92

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[47, 78, 56]. 5, 17, 22, 95, 120

SIEM Ein sammelt Daten aus verschieden 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[91].. 17, 120

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[94, 84]. 7

Split-brain Im Kapitel **??** werden die ursachen und folgenden eines Split-brains genauer besprochen. . 37, 126

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[65, 72]. . 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[88].. 7

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

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.. 120

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[89, 63].. 17

Transaktion Eine Transaktion ist beinhaltet Schreib-, Lese-, Mutatations- oder Löschoperatione 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

enstanden im Laufe der Zeit viele offene und Proprioritäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 120

VMware vSphere Die vSphere® ist ein Typ-1 Hypervisor der Firma VMware® der eine reihe leistungsstarker Tools und Funktionen mitbringt.. xxviii

VRRP VRRP . 120, 123

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. Zabbrix speichert die Metriken und nicht die Auswertungen, das heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert worden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge Abschliessen.. 9, 17

Selbstständigkeitserklärung

Ich versichere, dass die vorliegende Arbeit von den Autoren selbständig 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.

I 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	Anorderungskatalog	Anorderungskatalog erarbeiten			
22.02.2024	16:00	17:30	1.5	Evaluation	Anorderungskatalog	Anorderungskatalog 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	Anorderungskatalog	Anorderungskatalog 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	Projektcontrolling Arbeiten			
17.03.2024	14:45	16:30	1.8	Dokumentation	Dokumentation	Dokumentation erweitern	Zweiter Statusbericht verfassen		
17.03.2024	19:30	20:00	0.5	Dokumentation	Dokumentation	Dokumentation erweitern	ACID Exkurs erfassen 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 Schrieben 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 Anbgabe 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 organisiert			
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. Das KSGR hat Harbor im Einsatz, allerdings ist dieses nicht für das Evaluationssetting gedacht.	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			Nornan 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. Diese Installation benötigt zwingend eine Subscription. Wäre ein No-Go	YugayteDB (Repository yugabyte) verwenden. Dies ist nach wie vor Open-Source

Tabelle I: Arbeitsrapport

II Protokoll - Fachgespräche

Fachgespräch	Datum	Fachexperte	Nebenexperte	Studenten	Fragen	Antworten	Sonstige Themen	Bemerkungen
1	14.02.2024	Norman Süsstrunk	-	Michael Graber Curdin Roffler	- Darf eine Vorauswahl stattfinden, um den Aufwand zur reduzieren?	- Eine Vorauswahl ist Sinnvoll und in diesem Rahmen fast zwingend Notwendig, da sonst viel zuviel Zeit investiert werden müsste	- Vorstellung Norman Süsstrunk, Curdin Roffler und Michael Graber - Kontaktdaten shared - Bei Fragen jederzeit an Norman wenden - Norman braucht aber mindestens 1. Woche vorlaufzeit - Norman wird sich spätestens zur Halbzeit melden. - Norman wird sic	- 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	- Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden? - Hat Norman ggf. noch vorschläge zu PostgreSQL Clustern gefunden? - Soll ich die Gewichtung mit 100 Punkten machen oder 1000? Im Moment haben diverse Punkte eine sehr kleine Punktzahl - Soll die Disposition in den Anhang? Diese ist 50 Seiten lang		- Protokoll genehmigen	

Tabelle II: Fachgespräche - Protokoll

=:

III Statusbericht

III.I Status Report 1

III.II Status Report 2

IV Kommentare / Anmerkungen

Hier werden Kommentare und Anmerkungen, welche für das Fazit wichtig sein könnten, gesammelt.

Woche	Beschreibung / Event / Problem
KW10	Vier ganze Tage war ich in Thalwil für die Oracle Multitenant-Schulung für das ExaCC Projekt (Ablösung HP-UX). Am Freiutag war ich ebenfalls fast den ganzen Tag dran. Weitere Termine werden folgen, das Risiko durch das Projekt tritt langsam ein. Projekt Zeitlich im Verzug.
KW11	Nebst dem HP-UX Ablösungsprojekt schlagen auch diverse Betriebsthemen ein. Die analyse der PostgreSQL HA Cluster nimmt ebenfalls mehr Zeit in Anspruch, als erwartet.
KW12	- HP-UX Probleme am Montag. Backups sind über das Weekend nicht durchgelaufen. Die ganze Montagsplanung wurde über den Haufen geworfen. - Besprechung bezüglich Backup. Veeam Kasten steht noch nicht zur Verfügung. - Mittwochvormittag in Zürich, am Nachmittag Probleme mit dfs-Shares. So wenig Zeit.
KW12	- Mit Norman Termin für nächste Woche Fachgespräch organisiert. Freue mich darauf.
KW12	- Alle Gängigen PostgreSQL HA Lösungen dokumentiert. Aufwand für Die Dokumentation weit grösser als erwartet. - YugabyteDB entpuppt sich als recht fordernd. Es benötigt eine «private container registry», mir fehlt die Expertise dazu.
KW13	- Der Aufbau der Projektplanung entpuppt sich begrenzt nutzbar. Das erstellen der Evaluationsinfrast - Das Problem mit dem «private container registry»rührte daher, dass das YugabyteDB Anywhere (Repository yugaware) verwendet wurde.
KW13	Kurz ein Schock, dass YugabyteDB ausgeschieden ist. Später bemerkte ich, dass man das Repository yugabyte verwendet muss. - Bereits jetzt viel gerlernt über Kubernetes, Ranger (rke2) und Helm

Tabelle III: Kommentare - Anmerkung

V Evaluation

V.I Maintenance - CloudNativePG

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

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:

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.

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 verhältnissmässig hohe Requstrate:

Abbildung XVIII: Citus - Pulse

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

Abbildung XIX: Stackgres - Code Frequency

Abbildung XX: Citus - Code Frequency

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

Abbildung XXI: Stackgres - Community Standards

Abbildung XXII: Citus - Community Standards

Die Stackgres Constributors pflegen aktiv Additions ein, löschen Regelmässig und Commiten ebenfalls auf die main-Branch. Citus, dessen Repository länger Commited 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 sinned:

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 SStandards ein:

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:

```
sudo nano /etc/profile.d/proxy.sh

export https_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
export http_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
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
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

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:

```
mkdir -p /etc/rancher/rke2/
mkdir -p /var/lib/rancher/rke2/server/manifests/
```

Listing 2: rke2 server - Verzeichnis erstellen

```yaml
# /etc/rancher/rke2/
cluster-cidr: "198.18.0.0/16"
service-cidr: "198.19.0.0/16"
cni:
  - cilium
disable:
  - rke2-canal
```

Listing 3: rke2 server - config.yaml

Cilium muss separat manifestiert werden:

```yaml
# /var/lib/rancher/rke2/server/manifests/rke2-cilium-config.yaml
---
apiVersion: helm.cattle.io/v1
kind: HelmChartConfig
metadata:
  name: rke2-cilium
  namespace: kube-system
spec:
  valuesContent: |-
    eni:
      enabled: true
```

Listing 4: rke2 server - cilium-config.yaml

Das Package kann nun installiert und aktiviert werden:

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_VERSION=v1.29.0+rke2r1 sh -
systemctl enable rke2-server.service
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:

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" INSTALL_RKE2_VERSION=v1
    .29.0+rke2r1 sh -
systemctl enable rke2-agent.service
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:

```
# /etc/rancher/rke2/config.yaml
server: https://10.0.20.97:9345
token: K1042bf32f28282edad37cbac4b77ccfa1cd44a26f0ea2c19111ed664013954a326::server
    :7a430a28b29501b778543f0882a156b8
```

Listing 7: rke2 agent - config.yaml

Nun muss der Dienst restartet werden

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

Listing 8: -rke2 agent service restart

VI.I.II.III Cluster Konfiguration

VI.I.II.III.I server

Auch für Kubernetes und die Pots müssen die Proxy-Einstellungen gemacht werden:

```
nano /etc/default/rke2-server
HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16

CONTAINERD_HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
CONTAINERD_HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
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 Netzwerkteil 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:

```
nano /etc/iptables/rules.v4

# Generated by iptables-save v1.8.9 (nf_tables)
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
-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
-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
-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
-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
-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
-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
-A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.75" -j ACCEPT
-A INPUT -s 10.0.9.36/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.36" -j ACCEPT
-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
-A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6443 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 9345 -j ACCEPT
COMMIT
# Completed

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:

```
root@sks1183:~# mkdir /var/local-path-provisioner
root@sks1183:~# chmod -R 777 /var/local-path-provisioner/

root@sks1184:~# mkdir /var/local-path-provisioner
root@sks1184:~# chmod -R 777 /var/local-path-provisioner/

root@sks1185:~# mkdir /var/local-path-provisioner
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:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: local-path-config
  namespace: local-path-storage
data:
  config.json: |-
        {
                "nodePathMap":[
                {
                        "node":"DEFAULT_PATH_FOR_NON_LISTED_NODES",
                        "paths":["/var/local-path-provisioner"]
                }
                ]
        }
  setup: |-
        #!/bin/sh
        set -eu
        mkdir -m 0777 -p "$VOL_DIR"
  teardown: |-
        #!/bin/sh
        set -eu
        rm -rf "$VOL_DIR"
  helperPod.yaml: |-
        apiVersion: v1
        kind: Pod
        metadata:
          name: helper-pod
        spec:
          priorityClassName: system-node-critical
          tolerations:
```

```
            - key: node.kubernetes.io/disk-pressure
              operator: Exists
              effect: NoSchedule
          containers:
          - name: helper-pod
            image: busybox
```

Listing 17: local-path-provisioner definieren

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

```
kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/rke2/local-path-
    provisioner.yaml
```

Listing 18: local-path-storage aktualisieren

VI.I.II.VI MetalIB - Proxy / Load Balancer

MetalIB musste installiert werden:

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.4/config/
    manifests/metallb-native.yaml
```

Listing 19: MetalIB installieren

Das Konfigurationsmanifest wurde eingespielt:

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: distributed-sql
  namespace: metallb-system
spec:
  addresses:
  - 10.0.20.106-10.0.20.106
  - 10.0.20.150-10.0.20.155
---
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2adv
  namespace: metallb-system
spec:
  ipAddressPools:
  - distributed-sql
```

Listing 20: MetalIB konfigurieren

Das Manifest musste danach eingespielt werden:

```
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:

```
echo "- - -" | tee /sys/class/scsi_host/host*/scan && lsblk

fdisk /dev/sdb
  n -> all default values
  t -> 8e
  p -> Kontrolle
  w

pvcreate /dev/sdb1 && \
vgcreate vgdata /dev/sdb1 && \
lvcreate -l 100%FREE -n lvdata vgdata && \
mkfs.ext4 /dev/vgdata/lvdata && \
mkdir -p /srv/data && \
cp /etc/fstab /tmp/fstab.bak && \
echo "/dev/vgdata/lvdata  /srv/data ext4  defaults  0 0" >> /etc/fstab && \
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:

```
root@sks1183:~# mkdir -p /srv/data/local-path-provisioner
root@sks1183:~# chmod 777 /srv/data/local-path-provisioner

root@sks1184:~# mkdir -p /srv/data/local-path-provisioner
root@sks1184:~# chmod 777 /srv/data/local-path-provisioner

root@sks1185:~# mkdir -p /srv/data/local-path-provisioner
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:

```
kind: ConfigMap
apiVersion: v1
metadata:
  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:

```
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:

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: yb-storage
provisioner: rancher.io/local-path
parameters:
  nodePath: /var/local-path-provisioner
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: yb-storage-pv
  labels:
    type: local
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 3Gi
  storageClassName: "yb-storage"
  hostPath:
    path: /var/local-path-provisioner
```

Listing 26: yugabyteDB - StorageClass setzen

Die Storage Class und das PersistentVolume muss aktiviert werden:

```
gramic@cks4040:~$ kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/
    yugabytedb/yugabytedb/storageclass.yaml
storageclass.storage.k8s.io/yb-storage unchanged
persistentvolume/yb-storage-pv created
```

Listing 27: yugabyteDB - StorageClass / PersistentVolume aktivieren

VI.II.II Installation

Zuerst muss ein Namespace erstellt werden:

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

Listing 28: yugabyteDB - Namespace

```yaml
# Default values for yugabyte.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
Component: "yugabytedb"

fullnameOverride: ""
nameOverride: ""

Image:
  repository: "yugabytedb/yugabyte"
  tag: 2.20.2.1-b3
  pullPolicy: IfNotPresent
  pullSecretName: ""

storage:
  ephemeral: false  # will not allocate PVs when true
  master:
    count: 1
    size: 3Gi
    storageClass: "yb-storage"
  tserver:
    count: 1
    size: 3Gi
    storageClass: "yb-storage"

resource:
  master:
    requests:
      cpu: "1"
      memory: 2Gi
    limits:
      cpu: "1"
      ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating from these formats
      ## may result in setting an incorrect value for the 'memory_limit_hard_bytes' flag.
      ## Avoid using floating numbers for the numeric part of 'memory'. Doing so may lead to
      ## the 'memory_limit_hard_bytes' being set to 0, as the function expects integer values.
      memory: 2Gi
  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 boostrapSelfsigned=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: "
      LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ0lCQVRBTkJna3Foa2lHOXcwQkFRc0Z
      ="
107     key: "
      LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGt
      ="
108   ## When tls.certManager.enabled=false
109   ## nodeCert and clientCert will be used only when rootCA.key is empty.
110   ## Will be ignored and genSignedCert will be used to generate
111   ## node and client certs if rootCA.key is provided.
112   ## cert and key are base64 encoded content of certificate and key.
113   nodeCert:
114     cert: ""
115     key: ""
116   clientCert:
117     cert: ""
118     key: ""
119
120 gflags:
121   master:
122     default_memory_limit_to_ram_ratio: 0.85
```

```
  tserver: {}
#   use_cassandra_authentication: false

PodManagementPolicy: Parallel

enableLoadBalancer: true

ybc:
  enabled: false
  ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers
    /#resource-requests-and-limits-of-pod-and-container
  ## Use the above link to learn more about Kubernetes resources configuration.
  # resources:
  #   requests:
  #     cpu: "1"
  #     memory: 1Gi
  #   limits:
  #     cpu: "1"
  #     memory: 1Gi

ybCleanup: {}
  ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers
    /#resource-requests-and-limits-of-pod-and-container
  ## Use the above link to learn more about Kubernetes resources configuration.
  # resources:
  #   requests:
  #     cpu: "1"
  #     memory: 1Gi
  #   limits:
  #     cpu: "1"
  #     memory: 1Gi

domainName: "cluster.local"

serviceEndpoints:
  - name: "yb-master-ui"
    type: LoadBalancer
    annotations: {}
    clusterIP: ""
    ## Sets the Service's externalTrafficPolicy
    externalTrafficPolicy: ""
    app: "yb-master"
    loadBalancerIP: ""
    ports:
      http-ui: "7000"

  - name: "yb-tserver-service"
    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:
```

```
    enabled: true
    port: "http-ui"
    interval: ""
    path: "/prometheus-metrics"
  ycql:
    enabled: true
    port: "http-ycql-met"
    interval: ""
    path: "/prometheus-metrics"
  ysql:
    enabled: true
    port: "http-ysql-met"
    interval: ""
    path: "/prometheus-metrics"
  yedis:
    enabled: true
    port: "http-yedis-met"
    interval: ""
    path: "/prometheus-metrics"

  commonMetricRelabelings:
  # https://git.io/JJW5p
  # Save the name of the metric so we can group_by since we cannot by __name__
    directly...
  - sourceLabels: ["__name__"]
    regex: "(.*)"
    targetLabel: "saved_name"
    replacement: "$1"
  # The following basically retrofit the handler_latency_* metrics to label format
    .
  - sourceLabels: ["__name__"]
    regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(.*)"
    targetLabel: "server_type"
    replacement: "$1"
  - sourceLabels: ["__name__"]
    regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(.*)"
    targetLabel: "service_type"
    replacement: "$2"
  - sourceLabels: ["__name__"]
    regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(_sum|_count)?"
    targetLabel: "service_method"
    replacement: "$3"
  - sourceLabels: ["__name__"]
    regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(_sum|_count)?"
    targetLabel: "__name__"
    replacement: "rpc_latency$4"

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
  ## Example:
  # extraVolumeMounts:
  # - name: custom-nfs-vol
  #   mountPath: /home/yugabyte/nfs-backup
  extraVolumeMounts: []

  ## Set service account for master DB pods. The service account
  ## should exist in the namespace where the master DB pods are brought up.
  serviceAccount: ""

  ## Memory limit hard % (between 1-100) of the memory limit.
  memoryLimitHardPercentage: 85


tserver:
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
    affinity-v1-core
  ## This might override the default affinity from service.yaml
  # To successfully merge, we need to follow rules for merging nodeSelectorTerms
    that kubernentes
  # has. Each new node selector term is ORed together, and each match expression
    or match field in
  # a single selector is ANDed together.
  # This means, if a pod needs to be scheduled on a label 'custom_label_1' with a
    value
  # 'custom_value_1', we need to add this 'subterm' to each of our pre-defined
    node affinity
  # terms.
  #
  # Pod anti affinity is a simpler merge. Each term is applied separately, and the
     weight is tracked.
  # The pod that achieves the highest weight is selected.
  ## Example.
  # affinity:
  #   podAntiAffinity:
  #     requiredDuringSchedulingIgnoredDuringExecution:
  #     - labelSelector:
  #         matchExpressions:
  #         - key: app
  #           operator: In
  #           values:
  #           - "yb-tserver"
  #       topologyKey: kubernetes.io/hostname
  # For further examples, see examples/yugabyte/affinity_overrides.yaml
  affinity: {}

  ## 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
```

```
  skipAll: false
  # Set to true to skip port bind checks
  skipBind: false

  ## Set to true to skip ulimit verification
  ## SkipAll has higher priority
  skipUlimit: false

## Pod securityContext
## Ref: https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#security-context
## The following configuration runs YB-Master and YB-TServer as a non-root user
podSecurityContext:
  enabled: false
  ## Mark it false, if you want to stop the non root user validation
  runAsNonRoot: true
  fsGroup: 10001
  runAsUser: 10001
  runAsGroup: 10001

## Added to handle old universe which has volume annotations
## K8s universe <= 2.5 to >= 2.6
legacyVolumeClaimAnnotations: false
```

Listing 29: yugabyteDB - Helm Chart Manifest

Die Installation erfolgt dann wie folgt:

```
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:

```
helm delete yw-evaluation -n yb-platform
kubectl delete pvc --namespace yb-platform -l app=yb-master
kubectl delete pvc --namespace yb-platform -l app=yb-tserver
kubectl delete namespace yb-platform
kubectl delete pv yb-storage-pv
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 Aiffinity gesetzt wird, damit die Persistence Volumes auf den Node geschrieben werden:

```yaml
# https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/prepare-environment/kubernetes/#configure-storage-class
# https://github.com/rancher/local-path-provisioner
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: yb-storage
provisioner: rancher.io/local-path
parameters:
  nodePath: /srv/data/local-path-provisioner
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: yb-storage-pv
  labels:
    type: local
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  storageClassName: "yb-storage"
  hostPath:
    path: /srv/data/local-path-provisioner
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - sks1183
          - sks1184
          - sks1185
```

Listing 32: yugabyteDB - StorageClass setzen

```
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:

```
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:

```yaml
# Default values for yugabyte.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.
Component: "yugabytedb"

fullnameOverride: ""
nameOverride: ""

Image:
  repository: "yugabytedb/yugabyte"
  tag: 2.20.2.1-b3
  pullPolicy: IfNotPresent
  pullSecretName: ""

storage:
  ephemeral: false  # will not allocate PVs when true
  master:
    count: 1
    size: 2Gi
    storageClass: "yb-storage"
  tserver:
    count: 1
    size: 240Gi
    storageClass: "yb-storage"

resource:
  master:
    requests:
      cpu: "2"
      memory: 1Gi
    limits:
      cpu: "2"
      memory: 1Gi
  tserver:
    requests:
      cpu: "2"
      memory: 12Gi
    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: "
      LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ0lCQVRBTkJna3Foa2lHOXcwQkFRc0Z
      ="
91      key: "
      LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGt
      ="
92    nodeCert:
93      cert: ""
94      key: ""
95    clientCert:
96      cert: ""
97      key: ""
98
99  gflags:
100   master:
101     default_memory_limit_to_ram_ratio: 0.85
102   tserver:
103     ysql_beta_features: true                      # gramic, wird üfr vacuum
      öbentigt
104   follower_unavailable_considered_failed_sec: 300 # gramic, autobalancing auf 5
      minuten heruntersetzen
105
106 PodManagementPolicy: Parallel
107
108 enableLoadBalancer: true
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
```

```yaml
  tolerations: []

  ## Extra volumes
  ## extraVolumesMounts are mandatory for each extraVolumes.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
  extraVolumes: []

  ## Extra volume mounts
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volumemount-v1-core
  extraVolumeMounts: []

  ## Set service account for master DB pods. The service account
  ## should exist in the namespace where the master DB pods are brought up.
  serviceAccount: ""

  ## Memory limit hard % (between 1-100) of the memory limit.
  memoryLimitHardPercentage: 85


tserver:
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
  ## This might override the default affinity from service.yaml
  affinity: {}

  ## Extra environment variables passed to the TServer pods.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
  extraEnv: []

  ## secretEnv variables are used to expose secrets data as env variables in the tserver pods.
  ## If namespace field is not specified we assume that user already
  ## created the secret in the same namespace as DB pods.
  secretEnv: []

  ## Annotations to be added to the TServer pods.
  podAnnotations: {}

  ## Labels to be added to the TServer pods.
  podLabels: {}

  ## Tolerations to be added to the TServer pods.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
  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
```

```
  skipUlimit: false

## Pod securityContext
## Ref: https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod-v1/#security-context
## The following configuration runs YB-Master and YB-TServer as a non-root user
podSecurityContext:
  enabled: false
  ## Mark it false, if you want to stop the non root user validation
  runAsNonRoot: true
  fsGroup: 10001
  runAsUser: 10001
  runAsGroup: 10001

## Added to handle old universe which has volume annotations
## K8s universe <= 2.5 to >= 2.6
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 Tablespaces 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:

```
select
    cloud,
    region,
    zone
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:

```
--  Tabelle pgbench_eval_bench erstellen
drop database if exists pgbench_eval_bench;
create database pgbench_eval_bench;

--  Tablespace für Indices
drop tablespace if exists eval_index_tablespace;
CREATE TABLESPACE eval_index_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');
```

```

--  Genereller Tablespace erstellen
drop tablespace if exists eval_data_tablespace;
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:

```
select
  table_name,
  pg_size_pretty(pg_total_relation_size(quote_ident(table_name))),
  pg_total_relation_size(quote_ident(table_name))
from information_schema.tables
where table_schema = 'public'
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:

```
--  self-healing-Tabelle
create database self_healing_test;

--  Tablespace für Indices
drop tablespace if exists self_healing_indices_tablespace;
CREATE TABLESPACE self_healing_indices_tablespace WITH (replica_placement='{"
    num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"
    datacenter1","zone":"rack1","min_num_replicas":3}]}');

--  Genereller Tablespace erstellen
drop tablespace if exists self_healing_datas_tablespace;
```

```
10 CREATE TABLESPACE self_healing_datas_tablespace WITH (replica_placement='{"
    num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"
    datacenter1","zone":"rack1","min_num_replicas":3}]}');
11
12 --  longtext Tablespace erstellen
13 drop tablespace if exists self_healing_longtexts_tablespace;
14 CREATE TABLESPACE self_healing_longtexts_tablespace WITH (replica_placement='{"
    num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"
    datacenter1","zone":"rack1","min_num_replicas":3}]}');
15
16 --  Rollen erstellen
17 drop role if exists hrm;
18 create role hrm;
19 drop role if exists accountands;
20 create role accountands;
21 drop role if exists customer_service_officers;
22 create role customer_service_officers;
23 drop role if exists legal_affairs;
24 create role legal_affairs;
25
26 --  User erstellen
27 drop user if exists hrm_1;
28 drop user if exists hrm_2;
29 create user hrm_1 with password 'hrm1' role hrm;
30 create user hrm_2 with password 'hrm2' role hrm;
31
32 drop user if exists cso_1;
33 drop user if exists cso_2;
34 create user cso_1 with password 'cso1'role customer_service_officers;
35 create user cso_2 with password 'cso2' role customer_service_officers;
36
37 drop user if exists la_1;
38 drop user if exists la_2;
39 create user la_1 with password 'la1' role legal_affairs;
40 create user la_2 with password 'la2' role legal_affairs;
41
42 --  Schemas erstellen
43 drop schema if exists hrm;
44 create schema hrm authorization hrm;
45 drop schema if exists accountands;
46 create schema accountands authorization accountands;
47 drop schema if exists customer_service_officers;
48 create schema customer_service_officers authorization customer_service_officers;
49 drop schema if exists generell;
50 create schema generell;
51
52 --  GRANTS erstellen
53 grant all on all tables in schema hrm to legal_affairs;
```

```
54 grant all on all tables in schema accountands to legal_affairs;
55 grant all on all tables in schema customer_service_officers to legal_affairs;
56 grant all on all tables in schema generell to legal_affairs;
57 grant all on all tables in schema hrm to yadmin;
58 grant all on all tables in schema accountands to yadmin;
59 grant all on all tables in schema customer_service_officers to yadmin;
60 grant all on all tables in schema generell to yadmin;
61
62 -- self_healing_accounts für Schema customer_service_officers
63 drop table if exists customer_service_officers.self_healing_accounts;
64 create table customer_service_officers.self_healing_accounts (
65     account_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 ) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
75 create unique index accounts_personal_mark on customer_service_officers.
       self_healing_accounts(firstname, lastname, birthday) tablespace
       self_healing_indices_tablespace;
76
77 --  self_healing_employees für Schema hrm
78 drop table if exists hrm.self_healing_employees;
79 create table hrm.self_healing_employees (
80     employees_id int primary key,
81     firstname varchar(255) not null,
82     lastname varchar(255) not null,
83     birthday date not null,
84     postal_code varchar(50),
85     street varchar(255),
86     country_code varchar(2),
87     phone varchar(25),
88     mail varchar(255) check (mail like '%@%')
89 ) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
90 create unique index employees_personal_mark on hrm.self_healing_employees(
       firstname, lastname, birthday) tablespace self_healing_indices_tablespace;
91
92 --  self_healing_accountand_protocol für Schema accountands
93 drop table if exists accountands.self_healing_accountand_protocol;
94 create table accountands.self_healing_accountand_protocol (
95        acc_protocol_id int primary key,
96        description varchar(100) not null,
97        protocol_date date not null,
98        employees_id int not null,
```

```
99        rapport TEXT,
100       foreign key (employees_id) references hrm.self_healing_employees(
    employees_id) on update restrict on delete restrict
101 ) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;
102
103 --  self_healing_intranet für public Schema
104 drop table if exists generell.self_healing_intranet;
105 create table generell.self_healing_intranet (
106       intranet_id int primary key,
107       content text
108 ) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;
109
110 --  self_healing_intranet für public Schema
111 drop table if exists generell.self_healing_intranet_users;
112 create table generell.self_healing_intranet_users (
113       intranet_user_id int primary key,
114       employees_id int not null,
115       foreign key (employees_id) references hrm.self_healing_employees(
    employees_id) on update restrict on delete restrict
116 );
117 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_service_officers.self_healing_accounts (account_id, firstname
    , lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
2 insert into customer_service_officers.self_healing_accounts (account_id, firstname
    , lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
3 insert into customer_service_officers.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');
```

```sql

insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');

insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(100, 100);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(200, 200);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(300, 300);

select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet_users;
```

Listing 40: YugabyteDB - Self Healing Tests - Init Data

Während dem Failover-Test müssen Daten beschrieben werden:

```sql
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');

insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');

insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (400, 'bla', '07.04.2024', 200, 'blabla');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (500, 'yada', '07.04.2024', 600, 'ydayadyada');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (1000, 'something', '07.04.2024', 300, 'something');

insert into generell.self_healing_intranet(intranet_id, content) VALUES (200, 'yadada');
```

```
insert into generell.self_healing_intranet(intranet_id, content) VALUES (600, 'bla bla');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (900, 'talking and talking');

insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(400, 400);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(500, 500);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(600, 600);

select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet;
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:

```
select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet;
select * from generell.self_healing_intranet_users;

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');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (1100, 'talking and talking');

select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet;
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:

```
root@sks9016:~# wget https://downloads.yugabyte.com/releases/2.21.0.0/yugabyte
    -2.21.0.0-b545-linux-x86_64.tar.gz

```

Listing 43: sks9016 - Download yugabyteDB On-Premise

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

```
root@sks9016:/opt# tar xvfz yugabyte-2.21.0.0-b545-linux-x86_64.tar.gz && cd
    yugabyte-2.21.0.0/
...
root@sks9016:/opt/yugabyte-2.21.0.0# ./bin/post_install.sh

```

Listing 44: sks9016 - Installation yugabyteDB On-Premise

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

```
root@sks9016:/opt/yugabyte-2.21.0.0# cd /opt/yugabyte-2.21.0.0/postgres/bin/
root@sks9016:/opt/yugabyte-2.21.0.0/postgres/bin# ./ysqlsh "host=10.0.20.106 user=
    yadmin"
Password for user yadmin:
ysqlsh (11.2-YB-2.21.0.0-b0)
Type "help" for help.

No entry for terminal type "xterm-256color";
using dumb terminal settings.
yugabyte=# exit

```

Listing 45: sks9016 - Check yugabyteDB On-Premise

Damit ist der Benchmarking-Server ready.

VI.IV Stackgres mit Citus

VI.IV.I Prerequisites

VI.IV.I.I StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```
# https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/
    prepare-environment/kubernetes/#configure-storage-class
# https://github.com/rancher/local-path-provisioner
```

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: stackgres-storage
provisioner: rancher.io/local-path
parameters:
  nodePath: /srv/data/local-path-provisioner
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: stackgres-storage-pv
  labels:
    type: local
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  storageClassName: "stackgres-storage"
  hostPath:
    path: /srv/data/local-path-provisioner
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - sks1183
          - sks1184
          - sks1185
```

Listing 46: StackGres-Citus - StorageClass setzen

Die Storage Class und das PercistenVolume muss aktiviert werden:

```
gramic@cks4040:~$ kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/
    stackgres_citus/stackgres_citus/storageclass.yaml
storageclass.storage.k8s.io/stackgres-storage created
persistentvolume/stackgres-storage-pv created
```

Listing 47: StackGres-Citus - StorageClass / PersistentVolume aktivieren

VI.IV.II Installation

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

Listing 48: StackGres-Citus - Namespace

Das Manifest beinhaltet nur die wichtigsten Parameter.
Der Grossteil wird über das Cluster-Deployment gesetzt:

```
# -- The container registry host (and port) where the images will be pulled from.
containerRegistry: quay.io
# -- Image pull policy used for images loaded by the Operator
imagePullPolicy: "IfNotPresent"
# Section to configure Operator Installation ServiceAccount
serviceAccount:
  # -- If `true` the Operator Installation ServiceAccount will be created
  create: true
  # -- Section to configure Operator ServiceAccount annotations
  annotations: {}
  # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
  repoCredentials: []

# Section to configure Operator Pod
operator:
  # Section to configure Operator image
  image:
    # -- Operator image name
    name: "stackgres/operator"
    # -- Operator image tag
    tag: "1.9.0"
    # -- Operator image pull policy
    pullPolicy: "IfNotPresent"
  # -- Operator Pod annotations
  annotations: {}
  # -- Operator Pod resources. See https://kubernetes.io/docs/reference/generated/
    kubernetes-api/v1.27/#resourcerequirements-v1-core
#  resources: {}
  resources:
     requests:
       cpu: "1"
       memory: 1Gi
     limits:
       cpu: "1"
       memory: 1Gi
  # -- Operator Pod node selector
  nodeSelector: {}
  # -- Operator Pod tolerations. See https://kubernetes.io/docs/reference/
    generated/kubernetes-api/v1.27/#toleration-v1-core
  tolerations: []
  # -- Operator Pod affinity. See https://kubernetes.io/docs/reference/generated/
    kubernetes-api/v1.27/#affinity-v1-core
  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-manager.io/v1.CertificatePrivateKey
203     encoding: PKCS1
204     # -- 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.CertificatePrivateKey
205     size: 2048
206
207 # Section to configure RBAC for Web Console admin user
208 rbac:
209   # -- When set to `true` the admin user is assigned the `cluster-admin`
    ClusterRole by creating
210   #   ClusterRoleBinding.
211   create: true
212
213 # Section to configure Web Console authentication
214 authentication:
215   # -- Specify the authentication mechanism to use. By default is `jwt`, see https
    ://stackgres.io/doc/latest/api/rbac#local-secret-mechanism.
216   #   If set to `oidc` then see https://stackgres.io/doc/latest/api/rbac/#openid-
    connect-provider-mechanism.
217   type: jwt
218   # -- (boolean) When `true` will create the secret used to store the `admin` user
     credentials to access the UI.
219   createAdminSecret: true
220   # -- The admin username that will be required to access the UI
221   user: admin
222   # -- (string) The admin password that will be required to access the UI
223   #password: "TES2&Daggerfall"
224   password:
225   # Section to configure Web Console OIDC authentication
226   oidc:
227     # tlsVerification -- (string) Can be one of `required`, `certificate-
    validation` or `none`
228 # Section to configure Prometheus integration.
229 prometheus:
230
231   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 follwing 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%2
      Fsproxy.sivc.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 estensions 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
```

```
      volumeMounts: []
```

Listing 49: StackGres-Citus - Helm Chart Manifest

Die Installation erfolgt dann wie folgt:

```
helm install -n sg-platform stackgres-operator stackgres-charts/stackgres-operator
    -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus
    /values.yaml
```

Listing 50: StackGres-Citus - Installation

Damit nun aber der Cluster sauber läuft und auf den Cluster zugegriffen werden kann, müssen folgende Steps ebenfalls nachträglich ausgeführt werden:

```
# Check if the operator was successfully deployed and is available:
kubectl describe deployment -n sg-platform stackgres-operator
kubectl wait -n sg-platform deployment/stackgres-operator --for condition=
    Available
# Check if the restapi was successfully deployed and is available:
kubectl describe deployment -n sg-platform stackgres-restapi
kubectl wait -n sg-platform deployment/stackgres-restapi --for condition=Available
# To access StackGres Operator UI from localhost, run the below commands:
POD_NAME=$(kubectl get pods --namespace sg-platform -l "stackgres.io/restapi=true"
    -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward "$POD_NAME" 8443:9443 --namespace sg-platform
```

Listing 51: 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:

```
kubectl get secret -n sg-platform stackgres-restapi-admin --template '{{ printf "
    username = %s\n" (.data.k8sUsername | base64decode) }}'
```

Listing 52: StackGres-Citus - System Username

Dieses lässt sich mit folgendem Command auslesen:

```
kubectl get secret -n sg-platform stackgres-restapi-admin --template '{{ printf "
    password = %s\n" (.data.clearPassword | base64decode) }}'
```

Listing 53: StackGres-Citus - System Passwort

Am Schluss sollte das Passwort aber noch gesäubert werden:

```
kubectl patch secret --namespace sg-platform stackgres-restapi-admin --type json -
    p '[{"op":"remove","path":"/data/clearPassword"}]'
```

Listing 54: StackGres-Citus - System Passwort Cleanup

VI.IV.III Deployment - Benchmarking

Zuerts wurde das Instanz-Profil für den Coordinator und die Shards deployt:

```yaml
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-pgbench-coordinator
spec:
  cpu: "4"
  memory: "4Gi"
```

Listing 55: StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator

```yaml
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-pgbench-shard
spec:
  cpu: "4"
  memory: "8Gi"
```

Listing 56: StackGres-Citus - Benchmarking - SGInstanceProfile Shard

Deployt wird ebenfalls via `kubectl`:

```
kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/
    stackgres_citus/SGInstanceProfile_pgbench_coord.yaml
kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/
    stackgres_citus/SGInstanceProfile_pgbench_shard.yaml
```

Listing 57: StackGres-Citus - Benchmarking - Instanz-Profil Deploy

Auf ein Deployment einer `SGPostgresConfig` wurde verzichtet, da für Patroni in den ersten drei Benchmarks keine spezifischen Anpassungen erfuhr.
Das Manifest für den Benchmark-Cluster sieht entsprechend den vorgaben wie folgt aus:

```yaml
apiVersion: stackgres.io/v1alpha1
kind: SGShardedCluster
metadata:
  name: sg-pgbench
  namespace: sg-platform
spec:
  type: citus
```

```
  database: pgbench_eval_bench
  postgres:
    version: '16'
  coordinator:
    instances: 1
    pods:
      persistentVolume:
#         size: '75Gi'
        size: '230Gi'
        storageClass: "stackgres-storage"
      disableConnectionPooling: true
    sgInstanceProfile: "sg-pgbench-coordinator"
  shards:
    clusters: 3
    instancesPerCluster: 1
    pods:
      persistentVolume:
        size: '75Gi'
        storageClass: "stackgres-storage"
    sgInstanceProfile: "sg-pgbench-shard"
  postgresServices:
    coordinator:
      primary:
        type: LoadBalancer
      any:
        type: LoadBalancer
    shards:
      primaries:
        type: LoadBalancer
  metadata:
    annotations:
      primaryService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.106
      replicasService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.153
        externalTrafficPolicy: "Cluster"
  profile: "testing"
```

Listing 58: StackGres-Citus - Benchmarking - SGShardedCluster

Der Deploy:

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

Listing 59: 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:

```
kubectl get secrets -n sg-platform sg-pgbench -o jsonpath='{.data.superuser-
    password}' | base64 -d
```

Listing 60: StackGres-Citus - Benchmark DB Passwort

VI.IV.IV Deployment - Self Healing Tests

Auch hier wurde zuerst das Instanz-Profil für den Coordinator und die Shards deployt:

```yaml
#Not needed actualy
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-self-healing-coordinator
spec:
  cpu: "1"
  memory: "2Gi"
```

Listing 61: StackGres-Citus - Self Healing Testing - SGInstanceProfile Coordinator

```yaml
#Not needed actualy
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-self-healing-shard
spec:
  cpu: "1"
  memory: "2Gi"
```

Listing 62: StackGres-Citus - Self Healing Testing - SGInstanceProfile Shard

Deployt wird ebenfalls via `kubectl`:

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

Listing 63: 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:

```yaml
apiVersion: stackgres.io/v1alpha1
kind: SGShardedCluster
metadata:
  name: sg-healing-test
  namespace: sg-platform
```

```
spec:
  type: citus
  database: self_healing_test
  postgres:
    version: '16'
  coordinator:
    syncInstances: 3
    replication: "strict-sync"
    pods:
      persistentVolume:
        size: '2Gi'
        storageClass: "stackgres-storage"
    sgInstanceProfile: "sg-self-healing-coordinator"
  shards:
    clusters: 3
    instancesPerCluster: 3
    pods:
      persistentVolume:
        size: '5Gi'
        storageClass: "stackgres-storage"
    sgInstanceProfile: "sg-self-healing-shard"
  postgresServices:
    coordinator:
      primary:
        type: LoadBalancer
      any:
        type: LoadBalancer
    shards:
      primaries:
        type: LoadBalancer
  metadata:
    annotations:
      primaryService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.152
      replicasService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.151
        externalTrafficPolicy: "Cluster"
  profile: "testing"
```

Listing 64: StackGres-Citus - Self Healing Testing - SGShardedCluster

Der Deploy:

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

Listing 65: StackGres-Citus - Self Healing Testing - Cluster Deploy

Das Passwort für den Cluster (IP 10.0.20.152) muss ebenfalls ausgelesen werden:

```
kubectl get secrets -n sg-platform sg-healing-test -o jsonpath='{.data.superuser-
    password}' | base64 -d
```

Listing 66: StackGres-Citus - Self Healing Testing DB Passwort

VI.IV.V Rekonfiguration mit 250GiB Storage

VI.IV.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:

```
helm delete stackgres-operator --namespace sg-platform
kubectl delete namespace sg-platform
kubectl delete pv stackgres-storage-pv
kubectl delete storageclass stackgres-storage
kubectl delete pvc --namespace sg-platform
kubectl delete pvc --namespace sg-platform
```

Listing 67: StackGres-Citus - Deinstallieren

VI.IV.V.II StorageClass setzen

```
# https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/
    prepare-environment/kubernetes/#configure-storage-class
# https://github.com/rancher/local-path-provisioner
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: stackgres-storage-big
provisioner: rancher.io/local-path
parameters:
  nodePath: /srv/data/local-path-provisioner
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: stackgres-storage-pv
  labels:
    type: local
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
```

```
    storage: 340Gi
  storageClassName: "stackgres-storage"
  hostPath:
    path: /srv/data/local-path-provisioner
```

Listing 68: StackGres-Citus - StorageClass setzen

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

Listing 69: StackGres-Citus - StorageClass / PersistentVolume Grosse Volumes aktivieren

VI.IV.V.III Installation

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

Listing 70: StackGres-Citus - Namespace 250GiB

Natürlich muss auch wieder das helm-Manifest deployt werden:

```
helm install -n sg-platform stackgres-operator stackgres-charts/stackgres-operator
     -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/stackgres_citus
    /values.yaml
```

Listing 71: StackGres-Citus - Installation 250GiB

VI.IV.V.IV Deployment - Benchmarking mit 250GiB

Die Ressourcen für die Shards mussten erhöht werden:

```
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-pgbench-coordinator
spec:
  cpu: "4"
  memory: "4Gi"
```

Listing 72: StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB

```
apiVersion: stackgres.io/v1
kind: SGInstanceProfile
metadata:
  namespace: sg-platform
  name: sg-pgbench-shard
spec:
  cpu: "4"
  memory: "12Gi"
```

Listing 73: StackGres-Citus - Benchmarking - SGInstanceProfile Shard 250GiB

```
kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/
    stackgres_citus/SGInstanceProfile_pgbench_coord.yaml
kubectl apply -f /home/gramic/PycharmProjects/rke2_settings/stackgres_citus/
    stackgres_citus/SGInstanceProfile_pgbench_shard.yaml
```

Listing 74: 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:

```yaml
apiVersion: stackgres.io/v1alpha1
kind: SGShardedCluster
metadata:
  name: sg-pgbench
  namespace: sg-platform
spec:
  type: citus
  database: pgbench_eval_bench
  postgres:
    version: '16'
  coordinator:
    instances: 1
    pods:
      persistentVolume:
        size: '230Gi'
        storageClass: "stackgres-storage"

      disableConnectionPooling: true
    sgInstanceProfile: "sg-pgbench-coordinator"
  shards:
    clusters: 2 # gramic, 19.04.2024: 250GiB Tabelle
    instancesPerCluster: 1
    pods:
      persistentVolume:
        size: '230Gi'
        storageClass: "stackgres-storage"
    sgInstanceProfile: "sg-pgbench-shard"
  postgresServices:
    coordinator:
      primary:
        type: LoadBalancer
      any:
        type: LoadBalancer
    shards:
      primaries:
        type: LoadBalancer
  metadata:
    annotations:
      primaryService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.106
```

```
      replicasService:
        metallb.universe.tf/loadBalancerIPs: 10.0.20.153
        externalTrafficPolicy: "Cluster"
  profile: "testing"
```

Listing 75: StackGres-Citus - Benchmarking - SGShardedCluster 250GiB

Der Deploy:

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

Listing 76: StackGres-Citus - Benchmark - Cluster Deploy 250GiB

VI.IV.VI SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle `pgbench_eval_bench` bereits beim Deployment erstellt. Allerdings ohne Tablespaces.
Daher sind keine weiteren Schritte an dieser Stelle notwendig, die Tabellen werden von `pgbench` beim Initialisieren erstellt.

VI.IV.VII 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 Tablespaces 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:

```
--  Rollen erstellen
drop role if exists hrm;
create role hrm;
drop role if exists accountands;
create role accountands;
drop role if exists customer_service_officers;
create role customer_service_officers;
drop role if exists legal_affairs;
create role legal_affairs;

--  User erstellen
drop user if exists hrm_1;
drop user if exists hrm_2;
create user hrm_1 with password 'hrm1' role hrm;
create user hrm_2 with password 'hrm2' role hrm;

```

```
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;
30 drop schema if exists accountands;
31 create schema accountands authorization accountands;
32 drop schema if exists customer_service_officers;
33 create schema customer_service_officers authorization customer_service_officers;
34 drop schema if exists generell;
35 create schema generell;
36
37 --  GRANTS erstellen
38 grant all on all tables in schema hrm to legal_affairs;
39 grant all on all tables in schema accountands to legal_affairs;
40 grant all on all tables in schema customer_service_officers to legal_affairs;
41 grant all on all tables in schema generell to legal_affairs;
42 grant all on all tables in schema hrm to postgres;
43 grant all on all tables in schema accountands to postgres;
44 grant all on all tables in schema customer_service_officers to postgres;
45 grant all on all tables in schema generell to postgres;
46
47  -- self_healing_accounts für Schema customer_service_officers
48 drop table if exists customer_service_officers.self_healing_accounts;
49 create table customer_service_officers.self_healing_accounts (
50     account_id int primary key,
51     firstname varchar(255) not null,
52     lastname varchar(255) not null,
53     birthday date not null,
54     postal_code varchar(50),
55     street varchar(255),
56     country_code varchar(2),
57     phone varchar(25),
58     mail varchar(255) check (mail like '%@%')
59 );
60 create unique index accounts_personal_mark on customer_service_officers.
    self_healing_accounts(firstname, lastname, birthday);
61 SELECT create_reference_table('customer_service_officers.self_healing_accounts');
62
63 --  self_healing_employees für Schema hrm
```

```
drop table if exists hrm.self_healing_employees;
create table hrm.self_healing_employees (
    employees_id int primary key,
    firstname varchar(255) not null,
    lastname varchar(255) not null,
    birthday date not null,
    postal_code varchar(50),
    street varchar(255),
    country_code varchar(2),
    phone varchar(25),
    mail varchar(255) check (mail like '%@%')
);
create unique index employees_personal_mark on hrm.self_healing_employees(
    firstname, lastname, birthday);
SELECT create_reference_table('hrm.self_healing_employees');

--  self_healing_accountand_protocol für Schema accountands
drop table if exists accountands.self_healing_accountand_protocol;
create table accountands.self_healing_accountand_protocol (
       acc_protocol_id int primary key,
       description varchar(100) not null,
       protocol_date date not null,
       employees_id int not null,
       rapport TEXT,
       foreign key (employees_id) references hrm.self_healing_employees(
    employees_id) on update restrict on delete restrict
);
SELECT create_reference_table('accountands.self_healing_accountand_protocol');

--  self_healing_intranet für public Schema
drop table if exists generell.self_healing_intranet;
create table generell.self_healing_intranet (
       intranet_id int primary key,
       content text
);
SELECT create_reference_table('generell.self_healing_intranet');

--  self_healing_intranet für public Schema
drop table if exists generell.self_healing_intranet_users;
create table generell.self_healing_intranet_users (
       intranet_user_id int primary key,
       employees_id int not null,
       foreign key (employees_id) references hrm.self_healing_employees(
    employees_id) on update restrict on delete restrict
);
create unique index intranet_unique_combi on generell.self_healing_intranet_users(
    intranet_user_id, employees_id);
```

```
SELECT create_reference_table('generell.self_healing_intranet_users');
```

Listing 77: StackGres-Citus - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
insert into customer_service_officers.self_healing_accounts (account_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');

insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (100, 'a', 'b', '01.01.2000');
insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (200, 'c', 'd', '01.01.2000');
insert into hrm.self_healing_employees (employees_id, firstname, lastname, birthday) VALUES (300, 'f', 'g', '01.01.2000');

insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (100, 'bla', '07.04.2024', 100, 'blabla');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (200, 'yada', '07.04.2024', 100, 'ydayadyada');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id, description, protocol_date, employees_id, rapport)  values (300, 'something', '07.04.2024', 300, 'something');

insert into generell.self_healing_intranet(intranet_id, content) VALUES (100, 'yadada');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (500, 'bla bla');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (1000, 'talking and talking');

insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(100, 100);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(200, 200);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id) values(300, 300);

select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet_users;
```

Listing 78: StackGres-Citus - Self Healing Tests - Init Data

Während dem Failover-Test müssen Daten beschrieben werden:

```
insert into customer_service_officers.self_healing_accounts (account_id, firstname
    , lastname, birthday) VALUES (400, 'i', 'j', '01.01.2005');
insert into customer_service_officers.self_healing_accounts (account_id, firstname
    , lastname, birthday) VALUES (500, 'k', 'l', '01.01.2003');
insert into customer_service_officers.self_healing_accounts (account_id, firstname
    , lastname, birthday) VALUES (600, 'm', 'n', '01.01.2001');

insert into hrm.self_healing_employees (employees_id, firstname, lastname,
    birthday) VALUES (400, 'i', 'j', '01.01.2005');
insert into hrm.self_healing_employees (employees_id, firstname, lastname,
    birthday) VALUES (500, 'k', 'l', '01.01.2003');
insert into hrm.self_healing_employees (employees_id, firstname, lastname,
    birthday) VALUES (600, 'm', 'n', '01.01.2001');

insert into accountands.self_healing_accountand_protocol (acc_protocol_id,
    description, protocol_date, employees_id, rapport)  values (400, 'bla', '
    07.04.2024', 200, 'blabla');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id,
    description, protocol_date, employees_id, rapport)  values (500, 'yada', '
    07.04.2024', 600, 'ydayadyada');
insert into accountands.self_healing_accountand_protocol (acc_protocol_id,
    description, protocol_date, employees_id, rapport)  values (1000, 'something',
     '07.04.2024', 300, 'something');

insert into generell.self_healing_intranet(intranet_id, content) VALUES (200, '
    yadada');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (600, 'bla
     bla');
insert into generell.self_healing_intranet(intranet_id, content) VALUES (900, '
    talking and talking');

insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(400, 400);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(500, 500);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(600, 600);

select * from customer_service_officers.self_healing_accounts;
select * from hrm.self_healing_employees;
select * from accountands.self_healing_accountand_protocol;
select * from generell.self_healing_intranet;
select * from generell.self_healing_intranet_users;
```

Listing 79: StackGres-Citus - Self Healing Tests - Failover Data

Nach dem Recovery müssen die Daten entsprechend vorhanden sein und es müssen weitere Daten beschrieben werden können:

```
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(1000, 400);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(2000, 500);
insert into generell.self_healing_intranet_users(intranet_user_id, employees_id)
    values(3000, 600);

select count(*) from customer_service_officers.self_healing_accounts;
select count(*) from hrm.self_healing_employees;
select count(*) from accountands.self_healing_accountand_protocol;
select count(*) from generell.self_healing_intranet;
select count(*) from generell.self_healing_intranet_users;
```

Listing 80: StackGres-Citus - Self Healing Tests - Recovery Data

VI.V Patroni

VI.V.I Prerequisites

Zuerst muss der Proxy gesetzt werden:

```
# sks1232 / sks1233 / sks1234
# Proxy setzen
# nano /etc/profile.d/proxy.sh
export https_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
export http_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
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
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
# source /etc/profile.d/proxy.sh
```

Listing 81: 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:

```
# sks1232 / sks1233 / sks1234
# apt-Proxy setzen
# nano /etc/apt/apt.conf.d/proxy.conf
Acauire::http::Proxy "http://sproxy.sivc.first-it.ch:8080";
Acauire::https::Proxy "http://sproxy.sivc.first-it.ch:8080";
Acquire::http::proxy::foreman.ksgr.ch "DIRECT";
```

Listing 82: 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.

```
# sks1232 / sks1233 / sks1234
# PostgreSQL Repository einbinden
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

# Ausloggen und wieder einloggen
apt update
```

Listing 83: Patroni - PostgreSQL einbinden

etcd wird als nächstes Installiert.
Hierzu muss zuerst das Repository von GitHub heruntergeladen werden:

VI.V.II Installation

VI.V.III Konfiguration

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` mitgeben werden.
Anschliessend werden die Benchmarks an sich ausgeführt.
Die Resultate werden weggeschrieben.

```
###################
#   1. Lauf       #
#   ca. 5GiB      #
###################
# Init
./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

# Benchmarking mixed
./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
./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
./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
./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

# Benchmarking dql
./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
./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
./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
./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

###################
#   2. Lauf       #
#   ca. 15GiB     #
###################
# Init
./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

# Benchmarking mixed
./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
./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
./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
./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

```

```
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
51 # Benchmarking dql
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_1_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_2_yugabytedb_dql_benchmark.txt
54 ./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin
       pgbench_eval_bench > /home/gramic/3_3_yugabytedb_dql_benchmark.txt
55 ./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
56
57
58 ###################
59 #   4. Lauf       #
60 #   ca. 250GiB    #
61 ###################
62 ./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
63
64 # Benchmarking mixed
```

```
./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
./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
./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -U yadmin
    pgbench_eval_bench > /home/gramic/4_3_yugabytedb_mixed_benchmark.txt
./ysql_bench -h 10.0.20.106 -p 5433 -c 25 -C -j 4 -v -t 280 -U yadmin
    pgbench_eval_bench > /home/gramic/4_4_yugabytedb_mixed_benchmark.txt

# Benchmarking dql
./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
./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
./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
./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 84: YugabyteDB - Benchmarking-Commands

Die grösse der Tabellen lässt sich wie folgt auslesen:

```
select
  table_name,
  pg_size_pretty(pg_total_relation_size(quote_ident(table_name))),
  pg_total_relation_size(quote_ident(table_name))
from information_schema.tables
where table_schema = 'public'
order by 3 desc;
```

Listing 85: yugabyteDB - Benchmarking - Table Size SQL

VII.II StackGres - Citus

Pro Lauf werden erst die Daten initialisiert werden.
Wichtig ist dabei, dass keine Tablespaces mitgegeben werden da diese nicht existieren.
Anschliessend werden die Benchmarks an sich ausgeführt.
Die Resultate werden weggeschrieben.

```
####################
#   1. Lauf        #
#   ca. 5GiB       #
####################
# Init
pgbench --host=10.0.20.106 --port=5432 --initialize --scale=400 --foreign-keys --
    fillfactor=100 --username=dtgvpf  --username=postgres pgbench_eval_bench

```

```
# Benchmarking mixed
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

# Benchmarking dql
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

###################
#   2. Lauf       #
#   ca. 15GiB     #
###################
# Init
pgbench --host=10.0.20.106 --port=5432 --initialize --scale=1200 --foreign-keys --
    fillfactor=100 --username=dtgvpf  --username=postgres pgbench_eval_bench

# Benchmarking mixed
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
# Benchmarking dql
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

###################
#   3. Lauf       #
#   ca. 50GiB     #
###################
pgbench --host=10.0.20.106 --port=5432 --initialize --scale=3999 --foreign-keys --
    fillfactor=100 --username=dtgvpf  --username=postgres pgbench_eval_bench

# Benchmarking mixed
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
# Benchmarking dql
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


###################
#   4. Lauf       #
#   ca. 250GiB    #
###################
pgbench --host=10.0.20.106 --port=5432 --initialize --scale=16784 --foreign-keys
    --fillfactor=100 --username=dtgvpf  --username=postgres pgbench_eval_bench

# Benchmarking mixed
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
```

```
# Benchmarking dql
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
```

Listing 86: StackGres-Citus - Benchmarking-Commands

Die grösse der Tabellen lässt sich wie folgt auslesen:

```
SELECT * FROM citus_tables;
```

Listing 87: 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:

Abbildung XXXIX: StackGres Testing - Node sks1184 down

Entsprechend wurden die Pods ebenfalls auf `terminating` gesetzt:

Abbildung XL: 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 XLI: StackGres Testing - Patroni Übersicht

Während dieser Zeit, ist die DB immer erreichbar:

```
184  After rebuild
185  select * from customer_service_officers.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
```

u...self_healing_accounts 2 | self_healing_test.hr...elf_healing_employees | self_

9 rows | Tx: Auto | DDL

	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 XLII: StackGres Testing - DB Zugriff

Allerdings werden längere Transaktionen geschlossen:

```
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  | 198.18.1.106:7433 | 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 XLIII: 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:

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     1%
MEM:     36%
<0> tail   <6> 1h   <shift-c> Clear          <t> Toggle Timestamp
<1> head            <c>       Copy           <w> Toggle Wrap
<2> 1m              <m>       Mark
<3> 5m              <ctrl-s>  Save
<4> 15m             <s>       Toggle AutoScroll
<5> 30m             <f>       Toggle FullScreen
Logs(yb-platform/yb-tserver-i)[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_usec=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    @    0x55e8ddfb8e39  yb::tablet::TransactionCoordinator::Impl::Poll()
yb-tserver    @    0x55e8dddada3c  yb::rpc::Poller::Poll()
yb-tserver    @    0x55e8dddd1d6e  _ZN5boost4asio6detail18completion_handlerIZN2yb3rpc9Scheduler4Impl11HandleTimerERKNS_6system10error_codeEEUlvE_NS0_10io_context19basic_executor_typeINSt3__19alloca
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 XLIV: YugabyteDB - Too big clock skew is detected

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

localhost:7000/table?id=000040000000300080000000000400b

phone		7	string NULLABLE VALUE			
mail		8	string NULLABLE VALUE			

Tablet ID	Partition	SplitDepth	State	Hidden	Message	RaftConfig
a2c249ed8ebc4f068c00735c9acc40c5	hash_split: [0xAAAA, 0xFFFF]	0	Running	0	Tablet reported with an active leader	• **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	• 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_REPLICATED_LEASE)** Cannot replicate lease for past **4** heartbeats UUID: d44aa240148c4e15a0684e4ac6dc9a9d
a6609fe063354432bcb38b435fe1413d	hash_split: [0x0000, 0x5554]	0	Running	0	Tablet reported with an active leader	• 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_REPLICATED_LEASE)** Cannot replicate lease for past **29** heartbeats UUID: d44aa240148c4e15a0684e4ac6dc9a9d

Abbildung XLV: 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.
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     1%
MEM:     36%
<0> all                                  <f> Show PortForward
<1> yb-platform                          <t> Transfer
<2> kube-public                          <y> YAML
<3> local-path-storage   <d> Describe   <shift-f> Port-Forward
<4> metallb-system       <e> Edit       <z> Sanitize
<5> kube-system          <?> Help       <s> Shell
                         <ctrl-k> Kill  <o> Show Node
Pods(yb-platform)[6]
NAME↑          PF  READY  STATUS            RESTARTS  CPU  MEM   %CPU/R  %CPU/L  %MEM/R  %MEM/L  IP            NODE      AGE
yb-master-0    Ⓕ   2/2    Running                0    11  238        0     n/a      23     n/a  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 XLVI: 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:

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     1%
MEM:     22%
<0> tail   <6> 1h   <shift-c> Clear          <t> Toggle Timestamp
<1> head            <c>       Copy           <w> Toggle Wrap
<2> 1m              <m>       Mark
<3> 5m              <ctrl-s>  Save
<4> 15m             <s>       Toggle AutoScroll
<5> 30m             <f>       Toggle FullScreen
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 XLVII: YugabyteDB - Too big clock skew is detected - tmaster

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     0%
MEM:     22%
<0> tail   <6> 1h   <shift-c> Clear          <t> Toggle Timestamp
<1> head            <c>       Copy           <w> Toggle Wrap
<2> 1m              <m>       Mark
<3> 5m              <ctrl-s>  Save
<4> 15m             <s>       Toggle AutoScroll
<5> 30m             <f>       Toggle FullScreen
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 XLVIII: 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 korrumpiert wird.

IX Exkurs Architekturen - Umsysteme und Prinzipien

IX.I Raft-Konsensus

IX.II local-path-provisioner

X Python Utils

X.I zotero.py

```
import json
import pybtex
import requests
import os
from pybtex.database import BibliographyData, Entry, Person
from dateutil.parser import parse
import math
import yaml

#   Load the Configurations
def load_configuration(zotero_conf_filename):
    zotero_bibtex_config = dict()
    zotero_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
    configuration')

    yaml_path = os.path.join(zotero_conf_dir, zotero_conf_filename)

    with open(yaml_path, "r") as file:
        zotero_bibtex_config = yaml.load(file, Loader=yaml.FullLoader)

    return zotero_bibtex_config
def downlaod_zotero_datas(URL, API_KEY):
    zotero_result = list
    response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
    response = response.json()
    zotero_raw = json.dumps(response, ensure_ascii=False)  # json.loads(response)
    zotero_result = json.loads(zotero_raw)
    return zotero_result

#   Get the bibtex Datas from Zotero
def get_data(zotero_bibtex_config):
    result_limit = int(zotero_bibtex_config.get('result_limit'))
    access_type = zotero_bibtex_config.get('access_type')
    zotero_access_id = zotero_bibtex_config.get('zotero_access_id')
    collection_id = zotero_bibtex_config.get('collection_id')
```

```
    API_KEY = zotero_bibtex_config.get('api_key')
    zotero_data = list()
    URL = 'https://api.zotero.org/' + str(access_type) + '/' + str(
    zotero_access_id) + '/collections/' + str(
        collection_id) + '/items?limit=1?format=json?sort=dateAdded?direction=asc'

    response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})

    header_dict = response.headers
    total_elemets = int(header_dict.get('Total-Results'), 0)

    if total_elemets < result_limit:
        URL_ALL_ITEMS = 'https://api.zotero.org/' + str(access_type) + '/' + str(
            zotero_access_id) + '/collections/' + str(collection_id) + '/items?
    limit=' + str(
            result_limit) + '?format=json?sort=dateAdded?direction=asc'
        zotero_result = downlaod_zotero_datas(URL_ALL_ITEMS, API_KEY)

        zotero_data.extend(zotero_result)
    else:
        runs = int(math.ceil(total_elemets / result_limit))
        index = 0
        start_index = 0
        while index < runs:
            URL_Separated = 'https://api.zotero.org/' + str(access_type) + '/' +
    str(
                zotero_access_id) + '/collections/' + str(collection_id) + '/items
    ?limit=' + str(
                result_limit) + '?format=json?sort=dateAdded?direction=asc' + '&
    start=' + str(start_index)
            zotero_result = downlaod_zotero_datas(URL_Separated, API_KEY)

            zotero_data.extend(zotero_result)

            start_index += result_limit
            index += 1

    return zotero_data

#   Convert String to Datetime
def convert_to_datetime(input_str, parserinfo=None):
    return parse(input_str, parserinfo=parserinfo)

#   Get Dates from Datetime
def get_dates(date, bibtex_item_type, bibtex_month_attributes):
    dated_date = convert_to_datetime(date)
    return_value = dict()
    if bibtex_item_type in bibtex_month_attributes:
```

```
        year = dated_date.year
        month = dated_date.month
        return_value = {'year': year, 'month':month}
    else:
        year = dated_date.year
        return_value = {'year': year}

    return return_value

#   Split Creators into biblatex Creators
def split_creators(creators):
    if creators != []:

        creatorlist = ''
        for index, creator in enumerate(creators):
            type = creator.get('creatorType')
            firstname = creator.get('firstName')
            lastname = creator.get('lastName')
            name = creator.get('name')
            if type == 'author':

                if name and not (firstname or lastname):
                    creatorlist = creatorlist + name
                    if index != len(creators) - 1:
                        creatorlist = creatorlist + ' and '
                else:
                    creatorlist = creatorlist + lastname + ',' + firstname
                    if index != len(creators) - 1:
                        creatorlist = creatorlist + ' and '
    else:
        creatorlist = 'unknown author'

    bib_entry = 'author=' + '\"' + creatorlist + '\"'

    return bib_entry

#   Write the *.bib File
def write_bibliography(zotero_data, zotero_bibtex_config):
    keystore_file = zotero_bibtex_config.get('keystore_file')
    keystore_path = zotero_bibtex_config.get('keystore_filepath')
    tex_dir = os.path.join(os.path.dirname(os.getcwd()), keystore_path)

    yaml_path = os.path.join(tex_dir, keystore_file)

    with open(yaml_path, "r") as file:
        zotero_bibtex_keys = yaml.load(file, Loader=yaml.FullLoader)

    zotero_bibtex_keys_specials = {
```

```
        'thesis': {'phdthesis': ['dissertation', 'phd', 'doctorial', 'doctor', '
    doktor', 'doktorarbeit'],
                    'masterthesis': ['ma', 'master', 'masters']}
    }
    zotero_bibtex_attributes_special = {
        'date': 'get_dates',
        'creators': 'split_creators'
    }
    bibtex_month_attributes = ['booklet', 'mastersthesis', 'phdthesis', '
    techreport']
    # Bibliography
    # tex_dir = os.path.join(os.path.dirname(os.getcwd()), 'source')
    bibtex_path = zotero_bibtex_config.get('bibtex_filepath')
    tex_dir = os.path.join(os.path.dirname(os.getcwd()), bibtex_path)
    # tex_dir = os.path.join(os.getcwd(), 'src', 'content')
    # file_name = 'Datenbank_Projektauftrag_Michael_Graber.bib'
    file_name = zotero_bibtex_config.get('bibtex_filename')

    file_path = os.path.join(tex_dir, file_name)

    # bib_datas = BibliographyData()
    listKeys = list()
    bib_data = ''
    for zotero_items in zotero_data:
        biblio_item = zotero_items.get('data')
        itemkeys = biblio_item.keys()
        listKeys.extend(biblio_item.keys())
        zotero_item_key = biblio_item.get('key')
        zotero_item_title = biblio_item.get('title')
        zotero_item_nameofact = biblio_item.get('nameOfAct')
        zotero_item_nameofcase = biblio_item.get('caseName')
        zotero_item_subject = biblio_item.get('subject')
        zotero_item_type = biblio_item.get('itemType')

        # some item types have no titles
        # set the special names instead of the title
        if zotero_item_title:
            bibtex_item_titel = zotero_item_title
        else:
            if zotero_item_type == 'statute':
                biblio_item['title'] = zotero_item_nameofact
                bibtex_item_titel = zotero_item_nameofact
            elif zotero_item_type == 'case':
                biblio_item['title'] = zotero_item_nameofcase
                bibtex_item_titel = zotero_item_nameofcase
            elif zotero_item_type == 'email':
                biblio_item['title'] = zotero_item_subject
                bibtex_item_titel = zotero_item_subject
```

```
172
173         if zotero_item_type == 'thesis':
174             master_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('
    masterthesis')
175             phd_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('
    phdthesis')
176
177             # First Master thesis
178             if any(item in bibtex_item_titel for item in master_list):
179                 bibtex_item_key = 'masterthesis'
180             # Second PHD Thesis
181             elif any(item in bibtex_item_titel for item in phd_list):
182                 bibtex_item_key = 'phdthesis'
183             else:
184                 bibtex_item_key = 'masterthesis'
185         else:
186             if zotero_bibtex_keys.get(zotero_item_type).get('key'):
187                 bibtex_item_key = zotero_bibtex_keys.get(zotero_item_type).get('
    key')
188             else:
189                 bibtex_item_key = 'misc'
190
191         # get all Keys for the zotero item type
192         entryset = '\n'
193         entry = ''
194
195         zotero_item_attributes = zotero_bibtex_keys.get(zotero_item_type).get('
    attributes').keys()
196         item_attributes = sorted(zotero_item_attributes, reverse=True)
197
198         for index, item_attribute in enumerate(item_attributes):
199             bibtex_item_attribute = zotero_bibtex_keys.get(zotero_item_type).get('
    attributes').get(item_attribute)
200             zotero_item_value = biblio_item.get(item_attribute)
201             zotero_item_value_extra = ''
202             bibtex_item_attribute_extra = ''
203
204             # Special Cases
205             if bibtex_item_attribute == 'SPECIALCHECK' and zotero_item_value not
    in ['', None]:
206                 bibtex_special_attribute = zotero_bibtex_attributes_special.get(
    item_attribute)
207
208                 match bibtex_special_attribute:
209                     case 'get_dates':
210                         zotero_item_value = get_dates(zotero_item_value,
    bibtex_item_key, bibtex_month_attributes)
211                         if zotero_item_value.get('month'):
```

```
                            zotero_item_value_extra = zotero_item_value.get('month')
                            bibtex_item_attribute_extra = 'month'

                        zotero_item_value = zotero_item_value.get('year')
                        bibtex_item_attribute = 'year'
                    case 'split_creators':
                        authors = split_creators(zotero_item_value)
                        entryset = entryset + authors
            elif bibtex_item_attribute == 'howpublished':
                if zotero_item_value not in ['', None, []]:
                    zotero_item_value = '\\url{' + zotero_item_value + '}'

            if bibtex_item_attribute not in ['', 'None', 'author', 'SPECIALCHECK'] and zotero_item_value not in ['', None, []]:
                if zotero_item_value_extra:

                    if type(zotero_item_value_extra) == "string":
                        entryset = entryset + str(bibtex_item_attribute_extra) + '=\"' + str(zotero_item_value_extra) + '\"'
                    else:
                        entryset = entryset + str(bibtex_item_attribute_extra) + '=' + str(zotero_item_value_extra)

                    if index != len(item_attributes) - 1:
                        entryset = entryset + ',\n'
                    else:
                        entryset = entryset + '\n'

                if type(zotero_item_value) == str and not zotero_item_value.isnumeric():
                    entryset = entryset + str(bibtex_item_attribute) + '=\"' + str(zotero_item_value) + '\"'
                else:
                    entryset = entryset + str(bibtex_item_attribute) + '=' + str(zotero_item_value)

                if index != len(item_attributes) - 1:
                    entryset = entryset + ',\n'
                else:
                    entryset = entryset + '\n'

        # create the Entry
        entry = '@' + bibtex_item_key + '{' + zotero_item_key + ',\n'
        entry = entry + entryset + '}'
        bib_data = bib_data + '\n' + entry

    # parse String to pybtex.database Object
```

```
    # bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex", encoding='ISO-8859-1')
    bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex", encoding='Iutf-8')
    # Save pybtex.database to file
    # BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding='ISO-8859-1')
    BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding='utf-8')


zotero_bibtex_config = load_configuration('zotero_bibtex_configuration.yaml')
zotero_data = get_data(zotero_bibtex_config)
write_bibliography(zotero_data, zotero_bibtex_config)
```

Listing 88: Python LaTex - zotero.py - Zotero BibLaTex Importer

X.II zotero_bibtex_configuration.yaml

```
result_limit: 100
access_type: "groups"
zotero_access_id: "5222465"
collection_id: "PC3BW6EP"
api_key: "6Xgb3XhGjQXwA8NuZgu3bw3s"
keystore_file: "zotero_biblatex_keystore.yaml"
keystore_filepath: "source/configuration"
bibtex_filepath: "source"
bibtex_filename: "Diplomarbeit_Michael_Graber.bib"
```

Listing 89: Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer

X.III zotero_biblatex_keystore.yaml

```
---
artwork:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
audioRecording:
  key: misc
  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:
```

```
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    series: series
    proceedingsTitle: booktitle
    publisher: publisher
    pages: pages
    place: address
dictionaryEntry:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
document:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
email:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
encyclopediaArticle:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
film:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
```

```yaml
forumPost:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
hearing:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
instantMessage:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
interview:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
journalArticle:
  key: article
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    volume: volume
    pages: pages
    seriesNumber: number
    seriesTitle: journal
    url: url
letter:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
```

```
    creators: SPECIALCHECK
    url: howpublished
    extra: note
magazineArticle:
  key: article
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    volume: volume
    pages: pages
    seriesNumber: number
    seriesTitle: journal
    url: url
manuscript:
  key: unpublished
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
map:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
newspaperArticle:
  key: article
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    volume: volume
    pages: pages
    seriesNumber: number
    seriesTitle: journal
    url: url
patent:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
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:
```

```
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
tvBroadcast:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
videoRecording:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
webpage:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
attachment:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
note:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
standard:
```

```yaml
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
preprint:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
dataset:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
thesis:
  key: thesis
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    place: address
    university: school
```

Listing 90: Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer

X.IV riskmatrix.py

```python
import os
import matplotlib.pyplot as plt
import yaml

#   Load Configurations
def load_configuration(riskmatrix_conf_filename):
    riskmatrix_config = dict()

    riskmatrix_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
    configuration')
```

```
    yaml_path = os.path.join(riskmatrix_conf_dir, riskmatrix_conf_filename)

    with open(yaml_path, "r") as file:
        riskmatrix_config = yaml.load(file, Loader=yaml.FullLoader)

    return riskmatrix_config

#   Load x-y axis tuples
def load_xy_axis_tuples(riskmatrix_config):
    startpath = riskmatrix_config.get('riskmatrix').get('startpath')
    riskmatrix_xy_axis_tuples_dir = riskmatrix_config.get('riskmatrix').get('configfile_path')
    riskmatrix_xy_axis_tuples_config = riskmatrix_config.get('riskmatrix').get('configfile_name')

    if startpath == 'homedir':
        directory = os.path.join(os.getcwd(), riskmatrix_xy_axis_tuples_dir)
    else:  # parentdir
        directory = os.path.join(os.path.dirname(os.getcwd()), riskmatrix_xy_axis_tuples_dir)

    riskmatrix_xy_axis_tuples_path = os.path.join(directory, riskmatrix_xy_axis_tuples_config)
    riskmatrix_xy_axis_tuples = dict()
    riskmatrix_xy_axis_tuples_aux = dict()

    with open(riskmatrix_xy_axis_tuples_path, "r") as file:
        riskmatrix_xy_axis_tuples_aux = yaml.load(file, Loader=yaml.FullLoader)

    for string_key in riskmatrix_xy_axis_tuples_aux:
        value = riskmatrix_xy_axis_tuples_aux.get(string_key)
        int_key = eval(string_key)
        riskmatrix_xy_axis_tuples.update({int_key:value})
    return riskmatrix_xy_axis_tuples

#   Load Data from csv
def get_data(data_path):

    with open(data_path) as f:
        csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]

    (_, *header), *data = csv_list
    datas = {}
    for row in data:
        key, *values = row
        datas[key] = {key: value for key, value in zip(header, values)}

    return datas
```

```python

#   Generate Riskmatrix Image
#def riskmatrix(risk, conf, matrix):
def riskmatrix(conf, matrix):
    risks = conf.get('risk_inventory')
    for risk_conf in risks:
        # get the risk config datas
        startpath = conf.get('risks').get(risk_conf).get('startpath')
        destination = conf.get('risks').get(risk_conf).get('destination_path')
        imagename = conf.get('risks').get(risk_conf).get('imagename')
        datafilename = conf.get('risks').get(risk_conf).get('datafile')
        itemname = conf.get('risks').get(risk_conf).get('itemname')
        x_axis_title = conf.get('risks').get(risk_conf).get('x-axis-title')
        y_axis_title = conf.get('risks').get(risk_conf).get('y-axis-title')
        title = conf.get('risks').get(risk_conf).get('title')
        bubble_standard_size = conf.get('risks').get(risk_conf).get('bubble-standard-size')

        # Identify the index of the axes
        green = conf.get('risks').get(risk_conf).get('settings').get('green-boxes')
        yellow = conf.get('risks').get(risk_conf).get('settings').get('yellow-boxes')
        orange = conf.get('risks').get(risk_conf).get('settings').get('orange-boxes')
        red = conf.get('risks').get(risk_conf).get('settings').get('red-boxes')

        if startpath == 'homedir':
            directory = os.path.join(os.getcwd(), destination)
        else:  # parentdir
            directory = os.path.join(os.path.dirname(os.getcwd()), destination)

        data_path = os.path.join(directory, datafilename)
        image_path = os.path.join(directory, imagename)

        # get the Datas as dirct
        datas = get_data(data_path)

        fig = plt.figure()
        plt.subplots_adjust(wspace=0, hspace=0)
        plt.xticks([])
        plt.yticks([])
        plt.xlim(0, 5)
        plt.ylim(0, 5)
        plt.xlabel(x_axis_title)
        plt.ylabel(y_axis_title)
        plt.title(title)

```

```
98          # This example is for a 5 * 5 matrix
99          nrows = 5
100         ncols = 5
101         axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0,
    nrows) for c in range(0, ncols)]
102
103         # remove the x and y ticks
104         for ax in axes:
105             ax.set_xticks([])
106             ax.set_yticks([])
107             ax.set_xlim(0, 5)
108             ax.set_ylim(0, 5)
109
110         # Add background colors
111         # This has been done manually for more fine-grained control
112         # Run the loop below to identify the indice of the axes
113         for _ in green:
114             axes[_].set_facecolor('green')
115
116         for _ in yellow:
117             axes[_].set_facecolor('yellow')
118
119         for _ in orange:
120             axes[_].set_facecolor('orange')
121
122         for _ in red:
123             axes[_].set_facecolor('red')
124
125         # run throuh datas and generate axis datas
126         dict_bubble_axis = dict()
127         bubble_axis = list()
128         for datasets in datas:
129             # get the datas
130             riskid = datas.get(datasets).get('risk-id')
131             x_axis = int(datas.get(datasets).get('x-axis'))
132             y_axis = int(datas.get(datasets).get('y-axis'))
133             axis_point = matrix.get((x_axis, y_axis))
134             x_axis_text = float(datas.get(datasets).get('x-axis-text'))
135             y_axis_text = float(datas.get(datasets).get('y-axis-text'))
136             x_axis_bubble = float(datas.get(datasets).get('x-axis-bubble'))
137             y_axis_bubble = float(datas.get(datasets).get('y-axis-bubble'))
138             bubble_axis.append(axis_point)
139
140             # merge riks if two or more risks share the same axispoint
141             if dict_bubble_axis.get(axis_point):
142                 risktag = dict_bubble_axis.get(axis_point).get('risk')
143                 risktag = risktag + '/' + riskid
144                 x_axis_text = x_axis_text + 0.25
```

```
                y_axis_text = y_axis_text - 0.5
                bubble_size = bubble_standard_size * 2
            else:
                risktag = itemname + riskid
                bubble_size = bubble_standard_size
            dict_axis_value = dict()

            dict_axis_value['risk'] = risktag
            dict_axis_value['x-axis-text'] = x_axis_text
            dict_axis_value['y-axis-text'] = y_axis_text
            dict_axis_value['x-axis-bubble'] = x_axis_bubble
            dict_axis_value['y-axis-bubble'] = y_axis_bubble
            dict_axis_value['size'] = bubble_size
            dict_bubble_axis[axis_point] = dict_axis_value

        # cleanup the list, remove duplicated entries
        bubble_axis = set(bubble_axis)

        # plot the bubbles and texts in the bubbles
        for axispoint in bubble_axis:
            axes[axispoint].scatter(dict_bubble_axis[axispoint]['x-axis-bubble'],
                                    dict_bubble_axis[axispoint]['y-axis-bubble'],
                                    dict_bubble_axis[axispoint]['size'], alpha=1)
            axes[axispoint].text(dict_bubble_axis[axispoint]['x-axis-text'],
                                 dict_bubble_axis[axispoint]['y-axis-text'], s=
    dict_bubble_axis[axispoint]['risk'],
                                 va='bottom', ha='center')

        # save the plot as image
        plt.savefig(image_path)

riskmatrix_config = load_configuration('riskmatrix_plotter_conf.yaml')
riskmatrix_xy_axis_tuples = load_xy_axis_tuples(riskmatrix_config)
riskmatrix(riskmatrix_config, riskmatrix_xy_axis_tuples)
```

Listing 91: Python LaTex - riskmatrix.py - Risxikomatrizen

X.V riskmatrix_plotter_conf.yaml

```yaml
risk_inventory:
  - "postgresql"
  - "project"
  - "Postgresql-massnahme"
  - "Project-massnahme"
riskmatrix:
  startpath: "parentdir"
  configfile_path: "source/configuration"
```

```yaml
  configfile_name: "riskmatrix_xy_axis_tuple_matrix.yaml"
risks:
  postgresql:
    riskid: "postgresql"
    startpath: "parentdir"
    destination_path: "source/riskmatrix"
    imagename: "riskmatrixproblem.png"
    datafile_path: "source/tables"
    datafile: "riskmatrixproblem.csv"
    itemname: "R"
    x-axis-title: "Schadensausmass (SM)"
    y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
    title: "Risiko Cockpit PostgreSQL Datenbanken KSGR"
    bubble-standard-size: 1000
    settings:
      green-boxes:
        - 10
        - 15
        - 16
        - 20
        - 21
      yellow-boxes:
        - 0
        - 5
        - 6
        - 11
        - 17
        - 22
        - 23
      orange-boxes:
        - 1
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
  project:
    riskid: "project"
    startpath: "parentdir"
    destination_path: "source/riskmatrix"
```

```yaml
    imagename: "riskmatrix-project.png"
    datafile_path: "source/tables"
    datafile: "riskmatrix-project.csv"
    itemname: "R"
    x-axis-title: "Schadensausmass (SM)"
    y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
    title: "Risiko Cockpit Projekt"
    bubble-standard-size: 1000
    settings:
      green-boxes:
        - 10
        - 15
        - 16
        - 20
        - 21
      yellow-boxes:
        - 0
        - 5
        - 6
        - 11
        - 17
        - 22
        - 23
      orange-boxes:
        - 1
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
  Postgresql-massnahme:
    riskid: "Postgresql-massnahme"
    startpath: "parentdir"
    destination_path: "source/riskmatrix"
    imagename: "Riskmatrixproblem-massnahmen.png"
    datafile_path: "source/tables"
    datafile: "riskmatrixproblem-massnahmen.csv"
    itemname: "R"
    x-axis-title: "Schadensausmass (SM)"
    y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
```

```yaml
    title: "Risiko Cockpit PostgreSQL Datenbanken KSGR - Massnahme"
    bubble-standard-size: 1000
    settings:
      green-boxes:
        - 10
        - 15
        - 16
        - 20
        - 21
      yellow-boxes:
        - 0
        - 5
        - 6
        - 11
        - 17
        - 22
        - 23
      orange-boxes:
        - 1
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
  Project-massnahme:
    riskid: "Project-massnahme"
    startpath: "parentdir"
    destination_path: "source/riskmatrix"
    imagename: "Riskmatrix-project-massnahmen.png"
    datafile_path: "source/tables"
    datafile: "riskmatrix-project-massnahmen.csv"
    itemname: "R"
    x-axis-title: "Schadensausmass (SM)"
    y-axis-title: "Eintrittswahrscheinlichkeit (WS)"
    title: "Risiko Cockpit Projekt - Massnahme"
    bubble-standard-size: 1000
    settings:
      green-boxes:
        - 10
        - 15
```

```yaml
        - 16
        - 20
        - 21
      yellow-boxes:
        - 0
        - 5
        - 6
        - 11
        - 17
        - 22
        - 23
      orange-boxes:
        - 1
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
```

Listing 92: Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen

X.VI riskmatrix_xy_axis_tuple_matrix.yaml

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

```
(3, 1): 22
(3, 2): 17
(3, 3): 12
(3, 4): 7
(3, 5): 2
(4, 1): 23
(4, 2): 18
(4, 3): 13
(4, 4): 8
(4, 5): 3
(5, 1): 24
(5, 2): 19
(5, 3): 14
(5, 4): 9
(5, 5): 4
```

Listing 93: Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples

X.VII cost_benefit_diagram.py

```python
import os
import matplotlib.pyplot as plt
import yaml

# Get the Configuration
def load_configuration():
    cost_benefit_config = dict()
    cbd_conf_filename = 'scatter_plotter_conf.yaml'
    cbd_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
    configuration')
    yaml_path = os.path.join(cbd_conf_dir, cbd_conf_filename)

    with open(yaml_path, "r") as file:
        cost_benefit_config = yaml.load(file, Loader=yaml.FullLoader)

    return cost_benefit_config
# Get the Datas
def get_data(cost_benefit_config):
    # Config Variables
    startpath = cost_benefit_config.get('startpath')
    destination = cost_benefit_config.get('desitination_path')
    datafilename = cost_benefit_config.get('datafile')

    if startpath == 'homedir':
        directory = os.path.join(os.getcwd(), destination)
    else:   # parentdir
```

```
        directory = os.path.join(os.path.dirname(os.getcwd()), destination)

    # get the Datas as dirct
    data_path = os.path.join(directory, datafilename)

    # load datas from csv into dict
    with open(data_path) as f:
        csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]

    (_, *header), *data = csv_list
    datas = {}
    for row in data:
        key, *values = row
        datas[key] = {key: value for key, value in zip(header, values)}

    cost_benefit_data = {}
    for key, value in datas.items():
        variant_name = value['variant_name']
        x_axis = int(value['x-axis'])
        y_axis = int(value['y-axis'])
        cost_benefit_data[variant_name] = (x_axis, y_axis)

    return cost_benefit_data

# Plot the Datas
def cost_benefit_diagram (cost_benefit_config, cost_benefit_data):
    # Config Variables
    startpath = cost_benefit_config.get('startpath')
    destination = cost_benefit_config.get('desitination_path')
    imagename = cost_benefit_config.get('imagename')

    if startpath == 'homedir':
        directory = os.path.join(os.getcwd(), destination)
    else:   # parentdir
        directory = os.path.join(os.path.dirname(os.getcwd()), destination)

    # get the Datas as dirct
    data_path = os.path.join(directory, imagename)

    # Extract the Datas
    labels, values = zip(*cost_benefit_data.items())
    x, y = zip(*values)

    # Create Scatter-Diagram
    plt.scatter(x, y, color=cost_benefit_config.get('scatter-point-color'))

    # X-Lines
    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'))

    # Y-Lines
    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'))

    # Add Labels
    plt.xlabel(cost_benefit_config.get('x-axis-title'))
    plt.ylabel(cost_benefit_config.get('y-axis-title'))
    plt.title(cost_benefit_config.get('title'))

    # Labling Data Points
    for label, x_point, y_point in zip(labels, x, y):
        plt.text(x_point, y_point, label)

    # Show Legends
    plt.legend()

    # Show Grid
    plt.grid(True)

    # Save Diagram as PNG
    plt.savefig(data_path)

cost_benefit_config = load_configuration()
cost_benefit_data = get_data(cost_benefit_config)
cost_benefit_diagram(cost_benefit_config, cost_benefit_data)
```

Listing 94: Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm

X.VIII cost_benefit_diagram_plotter_conf.yaml

```yaml
startpath: "parentdir"
desitination_path: "source/cost_benefit_diagram"
datafile: "cost_benefit_diagram.csv"
imagename: "cost_benefit_diagram.png"
scatter-point-color: "blue"
x-axis-title: "Punkte"
x-axis-line-pos: 80
x-axis-line-label: "Kosten-Minimum"
x-axis-line-type: "--"
x-axis-line-color: "red"
y-axis-title: "Kosten"
y-axis-line-pos: 80
y-axis-line-label: "Punkte-Minimum"
```

```
y-axis-line-type: "--"
y-axis-line-color: "green"
title: "Kosten-Nutzen-Diagramm Beispiel"
```

Listing 95: Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm

X.IX pandas_dataframe_to_latex_table.py

```
import os
import pandas as pd
import yaml
from pathlib import Path
import chardet

import csv

# Get the Configuration
def load_configuration(plt_conf_filename):
    panda_latex_tables_config = dict()
    plt_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
    configuration')
    yaml_path = os.path.join(plt_conf_dir, plt_conf_filename)

    with open(yaml_path, "r") as file:
        panda_latex_tables_config = yaml.load(file, Loader=yaml.FullLoader)

    return panda_latex_tables_config


def get_data(startpath, destination, tablefilename, datafile_path, datafile,
    alternative_cvs_load, separator, decimal):
    # Config Variables
    if startpath == 'homedir':
        directory = os.path.join(os.getcwd(), datafile_path)
    else:  # parentdir
        directory = os.path.join(os.path.dirname(os.getcwd()), datafile_path)

    # get the Datas as dirct
    data_path = os.path.join(directory, datafile)

    # load datas from csv into dict
    detected = chardet.detect(Path(data_path).read_bytes())
    encoding = detected.get("encoding")

    # if alternative_cvs_load:
    #     with open(data_path, 'r', encoding=encoding) as file:
```

```
37     #         reader = csv.reader(file)
38     #         data = list(reader)
39     #
40     #     # panda_table_data = pd.DataFrame(data, columns=data[0])
41     #     # panda_table_data = pd.read_csv(data_path, sep=separator, decimal=
    decimal, encoding=encoding, lineterminator='\n', engine='python')
42     #     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal
    , encoding=encoding, lineterminator='\n')
43     #     df_dtype = {
44     #         "Nr.": int,
45     #         "Anforderung": str,
46     #         "Beschreibung": str,
47     #         "System": str,
48     #         "Muss / Kann": str
49     #     }
50     #     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".",
    encoding=encoding, lineterminator='\n', dtype=df_dtype)
51     #     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".",
    encoding=encoding)
52     # else:
53     #     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal
    , encoding=encoding)
54     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=
    encoding, low_memory=False, engine='python')
55     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=
    encoding, engine='python', dtype='unicode')
56     # readed = open(data_path, 'r', encoding=encoding)
57     # panda_table_data = pd.read_csv(open(data_path, 'r', encoding=encoding), sep
    =",", decimal=".", encoding=encoding)
58     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding = "
    ISO-8859-1")
59     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=
    encoding, chunksize=10)
60
61     # for chunk in pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding,
     chunksize=5):
62     #     print(chunk)
63     # panda_table_data = pd.DataFrame()
64     # temp = pd.read_csv(data_path, iterator=True, sep=",", decimal=".", encoding=
    encoding, chunksize=1000)
65     # panda_table_data = pd.concat(temp, ignore_index=True)
66
67     df_dtype = {
68         "Nr.": int,
69         "Anforderung": str,
70         "Beschreibung": str,
71         "System": str,
72         "Muss / Kann": str
```

```
    }
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, engine='python', dtype=df_dtype)
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, dtype=df_dtype)

    # import dask.dataframe as dd
    # df = dd.read_csv(data_path, sep=",", decimal=".", encoding=encoding)
    # panda_table_data = df
    print(encoding)
    panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
    # return data
    return panda_table_data


def create_latex_tables(panda_latex_tables_config):
    plt_tables = panda_latex_tables_config.get('tables_inventory')
    for table_item in plt_tables:
        # id and filesystem informations
        table_id = panda_latex_tables_config.get('tables').get(table_item).get('id')
        isbigfile = panda_latex_tables_config.get('tables').get(table_item).get('isbigfile')
        has_longtexts = panda_latex_tables_config.get('tables').get(table_item).get('has_longtexts')
        if isbigfile or has_longtexts:
            alternative_cvs_load = True
        else:
            alternative_cvs_load = False
        startpath = panda_latex_tables_config.get('tables').get(table_item).get('startpath')
        destination = panda_latex_tables_config.get('tables').get(table_item).get('destination_path')
        tablefilename = panda_latex_tables_config.get('tables').get(table_item).get('tablefilename')
        datafile_path = panda_latex_tables_config.get('tables').get(table_item).get('datafile_path')
        datafile = panda_latex_tables_config.get('tables').get(table_item).get('datafile')
        if startpath == 'homedir':
            directory = os.path.join(os.getcwd(), destination)
        else:  # parentdir
            directory = os.path.join(os.path.dirname(os.getcwd()), destination)
        tablefile = os.path.join(directory, tablefilename)
        separator = panda_latex_tables_config.get('tables').get(table_item).get('separator')
        decimal = panda_latex_tables_config.get('tables').get(table_item).get('
```

```
decimal')

    # column operations
    column_operations = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('datas')

    # group by / aggregation
    groupby_values = panda_latex_tables_config.get('tables').get(table_item).get('group_by')
    group_by_function = panda_latex_tables_config.get('tables').get(table_item).get('group_by_function')
    # selected_rows = panda_latex_tables_config.get('tables').get(table_item).get('selected_rows')
    agg_funtion = panda_latex_tables_config.get('tables').get(table_item).get('agg_funtion')
    agg_colums = panda_latex_tables_config.get('tables').get(table_item).get('agg_colums')
    # dropping and renaming columns
    drop_columns = panda_latex_tables_config.get('tables').get(table_item).get('drop_columns')
    rename_columns = panda_latex_tables_config.get('tables').get(table_item).get('rename_columns')

    # table filtering and sorting
    where_clausel = panda_latex_tables_config.get('tables').get(table_item).get('where_clausel')
    order_by = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('order_by')
    sort_acending = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('sort_acending')
    sort_inplace = panda_latex_tables_config.get('tables').get(table_item).get('sorting').get('sort_inplace')

    # pivot settings
    pivot = panda_latex_tables_config.get('tables').get(table_item).get('pivot')
    pivot_column = panda_latex_tables_config.get('tables').get(table_item).get('pivot_columns')
    pivot_value = panda_latex_tables_config.get('tables').get(table_item).get('pivot_values')

    # pivot_table settings
    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).get('pivot_table').get(
        'pivot_columns')
    pivot_table_value = panda_latex_tables_config.get('tables').get(table_item
```

```
).get('pivot_table').get(
            'pivot_values')
        pivot_table_agg_function = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
            'pivot_agg_func')
        pivot_table_indizes = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_indizes')
        pivot_table_indizes_visible = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_indizes_visible')
        pivot_table_rename_indizes = panda_latex_tables_config.get('tables').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_rename_indizes')

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

        # table settings
        table_caption = panda_latex_tables_config.get('tables').get(table_item).get('caption')
        table_label = panda_latex_tables_config.get('tables').get(table_item).get('label')
        table_style = panda_latex_tables_config.get('tables').get(table_item).get('table_styles')
        sparse_columns = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get(
            'sparse_columns')
        table_caption_position = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get(
            'props').get('caption-side')
        table_position = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
            'position')
        longtable = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
            'longtable')
        linebreak_columns = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
            'linebreak_columns')
        resize_textwidth = panda_latex_tables_config.get('tables').get(table_item).get('table_styles').get('props').get(
            'resize_textwidth')

        # get the pandas (panda data)
```

```
171         panda_table_data = get_data(startpath, destination, tablefilename,
    datafile_path, datafile, alternative_cvs_load, separator, decimal)
172
173         # filter by where clausel
174         if where_clausel:
175             panda_table_data = panda_table_data.query(where_clausel)
176
177         # Drop unused columns
178         if drop_columns:
179             panda_table_data = panda_table_data.drop(columns=drop_columns)
180
181         # set aggregation functions
182         # if groupby_values and not agg_funtion and not pivot_column and not
    pivot_table_column:
183         if groupby_values and not (pivot_column or (pivot_table_column or
    pivot_table_value or pivot_table_indizes)):
184             match group_by_function:
185                 case 'max':
186                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).max()
187                 case 'min':
188                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).min()
189                 case 'head':
190                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).head()
191                 case 'sum':
192                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).sum()
193                 case 'mean':
194                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).mean()
195         else:
196             panda_table_data = panda_table_data
197
198         # pivot if pivot is selected
199         if pivot_table_column or pivot_table_value or pivot_table_indizes:
200             if type(pivot_table_agg_function) is list:
201                 agg_tuple = tuple(pivot_table_agg_function)
202                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
203                                                   columns=pivot_table_column,
    values=pivot_table_value,
204                                                   aggfunc=agg_tuple, margins=
    margin, margins_name=margin_name)
205             elif type(pivot_table_agg_function) is dict:
206                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
```

```
                                                    columns=pivot_table_column,
                                                    values=pivot_table_value, aggfunc=pivot_table_agg_function,
                                                    margins=margin, margins_name=margin_name)
        else:
            panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
                                                    columns=pivot_table_column, values=pivot_table_value,
                                                    aggfunc=pivot_table_agg_function, margins=margin,
                                                    margins_name=margin_name)

    # set column operations
    if column_operations:
        for column_ops in column_operations:
            operation_function = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('operation_function')
            operation_columns = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('columns')
            operation_axis = panda_latex_tables_config.get('tables').get(table_item).get('column_operations').get('operations').get(column_ops).get('axis_number')
            match operation_function:
                case 'max':
                    panda_table_data[column_ops] = panda_table_data[operation_columns].max()
                case 'min':
                    panda_table_data[column_ops] = panda_table_data[operation_columns].min()
                case 'head':
                    panda_table_data[column_ops] = panda_table_data[operation_columns].head()
                case 'sum':
                    panda_table_data[column_ops] = panda_table_data[operation_columns].sum(axis=operation_axis)
                case 'mean':
                    panda_table_data[column_ops] = panda_table_data[operation_columns].mean()
                case 'diff':
                    panda_table_data[column_ops] = panda_table_data[operation_columns[1]] - panda_table_data[operation_columns[0]]


    # order by
```

```
        if order_by:
            panda_table_data.sort_values(by=order_by, inplace=sort_inplace, ascending=sort_acending)

        # rename columns
        if rename_columns:
            panda_table_data = panda_table_data.rename(columns=rename_columns)

        # rename indices
        if pivot_table_rename_indizes:
            panda_table_data = panda_table_data.rename_axis(index=pivot_table_rename_indizes)

        # frame carriage return columns in subtable
        if linebreak_columns:
            for lbr_column in linebreak_columns:
                panda_table_data[lbr_column] = "\\begin{tabular}[c]{@{}l@{}}" + panda_table_data[lbr_column].astype(str) + "\\end{tabular}"
        # convert python panda to latex table
        latex_table = panda_table_data.to_latex(header=True, bold_rows=False, longtable=longtable,
                                                sparsify=sparse_columns, label=table_label, caption=table_caption,
                                                position=table_position, na_rep='', index=pivot_table_indizes_visible)

        # textwidth resize
        if resize_textwidth:
            with open(tablefile, 'w') as wrlt:
                wrlt.write(latex_table)

            with open(tablefile) as file:
                lines = file.readlines()

            # replace table with resize
            resize_line_nr = 0
            resize_line = ""
            if longtable:
                table_type = '\\begin{longtable}'
            else:
                table_type = '\\begin{table}'

            for number, line in enumerate(lines, 1):
            # for number, line in latex_table.splitlines():
            # for number, line in latex_table.readlines():
            # for number, line in latex_table.splitlines('\n'):
            # for number, line in lines.split('\n'):

```

```
                # Condition true if the key exists in the line
                # If true then display the line number
                if table_type in line:
                    # print(f'{key} is at line {number}')
                    resize_line_nr = number
                    resize_line = line

            line_table_resize = resize_line + "\n" + "\\resizebox{\\columnwidth}{!}{%"
            latex_table = latex_table.replace(resize_line, line_table_resize)

            # replace table end with bracket
            resize_line_nr = 0
            resize_line = ""
            if longtable:
                table_type = '\\end{longtable}'
            else:
                table_type = '\\end{table}'

            for number, line in enumerate(lines, 1):

                # Condition true if the key exists in the line
                # If true then display the line number
                if table_type in line:
                    # print(f'{key} is at line {number}')
                    resize_line_nr = number
                    resize_line = line

            line_table_resize =  "}" + "\n" + resize_line
            latex_table = latex_table.replace(resize_line, line_table_resize)

        # caption below is not supported yet (pandas 2.2)
        # replace caption and replace table end with the caption line and table end
        if table_caption_position == 'below':
            caption_label = "\\caption{" + table_caption + "} \\label{" + table_label + "} \\\\"
            caption_label_nbr = "\\caption{" + table_caption + "} \\label{" + table_label + "}"
            caption_only = "\\caption{" + table_caption + "} \\\\"
            caption_only_nbr = "\\caption{" + table_caption + "}"
            label_only = "\\label{" + table_label + "} \\\\"
            label_only_nbr = "\\label{" + table_label + "}"
            latex_table = latex_table.replace(caption_label, '')
            latex_table = latex_table.replace(caption_only, '')
            latex_table = latex_table.replace(label_only, '')
            latex_table = latex_table.replace(caption_label_nbr, '')
            latex_table = latex_table.replace(caption_only_nbr, '')
```

```
            latex_table = latex_table.replace(label_only_nbr, '')

            if longtable:
                table_string = '\\end{longtable}'
                new_caption = caption_label_nbr + "\n" + table_string
                latex_table = latex_table.replace(table_string, new_caption)
            else:
                table_string = '\\end{table}'
                new_caption = caption_label_nbr + "\n" + table_string
                latex_table = latex_table.replace(table_string, new_caption)


        # write latex table to filesystem
        with open(tablefile, 'w') as wrlt:
            wrlt.write(latex_table)


# run the methods / functions
panda_latex_tables_config = load_configuration('csv_to_latex_diplomarbeit.yaml')
create_latex_tables(panda_latex_tables_config)
```

Listing 96: Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle

X.X csv_to_latex_diplomarbeit.yaml

```
tables_inventory:
 - "db_inventory"
 - "db_inventory_per_rdbms"
 - "db_inventory_per_os"
 - "anforderungskatalog"
 - "arbeitsrapport"
 - "projektcontrolling"
 - "evaluation_inventory"
 - "dependencis"
 - "predecision_out"
 - "predecision_in"
 - "project_comments"
 - "evaluation_distributed_sql"
 - "expert_discussions_overview"
 - "expert_discussions_full_list"
 - "stakeholder"
tables:
 db_inventory:
  id: "db_inventory"
  isbigfile:
  has_longtexts: False
  separator: ","
```

```yaml
  decimal: "."
  caption: "Datenbankinventar - Roh"
  label: "db_inventory"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "inventory.csv"
  tablefilename: "db_inventory.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
   - "comment"
   - "eol"
   - "eol_since"
   - "releasedate"
  column_operations:
   datas:
   operations:
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
    pivot_indizes_visible:
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
   server: "Server - Hostname"
   os: "OS"
   rdbms: "RDBMS"
   instance: "Instanz"
   databases: "Datenbanken"
   appliance: "Appliance"
   comment: "Kommentar"
   version: "Version"
   releasedate: "Version - Releasedatum"
   eol: "EoL"
   age: "Version - Alter"
   eol_since: "EoL seit"
```

```
  where_clausel:
  sorting:
   order_by:
    - "server"
    - "rdbms"
   sort_acending: True
   sort_inplace: True
  margins:
   margin: False
   margin_name:
  table_styles:
   selector: "caption"
   props:
    caption-side: "below"
    position: "H"
    sparse_columns: True
    longtable: True
    resize_textwidth: False
    linebreak_columns:
    table_header: True
db_inventory_per_rdbms:
  id: "db_inventory_per_rdbms"
  isbigfile:
  has_longtexts: False
  separator: ","
  decimal: "."
  caption: "Datenbankinventar"
  label: "db_inventory_per_rdbms"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "inventory.csv"
  tablefilename: "db_inventory_per_rdbms.tex"
  decimal_format:
  group_by:
   - "rdbms"
  group_by_function: "sum"
  agg_funtion:
  agg_colums:
   - "rdbms"
  drop_columns:
   - "server"
   - "os"
   - "version"
   - "releasedate"
   - "eol"
   - "age"
   - "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_indizes_visible:
133     pivot_rename_indizes:
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_clausel:
143   sorting:
144    order_by:
145     - "rdbms"
146    sort_acending: True
147    sort_inplace: 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 üaufgeschlsselt"
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_clausel:
224   sorting:
225    order_by:
226    sort_acending: 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_indizes_visible: False
273     pivot_rename_indizes:
274    pivot_columns:
275    pivot_values:
276    pivot_agg_func:
277   rename_columns:
278   where_clausel:
279   sorting:
280    order_by:
281     - "Nr."
282    sort_acending: True
283    sort_inplace: True
284   margins:
285    margin: False
286    margin_name:
287   table_styles:
288    selector: "caption"
289    props:
290     caption-side: "below"
291     position: "H"
292     sparse_columns: False
293     longtable: False
294     resize_textwidth: True
295     linebreak_columns:
296      - "Beschreibung"
297     table_header: True
298 arbeitsrapport:
299  id: "arbeitsrapport"
300  isbigfile:
301  has_longtexts: False
302  separator: ";"
303  decimal: "."
304  caption: "Arbeitsrapport"
305  label: "arbeitsrapport"
306  startpath: "parentdir"
307  destination_path: "content/latex_tables"
308  datafile_path: "source/tables"
309  datafile: "arbeitsrapport.CSV"
310  tablefilename: "arbeitsrapport.tex"
```

```
311   decimal_format: "{:0.1f}"
312   group_by:
313   group_by_function:
314   agg_funtion:
315   agg_colums:
316   drop_columns:
317    - "Hide"
318    - "Geplante Dauer [h]"
319    - "dauer_summe"
320   column_operations:
321    datas:
322    operations:
323     dauer_summe:
324      operation_function:
325      axis_number:
326      columns:
327   pivot:
328    pivot_columns:
329    pivot_values:
330   pivot_table:
331    pivot_index:
332     pivot_indizes_visible: False
333     pivot_rename_indizes:
334    pivot_columns:
335    pivot_values:
336    pivot_agg_func:
337   rename_columns:
338   where_clausel: "Hide == 0"
339   sorting:
340    order_by:
341     - "Datum"
342     - "Von"
343    sort_acending: 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:
```

```
  pivot_table:
   pivot_index:
    pivot_indizes_visible:
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
   dauer_summe: "Verbleibende Zeit [h]"
  where_clausel:
  sorting:
   order_by:
    - "Phase"
    - "Subphase"
   sort_acending: True
   sort_inplace: True
  margins:
   margin: True
   margin_name: "Total"
  table_styles:
   selector: "caption"
   props:
    caption-side: "below"
    position: "H"
    sparse_columns: True
    longtable: False
    resize_textwidth: True
    linebreak_columns:
    table_header: True
 evaluation_inventory:
  id: "evaluation_inventory"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "Evaluationssyssteme"
  label: "evaluation_inventory"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "evaluation_platform_serverlist.csv"
  tablefilename: "evaluation_inventory.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  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_acending: 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     - "äAbhngigkeit"
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_clausel: "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         - "üBegrndung"
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:
```

```
    pivot_values:
    pivot_agg_func:
  rename_columns:
  where_clausel: "hide_state == 2"
  sorting:
    order_by:
      - "Nr."
    sort_acending: True
    sort_inplace: True
  margins:
    margin: False
    margin_name:
  table_styles:
    selector: "caption"
    props:
      caption-side: "below"
      position: "H"
      sparse_columns: True
      longtable: False
      resize_textwidth: True
      linebreak_columns:
        - "üBegrndung"
      table_header: True
 project_comments:
  id: "project_comments"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "Kommentare - Anmerkung"
  label: "project_comments"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "pre-fazit.csv"
  tablefilename: "pre-fazit.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
  column_operations:
    datas:
    operations:
      dauer_summe:
        operation_function:
        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_clausel:
708   sorting:
709    order_by:
710     - "Woche"
711    sort_acending: 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: "Evaluationssysstem - 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_funtion:
744   agg_colums:
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_indizes_visible: False
759     pivot_rename_indizes:
760    pivot_columns:
761    pivot_values:
762    pivot_agg_func:
763   rename_columns:
764   where_clausel:
765   sorting:
766    order_by:
767    sort_acending: 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: "äFachgesprche"
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_funtion:
799   agg_colums:
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_indizes_visible: False
817     pivot_rename_indizes:
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_indizes_visible: False
875     pivot_rename_indizes:
876    pivot_columns:
877    pivot_values:
878    pivot_agg_func:
879   rename_columns:
880   where_clausel:
881   sorting:
882    order_by:
883     - "äFachgesprch"
884    sort_acending: 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_funtion:
921   agg_colums:
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:
```

```
      pivot_indizes_visible: False
      pivot_rename_indizes:
     pivot_columns:
     pivot_values:
     pivot_agg_func:
    rename_columns:
    where_clausel:
    sorting:
     order_by:
     sort_acending: True
     sort_inplace: True
    margins:
     margin: False
     margin_name:
    table_styles:
     selector: "caption"
     props:
      caption-side: "below"
      position: "H"
      sparse_columns: True
      longtable: False
      resize_textwidth: True
      linebreak_columns:
      table_header: True
```

Listing 97: Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle

X.XI pandas_data_chart_plotter.py

```
import os
from pathlib import Path
import chardet
import pandas as pd
import yaml


def load_configuration(panda_diagram_plotter_conf_filename):
    panda_diagram_plotter_config = dict()

    riskmatrix_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
    configuration')
    yaml_path = os.path.join(riskmatrix_conf_dir,
    panda_diagram_plotter_conf_filename)

    with open(yaml_path, "r") as file:
        panda_diagram_plotter_config = yaml.load(file, Loader=yaml.FullLoader)
```

```
16
17      return panda_diagram_plotter_config
18
19  def get_data(startpath, destination, tablefilename, datafile_path, datafile,
    separator, decimal):
20      # Config Variables
21      if startpath == 'homedir':
22          directory = os.path.join(os.getcwd(), datafile_path)
23      else:  # parentdir
24          directory = os.path.join(os.path.dirname(os.getcwd()), datafile_path)
25
26      # get the Datas as dirct
27      data_path = os.path.join(directory, datafile)
28
29      # load datas from csv into dict
30      detected = chardet.detect(Path(data_path).read_bytes())
31      encoding = detected.get("encoding")
32
33      print(datafile, ':', encoding)
34      panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal,
    encoding=encoding)
35      # return data
36      return panda_table_data
37  def create_panda_diagram_plotter(panda_diagram_plotter_config):
38      pdp_tables = panda_diagram_plotter_config.get('diagram_inventory')
39      for table_item in pdp_tables:
40          print(table_item)
41          startpath = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('startpath')
42          destination = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('destination_path')
43          imagename = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('imagename')
44          datafile_path = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('datafile_path')
45          datafile = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('datafile')
46          if startpath == 'homedir':
47              directory = os.path.join(os.getcwd(), destination)
48          else:  # parentdir
49              directory = os.path.join(os.path.dirname(os.getcwd()), destination)
50          image_path = os.path.join(directory, imagename)
51          separator = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('separator')
52          decimal = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('decimal')
53
54          # column operations
```

```
55        column_operations = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('datas')
56
57        # group by / aggregation
58        groupby_values = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('group_by')
59        group_by_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('group_by_function')
60
61        agg_funtion = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('agg_funtion')
62        agg_colums = panda_diagram_plotter_config.get('panda_diagram_plotter').get
    (table_item).get('agg_colums')
63        # dropping and renaming columns
64        drop_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('drop_columns')
65        rename_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('rename_columns')
66
67        # table filtering and sorting
68        where_clausel = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('where_clausel')
69        order_by = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('sorting').get('order_by')
70        sort_acending = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('sorting').get('sort_acending')
71        sort_inplace = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('sorting').get('sort_inplace')
72
73        # pivot settings
74        pivot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('pivot')
75        pivot_column = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_columns')
76        pivot_value = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_values')
77
78        # pivot_table settings
79        pivot_table = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_table')
80        pivot_table_column = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
81            'pivot_columns')
82        pivot_table_value = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
83            'pivot_values')
84        pivot_table_agg_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
```

```
85              'pivot_agg_func')
86          pivot_table_indizes = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
87              'pivot_index').get('pivot_indizes')
88          pivot_table_indizes_visible = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
89              'pivot_index').get('pivot_indizes_visible')
90          pivot_table_rename_indizes = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
91              'pivot_index').get('pivot_rename_indizes')
92
93          # margins (subtotals)
94          margin = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('margins').get('margin')
95          margin_name = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('margins').get('margin_name')
96
97          # chart settings
98          chart = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-kind')
99          title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('title')
100         x_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('x-axis-title')
101         y_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('y-axis-title')
102         x_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('x-axis-columns')
103         y_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('y-axis-columns')
104         index = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-index')
105
106         # chart styles
107         grid = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('grid')
108         legend = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('legend')
109         rot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('rot')
110         fontsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('fontsize')
111         figsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('figsize')
112         stacked = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('stacked')
113         secondary_y = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('chart-designs').get('secondary_y')
```

```
114         stylelist = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('stylelist')
115         subplots = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('subplots')
116         autopct = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('autopct')
117         loc = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('loc')
118         bbox_to_anchor = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('chart-designs').get('bbox_to_anchor')
119 
120         # get the pandas (panda data)
121         panda_table_data = get_data(startpath, destination, imagename,
    datafile_path, datafile, separator, decimal)
122 
123         # filter by where clausel
124         if where_clausel:
125             panda_table_data = panda_table_data.query(where_clausel)
126 
127         # Drop unused columns
128         if drop_columns:
129             panda_table_data = panda_table_data.drop(columns=drop_columns)
130 
131         # set aggregation functions
132         # if groupby_values and not agg_funtion and not pivot_column and not
    pivot_table_column:
133         if groupby_values and not (pivot_column or (pivot_table_column or
    pivot_table_value or pivot_table_indizes)):
134             match group_by_function:
135                 case 'max':
136                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).max()
137                 case 'min':
138                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).min()
139                 case 'head':
140                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).head()
141                 case 'sum':
142                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).sum()
143                 case 'mean':
144                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).mean()
145         else:
146             panda_table_data = panda_table_data
147 
148         # pivot if pivot is selected
```

```
149         if pivot_table_column or pivot_table_value or pivot_table_indizes:
150             if type(pivot_table_agg_function) is list:
151                 agg_tuple = tuple(pivot_table_agg_function)
152                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
153                                                   columns=pivot_table_column,
    values=pivot_table_value,
154                                                   aggfunc=agg_tuple, margins=
    margin, margins_name=margin_name)
155             elif type(pivot_table_agg_function) is dict:
156                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
157                                                   columns=pivot_table_column,
158                                                   values=pivot_table_value,
    aggfunc=pivot_table_agg_function,
159                                                   margins=margin, margins_name=
    margin_name)
160             else:
161                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
162                                                   columns=pivot_table_column,
    values=pivot_table_value,
163                                                   aggfunc=pivot_table_agg_function
    , margins=margin,
164                                                   margins_name=margin_name)
165
166         # set column operations
167         if column_operations:
168             for column_ops in column_operations:
169                 operation_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('operation_function')
170                 operation_columns = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('columns')
171                 operation_axis = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('axis_number')
172                 match operation_function:
173                     case 'max':
174                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].max()
175                     case 'min':
176                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].min()
177                     case 'head':
178                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].head()
```

```
                    case 'sum':
                        panda_table_data[column_ops] = panda_table_data[operation_columns].sum(axis=operation_axis)
                    case 'mean':
                        panda_table_data[column_ops] = panda_table_data[operation_columns].mean()
                    case 'diff':
                        panda_table_data[column_ops] = panda_table_data[operation_columns[1]] - panda_table_data[operation_columns[0]]


        # order by
        if order_by:
            panda_table_data.sort_values(by=order_by, inplace=sort_inplace, ascending=sort_acending)

        # rename columns
        if rename_columns:
            panda_table_data = panda_table_data.rename(columns=rename_columns)

        # set indices
        if index:
            index_values = panda_table_data.get(index)
            #panda_table_data.set_index(index_values)
            panda_table_data = panda_table_data.set_index(index)

        # rename indices
        if pivot_table_rename_indizes:
            panda_table_data = panda_table_data.rename_axis(index=pivot_table_rename_indizes)

        #   Plotter
        #   Plotter Process starts here!
        if autopct:
            panda_chart_plot = panda_table_data.plot(kind=chart, title=title, y=y_axis_columns, x=x_axis_columns, xlabel=x_axis_title,
                                                     ylabel=y_axis_title, grid=grid, stacked=stacked, legend=legend,
                                                     secondary_y=secondary_y, subplots=subplots, rot=rot, fontsize=fontsize,
                                                     figsize=figsize, autopct=autopct)
        else:
            panda_chart_plot = panda_table_data.plot(kind=chart, title=title, y=y_axis_columns, x=x_axis_columns, xlabel=x_axis_title,
                                                     ylabel=y_axis_title, grid=grid, stacked=stacked, legend=legend,
                                                     secondary_y=secondary_y, subplots=subplots, rot=rot, fontsize=fontsize,
```

```
                                                figsize=figsize)

        match chart:
            case 'pie':
                panda_chart_plot[0].legend(loc=loc, bbox_to_anchor=bbox_to_anchor)
                plt = panda_chart_plot[0].get_figure()
                plt.savefig(image_path, bbox_inches='tight')
            case _:
                plt = panda_chart_plot.get_figure()
                plt.savefig(image_path, bbox_inches='tight')


    return "blade runner"
panda_diagram_plotter_config = load_configuration('pandas_data_chart_plotter_conf.
    yaml')
create_panda_diagram_plotter(panda_diagram_plotter_config)
```

Listing 98: Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm

X.XII pandas_data_chart_plotter_conf.yaml

```yaml
diagram_inventory:
  - "tps_mixed"
  - "db_inventory_per_rdbms"
  - "db_inventory_per_os"
panda_diagram_plotter:
  tps_mixed:
    id: "tps_mixed"
    startpath: "parentdir"
    destination_path: "source/pandas_data_chart_plotter"
    imagename: "tps_mixed.png"
    datafile_path: "source/pandas_data_chart_plotter"
    datafile: "tps_evaluation.csv"
    separator: ","
    decimal: "."
    x-axis-columns: "Varianten"
    y-axis-columns:
      - "2. Iteration"
      - "3. Iteration"
      - "4. Iteration"
    x-axis-title: "Varianten"
    y-axis-title: "Transaktionen pro Sekunde (tps) Bsp."
    title: "Transaktionen pro Sekunden - mixed"
    chart-index:
    chart-kind: "bar"
    chart-designs:
      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_colums:
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_indizes_visible: False
53         pivot_rename_indizes:
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_clausel: "tps_typ == 'mixed'"
63      sorting:
64        order_by:
65        sort_acending: 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_colums:
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_indizes_visible: False
```

```
123        pivot_rename_indizes:
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_clausel:
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_colums:
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"
```

```
    where_clausel:
    sorting:
     order_by:
     sort_acending: False
     sort_inplace: True
    margins:
     margin: False
     margin_name:
    table_styles:
     selector: "caption"
     props:
      caption-side: "below"
      position: "H"
      sparse_columns: True
      longtable: True
      resize_textwidth: False
      linebreak_columns:
      table_header: True
```

Listing 99: Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme