Kantonsspital Graubünden
Departement ICT
Data Center
www.ksgr.ch

Michael
Graber
Datenbank Administrator
michael.graber@ksgr.ch
+41 81 256 68 25

Lehrgang: Diplomarbeit Technik und Wirtschaftsinformatik 2023-2024

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

Fachexperte: **Norman Süsstrunk**

Student: **Michael Graber**

24.05.2024

Management Summary

Das KSGR (Kantonsspital Graubünden) betreibt über zwanzig PostgreSQL Datenbanken.
Diese Datenbanken werden nicht auf einem zentralen Cluster betrieben, sondern oft als single Instanz auf diversen Servern.
In Zukunft wird das KSGR vermehrt auf Kubernetes setzen, was eine noch unbekannte Anzahl weiterer PostgreSQL Datenbanken mit sich bringen wird.
Auch sollen eine ganze Reihe MariaDB / MySQL und Oracle Datenbanken auf PostgreSQL migriert werden.

Um den administrativen Aufwand zu verringern, die Anzahl der Server zu konsolidieren und um die Kontrolle zu erhöhen,
soll nun eine hochverfügbare Clusterlösung für die PostgreSQL Datenbanken evaluiert werden.
Ziel dieser Arbeit ist demnach, die Evaluation einer Clusterlösung aus drei Evaluationssystemen zu ermitteln und das geeignetste als Testsystem zu implementieren.
Da, dass KSGR in Zukunft stark auf die Cloud setzen wird, muss ein monolithisches und mindestens ein verteiltes System genauer untersucht werden.

Auf dem Markt lassen sich einige Open-Source, Closed-Source- und proprietäre Lösungen finden.
Der Scope der Arbeit lag dabei nur auf den Open-Source-Lösungen.
In der Evaluation wurden die drei Systeme Patroni, StackGres und YugabyteDB genauer untersucht und jeweils als Evaluationssystem installiert und getestet.
Nach dem Testing und den Benchmark-Tests setzte sich Patroni gegen die beiden anderen Varianten durch.
YugabyteDB hat sich indes für als PostgreSQL Clustersystem für Applikationen, die in Zukunft auf Kubernetes betrieben werden sollen.

Im Anschluss wurde ein Patroni-Testsystem mithilfe des Ansible-Repositories vitabacks / postgresql_cluster installiert.
Das Ansible-Repository bietet eine Reihe von Playbooks, von welchen das Deployment-Playbook, das Maintenance-Playbook sowie ein Playbook zur Erweiterung von Patroni- und Proxy Nodes getestet wurden.
Die Umsetzung mit vitabacks / postgresql_cluster bestand die Tests.
Bis zur Produktivsetzung müssen im Nachgang der Diplomarbeit noch Arbeitspakete umgesetzt werden,
welche während der Diplomarbeit nicht umgesetzt werden konnten, da die Voraussetzungen nicht gegeben waren.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Ausgangslage und Problemstellung . . . 1

1.1.1 Das Kantonsspital Graubünden . . . 1

1.1.2 Die ICT des Kantonsspital Graubünden . . . 3

1.1.3 Rolle in der ICT vom Kantonsspital Graubünden . . . 5

1.1.4 Ausgangslange . . . 6

1.1.5 Problemstellung . . . 11

1.2 Zieldefinition . . . 17

1.3 Abhängigkeiten . . . 23

1.4 Risikomanagement . . . 25

1.4.1 Riskcontrolling . . . 27

1.5 Vorgehensweise und Methoden . . . 32

2 Projektmanagement **33**

2.1 Projektplanung . . . 33

2.1.1 Projektcontrolling . . . 33

2.1.2 GANTT-Diagramm . . . 34

2.2 Expertengespräche . . . 38

3 Umsetzung **39**

3.1 Evaluation . . . 39

3.1.1 Exkurs Architektur . . . 39

3.1.2 Erheben und Gewichten der Anforderungen . . . 45

3.1.3 Testziele erarbeiten . . . 49

3.1.4 PostgreSQL Benchmarking . . . 52

3.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen . . . 57

3.1.6 Vorauswahl . . . 80

3.1.7 Installation verschiedener Lösungen . . . 80

3.1.8 Testing Evaluationssysteme . . . 98

3.1.9 Gegenüberstellung der Lösungen . . . 101

3.1.10 Entscheid . . . 118

3.2 Aufbau und Implementation Testsystem . . . 119

3.2.1 Architektur . . . 119

3.2.2 Bereitstellen der Grundinfrastruktur . . . 121

3.2.3 Installation und Konfiguration PostgreSQL HA Cluster . . . 121

3.2.4 Technical Review der Umgebung . . . 126
3.3 Testing . . . 130
3.3.1 Protokollierung . . . 133
3.3.2 Review und Auswertung . . . 134
3.4 Maintenance-Tool . . . 134
3.5 Monitoring . . . 138
3.6 Troubleshooting und Lösungsfindung . . . 140
3.7 Ausblick . . . 142

4 Resultate **143**
4.1 Zielüberprüfung . . . 143
4.1.1 SQL optimierungen - Indizes tracken und verwalten . . . 144
4.1.2 Testziele . . . 144
4.2 Schlussfolgerung . . . 144
4.2.1 Kubernetes / Rke2 . . . 144
4.2.2 YugabyteDB . . . 145
4.2.3 StackGres - Citus . . . 145
4.2.4 Patroni . . . 145
4.2.5 vitabacks / postgresql_cluster . . . 146
4.3 Weiteres Vorgehen / offene Arbeiten . . . 146
4.4 Persönliches Fazit . . . 147

Abbildungsverzeichnis **149**

Tabellenverzeichnis **153**

Listings **155**

Literatur **161**

Abkürzungen **167**

Glossar **169**

Anhang **i**
I Disposition . . . i
II Arbeitsrapport . . . ii
III Protokoll - Fachgespräche . . . v
III.I Fachgespräch 14.02.2024 . . . v
III.II Fachgespräch 27.03.2024 . . . viii
IV Statusbericht . . . xii
IV.I Status Report 1 . . . xii

IV.II Status Report 2 . . . xiv
IV.III Status Report 3 . . . xvii
IV.IV KSGR Abschlussbericht . . . xix
V Kommentare / Anmerkungen . . . xxiv
VI Evaluation . . . xxvii
VI.I Maintenance - CloudNativePG . . . xxvii
VI.II Maintenance - Patroni . . . xxx
VI.III Maintenance - StackGres - Citus . . . xxxiii
VI.IV Maintenance - YugabyteDB . . . xxxix
VII Evaluationssysteme - Installation . . . xlii
VII.I rke2 . . . xlii
VII.II yugabyteDB . . . xlix
VII.III sks9016 - yugabyteDB . . . lxxiv
VII.IV Patroni . . . lxxv
VII.V Stackgres mit Citus . . . xciv
VIII Evaluationssysteme - Benchmarking . . . cxiv
VIII.I yugabyteDB . . . cxiv
VIII.II Patroni . . . cxvi
VIII.III StackGres - Citus . . . cxviii
IX Evaluationssysteme - Testing . . . cxix
IX.I StackGres - Citus . . . cxix
IX.II YugabyteDB . . . cxxiii
X Testsystem - Installation . . . cxxvii
X.I Prequenteries . . . cxxviii
X.II Deployment . . . cxxix
X.III Maintenance . . . clxv
X.IV Prequenteries - Erweiterungstests . . . clxxvi
X.V Haproxy erweitern . . . clxxvi
X.VI Patroni Node . . . clxxvii
XI Testsystem - Testing . . . clxxviii
XII Maintenance-Tool . . . clxxviii
XII.I Maintenance-Tool - Bloated Tables . . . clxxviii
XII.II Maintenance-Tool - AUTOVACUUM . . . clxxxiv
XIII Monitoring - PRTG . . . cxc
XIV Exkurs Architekturen - Umsysteme und Prinzipien . . . cxcv
XIV.I Raft-Consensus . . . cxcv
XIV.II local-path-provisioner . . . cxcvi
XV Python Utils . . . cxcviii
XV.I zotero.py . . . cxcviii

XV.II zotero_bibtex_configuration.yaml cciii
XV.III zotero_biblatex_keystore.yaml cciii
XV.IV riskmatrix.py ccix
XV.V riskmatrix_plotter_conf.yaml ccxii
XV.VI riskmatrix_xy_axis_tuple_matrix.yaml ccxvi
XV.VII cost_benefit_diagram.py ccxvii
XV.VIII cost_benefit_diagram_plotter_conf.yaml ccxviii
XV.IX pandas_dataframe_to_latex_table.py ccxix
XV.X csv_to_latex_diplomarbeit.yaml ccxxv
XV.XI pandas_data_chart_plotter.py ccxliii
XV.XII pandas_data_chart_plotter_conf.yaml ccxlviii

1 Einleitung

1.1 Ausgangslage und Problemstellung

1.1.1 Das Kantonsspital Graubünden

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

Das KSGR deckt dabei die Spitalregion Churer Rheintal ab.

Abbildung 1.1: Spitalregionen Kanton Graubünden[67]

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

Abbildung 1.2: Wahlkreise Kanton St. Gallen[95]

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[61]

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

Basierend auf der Liste *DB-Engines Ranking*[58] 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 - Datenbanken

Abbildung 1.7: Datenbanken - Aufgeschlüsselt nach RDBMS - Instanzen

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

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

Continued on next page

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

OS	RDBMS	Appliance	Datenbanken	Instanz
	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

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

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

Aktuell werden die Daten des Zabbix der Netzwerktechniker auf eine MariaDB Datenbank gespeichert, dies soll sich aber ändern. Da das Zabbix alle Netzwerkgeräte Überwacht, pro Sekunde werden im Moment ca. 950 Datenpunkte abgefragt,
pro Sekunde werden 23 Connections geöffnet resp. offen gehalten. Die 23 Connections setzen pro Sekunde über 7’000 Transaktionen ab, wobei über 4’000 direkt von den Clients kommen.
Mittlerweile ist die Datenbank auf über 230GiB angewachsen.
Genauere Informationen sind im Kapitel **Benchmark Settings** zu finden.

1.1.5 Problemstellung

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

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

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

Dadurch ergeben sich nach BSI folgende Risiken:

Identifikation						Abschätzung		Behandlung			
								Massnahmen	Zielwert		
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	ergreifen?	WS	SM	Massnahme
1	I	G0.22	Manipulation von Informationen	Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja	1	2	Best-Practice bei Härtung der Systeme. Redundanzen einführen
2	A	G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme ausfallen	Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung.	4	4	Ja	2	2	Redundanzen einführen
3	C, I, A	G0.26	Fehlfunktion von Geräten oder Systemen	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	2	2	Systeme zentralisieren Lifecycle etablieren
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	2	3	Systeme zentralisieren
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	1	2	Monitoring verschärfen
6	C, I, A	G0.31	Fehlerhafte Nutzung oder Administration von Geräten und Systemen	Durch die Vielfalt an Datenbankversionen und Betriebssystemen und Plattformen worauf diese betrieben werden, besteht allen voran das Risiko einer fehlerhaften Administration und Konfiguration.	Doch die folgen bleiben nichtsdestotrotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen. Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist.	4	3	Ja	2	3	Systeme zentralisieren
7	C, I, A	G0.32	Missbrauch von Berechtigungen	Obwohl das Microsoft Active Directory die zentrale Benutzerverwaltung ist, sind die wenigsten Datenbanken an dieses angeschlossen. Hinzu kommt der Umstand, dass in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen. Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen wissentlich oder unwissentlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen.	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	2	2	Systeme zentralisieren Übergreifendes Berechtigungskonzept einführen Monitoring der Zugriffe
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	1	3	Systeme zentralisieren Einheitliches Backupkonzept Regelmässige Restore-Tests

Tabelle 1.4: Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken

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

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

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

Abbildung 1.8: Risikomanagement PostgreSQL

1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [77, 83] folgende zehn Tasks die zum täglichen Alltag gehören:

Nr.	Aufgabe	Beschreibung	Wichtigkeit
1	Failover	In einem Fehlerfall soll die DB-Node auf einen Standby-Node übergeben werden. Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den primären Node zurückgesetzt werden.	Hoch
2	Failover Restore	Dabei darf es zu keinem Datenverlust kommen, also alle Daten die auf dem Standby-Node erfasst wurden, müssen beim Failover Restore auf den primären DB-Node zurückgeschrieben werden.	Hoch
3	Filesystem Management	Die Datenmenge in 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 observiert 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[56]. 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[3]. Während der Hauptarbeitszeit muss dies mindestens alle 90 Minuten geprüft und ggf. reagiert werden.	Hoch
8	Housekeeping	Mit Housekeeping Jobs werden regelmässig Trace- und Alertlogfiles aufgeräumt, um Platz auf den Disken zu sparen, aber auch, um die Übersichtlichkeit zu wahren.	Mittel
9	Verwalten von DB Objekten	Regelmässig müssen DB Objekte wie Datenbanken, Tabellen, Trigger, Views etc. angepasst oder erstellt werden. Dies richtet sich nach den 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[56].	2	Kann	
7	Health Checks und Aktionen (Maintenance)	Hoch	Tabellen und Indizes können sich aufblähen (bloated table / bloated index) Ist ein Index aufgebläht, kann dies mittels eines REINDEX mit geringem Impact auf die Datenbank gelöst werden[3].	2	Muss	
8	Housekeeping	Mittel	Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden.	3	Kann	
9	Verwalten von DB Objekten	Tief	Keine Automatisierung möglich	5		
10	User Management	Tief	Regelmässige Reports sollen User aufzeigen, die seit mehr als einer Woche nicht mehr aktiv waren.	4	Kann	

Tabelle 1.6: Automatisierung Administrativer Aufgaben

Mit dieser Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

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

Daraus ergeben sich folgende Ziele:

Nr.	Ziel	Beschreibung	Priorität
1	Evaluation	Am Ende der Evaluationsphase müssen mindestens drei Methoden für einen PostgreSQL HA Cluster müssen 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, bloated Tables), der Replikation (Replay Lag, Standby alive) und des PostgreSQL HA Clusters müssen überwacht werden.	Mittel
6	AUTOVACUUM - Parameter verwalten	Täglich müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden	Mittel
7	SQL optimierungen - Indizes tracken und verwalten	Täglich fehlende Indizes automatisiert erstellen und nicht mehr verwendete Indizes automatisiert entfernen	Mittel
8	Maintenance - Indizes säubern	Täglich bloated Indices, also aufgeblähte Indizes, automatisiert erkennen und mittels REINDEX bereinigen	Hoch
9	Housekeeping - Log Rotation	Die Log Rotation muss aktiviert werden. Die Logs müssen aber auch in das KSGR-Log Repository geschrieben werden	Hoch
10	User Management - Monitoring	Nicht verwendete User sollen einmal pro Woche automatisiert erkannt und in einem Report gemeldet werden.	Tief
11	Evaluationsziel	Am Ende der Evaluationsphase muss ein Entscheid getroffen worden sein, welche Methode verwendet wird.	Hoch
12	Installationsziel	Die Testinstallation muss lauffähig sein und zudem alle Anforderungen und Ziele (3 und 4) erfüllen	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 sekundären Nodes muss möglich sein 3. Der Fallback auf den primären Node muss erfolgreich sein, unabhängig davon, ob ein Failover oder Switchover stattgefunden hat 4. Das Timeout bei einem Failover / Switchover muss unterhalb der Default Timeouts der Applikationen GitLab und Harbor liegen. 5. Das Replay Lag zwischen Primary und Secondary darf beim Initialen Start nicht über eine Minute dauern oder 1KiB nicht überschreiten	Hoch

Tabelle 1.7: Ziele

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

	Produkt	**Beschreibung**
Storage	HPE 3PAR 8450 SAN Storage System	
Virtualisierungsplattform	VMware® vSphere®	
Primäres Backupsystem	VEEAM Backup System	
Provisioning / Lifecycle Management System	Foreman	Ist zurzeit nur für Linux angedacht
Primäre Linux-Distribution	Debian	
Sekundäre Linux-Distributionen	Rocky Linux Oracle Linux RedHat Enterprise Linux (RedHat 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
Log-Plattform / 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[110] folgende Abgrenzungen:

Abbildung 1.9: Systemabgrenzung

1.3 Abhängigkeiten

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

Nr.	Objekt	Abhängigkeit	Beschreibung	Status	Risiko	Impact
1	Foreman	VMs	Das Lifecycle Management und Provisioning System muss zur Verfügung stehen um in der Evaluationsphase Develop-VMs und in der Installationsphase Test-VMs erstellen zu können.	Im Moment ist Foreman in einer Proof of Concept Phase.	Das Risiko besteht. dass Foreman nicht betriebsbereit ist	VMs müssen von Hand aufgesetzt werden. Entsprechend wird sehr viel mehr Zeit in der Evaluations- und Installationsphase benötigt.
2	Storage	Speicher für VMs / Daten	Es müssen genügend Kapazitäten auf dem Storage vorhanden sein, um die VMs und Datenbanken in Betrieb zu nehmen	Storage wurde bereits erweitert, neue Disks für den SAN Storage wurden bestellt.	Auf dem SAN ist keine Kapazität mehr vorhanden	Es können keine VMs oder Datenbanken erstellt werden
3	Log Management / SIEM System	Sichern der Logfiles für Log Rotation	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[4]	Backup	Kubernetes Nodes und Pods können nicht mit Klassivem Veeam gesichert werden. Hierfür hat Veeam mit der Version V12 die spezialisierte Veeam Kasten K10 Lösung heraus.	Kann erst beim 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

1.4 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 1.10: Risiko-Matrix der Diplomarbeit

Daraus ergeben sich, mit den Massnahmen, folgende Risikomatrizen:

Abbildung 1.10: Risikomanagement Projekt

1.4.1 Riskcontrolling

1.4.1.1 Neu erfasste Risiken

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

Tabelle 1.11: Neu Erkannte / Erfasste Risiken

1.4.1.2 Assessment 21.03.2024

ID	Risiko	Assessment-Datum	WS	SM	Status	Ergriffene Massnahmen	Wirksamkeit	Begründung
1	Fehlende Ressourcen	21.03.2024	3	4	hoch	Dokumentation ausserhalb Arbeitszeit	begrenzt	Mentale Ressourcen setzen Limits. ExaCC Server werden mitten während der Diplomarbeit geliefert.
2	HP-UX Ablöseprojekt	21.03.2024	5	4	sehr hoch	Ressourcen reserviert	begrenzt	Von KSGR Seite fehlt eine Stellvertretung. Mithilfe notwendig.
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	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[4] nicht betriebsbereit	21.03.2024	5	5	sehr hoch	noch keine		

Tabelle 1.12: Risiko-Assessment 21.03.2024

Abbildung 1.11: Riskikomatrix - Assessment 21.03.2024

1.4.1.3 Assessment 29.04.2024

ID	Risiko	Assessment-Datum	WS	SM	Status	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	2	3	hoch	Ziele SMART definiert	wirksam	Vorauswahl getroffen, nur noch drei Evaluiert.
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[4] nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
10	Kubernetes Testumgebung	29.04.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
11	Verzug	29.04.2024	4	5	sehr hoch			

Tabelle 1.13: Risiko-Assessment 29.04.2024

Abbildung 1.12: Riskikomatrix - Assessment 29.04.2024

1.4.1.4 Assessment 13.05.2024 / Abschluss

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	2	3	hoch	Ziele SMART definiert	wirksam	Vorauswahl getroffen, nur noch drei Evaluiert.
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[4] nicht betriebsbereit	21.03.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
10	Kubernetes Testumgebung	29.04.2024	5	1	sehr hoch	keine		Es muss nur ein Testsystem aufgebaut werden. Da dies Patroni ist, wird YugabyteDB später installiert.
11	Verzug	29.04.2024	4	5	sehr hoch			

Tabelle 1.14: Risiko-Assessment 13.05.2024

Abbildung 1.13: Riskikomatrix - Assessment 13.05.2024

1.5 Vorgehensweise und Methoden

2 Projektmanagement

2.1 Projektplanung

2.1.1 Projektcontrolling

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

Tabelle 2.1: Projektcontrolling

2.1.2 GANTT-Diagramm

Diplomarbeit - PostgreSQL HA Cluster - Konzeption und Implementation - aut4.mpp

Nr.		Vorgangsm	Vorgangsname	Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang
1			**Phase I: Themenerarbeitung- und eingabe**	**81 Tage**		**Nein**	**Don 17.08.23**	**Don 07.12.23**	**Don 17.08.23**
2			Einführung Diplomarbeit	81 Tage		Nein	Don 17.08.23	Don 07.12.23	Don 17.08.23
3			Ideensammlung	30 Tage		Nein	Don 17.08.23	Mit 27.09.23	NV
4			Abgabe Ideensammlung	1 Tag		Ja	Mit 27.09.23	Mit 27.09.23	NV
5			Disposition erstellen	2.5 Tage		Nein	Mit 15.11.23	Son 19.11.23	Mit 15.11.23
6			Abgabe Disposition	1 Tag		Ja	Mon 20.11.23	Mon 20.11.23	Mon 20.11.23
7			Bewilligung Disposition durch FV	1 Tag	6	Ja	Don 07.12.23	Don 07.12.23	Don 07.12.23
8			**Phase II: Vorarbeitungsarbeiten**	**48 Tage**	**7**	**Nein**	**Fre 08.12.23**	**Die 13.02.24**	**NV**
9			Vorebereitungsmassnahmen	43 Tage	7	Nein	Fre 08.12.23	Die 06.02.24	NV
10			Abgabe Statusbericht	1 Tag	9	Ja	Mit 07.02.24	Mit 07.02.24	NV
11			Vernissage	1 Tag?	10	Ja	Die 13.02.24	Die 13.02.24	NV
12			**Phase III: Hauptarbeiten**	**64.75 Tage?**	**11**	**Nein**	**Mit 14.02.24**	**Don 06.06.24**	**NV**
13			**Haupteil Diplomarbeit**	**55.75 Tage?**		**Nein**	**Mit 14.02.24**	**Fre 24.05.24**	**NV**
14			**Umsetzung**	**25 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 27.03.24**	**NV**
15			**Evaluation**	**13 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 06.03.24**	**NV**
16			Anforderungskatalog	6 Tage		Nein	Mit 21.02.24	Fre 08.03.24	Mit 21.02.24
17			Vorbereitung Benchmarking	1 Tag	16	Nein	Mon 18.03.24	Fre 22.03.24	Mon 18.03.24
18			Analyse PostgreSQL HA Cluster Lösungen	4 Tage	16	Nein	Mon 11.03.24	Sam 20.04.24	Mon 11.03.24
19			Gegenüberstellung	2 Tage	16;17;18	Nein	Mon 22.04.24	Don 25.04.24	Mon 22.04.24
20			Variantenentscheid	1 Tag?	19	Ja	Fre 26.04.24	Fre 26.04.24	Fre 26.04.24
21			**Aufbau und Implementation Testsystem**	**8 Tage?**	**20**	**Nein**	**Fre 08.03.24**	**Mit 20.03.24**	**NV**
22			Basisinfrastruktur	2 Tage		Nein	Fre 19.04.24	Mon 29.04.24	Fre 19.04.24
23			Installation und Konfiguration PostgreSQL HA Cluster	5 Tage	22	Nein	Fre 19.04.24	Don 09.05.24	Fre 19.04.24
24			Technical Review	1 Tag?	23	Ja	Mit 01.05.24	Mit 01.05.24	Mit 01.05.24
25			**Testing**	**4 Tage?**	**24**	**Nein**	**Fre 22.03.24**	**Mit 27.03.24**	**NV**
26			Testing Testsystem	1 Tag	24	Nein	Fre 03.05.24	Don 09.05.24	Fre 03.05.24
27			Protokollierung	2 Tage	26	Nein	Fre 03.05.24	Fre 03.05.24	Fre 03.05.24
28			Review und Auswertung	1 Tag?	27	Ja	Son 12.05.24	Son 12.05.24	Son 12.05.24
29			Troubleshooting und Lösungsfindung	1 Tag?	25	Nein	Mit 27.03.24	Die 30.04.24	Mit 27.03.24
30			**Resultate**	**6 Tage?**	**25**	**Nein**	**Die 02.04.24**	**Mit 10.04.24**	**NV**
31			Zielüberprüfung	4 Tage		Nein	Son 12.05.24	Son 12.05.24	Son 12.05.24
32			Schlussfolgerung	1 Tag?	31	Nein	Son 12.05.24	Son 12.05.24	Son 12.05.24
33			Weiteres Vorgehen / offene Arbeiten	1 Tag?		Nein	Son 12.05.24	Son 12.05.24	Son 12.05.24
34			Persönliches Fazit	1 Tag	31;32	Nein	Son 12.05.24	Son 12.05.24	Son 12.05.24
35			1. Expertengespräch	1 Tag?		Ja	Mit 14.02.24	Mit 14.02.24	Mit 14.02.24
36			2. Expertengespräch	1 Tag?		Ja	Die 26.03.24	Mit 27.03.24	Die 26.03.24
37			Letztes Expertengespräch	1 Tag?		Ja	Mit 15.05.24	Mit 15.05.24	NV
38			Dokumentation	55.75 Tage		Nein	Mit 14.02.24	Mon 13.05.24	Mit 14.02.24
39			Puffer	9 Tage		Nein	Die 16.04.24	Die 30.04.24	NV

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

Nr.		Vorgangsm	Vorgangsname	Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang
40			Abgabe Diplomarbeit	1 Tag?	13;30;37	Ja	Fre 24.05.24	Fre 24.05.24	NV
41			Vorbereitung Präsentation	5 Tage	40	Nein	Die 28.05.24	Mon 03.06.24	NV
42			Präsentation und Fachgespräch	1 Tag?	40;41	Nein	Die 04.06.24	Die 04.06.24	NV
43			Diplomausstellung	1 Tag?	42	Nein	Mit 05.06.24	Mit 05.06.24	NV
44			Diplomfeier	1 Tag?	42;43	Nein	Don 06.06.24	Don 06.06.24	NV

3. Qtl, 2023
Jul | Aug | Sep

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

Diplomarbeit - PostgreSQL HA Cluster - Konzeption und Implementation - aut4.mpp

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

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

2.2 Expertengespräche

Folgende Expertengespräche fanden statt:

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

Tabelle 2.2: Fachgespräche

Das Protokoll ist im Anhang zu finden.

3 Umsetzung

3.1 Evaluation

3.1.1 Exkurs Architektur

3.1.1.1 Sharding

3.1.1.1.1 Vertikales / Horizontales Sharding

Tabellen können horizontal oder vertikal partitioniert werden.

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

Komplette Tabelle

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

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

Vertikale Partitionen

Abbildung 3.1: Sharding - Vertikale Partitionierung

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

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

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

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

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

Abbildung 3.2: Sharding - Horizontales Partitionierung

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

3.1.1.1.2 Key Based Sharding

Hierbei wird das sharding anhand eines oder mehreren Keys ausgeführt.

3.1.1.1.3 Range Based Sharding

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

3.1.1.1.4 Directory Based Sharding

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

3.1.1.1.5 Hash Based Sharding

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

3.1.1.2 Monolithische vs. verteilte SQL Systeme

Klassische SQL-Datenbanken sind monolithische Systeme, selbst wenn sie mittels Replikation eine Primary/Stand-by-Architektur aufweisen. Man kann mittels eines SQL-Proxys ein gewisses Mass an Load Balancing betreiben, hat aber immer noch das Problem, das es einen primären 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[81]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

Synchrone Replikation

Wenn bei einer synchronen Replikation eine Transaktion abgesetzt wird, wird der Commit auf der primären Seite erst gesetzt, wenn die Änderung auf der sekundären Seite oder den sekundären Seiten ebenfalls eingetragen und committet ist. Bis zu diesem Moment ist die Transaktion nicht als committet.
Dies wird dann zum Problem, wenn keine Verbindung mehr zu mindesten einer sekundären Seite vorhanden ist. Zudem wird die synchrone Replikation bei hohen Latenzen zum Bottleneck der Datenbank.

Asynchrone Replikation

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

3.1.1.4 Quorum

Ein Quorum-System soll die Integrität und Konsistenz in einem Datenbank-Cluster sicherstellen. Dabei gilt zu beachten, 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 Commitment von den Replica-Nodes gewartet werden muss.

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

Throughput
: Beschreibt, wie sich die Anzahl Nodes auf die Schreibgeschwindigkeit der Commitments auf die restlichen Nodes auswirkt.
Die Verdopplung der Server halbiert i. d. R. den Throughput.

Fehlertoleranz
: Beschreibt, wie viele Nodes ausfallen können, damit der Cluster noch arbeitsfähig ist. Wobei eine Erhöhung der Nodes von 3 auf 4 die Fehlertoleranz nicht erhöht, da nun eine Split-brain-Situation entstehen kann.

Hier ein Beispiel wie sie in den Artikeln [79, 91, 74] beschrieben werden. Es zeigt auf, ab wie vielen Nodes die Fehlertoleranz erhöht wird und wie sich der repräsentative Throughput verhält.

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

Tabelle 3.1: Quorum Beispiele

3.1.1.5 CAP-Theorem

Das CAP-Theorem besagt, das nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[63].

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

Verfügbarkeit - Availability
Jeder Client, der eine Anfrage sendet, muss auch eine Antwort erhalten. Unabhängig davon, wie viele Nodes im Cluster noch aktiv 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[76]:

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

Switchover

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

Restore

8. Mit der REST-API wird der Node erst mit dem `reinitialize` wiederhergestellt
   und dann mit einem Switchover wieder als Primary gesetzt.
9. Mit dem `patronictl`-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 des Failover müssen Daten via SQL
   eingeführt und ausgelesen werden.
3. Während des Failover muss mindestens eine längere Abfrage gestartet werden.

Sharding

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

Self Healing

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

3.1.3.1.3 YugabyteDB

Failover

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

Sharding

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

Self Healing

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

3.1.3.2 Evaluation - ERD self_healing_test

Die Tests müssen bei allen drei Varianten anhand der Datenbank `self_healing_test` durchgeführt werden.
Dabei werden die Tabellen im Hinblick auf das Citus Schema Based Sharding in Schemas organisiert.
Zwischen den einzelnen Schemas sollen einige Tabellen einen Foreign-Key auf andere Tabellen legen:

Abbildung 3.7: Testing - ERD DB self_healing_test

3.1.4 PostgreSQL Benchmarking

3.1.4.1 pgBench - Basis-Benchmarking

PostgreSQL bietet ein Benchmarking-Tool[70, 2], mit dem die DB vermessen werden kann.

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

Es gibt einiges zu beachten, wenn PostgreSQL einem Benchmarking unterzogen wird. Aus diversen Quellen [51, 47, 108, 2] sind dies folgende Anforderungen:

pgGather
Mit pgGather [78] 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 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 [104] 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 die Netzwerklatenz sowie andere Latenzen hinzu [89].

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[71]:

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 [46] aller Art für PostgreSQL an.

Percona arbeitet oft auf Basis von Patroni,
bietet aber auch eigene Erweiterungen und eigene Software an[41].

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[25].
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[93]. 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[68]:

- 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. [29]

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[37].

3.1.5.5.5 Architektur

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

Abbildung 3.14: pg_auto_failover-Architektur - Single Standby

Aber auch Multi-Nodes können eingebunden werden[40]:

Abbildung 3.15: pg_auto_failover-Architektur - Multi-Node Standby

pg_auto_failover kann Citus einbinden[14]. 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[16]:

- 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[6].

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[29].

3.1.5.7.4 Pooling

Patroni benötigt einen externen Pooler.
Hier wird oft PgBouncer [42] 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`[109] 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[28]:

- 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[26] implementiert.

3.1.5.8.4 Pooling

PgBounder[42] 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. [19, 11]

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[18], ab Version 11 wurde auch diese Variante gestrichen und man konnte noch eine 1:1 Repliacation auf einen Worker fahren[13]. 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[17]. Daraus ergibt sich aber ein nicht zu vernachlässigbarer Mehraufwand, wenn man self-healing Nodes implementieren möchte.
Jeder Node ist für sich genommen, eine eigene Zone, um sicherzustellen, dass es zu keinem Datenverlust kommt,
müsste jeder Shared-Node in eine der jeweiligen Zonen repliziert werden.
Das heisst, es müssten $Shard - Nodes^2$ Replika-Nodes erstellt werden, hier ein Schematisches-Beispiel mit drei Shard-Nodes:

Abbildung 3.25: StackGres-Citus - Shard-Replikation

Die Automation und nur schon die Konfiguration für das Mitskalieren, dürfte einiges an Zeit in Anspruch nehmen.
Eine nicht unwesentliche folge, wäre ein starker Rückgang des Throughput's und Performance-Einbussen.

Alternativ kann natürlich ein Klassischer Replika-Server verwendet werden, wo die ganze Datenbank gesichert wird.
Bis alle Daten wieder in den StackGres-Citus-Cluster zurückgeholt wurden, das Re-Balancing abgeschlossen ist usw.,
muss der ganze Cluster für die User unerreichbar sein, da dieser in dieser Zeit nicht mehr konsistent ist.

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

3.1.5.8.7 Maintenance

Bei StackGres und Citus ist Citusdata, welches mittlerweile zu Microsoft gehört, sehr aktiv.
Citusdata hält nicht nur die Community Standards ein sondern Commited regelmässig und räumt alten Code regelmässig auf.
Anders sieht es bei Ongres, dem Maintainer von StackGres aus.
Weder werden besonders viele Community Standards eingehalten, noch wird Regelmässig Commited.
Die aktivitäten gingen in den letzten Jahren und Monaten zurück, was dazu führt das immer wieder mal grössere Commits notwendig werden.
DIe genaue Analyse ist im Anhang - Maintenance zu finden.

3.1.5.9 yugabyteDB - Distributed SQL 101

yugabyteDB - Distributed SQL 101 ist eine nahezu komplett PostgreSQL kompatible Datenbank. Sie ist eine Distributed SQL Datenbank, also eine verteilte Datenbank[105].

3.1.5.9.1 Core-Features

Die wichtigsten Features von YugabyteDB sind[10]:

- 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[72].

3.1.5.9.5 API / Skripte

YugabyteDB bietet eigene APIs[7] und CLIs[15] 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 [49] geregelt:

Abbildung 3.30: yugabyteDB - Tablet - Replikationsfaktor

Durch das Raft-Protokoll werden die Tablet-Leader regelmässig gewechselt.
Das Raft-Protokoll ist im Anhang - Raft-Consensus beschrieben.

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`[45].

Abbildung 3.32: yugabyteDB - Zone outage Tolerance

3.1.5.9.7 Maintenance

YugabyteDB hat viele gemeldete Issues.
Zudem hält das Projekt nur die notwendigsten Community Standards ein.
Dafür werden in vielen kurzen Sprints viele Commits durchgeführt,
obwohl Yugabyte der einzige Maintainer ist.
Die genaue Analyse ist im Anhang - Maintenance zu finden.

3.1.5.9.8 Synergien und Mehrwert

Der grosse Benefit von YugabyteDB ist sein Distributed SQL Ansatz.

Zudem bietet YugabyteDB eine vollständige Cassandra Integration.

Der Benefit ist auf jeden Fall gegeben.

3.1.6 Vorauswahl

Folgende Lösungen werden nicht 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

Ursprünglich sollte auf jedem Patroni Server (`sks1232`, `sks1233` und `sks1234`) ein etcd-Node installiert werden.
Auch sollte auf `sks1234` der HAProxy installiert werden.
Im Kapitel Installation wird erklärt, wieso sich die Architektur folgendermassen aussieht:

Abbildung 3.34: Patroni - Evaluationsarchitektur

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

3.1.7.2.2 Installation

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

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

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

Listing 3.1: Patroni - etcd API V2 Error

Ursache war, dass etcd V3 ab Version v3.4 die API V2 nicht mehr per Default aktiv ist.
Man könnte bis v3.7 die API V2 noch aktivieren:

```
ETCDCTL_API=2
```

Listing 3.2: Patroni - etcd API V2 Enable

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

```
...
etcd3:
    host: <ip / hostname>:2379
...
```

Listing 3.3: Patroni - etcd3 Flag

Der Primary-Node konnte jeweils installiert und deployt werden.
Sobald aber jeweils die Replika-Nodes gestartet wurden, kam es zu einem Fehler.
Die Ursache war, dass es ja bereits einen Host für den jeweiligen Hostnamen resp. die jeweilige IP gab, nämlich den etcd-Node.
So kam es jeweils zu einem Key-Error.
Nach einigem Versuchen, etwa die Keys neu zu beschreiben, brach ich die übung ab.
Der etcd-Node wurde nun nur noch auf dem Server `sks9060` installiert.
Resultat war, dass der Cluster lauffähig wurde.

Passwörter können mit dem Bootstrap mitgegeben werden.
Dazu muss im `postgresql`-Segment das Subsegment `authentication` erstellt werden.
Der Replikationsuser muss mit einem subsegment `replication` und der postgres-User mit `superuser` angegeben werden:

```
...
postgresql:
    ...
    authentication:
        replication:
            username: replicator
            password: <password>
        superuser:
            username: postgres
            password: <password>
...
```

Listing 3.4: Patroni - Passwörter

Zuerst lief der Cluster nur Asynchron, auch die ersten beiden Benchmarks wurden so ausgeführt.
Der Cluster lässt sich nämlich nicht Synchron Bootstrapen, die Konfiguration muss nachträglich gemacht werden.
Dazu kann ein JSON mit den geänderten Konfiguration übergeben werden, in dieser Konfiguration musste dabei das yml-File mit der Konfiguration angegeben werden:

```
patronictl -c /etc/patroni/config.yml edit-config --apply - --force <<'JSON'
{
 synchronous_mode: "on",
 synchronous_mode_strict: "on",
 synchronous_node_count: 2,
 "postgresql":
   {
   "parameters":{
     "synchronous_commit": "on",
     "synchronous_standby_names": "*"
   }
  }
}
JSON
```

Listing 3.5: Patroni - Synchrone Replikation setzen

Die Vergrösserung der Disks hat nur einen begrenzten Impact.
Lediglich das Verzeichnis `pg_wal`, welches die `WAL`-Files aufnimmt, muss angepasst werden.
Um die Umstellung ohne Reinstallation durchführen zu können, muss mit symlinks gearbeitet werden.
Die Daten können via Tablespace auf die grösseren mounts gesetzt werden.

Die vollständige Dokumentation der Evaluationsinstallation ist im Anhang - Installation Patroni zu finden.

3.1.7.3 StackGres - Citus

3.1.7.3.1 Architektur

Für das Benchmarking wurde ein minimales setting ausgewählt.
Ein Coordinator und einen Shard-Node mit einem Leader- und Replica-Pod.

Abbildung 3.35: Stackgres - Citus - Evaluationsarchitektur Benchmarking

Für die Self Healing Tests wurde eine umfangreichere Architektur vorgenommen.
Es 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 3.6: 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 3.7: StackGres - values.yaml - Extension Proxy

Die Ursachenforschung hat viel Zeit in Anspruch genommen.
Es sind nebst dem Versuch, eine Freigabe via Pod-Affinität zu lösen, drei Tage verstrichen bis StackGres die Extensions ausführen konnte.

Anders als bei YugabyteDB, kann man das Web-GUI nicht mittels eines Load Balancer Exposing nach aussen präsentieren, auch wenn eine Cluster-IP gesetzt werden kann.
Soll das Web-GUI und die REST-API permanent von aussen verfügbar sein, muss auf dem rest-api Pod ein permanentes Forwarding implementiert werden.
Umsetzen lässt sich dies mittels der entsprechenden Keys im `values.yaml` oder den Parametern beim Deploy.

3.1.7.3.4 Cluster Deployment

Mit der Installation von StackGres steht noch keine Datenbank, es steht nur der Operator- und REST-API Pod.
Dabei muss unterschieden werden, ob eine normale Patroni-Instanz deployt werden soll oder eine Sharded-Instanz (mit Citus).

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

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

SGInstanceProfile
: Das Instanz-Profil definiert, wie viel CPU und Memory der Pod erhält.
Die konfigurationsmöglichkeiten gehen so tief,
das innerhalb von Pods auch Containern Ressourcen zugewiesen werden können.
Empfehlenswert ist, Coordinator und Worker zu trennen.
Ohne ein separates Instanz-Profil wird ein Standardprofil mit einer CPU und einem GiB Memory allokiert.

SGPostgresConfig
: PostgreSQL-Spezifische Einstellungen können hierüber konfiguriert werden.
Wie das Instanz-Profil auch, ist dies nicht zwingend.

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 3.8: StackGres-Citus - LoadBalancer -Annotation

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

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

Listing 3.9: StackGres-Citus - StorageClass -PVC Binding

Die Instanz-Profile lassen sich wie folgt zuweisen:

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

Listing 3.10: StackGres-Citus - Instanz-Profile

Beim Benchmarking kam es zu einem sehr unschönen Fehler.
PgBounder verlor die Verbindung.
Nach einer kurzen Suche, zeigte sich das es wohl einen Bug bei grösseren Workloads gibt, zumindest ist dies meine Interpretation.
Die Lösung bei einigen schien zu sein, dass das Pooling abgeschaltet wird[43].
Für das Benchmarking wurde dies dann auch umgesetzt.
Dies wird folgendermassen gemacht:

```
...
  coordinator:
    pods:
      ...
      disableConnectionPooling: true
...
```

Listing 3.11: StackGres-Citus - StorageClass -PVC Binding

Bei einem produktiven System müsste dieser Bug aber gefixt werden.

Bei einem Drei-Node Environment wie es für die Evaluation verwendet wird,
kommt es zu einem Konflikt, wenn drei Shard-Pods erzeugt werden.
In diesem Fall muss ein Testing- oder Development-Profil gesetzt werden.
Alternativ können Pod Anti-Affinities oder Pod Affinities gesetzt werden, was sich aber als schwieriges unterfangen auszeichnete da Ongres dies nicht wirklich Dokumentiert hat.
Daher wurde immer ein Testing-Profil gesetzt:

```
apiVersion: stackgres.io/v1alpha1
kind: SGShardedCluster
metadata:
  name: <cluster / db name>
  namespace: <cluster namespace>
spec:
  ...
  profile: "testing"
```

Listing 3.12: StackGres-Citus - Cluster Profil

Es gäbe zwar die Möglichkeit, Passwörter via Manifest zu setzen, aber dies funktioniert nicht für postgres- und replicator sowie backup.
Laut der StackGres-Dokumentation müssen z.B. die Passwörter vom `postgres`-User im Nachgang via SQL geändert werden.
Während der Evaluation wurde darauf verzichtet und das generische Passwort aus dem Cluster geholt.

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

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

```
...
coordinator:
  ...
  configurations:
    sgPostgresConfig: "<SGPostgresConfig Name>"
...
shards:
  ...
  configurations:
    sgPostgresConfig: "<SGPostgresConfig Name>"

```

Listing 3.13: StackGres-Citus - SGPostgresConfig

Die vollständige Dokumentation der Evaluationsinstallation ist im Anhang - Installation StackGres - Citus zu finden.

3.1.7.4 yugabyteDB

3.1.7.4.1 Architektur

Abbildung 3.39: YugabyteDB - Evaluationsarchitektur

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

3.1.7.4.2 Installation

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 3.40: 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
metadata:
  name: l2adv
  namespace: metallb-system
spec:
  ipAddressPools:
  - distributed-sql

```

Listing 3.14: 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[38], wo oben genannter Ansatz empfohlen wurde.

3.1.7.4.3 Konfiguration

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

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

```

Listing 3.15: yugabyteDB - Helm Chart Manifest - Detail Image

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

```
...
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 3.16: 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 3.17: 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 3.18: yugabyteDB - Helm Chart Manifest - Detail Replika

Wichtig ist auch, dass der `YSQL`-Dienst aktiv ist, damit PostgreSQL Abfragen abgesetzt werden können.
Deshalb muss der Dienst aktiv sein und darf nicht deaktiviert werden:

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

```
```

Listing 3.19: 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.

```yaml
...
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"
...

```

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

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

```
...
                "nodePathMap":[
                {
                        "node":"DEFAULT_PATH_FOR_NON_LISTED_NODES",
                        "paths":["<Lokaler Pfad>"]
                },
                {
                        "node":"<Nodename>",
                        "paths":["<Lokaler Pfad>"]
                },
...
```

Listing 3.21: local-path-provisioner nodePathMap

Hier ein Beispiel wie es mit den grossen Volumes aussieht:

```
...
                "nodePathMap":[
                {
                        "node":"DEFAULT_PATH_FOR_NON_LISTED_NODES",
                        "paths":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1183",
                        "paths":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1184",
                        "paths":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1185",
                        "paths":["/srv/data/local-path-provisioner"]
                }
                ]
...
```

Listing 3.22: local-path-provisioner nodePathMap Beispiel

Wird dies nicht gemacht, so wird auf den Default-Path geschrieben.
Das ist zufällig und hat dann zur Folge, dass alle Volumes auf einem Node präsentiert werden.
Was sehr schnell 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 3.23: yugabyteDB - StorageClass nodeAffinity

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

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

Die vollständige Dokumentation der Evaluationsinstallation ist im Anhang - Installation YugabyteDB zu finden.

3.1.8 Testing Evaluationssysteme

3.1.8.1 Patroni

Patroni funktionierte wie gewollt.
Da kein Connection Pooler auf dem Proxy-Host installiert wurde, kam nicht alles erfüllt werden.
Wichtig dazu ist zu sagen, dass die REST-API und das Command vollständig funktioniert.
Ein Switchover wurde mit folgendem Command ausgeführt:

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

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

Listing 3.24: Patroni - Testing - Switchover

Ein gestoppter Node wurde wie folgt wieder neu aufgebaut:

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

Listing 3.25: Patroni - Testing - Reinit

Das vollständige ergebnis:

Art	Test Case Nr.	Test Case	Erwartetes Ergebnis	Eingetretenes Ergebnis	Begründung
Failover	1	Automatismus	Wird der Primary Server vom Netz genommen, führt Patroni einen Failover auf einen Replika-Node	Eingetroffen	
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	Datenkonsistenz	Daten sind Konsistent und Inetger.	Eingetroffen
Sharding	5	Schutz vor Datenverlust	Die Daten müssen Konsistent und schnell auf die Tablets verteilt werden	Eingtetroffen
Self Healing	6	Node stellt sich selber wieder her	Tablet wird automatisch synchronisiert	Eingtetroffen

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[94].
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 muss pro Lauf erst initialisiert werden, dann kann mit dem eigentlichen Benchmarking gestartet werden.
Alle Benchmarking-Commands sind im Anhang - YugabyteDB Benchmarking Commands zu finden.

3.1.9.1.2 Patroni

Als die 250GiB DB getestet wurde, zeigte sich, dass die Parameter nicht darauf optimiert waren.
Die Standby-Server konnten die `WAL`-Files nicht mehr abarbeiten, so stauten sich auf dem Primary die Files und die Disk lief jeweils voll.
Es wurde an verschiedenen Stellschrauben geschraubt, etwa an der Anzahl Worker, der grösse der `WAL`-Files und anderen.
Mit den anpassungen wurde die Datenmenge gestemmt.

Filesystem und Cluster mussten beim initialiseren der letzten Benchmarks permanent überwacht werden.
Dafür wurden folgende Commands verwendet:

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

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

Listing 3.26: Patroni - Benchmarking - Monitoring

Alle Benchmarking-Commands und SQLs zur ermittlung der grösse sind im Anhang - Patroni Benchmarking Commands zu finden.

3.1.9.1.3 StackGres - Citus

Beim Benchmarking zeigten sich die Grenzen des Citus Shardings.
Bereits beim Lesen der Anleitung fiel auf, dass für das Sharding der Tabellen `pgbench_accounts` und `pgbench_history` jeweils neu initialisiert wurde[9].
Das hat einen besonderen Grund.
Wenn die Tabellen mittels des SELECT-Statements `create_distributed_table` einem Sharding unterzogen werden sollen, kommt es zu einer Fehlermeldung.
Die Shards würden mit folgenden SQL-Statements erzeugt:

```
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 3.27: Citus - Benchmarking - Distributed Table Sharding

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

Abbildung 3.41: Benchmarking - ERD pgbench

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

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

```
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 3.28: Citus - Benchmarking - Reference Table Sharding

Referenzierte Tabellen werden auf alle Shards repliziert und erfüllen somit die Anforderung an das Sharding.
Diese Art des Sharding wäre allerdings nur für kleinere Tabellen, Multi-Tenant Sharding (wenn Tabellen bei allen Tenants verfügbar sein sollen),
Tabellen die mit verschiedenen Distributed Tables gejoint werden oder wenn eben, wie in unserem Fall, Foreign-Key Constraints im Spiel sind[22].

Aber auch in diesem Fall muss das Sharding im Nachgang des `pgbench`-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.

Allerdings war es nicht möglich, die Shards mit rund 250GiB auszuführen.
Trotz grösserer Disk und mehr Ressourcen auf den Shards und Coordinators, brach das Sharding der Tabelle `pgbench_accounts` immer ab.
Die PostgreSQL-Instanz wurde optimiert, dass heisst, es wurden so viele Parameter wie möglich der Patroni-Evaluation gesetzt, allerdings gab es von Seitens StackGres einschränkungen.
Das SQL lief für jeweils knapp 13 minuten, ehe es zu einem Disconnect im Java-Code von einem der beiden Shards kam.
Die vermutung liegt nahe, dass es ein Memory-Leak im Code gibt, das Log gibt allerdings nicht genug her, um die Fehlerursache und eine mögliche Lösung zu finden.

☞ Aufgrund von diesem Fehler, ist das Benchmarking für StackGres unvollständig.

Alle Benchmarking-Commands und SQLs zur ermittlung der grösse sind im Anhang - StackGres - Citus Benchmarking Commands zu finden.

3.1.9.2 Benchmarks

Der vergleich zwischen den verschiedenen Varianten.

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

Abbildung 3.42: Benchmarks - tps

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

Abbildung 3.43: Benchmarks - latency

Die ersten beiden läufe mit Patroni wurde erst nur mit der Asynchronen Standard-Replikation von Patroni vorgenommen.

Später wurden die Benchmarks mit der Synchronen Replikation wiederholt.
Daraus ergab sich die Möglichkeit, beide Methoden direkt zu vergleichen:

Abbildung 3.44: Benchmarks - tps Patroni Replica

Abbildung 3.45: Benchmarks - latency Patroni Replica

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

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

auftreten können.

Abbildung 3.46: Benchmarks - Fehler bei mixed-Transaktionen

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

Abbildung 3.47: Benchmarks - Initialisierungszeit - sekunden

Abbildung 3.48: Benchmarks - Initialisierungszeit - minuten

Abbildung 3.49: Benchmarks - Initialisierungszeit - stunden

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

3.1.9.2.1 Gegebene Parameter und Annahmen

Es wird mit fünf Jahren gerechnet.
Daher werden also die Zeitaufwände der Betriebstasks pro Jahr berechnet und mit fünf multipliziert.
Es wurden also folgende Annahmen getroffen resp. folgende Parameter sind gegeben:

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

Tabelle 3.16: Kostenberechnung - Annahmen

3.1.9.2.2 Varianten

Patroni wurde in zwei Varianten aufgeteilt.
Einmal die Vanilla-Version, die manuell Aufgesetzt und verwaltet wird.
Also so wie es bei der Evaluation gemacht wurde.

Es gibt allerdings ein GitHub-Repository, welches die ganze Installation in Ansible-Playbooks verpackt hat.
Der ganze Prozess wurde analysiert und die Aufwände auch für diese Variante geschätzt.
An der Punkteverteilung ändert sich entsprechend nichts, da die Architektur die gleiche ist.
Diese Variante wird nachfolgend `Patroni - postgresql_cluster` oder vereinfacht nur `postgresql_cluster` genannt.

3.1.9.2.3 Zeitvergleiche

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

Phase	Subphase	Patroni - vanilla	Patroni - postgresql-cluster	StackGres - Citus	YugabyteDB
Initialer Aufwand					
	Basisinstallation	5.0	6.0	4.0	5.0
	Basiskonfiguration	5.0	5.0	5.0	5.0
	Backup Konfiguration	1.0	1.0	1.0	2.0
	Monitoring Konfiguration	2.0	2.0	2.0	2.0
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 3.17: Gemessene und Extrapolierte Aufwände Bsp.

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

Abbildung 3.50: Zeitaufwände pro Betriebstask

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

Abbildung 3.51: Zeitaufwände

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

Abbildung 3.52: Zeitaufwände summiert

3.1.9.2.4 Kostenvergleiche

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

Für die Betriebstasks wurden daher mit folgenden Kosten gerechnet:

Abbildung 3.53: Kostenaufwände pro Betriebstask

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

Abbildung 3.54: Kostenaufwände

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

Abbildung 3.55: Kostenaufwände Total

3.1.9.2.5 Schlussfolgerung

3.1.9.3 Kosten-Nutzen

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

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

berechnete Felder

Zellbezüge

Eingabefelder

Abbildung 3.56: Kosten-Nutzen-Analyse

Bei den Kosten zeigt sich, dass die Patroni-Variante `postgresql_cluster` klar am günstigsten kommt.
Dies resultiert auf den relativ Tiefen Installationskosten und der schnellen erweiterungen.

Da dass Repository schon einiges an Ansible-Playbooks mitbringt, ist auch der Erweiterungsaufwand gering.
YugabyteDB liegt auf Platz zwei, gefolgt vom Klassischen (vanilla) Patroni.

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

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

berechnete Felder

Zellbezüge

Eingabefelder

Abbildung 3.57: Kosten-Nutzen-Ranking

Daraus ergibt sich, folgendes Diagramm:

Abbildung 3.58: Kosten-Nutzen-Diagramm

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

3.1.10 Entscheid

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

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

3.2 Aufbau und Implementation Testsystem

3.2.1 Architektur

Das Testsystem wird mit Patroni umgesetzt.
Dabei werden folgende Komponenten eingesetzt:

Aufgabe	System
Orchestrator	Patroni
Proxy	Haproxy
Connection Pooler	PgBouncer
DCS	etcd

Tabelle 3.18: Testsystem - Komponenten

Entsprechend sieht das Architekturschema aus:

Abbildung 3.59: Testsystem - Architektur

`vitabacks / postgresql_cluster` hat eine Schwachstelle.
Der Connection Pooler ist nicht auf dem Level des Proxy sondern auf dem Patroni Node.

Das Resultat bei einem Node Failure ist zwangsläufig, dass die Connections bei einem Switchover / Failover unterbrochen werden.
Sollte die Applikation nicht fähig sein, die Connection zu halten und innerhalb eines Timeouts reconnecten zu können,
kommt es zu einem Finalen Disconnect.
Ohne weiteres lässt sich der Connection Pooler aber nicht auf den Proxy-Server verschieben, zu tief müsste in den Code eingegriffen werden.

3.2.2 Bereitstellen der Grundinfrastruktur

Folgende Server wurden mittels Foreman deployt:

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

Tabelle 3.19: Testsystem - Inventar

Um die erweiterungstests Fahren zu können und überhaupt Ansible Playbooks ohne die Red Hat Ansible Automation Platform deployen zu können,
musste Ansible auf einem separaten Host installiert werden.
Auch wurde ein Server benötigt, mit dem zuerst ein weiterer HAProxy und später Patroni Node hinzugefügt werden konnte.

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

Tabelle 3.20: Testsystem - Auxiliar Inventar

3.2.3 Installation und Konfiguration PostgreSQL HA Cluster

`vitabacks / postgresql_cluster` GitHub Repository hat zwei zentrale Komponten.

Das eine ist das `Inventory`-File, welches die IP-Adressen und Hostnamen der Server beinhaltet. Dabei ist zu beachten, dass die Host-Einträge überschrieben werden, wenn `hostname=<hostname>` eingetragen wird.
Was entsprechend zu Problemen mit den DNS-Einträgen, dem Active Directory oder anderen komponenten führen kann.

Die zweite zentrale Komponente ist das `main.yml`-File.
Über dieses YAML-File werden alle Aspekte des Patroni-Clusters einstellen.
Für dieses Testsystem zentrale Konfigurationsabschnitte waren:

Clustername

Der Clustename lautet «k8s-core-psql».
Dazu muss folgender Eintrag gesetzt werden: `patroni_cluster_name: ̈k8s-core-psql¨`

DCS-Typ

Es stehen etcd oder Consul zur Auswahl, entsprechend wird etcd ausgewählt.
Es könnten auch bestehende etcd-Nodes verwendet werden.

Load Balancing

Grundsätzlich funktionierte der Cluster auch ohne Load Balancer.
Allerdings muss in diesem Fall HAProxy als Load Balancer eingesetzt werden.

Virtual IP

Auch kann definiert werden ob eine Virtuelle IP verwendet wird.
Ohne Load Balancer würde die virtuelle IP dem Primary Patroni Node zugewiesen.

Connection Pooler

`PgBounder` kann aktiviert oder deaktiviert werden, entsprechend wurde dieser aktiviert.
Die Konfiguration wurde belassen.
Ein andererer Connection Pooler steht nicht zur verfügung.

PostgreSQL - User

Alle notwendigen User können bereits definiert werden.

PostgreSQL - Datenbanken

Selbiges gilt für die Datenbanken.
Folgende Datenbanken wurden eingetragen:

- **k8s_core_gitlab_prod**
  Leere DB Hülle für die GitLab Datenbank
- **k8s_core_harbor_prod**
  Leere DB Hülle für die Harbor Datenbank
- **k8s_core_keycloak_prod**
  Leere DB Hülle für die Keycloak Datenbank

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

PostgreSQL - Extensions

Alle Datenbanken benötigen die Extension `pgstattuple`.
Diese wird verwendet, um Bloated Tables und Indices zu ermitteln.

PostgreSQL - pg_hba.conf

Ist eine IP oder ein Netzwerk nicht erfasst, kann trotz korrekten Zugangsdaten nicht auf die DB zugegriffen werden.
Entsprechend müssen die User und Client Adressen erfasst werden.

PostgreSQL - .pgpass

Im Passwortfile `.pgpass` werden die Passwörter zu den Usern erfasst.
Zusätzlich kann hier der Zugriff ebenfalls auf bestimmte Adressen eingeschränkt werden.
Als Alternative würde `postgresql_pg_ident` zur Auswahl stehen.

Patroni - REST-API

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

3.2.3.1 Vorbedingungen

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

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

Tabelle 3.21: Testsystem - Benötigte Ports

Es reicht nicht, auf den Servern einen Proxy einzustellen.

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

```
proxy_env: {}

```

Listing 3.29: main.xyml - No Proxy

Der Gruppenproxy funktioniert mit folgenden Proxy-Settings:

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

```

Listing 3.30: main.xyml - Gruppenproxy

Der Proxy für alles andere benötigt folgenden Eintrag:

```
proxy_env:
  http_proxy: http://sproxy.sivc.first-it.ch:8080
  https_proxy: http://sproxy.sivc.first-it.ch:8080

```

Listing 3.31: main.xyml - SProxy

3.2.3.2 Installation / Deploy - deploy_pgcluster.yml

Die Installation ist simpel.
Nachdem alle Konfigurationen vollzogen sind, muss nur das entsprechende Playbook ausgeführt werden:

```
ansible-playbook deploy_pgcluster.yml

```

Listing 3.32: Deploy - deploy_pgcluster.yml

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

3.2.3.3 Maintenance - config_pgcluster.yml

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

```
ansible-playbook config_pgcluster.yml

```

Listing 3.33: Maintenance - config_pgcluster.yml

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

Zum Testen wurden nachträglich folgende Werte hinzugefügt:

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

3.2.3.4 HAProxy Node hinzufügen - add_balancer.yml

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

```
[balancers]
10.0.22.176
10.0.22.177
10.0.20.43 new_node=true

```

Listing 3.34: HAProxy Node erweitern - Inventory

Zum Schluss muss nur noch das Playbook ausgeführt werden:

```
ansible-playbook add_balancer.yml

```

Listing 3.35: HAProxy Node erweitern - add_balancer.yml

3.2.3.5 Patroni Node hinzufügen - add_pgnode.yml

Das vorgehen ist gleich wie beim erweitern von HAProxy.
Das `Inventory` wurde erweitert:

```
[master]
10.0.22.173 postgresql_exists=false

[replica]
10.0.22.174 postgresql_exists=false
10.0.22.175 postgresql_exists=false
10.0.28.16 postgresql_exists=false new_node=true

```

Listing 3.36: Patroni Node erweitern - Inventory

Das Deployment erfolgt nun ebenfalls via Playbook:

```
ansible-playbook add_pgnode.yml

```

Listing 3.37: Patroni Node erweitern - add_pgnode.yml

Die vollständige Dokumentation ist im Anhang - Installation Testsystem zu finden.

3.2.4 Technical Review der Umgebung

Die beiden Patroni-Nodes `sks1264` und `sks1265` sind Synchrone Replicas und der `sks1263` ist der Primary:

Abbildung 3.60: Technical Review - Patroni Nodes

Alle Datenbanken, die übergeben wurden, sind erstellt worden:

console [postgres@vks0041.ksgr.ch]

Tx: Auto

```
1  SELECT datname FROM pg_datab
```

Output | datname

7 rows

	datname
1	postgres
2	k8s_core_gitlab_prod
3	template1
4	template0
5	k8s_core_harbor_prod
6	k8s_core_keycloak_prod
7	gramic_test

Abbildung 3.61: Technical Review - Datenbanken

Die Extensions wurden ebenfalls installiert:

Tx: Auto | Playground

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

Output | Result 2

2 rows

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

Abbildung 3.62: Technical Review - PostgreSQL Extensions

Die Rollen und User wurden erstellt:

Abbildung 3.63: Technical Review - PostgreSQL User und Rollen

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

sks1266.ksgr.ch:7000 90%

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

Statistics Report for pid 57563

> General process information

pid = 57563 (process #1, nbproc = 1, nbthread = 4)
uptime = 0d 2h28m44s
system limits: memmax = unlimited; ulimit-n = 200063
maxsock = 200063; **maxconn** = 100000; **maxpipes** = 0
current conns = 2; current pipes = 0/0; conn rate = 0/sec; bit rate = 2.719 kbps
Running tasks: 0/34; idle = 100 %

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

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export
- JSON export (schema)

External resources:
- Primary site
- Updates (v2.8)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	2	3	100 000	3			1 157	134 395	0	0	1					OPEN								
Backend	0	0		0	0		0	0	10 000	0	0	0s	1 157	134 395	0	0		0	0	0	0	2h28m UP		0/0	0	0		0		

master

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	0	2	10 000	4			40 652	96 278	0	0	0					OPEN								
sks1263	0	0	-	0	1		0	1	-	4	4	2h15m	40 652	96 278		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	0	0	0s	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 3ms	1/1	Y	-	1	1	2h28m	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h28m	-
Backend	0	0		0	1		0	1	1 000	4	4	2h15m	40 652	96 278	0	0		0	0	0	0	2h28m UP		1/1	1	0		0	0s	

replicas

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h28m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	1	1	4s	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	1	1	4s	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h28m UP		2/2	2	0		1	4s	

replicas_sync

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h28m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	1	1	31s	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	1	1	31s	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h28m UP		2/2	2	0		1	31s	

replicas_async

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 3ms	1/1	Y	-	1	1	2h28m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 2ms	1/1	Y	-	4	2	2h28m	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 3ms	1/1	Y	-	4	2	2h28m	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h28m DOWN		0/0	0	0		2	2h28m	

Abbildung 3.64: Technical Review - HAproxy sks1266

sks1267.ksgr.ch:7000 90%

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

Statistics Report for pid 58804

> General process information

pid = 58804 (process #1, nbproc = 1, nbthread = 4)
uptime = 0d 2h29m20s
system limits: memmax = unlimited; ulimit-n = 200063
maxsock = 200063; **maxconn** = 100000; **maxpipes** = 0
current conns = 4; current pipes = 0/0; conn rate = 1/sec; bit rate = 0.000 kbps
Running tasks: 0/36; idle = 100 %

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

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export
- JSON export (schema)

External resources:
- Primary site
- Updates (v2.8)
- Online manual

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	2	-	2	2	100 000	3			423	44 656	0	0	1					OPEN								
Backend	0	0		0	0		0	0	10 000	0	0	0s	423	44 656	0	0		0	0	0	0	2h29m UP		0/0	0	0		0		

master

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	5	-	2	4	10 000	7			11 465	11 068	0	0	0					OPEN								
sks1263	0	0	-	0	5		2	3	-	7	7	11m59s	11 465	11 068		0		0	0	0	0	2h29m UP	L7OK/200 in 2ms	1/1	Y	-	0	0	0s	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h29m	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h29m	-
Backend	0	0		0	5		2	3	1 000	7	7	11m59s	11 465	11 068	0	0		0	0	0	0	2h29m UP		1/1	1	0		0	0s	

replicas

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h29m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m UP	L7OK/200 in 1ms	1/1	Y	-	1	1	4s	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m UP	L7OK/200 in 1ms	1/1	Y	-	1	1	4s	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h29m UP		2/2	2	0		1	4s	

replicas_sync

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m DOWN	L7STS/503 in 2ms	1/1	Y	-	1	1	2h29m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 2ms	1/1	Y	-	1	1	31s	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m UP	L7OK/200 in 1ms	1/1	Y	-	1	1	31s	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h28m UP		2/2	2	0		1	31s	

replicas_async

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server									
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	10 000	0			0	0	0	0	0					OPEN								
sks1263	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h29m DOWN	L7STS/503 in 3ms	1/1	Y	-	1	1	2h29m	-
sks1264	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 3ms	1/1	Y	-	4	2	2h28m	-
sks1265	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	2h28m DOWN	L7STS/503 in 1ms	1/1	Y	-	4	2	2h28m	-
Backend	0	0		0	0		0	0	1 000	0	0	?	0	0	0	0		0	0	0	0	2h28m DOWN		0/0	0	0		2	2h28m	

Abbildung 3.65: Technical Review - HAproxy sks1267

Die PgBouncer konfiguration passt.

Alle via Ansible hinterlegten Datenbanken wurden erfasst:

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

Abbildung 3.66: Technical Review - PgBouncer - Datenbanken

Beim Auslesen der Stats und Datenbanken muss der richtige User gewählt werden.
So hat der User `pgbouncer` keine Berechtigung, Lokal zuzugreifen.
Auch nicht auf die Lokale Datenbank `pgbouncer`.
Hierfür muss der User `postgres` verwendet werden:

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

Listing 3.38: PgBouncer - Connect

Die Abfrage der Tabellen wird dann mittels folgendem Command gemacht:

```
show databases;
```

Listing 3.39: PgBouncer - Databases

Die Logs werden einmal pro Tag neu erfasst:

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

Abbildung 3.67: Technical Review - PostgreSQL - Log Rotation

Die Einstellungen werden vorerst so belassen.
Sobald dass SIEM betriebsbereit ist, wird das Format und die Frequenz entsprechend den Anforderungen angepasst.

3.3 Testing

3.3.0.1 Testfälle

Basistests

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

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

Failover

4. Der Server des Primary-Node wird manuell rebooted.

Switchover

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

Restore

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

Ansible - Deploy

10. Mit Ansible kann der Patroni Cluster deployed werden.

Ansible - Maintenance

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

Ansible - Patroni Node Extend

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

Ansible - HAproxy Node Extend

13. Mit hilfe eines Ansible Playbooks kann ein HAproxy Node angehängt werden.

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

3.3.1 Protokollierung

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

Tabelle 3.22: Testresultate Testsystem

Der Patroni Node wurde erfolgreich angefügt:

```
root@sks1265:~# patronictl -c /etc/patroni/patroni.yml list
+ Cluster: k8s-core-psql (7366694452879094871) ----+----+-----------+
| Member  | Host        | Role         | State     | TL | Lag in MB |
+---------+-------------+--------------+-----------+----+-----------+
| sks1263 | 10.0.22.173 | Leader       | running   |  1 |           |
| sks1264 | 10.0.22.174 | Sync Standby | streaming |  1 |         0 |
| sks1265 | 10.0.22.175 | Sync Standby | streaming |  1 |         0 |
| sks9016 | 10.0.28.16  | Replica      | streaming |  1 |         0 |
+---------+-------------+--------------+-----------+----+-----------+
```

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

3.3.2 Review und Auswertung

3.3.2.1 vitabaks/postgresql_cluster

vitabaks/postgresql_cluster ist sehr schnell und zuverlässig deployt.
Das Ansible-Repository hat allerdings eine schwachstelle.
Der Connection Pooler ist auf dem Level der Patroni Nodes, nicht auf dem Proxy-Level.
Dadurch verliert ein Client die Verbindung, wenn der entsprechende Server nicht mehr erreichbar ist.

Auch der HAProxy hat eine Schwäche.
Es handelt sich um einen Layer 7 Proxy, dadurch ist HAProxy nicht in der Lage, SQL Statements zu analysieren.
Wenn also die Read-Only Statements auf die Replikas geleitet werden sollen, muss die Applikation diese auf den entsprechenden Port zugreifen.
Ist die Applikation selber nicht in der Lage, die Lese- und Schreibprozesse an zwei verschiedene Ports zu leiten, geht der gesamte Load auf den Primary Node.

3.3.2.2 Maintenance-Tool - Bloating Tables

Aufgeblähte Tabellen werden zuverlässig erkannt und die Tabelle vakuumiert.
Zudem werden die Indices neu aufgebaut und somit geordnet.

3.3.2.3 Maintenance-Tool - AUTOVACUUM

Der Parameter `autovacuum_vacuum_scale_factor` wird zuverlässig berechnet und abgelegt.

3.4 Maintenance-Tool

3.4.0.1 Maintenance-Tool - Generell

Da das Kubernetes Testsystem des KSGR noch nicht freigegeben ist,
wurden die Maintenance-Jobs auf der Evaluationsplattform betrieben.
Dadurch ergaben sich ein paar einschränkungen.

Ressourcen

Ressourcen wie Python-Skripte sollten in einer Produktiven Umgebung z.B. in einem Helm Chart abgebildet werden.
Da die erstellung eines helm Charts den Rahmen der Diplomarbeit sprengen würde, wurden Python-Skripte als `ConfigMap` deployt.
In der Produktiven Umgebung ist dies tunlichst zu vermeiden!

HashiCorp Vault / Sectrets

Die Zugangsdaten (Username und Passwort) werden mit grosser Wahrscheinlichkeit in HashiCorp Vault[33] abgebildet.
In diesem Fall wird ein Secret deployt und das Secret als Filesystem gemountet.
Die grösse Schwäche hierbei ist, dass die Secrets unverschlüsselt auf dem Filesystem des Pods liegen, selbst wenn mit Base64 ein Hash erzeugt wurde.
Daher ist auch dieses Verfahren nicht für die Produktion geeignet.

Mailing

Mailing steht auf der Evaluationsumgebung nicht zur verfügung.

Die Mainenance-Jobs sollen nach dem Microservice-Gedanken aufgebaut werden.
So soll die Automatisierbarkeit erhöht und flexibilisiert werden.

3.4.0.2 Maintenance-Tool - Bloated Tables / Indices

3.4.0.2.1 Ziel und Zweck

PostgreSQL hat mit AUTOVACUUM eine Funktionalität, welche Tabellen und Indizes aufräumt und bereinigt.
Allerdings gelten die Parameter für den ganzen PostgreSQL Cluster, einzelne Tabellen werden zwangsläufig nicht optimal vakuumiert.

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

3.4.0.2.2 Funktionsweise

Der Maintenance-Job soll anhand der Auf allen Datenbanken muss die Extension `pgstattuple` installiert werden.
Die Extension sammelt und stellt Informationen über Tabellen, Indizes, Tupels usw., zur verfügung.

Mit Python werden alle Tabellen, welche bei `pgstattuple` mehr als einen konfigurationen Wert, aktuell 1.5%, gelistet.
Das SQL Statement hierzu, ist wie folgt aufgebaut:

```
select
    relname as table,
    (pgstattuple(oid)).dead_tuple_percent
from pg_class
where
```

```
    relkind = 'r' and
    (pgstattuple(oid)).dead_tuple_percent > %s
order by dead_tuple_percent desc;

```

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

Anschliessend werden die entsprechenden Tabellen mittels VACUUM-Befehl manuell vakuumiert:

```
vacuum <table>;

```

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

> ☞ Ein `VACUUM` kann nicht in einer normalen Transaktion abgesetzt werden!
> Bei `psycopg2` muss der Transaktions-Isolations Level auf `ISOLATION_LEVEL_AUTOCOMMIT` gesetzt werden.
> Anders lässt sich das SQL nicht absetzen.

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

```
reindex table <table>;

```

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

Wichtig ist eine vorangehende Prüfung, ob ein AUTOVACUUM-Job am laufen ist.
Trifft dies zu, darf das Skript in diesem Lauf nicht ausgeführt werden.
Überprüft kann dies mit folgendem SQL werden:

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

```

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

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

Die vollständige Dokumentation ist im Anhang - Maintenance-Tool - Bloated Tables zu finden.

3.4.0.3 Maintenance-Tool - AUTOVACUUM

3.4.0.3.1 Ziel und Zweck

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

autovacuum_vacuum_threshold

Mindestanzahl geänderter oder gelöschter Tupels die es in einer Tabelle braucht,
damit AUTOVACUUM startet.
Der Default liegt bei 50 toten Tuples.

autovacuum_vacuum_scale_factor

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

Die Grundformel, wann vakuumiert wird, sieht folgendermassen aus:
$n_dead_tup > (pg_class.reltuples \times autovacuum_vacuum_scale_factor) + autovacuum_vacuum_threshold$

Der autovacuum_vacuum_scale_factor über den gesamten PostgreSQL Cluster muss nun also mit der Zeit nachjustiert werden.
Dazu wurde obige Formel entsprechend umgestellt, da autovacuum_vacuum_threshold grundsätzlich statisch bleibt:

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

Die Aufgabe von diesem Maintenance-Skripts ist es also nun, einmal pro Tag den autovacuum_vacuum_scale_factor neu zu berechnen.
Da keine Mails versendet werden können, soll bei einem sich geänderten Parameter die generierte HTML-Tabelle auf das Verzeichnis des Kubernetes Nodes gespeichert werden.
Im Produktiven Betrieb soll die Tabelle als HTML in die Mail eingefügt werden.

3.4.0.3.2 Funktionsweise

In einem ersten Schritt werden alle benötigten Parameter aus der PostgreSQL-Konfiguration oder den entsprechenden Katalogtabellen ausgelsen.
Dies lässt sich mit einem SQL bewerkstelligen, welches wie folgt aufgebaut ist:

```sql
select
    (
        select
            setting as autovacuum_vacuum_scale_factor
        from pg_settings
        where
            name = 'autovacuum_vacuum_scale_factor'
    ) as autovacuum_vacuum_scale_factor,
    (
        select
            setting as autovacuum_vacuum_threshold
        from pg_settings
        where
            name = 'autovacuum_vacuum_threshold'
    ) as autovacuum_vacuum_threshold,
    (
        select
            sum(reltuples) as reltuples
        from pg_class
    ) as reltuples,
    (
        select
            sender_host
        from pg_stat_wal_receiver
```

```
25     ) as sender_host
26
```

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

Anschliessend wird ein Python Dictionary erstellt, welches wiederum in ein pandas DataFrame umgewandelt wird.
Mit der Methode `to_html()` wird die Tabelle dann simpel auf das gemountete Host Filesystem geschrieben.
Die Tabelle sieht wie folgt aus:

	Bestehend	Berechnet
autovacuum_vacuum_scale_factor	0.01	0.02

Abbildung 3.69: AUTOVACUUM - Berechneter autovacuum_vacuum_scale_factor

Die vollständige Dokumentation ist im Anhang - Maintenance-Tool - Maintenance-Tool - AUTOVACUUM zu finden.

3.5 Monitoring

Das Monitoring im PRTG besteht aus zwei Teilen.
Zum einen werden die Standard-Werte wie Ping, CPU Load, Load Average, Disk Free, Traffic und die Uptime:

Abbildung 3.70: PRTG - Patroni Monitoring

Zusätzlich wurde ein Python Custom Sensor erstellt.
Dieser nutzt die Patroni REST-API um folgende Werte abzufragen:

Cluster Status

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

Cluster Unlock

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

Failsafe Mode Activity

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

Replication Lag

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

Replication State

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

Die Konfiguration wird im PRTG gemacht.
Dabei müssen im Feld «`Additional Parameters`»folgende Parameter übergeben werden:

replication_mode

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

lag_warning

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

lag_error

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

Die Werte werden Kommasepariert übergeben:
`replication_mode=sync,lag_warning=2GB,lag_error=4GB`

Abbildung 3.71: PRTG - Patroni Cluster Sensor Setting

Der Sensor sieht im PRTG dann so aus:

Abbildung 3.72: PRTG - Patroni Cluster Sensor

Die vollständige Dokumentation ist im Anhang - Monitoring - PRTG zu finden.

3.6 Troubleshooting und Lösungsfindung

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

Evaluation - Yugaware

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

Evaluation - etcd für Patroni

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

Evaluation - MetalLB

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

Evaluation - local-path-provisioner

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

Evaluation - StackGres Proxy für Extension

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

Evaluation - Benchmarking automatisation

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

Testsystem - Proxy

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

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

3.7 Ausblick

4 Resultate

4.1 Zielüberprüfung

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

Tabelle 4.1: Result - Zielüberprüfung

4.1.1 SQL optimierungen - Indizes tracken und verwalten

Um Indizes effektiv setzen zu können, muss der Ausführungplan, der Explaining Plan, eines SQL Statements analysiert werden.
EXPLAIN kann allerdings nur ausgeführt werden, wenn ein Statement angehängt wird[5].

Bevor also ein Explaining Plane analysiert werden kann, muss dieser erst generiert werden.
Dazu müssen die Statements gespeichert werden und abschliessend das Query ausgeführt werden.

EXPLAIN bietet weit mehr Informationen.
Wie der EXPLAIN für das automatische setzen von Indizes benutzt werden kann,
benötigt viel analyse und ein Konzept.

Es lassen sich zwar nicht benutzte Indizes auslesen und auch dropen.
Aus Erfahrung mit Oracle Database zeigte sich aber, dass das löschen eines Index einen gesamten Ausführungsplan umwerfen kann.
Ohne einer Analyse mit EXPLAIN, ist das Risiko für eine solche Aktion zu gross.

Alles in allem bräuchte es für diesesn Ziel ein Performance Warehouse.
Also eine Datenbank, die EXPLAIN Resultate, Systemparameter und weitere Metriken speichert.
Nur so lassen sich SQL Statements sauber analysieren und optimieren.
Dieses Unterfangen wäre ein eigenes Thema für eine Diplomarbeit.

4.1.2 Testziele

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

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

4.2 Schlussfolgerung

4.2.1 Kubernetes / Rke2

Kubernetes mit Rke2 lässt sich vergleichsweise schnell installieren.
Damit die Containerplattform voll Funktionsfähig sein soll, braucht es noch einige Module.
Mit MetalLB kann ein Load Balancer, den die Applikationen nutzen können, deployt werden.
Wenn nicht, wie zukünftig beim KSGR ein Storagesystem wie Rook / CephFS ein Storage System aufgebaut werden soll,
kann mittels Local-path-provisioner das Host-Storagesystem den Containern präsentiert werden.

Für Private Zwecke oder als Evaluationssystem kann so rasch ein System aufgebaut werden.
Für Test- oder Produktivsysteme ist dieses Setting allerdings eher weniger geeignet.

4.2.2 YugabyteDB

YugabyteDB bietet auch in der Open-Source Variante einiges.
Besonders die effiziente Speicherung, selbst grosser Datenmengen, hat in einer Zukunft,
wo immer mehr Applikationen in die Cloud verschoben werden und Datenbank Storage sehr teuer wird,
einen Trumpf im ärmel.

YugabyteDB stellt als Distributed SQL System hohe Anforderungen an ein Synchrones Zeitsystem.
Auch ist YugabyteDB keine aktuelle Umsetzung von PostgreSQL, für Applikationen die auf eine aktuelle PostgreSQL Version angewiesen ist, ist YugabyteDB daher nicht geeignet.

4.2.3 StackGres - Citus

Ongres hat mit StackGres den Patroni Stack in eine CloudNative Umgebung gegossen.
Das pro Datenbank ein eigenener Cluster deployt wird, sorgt für ein maximum an Isolation.
Durch diesen Architekturentscheid steigt aber auch der ohnehin grosse Ressourcenverbrauch des gesamten Systems nochmals stark an.

Problematisch ist auch, dass einige Komponenten in einer Blackbox liegen.
Als während dem Benchmarking bei der Evaluation das Sharding immer wieder an der gleichen Stelle scheiterte,
war zwar ersichtlich, dass es zu einem Java Fehler gekommen war, doch genauere Informationen liessen sich den Logs nicht entnehmen.

StackGres bietet eine direkte Integration von Citus, im Benchmarking stiess diese Infrastruktur an ihre Grenzen.
Das Sharding-System Citus wiederum hat seine Stärken klar bei lesenden Zugriffen.
Das alte Citus Sharding, bei dem Tabellen Distributed wurden, kann nur mit Eingriffen in die Statements der Applikation effizient genutzt werden.
Das Schema Based Sharding bietet hier einen Vorteil doch beide Sharding-Systeme können Tabellen nicht uneingeschränkt Sharden.
Wurden zwischen Tabellen Foreign-Key Constraints gesetzt, können diese entweder nicht mehr Distributed werden oder aber beide müssen im gleichen Schema liegen.
Es können zwar Referenced Tables, Tabellen die auf allen Shards liegen, erzeugt werden, doch diese Methode ist ineffizient beim Schreibprozess.
Zudem rät Citus Data selbst, Referenced Tables nur für Referenzierte Tabellen, die für alle Distributionen und Schemas bereitstehen sollen, zu verwenden.
Und dann auch nur, wenn diese Tabellen nicht zu gross werden.

4.2.4 Patroni

Patroni mit HAProxy bietet die möglichkeit, einen stabilen Monolithischen PostgreSQL HA Cluster aufzubauen.
Basis bleibt dabei ein PostgreSQL Cluster, der von Patroni orchestriert und verwaltet wird.
Patroni benötigt allerdings Zusatztools wie einen Connection Pooler oder einen Proxy sowie z.B. etcd als DCS.

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

4.2.5 vitabacks / postgresql_cluster

Das Ansible Repository bietet eine ganze Reihe von Playbooks, mit dem ein Patroni Cluster deployt, erweitert und verwaltet werden kann.
Die Dokumentation des GitHub Repositories ist qualitativ hochwertig, ein Cluster ist schnell deployt.
Weitere Schritte, wie das Härten des gesamten Systems ist ebenfalls bereits angedacht und vorbereitet.

Der grosse Schwachpunkt ist zurzeit die Architektur.
Der Connection Pooler wird auf dem Patroni Node installiert, geht die Verbindung zu diesem verloren, ist die Applikation Disconnected.
So wird der Connection Pooler seiner Funktion beraubt.
HAProxy ist zudem ein Layer 7 Proxy, der Cluster ist darauf angewiesen, dass die Applikation Lese- und Schreibanfragen trennen kann und an zwei verschiedene Ports sendet.
Im Moment laufen aber bestrebungen, mit der Software PgCat sowohl den Proxy als auch den Connection Pooler zu vereinen.
PgCat vereint beides in einer Open-Source-Software und ist in der Lage, die Statements zu analysieren und lesenden Verkehr auf die Replikas zu verteilen.

4.3 Weiteres Vorgehen / offene Arbeiten

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

SIEM

Das SIEM, in Form von Elastic, wurde installiert.
Daher wird einer der nächsten Schritte sein, dass Logging des Testsystems an das SIEM anzubinden.

Backup

Das Testsystem muss in die Backup-Infrastruktur eingebunden werden.

PKI

Die Ausschreibungsphase für den PKI wurde zwar abgeschlossen, doch das System ist noch nicht betriebsbereit.
Sobald das System steht ist und ein Konzept vorliegt, muss das Testsystem integriert werden.

Hardening

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

KSGR Ansible

Wenn das Hardening und Monitoring umgesetzt sind, muss das vitabacks / postgresql_cluster Repository in den KSGR Red Hat Ansible Automation Platform eingebunden werden.
Dazu muss ggf. ein Fork des GitHub Repositories vorgenommen werden.

Maintenance-Tool

Das Kubernetes Test- und Produktivsystem des KSGR wurde noch nicht freigegeben.
Sobald dies der Fall ist, muss das Maintenance-Tool sauber implementiert werden.
Secrets müssen mit HashiCorp Vault[33] angebunden, das Logging dem KSGR Standard angeglichen werden.

Produktivsystem

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

YugabyteDB

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

Performance Warehouse für PostgreSQL

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

4.4 Persönliches Fazit

Aus heutiger Sicht würde ich die Ziele weiter reduzieren.
Die Evaluation alleine hat wesentlich mehr Zeit in Anspruch genommen, als geplant.
Heute würde ich auf die Umsetzung eines Testsystems verzichten und stattdessen ein oder zwei Systeme mehr evaluieren und installieren.
Generell habe ich wohl unterschätzt, wie Tief der Dive in die Konzepte und Architekturen gehen wird.

Den gesamten Zeitplan aber auch die Dokumentenstruktur würde ich heute anders aufbauen.
Die Evaluation hätte wesentlich mehr Raum und auch den Maintenance-Tools stünde eigene Arbeitspakete zur Verfügung.
Das Projektmanagement hat keine eigenen Arbeitspakete erhalten und musste daher immer an anderer Stelle abgebucht werden.
Ein weiterer Punkt den ich anders machen würde, gerade mit einer etwas ausgedehnteren Evaluationsphase wäre mehr Zeit für das Projektmanagement vorhanden gewesen.

Die Diplomarbeit zwang mich aus meiner Komfortzone herauszutreten.
Ich habe ungleich mehr DevOps und System Engineering Arbeiten als DBA gemacht, von denen gab es schlussendlich nur eine Handvoll.

Erkenntnisse und Lehren habe ich auf zwei Perspektiven gewonnen.
Einmal aus der Technischen Sicht.
So konnte ich mit rke2 von Grund auf ein Kubernetes System aufbauen und mit local-path-provisioner sowie MetalLB Sinnvoll erweitern.
Zudem durfte ich tiefe Einblicke in neue und zukunftsfähige Konzepte wie den Distributed SQL Systemen gewinnen.
Als DBA gehörte ich eher zu einer Konservativen IT-Sparte, die nach wie vor stark auf Monolithischer Systeme setzt und neuen Ansätzen eher Misstrauisch gegenübersteht.

Durch die Diplomarbeit habe ich meinen Horizont um verteilte Systeme und dem CloudNative Ansatz erweitert.
Kommt hinzu, dass es für die Evaluation sehr reizvoll war, ein falsch konfiguriertes System einfach zu löschen und angepasst neu zu deployen.
Bei Patroni, dass ein monolithisches System war, brauchte es immer einen Snapshot Restore.

Auf der anderen Seite habe ich auch aus der Sicht der Herangehensweise eine wichtige Lektion gelernt.
Beim Aufbau von rke2 sowie dem Umgang mit local-path-provisioner und MetalLB hatte ich mit diversen Problemen zu kämpfen,
doch gelang es mir, durch intensive Recherche jeweils Lösungen für diese Probleme zu finden.
Beim Aufbau des Patroni Evaluationssystems brachte dieser Zeitraubende Ansatz keinen Erfolg mehr.
Daher war ich gezwungen, einen Schritt zurückzutreten und das System auf das absolut notwendige Minimum zu vereinfachen.
Mit nur noch einem etcd Node, der auf einem anderen Server installiert war, gelang der Aufbau des Evaluationssystems.
Diesen Ansatz nahm ich mit Erfolg mit, als ich bei den Maintenance-Tools wieder auf herausforderungen traf.

Auf einen Satz reduziert, dürfte daher die wichtigste Erkenntnis aus der Diplomarbeit folgende sein:
«Im Zweifelsfall geht man besser einen Schritt zurück und nimmt soviel Komplexität aus einem System wie möglich!»

Abbildungsverzeichnis

1.1 Spitalregionen Kanton Graubünden[67] . . . 1
1.2 Wahlkreise Kanton St. Gallen[95] . . . 2
1.3 Spitalregionen / Spitalstrategie Kanton St. Gallen[61] . . . 3
1.4 Organigramm Kantonsspital Graubünden . . . 4
1.5 Organigramm Departement 10 - ICT . . . 5
1.6 Datenbanken - Aufgeschlüsselt nach RDBMS - Datenbanken . . . 9
1.7 Datenbanken - Aufgeschlüsselt nach RDBMS - Instanzen . . . 10
1.8 Risikomanagement PostgreSQL . . . 16
1.9 Systemabgrenzung . . . 21
1.10 Risikomanagement Projekt . . . 26
1.11 Riskikomatrix - Assessment 21.03.2024 . . . 28
1.12 Riskikomatrix - Assessment 29.04.2024 . . . 30
1.13 Riskikomatrix - Assessment 13.05.2024 . . . 32

3.1 Sharding - Vertikale Partitionierung . . . 39
3.2 Sharding - Horizontales Partitionierung . . . 40
3.3 Monolithische vs. verteilte SQL Systeme . . . 42
3.4 CAP-Theorem . . . 44
3.5 Datenbankskalierung . . . 45
3.6 Präferenzmatrix . . . 49
3.7 Testing - ERD DB self_healing_test . . . 52
3.8 Benchmark Settings - Zabbix - Systeminformationen . . . 53
3.9 Benchmark Settings - Zabbix - Connections per Seconds . . . 54
3.10 Benchmark Settings - Zabbix - Queries per Seconds . . . 54
3.11 Benchmark Settings - Zabbix - Client Queries per Seconds . . . 54
3.12 Benchmark Settings - Zabbix - DB Size . . . 55
3.13 Benchmark Settings - Anzahl Records / Skalierungsfaktor . . . 57
3.14 pg_auto_failover-Architektur - Single Standby . . . 61
3.15 pg_auto_failover-Architektur - Multi-Node Standby . . . 61
3.16 pg_auto_failover-Architektur - Citus . . . 62
3.17 CloudNativePG - Kubernetes - PostgreSQL . . . 63
3.18 CloudNativePG - Kubernetes - Read-write workloads . . . 64
3.19 CloudNativePG - Kubernetes - Read-only workloads . . . 64
3.20 Patroni-Architektur . . . 67
3.21 Stackgres - Grundarchitektur . . . 70
3.22 Citus - Coordinator und Workers . . . 71
3.23 Citus - Row-Based-Sharding . . . 71
3.24 Citus - Schema-Based-Sharding . . . 72
3.25 StackGres-Citus - Shard-Replikation . . . 73
3.26 yugabyteDB - Grundkonzept . . . 75

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

3.71 PRTG - Patroni Cluster Sensor Setting . . . 140
3.72 PRTG - Patroni Cluster Sensor . . . 140

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

XLII StackGres Testing - Patroni Übersicht cxxi
XLIII StackGres Testing - DB Zugriff cxxii
XLIV StackGres Testing - Connection Timeout cxxiii
XLV YugabyteDB - Too big clock skew is detected cxxiv
XLVI YugabyteDB - Tablet Leader - No Lease cxxv
XLVII YugabyteDB - CrashLoopBackOff cxxv
XLVIII YugabyteDB - Too big clock skew is detected - tmaster cxxvi
XLIX YugabyteDB - Too big clock skew is detected - tserver cxxvii
L local-path-provisioner[36] cxcvii

Tabellenverzeichnis

1.1 Inventarisierte Datenbanksysteme . . . 7
1.2 Datenbankinventar . . . 8
1.3 Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt . . . 11
1.4 Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken . . . 13
1.5 Administrative Aufgaben . . . 17
1.6 Automatisierung Administrativer Aufgaben . . . 18
1.7 Ziele . . . 19
1.8 Gegebene Systeme . . . 20
1.9 Abhängigkeiten . . . 23
1.10 Risiko-Matrix der Diplomarbeit . . . 25
1.11 Neu Erkannte / Erfasste Risiken . . . 27
1.12 Risiko-Assessment 21.03.2024 . . . 27
1.13 Risiko-Assessment 29.04.2024 . . . 29
1.14 Risiko-Assessment 13.05.2024 . . . 31

2.1 Projektcontrolling . . . 33
2.2 Fachgespräche . . . 38

3.1 Quorum Beispiele . . . 43
3.2 Anforderungskatalog . . . 48
3.3 Stakeholder . . . 49
3.4 Benchmark Settings - Mixed Transaktionen . . . 55
3.5 Benchmark Settings - DQL Transaktionen . . . 55
3.6 Benchmark Settings - Skalierungsfaktoren . . . 56
3.7 Benchmark Settings - Datenbankgrössen / Skalierungsfaktor . . . 56
3.8 Benchmark Settings - Tabellengrössen / Skalierungsfaktor . . . 56
3.9 Vorauswahl - Ausgeschieden . . . 80
3.10 Vorauswahl - Evaluation . . . 80
3.11 Evaluationssyssteme . . . 81
3.12 Evaluationssysstem - Distributed SQL / Sharding . . . 81
3.13 Testresultate Evaluation Patroni . . . 99
3.14 Testresultate Evaluation StackGres - Citus . . . 100
3.15 Testresultate Evaluation YugabyteDB . . . 100
3.16 Kostenberechnung - Annahmen . . . 109
3.17 Gemessene und Extrapolierte Aufwände Bsp. . . . 110
3.18 Testsystem - Komponenten . . . 119
3.19 Testsystem - Inventar . . . 121
3.20 Testsystem - Auxiliar Inventar . . . 121
3.21 Testsystem - Benötigte Ports . . . 123
3.22 Testresultate Testsystem . . . 133

4.1 Result - Zielüberprüfung 143

I Arbeitsrapport iii
II Kommentare - Anmerkung xxv

Listings

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

3.40 Maintenance-Tool - List - Maintenance-Tool - Bloated Tables / Indices 135
3.41 Maintenance-Tool - VACUUM - Maintenance-Tool - Bloated Tables / Indices 136
3.42 Maintenance-Tool - REINDEX - Maintenance-Tool - Bloated Tables / Indices 136
3.43 Maintenance-Tool - Check AUTOVACUUM - Maintenance-Tool - Bloated Tables / Indices . . 136
3.44 Maintenance-Tool - Parameter - Maintenance-Tool - AUTOVACUUM 137

1 Proxy Settings xlii
2 rke2 server - Verzeichnis erstellen xliii
3 rke2 server - config.yaml xliii
4 rke2 server - cilium-config.yaml xliii
5 rke2 server installieren xliii
6 rke2 agenten installieren xliv
7 rke2 agent - config.yaml xliv
8 -rke2 agent service restart xliv
9 rke2 server proxy xliv
10 rke2 server proxy kopieren xliv
11 rke2 server cilium installieren xliv
12 rke2 server cilium aktivieren xliv
13 rke2 server starten xliv
14 iptables entries server xlv
15 rke2 server token xlv
16 local-path-storage auf Linux Bereitstellen xlv
17 local-path-provisioner definieren xlvi
18 local-path-storage aktualisieren xlvi
19 MetallB installieren xlvi
20 MetallB konfigurieren xlvii
21 MetallB Konfiguration einspielen xlvii
22 rke2 - 250GiB Disk mount xlvii
23 local-path-storage 250GiB auf Linux Bereitstellen xlvii
24 local-path-provisioner Grosse Volumes xlviii
25 local-path-storage 250GiB aktualisieren xlix
26 yugabyteDB - StorageClass setzen xlix
27 yugabyteDB - StorageClass / PersistentVolume aktivieren xlix
28 yugabyteDB - Namespace l
29 yugabyteDB - Helm Chart Manifest l
30 yugabyteDB - Installation lx
31 yugabyteDB - Deinstallieren lx
32 yugabyteDB - StorageClass setzen lx
33 yugabyteDB - StorageClass / PersistentVolume Grosse Volumes aktivieren lxi
34 yugabyteDB - Namespace 250GiB lxi
35 yugabyteDB - Helm Chart Manifest 250GiB lxi
36 yugabyteDB - Cloud - Region - Zone lxix
37 yugabyteDB - Benchmarking - DB erstellen lxix
38 yugabyteDB - Benchmarking - Table Size lxx

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

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

127 Patroni - Benchmarking - Table Size SQL . . . cxviii
128 StackGres-Citus - Benchmarking-Commands . . . cxviii
129 StackGres-Citus - Benchmarking - Table Size SQL . . . cxix
130 Ansible - Installation . . . cxxvii
131 Ansible - Repository Clone . . . cxxviii
132 Ansible - cd Repository . . . cxxviii
133 Testsystem - Firewall Settings . . . cxxviii
134 Testsystem - Proxy Settings . . . cxxix
135 Testsystem - apt-Proxy Settings . . . cxxix
136 Testsystem - Deployment - inventory . . . cxxix
137 Testsystem - Deployment - main.yml . . . cxxx
138 Deploy - Anhang - Ansible Ping . . . cxli
139 Deploy - Anhang - deploy_pgcluster.yml . . . cxli
140 Deploy - Anhang - Deployt . . . cxlii
141 Testsystem - Anhang - Maintenance - main.yml . . . clxv
142 Testsystem - Anhang - Maintenance - config_pgcluster.yml . . . clxxvi
143 Testsystem - sks9016 - apt-Proxy Settings . . . clxxvi
144 Testsystem - sks9016 - Proxy Settings . . . clxxvi
145 Testsystem - Anhang - add_balancer.yml - inventory . . . clxxvi
146 Testsystem - Anhang - add_balancer.yml . . . clxxvii
147 Testsystem - Anhang - add_pgnode.yml - inventory . . . clxxvii
148 Testsystem - Anhang - add_pgnode.yml . . . clxxviii
149 Maintenance-Tool - Bloated Tables / Indices - ksgr_postgresql_maintenance_bloated_tables.pyclxxviii
150 Maintenance-Tool - Bloated Tables / Indices - Namespace . . . clxxxi
151 Maintenance-Tool - Bloated Tables / Indices - Python > ConfigMap . . . clxxxii
152 Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy . . . clxxxii
153 Maintenance-Tool - Bloated Tables / Indices - Base64 . . . clxxxii
154 Maintenance-Tool - Bloated Tables / Indices - Secret . . . clxxxii
155 Maintenance-Tool - Bloated Tables / Indices - Secret Deploy . . . clxxxii
156 Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test . . . clxxxii
157 Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test Deploy . . . clxxxiii
158 Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-testclxxxiii
159 Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-test Deploy . . . clxxxiv
160 Maintenance-Tool - AUTOVACUUM - ksgr_postgresql_maintenance_autovacuum_calculation.pyclxxxiv
161 Maintenance-Tool - AUTOVACUUM - Python > ConfigMap . . . clxxxviii
162 Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy . . . clxxxviii
163 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres . . . clxxxviii
164 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres Deploy . . . clxxxviii
165 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum . . . clxxxviii
166 Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum Deploy . . . clxxxix
167 Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation . . . clxxxix
168 Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation Deploy . . . cxc
169 Monitoring - KSGR - Patroni - Healthcheck.py . . . cxci

170 Python LaTex - zotero.py - Zotero BibLaTex Importer cxcviii
171 Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer cciii
172 Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer cciii
173 Python LaTex - riskmatrix.py - Risxikomatrizen ccix
174 Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen ccxii
175 Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples ccxvi
176 Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm ccxvii
177 Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm ccxix
178 Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle ccxix
179 Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle ccxxv
180 Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm ccxliii
181 Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme ccxlviii

Literatur

[1] *7210-vrrp.pdf*. https://www.cisco.com/c/de_de/support/docs/security/vpn-3000-series-concentrato 7210-vrrp.pdf.

[2] *About pgbench-tools*. https://github.com/gregs1104/pgbench-tools. original-date: 2010-02-17T13:33:28 2023.

[3] Satyadeep Ashwathnarayana und Inc. Netdata. *How to monitor and fix Database bloats in PostgreSQL? | Netdata Blog*. https://blog.netdata.cloud/postgresql-database-bloat/. 2022.

[4] unknown author. *#1 Backup-Lösung für Kubernetes*. https://www.veeam.com/de/kubernetes-native-back html?ck=1697900263871.

[5] unknown author. *14.1. Using EXPLAIN*. https://www.postgresql.org/docs/16/using-explain.html. 2024.

[6] unknown author. *API Reference - CloudNativePG*. https://cloudnative-pg.io/documentation/1.22/cloudnative-pg.v1/.

[7] unknown author. *API reference (for YSQL and YCQL)*. https://docs.yugabyte.com/preview/api/.

[8] unknown author. *Architecture Basics — pg_auto_failover 2.0 documentation*. https://pg-auto-failover.readthedocs.io/en/main/architecture.html.

[9] unknown author. *Benchmark Setup with Citus and pgbench - Citus 12.1 documentation*. https://docs.citusdata.com/en/stable/extra/write_throughput_benchmark.html.

[10] unknown author. *Benefits of using YugabyteDB*. https://docs.yugabyte.com/preview/features/. Section: preview.

[11] unknown author. *Choosing Distribution Column - Citus 12.1 documentation*. https://docs.citusdata.com/en/v12.1/sharding/data_modeling.html#distributed-data-modeling.

[12] unknown author. *Cilium - Cloud Native, eBPF-based Networking, Observability, and Security*. https://cilium.io.

[13] unknown author. *Citus Replication Model: Today and Tomorrow - Replication Groups*. https://www.citusdata.com/blog/2016/12/15/citus-replication-model-today-and-tomorrow/.

[14] unknown author. *Citus Support — pg_auto_failover 2.0 documentation*. https://pg-auto-failover.readthedocs.io/en/main/citus.html.

[15] unknown author. *CLIs and command line tools*. https://docs.yugabyte.com/preview/admin/.

[16] unknown author. *CloudNativePG - Main Features*. https://cloudnative-pg.io/documentation/1.22/#main-features.

[17] unknown author. *Cluster Management - Citus 12.1 documentation - worder-node-failure*. https://docs.citusdata.com/en/v12.1/admin_guide/cluster_management.html#worker-node-failures.

[18] unknown author. *Cluster Management — Citus Docs 7.2 documentation*. https://docs.citusdata.com/en/v7.2/admin_guide/cluster_management.html.

[19] unknown author. *Concepts - Citus 12.1 documentation - row-based-sharding*. https://docs.citusdata.com/en/v12.1/get_started/concepts.html#row-based-sharding.

[20] unknown author. *Consensus Protocol | Raft | Consul | HashiCorp Developer*. https://developer.hashicorp.com/consul/docs/architecture/consensus.

[21] unknown author. *Consul Architecture | Consul | HashiCorp Developer*. https://developer.hashicorp.com/consul/docs/architecture.

[22] unknown author. *Creating and Modifying Distributed Objects (DDL) - Citus 12.1 documentation*. https://docs.citusdata.com/en/stable/develop/reference_ddl.html?highlight=create_reference_table#reference-tables.

[23] unknown author. *DCS Failsafe Mode — Patroni 3.3.0 documentation*. https://patroni.readthedocs.io/en/latest/dcs_failsafe_mode.html#dcs-failsafe-mode.

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

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

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

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

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

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

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

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

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

[33] unknown author. *Introduction | Vault | HashiCorp Developer*. https://developer.hashicorp.com/vault/docs/what-is-vault.

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

[35] unknown author. *Keycloak*. https://www.keycloak.org/.

[36] unknown author. *Local Disk Storage for Kubernetes | ITNEXT*. https://itnext.io/local-disk-storage-fo

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

[38] 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.

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

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

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

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

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

[44] unknown author. *Red Hat Ansible Automation Platform automation controller*. https://www.redhat.com/en/technologies/management/ansible/automation-controller.

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

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

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

[48] unknown author. *VRRP verstehen | Junos OS | Juniper Networks*. https://www.juniper.net/documentation/de/de/software/junos/high-availability/topics/topic-map/vrrp-understanding.html.

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

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

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

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

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

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

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

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

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

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

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

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

[61] Kanton St. Gallen - Amt für Gesundheitsversorgung und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Weiterentwicklung der Strategie der St.Galler Spitalverbunde | sg.ch*. https:

//www.sg.ch/gesundheit-soziales/gesundheit/gesundheitsversorgung--spitaeler-spitex/spitaele
spitalzukunft.html.

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

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

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

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

[66] 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.

[67] 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.

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

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

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

[71] CYBERTEC Guest. *A formula to calculate pgbench scaling factor for target DB size*. https://www.cybertec-postgresql.com/en/a-formula-to-calculate-pgbench-scaling-factor-for-target-db-size 2018.

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

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

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

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

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

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

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

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

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

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

[82] Anatoli Kreyman. *Was ist eigentlich Splunk?* https://www.kreyman.de/index.php/splunk/76-was-ist-eig

[83] 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.

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

[85] 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.

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

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

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

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

[90] Sujoy Nath. *Database Connection Pool*. https://medium.com/@sujoy.swe/database-connection-pool-647 2023.

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

[92] Diego Ongaro und John Ousterhout. „In Search of an Understandable Consensus Algorithm“. In: ().

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

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

[95] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[110] 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	Public Key Infrastructure
DCS	Distributed Configuration Store
DQL	Data Query Language

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

Glossar

Ansible Ansible ist ein Open-Soure Automatisierungstool zur Provisionierung, Konfiguration, Deployment und Orchestrierung. Ansible verbindet sich auf auf die Zielgeräte und führt dort die Hinterlegten Module aus. Oft werden die verschieden Aufgaben in einem Skript, in einem sogenannten Playbook geschrieben werden[84].. 1, 20, 109, 121, 129, 131, 134, 142, 143, 146, 147, 173, cxxvii

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[69].. 4, 18, 19, 52, 134, 135, 136, 138, 143, 150, 159, iii, lxxxviii, lxxxix, clxxxiv, clxxxviii, clxxxix, cxc

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[97] wurde.. 7, 74, 80

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[85].. 167, 173

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[12, 34].. 93

Connection Pooler Ein Connection Pooler hält permanent eine spezifische Menge an Connections zur Datennbank offen, die er wiederum einer Applikation präsentiert. Schliesst eine Applikation ihre Connection, behält sie der Pooler offen und verwendet sie wieder. Dadurch entfallen für die Applikation das öffnen der Connection, es wird an Overhead gespart die Anzahl Connections auf Seiten DB reduziert[90].. 119, 120, 121, 122, 134, 143, 145, 146

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

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[98].. 167

DCS Der DSC ist eine Kernkomponente von Patroni [24]. Realisiert wird der DCS bei Patroni mit Etcd.. 119, 122, 167

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.. 20, 124

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[87].. 7

etcd etcd ist Key-Value-Store, der konfigurationen in einem HA System speichert. etcd ist Konsensbasiert und dient zum erhalten eines Quorums[27].. 66, 82, 83, 119, 122, 123, 139, 145, 148, 169, iii, lxxvi

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 19, 42, 43, 58, 121, 143, 170

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

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[62, 107].. 170

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

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

HAProxy HAProxy ist ein Layer 7 Proxy und Load Balancer[30].. 60, 65, 82, 121, 122, 123, 125, 126, 131, 133, 134, 143, 145, 146

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[60]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 19, 58, 122

helm helm bietet mit seinen helm charts Paketressourcen, die das deployment von Kubernetes-Ressourcen erleichtert[31].. 87, 135, cvii, clxxxiii, clxxxix

HP-UX Dieses UNIX-Derivat ist ein abkömmling von System III, System V R3 und System V R4 und wurde von HP zum ersten Mal 1982 veröffentlicht.. 5, 11, 25, 167

IBM DB2 IBM DB2 ist eine Relationale Datenbank[80] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 44

keepalived keepalived nutzt VRRP um eine leichtgewichtige Lösung für ein HA-Failover zu 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[86, 52].. 58

Key-Value-orientierte Siehe Key-Value-Datenbank. 173

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. 170, 171, 172

Keycloak Keycloak ist ein Identity und Access Managament Tool[35], welches vom KSGR für SSO resp. Authentifizierung und das Management von Secrets für Kubernetes, und Applikationen die in Kubernetes betrieben werden, verwendet wird.. 122

Key-Value-Store Siehe Key-Value-Datenbank. 56, 74, 169

Kubernetes Kubernetes, oder k8s, ist eine Open-Source Containerplattform die ursprünglich von Google 2014 für die Bereitstellung und Orchestrierung von Containern entwickelt wurde aber 2015 an eine Tochter Foundation der Linux Foundation gespendet. Kubernetes kommt aus dem Griechischen und bedeutet Steuermann.. 1, 3, 11, 20, 100, 118, 134, 137, 142, 143, 144, 147, 167, 170, 171

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, 171

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

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.. 1, 5, 7, 11, 57

MetalLB MetalLB ist ein für Bare-Metal k8s Systeme ausgelegter Load-Balancer. Er kann sowohl auf Layer 2, mit OS-Boardmitteln arbeiten, bietet aber auch BGP-Routing an, so dass Pods direkt von Routern angesteuert werden können, ohne via Host gehen zu müssen[39].. 89, 93, 141, 144, 147, 148

Microsoft Azure SQL Database Microsoft Azure SQL Database oder auch Azure SQL ist eine Relationale Datenbank die von Microsoft für die Azure Cloud optimiert 2010 Entwickelt wurde[53].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[54].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[55]. 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, 172

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zurm ersten Mal 2007 veröffentlicht wurde[99].. 7

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

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

NoSQL NoSQL steht für Not only SQL. Das heisst, Relationale Datenbanken haben komponenten wie Dokumentendatenbanken, Graphendatenbanken, Key-Value-Datenbanken und Spaltenorientiert Datenbanken. Viele der grossen Datenbanklösungen wie Oracle Database oder Microsoft SQL Server sind NoSQL Datenbanken resp. bieten diese option an.. 7, 169, 170, 171, 173, 174

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[65, 64]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 7, 167

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.. 20

Oracle Database Die erste verfügbare Version der Oracle Datenbank kam im Jahr 1979 mit Version 2 (statt Version 1) heraus, damals allerdings nur mit den Basisfunktionen. Im Laufe der Zeit wuchs der Funktionsumfang sehr stark an, die Grundlage des Client-Server-Designs kam erstmals im Jahr 1985 mit Version auf den Markt und hat sich im Prinzip bis heute gehalten. Mit der mit Version 8/8i 1997 erschienen Optimizer und mit der Version 9i 2001 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, 11, 44, 144, 147, 172

PKI Eine Public Key Infrastructure ist ein System, dass für das Verschlüssen und Signieren von Daten oder die Zertifizierung verwendet wird.. 118, 146, 147, 167

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.. 1, 5, 7, 11, 17, 44, 58, 74, 135, 137, 139, 143, 145, 147

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

PostgreSQL Cluster Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wei Oracle.. 18, 19, 58, 91, 135, 137, 145, 146, 172, lxxvi, lxxxiii, lxxxviii

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.. 4, 5, 18, 20, 138, 139, 140, 143, 150, 151, 167, iii, cxc, cxcv

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. . 59, 144, 169, 170, cxcvi

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[59].. 74, 167

Red Hat Ansible Automation Platform Die Red Hat Ansible Automation Platform, ehemals Ansible Tower, ist eine Automatisierungsplattform für Ansible. Sie bietet eine mit Ansible gesteuerte CI/CD Umgebung an[44].. 121, 146

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.. 20

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[102]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 170

rke2 rke2 ist eine Leichgewichtige Kubernetes Distribution, die alles notwendige mitbringt, um einen k8s Cluster zu betreiben[32].. 3, 81, 88, 93, 101, 141, 144, 147, 148

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

SAN Ein Storage Area Network ist ein dediziertes Netzwerk aus Storage Komponenten. SAN Systeme bieten redundante Pools an Speicher. Die Physischen Festplatten werden zu Virtuellen Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[57, 88, 66]. 5, 20, 25, 104, 167

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[103].. 20, 118, 130, 146, 147, 167

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[106, 96]. 7

Split-brain Im Kapitel **??** werden die ursachen und folgenden eines Split-brains genauer besprochen. . 43, 173

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[75, 82]. . 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[100].. 7

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

Switchover In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geswitched.. 19, 121

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.. 167

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[101, 73].. 20

Transaktion Eine Transaktion ist beinhaltet Schreib-, Lese-, Mutatations- oder Löschoperatione auf Daten.. 53, 55

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.. 167

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

VRRP VRRP ist ein Protokoll, mit welchem Fallback-Router bereitgestellt werden können. Fällt der Primäre Router aus, so kommt der Fallback-Router zum Einsatz. Dank gewichtung wird der Primäre Router wieder aktiv, sobald er wieder erreichbar ist[48, 1].. 167, 170

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.. 11, 20

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 Disposition

Disposition Diplomarbeit

7 Bewilligung (wird durch die Schulleitung ausgefüllt)

Das Thema dieser Diplomarbeitsdisposition ist bewilligt.

Ort und Datum	Unterschrift Fachvorsteher
Sargans, 4.12.2023	

Anmerkungen und Hinweise zur Disposition:

Auf die folgenden Punkte ist bei der Bearbeitung der Diplomarbeit Rücksicht zu nehmen:

Legende:
! Hinweis auf kleine Ungereimtheiten
!! Grössere Schwächen, Fehler oder Unklarheiten
!!! gravierende Schwäche, kann so nicht verwendet werden

2 Ziele und Nutzen

!! Bei den zu erreichenenden Ziele sind Tasks nach Wichtigkeit und Priorität eingestuft worden. Hier gib es noch ungereimtheiten zwischen Wichtigkeit und Prriorität. Als Beispiel wird ein Task als Tief eingestuft und gleichzeitig als Muss Priorität gesetzt.

3 Abrenzungen

!! Die Systemabgrenzungsgrafik hilft nicht beim Verständnis deiner Arbeit. Die Abgrenzung deiner Arbeit sollte kar und präziese formuliert werden. Im Moment ist nicht klar, wo genau die Abgrenzungen sind. Die Grafik kann als zusätzliche Information beigefügt werden, wenn sie nicht zu überladen ist und Begriffe aufweisst, die in der Arbeit nicht weiter erwähnt werden.

4 Abhängigkeiten und Risiken

! Wie genau hilft eine SWOT-Analyse bei der Zielerreichung? wenn eine Swot Analyse eingesetzt wird, muss die Erkenntnis daraus in die Arbeit einfliessen und eine klaren Mehrwert ausweisen.

II 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		
17.03.2024	20:15	21:00	0.8	Dokumentation	Dokumentation	Dokumentation erweitern	Listings sauber machen. 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
27.03.2024	10:00	11:30	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	yugabytedb Installation			
27.03.2024	15:00	15:30	0.5	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	MetalLB Installation		YugabyteDB von aussen nicht ansprechbar.	MetalLB installiert und eingesetzt.
27.03.2024	13:00	16:00	4.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	MetalLB Troubleshooting		Trotz MetalLB nicht erreichbar	L2Advertisement gesetzt.
27.03.2024	16:30	17:30	1.0	2. Expertengespräch	2. Expertengespräch				
01.04.2024	09:00	09:45	0.8	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmaking	Versuch, das Benchmarking via Cronjobs auszuführen	ysql_bench hat keinen Passwort-Parameter. Ist zudem zu gut geschützt um mit Tricks das Passwort zu übergeben.	Manuell alle 10min ausführen.
01.04.2024	14:00	16:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmaking	Viel Zeit verloren für das manuelle Benchmarking		
02.04.2024	10:00	12:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation	StackGres verfolgt ein anderes Konzept als Yugabyte.		
03.04.2024	09:00	11:30	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation		Extension Server nicht erreichbar	
03.04.2024	14:00	15:30	0.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation	Installation von Patroni begonnen.	Foreman Repositories waren erreichbar, Internet-Repository nicht.	apt-Proxy setzen
09.04.2024	12:30	14:00	1.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation		Extension Server nicht erreichbar	
10.04.2024	10:00	11:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation		etcd-Server bereitet Probleme	
12.04.2024	12:30	14:45	2.2	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Installation		Extension Server nicht erreichbar	Proxy gesetzt und http erzwungen
15.04.2024	18:00	20:00	2.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	rke2 - local-path-provisioner 250GiB	Versuch, grosse Volumes einzubinden	Alles landet auf einem Node	
16.04.2024	13:00	16:00	3.0	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	rke2 - local-path-provisioner 250GiB	Versuch, grosse Volumes einzubinden	Alles landet auf einem Node	Node Annotations auf local-path-provisioner und StorageClass setzen
17.04.2024	10:00	11:00	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	YugabyteDB Benchmaking / Testing	Grosse Volumes testen		
17.04.2024	12:30	19:00	6.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Deployment / Testing		Table Distribution und Schema based Sharding nicht tauglich. Foreign-Key-Constraints werfen fehler.	Referenced Tables
19.04.2024	08:00	11:00	3.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	StackGres Benchmarking	Auch pgbench muss manuell ausgeführt werden		
19.04.2024	12:00	14:00:00:00	2.0	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	Analyse vitabaks/postgresql_cluster// / Architektur	Analyse von vitabaks/postgresql_cluster auf GitHub. Aufbau analog ursprüngliche Evaluationsplattform von Patroni. Architektur entsprechend zeichnen.		
19.04.2024	14:00	20:00	6.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Installation / Testing		etcd-Server bereitet Probleme	Vereinfachen soweit möglich
19.04.2024	20:00	21:00	1.0	Aufbau und Implementation Testsystem	Basisinfrastruktur	Patroni Test Server DMT / Auftrag	DMT Einträge für Patroni Testserver erstellt. Ticket an Niculin Fürst für Foreman Job.		
20.04.2024	10:00	12:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking			
20.04.2024	14:00	16:00	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Patroni Benchmarking / grosse Volumes	Auf erweiterte Disk umstellen und letzten Benchmark fahren		
22.04.2024	09:00	09:15	0.2	Evaluation	Gegenüberstellung				
22.04.2024	09:30	11:00	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
23.04.2024	13:45	14:00	0.2	Evaluation	Gegenüberstellung				
24.04.2024	15:00	17:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern			
25.04.2024	14:00	16:00	0.0	Evaluation	Gegenüberstellung				
26.04.2024	13:30	14:30	1.0	Evaluation	Variantenentscheid				
26.04.2024	14:30	17:30	3.0	Dokumentation	Dokumentation	Dokumentation erweitern			
29.04.2024	07:00	09:00	2.0	Aufbau und Implementation Testsystem	Basisinfrastruktur	Anpassungen / Passwörter	Prequenteries umgesetzt. Snapshots auf VMware vSphere gesetzt.		

III Protokoll - Fachgespräche

III.I Fachgespräch 14.02.2024

Kantonsspital Graubünden
Departement ICT
Michael Graber
Datenbank Administrator
Gäuggelistrasse 7
CH-7000 Chur

Standort Informatik
Tel. +41 (0)81 256 68 25
michael.graber@ksgr.ch
www.ksgr.ch

Protokoll

DIPLOMARBEIT - VERNISSAGE

Dienstag, 14. Februar 2024
18:00 – 21:15 Uhr

Zoom BreakoutSession Nr. 5 / 9

Chur, 13. Mai 2024

Teilnehmer	Bereich od. Funktion	Stellvertretung
Norman Süsstrunk (Vorsitz)	Fachexperte	
Curdin Roffler	Studierender	
Michael Graber	Studierender	

Beratend ohne Stimmrecht

Protokoll
Michael Graber

Gäste

Traktanden

	1.	**Begrüssung**
	2.	**Termine**
	3.	**Coaching**
	4.	**Administratives**
	5.	**Fragen und Antworten**
		Vorauswahl
	6.	**Varia / Aktualitäten**
	6.1	Zusammenlegung Breakout Session

I = Informationen **D** = Diskussion **E** = Entscheid **A** = Auftrag

Traktanden		Art	Verantw.	Termin
1.	**Begrüssung** Kurze Vorstellungsrunde von Norman Süsstrunk, Curdin Roffler und Michael Graber	I		
2. **2.1** **2.2** **2.3**	**Termine** Flexible Termingestaltung möglich Norman braucht eine Woche Vorlaufszeit Die Initiative für Expertengespräche soll von den Studenten kommen. Norman wird sich aber spätestens zur Halbzeit melden.	 I I A	 Curdin Michael	
3. **2.1** **2.2** **2.3**	**Coaching** Kriterien müssen klar Aufgenommen werden und Grob gefiltert werden Bei Fragen zur Diplomarbeit können die Studenten jederzeit um eine Coaching Session bitten Früh genug mit dem Verfassen des Berichts anfangen. Sonst wird die Zeit zu knapp	 A A	 Curdin Michael Curdin Michael	
4. **4.1** **4.2**	**Administratives** Spätestens zur Halbzeit muss ein Status an Norman gesendet werden Das Intervall des Statusreports muss KSGR Intern abgeklärt werden	 A A	 Curdin Michael Michael	
5. **5.1**	**Fragen und Antworten** **Darf eine Vorauswahl stattfinden, um den Aufwand zur reduzieren?** Eine Vorauswahl ist Sinnvoll und in diesem Rahmen fast zwingend Notwendig, da sonst viel Zuviel Zeit investiert werden müsste	 D	 Norman Michael	
6. **6.1**	**Varia / Aktualitäten** **Zusammenlegung Breakout Session** Die Breakout Sessions 5 (mit Curdin Roffler) und 9 (mit Michael Graber) wurden der Einfachheit halber zusammengelegt.			

Pendenzen
Pendenzenliste

Nächster Termin:	
Name der Sitzung	
Einreichung Traktanden inkl. Beilagen	
Versand Einladung	

Norman Süsstrunk
Experte
(Vorsitzender)

Michael Graber
Qualifikant
(Protokoll)

Kantonsspital Graubünden
Departement ICT
Michael Graber
Datenbank Administrator
Gäuggelistrasse 7
CH-7000 Chur

Standort Informatik
Tel. +41 (0)81 256 68 25
michael.graber@ksgr.ch
www.ksgr.ch

Protokoll

Sync Diplomarbeit

Mittwoch, 27. März 2024
16:30 – 17:30 Uhr

MS Teams

Chur, 7. April 2024

Teilnehmer	Bereich od. Funktion	Stellvertretung
Norman Süsstrunk (Vorsitz)	Experte	
Michael Graber	Qualifikant	

Beratend ohne Stimmrecht

Protokoll
Michael Graber

Gäste

Traktanden

	1.	**Begrüssung**
	2.	**Fragen und Antworten**
	2.1	Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden?
	2.2	Hast Du noch Vorschläge zu PostgreSQL HA Clustern gefunden?
	2.3	Soll ich die Gewichtung mit 100 Punkten oder 1000 machen?
	2.4	Soll die Disposition in den Anhang? Diese ist über 50 Seiten lang?
	2.5	Falls ich die Diplomarbeit zwecks Gegenlesen / Rechtsschreibeprüfung geben würde, müsste ich dies irgendwo angeben da Fremdleistung?
	3.	**Inputs Norman Süsstrunk**
	3.1	Siehe Details
	4.	**Varia / Aktualitäten**
	4.1	…

I = Informationen **D** = Diskussion **E** = Entscheid **A** = Auftrag

Traktanden		Art	Verantw.	Termin
1.	**Begrüssung**	I	Michael	
2.	**Fragen und Antworten**			
2.1	**Frage: Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden?** Gut wäre es, binnen ein bis zwei Wochen zuzusenden.	D		
2.2	**Hast Du noch Vorschläge zu PostgreSQL HA Clustern gefunden?** Norman hat nicht sehr viel Erfahrung. Hat bisher keine Erfahrung z.B. CloudNativePG. Hat mit einem Team mit Wien auf On-Premises gearbeitet für Non-HA Lösungen PostgreSQL bietet ja von Haus aus keine HA-Lösung	D		
2.3	**Soll ich die Gewichtung mit 100 Punkten oder 1000 machen?** Mit Hundert Punkten arbeiten. Ist oft eine grobe Abschätzung. Input / Vorschlag: Anforderungen reduzieren auf die wichtigsten. Sonst wird die Zeit knapp. Vorschlag: Durchgehen und vereinfachen und Norman vereinfachte Version senden	D/A	Michael	
2.4	**Soll die Disposition in den Anhang? Diese ist über 50 Seiten lang?** Disposition nicht in den Anhang setzen. Disposition ist ja Teil der Diplomarbeit	D		
2.5	**Falls ich die Diplomarbeit zwecks Gegenlesen / Rechtsschreibeprüfung geben würde, müsste ich dies irgendwo angeben da Fremdleistung?** Nein, muss nicht angegeben werden. Manchmal werden Personen, die z.B. Mithilfe bei der Korrektur gemacht haben, im Vorwort mit einem Dank bedacht. Es gibt aber keine Formale Anforderungen **Aufträge / Kommunikation / nächste Schritte** • Vereinfachen der Präferenzmatrix	D		
3.	**Inputs Norman Süsstrunk**			
3.1	**Rechtschreibung** Mit Rechtschreibeprogramm Prüfen. Mehr Typos, dafür viele. Gäbe eine Ungenügend mit den Fehlern.			
3.2	**Sprache** Nicht zu komplex geschrieben. Technisch beschrieben.			
3.3	**Inhalt** • Sehr viele Sachen drin, die nicht wirklich mit dem Thema zu tun haben	D/A		

Traktanden	Art	Verantw.	Termin
Bei der Einleitung => Kenndaten des KSGR. Drin lassen. Hat nichts mit dem Thema zu tun, Nicht noch mehr reintun, muss nicht raus. • Auflisten der DBs müssten nicht rein • Weniger ist mehr. Mehr auf den Punkt kommen • Kondensiert machen mit den wichtigsten Listenings. Es muss danach das System anhand der Doku gebaut werden können • Doku so machen wie man sie macht, nur etwas hübscher für die Arbeit **3.4 Kurze Präsentation YugabyteDB** • Kurz erklärt, wie YugabyteDB funktioniert • Aktuellen Stand der evaluation präsentiert **Aufträge / Kommunikation / nächste Schritte** • ACID aus dem Exkurs Architektur rausnehmen • Prüfen, ob Dokumentation etwas kompakter wird		Michael	
4. Varia / Aktualitäten **4.1 Protokoll** Protokolll verfassen **Aufträge / Kommunikation / nächste Schritte** • Protokoll genehmigen	A A	Michael Norman	

Pendenzen
<u>Pendenzenliste</u> • Protokoll binnen 1-2 Wochen • Vorschlag reduzierte Präferenzmatrix

Nächster Termin:	
Name der Sitzung	
Einreichung Traktanden inkl. Beilagen	
Versand Einladung	

Norman Süsstrunk
Experte
(Vorsitzender)

Michael Graber
Qulaifikant
(Protokoll)

IV Statusbericht

IV.I Status Report 1

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht	13.02.2024
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	
ICT veratw. Person	Michael Graber	PMA	-

Status	Ampel	Tendenz	Begründung
Gesamtprojekt			
Zeitplanung	◆	↓	Projekt ist Umfangreich und hat viele Teilaspekte, die es zu planen und berücksichtigen gilt
Ressourcen	▲	↓	Parallel läuft das Grossprojket "Erneuerung HP UX Plattform", wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.
Kosten	●	→	Kosten sind noch im Soll-Bereich

Tätigkeiten vergangene Berichtsperiode	Tätigkeiten nächste Berichtsperiode
- Dokumentenstruktur erstellt - Projektplanung erstellt - Versnissage vorbereitet - Statusbericht erstellt	- Anforderungskatalog erarbeiten - Vorbereitung Benchmarking

# nächste Lieferobjekte (inkl. allfällige Links)		Status	Erledigungsgrad	Soll Datum
LO-001	Anforderungskatalog	in arbeit	**60%**	23.02.2024
LO-002	Vorbereiten Benchmarking	in arbeit	**0%**	26.02.2024
LO-003				
LO-004				
LO-005				

# Risiken		Auswirkungsgrad	Massnahmen	Verantw.
R-001	Fehlende Ressourcen	●	Organisation und Selbstmanagement	
R-002	HP-UX Ablöseprojekt	●	Ressourcen reservieren	
R-003	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	●	Monitoring vorgängig ausbauen und Massnahmen definieren	
R-004	Schwächen beim Selbstmanagement und in der Selbstorganisation	●	Werkzeuge im Vorfeld definieren und bereitstellen	

Kostenübersicht	Abhängigkeiten zu anderen Projekten	Massnahmen
Verfügbare Finanzen bis Ende Projekt: $\frac{100\,\text{CHF}}{h} * 200h = 24\,000\,\text{CHF}$	Erneuerung HP UX Plattform 60002201	Ressourcen reservieren
	KSGR Provisioning System (KPS) =>Foreman Umgebung	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten

Bemerkungen / Informationen	Anträge

Eingereicht	Geprüft	Bemerkungen/Auftrag PMO
PL:	PMO:	
Datum:	Datum:	

erledigte Lieferobjekte (inkl. allfällige Links)

Status

Dokumentenstruktur und Dokument wurde mit LaTeX angelegt. Aus der Disposition wurde die Ausgangslage, das Riskmanagement, die Abgrenzung, die Zieldefinition sowie Abkürzungen und das Glossar übernommen. Nicht aus der Disposition wurde die SWOT-Analyse übernommen, da diese mehr Fragen aufwerfen denn Antworten liefern würde.

Mit MS Project 2016 wurde die Projektplanung erstellt. Die Struktur richtet sich weitestgehend nach der Dokumentenstruktur.

Risikomanagement

HP-UX Ablösung ExaCC-Projekt

Das grösste Risiko besteht nach wie vor im Parallelen Grossprojekt der HP-UX Ablösung. Dieses Projekt wird ab Mitte-Ende Februar eine nicht unerhebliche Menge an Ressourcen für die Architektur, Planung, Migrationsplanung usw., absorbieren.

Als Lösung für dieses Problem, werden Firmenintern Ressourcen reserviert. Nichtsdesto trotz besteht gefahr bezüglich der Priorisierung der beiden Projekte.

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, das es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Ein möglicher Ansatz besteht darin, die gängigsten Monolothischen und Distributed SQL Systeme vorzusortieren. Daraus die besten drei (zwei Distributed SQL und ein Monolothisches) genauer zu evaluieren. Dies ist ein Thema, das beim ersten Fachgespräch von meiner Seite traktandiert wird.

Weiteres vorgehen

Als nächstest steht die erarbeitung des Anforderungskatalog an.
Diese werden bei den internen Stakeholdern abgeholt.

Anschliessend wird anhand des Anforderungskatalog das Benchmarking vorbereitet.
So das die Evaluation mit sauberen Messdaten vonstatten gehen kann.

Anhand des Benchmarkings und der Anforderungen soll im Anschluss ein Testszenario für die Evaluation abgeleitet werden.

IV.II Status Report 2

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusberich	18.03.2024
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	
ICT veratw. Person	Michael Graber	PMA	-

Status	Ampel	Tendenz	Begründung
Gesamtprojekt			
Zeitplanung			In verzug. Grossprojket `Erneuerung HP UX Plattform` nimmt viel Zeit in Anschlag. Hinzu kommt, das die analyse gängiger PostgreSQL HA Lösungen ebenfalls viel Zeit kostet. Dokumentionsaufwand unterschätzt.
Ressourcen			Parallel läuft das Grossprojket `Erneuerung HP UX Plattform`, wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.
Kosten			Kosten sind noch im Soll-Bereich

Tätigkeiten vergangene Berichtsperiode	Tätigkeiten nächste Berichtsperiode
- Anforderungskatalog erstellt - Parallel dokumentiert	- Analyse der PostgreSQL HA Clusterlösungen abgeschlossen - Benchmarking abgeschlossen - Variantenentscheid getroffen - Basissystem für Testsystem aufgebaut

# nächste Lieferobjekte (inkl. allfällige Links)		Status	Erledigungsgrad	Soll Datum
LO-002	Vorbereiten Benchmarking	offen		
LO-003	Analyse PostgreSQL HA Cluster Lösungen	in Arbeit		
LO-004	Gegenüberstellung	offen		
LO-005	Variantenentscheid	offen		
LO-006	Aufbau Basisinfrakstruktur Testsystem	offen		

# Risiken		Auswirkungsgrad	Massnahmen	Verantw.
R-001	Fehlende Ressourcen		Organisation und Selbstmanagement	
R-002	HP-UX Ablöseprojekt		Ressourcen reservieren	
R-003	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden		Monitoring vorgängig ausbauen und Massnahmen definieren	
R-004	Schwächen beim Selbstmanagement und in der Selbstorganisation		Werkzeuge im Vorfeld definieren und bereitstellen	
R-005	Scope verlust während des Projekts		Ziele klar definieren	
R-006	Scope Creep		Ziele SMART definieren	
R-007	SIEM / Log Plattform nicht betriebsbereit			
R-008	Foreman nicht betriebsbereit		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	

Kostenübersicht	Abhängigkeiten zu anderen Projekten	Massnahmen
Verfügbare Finanzen bis Ende Projekt: $100\,\mathrm{CHF}\,\mathrm{h}^{-1} * 200 = 24\,000\,\mathrm{CHF}$	Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) =¿ Foreman Umgebung	Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten

Bemerkungen / Informationen	Anträge

Eingereicht	Geprüft	Bemerkungen/Auftrag PMO
PL: Datum:	PMO: Datum:	

# erledigte Lieferobjekte (inkl. allfällige Links)	
LO-001	Anforderungskatalog

Table 1: Zweiter Statusbericht

Status

Der Anforderungskatalog und die gewichtung wurde erstellt. Die Stakeholder wurden entsprechend eingebunden.

Analyse der bestehenden PostgreSQL HA Cluster lösungen begonnen. Die Analyse ist allerdings komplex und nimmt mehr Zeit in Anspruch, als erwartet.

Im Moment ist das Projekt in Verzug.

Risikomanagement

HP-UX Ablösung ExaCC-Projekt

Das grösste Risiko besteht nach wie vor im Parallelen Grossprojekt der HP-UX Ablösung. Dieses Projekt wird ab Mitte-Ende Februar eine nicht unerhebliche Menge an Ressourcen für die Architektur, Planung, Migrationsplanung usw., absorbieren.

Anfang April werden die beiden Nodes geliefert, entsprechend wird sich der Aufwand Ende April massiv erhöhen.

Als Lösung für dieses Problem, ist nach wie vor die Reservation von Terminen im KSGR am Zug. Dokumentationen werden aber auch am Abend und am Wochenende geschrieben.

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, das es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Es wurde eine vorsortierung der gängigen Methoden vorgenommen.

Weiteres vorgehen

Die Analyse muss fertiggestellt werden. Die Architektur und die Server für die Evaluationssysteme stehen und sind bereit für Installation von rke2 usw. Damit müssen noch die Architektur für das Monolithische System erstellt werden und dann der Aufbau der Evaluations-DBs in Angriff genommen werden.

Anschliessend wird anhand des Anforderungskatalog das Benchmarking vorbereitet.
So das die Evaluation mit sauberen Messdaten vonstatten gehen kann.

Anhand des Benchmarkings und der Anforderungen soll im Anschluss ein Testszenario für die Evaluation abgeleitet werden.

IV.III Status Report 3

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusberich	29.04.2024
Projektbeschreibung	Evaluation und Implementation PostgreSQL HA Cluster	Priorität	
ICT veratw. Person	Michael Graber	PMA	-

Status	Ampel	Tendenz	Begründung
Gesamtprojekt			
Zeitplanung			Stark im Verzug. Evaluation benötigte weitaus mehr Zeit, als geplant.
Ressourcen			Parallel läuft das Grossprojket `Erneuerung HP UX Plattform`, wo die bestehende HP-UX Plattform durch eine Oracle Exadata Cloud@Customer Plattform abgelöst wird. Ab dem Zeitpunkt der Lieferung der Hardware werden die Oracle Datenbanken der Kernapplikation auf die neue Plattform migriert. Dies über das gesamte Jahr und auch während der Diplomarbeit sehr viele Ressourcen binden.
Kosten			Kosten sind noch im Soll-Bereich

Tätigkeiten vergangene Berichtsperiode	Tätigkeiten nächste Berichtsperiode
- Evaliert - Variantenentscheid getroffen - Parallel dokumentiert	- Basissystem für Testsystem aufgebaut - Testsystem aufgebaut - Testsystem geprüft und getestet

# nächste Lieferobjekte (inkl. allfällige Links)		Status	Erledigungsgrad	Soll Datum
LO-006	Aufbau Basisinfrakstruktur Testsystem	In Arbeit		
LO-007	Installation Patroni PostgreSQL Cluster	offen		
LO-008	Technical Review Testsystem	offen		
L0-009	Testing	offen		
L0-010	Resultate	offen		

# Risiken		Auswirkungsgrad	Massnahmen	Verantw.
R-001	Fehlende Ressourcen		Organisation und Selbstmanagement	
R-002	HP-UX Ablöseprojekt		Ressourcen reservieren	
R-003	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden		Monitoring vorgängig ausbauen und Massnahmen definieren	
R-004	Schwächen beim Selbstmanagement und in der Selbstorganisation		Werkzeuge im Vorfeld definieren und bereitstellen	
R-005	Scope verlust während des Projekts		Ziele klar definieren	
R-006	Scope Creep		Ziele SMART definieren	
R-007	SIEM / Log Plattform nicht betriebsbereit			
R-008	Foreman nicht betriebsbereit		Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten	

Kostenübersicht	Abhängigkeiten zu anderen Projekten	Massnahmen
Verfügbare Finanzen bis Ende Projekt: $100\,\text{CHF}\,\text{h}^{-1} * 200 = 24\,000\,\text{CHF}$	Erneuerung HP UX Plattform 60002201 KSGR Provisioning System (KPS) = Foreman Umgebung	Ressourcen reservieren Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten

Bemerkungen / Informationen	Anträge

Eingereicht	Geprüft	Bemerkungen/Auftrag PMO
PL: Datum:	PMO: Datum:	

# erledigte Lieferobjekte (inkl. allfällige Links)	
LO-001	Anforderungskatalog
LO-002	Vorbereiten Benchmarking
LO-003	Analyse PostgreSQL HA Cluster Lösungen
LO-004	Gegenüberstellung
LO-005	Variantenentscheid

Status

Die PostgreSQL HA Löungen wurden evaluiert.
Es wurden drei Systeme Installiert:

- YugabyteDB
- StackGres - Citus
- Patroni

Die Analyse hat mehr Zeit in Anspruch genommen als geplant.
Es tauchten einige Probleme auf, welche nur bedingt mit der Arbeit zu tun haben.

Projekt stark im Verzug.

Risikomanagement

Umfang der Diplomarbeit

Ein weiteres Risiko besteht darin, das es eine relativ grosse Anzahl an Lösungen für einen PostgreSQL Cluster gibt. Die Gefahr sich hier zu verzetteln ist nicht unerheblich.

Es wurde eine vorsortierung der gängigen Methoden vorgenommen.

Weiteres vorgehen

Testsystem wird Installiert.

Anschliessend wird das System kurz einem review unterzogen und getestet.

Im Anschluss wird das Projekt abgeschlossen.

Kantonsspital Graubünden

Kantonsspital Graubünden
Departement 10 ICT
Data Center
www.ksgr.ch
Loëstrasse 170
CH-7000 Chur

AIP - Board

Projektkoordinator AIP
Tel. +41 (0)81 256 67 04
michael.guettler@ksgr.ch

13. Mai 2024

Abschlussbericht Projekt

PostgreSQL HA Cluster - Konzeption und Implementation

Projektcoach	Klicken oder tippen Sie hier, um Text einzugeben.	Projektleiter/-in	**Michael Graber**
		Co- Projektleiter	Klicken oder tippen Sie hier, um Text einzugeben.
Antragsteller	**Michael Graber**	KST	**P** Klicken oder tippen Sie hier, um Text einzugeben.
Departement	Departement 10		
PM-Auftragsnr.	**OTRS549108**		Klicken oder tippen Sie hier, um Text einzugeben.
Projekt - Art	ICT - Hard- und Software		

Gesamtprojekt Start/Ende **Von 14.02.2024 bis 24.05.2024** ☐ wie geplant ☒ ursprünglich anders geplant
(kurze Erklärung bei *Termine eingehalten*)

Initialisierung → Konzeption → Realisierung → Einführung → Abschluss

Initialisierung **AIP Board**	Klicken oder tippen Sie, um ein Datum einzugeben.	Start **Realisierung**	01.05.2024
Inbetriebnahme / Übergabe an Betrieb der Beschaffung / des Projekts			Klicken oder tippen Sie, um ein Datum einzugeben.
Equipment-Nr. / Inventar-Nr. (anzugeben, wenn Anlagen in Betrieb genommen wurden)			Klicken oder tippen Sie hier, um Text einzugeben.
Welcher Kostenstelle wird der künftig Betriebsunterhalt belastet?			**10890 Informatik**
PMA der Betriebskosten			Klicken oder tippen Sie hier, um Text einzugeben.

Welche Ziele wurden im Projekt erarbeitet/erreicht?

Plan-Ziele	Ziele erreicht?	Begründung
• Termine eingehalten	☐ **Ja** ☒ **Nein**	Der Endtermin wurde eingehalten. Das Restliche Projekt litt unter Verzug
• Interne Personalressourcen eingehalten	☒ **Ja** ☐ **Nein < 5%** ☐ **Nein > 5%**	Klicken oder tippen Sie hier, um Text einzugeben.
• Externe Kosten eingehalten	☒ **Ja** ☐ **Nein < 5%** ☐ **Nein > 5%**	Klicken oder tippen Sie hier, um Text einzugeben.

© Kantonsspital Graubünden, Chur

Ergebnisse

Drei Clustersysteme analysiert.
Patroni als zu bauenden Testsystem evaluiert.
Patroni mittels vitabacks / postgresql_cluster als Testsystem umgesetzt und getestet.

Erfolgserlebnisse

Sehr viel Erfahrung mit Kubernetes / rke2 sowie Ansible gesammelt.
Herausforderungen bei Kubernetes / rke2 gemeistert.
Funktionierendes Patroni Testsystem aufgebaut.

Schwierigkeiten / unvorhergesehene Risiken

- YugabyteDB kann als Open-Source DB oder als Subscription Version Yugaware Installiert werden
- etcd wurde während der Evaluation auf den Patroni Nodes installiert. Es kam zu einem Namenskonflikt der etcd- und Patroni Nodes
- Der Kubernetes Load Balancer MetalLB benötigte ein L2Advertisement auf den Adresspool. Ohne waren die Evaluationsdatenbanken von aussen nicht erreichbar
- Das Persistente Storage System local-path-provisioner wurde erst falsch konfiguriert.
  Daher wurden alle Volumes auf einen Kubernetes Node geschrieben, so dass ab einer gewissen DB Grösse der Storage nicht mehr ausreichte
- Auf den Kubernetes Hosts wurden die sproxy-Zertifikate nicht installiert.
  Dadurch konnte StackGres nicht auf sein Extension GitHub Repository zugreifen.
  Auch die Maintenance-Tools mussten via Proxy ihre Dependencis installieren
- Das Benchmarking konnte nicht automatisiert werden
- Mit vitabacks / postgresql_cluster konnte erst nicht auf Repositories von ausserhalb zugreifen.
  Zum einen wurden die Server auf die Zukünftige Firewall ohne Proxy geroutet und entsprechende Firewall Rules erstellt.
  Der andere Lösungsweg bestand darin, im Zentralen yml-File (main.yml) den Proxy sauber einzubinden
- Veeam Kasten K10 stand während der Diplomarbeit nicht zur Verfügung.
  Daher konnten bei den Distributed SQL Systemen das Backup nur sehr rudimentär betrachtet werden
- Während der gesamten Diplomarbeit stand, dass KSGR Kubernetes Testsystem nicht zur Verfügung. Daher musste das Maintenance-Tool auf die Evaluationsumgebung aufgebaut werden.
- Das HP-UX Ersatzprojekt nahm während der Diplomarbeit stark an Fahrt auf.
  Entsprechend musste die Diplomarbeit immer wieder unterbrochen werden.
- Die Evaluationsphase nahm weit mehr Zeit in Anspruch als geplant.
  Entsprechend kam es zu Verzug beim Projekt

Offene Punkte

- Das neue SIEM in Form von Elastic muss an das Testsystem angebunden werden
- Mittels des PKI und HashiCorp Vault muss die Testdatenbank sowie der Traffic verschlüsselt werden
- Das Testsystem muss an die Veeam Backup Infrastruktur angebunden werden
- vitabacks / postgresql_cluster muss in die KSGR Ansible Umgebung eingebunden werden
- Die beiden Maintenance-Tools müssen in das KSGR Kubernetes Testsystem eingebunden werden
- Das Produktivsystem muss aufgebaut und in Betrieb genommen werden

Zukunftsprognose / Trends

- YugabyteDB muss als Test- und Produktivsystem für die Cloud Native Applikationen aufgebaut werden
- Es muss ein Performance Warehouse für PostgreSQL konzipiert und aufgebaut werden

Erfahrungen / Empfehlungen

- rke2 bietet die Möglichkeit, mit einfachen mittel einen Kubernetes Cluster aufzubauen.
  Allerdings braucht es zusätzliche Komponenten wie local-path-provisioner oder MetalLB um das

© Kantonsspital Graubünden, Chur

System für Datenbanken nutzbar zu machen.
Für Produktive Umgebung braucht es daher eine komplexere und stabilere Umgebung

- YugabyteDB hat den Vorteil, selbst grössere Datenmengen komprimiert zu speichern.
  Setzt aber ein funktionierendes und synchrones Zeitsystem voraus.
  Zudem ist nicht der neueste PostgreSQL Stand implementiert.
- StackGres - Citus benötigt Architekturbedingt grosse Ressourcenmengen.
  Das Sharding von Citus hat Grenzen, so lassen sich Foreign Key Constraints nicht ohne weiteres sharden.
  StackGres ist zudem bis zu einem gewissen Grad eine Blackbox, Fehler lassen sich dadurch nur schwer evaluieren
- Patroni ist mit vitabacks / postgresql_cluster rasch deployt.
  Dessen Konzept birgt aber eine Schwäche, der Connection Pooler ist auf dem gleichen Server installiert wie Patroni. Dadurch wird eine Stärke des Poolers aufgehoben.
  Patroni selbst ist State oft he Art wenn es um PostgreSQL Clustering geht.

Was sind die Auswirkungen der Gross-Investition / des Projekts auf:

(zu jedem Stichwort ist eine Aussage zu machen, welche Auswirkungen eingetreten sind)

Personal:
☒ keine Veränderung ☐ Mehrbedarf ☐ Mehrbedarf andere Abteilung
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Immobilien / Räume:
☐ weitere Räume nötig ☐ Umbau bestehende Räume nötig
☐ wenig bauliche Massnahmen nötig ☒ Keine
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Mobilien / Geräte:
☐ zusätzliche Geräte nötig ☐ bestehende Geräte mussten ersetzt werden
☐ zusätzliche Büroeinrichtungen nötig
☐ bestehende Büroeinrichtungen mussten ersetzt werden
☐ zusätzliche ICT-Infrastruktur nötig
☐ bestehende ICT-Infrastruktur musste ersetzt werden
☒ Keine
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Material (Verbrauch-):
☐ neues Material nötig (Bezeichnung): Klicken oder tippen Sie hier, um Text einzugeben.
☐ Mehrbedarf bestehendes Material nötig
☒ Keine Veränderung zum IST vor Start der Gross-Investition / des Projekts

Wartung-/Servicevertrag:
☐ zusätzlicher Vertrag nötig (Vertragsart): Klicken oder tippen Sie hier, um Text einzugeben.
☐ bestehender Vertrag wurden verlängert
☒ Keine Veränderungen zum IST vor Start der Gross-Investition / des Projekts
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Prozesse:
☐ Prozessanpassungen im eigenen Fachbereich nötig
☐ Anpassungen bei anderen nötig
☒ Keine Anpassungen nötig
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Wirtschaftlichkeit:
☐ mehr Erträge ☐ tiefere Kosten ☐ höhere Kosten
☐ Medizinischer Nutzen ☐ betrieblicher Nutzen
☐ Patienten Nutzen ☒ keine Auswirkungen
Erläuterung: Klicken oder tippen Sie hier, um Text einzugeben.

Antrag

Klicken oder tippen Sie hier, um Text einzugeben.

© Kantonsspital Graubünden, Chur

Beilagen

☐ *Auszug Projekt- / Beschaffungskosten --> Finanzauszug ☐ ** Tabelle Auftragsvergaben

☐ Klicken oder tippen Sie hier, um Text einzugeben.

*ist zwingend beizulegen (von Projektkoordinator/-in einfordern)
** ist zwingend beizulegen, wenn im Projekt Aufträge an Externe vergeben wurden, die der Submissionspflicht unterstehen (durch Projektleiter [i.d.R. ST oder IT] auszufüllen)

13.05.2024	Michael Graber
Datum	Unterschrift Projektleitung
Klicken oder tippen Sie, um ein Datum einzugeben.	
Datum	Unterschrift Projektcoach

© Kantonsspital Graubünden, Chur

V 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. - HP-UX Probleme am Montag. Backups sind über das Weekend nicht durchgelaufen.
KW12	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.
KW13	Es benötigt eine «private container registry», mir fehlt die Expertise dazu. - Der Aufbau der Projektplanung entpuppt sich begrenzt nutzbar. Das erstellen der Evaluationsinfrast
KW13	- Das Problem mit dem «private container registry»rührte daher, dass das YugabyteDB Anywhere (Repository yugaware) verwendet wurde. Kurz ein Schock, dass YugabyteDB ausgeschieden ist.
KW13	Später bemerkte ich, dass man das Repo yugabytedb auswählen muss.
KW13	- MetalLB benötigt zwingend L2Advertisement, damit Linux die Kommunikation von aussen nach innen leiten kann. - Bereits jetzt viel über Kubernetes, Ranger (rke2) und Helm gelernt.

VI Evaluation

VI.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

VI.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

VI.III Maintenance - StackGres - Citus

Bei StackGres gab es im letzten Monat keine wirkliche Bewegung:

Abbildung XVII: Stackgres - Pulse

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

VI.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

VII Evaluationssysteme - Installation

VII.I rke2

VII.I.I Vorbereitung

Da Package aus WAN-Repositories geladen werden, muss eine Proxy-Connection nach aussen gemacht werden können:

```
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

VII.I.II Installation

VII.I.II.I server - sks1183

Es gibt kein apt-Package. Daher muss zuerst das tarball-Package heruntergeladen werden.
Zuerst wird das Verzeichnis für rke2 erstellt:

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

Listing 2: rke2 server - Verzeichnis erstellen

```
# /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:

```
# /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

VII.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

VII.I.II.III Cluster Konfiguration

VII.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

VII.I.II.IV agents

VII.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:

```yaml
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

VII.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:

```yaml
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: MetallB konfigurieren

Das Manifest musste danach eingespielt werden:

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

Listing 21: MetallB Konfiguration einspielen

VII.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:

```yaml
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":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1183",
                        "paths":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1184",
                        "paths":["/srv/data/local-path-provisioner"]
                },
                {
                        "node":"sks1185",
                        "paths":["/srv/data/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 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

VII.II yugabyteDB

VII.II.I Prerequisites

VII.II.I.I StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```
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

VII.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:
    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.
```

```yaml
      memory: 4Gi

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

partition:
  master: 0
  tserver: 0

updateStrategy:
  type: RollingUpdate

# Used in Multi-AZ setup
masterAddresses: ""

isMultiAz: false
AZ: ""

# Disable the YSQL
disableYsql: false

tls:
  # Set to true to enable the TLS.
  enabled: false
  nodeToNode: true
  clientToServer: true
  # Set to false to disallow any service with unencrypted communication from joining this cluster
  insecure: false
  # Set enabled to true to use cert-manager instead of providing your own rootCA
  certManager:
    enabled: false
    # Will create own ca certificate and issuer when set to true
    bootstrapSelfsigned: true
    # Use ClusterIssuer when set to true, otherwise use Issuer
    useClusterIssuer: false
    # Name of ClusterIssuer to use when useClusterIssuer is true
    clusterIssuer: cluster-ca
    # Name of Issuer to use when useClusterIssuer is false
    issuer: yugabyte-ca
    certificates:
      # The lifetime before cert-manager will issue a new certificate.
      # The re-issued certificates will not be automatically reloaded by the service.
      # It is necessary to provide some external means of restarting the pods.
      duration: 2160h # 90d
      renewBefore: 360h # 15d
      algorithm: RSA # ECDSA or RSA
      # Can be 2048, 4096 or 8192 for RSA
      # Or 256, 384 or 521 for ECDSA
      keySize: 2048

  ## When certManager.enabled=false, rootCA.cert and rootCA.key are used to generate TLS certs.
```

```yaml
  ## When certificate.enabled=true and boostrapSelfsigned=true, rootCA is ignored.
  ## When certificate.enabled=true and bootstrapSelfsigned=false, only rootCA.cert is used
  ## to verify TLS certs generated and signed by the external provider.
  rootCA:
    cert: "
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ0lCQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJRd0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakFlRncweE9UQXlNRGd3TURRd01qSmFGdzB5T1RBeU1EVXdNRFF3TWpK
    ="
    key: "
    LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBxOXJpWDZqRXJlYnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQL0JlM2RxMW4yb0RDZkZUTDB4aTI0V01kTFFyckEy
    ="
  ## When tls.certManager.enabled=false
  ## nodeCert and clientCert will be used only when rootCA.key is empty.
  ## Will be ignored and genSignedCert will be used to generate
  ## node and client certs if rootCA.key is provided.
  ## cert and key are base64 encoded content of certificate and key.
  nodeCert:
    cert: ""
    key: ""
  clientCert:
    cert: ""
    key: ""

gflags:
  master:
    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
```

```yaml
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"

Services:
  - name: "yb-masters"
    label: "yb-master"
    skipHealthChecks: false
    memory_limit_to_ram_ratio: 0.85
    ports:
      http-ui: "7000"
      tcp-rpc-port: "7100"

  - name: "yb-tservers"
    label: "yb-tserver"
    skipHealthChecks: false
    ports:
      http-ui: "9000"
      tcp-rpc-port: "9100"
      tcp-yql-port: "9042"
      tcp-yedis-port: "6379"
      tcp-ysql-port: "5433"
      http-ycql-met: "12000"
      http-yedis-met: "11000"
      http-ysql-met: "13000"
      grpc-ybc-port: "18018"


## Should be set to true only if Istio is being used. This also adds
```

```yaml
## the Istio sidecar injection labels to the pods.
## TODO: remove this once
## https://github.com/yugabyte/yugabyte-db/issues/5641 is fixed.
##
istioCompatibility:
  enabled: false

## Settings required when using multicluster environment.
multicluster:
  ## Creates a ClusterIP service for each yb-master and yb-tserver
  ## pod.
  createServicePerPod: false
  ## creates a ClusterIP service whos name does not have release name
  ## in it. A common service across different clusters for automatic
  ## failover. Useful when using new naming style.
  createCommonTserverService: false

  ## Enable it to deploy YugabyteDB in a multi-cluster services enabled
  ## Kubernetes cluster (KEP-1645). This will create ServiceExport.
  ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#registering_a_service_for_export
  ## You can use this gist for the reference to deploy the YugabyteDB in a multi-cluster scenario.
  ## Gist - https://gist.github.com/baba230896/78cc9bb6f4ba0b3d0e611cd49ed201bf
  createServiceExports: false

  ## Mandatory variable when createServiceExports is set to true.
  ## Use: In case of GKE, you need to pass GKE Hub Membership Name.
  ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#enabling
  kubernetesClusterId: ""

  ## mcsApiVersion is used for the MCS resources created by the
  ## chart. Set to net.gke.io/v1 when using GKE MCS.
  mcsApiVersion: "multicluster.x-k8s.io/v1alpha1"

serviceMonitor:
  ## If true, two ServiceMonitor CRs are created. One for yb-master
  ## and one for yb-tserver
  ## https://github.com/coreos/prometheus-operator/blob/master/Documentation/api.md#servicemonitor
  ##
  enabled: false
  ## interval is the default scrape_interval for all the endpoints
  interval: 30s
  ## extraLabels can be used to add labels to the ServiceMonitors
  ## being created
  extraLabels: {}
    # release: prom

  ## Configurations of ServiceMonitor for yb-master
  master:
    enabled: true
    port: "http-ui"
    interval: ""
    path: "/prometheus-metrics"

  ## Configurations of ServiceMonitor for yb-tserver
```

```yaml
  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: {}

nodeSelector: {}

affinity: {}

statefulSetAnnotations: {}
```

```yaml
networkAnnotation: {}

commonLabels: {}

master:
  ## 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-master"
  #       topologyKey: kubernetes.io/hostname
  #
  # For further examples, see examples/yugabyte/affinity_overrides.yaml
  affinity: {}

  ## Extra environment variables passed to the Master pods.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
  ## Example:
  # extraEnv:
  # - name: NODE_IP
  #   valueFrom:
  #     fieldRef:
  #       fieldPath: status.hostIP
  extraEnv: []

  # secretEnv variables are used to expose secrets data as env variables in the master pod.
  # TODO Add namespace also to support copying secrets from other namespace.
  # secretEnv:
  # - name: MYSQL_LDAP_PASSWORD
  #   valueFrom:
  #     secretKeyRef:
  #       name: secretName
  #       key: password
  secretEnv: []

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

```

```yaml
  ## Labels to be added to the Master pods.
  podLabels: {}

  ## Tolerations to be added to the Master pods.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#toleration-v1-core
  ## Example:
  # tolerations:
  # - key: dedicated
  #   operator: Equal
  #   value: experimental
  #   effect: NoSchedule
  tolerations: []

  ## Extra volumes
  ## extraVolumesMounts are mandatory for each extraVolumes.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
  ## Example:
  # extraVolumes:
  # - name: custom-nfs-vol
  #   persistentVolumeClaim:
  #     claimName: some-nfs-claim
  extraVolumes: []

  ## Extra volume mounts
  ## 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:
```

```yaml
#     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.
## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#envvar-v1-core
## Example:
# extraEnv:
# - name: NODE_IP
#   valueFrom:
#     fieldRef:
#       fieldPath: status.hostIP
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.
## Example
# secretEnv:
# - name: MYSQL_LDAP_PASSWORD
#   valueFrom:
#     secretKeyRef:
#       name: secretName
#       namespace: my-other-namespace-with-ldap-secret
#       key: password
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
## Example:
# tolerations:
# - key: dedicated
#   operator: Equal
#   value: experimental
#   effect: NoSchedule
tolerations: []

## Sets the --server_broadcast_addresses flag on the TServer, no
## preflight checks are done for this address. You might need to add
## ‘use_private_ip: cloud‘ to the gflags.master and gflags.tserver.
serverBroadcastAddress: ""
```

```yaml

  ## Extra volumes
  ## extraVolumesMounts are mandatory for each extraVolumes.
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volume-v1-core
  ## Example:
  # extraVolumes:
  # - name: custom-nfs-vol
  #   persistentVolumeClaim:
  #     claimName: some-nfs-claim
  extraVolumes: []

  ## Extra volume mounts
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#volumemount-v1-core
  ## Example:
  # extraVolumeMounts:
  # - name: custom-nfs-vol
  #   path: /home/yugabyte/nfs-backup
  extraVolumeMounts: []

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

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


helm2Legacy: false

ip_version_support: "v4_only" # v4_only, v6_only are the only supported values at the moment

# For more https://docs.yugabyte.com/latest/reference/configuration/yugabyted/#environment-variables
authCredentials:
  ysql:
    user: "yadmin"
    password: "TES2&Daggerfall"
    database: ""
  ycql:
    user: ""
    password: ""
    keyspace: ""

oldNamingStyle: true

preflight:
  # Set to true to skip disk IO check, DNS address resolution, and
  # 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

VII.II.III Rekonfiguration mit 250GiB Storage

VII.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

VII.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:

```
# 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
---
```

```yaml
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

VII.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
```

```yaml
  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:
      cpu: "2"
      memory: 12Gi

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

partition:
  master: 0
  tserver: 0

updateStrategy:
  type: RollingUpdate

# Used in Multi-AZ setup
masterAddresses: ""

isMultiAz: false
AZ: ""

# Disable the YSQL
disableYsql: false

tls:
  # Set to true to enable the TLS.
  enabled: false
```

```yaml
  nodeToNode: true
  clientToServer: true
  # Set to false to disallow any service with unencrypted communication from joining this cluster
  insecure: false
  # Set enabled to true to use cert-manager instead of providing your own rootCA
  certManager:
    enabled: false
    # Will create own ca certificate and issuer when set to true
    bootstrapSelfsigned: true
    # Use ClusterIssuer when set to true, otherwise use Issuer
    useClusterIssuer: false
    # Name of ClusterIssuer to use when useClusterIssuer is true
    clusterIssuer: cluster-ca
    # Name of Issuer to use when useClusterIssuer is false
    issuer: yugabyte-ca
    certificates:
      duration: 2160h # 90d
      renewBefore: 360h # 15d
      algorithm: RSA # ECDSA or RSA
      # Can be 2048, 4096 or 8192 for RSA
      # Or 256, 384 or 521 for ECDSA
      keySize: 2048
  rootCA:
    cert: "
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ0lCQVRBTkJna3Foa2lHOXcwQkFRc0ZBREFXTVJRd0VnWURWUVFERXd0WmRXZGgKWW5sMFpTQkVRakFlRncweE9UQXlNRGd3TURRd01qSmFGdzB5T1RBeU1EVXdNRFF3TWpK
    ="
    key: "
    LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeGtPYUJoeTBxOXJpWDZqRXJlYnJMck5YMk54d1ZCCmNVcWJkUlhVc3VZNS96RURQL0JlM2RxMW4yb0RDZkZUTDB4aTI0V01kTFFyckEy
    ="
  nodeCert:
    cert: ""
    key: ""
  clientCert:
    cert: ""
    key: ""

gflags:
  master:
    default_memory_limit_to_ram_ratio: 0.85
  tserver:
    ysql_beta_features: true                      # gramic, wird üfr vacuum öbentigt
  follower_unavailable_considered_failed_sec: 300 # gramic, autobalancing auf 5 minuten heruntersetzen

PodManagementPolicy: Parallel

enableLoadBalancer: true

ybc:
  enabled: false


ybCleanup:
   resources:
     requests:
```

```yaml
      cpu: "1"
      memory: 0.5Gi
    limits:
      cpu: "1"
      memory: 0.5Gi

domainName: "cluster.local"

serviceEndpoints:
  - name: "yb-master-ui"
    type: LoadBalancer
    annotations:
      metallb.universe.tf/loadBalancerIPs: 10.0.20.151
    clusterIP: ""
    ## Sets the Service's externalTrafficPolicy
    externalTrafficPolicy: "Cluster"
    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: "Cluster"
    app: "yb-tserver"
    loadBalancerIP: ""
    ports:
      tcp-yql-port: "9042"
      tcp-yedis-port: "6379"
      tcp-ysql-port: "5433"

Services:
  - name: "yb-masters"
    label: "yb-master"
    skipHealthChecks: false
    memory_limit_to_ram_ratio: 0.85
    ports:
      http-ui: "7000"
      tcp-rpc-port: "7100"

  - name: "yb-tservers"
    label: "yb-tserver"
    skipHealthChecks: false
    ports:
      http-ui: "9000"
      tcp-rpc-port: "9100"
      tcp-yql-port: "9042"
      tcp-yedis-port: "6379"
      tcp-ysql-port: "5433"
      http-ycql-met: "12000"
```

```yaml
      http-yedis-met: "11000"
      http-ysql-met: "13000"
      grpc-ybc-port: "18018"


## Should be set to true only if Istio is being used. This also adds
## the Istio sidecar injection labels to the pods.
## TODO: remove this once
## https://github.com/yugabyte/yugabyte-db/issues/5641 is fixed.
##
istioCompatibility:
  enabled: false

## Settings required when using multicluster environment.
multicluster:
  ## Creates a ClusterIP service for each yb-master and yb-tserver
  ## pod.
  createServicePerPod: false
  ## creates a ClusterIP service whos name does not have release name
  ## in it. A common service across different clusters for automatic
  ## failover. Useful when using new naming style.
  createCommonTserverService: false

  ## Enable it to deploy YugabyteDB in a multi-cluster services enabled
  ## Kubernetes cluster (KEP-1645). This will create ServiceExport.
  ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#registering_a_service_for_export
  ## You can use this gist for the reference to deploy the YugabyteDB in a multi-cluster scenario.
  ## Gist - https://gist.github.com/baba230896/78cc9bb6f4ba0b3d0e611cd49ed201bf
  createServiceExports: false

  ## Mandatory variable when createServiceExports is set to true.
  ## Use: In case of GKE, you need to pass GKE Hub Membership Name.
  ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-cluster-services#enabling
  kubernetesClusterId: ""

  ## mcsApiVersion is used for the MCS resources created by the
  ## chart. Set to net.gke.io/v1 when using GKE MCS.
  mcsApiVersion: "multicluster.x-k8s.io/v1alpha1"

serviceMonitor:
  ## If true, two ServiceMonitor CRs are created. One for yb-master
  ## and one for yb-tserver
  ## https://github.com/coreos/prometheus-operator/blob/master/Documentation/api.md#servicemonitor
  ##
  enabled: false
  ## interval is the default scrape_interval for all the endpoints
  interval: 30s
  ## extraLabels can be used to add labels to the ServiceMonitors
  ## being created
  extraLabels: {}
    # release: prom

  ## Configurations of ServiceMonitor for yb-master
  master:
```

```yaml
    enabled: true
    port: "http-ui"
    interval: ""
    path: "/prometheus-metrics"

  ## Configurations of ServiceMonitor for yb-tserver
  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: {}
```

```yaml
nodeSelector: {}

affinity: {}

statefulSetAnnotations: {}

networkAnnotation: {}

commonLabels: {}

master:
  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#affinity-v1-core
  # For further examples, see examples/yugabyte/affinity_overrides.yaml
  affinity: {}

  ## Extra environment variables passed to the Master 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 master pod.
  # TODO Add namespace also to support copying secrets from other namespace.
  secretEnv: []

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

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

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

```yaml

  ## 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: []

  ## Sets the --server_broadcast_addresses flag on the TServer, no
  ## preflight checks are done for this address. You might need to add
  ## `use_private_ip: cloud` to the gflags.master and gflags.tserver.
  serverBroadcastAddress: ""

  ## Extra volumes
  ## extraVolumesMounts are mandatory for each extraVolumes.
  extraVolumes: []

  ## Extra volume mounts
  extraVolumeMounts: []

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

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

helm2Legacy: false

ip_version_support: "v4_only" # v4_only, v6_only are the only supported values at the moment

# For more https://docs.yugabyte.com/latest/reference/configuration/yugabyted/#environment-variables
authCredentials:
  ysql:
    user: "yadmin"
    password: "TES2&Daggerfall"
    database: ""
  ycql:
    user: ""
    password: ""
    keyspace: ""

```

```yaml
oldNamingStyle: true

preflight:
  # Set to true to skip disk IO check, DNS address resolution, and
  # 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 35: yugabyteDB - Helm Chart Manifest 250GiB

VII.II.IV SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle `pgbench_eval_bench` erstellt.
Zudem wird je ein Tablespace für die Indizes (`eval_index_tablespace`) und Daten (`eval_data_tablespace`) erstellt.
Die 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:

```sql
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:

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

--  Tablespace für Indices
```

```sql
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:

```sql
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

VII.II.V SQL Statements - Testing

Entsprechend dem ERD, müssen die Tabellen erstellt werden: Evaluation - ERD self_healing_test Die Tabelle heisst entsprechend `self_healing_test`. Auch hier soll ein Index-Tablespace (`self_healing_indices_tablespace`) und ein Daten-Tablespace (`self_healing_datas_tablespace`) erstellt werden.
Neu dazu kommt der Longtext-Tablespace `self_healing_longtexts_tablespace`. Erst werden die Rollen erstellt, gefolgt von den Usern und Schemas.
Die Schemas müssen entsprechend mittels `GRANT` berechtigt werden. Die Tabellen müssen entsprechend den Schemas erstellt werden, wobei einige Tabellen in 3 Tablets pro `tserver` gesplittet werden sollen (also insgesamt 9). Das gesamte `CREATE`-Skript:

```sql
--  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;
CREATE TABLESPACE self_healing_datas_tablespace WITH (replica_placement='{"num_replicas": 3, "placement_blocks": [ {"cloud":"cloud1","region":"datacenter1","zone":"rack1","min_num_replicas":3}]}');

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

--  Rollen erstellen
drop role if exists hrm;
create role hrm;
```

```sql
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;

drop user if exists cso_1;
drop user if exists cso_2;
create user cso_1 with password 'cso1'role customer_service_officers;
create user cso_2 with password 'cso2' role customer_service_officers;

drop user if exists la_1;
drop user if exists la_2;
create user la_1 with password 'la1' role legal_affairs;
create user la_2 with password 'la2' role legal_affairs;

--  Schemas erstellen
drop schema if exists hrm;
create schema hrm authorization hrm;
drop schema if exists accountands;
create schema accountands authorization accountands;
drop schema if exists customer_service_officers;
create schema customer_service_officers authorization customer_service_officers;
drop schema if exists generell;
create schema generell;

--  GRANTS erstellen
grant all on all tables in schema hrm to legal_affairs;
grant all on all tables in schema accountands to legal_affairs;
grant all on all tables in schema customer_service_officers to legal_affairs;
grant all on all tables in schema generell to legal_affairs;
grant all on all tables in schema hrm to yadmin;
grant all on all tables in schema accountands to yadmin;
grant all on all tables in schema customer_service_officers to yadmin;
grant all on all tables in schema generell to yadmin;

-- self_healing_accounts für Schema customer_service_officers
drop table if exists customer_service_officers.self_healing_accounts;
create table customer_service_officers.self_healing_accounts (
    account_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),
```

```sql
    mail varchar(255) check (mail like '%@%')
) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
create unique index accounts_personal_mark on customer_service_officers.self_healing_accounts(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;

--  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 '%@%')
) tablespace self_healing_datas_tablespace SPLIT INTO 3 TABLETS;
create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;

--  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
) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;

--  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
) tablespace self_healing_longtexts_tablespace SPLIT INTO 3 TABLETS;

--  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);
```

Listing 39: YugabyteDB - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```sql
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');
```

```sql
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 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

VII.III sks9016 - yugabyteDB

VII.III.I yugabyteDB - Download und Installation yugabyteDB

Ohne yugabyteDB zu installieren, lässt sich `ysql_bench` nicht ausführen. Daher muss das ganze Package erst heruntergeladen werden:

```
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.

VII.IV Patroni

VII.IV.I Prerequisites

Ganz am Anfang steht die Firewall.

Die Rules müssen auf `sks1232`, `sks1233`, `sks1234` und `sks9016` gesetzt werden:

```
# sks1232 / sks1233 / sks1234 / sks9016(10.0.28.16)
nano /etc/iptables/rules.v4
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-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
# generell
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 443 -j ACCEPT
# postgres
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
# patroni
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2379 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2380 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2376 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6432 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8008 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 7000 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8080 -j ACCEPT
COMMIT
# Completed

systemctl restart iptables
systemctl status iptables
```

Listing 46: Patroni - Firewall Settings

Danach 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 47: Patroni - Proxy Settings

Damit das PostgreSQL-Repository eingebunden werden kann,
muss dem apt-Proxy gesetzt werden.
Da via Foreman Installiert wurde, muss dieser ausgenommen werden:

```
# 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 48: Patroni - apt-Proxy Settings

Im nächsten Schritt kann das PostgreSQL-Repository eingebunden werden.

☞ Achtung, die von PostgreSQL beschriebene Variante wurde in Debian 10 als Deprecated gesetzt, mit Debian 13 wird diese Repository-Integration einen Fehler werden.

```
# 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 49: Patroni - PostgreSQL einbinden

Nun muss der PostgreSQL Cluster, Patroni, python3-etcd und python3-psycopg2 installiert werden:

```
apt install postgresql-16 postgresql-server-dev-16 patroni python3-etcd python3-psycopg2
```

Listing 50: Patroni - Prerequisites installieren

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

```
systemctl stop postgresql patroni
```

Listing 51: Patroni - Stop Patroni und PostgreSQL

Anschliessend muss noch vom PostgreSQL-Verzeichnis `/usr/lib/postgresql/16/bin/` ein Symlink nach `/usr/sbin/` gesetzt werden:

```
ln -s /usr/lib/postgresql/16/bin/* /usr/sbin/
```

Listing 52: Patroni - Symlink binaries

Zu guter Letzt sollte geprüft werden, ob alle Versionen passen und am richtigen Ort sind:

```
which patroni
which psql
patroni --version
```

Listing 53: Patroni - Checks

Damit kann zum etcd übergegangen werden.

VII.IV.II Installation etcd

Auf `sks9016` sollte erst das Repository angepasst werden und anschliessend der `etcd-server` installiert werden:

```
apt update
apt install etcd-server
```

Listing 54: etcd - Installation

Die Konfiguration ist simpel.
Die IP muss gesetzt werden, ein Listener auf Localhost und IP gesetzt werden:

```
# nano /etc/default/etcd
ETCD_LISTEN_PEER_URLS="http://10.0.28.16:2380"
ETCD_LISTEN_CLIENT_URLS="http://localhost:2379,http://10.0.28.16:2379"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.0.28.16:2380"
ETCD_INITIAL_CLUSTER="default=http://10.0.28.16:2380,"
ETCD_ADVERTISE_CLIENT_URLS="http://10.0.28.16:2379"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster"
ETCD_INITIAL_CLUSTER_STATE="new"
```

Listing 55: etcd - Konfiguration

Der Service sollte neu gestartet und seine lauffähigkeit getestet werden:

```
systemctl restart etcd
systemctl is-enabled etcd
systemctl status etcd
```

Listing 56: etcd - restart

Es sollte nun ein Member gelistet werden:

```
etcdctl member list
```

Listing 57: etcd - member list

Damit kann nun das Bootstrapping gestartet werden.

VII.IV.III Bootstrapping

Zuerst müssen die yml-Konfigurationen gesetzt werden.
Am besten wird das bestehende File jeweils gelöscht:

```
rm /etc/patroni/config.yml
nano /etc/patroni/config.yml
```

Listing 58: Patroni Bootstrap - Konfiguration bereinigen

Die yml-Files sehen wie folgt aus:

```
# Scope of PostgreSQL
scope: postgres
# Namespace for the PostgreSQL database
namespace: /db/
# Name of the PostgreSQL instance
name: postgres01
# Patroni REST API Configuration
restapi:
    # The IP address and port on which the REST API should listen
```

```yaml
    listen: 10.0.20.110:8008

    # The IP address and port to which clients should connect
    connect_address: 10.0.20.110:8008
# Patroni Etcd Configuration
etcd3:
    # The host address and port of the Etcd server
    host: 10.0.28.16:2379
# Patroni Bootstrap Configuration
bootstrap:
    # Configuration parameters for distributed configuration store (DCS)
    dcs:
        ttl: 30
        loop_wait: 10
        retry_timeout: 10
        maximum_lag_on_failover: 1048576
        postgresql:
            # Use pg_rewind during bootstrap
            use_pg_rewind: true
            #   Change pg_wal
            create_replica_methods:
                - basebackup
            basebackup:
                max-rate: '100M'
                waldir: "/srv/data/pg_wal"
    # Initdb configuration
    # Initdb configuration
    initdb:
        - auth: scram-sha-256
        - encoding: UTF8
        - data-checksums
    # pg_hba.conf entries for replication and general access
    pg_hba:
        - host replication replicator 127.0.0.1/32 scram-sha-256
        - host replication replicator 10.0.20.110/0 scram-sha-256
        - host replication replicator 10.0.20.111/0 scram-sha-256
        - host replication replicator 10.0.20.112/0 scram-sha-256
        - host all all 0.0.0.0/0 scram-sha-256
    # Adding default user admin with password admin
    users:
      admin:
          password: admin
          options:
              - createrole
              - createdb
# Patroni PostgreSQL Configuration
postgresql:
    # PostgreSQL server listening address and port
    listen: 10.0.20.110:5432
    # Connect address for PostgreSQL clients
    connect_address: 10.0.20.110:5432
    # Data directory for PostgreSQL
    data_dir: /var/lib/patroni
    # Path to the pgpass file
```

```yaml
    pgpass: /tmp/pgpass
    # Authentication configuration
    authentication:
        # Replication of user credentials
        replication:
            username: replicator
            password: replicator
        # Superuser credentials
        superuser:
            username: postgres
            password: postgres
    # Additional PostgreSQL parameters
    parameters:

        # Directory for Unix socket
        unix_socket_directories: '.'
        # Password encryption method
        password_encryption: 'scram-sha-256'
# Patroni Tags Configuration
tags:
    # Prevents a node from being promoted in case of failure
    nofailover: false
    # Prevents the load balancer from considering this node
    noloadbalance: false
    # Prevents a replica from being created by cloning
    clonefrom: false
    # Prevents synchronous replication from being enforced
    nosync: false
```

Listing 59: Patroni - Konfiguration - sks1232

```yaml
# Scope of PostgreSQL
scope: postgres
# Namespace for the PostgreSQL database
namespace: /db/
# Name of the PostgreSQL instance
name: postgres02
# Patroni REST API Configuration
restapi:
    # The IP address and port on which the REST API should listen
    listen: 10.0.20.111:8008

    # The IP address and port to which clients should connect
    connect_address: 10.0.20.111:8008
# Patroni Etcd Configuration
etcd3:
    # The host address and port of the Etcd server
    host: 10.0.28.16:2379
# Patroni Bootstrap Configuration
bootstrap:
    # Configuration parameters for distributed configuration store (DCS)
    dcs:
        ttl: 30
        loop_wait: 10
```

```
24          retry_timeout: 10
25          maximum_lag_on_failover: 1048576
26          postgresql:
27              # Use pg_rewind during bootstrap
28              use_pg_rewind: true
29              #   Change pg_wal
30              create_replica_methods:
31                  - basebackup
32              basebackup:
33                  max-rate: '100M'
34                  waldir: "/srv/data/pg_wal"
35      # Initdb configuration
36      # Initdb configuration
37      initdb:
38          - auth: scram-sha-256
39          - encoding: UTF8
40          - data-checksums
41      # pg_hba.conf entries for replication and general access
42      pg_hba:
43          - host replication replicator 127.0.0.1/32 scram-sha-256
44          - host replication replicator 10.0.20.110/0 scram-sha-256
45          - host replication replicator 10.0.20.111/0 scram-sha-256
46          - host replication replicator 10.0.20.112/0 scram-sha-256
47          - host all all 0.0.0.0/0 scram-sha-256
48      # Adding default user admin with password admin
49      users:
50        admin:
51            password: admin
52            options:
53                - createrole
54                - createdb
55  # Patroni PostgreSQL Configuration
56  postgresql:
57      # PostgreSQL server listening address and port
58      listen: 10.0.20.111:5432
59      # Connect address for PostgreSQL clients
60      connect_address: 10.0.20.111:5432
61      # Data directory for PostgreSQL
62      data_dir: /var/lib/patroni
63      # Path to the pgpass file
64      pgpass: /tmp/pgpass
65
66      # Authentication configuration
67      authentication:
68          # Replication of user credentials
69          replication:
70              username: replicator
71              password: replicator
72          # Superuser credentials
73          superuser:
74              username: postgres
75              password: postgres
76      # Additional PostgreSQL parameters
77      parameters:
```

```
        # Directory for Unix socket
        unix_socket_directories: '.'
        # Password encryption method
        password_encryption: 'scram-sha-256'
# Patroni Tags Configuration
tags:
    # Prevents a node from being promoted in case of failure
    nofailover: false
    # Prevents the load balancer from considering this node
    noloadbalance: false
    # Prevents a replica from being created by cloning
    clonefrom: false
    # Prevents synchronous replication from being enforced
    nosync: false
```

Listing 60: Patroni - Konfiguration - sks1233

```
# Scope of PostgreSQL
scope: postgres
# Namespace for the PostgreSQL database
namespace: /db/
# Name of the PostgreSQL instance
name: postgres03
# Patroni REST API Configuration
restapi:
    # The IP address and port on which the REST API should listen
    listen: 10.0.20.112:8008

    # The IP address and port to which clients should connect
    connect_address: 10.0.20.112:8008
# Patroni Etcd Configuration
etcd3:
    # The host address and port of the Etcd server
    host: 10.0.28.16:2379
# Patroni Bootstrap Configuration
bootstrap:
    # Configuration parameters for distributed configuration store (DCS)
    dcs:
        ttl: 30
        loop_wait: 10
        retry_timeout: 10
        maximum_lag_on_failover: 1048576
        postgresql:
            # Use pg_rewind during bootstrap
            use_pg_rewind: true
            #   Change pg_wal
            create_replica_methods:
                - basebackup
            basebackup:
                max-rate: '100M'
                waldir: "/srv/data/pg_wal"
    # Initdb configuration
    initdb:
        - auth: scram-sha-256
```

```yaml
        - encoding: UTF8
        - data-checksums
    # pg_hba.conf entries for replication and general access
    pg_hba:
        - host replication replicator 127.0.0.1/32 scram-sha-256
        - host replication replicator 10.0.20.110/0 scram-sha-256
        - host replication replicator 10.0.20.111/0 scram-sha-256
        - host replication replicator 10.0.20.112/0 scram-sha-256
        - host all all 0.0.0.0/0 scram-sha-256
    # Adding default user admin with password admin
    users:
      admin:
          password: admin
          options:
              - createrole
              - createdb
# Patroni PostgreSQL Configuration
postgresql:
    # PostgreSQL server listening address and port
    listen: 10.0.20.112:5432
    # Connect address for PostgreSQL clients
    connect_address: 10.0.20.112:5432
    # Data directory for PostgreSQL
    data_dir: /var/lib/patroni
    # Path to the pgpass file
    pgpass: /tmp/pgpass
    # Authentication configuration
    authentication:
        # Replication of user credentials
        replication:
            username: replicator
            password: replicator
        # Superuser credentials
        superuser:
            username: postgres
            password: postgres
    # Additional PostgreSQL parameters
    parameters:
        unix_socket_directories: '.'
        # Password encryption method
        password_encryption: 'scram-sha-256'
# Patroni Tags Configuration
tags:
    # Prevents a node from being promoted in case of failure
    nofailover: false
    # Prevents the load balancer from considering this node
    noloadbalance: false
    # Prevents a replica from being created by cloning
    clonefrom: false
    # Prevents synchronous replication from being enforced
    nosync: false
```

Listing 61: Patroni - Konfiguration - sks1234

Sehr wichtig ist, dass das `pg_hba.conf`-File mitgegeben wird.

Ansonsten kann nicht auf die Datenbank zugegriffen werden.
Für den User `replication` müssen die IP-Adressen aller Nodes gesetzt werden.
Im Evaluations-Enviroment soll zudem der Zugriff von überall mit Passwort möglich sein.
Dies wird mit der letzten Zeile ermöglicht:

```
host replication replicator 127.0.0.1/32 scram-sha-256
host replication replicator 10.0.20.110/0 scram-sha-256
host replication replicator 10.0.20.111/0 scram-sha-256
host replication replicator 10.0.20.112/0 scram-sha-256
host all all 0.0.0.0/0 scram-sha-256
```

Listing 62: Patroni Bootstrap - pg_hba

Nun muss das Verzeichnis für PostgreSQL im Patroni-Verzeichnis erstellt und berechtigt werden:

```
mkdir -p /var/lib/patroni
chown -R postgres:postgres /var/lib/patroni
chmod 700 /var/lib/patroni
```

Listing 63: Patroni Bootstrap - Patroni-Verzeichnis

Jetzt können die Dienste auf den Patroni-Servern neu gestartet werden, Patroni sollte nun lauffähig sein:

```
systemctl start patroni
systemctl status patroni
patronictl -c /etc/patroni/config.yml list
```

Listing 64: Patroni Bootstrap - Neu starten

Wenn es lauffähig ist, muss noch aufgeräumt werden.
Der bestehende PostgreSQL Cluster muss disabled werden:

```
sudo systemctl disable --now postgresql
```

Listing 65: Patroni Bootstrap - Disable PostgreSQL

VII.IV.IV HAproxy

Final kann nun HAproxy installiert werden.
Zuerst müssen die Hosts hinterlegt werden:

```
#nano /etc/hosts
10.0.20.110    postgres01
10.0.20.111    postgres02
10.0.20.112    postgres03
```

Listing 66: HAproxy - Hostliste

Nun müssen die Repositories aktualisiert werden und HAproxy installiert werden:

```
apt update
apt install haproxy
```

Listing 67: HAproxy - Installation

Mit der Installation kommt ein Konfig-File mit.
Das soll gesichert werden:

```
mv /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.orig
```

Listing 68: HAproxy - Safe Alte Config

Nun muss die neue Konfiguration geschrieben werden:

```
#nano /etc/haproxy/haproxy.cfg
# Global configuration settings
global
    # Maximum connections globally
    maxconn 100
    # Logging settings
    log 127.0.0.1 local2

# Default settings
defaults
    # Global log configuration
    log global
    # Set mode to TCP
    mode tcp
    # Number of retries
    retries 2
    # Client timeout
    timeout client 30m
    # Connect timeout
    timeout connect 4s
    # Server timeout
    timeout server 30m
    # Check timeout
    timeout check 5s

# Stats configuration
listen stats
    # Set mode to HTTP
    mode http
    # Bind to port 8080
    bind *:8080
    # Enable stats
    stats enable
    # Stats URI
    stats uri /

# PostgreSQL configuration
listen postgres
    # Bind to port 5432
    bind *:5432
    # Enable HTTP check
    option httpchk
    # Expect status 200
    http-check expect status 200
    # Server settings
    default-server inter 3s fall 3 rise 2 on-marked-down shutdown-sessions
    # Define PostgreSQL servers
    server postgres01 10.0.20.110:5432 maxconn 100 check port 8008
    server postgres02 10.0.20.111:5432 maxconn 100 check port 8008
```

```
    server postgres03 10.0.20.112:5432 maxconn 100 check port 8008
```

Listing 69: HAproxy - Konfiguration

Nun kann HAproxy neu gestartet werden:

```
systemctl restart haproxy
systemctl status haproxy
```

Listing 70: HAproxy - Restart

Nun kann HAproxy geöffnet werden: `http://10.0.28.16:8080/`
Es sollte nun so aussehen:

Abbildung XXXIX: HAproxy - Web-GUI

VII.IV.V Rekonfiguration mit 250GiB Storage

Sobald die neue Disk im VMware vSphere erfasst wurde, muss die Disk auf den Servern gemountet werden:

```
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 71: Patroni - 250GiB Disk mount

Nun können die Verzeichnisse angelegt werden.
Nebst dem Verzeichnis für die Indizes und Daten braucht es auch ein neues `pg_wal`-Verzeichnis:

```
mkdir -p /srv/data/eval_index_tablespace
chown -R postgres:postgres /srv/data/eval_index_tablespace
chmod 700 /srv/data/eval_index_tablespace

mkdir -p /srv/data/eval_data_tablespace
chown -R postgres:postgres /srv/data/eval_data_tablespace
chmod 700 /srv/data/eval_data_tablespace

mkdir -p /srv/data/self_healing_test_index_tablespace
chown -R postgres:postgres /srv/data/self_healing_test_index_tablespace
chmod 700 /srv/data/self_healing_test_index_tablespace

mkdir -p /srv/data/self_healing_test_data_tablespace
chown -R postgres:postgres /srv/data/self_healing_test_data_tablespace
chmod 700 /srv/data/self_healing_test_data_tablespace

mkdir -p /srv/data/pg_wal
chown -R postgres:postgres /srv/data/pg_wal
chmod 700 /srv/data/pg_wal
```

Listing 72: Patroni - 250GiB Verzeichnisse

Um das `WAL`-Verzeichnis umzuhängen, muss erst die der Patroni-Cluster gestoppt werden:

```
patronictl -c /etc/patroni/config.yml pause
```

Listing 73: Patroni - 250GiB Cluster Pause

Wichtig ist, dass auch PostgreSQL gestoppt ist.
Da die PostgreSQL Binaries nun im Patroni-Verzeichnis liegen und auch dort das Socket-File läuft,
muss das Verzeichnis zwingend als Daten-Verzeichnisparameter (`-D`) angegeben werden,
sonst wird es zu einem Socket-Fehler kommen.
Zwingend ist zudem, das Command als `postgres`-User auszuführen:

```
sudo su - postgres
/usr/sbin/pg_ctl stop -D /var/lib/patroni/
```

Listing 74: Patroni - 250GiB PostgreSQL stoppen

Ganz wichtig ist nun, das ganze `pg_wal`-Verzeichnis zu verschieben.
So das die `WAL`-Files kopiert werden.
Da ein symlink gezogen wird, muss das bestehende Verzeichnis garantiert gelöscht sein:

```
mv /var/lib/patroni/pg_wal/* /srv/data/pg_wal
ln -s /srv/data/pg_wal /var/lib/patroni/pg_wal
ls -l /var/lib/patroni/pg_wal
```

Listing 75: Patroni - 250GiB move pg_wal

Die Berechtigung muss anschliessend wieder mit root-Rechten gesetzt werden:

```
chown -R postgres:postgres /var/lib/patroni/pg_wal
chmod 700 /var/lib/patroni/pg_wal
```

Listing 76: Patroni - 250GiB chmod - chown pg_wal

Jetzt muss erst PostgreSQL wieder gestartet werden.
Wenn alle Datenbanken auf allen Patroni-Nodes wieder lauffähig sind, kann der Cluster wieder aktiviert werden:

```
/usr/sbin/pg_ctl start -D /var/lib/patroni/
patronictl -c /etc/patroni/config.yml resume
```

Listing 77: Patroni - 250GiB PostgreSQL - Patroni resume

Zum Schluss sollte noch zwingend geprüft werden, ob der Cluster funktionsfähig ist:

```
patronictl -c /etc/patroni/config.yml list
```

Listing 78: Patroni - 250GiB Finaler Check

Wenn Patroni nun wieder einsatzfähig ist, müssen noch die optimierten Parameter gesetzt werden.
Folgende Parameter wurden gesetzt:

max_connections
Damit die 7000 Transaktionen sicher abgesetzt werden können.

superuser_reserved_connections
Mindestens 10 `postgres`-Verbindungen müssen gemacht werden können.

max_worker_processes
Damit genügend Worker vorhanden sind, um die `WAL`-Files abzuarbeiten, wurde deren Zahl auf 16 erhöht.

wal_log_hints
Stellt sicher, dass der gesamte Diskpage-inhalt ins `WAL` geschrieben wird.

max_wal_senders
Um den Replikations-Throughput zu erhöhen, wurden 16 Sender aktiviert

max_replication_slots
Damit die 16 Sender auch auf der Gegenseite abgearbeitet werden,
müssen die entsprechenden Slots bereitgestellt werden.

wal_keep_size
Damit die `WAL`-Files nicht zu gross werden und somit auch das Replication-Lag,
wurde die Grösse auf 1GiB reduziert.
Das zwingt den Primary dazu, die `WAL`-Files nur bis zu Maximal 1GiB aufzubewahren bevor sie archiviert werden.
Das ist wichtig, wenn Diskspace nicht im Überfluss vorhanden ist.

wal_level
Der Level wurde auf logical gesetzt, damit alle Informationen geschrieben werden.

wal_buffers
Der Buffer wurde auf 16MiB erhöht.

wal_writer_delay
Wenn der WAL-Writer einen Flush durchgeführt hat, geht er in ein Timeout.
Dies wurde auf 20ms heruntergesetzt, damit rasch wieder geschrieben wird.

wal_writer_flush_after
Definiert, ab welcher Grösse der Cache des WAL-Writers auf die Disk geschrieben wird.
Wurde auf 1MiB gesetzt um rasch auf die Disk schreiben zu können.

min_wal_size
Damit die `WAL`-Files nicht zu schnell repliziert werden und die Standby-Server aber auch der Primary-Server nicht überlastet werden, sollen die Files mindestens 1GiB gross werden bevor sie repliziert werden.

max_wal_size
Auf der anderen Seite soll trotzdem nicht zu lange gewartet werden, die maximale grösse wurde auf 5GiB begrenzt.

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

commit_siblings
Maximal 10 Transktionen sollen offen bleiben, bevor es zu einem Flush kommt.

checkpoint_timeout
Die maximale Zeit zwischen den Checkpoints soll 5min betragen.

archive_mode
Um Diskspace zu sparen sollen die `WAL`-Files nicht aktiviert werden.

checkpoint_completion_target
Ab 95% des Checkout-Timeouts soll mit dem Abschluss begonnen werden.

max_standby_archive_delay
Erst nach zehn Minuten sollen die Standby-Queries abgebrochen werden.
Ist notwendig, da es wegen der grossen Datenmenge und entsprechendem lag sonst schnell zum Abbruch kommen kann.

max_standby_streaming_delay
Beim normalen Standby soll die Zeit nur 3min sein.

wal_receiver_status_interval
Alle 60 sekunden soll der Status ausgetauscht werden.

max_logical_replication_workers
Acht logische Replication worker sollen zum einsatz kommen.

max_sync_workers_per_subscription
Es soll die maximale Anzahl an workern (8) für das parallelisieren verwendet werden.

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

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

work_mem
Der ganze PostgreSQL Cluster soll 12GiB Memory zur Verfügung haben.

temp_file_limit
Damit für das Erzeugen von Primary Keys, Foreign-Keys oder Indizes genügend Platz vorhanden ist, sollen temporäre Tabellen 200GiB Diskspace allokieren können.

vacuum_cost_delay
Der AUTOVACUUM-Job soll maximal 2ms ruhen.

vacuum_cost_limit
Der gesamte Prozess darf maximal zehn sekunden Schlafen.

bgwriter_delay
Der Hintergrundprozess soll nur 10ms ruhen.

bgwriter_lru_maxpages
Maximal dürfen 800 Buffers vom Hintergrundprozess beschrieben werden.

bgwriter_lru_multiplier
Insgesamt sollen beim nächsten Schreibzyklus 5 x soviel neue Writer erzeugt werden, wie im aktuellen Zyklus benötigt werden.

Entsprechend auch hier mittels `patronictl`:

```
patronictl -c /etc/patroni/config.yml edit-config --apply - --force <<'JSON'
{
 synchronous_mode: "on",
 synchronous_mode_strict: "on",
 synchronous_node_count: 2,
 "postgresql":
   {
   "parameters":{
     "synchronous_commit": "on",
     "synchronous_standby_names": "*",
     "max_connections": 1100,
     "superuser_reserved_connections":10,
     "max_worker_processes": 16,
     "wal_log_hints": "on",
     "max_wal_senders": 32,
     "max_replication_slots": 32,
     "wal_keep_size": "1GB",
     "wal_level": "logical",
     "wal_buffers": "16MB",
     "wal_writer_delay": "20ms",
     "wal_writer_flush_after": "1MB",
     "min_wal_size": "1GB",
     "max_wal_size": "5GB",
     "commit_delay": 20,
     "commit_siblings": 10,
     "checkpoint_timeout": "5min",
     "checkpoint_completion_target": 0.95,
     "archive_mode": "off",
     "max_standby_archive_delay": "10min",
     "max_standby_streaming_delay": "3min",
     "wal_receiver_status_interval": "1s",
     "hot_standby_feedback": "on",
     "wal_receiver_timeout": "60s",
     "max_logical_replication_workers": 8,
     "max_sync_workers_per_subscription": 8,
     "shared_buffers": "4GB",
     "maintenance_work_mem": "1GB",
```

```
    "work_mem": "12GB",
    "temp_file_limit": "200GB",
    "vacuum_cost_delay": "2ms",
    "vacuum_cost_limit": 10000,
    "bgwriter_delay": "10ms",
    "bgwriter_lru_maxpages": 800,
    "bgwriter_lru_multiplier": "5.0"
  }
 }
}
JSON
```

Listing 79: Patroni - 250GiB set Parameter

VII.IV.VI SQL Statements - Benchmarking

Für das Benchmarking wird die Tabelle `pgbench_eval_bench` erstellt.
Zudem wird je ein Tablespace für die Indizes (`eval_index_tablespace`) und Daten (`eval_data_tablespace`) erstellt.
Dies muss aber zweimal gemacht werden, einmal für die normalen Benchmarks und einmal mit den grossen Volumes:

```
--  Datenbank 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 owner postgres location '/var/lib/patroni/eval_index_tablespace';

--  Genereller Tablespace erstellen
drop tablespace if exists eval_data_tablespace;
CREATE TABLESPACE eval_data_tablespace owner postgres location '/var/lib/patroni/eval_data_tablespace';
```

Listing 80: Patroni - Benchmarking - DB erstellen

Wird auf die grossen Volumes gewechselt, müssen vorher die Tabellen bereinigt und gelöscht werden:

```
--  Daten löschen
truncate table pgbench_accounts;
truncate table pgbench_tellers;
truncate table pgbench_history;
truncate table pgbench_branches;

--  Tabellen löschen
drop table pgbench_accounts;
drop table pgbench_tellers;
drop table pgbench_history;
drop table pgbench_branches;
```

Listing 81: Patroni - Benchmarking - DB Cleanup

Anschliessend werden die neuen Tablespaces erzeugt:

```
--  Tablespace für Indices
drop tablespace if exists eval_index_tablespace;
CREATE TABLESPACE eval_index_tablespace owner postgres location '/srv/data/eval_index_tablespace';

```

```sql
--  Genereller Tablespace erstellen
drop tablespace if exists eval_data_tablespace;
CREATE TABLESPACE eval_data_tablespace owner postgres location '/srv/data/eval_data_tablespace';
```

Listing 82: Patroni - Benchmarking - Tablespaces erneut erstellen

VII.IV.VII SQL Statements - Testing

Entsprechend dem ERD, müssen die Tabellen erstellt werden: Evaluation - ERD self_healing_test Die Tabelle heisst entsprechend `self_healing_test`. Die Tests wurden allerdings erst gemacht, als auf die grossen Volumes gewechselt war. Das gesamte `CREATE`-Skript:

```sql
--  self-healing-Tabelle
drop table if exists self_healing_test;
create database self_healing_test;

--  Tablespace für Indices
drop tablespace if exists self_healing_indices_tablespace;
CREATE TABLESPACE self_healing_indices_tablespace owner postgres location '/srv/data/self_healing_test_index_tablespace';

--  Genereller Tablespace erstellen
drop tablespace if exists self_healing_datas_tablespace;
CREATE TABLESPACE self_healing_datas_tablespace owner postgres location '/srv/data/self_healing_test_data_tablespace';

--  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;

drop user if exists cso_1;
drop user if exists cso_2;
create user cso_1 with password 'cso1'role customer_service_officers;
create user cso_2 with password 'cso2' role customer_service_officers;

drop user if exists la_1;
drop user if exists la_2;
create user la_1 with password 'la1' role legal_affairs;
create user la_2 with password 'la2' role legal_affairs;

--  Schemas erstellen
drop schema if exists hrm;
create schema hrm authorization hrm;
drop schema if exists accountands;
create schema accountands authorization accountands;
```

```sql
drop schema if exists customer_service_officers;
create schema customer_service_officers authorization customer_service_officers;
drop schema if exists generell;
create schema generell;

--  GRANTS erstellen
grant all on all tables in schema hrm to legal_affairs;
grant all on all tables in schema accountands to legal_affairs;
grant all on all tables in schema customer_service_officers to legal_affairs;
grant all on all tables in schema generell to legal_affairs;
grant all on all tables in schema hrm to postgres;
grant all on all tables in schema accountands to postgres;
grant all on all tables in schema customer_service_officers to postgres;
grant all on all tables in schema generell to postgres;

 -- self_healing_accounts für Schema customer_service_officers
drop table if exists customer_service_officers.self_healing_accounts;
create table customer_service_officers.self_healing_accounts (
    account_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 '%@%')
) tablespace self_healing_datas_tablespace;
create unique index accounts_personal_mark on customer_service_officers.self_healing_accounts(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;

--  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 '%@%')
) tablespace self_healing_datas_tablespace;
create unique index employees_personal_mark on hrm.self_healing_employees(firstname, lastname, birthday) tablespace self_healing_indices_tablespace;

--  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
```

```sql
) tablespace self_healing_datas_tablespace;

--  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
) tablespace self_healing_datas_tablespace;

--  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
) tablespace self_healing_datas_tablespace;
create unique index intranet_unique_combi on generell.self_healing_intranet_users(intranet_user_id, employees_id) tablespace self_healing_indices_tablespace;
```

Listing 83: Patroni - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```sql
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 84: Patroni - 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');
```

```sql

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 85: Patroni - Self Healing Tests - Failover Data

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

```sql
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 86: Patroni - Self Healing Tests - Recovery Data

VII.V Stackgres mit Citus

VII.V.I Prerequisites

VII.V.I.I StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```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: 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 87: 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 88: StackGres-Citus - StorageClass / PersistentVolume aktivieren

VII.V.II Installation

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

Listing 89: StackGres-Citus - Namespace

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

```yaml
# -- 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: {}
  # Section to configure Operator ServiceAccount
  serviceAccount:
    # -- Section to configure Operator ServiceAccount annotations
    annotations: {}
    # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
    repoCredentials: []
  # Section to configure Operator Service
  service:
    # -- Section to configure Operator Service annotations
    annotations: {}

# Section to configure REST API Pod
restapi:
  # -- REST API Pod name
```

```yaml
  name: stackgres-restapi
  # Section to configure REST API image
  image:
    # -- REST API image name
    name: "stackgres/restapi"
    # -- REST API image tag
    tag: "1.9.0"
    # -- REST API image pull policy
    pullPolicy: "IfNotPresent"
  # -- REST API Pod annotations
  annotations: {}
  resources:
     requests:
       cpu: "1"
       memory: 1Gi
     limits:
       cpu: "1"
       memory: 1Gi
  # -- REST API Pod node selector
  nodeSelector: {}
  # -- REST API Pod tolerations. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#toleration-v1-core
  tolerations: []
  # -- REST API Pod affinity. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#affinity-v1-core
  affinity: {}
  # Section to configure REST API ServiceAccount
  serviceAccount:
    # -- REST API ServiceAccount annotations
    annotations: {}
    # -- Repositories credentials Secret names to attach to ServiceAccounts and Pods
    repoCredentials: []
  # Section to configure REST API Service
  service:
    # -- REST API Service annotations
    annotations: {}

# Section to configure Web Console container
adminui:
  # Section to configure Web Console image
  image:
    # -- Web Console image name
    name: "stackgres/admin-ui"
    # -- Web Console image tag
    tag: "1.9.0"
    # -- Web Console image pull policy
    pullPolicy: "IfNotPresent"
  resources:
     requests:
       cpu: "1"
       memory: 1Gi
     limits:
       cpu: "1"
       memory: 1Gi
  # Section to configure Web Console service.
  service:
```

```yaml
    # -- When set to ‘true‘ the HTTP port will be exposed in the Web Console Service
    exposeHTTP: true
    # -- The type used for the service of the UI:
    type: ClusterIP
    # -- (string) LoadBalancer will get created with the IP specified in
    loadBalancerIP:          # gramic, load-balancer-IP
    # -- (array) If specified and supported by the platform,
    loadBalancerSourceRanges:
    # -- (integer) The HTTPS port used to expose the Service on Kubernetes nodes
    nodePort:
    # -- (integer) The HTTP port used to expose the Service on Kubernetes nodes
    nodePortHTTP:

# Section to configure Operator Installation Jobs
jobs:
  # Section to configure Operator Installation Jobs image
  image:
    # -- Operator Installation Jobs image name
    name: "stackgres/jobs"
    # -- Operator Installation Jobs image tag
    tag: "1.9.0"
    # -- Operator Installation Jobs image pull policy
    pullPolicy: "IfNotPresent"
  # -- Operator Installation Jobs annotations
  annotations: {}
  # -- Operator Installation Jobs resources. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#resourcerequirements-v1-core
  resources: {}
  # -- Operator Installation Jobs node selector
  nodeSelector: {}
  # -- Operator Installation Jobs tolerations. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#toleration-v1-core
  tolerations: []
  # -- Operator Installation Jobs affinity. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#affinity-v1-core
  affinity: {}

# Section to configure deployment aspects.
deploy:
  # -- When set to ‘true‘ the Operator will be deployed.
  operator: true
  # -- When set to ‘true‘ the Web Console / REST API will be deployed.
  restapi: true

# Section to configure the Operator, REST API and Web Console certificates and JWT RSA key-pair.
cert:
  # -- If set to ‘true‘ the CertificateSigningRequest used to generate the certificate used by
  #   Webhooks will be approved by the Operator Installation Job.
  autoapprove: true
  # -- When set to ‘true‘ the Operator certificate will be created.
  createForOperator: true
  # -- When set to ‘true‘ the Web Console / REST API certificate will be created.
  createForWebApi: true
  # -- (string) The Secret name with the Operator Webhooks certificate issued by the Kubernetes cluster CA
  #   of type kubernetes.io/tls. See https://kubernetes.io/docs/concepts/configuration/secret/#tls-secrets
  secretName:
  # -- When set to ‘true‘ the Operator certificates will be regenerated if ‘createForOperator‘ is set to ‘true‘, and the certificate is expired or invalid.
```

```yaml
  regenerateCert: true
  # -- (integer) The duration in days of the generated certificate for the Operator after which it will expire and be regenerated.
  #   If not specified it will be set to 730 (2 years) by default.
  certDuration: 730
  # -- (string) The Secret name with the Web Console / REST API certificate
  #   of type kubernetes.io/tls. See https://kubernetes.io/docs/concepts/configuration/secret/#tls-secrets
  webSecretName:
  # -- 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.
  regenerateWebCert: true
  # -- 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.
  regenerateWebRsa: true
  # -- (integer) The duration in days of the generated certificate for the Web Console / REST API after which it will expire and be regenerated.
  #   If not specified it will be set to 730 (2 years) by default.
  webCertDuration:
  # -- (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.
  #   If not specified it will be set to 730 (2 years) by default.
  webRsaDuration:
  # -- (string) The private RSA key used to create the Operator Webhooks certificate issued by the
  #   Kubernetes cluster CA.
  key:
  # -- (string) The Operator Webhooks certificate issued by Kubernetes cluster CA.
  crt:
  # -- (string) The private RSA key used to generate JWTs used in REST API authentication.
  jwtRsaKey:
  # -- (string) The public RSA key used to verify JWTs used in REST API authentication.
  jwtRsaPub:
  # -- (string) The private RSA key used to create the Web Console / REST API certificate
  webKey:
  # -- (string) The Web Console / REST API certificate
  webCrt:
  # Section to configure cert-manager integration to generate Operator certificates
  certManager:
    # -- When set to ‘true‘ then Issuer and Certificate for Operator and Web Console / REST API
    #   Pods will be generated
    autoConfigure: false
    # -- The requested duration (i.e. lifetime) of the Certificates. See https://cert-manager.io/docs/reference/api-docs/#cert-manager.io%2fv1
    duration: "2160h"
    # -- 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
    renewBefore: "360h"
    # -- 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
    encoding: PKCS1
    # -- 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
    size: 2048

# Section to configure RBAC for Web Console admin user
rbac:
  # -- When set to ‘true‘ the admin user is assigned the ‘cluster-admin‘ ClusterRole by creating
  #   ClusterRoleBinding.
  create: true

# Section to configure Web Console authentication
```

```yaml
authentication:
  # -- Specify the authentication mechanism to use. By default is ‘jwt‘, see https://stackgres.io/doc/latest/api/rbac#local-secret-mechanism.
  #   If set to ‘oidc‘ then see https://stackgres.io/doc/latest/api/rbac/#openid-connect-provider-mechanism.
  type: jwt
  # -- (boolean) When ‘true‘ will create the secret used to store the ‘admin‘ user credentials to access the UI.
  createAdminSecret: true
  # -- The admin username that will be required to access the UI
  user: admin
  # -- (string) The admin password that will be required to access the UI
  #password: "TES2&Daggerfall"
  password:
  # Section to configure Web Console OIDC authentication
  oidc:
    # tlsVerification -- (string) Can be one of ‘required‘, ‘certificate-validation‘ or ‘none‘
# Section to configure Prometheus integration.
prometheus:

  allowAutobind: true
# Section to configure Grafana integration
grafana:
  # -- When set to ‘true‘ embed automatically Grafana into the Web Console by creating the
  #   StackGres dashboards and the read-only role used to read it from the Web Console
  autoEmbed: false
  # -- The schema to access Grafana. By default http. (used to embed manually and
  #   automatically grafana)
  schema: http
  # -- (string) The service host name to access grafana (used to embed manually and
  #   automatically Grafana).
  webHost:
  # -- The datasource name used to create the StackGres Dashboards into Grafana
  datasourceName: Prometheus
  # -- The username to access Grafana. By default admin. (used to embed automatically
  #   Grafana)
  user: admin
  # -- The password to access Grafana. By default prom-operator (the default in for
  #   kube-prometheus-stack helm chart). (used to embed automatically Grafana)
  password: prom-operator
  # -- Use follwing fields to indicate a secret where the grafana admin credentials are stored (replace user/password)

  # -- (string) The namespace of secret with credentials to access Grafana. (used to
  #   embed automatically Grafana, alternative to use ‘user‘ and ‘password‘)
  secretNamespace:
  # -- (string) The name of secret with credentials to access Grafana. (used to embed
  #   automatically Grafana, alternative to use ‘user‘ and ‘password‘)
  secretName:
  # -- (string) The key of secret with username used to access Grafana. (used to embed
  #   automatically Grafana, alternative to use ‘user‘ and ‘password‘)
  secretUserKey:
  # -- (string) The key of secret with password used to access Grafana. (used to
  #   embed automatically Grafana, alternative to use ‘user‘ and ‘password‘)
  secretPasswordKey:
  # -- (string) The ConfigMap name with the dashboard JSONs
  #   that will be created in Grafana. If not set the default
  #   StackGres dashboards will be created. (used to embed automatically Grafana)
```

```yaml
  dashboardConfigMap:
  # -- (array) The URLs of the PostgreSQL dashboards created in Grafana (used to embed manually
  urls:
  # Create and copy/paste grafana API token:
  # - Grafana > Configuration > API Keys > Add API key (for viewer) > Copy key value
  token:

# Section to configure extensions
extensions:
  repositoryUrls:
  - https://extensions.stackgres.io/postgres/repository?proxyUrl=http%3A%2F%2Fsproxy.sivc.first-it.ch%3A8080?skipHostnameVerification:true&setHttpScheme:true
  cache:
    # -- When set to `true` enable the extensions cache.
    enabled: false
    # -- An array of extensions pattern used to pre-loaded estensions into the extensions cache
    preloadedExtensions:
    - x86_64/linux/timescaledb-1\.7\.4-pg12
    # Section to configure the extensions cache PersistentVolume
    persistentVolume:
      size: 60Gi
      storageClass: "stackgres-storage"
    hostPath:
developer:
  version:
  # -- (string) Set `quarkus.log.level`. See https://quarkus.io/guides/logging#root-logger-configuration
  logLevel:
  # -- If set to `true` add extra debug to any script controlled by the reconciliation cycle of the operator configuration
  showDebug: false
  # -- 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
  showStackTraces: false
  # -- Only work with JVM version and allow connect
  # on port 8000 of operator Pod with jdb or similar
  enableJvmDebug: false
  # -- Only work with JVM version and if `enableJvmDebug` is `true`
  #   suspend the JVM until a debugger session is started
  enableJvmDebugSuspend: false
  # -- (string) Set the external Operator IP
  externalOperatorIp:
  # -- (integer) Set the external Operator port
  externalOperatorPort:
  # -- (string) Set the external REST API IP
  externalRestApiIp:
  # -- (integer) Set the external REST API port
  externalRestApiPort:
  # -- If set to `true` and `extensions.cache.enabled` is also `true`
  #   it will try to download extensions from images (experimental)
  allowPullExtensionsFromImageRepository: false
  # -- It set to `true` disable arbitrary user that is set for OpenShift clusters
  disableArbitraryUser: false
  # Section to define patches for some StackGres Pods
  patches:
    # Section to define volumes to be used by the operator container
    operator:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
```

```yaml
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
    # Section to define volumes to be used by the restapi container
    restapi:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
    # Section to define volumes to be used by the adminui container
    adminui:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
    # Section to define volumes to be used by the jobs container
    jobs:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
    # Section to define volumes to be used by the cluster controller container
    clusterController:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
    # Section to define volumes to be used by the distributedlogs controller container
    distributedlogsController:
      # -- Pod volumes. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volume-v1-core
      volumes: []
      # -- Pod's container volume mounts. See https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#volumemount-v1-core
      volumeMounts: []
```

Listing 90: StackGres-Citus - Helm Chart Manifest

Die Installation erfolgt dann wie folgt:

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

Listing 91: StackGres-Citus - Installation

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

```bash
# 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 92: StackGres-Citus - Post-Installation

Jetzt kann das GUI unter folgendem Link geöffnet werden: https://localhost:8443

☞ **Port Forwarding**
Wenn das Port Forwarding so gemacht wird, muss die CLI offenbleiben.
Sonst wird die Verbindung umgehend gekappt.
Daher empfiehlt es sich, mehrere Terminals offen zu halten.

Der Username ist `admin` aber das Passwort ist generisch.
Der Username liesse sich mit folgendem Command auslesen:

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

Listing 93: StackGres-Citus - System Username

Dieses lässt sich mit folgendem Command auslesen:

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

Listing 94: StackGres-Citus - System Passwort

Am Schluss sollte das Passwort aber noch gesäubert werden:

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

Listing 95: StackGres-Citus - System Passwort Cleanup

VII.V.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 96: 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 97: 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 98: StackGres-Citus - Benchmarking - Instanz-Profil Deploy

Auf ein Deployment einer `SGPostgresConfig` wurde bei den ersten drei Benchamrks 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 99: StackGres-Citus - Benchmarking - SGShardedCluster

Der Deploy:

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

Listing 100: StackGres-Citus - Benchmark - Cluster Deploy

Damit nun aber eine Verbindung auf die DB (IP 10.0.20.106) gemacht werden kann, muss das Passwort ausgelesen werden:

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

Listing 101: StackGres-Citus - Benchmark DB Passwort

VII.V.IV Deployment - Self Healing Tests

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

```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 102: 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 103: 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 104: StackGres-Citus - Self Healing Testing - Instanz-Profil Deploy

Auch beim Self Healing Testing wurde auf ein Deployment einer `SGPostgresConfig` verzichtet.
Dafür wurden drei Coordinator-Instanzen und drei Shard-Instanzen pro Cluster deklariert.
Das Manifest für den Benchmark-Cluster sieht entsprechend den vorgaben wie folgt aus:

```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
```

```yaml
    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 105: StackGres-Citus - Self Healing Testing - SGShardedCluster

Der Deploy:

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

Listing 106: StackGres-Citus - Self Healing Testing - Cluster Deploy

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

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

Listing 107: StackGres-Citus - Self Healing Testing DB Passwort

VII.V.V Rekonfiguration mit 250GiB Storage

VII.V.V.I Bereinigen

Zuerst wurde auch hier der komplette Namespace bereinigt, wobei es zwingend ist, vorher im GUI die Cluster zu löschen und mittels CLI den Operator zu deinstallieren.
Andernfalls wird nicht alles sauber entfernt was zu schwer Nachvollziehbaren Sideeffects führt:

```bash
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 108: StackGres-Citus - Deinstallieren

VII.V.V.II StorageClass setzen

```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: 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 109: StackGres-Citus - StorageClass setzen

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

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

VII.V.V.III Installation

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

Listing 111: 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 112: StackGres-Citus - Installation 250GiB

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

Die Ressourcen für die Shards mussten erhöht werden:

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

Listing 113: StackGres-Citus - Benchmarking - SGInstanceProfile Coordinator 250GiB

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

Listing 114: 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 115: StackGres-Citus - Benchmarking - Instanz-Profil Deploy 250GiB

Nun wurde ein `SGPostgresConfig` erstellt.
Die Settings entsprechen der Patroni Konfiguration.
Bei StackGres gibt es allerdings einige Parameter, die für die Konfiguration gesperrt sind.
In diesem Fall waren es folgende:

- wal_log_hints
- wal_level
- archive_mode

Das Manifest sieht folgendermassen aus:

```yaml
#Not needed actualy
apiVersion: stackgres.io/v1
kind: SGPostgresConfig
metadata:
  namespace: sg-platform
  name: sg-evaluation-250gib
spec:
  postgresVersion: '16'
  postgresql.conf:
    random_page_cost: '1.5'
    password_encryption: 'scram-sha-256'
    log_checkpoints: 'on'
```

```yaml
    max_connections: '1100'
    superuser_reserved_connections: '10'
    max_worker_processes: '16'
#    wal_log_hints: 'on'
    max_wal_senders: '32'
    max_replication_slots: '32'
    wal_keep_size: '1GB'
#    wal_level: 'logical'
    wal_buffers: '16MB'
    wal_writer_delay: '20ms'
    wal_writer_flush_after: '1MB'
    min_wal_size: '1GB'
    max_wal_size: '5GB'
    commit_delay: '20'
    commit_siblings: '10'
    checkpoint_timeout: '5min'
    checkpoint_completion_target: '0.95'
#    archive_mode: 'off'
    max_standby_archive_delay: '10min'
    max_standby_streaming_delay: '3min'
    wal_receiver_status_interval: '1s'
    hot_standby_feedback: 'on'
    wal_receiver_timeout: '60s'
    max_logical_replication_workers: '8'
    max_sync_workers_per_subscription: '8'
    shared_buffers: '4GB'
    maintenance_work_mem: '1GB'
    work_mem: '12GB'
    temp_file_limit: '200GB'
    vacuum_cost_delay: '2ms'
    vacuum_cost_limit: '10000'
    bgwriter_delay: '10ms'
    bgwriter_lru_maxpages: '800'
    bgwriter_lru_multiplier: '5.0'
```

Listing 116: StackGres-Citus - Benchmarking - SGPostgresConfig

Der Deploy:

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

Listing 117: StackGres-Citus - Benchmark - Deploy SGPostgresConfig

Das Manifest für den Benchmark-Cluster wurde die Shard-Cluster Anzahl auf 2 reduziert und die PVC entsprechend dimensioniert.
Zudem wurde das `SGPostgresConfig`-Profil `sgPostgresConfig` eingebunden:

```yaml
apiVersion: stackgres.io/v1alpha1
kind: SGShardedCluster
metadata:
  name: sg-pgbench
  namespace: sg-platform
spec:
  type: citus
  database: pgbench_eval_bench
  postgres:
    version: '16'
```

```yaml
  coordinator:
    instances: 1
    pods:
      persistentVolume:
        size: '230Gi'
        storageClass: "stackgres-storage"
      disableConnectionPooling: true   # gramic, 19.04.2024: create_distributed_table auf pgbench_accounts älsst sich nicht üausfhren wegen pgbouncer problemen
    sgInstanceProfile: "sg-pgbench-coordinator"
    configurations:
      sgPostgresConfig: "sg-evaluation-250gib"
  shards:
    clusters: 2 # gramic, 19.04.2024: 250GiB Tabelle
    instancesPerCluster: 1
    pods:
      persistentVolume:
        size: '230Gi'
        storageClass: "stackgres-storage"
    sgInstanceProfile: "sg-pgbench-shard"
    configurations:
      sgPostgresConfig: "sg-evaluation-250gib"
  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 118: StackGres-Citus - Benchmarking - SGShardedCluster 250GiB

Der Deploy:

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

Listing 119: StackGres-Citus - Benchmark - Cluster Deploy 250GiB

VII.V.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.

VII.V.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:

```sql
--  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;

drop user if exists cso_1;
drop user if exists cso_2;
create user cso_1 with password 'cso1'role customer_service_officers;
create user cso_2 with password 'cso2' role customer_service_officers;

drop user if exists la_1;
drop user if exists la_2;
create user la_1 with password 'la1' role legal_affairs;
create user la_2 with password 'la2' role legal_affairs;

--  Schemas erstellen
drop schema if exists hrm;
create schema hrm authorization hrm;
drop schema if exists accountands;
create schema accountands authorization accountands;
drop schema if exists customer_service_officers;
create schema customer_service_officers authorization customer_service_officers;
drop schema if exists generell;
create schema generell;

--  GRANTS erstellen
grant all on all tables in schema hrm to legal_affairs;
grant all on all tables in schema accountands to legal_affairs;
grant all on all tables in schema customer_service_officers to legal_affairs;
grant all on all tables in schema generell to legal_affairs;
grant all on all tables in schema hrm to postgres;
grant all on all tables in schema accountands to postgres;
grant all on all tables in schema customer_service_officers to postgres;
grant all on all tables in schema generell to postgres;

```

```sql
 -- self_healing_accounts für Schema customer_service_officers
drop table if exists customer_service_officers.self_healing_accounts;
create table customer_service_officers.self_healing_accounts (
    account_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 accounts_personal_mark on customer_service_officers.self_healing_accounts(firstname, lastname, birthday);
SELECT create_reference_table('customer_service_officers.self_healing_accounts');

--  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;
```

```sql
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 120: StackGres-Citus - Self Healing Tests - CREATE-SQL

Es sollen aber auch gleich Daten initial geschrieben werden:

```sql
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 121: StackGres-Citus - 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');
```

```sql

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 122: StackGres-Citus - Self Healing Tests - Failover Data

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

```sql
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 123: StackGres-Citus - Self Healing Tests - Recovery Data

VIII Evaluationssysteme - Benchmarking

VIII.I yugabyteDB

Pro Lauf werden erst die Daten initialisiert werden.
Wichtig ist dabei, dass die beiden Tablespaces `eval_index_tablespace` und `eval_data_tablespace` mitgeben werden.
Anschliessend werden die Benchmarks an sich ausgeführt.
Die Resultate werden weggeschrieben.

```bash
###################
#   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

# Benchmarking dql
./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
./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
./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
./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

###################
#   3. Lauf       #
#   ca. 50GiB     #
###################
./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

# Benchmarking mixed
./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
./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
./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
./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

# Benchmarking dql
./ysql_bench -h 10.0.20.106 -p 5433 -c 100 -C -j 4 -v -t 50 -S -U yadmin pgbench_eval_bench > /home/gramic/3_1_yugabytedb_dql_benchmark.txt
./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
./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
./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


###################
#   4. Lauf       #
#   ca. 250GiB    #
###################
./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

# 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 124: 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 125: yugabyteDB - Benchmarking - Table Size SQL

VIII.II Patroni

Pro Lauf werden erst die Daten initialisiert werden.
Wichtig ist dabei, dass die beiden Tablespaces `eval_index_tablespace` und `eval_data_tablespace` mitgeben werden.
Anschliessend werden die Benchmarks an sich ausgeführt.
Die Resultate werden weggeschrieben.

```
###################
#   1. Lauf       #
#   ca. 5GiB      #
###################
# Init
pgbench --host=10.0.28.16 --port=5432 --initialize --scale=400 --foreign-keys --fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace=
    eval_data_tablespace --username=postgres pgbench_eval_bench

# Benchmarking mixed
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_1_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_2_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_3_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_4_patroni_mixed_benchmark.txt

# Benchmarking dql
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_1_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_2_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_3_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_4_patroni_dql_benchmark.txt

###################
#   2. Lauf       #
```

```
#   ca. 15GiB     #
###################
# Init
pgbench --host=10.0.28.16 --port=5432 --initialize --scale=1200 --foreign-keys --fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace=
    eval_data_tablespace --username=postgres pgbench_eval_bench

# Benchmarking mixed
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_1_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_2_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_3_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_4_patroni_mixed_benchmark.txt
# Benchmarking dql
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_1_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_2_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_3_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_4_patroni_dql_benchmark.txt

###################
#   3. Lauf       #
#   ca. 50GiB     #
###################
pgbench --host=10.0.28.16 --port=5432 --initialize --scale=3999 --foreign-keys --fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace=
    eval_data_tablespace --username=postgres pgbench_eval_bench

# Benchmarking mixed
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_1_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_2_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_3_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_4_patroni_mixed_benchmark.txt
# Benchmarking dql
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_1_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_2_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_3_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_4_patroni_dql_benchmark.txt


###################
#   4. Lauf       #
#   ca. 250GiB    #
###################
pgbench --host=10.0.28.16 --port=5432 --initialize --scale=16784 --foreign-keys --fillfactor=100 --username=dtgvpf --index-tablespace=eval_index_tablespace --tablespace=
    eval_data_tablespace --username=postgres pgbench_eval_bench

# Benchmarking mixed
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_1_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_2_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_3_patroni_mixed_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_4_patroni_mixed_benchmark.txt
# Benchmarking dql
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_1_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_2_patroni_dql_benchmark.txt
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_3_patroni_dql_benchmark.txt
```

```
pgbench -h 10.0.28.16 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_4_patroni_dql_benchmark.txt
```

Listing 126: Patroni - Benchmarking-Commands

Die grösse der Tabellen lässt sich wie folgt auslesen:

```
SELECT pg_size_pretty( pg_database_size('pgbench_eval_bench') );
```

Listing 127: Patroni - Benchmarking - Table Size SQL

VIII.III 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
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_2_stackgresmixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_3_stackgresmixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -U postgres pgbench_eval_bench > /home/itgramic/1_4_stackgresmixed_benchmark.txt

# 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
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_2_stackgresdql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_3_stackgresdql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 10 -C -j 4 -v -t 10 -S -U postgres pgbench_eval_bench > /home/itgramic/1_4_stackgresdql_benchmark.txt

###################
#   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
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_2_stackgres_mixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_3_stackgres_mixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -U postgres pgbench_eval_bench > /home/itgramic/2_4_stackgres_mixed_benchmark.txt
# 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
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_2_stackgres_dql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_3_stackgres_dql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 50 -S -U postgres pgbench_eval_bench > /home/itgramic/2_4_stackgres_dql_benchmark.txt
```

```

###################
#   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
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_2_stackgres_mixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_3_stackgres_mixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -U postgres pgbench_eval_bench > /home/itgramic/3_4_stackgres_mixed_benchmark.txt
# 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
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_2_stackgres_dql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_3_stackgres_dql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 50 -C -j 4 -v -t 100 -S -U postgres pgbench_eval_bench > /home/itgramic/3_4_stackgres_dql_benchmark.txt


###################
#   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
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_2_stackgresmixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_3_stackgresmixed_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -U postgres pgbench_eval_bench > /home/itgramic/4_4_stackgresmixed_benchmark.txt
# 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
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_2_stackgresdql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_3_stackgresdql_benchmark.txt
pgbench -h 10.0.20.106 -p 5432 -c 25 -C -j 4 -v -t 280 -S -U postgres pgbench_eval_bench > /home/itgramic/4_4_stackgresdql_benchmark.txt
```

Listing 128: StackGres-Citus - Benchmarking-Commands

Die grösse der Tabellen lässt sich wie folgt auslesen:

```
SELECT * FROM citus_tables;
```

Listing 129: StackGres-Citus - Benchmarking - Table Size SQL

Das Resultat ist aber an sich zu gross, da bei den Reference Tables alle Shards und die Coordinators zusammengerechnet werden.
Die korrekte Grösse muss wie folgt kalkuliert werden:

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

IX Evaluationssysteme - Testing

IX.I StackGres - Citus

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

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     11%↑
MEM:     22%

<c>      Cordon     <u> Uncordon
<ctrl-d> Delete     <y> YAML
<d>      Describe
<r>      Drain
<e>      Edit
<?>      Help

Nodes(all)[3]
NAME↑    STATUS    ROLE                      TAINTS  VERSION          PODS  CPU   MEM   %CPU  %MEM  CPU/A  MEM/A  AGE
sks1183  Ready     control-plane,etcd,master 0       v1.29.0+rke2r1   25    3105  5685  25    35    12000  15958  82d
sks1184  NotReady  worker                    3       v1.29.0+rke2r1   14    0     0     0     0     12000  15958  82d
sks1185  Ready     worker                    0       v1.29.0+rke2r1   15    934   5310  7     33    12000  15958  82d
```

Abbildung XL: StackGres Testing - Node sks1184 down

Entsprechend wurden die Pods ebenfalls auf `terminating` gesetzt:

```
gramic@cks4040: ~
Context: default
Cluster: default
User:    default
K9s Rev: v0.31.8 ⚡v0.32.4
K8s Rev: v1.29.0+rke2r1
CPU:     8%
MEM:     22%

<0> all                  <6> default  <a>      Attach     <l>      Logs            <f> Show PortForward
<1> sg-platform                       <ctrl-d> Delete     <p>      Logs Previous   <t> Transfer
<2> kube-public                       <d>      Describe   <shift-f> Port-Forward    <y> YAML
<3> local-path-storage                <e>      Edit       <z>      Sanitize
<4> metallb-system                    <?>      Help       <s>      Shell
<5> kube-system                       <ctrl-k> Kill       <o>      Show Node

Pods(sg-platform)[15]
NAME↑                                  PF  READY  STATUS       RESTARTS  CPU  MEM  %CPU/R  %CPU/L  %MEM/R  %MEM/L  IP            NODE     AGE
sg-healing-test-coord-0                ●   6/6    Terminating  0         0    0    0       n/a     0       n/a     198.18.1.60   sks1184  3h8m
sg-healing-test-coord-1                ●   6/6    Running      7         25   363  2       n/a     17      n/a     198.18.0.59   sks1183  3h7m
sg-healing-test-coord-2                ●   6/6    Running      0         305  394  30      n/a     19      n/a     198.18.2.90   sks1185  3h7m
sg-healing-test-shard0-0               ●   6/6    Running      7         444  359  22      n/a     17      n/a     198.18.0.106  sks1183  3h8m
sg-healing-test-shard0-1               ●   6/6    Terminating  0         0    0    0       n/a     0       n/a     198.18.1.213  sks1184  3h7m
sg-healing-test-shard0-2               ●   6/6    Running      0         378  414  18      n/a     20      n/a     198.18.2.3    sks1185  3h7m
sg-healing-test-shard1-0               ●   6/6    Terminating  0         0    0    0       n/a     0       n/a     198.18.1.227  sks1184  3h7m
sg-healing-test-shard1-1               ●   6/6    Running      7         574  348  28      n/a     17      n/a     198.18.0.163  sks1183  3h7m
sg-healing-test-shard1-2               ●   6/6    Running      0         38   421  1       n/a     20      n/a     198.18.2.253  sks1185  3h7m
sg-healing-test-shard2-0               ●   6/6    Running      0         29   445  1       n/a     21      n/a     198.18.2.31   sks1185  3h7m
sg-healing-test-shard2-1               ●   6/6    Running      7         251  349  12      n/a     17      n/a     198.18.0.186  sks1183  3h7m
sg-healing-test-shard2-2               ●   6/6    Terminating  0         0    0    0       n/a     0       n/a     198.18.1.105  sks1184  3h7m
stackgres-operator-845d6b7c47-6tcqb    ●   1/1    Running      0         44   214  4       4       20      20      198.18.0.18   sks1183  3m56s
stackgres-operator-845d6b7c47-prk6k    ●   1/1    Terminating  1         0    0    0       0       0       0       198.18.1.165  sks1184  3h11m
stackgres-restapi-685c4688b5-pks79     ●   2/2    Running      0         44   401  2       2       19      19      198.18.2.243  sks1185  3h11m
```

Abbildung XLI: StackGres Testing - Pods Down

Der Patroni-Leader des Coordinators aber auch die der Shards wurden einem `Failover` ausgeführt:

```
gramic@cks4040:~$ kubectl exec -it sg-healing-test-coord-0 -n sg-platform -c patroni -- patronictl list
+ Citus cluster: sg-healing-test --+--------------------+--------------+-----------+----+-----------+
| Group | Member                    | Host               | Role         | State     | TL | Lag in MB |
+-------+---------------------------+--------------------+--------------+-----------+----+-----------+
|     0 | sg-healing-test-coord-0   | 198.18.1.60:7433   | Leader       | running   |  2 |           |
|     0 | sg-healing-test-coord-1   | 198.18.0.59:7433   | Sync Standby | streaming |  2 |         0 |
|     0 | sg-healing-test-coord-2   | 198.18.2.90:7433   | Sync Standby | streaming |  2 |         0 |
|     1 | sg-healing-test-shard0-0  | 198.18.0.106:7433  | Replica      | streaming |  2 |         0 |
|     1 | sg-healing-test-shard0-1  | 198.18.1.213:7433  | Leader       | running   |  2 |           |
|     1 | sg-healing-test-shard0-2  | 198.18.2.3:7433    | Replica      | streaming |  2 |         0 |
|     2 | sg-healing-test-shard1-0  | 198.18.1.227:7433  | Replica      | streaming |  2 |         0 |
|     2 | sg-healing-test-shard1-1  | 198.18.0.163:7433  | Replica      | streaming |  2 |         0 |
|     2 | sg-healing-test-shard1-2  | 198.18.2.253:7433  | Leader       | running   |  2 |           |
|     3 | sg-healing-test-shard2-0  | 198.18.2.31:7433   | Replica      | streaming |  2 |         0 |
|     3 | sg-healing-test-shard2-1  | 198.18.0.186:7433  | Replica      | streaming |  2 |         0 |
|     3 | sg-healing-test-shard2-2  | 198.18.1.105:7433  | Leader       | running   |  2 |           |
+-------+---------------------------+--------------------+--------------+-----------+----+-----------+
gramic@cks4040:~$ kubectl exec -it sg-healing-test-coord-0 -n sg-platform -c patroni -- patronictl list
Error from server: error dialing backend: proxy error from 127.0.0.1:9345 while dialing 10.0.20.104:10250, code 502: 502 Bad Gateway
gramic@cks4040:~$ kubectl exec -it sg-healing-test-coord-1 -n sg-platform -c patroni -- patronictl list
+ Citus cluster: sg-healing-test --+--------------------+--------------+-----------+----+-----------+
| Group | Member                    | Host               | Role         | State     | TL | Lag in MB |
+-------+---------------------------+--------------------+--------------+-----------+----+-----------+
|     0 | sg-healing-test-coord-0   | 198.18.1.60:7433   | Replica      | running   |  2 |         0 |
|     0 | sg-healing-test-coord-1   | 198.18.0.59:7433   | Sync Standby | streaming |  3 |         0 |
|     0 | sg-healing-test-coord-2   | 198.18.2.90:7433   | Leader       | running   |  3 |           |
|     1 | sg-healing-test-shard0-0  | 198.18.0.106:7433  | Replica      | streaming |  3 |         0 |
|     1 | sg-healing-test-shard0-1  | 198.18.1.213:7433  | Replica      | running   |  2 |         0 |
|     1 | sg-healing-test-shard0-2  | 198.18.2.3:7433    | Leader       | running   |  3 |           |
|     2 | sg-healing-test-shard1-0  | 198.18.1.227:7433  | Replica      | streaming |  2 |         0 |
|     2 | sg-healing-test-shard1-1  | 198.18.0.163:7433  | Replica      | streaming |  2 |         0 |
|     2 | sg-healing-test-shard1-2  | 198.18.2.253:7433  | Leader       | running   |  2 |           |
|     3 | sg-healing-test-shard2-0  | 198.18.2.31:7433   | Leader       | running   |  3 |           |
|     3 | sg-healing-test-shard2-1  | 198.18.0.186:7433  | Replica      | streaming |  3 |         0 |
|     3 | sg-healing-test-shard2-2  | 198.18.1.105:7433  | Replica      | running   |  2 |         0 |
+-------+---------------------------+--------------------+--------------+-----------+----+-----------+
```

Abbildung XLII: StackGres Testing - Patroni Übersicht

Während dieser Zeit, ist die DB immer erreichbar:

```
184  -- After rebuild
185  select * from customer_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 XLIII: 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 XLIV: StackGres Testing - Connection Timeout

IX.II YugabyteDB

Zum einen, kann der Fehler irgendwann auftreten.
In diesem Fall wird erst im Log die Fehlermeldung geworfen, dass die Zeitdifferenz zu gross ist:

```
gramic@cks4040: ~

Context: default ✏️                <0> tail    <6> 1h    <shift-c> Clear              <t> Toggle Timestamp
Cluster: default                    <1> head              <c>       Copy               <w> Toggle Wrap
User:    default                    <2> 1m                <m>       Mark
K9s Rev: v0.31.8 ⚡v0.32.4          <3> 5m                <ctrl-s>  Save
K8s Rev: v1.29.0+rke2r1             <4> 15m               <s>       Toggle AutoScroll
CPU:     1%                         <5> 30m               <f>       Toggle FullScreen
MEM:     36%

Logs(yb-platform/yb-tserver-1)[tail]
Autoscroll:On    FullScreen:Off    Timestamps:Off    Wrap:Off
yb-cleanup Permitted disk usage for core dump files in kb: 25691590
yb-cleanup Disk usage by core dump files in kb: 4
yb-cleanup Permitted disk usage for yb-tserver*log.* files in kb: 5000000
yb-cleanup Disk usage by yb-tserver*log.* files in kb: 126952
yb-cleanup Permitted disk usage for postgres*log* files in kb: 100000
yb-cleanup Disk usage by postgres*log* files in kb: 9235
yb-cleanup Permitted disk usage for core dump files in kb: 25691590
yb-cleanup Disk usage by core dump files in kb: 4
yb-cleanup Permitted disk usage for yb-tserver*log.* files in kb: 5000000
yb-cleanup Disk usage by yb-tserver*log.* files in kb: 126953
yb-tserver bash: chronyc: command not found
yb-cleanup Permitted disk usage for postgres*log* files in kb: 100000
yb-cleanup Disk usage by postgres*log* files in kb: 9314
yb-tserver bash: chronyc: command not found
yb-tserver Fatal failure details written to /mnt/disk0/yb-data/tserver/logs/yb-tserver.FATAL.details.2024-04-16T18_23_18.pid28.txt
yb-tserver F20240416 18:23:18 ../../src/yb/server/hybrid_clock.cc:177] Too big clock skew is detected: 0.506s, while max allowed is: 0.500s; clock_skew_force_crash_bound_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     @     0x55e8dddd5d6e  _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 XLV: 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 XLVI: YugabyteDB - Tablet Leader - No Lease

Als nächstes wird der komplette `tserver` in einem `CrashLoopBackOff` fallen:

Abbildung XLVII: YugabyteDB - CrashLoopBackOff

Der ganze Cluster an sich aber bleibt Arbeitsfähig.

Anders sieht es aus, wenn auch `tmaster`-Nodes von Start weg betroffen sind.
Es werden aber primär nur die Logs überall geschrieben:

```
gramic@cks4040: ~
Context: default ✏️                <0> tail   <6> 1h   <shift-c> Clear          <t> Toggle Timestamp
Cluster: default                    <1> head            <c>       Copy           <w> Toggle Wrap
User:    default                    <2> 1m              <m>       Mark
K9s Rev: v0.31.8 ⚡v0.32.4          <3> 5m              <ctrl-s>  Save
K8s Rev: v1.29.0+rke2r1             <4> 15m             <s>       Toggle AutoScroll
CPU:     1%                         <5> 30m             <f>       Toggle FullScreen
MEM:     22%
                                                  Logs(yb-platform/yb-master-0)[tail]
                                                Autoscroll:On   FullScreen:Off   Timestamps:Off   Wrap:Off
yb-master E0423 12:16:45.676072   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.550s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:46.785388   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.909s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:47.802490   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.708s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:49.147837   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.841s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:50.516947   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.875s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:51.640313   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.517s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:52.773218   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.550s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:53.887251   178 hybrid_clock.cc:181] Too big clock skew is detected: 1.006s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:55.343684   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.932s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:56.360834   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.948s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:57.771939   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.665s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:16:59.056317   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.954s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:00.150394   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:01.252478   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.718s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:02.274430   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.789s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:03.282796   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.895s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:04.575225   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.770s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:06.040117   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.964s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:07.525339   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.512s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:08.979657   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.983s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:10.274869   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.726s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:11.373963   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.682s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:12.413090   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.760s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:13.464146   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.618s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:14.517447   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.782s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:15.871708   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.638s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:17.392766   178 hybrid_clock.cc:181] Too big clock skew is detected: 1.000s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:18.841015   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.661s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:19.903977   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.629s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:21.329376   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.832s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:22.704488   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.669s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:23.861863   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.961s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:24.931900   105 hybrid_clock.cc:181] Too big clock skew is detected: 0.827s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:26.376091   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.785s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:27.492267   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.568s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:28.507580   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.675s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:29.576633   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.558s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:30.750717   178 hybrid_clock.cc:181] Too big clock skew is detected: 1.039s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:31.845826   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.857s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-master E0423 12:17:32.951920   178 hybrid_clock.cc:181] Too big clock skew is detected: 0.864s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
```

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

```
gramic@cks4040: ~
gramic@cks4040: ~                                                    gramic@cks4040: ~
Context: default ✏️                <0> tail   <6> 1h   <shift-c> Clear          <t> Toggle Timestamp
Cluster: default                    <1> head           <c>       Copy           <w> Toggle Wrap
User:    default                    <2> 1m             <m>       Mark
K9s Rev: v0.31.8 ⚡v0.32.4          <3> 5m             <ctrl-s>  Save
K8s Rev: v1.29.0+rke2r1             <4> 15m            <s>       Toggle AutoScroll
CPU:     0%                         <5> 30m            <f>       Toggle FullScreen
MEM:     22%
                                                  Logs(yb-platform/yb-tserver-1)[tail]
                                          Autoscroll:On     FullScreen:Off     Timestamps:Off     Wrap:Off
yb-tserver E0423 11:44:56.039235   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:44:57.039614   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.798s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:44:58.039772   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.779s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:44:59.039928   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.718s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:00.040057   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.781s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:01.040220   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.672s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:02.040351   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.777s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:03.040545   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.759s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:04.040710   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.759s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:05.040827   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.810s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:06.040973   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.810s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:07.041210   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.800s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:08.041371   139 hybrid_clock.cc:181] Too big clock skew is detected: 0.786s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:09.041580   139 hybrid_clock.cc:181] Too big clock skew is detected: 0.799s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:10.041759   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.811s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:11.041956   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.794s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:12.042100   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.815s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:13.042212   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.818s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:14.042308   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.708s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:15.042454   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.776s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:16.042851   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.807s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:17.043002   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.804s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:17.043002   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.804s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:18.043179   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.803s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:19.043371   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.791s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:20.043519   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.652s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:21.043641   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.717s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:22.043789   139 hybrid_clock.cc:181] Too big clock skew is detected: 0.820s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:23.043915   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.745s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:24.044063   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.822s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:25.044202   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.824s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:26.044380   142 hybrid_clock.cc:181] Too big clock skew is detected: 0.827s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:27.044510   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.823s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:28.044632   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.792s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:29.044777   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.806s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:30.044929   141 hybrid_clock.cc:181] Too big clock skew is detected: 0.818s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:31.045047   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.829s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:32.045161   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.773s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:33.045315   140 hybrid_clock.cc:181] Too big clock skew is detected: 0.822s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
yb-tserver E0423 11:45:34.045454   139 hybrid_clock.cc:181] Too big clock skew is detected: 0.781s, while max allowed is: 0.500s; clock_skew_force_crash_bound_usec=60000000
```

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

YugabyteDB erlaubt in so einem Fall keine Zugriffe mehr auf den Cluster.
So wird verhindert, dass der Cluster korrumpiert wird.

X Testsystem - Installation

X..I Ansible - Installation

Auf `sks1000` wurde zuerst Ansible installiert:

```
sudo apt update && sudo apt install -y python3-pip sshpass git
pip3 install ansible

```

Listing 130: Ansible - Installation

Das GitHub Repository von `vitabacks / postgresql_cluster` muss lokal geklont werden:

```
git clone https://github.com/vitabaks/postgresql_cluster.git

```

Listing 131: Ansible - Repository Clone

Das Verzeichnis ist `postgresql_cluster` beinhaltet die Playbooks und Ressourcen.

```
cd postgresql_cluster/

```

Listing 132: Ansible - cd Repository

Die wichtigen Files liegen nun an folgenden Verzeichnissen:

- **inventory**: `postgresql_cluster/inventory`
- **main.yml**: `postgresql_cluster/vars/main.yml`
- **inventory**: `postgresql_cluster/deploy_pgcluster.yml`
- **inventory**: `postgresql_cluster/config_pgcluster.yml`
- **inventory**: `postgresql_cluster/add_balancer.yml`
- **inventory**: `postgresql_cluster/add_pgnode.yml`

X.I Prequenteries

Auf allen Hosts müssen die Firewalls angepasst werden:

```
# nano /etc/iptables/rules.v4
# Generated by iptables-save v1.8.9 (nf_tables)
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-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
# generell
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 443 -j ACCEPT
```

```
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8080 -j ACCEPT
# postgres
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5432 -j ACCEPT
# pgbounder
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6432 -j ACCEPT
# etcd
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2379 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 2380 -j ACCEPT
# patroni
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 8008 -j ACCEPT
# haproxy
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5000 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5001 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5002 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 5003 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 7000 -j ACCEPT
COMMIT
# Completed
# systemctl restart iptables

```

Listing 133: Testsystem - Firewall Settings

Als nächstes muss der Proxy gesetzt werden:

```
# nano /etc/profile.d/proxy.sh
export http_proxy="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
export https_proxy="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
export no_proxy="127.0.0.1,localhost,10.0.0.0/8,.ksgr.ch,.sivc.first-it.ch"

export HTTP_PROXY="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
export HTTPS_PROXY="http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
export NO_PROXY="127.0.0.1,localhost,10.0.0.0/8,.ksgr.ch,.sivc.first-it.ch"
# source /etc/profile.d/proxy.sh

```

Listing 134: Testsystem - Proxy Settings

Auch der apt Proxy muss gesetzt werden:

```
# nano /etc/apt/apt.conf.d/proxy.conf
Acauire::http::Proxy "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
Acauire::https::Proxy "http://xksgr_vks0041_inet<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090";
Acquire::http::proxy::foreman.ksgr.ch "DIRECT";

```

Listing 135: Testsystem - apt-Proxy Settings

X.II Deployment

Zuerst muss das `Inventory` angepasst hat werden:

```
# if dcs_exists: false and dcs_type: "etcd"
[etcd_cluster]  # recommendation: 3, or 5-7 nodes
10.0.22.170
10.0.22.171
```

```
10.0.22.172

# if dcs_exists: false and dcs_type: "consul"
[consul_instances]  # recommendation: 3 or 5-7 nodes

# if with_haproxy_load_balancing: true
[balancers]
10.0.22.176
10.0.22.177

# PostgreSQL nodes
[master]
10.0.22.173 postgresql_exists=false

[replica]
10.0.22.174 postgresql_exists=false
10.0.22.175 postgresql_exists=false

[postgres_cluster:children]
master
replica

# if pgbackrest_install: true and "repo_host" is set
[pgbackrest]  # optional (Dedicated Repository Host)


# Connection settings
[all:vars]
ansible_connection='ssh'
ansible_ssh_port='22'
ansible_user='itgramic'
ansible_ssh_pass='<Secret ;-)>'  # "sshpass" package is required for use "ansible_ssh_pass"
#ansible_ssh_private_key_file=
ansible_python_interpreter='/usr/bin/env python3'

[pgbackrest:vars]


```

Listing 136: Testsystem - Deployment - inventory

Nun muss das `main.yml` dran. Wichtig sind folgende einstellungen.

```
---
#--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
# Proxy variables (optional) for download packages using a proxy server
proxy_env: {}  # yamllint disable rule:braces
#--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
#proxy_env:
#  http_proxy: "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090"
#  https_proxy: "http://xksgr_vks0041_inet:<Password Secure / Safe>@webproxy.sivc.first-it.ch:9090"

# Cluster variables
#cluster_vip: ""  # IP address for client access to the databases in the cluster (optional).
cluster_vip: "10.0.22.178"  # IP address for client access to the databases in the cluster (optional).
```

```yaml
vip_interface: "{{ ansible_default_ipv4.interface }}"  # interface name (e.g., "ens32").
# Note: VIP-based solutions such as keepalived or vip-manager may not function correctly in cloud environments like AWS.

#patroni_cluster_name: "postgres-cluster"  # the cluster name (must be unique for each cluster)
patroni_cluster_name: "k8s-core-psql"  # the cluster name (must be unique for each cluster)
patroni_install_version: "3.3.0"  # or 'latest'

patroni_superuser_username: "postgres"
patroni_superuser_password: "<Password Secure / Safe>"
patroni_replication_username: "replicator"
patroni_replication_password: "<Password Secure / Safe>"

#synchronous_mode: false  # or 'true' for enable synchronous database replication
synchronous_mode: true  # or 'true' for enable synchronous database replication
synchronous_mode_strict: false  # if 'true' then block all client writes to the master, when a synchronous replica is not available
#synchronous_node_count: 1  # number of synchronous standby databases
synchronous_node_count: 2  # number of synchronous standby databases

# Load Balancing
with_haproxy_load_balancing: true  # or 'true' if you want to install and configure the load-balancing
haproxy_listen_port:
  master: 5000
  replicas: 5001
  replicas_sync: 5002
  replicas_async: 5003
# The following ('_direct') ports are used for direct connections to the PostgreSQL database,
# bypassing the PgBouncer connection pool (if 'pgbouncer_install' is 'true').
# Uncomment the relevant lines if you need to set up direct connections.
  stats: 7000
haproxy_maxconn:
  global: 100000
  master: 10000
  replica: 10000
haproxy_timeout:
  client: "60m"
  server: "60m"

# keepalived (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'true')
keepalived_virtual_router_id: "{{ cluster_vip.split('.')[3] | int }}" # The last octet of 'cluster_vip' IP address is used by default.
# virtual_router_id - must be unique in the network (available values are 0..255).

# vip-manager (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'false')
vip_manager_version: "2.4.0"  # version to install
vip_manager_conf: "/etc/patroni/vip-manager.yml"
vip_manager_interval: "1000"  # time (in milliseconds) after which vip-manager wakes up and checks if it needs to register or release ip addresses.
vip_manager_iface: "{{ vip_interface }}"  # interface to which the virtual ip will be added
vip_manager_ip: "{{ cluster_vip }}"  # the virtual ip address to manage
vip_manager_mask: "24"  # netmask for the virtual ip


# DCS (Distributed Consensus Store)
dcs_exists: false  # or 'true' if you don't want to deploy a new etcd cluster
dcs_type: "etcd"  # or 'consul'

```

```yaml
# if dcs_type: "etcd" and dcs_exists: false
etcd_version: "3.5.11"  # version for deploy etcd cluster
etcd_data_dir: "/var/lib/etcd"
etcd_cluster_name: "etcd-{{ patroni_cluster_name }}"  # ETCD_INITIAL_CLUSTER_TOKEN

# if dcs_type: "etcd" and dcs_exists: true
patroni_etcd_hosts: []  # list of servers of an existing etcd cluster

patroni_etcd_namespace: "service"  # (optional) etcd namespace (prefix)
patroni_etcd_username: "" # (optional) username for etcd authentication
patroni_etcd_password: "" # (optional) password for etcd authentication
patroni_etcd_protocol: "" # (optional) http or https, if not specified http is used

# if dcs_type: "consul"
consul_version: "1.15.8"
consul_config_path: "/etc/consul"
consul_configd_path: "{{ consul_config_path }}/conf.d"
consul_data_path: "/var/lib/consul"
consul_domain: "consul"  # Consul domain name
consul_datacenter: "dc1"  # Datacenter label (can be specified for each host in the inventory)
consul_disable_update_check: true  # Disables automatic checking for security bulletins and new version releases
consul_enable_script_checks: true  # This controls whether health checks that execute scripts are enabled on this agent
consul_enable_local_script_checks: true  # Enable them when they are defined in the local configuration files
consul_ui: false  # Enable the consul UI?
consul_syslog_enable: true  # Enable logging to syslog
consul_iface: "{{ ansible_default_ipv4.interface }}"  # specify the interface name with a Private IP (ex. "enp7s0")
# TLS
# You can enable TLS encryption by dropping a CA certificate, server certificate, and server key in roles/consul/files/
consul_tls_enable: false
consul_tls_ca_crt: "ca.crt"
consul_tls_server_crt: "server.crt"
consul_tls_server_key: "server.key"
# DNS
consul_recursors: []  # List of upstream DNS servers
consul_dnsmasq_enable: true  # Enable DNS forwarding with Dnsmasq
consul_dnsmasq_cache: 0  # dnsmasq cache-size (0 - disable caching)
consul_dnsmasq_servers: "{{ nameservers }}" # Upstream DNS servers used by dnsmasq
consul_join: []  # List of LAN servers of an existing consul cluster, to join.

# https://developer.hashicorp.com/consul/docs/discovery/services
consul_services:
  - name: "{{ patroni_cluster_name }}"
    id: "{{ patroni_cluster_name }}-master"
    tags: ['master', 'primary']
    port: "{{ pgbouncer_listen_port }}"  # or "{{ postgresql_port }}" if pgbouncer_install: false
    checks:
      - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/primary", interval: "2s" }
      - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }  # comment out this check if pgbouncer_install: false
  - name: "{{ patroni_cluster_name }}"
    id: "{{ patroni_cluster_name }}-replica"
    tags: ['replica']
    port: "{{ pgbouncer_listen_port }}"
    checks:
      - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/replica?lag={{ patroni_maximum_lag_on_replica }}", interval: "2s" }
```

```yaml
      - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }

# PostgreSQL variables
postgresql_version: "16"
# postgresql_data_dir: see vars/Debian.yml or vars/RedHat.yml
postgresql_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }},127.0.0.1" to listen on a specific IP address.
postgresql_port: "5432"
postgresql_encoding: "UTF8"  # for bootstrap only (initdb)
postgresql_locale: "en_US.UTF-8"  # for bootstrap only (initdb)
postgresql_data_checksums: true  # for bootstrap only (initdb)
postgresql_password_encryption_algorithm: "scram-sha-256"  # or "md5" if your clients do not work with passwords encrypted with SCRAM-SHA-256

# (optional) list of users to be created (if not already exists)
postgresql_users:
  - { name: "{{ pgbouncer_auth_username }}", password: "{{ pgbouncer_auth_password }}", flags: "LOGIN", role: "" }
  - { name: "xksgr_sks1160_gitlab", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_sks1172_harbor", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_sks1195_kcsso", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_k8s_core_psql_monitor", password: "<Password Secure / Safe>", flags: "LOGIN", role: "pg_monitor" }
  - { name: "xksgr_k8s_core_psql_backup", password: "<Password Secure / Safe>", flags: "SUPERUSER" }

# (optional) list of databases to be created (if not already exists)
#postgresql_databases: []
postgresql_databases:
  - { db: "k8s_core_gitlab_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1160_gitlab" }
  - { db: "k8s_core_harbor_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1172_harbor" }
  - { db: "k8s_core_keycloak_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1195_kcsso" }
  - { db: "gramic_test", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "postgres" }

# (optional) list of schemas to be created (if not already exists)
postgresql_schemas: []
#  - { schema: "myschema", db: "mydatabase", owner: "mydb-user" }

# (optional) list of database extensions to be created (if not already exists)
#postgresql_extensions: []
postgresql_extensions:
  - { ext: "pgstattuple", db: "postgres" }
  - { ext: "pgstattuple", db: "k8s_core_gitlab_prod" }
  - { ext: "pgstattuple", db: "k8s_core_harbor_prod" }
  - { ext: "pgstattuple", db: "k8s_core_keycloak_prod" }
  - { ext: "pgstattuple", db: "gramic_test" }

# postgresql parameters to bootstrap dcs (are parameters for example)
postgresql_parameters:
  - { option: "max_connections", value: "500" }
  - { option: "superuser_reserved_connections", value: "5" }
  - { option: "password_encryption", value: "{{ postgresql_password_encryption_algorithm }}" }
  - { option: "max_locks_per_transaction", value: "512" }
  - { option: "max_prepared_transactions", value: "0" }
  - { option: "huge_pages", value: "try" }  # or "on" if you set "vm_nr_hugepages" in kernel parameters
  - { option: "shared_buffers", value: "{{ (ansible_memtotal_mb * 0.25) | int }}MB" }  # by default, 25% of RAM
  - { option: "effective_cache_size", value: "{{ (ansible_memtotal_mb * 0.75) | int }}MB" }  # by default, 75% of RAM
  - { option: "work_mem", value: "128MB" }  # please change this value
  - { option: "maintenance_work_mem", value: "256MB" }  # please change this value
```

```yaml
- { option: "checkpoint_timeout", value: "15min" }
- { option: "checkpoint_completion_target", value: "0.9" }
- { option: "min_wal_size", value: "2GB" }
- { option: "max_wal_size", value: "8GB" }  # or 16GB/32GB
- { option: "wal_buffers", value: "32MB" }
- { option: "default_statistics_target", value: "1000" }
- { option: "seq_page_cost", value: "1" }
- { option: "random_page_cost", value: "1.1" }  # or "4" for HDDs with slower random access
- { option: "effective_io_concurrency", value: "200" }  # or "2" for traditional HDDs with lower I/O parallelism
- { option: "synchronous_commit", value: "on" }  # or 'off' if you can you lose single transactions in case of a crash
- { option: "autovacuum", value: "on" }  # never turn off the autovacuum!
- { option: "autovacuum_max_workers", value: "5" }
- { option: "autovacuum_vacuum_scale_factor", value: "0.01" }  # or 0.005/0.001
- { option: "autovacuum_analyze_scale_factor", value: "0.01" }
- { option: "autovacuum_vacuum_cost_limit", value: "500" }  # or 1000/5000
- { option: "autovacuum_vacuum_cost_delay", value: "2" }
- { option: "autovacuum_naptime", value: "1s" }
- { option: "max_files_per_process", value: "4096" }
- { option: "archive_mode", value: "on" }
- { option: "archive_timeout", value: "1800s" }
- { option: "archive_command", value: "cd ." }  # not doing anything yet with WAL-s
- { option: "wal_level", value: "logical" }
- { option: "wal_keep_size", value: "2GB" }
- { option: "max_wal_senders", value: "10" }
- { option: "max_replication_slots", value: "10" }
- { option: "hot_standby", value: "on" }
- { option: "wal_log_hints", value: "on" }
- { option: "wal_compression", value: "on" }
- { option: "shared_preload_libraries", value: "pg_stat_statements,auto_explain" }
- { option: "pg_stat_statements.max", value: "10000" }
- { option: "pg_stat_statements.track", value: "all" }
- { option: "pg_stat_statements.track_utility", value: "false" }
- { option: "pg_stat_statements.save", value: "true" }
- { option: "auto_explain.log_min_duration", value: "10s" }  # enable auto_explain for 10-second logging threshold. Decrease this value if necessary
- { option: "auto_explain.log_analyze", value: "true" }
- { option: "auto_explain.log_buffers", value: "true" }
- { option: "auto_explain.log_timing", value: "false" }
- { option: "auto_explain.log_triggers", value: "true" }
- { option: "auto_explain.log_verbose", value: "true" }
- { option: "auto_explain.log_nested_statements", value: "true" }
- { option: "auto_explain.sample_rate", value: "0.01" }  # enable auto_explain for 1% of queries logging threshold
- { option: "track_io_timing", value: "on" }
- { option: "log_lock_waits", value: "on" }
- { option: "log_temp_files", value: "0" }
- { option: "track_activities", value: "on" }
- { option: "track_activity_query_size", value: "4096" }
- { option: "track_counts", value: "on" }
- { option: "track_functions", value: "all" }
- { option: "log_checkpoints", value: "on" }
- { option: "logging_collector", value: "on" }
- { option: "log_truncate_on_rotation", value: "on" }
- { option: "log_rotation_age", value: "1d" }
- { option: "log_rotation_size", value: "0" }
- { option: "log_line_prefix", value: "'%t [%p-%l] %r %q%u@%d '" }
```

```yaml
  - { option: "log_filename", value: "postgresql-%a.log" }
  - { option: "log_directory", value: "{{ postgresql_log_dir }}" }
  - { option: "hot_standby_feedback", value: "on" }  # allows feedback from a hot standby to the primary that will avoid query conflicts
  - { option: "max_standby_streaming_delay", value: "30s" }
  - { option: "wal_receiver_status_interval", value: "10s" }
  - { option: "idle_in_transaction_session_timeout", value: "10min" }  # reduce this timeout if possible
  - { option: "jit", value: "off" }
  - { option: "max_worker_processes", value: "24" }
  - { option: "max_parallel_workers", value: "8" }
  - { option: "max_parallel_workers_per_gather", value: "2" }
  - { option: "max_parallel_maintenance_workers", value: "2" }
  - { option: "tcp_keepalives_count", value: "10" }
  - { option: "tcp_keepalives_idle", value: "300" }
  - { option: "tcp_keepalives_interval", value: "30" }

# Set this variable to 'true' if you want the cluster to be automatically restarted
# after changing the 'postgresql_parameters' variable that requires a restart in the 'config_pgcluster.yml' playbook.
# By default, the cluster will not be automatically restarted.
pending_restart: false

# specify additional hosts that will be added to the pg_hba.conf
postgresql_pg_hba:
  - { type: "local", database: "all", user: "{{ patroni_superuser_username }}", address: "", method: "trust" }
  - { type: "local", database: "all", user: "{{ pgbouncer_auth_username }}", address: "", method: "trust" } # required for pgbouncer auth_user
  - { type: "local", database: "replication", user: "all", address: "", method: "trust" }
  - { type: "host", database: "replication", user: "all", address: "127.0.0.1/32", method: "trust" }
  - { type: "host", database: "replication", user: "all", address: "::1/128", method: "trust" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.173/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.174/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.175/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "local", database: "all", user: "all", address: "", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "all", address: "127.0.0.1/32", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "all", address: "::1/128", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_gitlab_prod", user: "xksgr_sks1160_gitlab", address: "10.0.20.88/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_harbor_prod", user: "xksgr_sks1172_harbor", address: "10.0.20.92/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_keycloak_prod", user: "xksgr_sks1195_kcsso", address: "10.0.20.98/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.63/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.43/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.77/24", method: "{{ postgresql_password_encryption_algorithm }}" }

# list of lines that Patroni will use to generate pg_ident.conf
postgresql_pg_ident: []

# the password file (~/.pgpass)
postgresql_pgpass:
  - "localhost:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "{{ inventory_hostname }}:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "*:{{ pgbouncer_listen_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "*:{{ postgresql_port }}:*:{{ patroni_replication_username }}:{{ patroni_replication_password }}"
  - "10.0.20.88:5432:k8s_core_gitlab_prod:xksgr_sks1160_gitlab:<Password Secure / Safe>"
  - "10.0.20.92:5432:k8s_core_harbor_prod:xksgr_sks1172_harbor:j<Password Secure / Safe>"
  - "10.0.20.98:5432:k8s_core_keycloak_prod:xksgr_sks1195_kcsso:<Password Secure / Safe>"

# PgBouncer parameters
```

```yaml
pgbouncer_install: true  # or 'false' if you do not want to install and configure the pgbouncer service
pgbouncer_processes: 1  # Number of pgbouncer processes to be used. Multiple processes use the so_reuseport option for better performance.
pgbouncer_conf_dir: "/etc/pgbouncer"
pgbouncer_log_dir: "/var/log/pgbouncer"
pgbouncer_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
pgbouncer_listen_port: 6432
pgbouncer_max_client_conn: 10000
pgbouncer_max_db_connections: 1000
pgbouncer_max_prepared_statements: 1024
pgbouncer_default_pool_size: 20
pgbouncer_query_wait_timeout: 120
pgbouncer_default_pool_mode: "session"
pgbouncer_admin_users: "{{ patroni_superuser_username }}"  # comma-separated list of users, who are allowed to change settings
pgbouncer_stats_users: "{{ patroni_superuser_username }}"  # comma-separated list of users who are just allowed to use SHOW command
pgbouncer_ignore_startup_parameters: "extra_float_digits,geqo,search_path"
pgbouncer_auth_type: "{{ postgresql_password_encryption_algorithm }}"
pgbouncer_auth_user: true # or 'false' if you want to manage the list of users for authentication in the database via userlist.txt
pgbouncer_auth_username: pgbouncer # user who can query the database via the user_search function
pgbouncer_auth_password: "<Password Secure / Safe>"
pgbouncer_auth_dbname: "postgres"
pgbouncer_client_tls_sslmode: "disable"
pgbouncer_client_tls_key_file: ""
pgbouncer_client_tls_cert_file: ""
pgbouncer_client_tls_ca_file: ""
pgbouncer_client_tls_protocols: "secure" # allowed values: tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3, all, secure (tlsv1.2,tlsv1.3)
pgbouncer_client_tls_ciphers: "default" # allowed values: default, secure, fast, normal, all (not recommended)

pgbouncer_pools:
  - { name: "postgres", dbname: "postgres", pool_parameters: "" }
  - { name: "k8s_core_gitlab_prod", dbname: "k8s_core_gitlab_prod", pool_parameters: "" }
  - { name: "k8s_core_harbor_prod", dbname: "k8s_core_harbor_prod", pool_parameters: "" }
  - { name: "k8s_core_keycloak_prod", dbname: "k8s_core_keycloak_prod", pool_parameters: "" }
  - { name: "gramic_test", dbname: "gramic_test", pool_parameters: "" }

# Extended variables (optional)
patroni_restapi_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
patroni_restapi_port: 8008
patroni_ttl: 30
patroni_loop_wait: 10
patroni_retry_timeout: 10
patroni_master_start_timeout: 300
patroni_maximum_lag_on_failover: 1048576 # (1MB) the maximum bytes a follower may lag to be able to participate in leader election.
patroni_maximum_lag_on_replica: "100MB" # the maximum of lag that replica can be in order to be available for read-only queries.

# https://patroni.readthedocs.io/en/latest/yaml_configuration.html#postgresql
patroni_callbacks: []

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#standby-cluster
# Requirements:
# 1. the cluster name for Standby Cluster must be unique ('patroni_cluster_name' variable)
# 2. the IP addresses (or network) of the Standby Cluster servers must be added to the pg_hba.conf of the Main Cluster ('postgresql_pg_hba' variable).
patroni_standby_cluster:
  host: ""  # an address of remote master
  port: "5432"  # a port of remote master
```

```yaml

# Permanent replication slots.
# These slots will be preserved during switchover/failover.
# https://patroni.readthedocs.io/en/latest/dynamic_configuration.html
patroni_slots: []

patroni_log_destination: stderr  # or 'logfile'
# if patroni_log_destination: logfile
patroni_log_dir: /var/log/patroni
patroni_log_level: info
patroni_log_traceback_level: error
patroni_log_format: "%(asctime)s %(levelname)s: %(message)s"
patroni_log_dateformat: ""
patroni_log_max_queue_size: 1000
patroni_log_file_num: 4
patroni_log_file_size: 25000000  # bytes
patroni_log_loggers_patroni_postmaster: warning
patroni_log_loggers_urllib3: warning  # or 'debug'

patroni_watchdog_mode: automatic  # or 'off', 'required'
patroni_watchdog_device: /dev/watchdog

patroni_postgresql_use_pg_rewind: true  # or 'false'
# try to use pg_rewind on the former leader when it joins cluster as a replica.

patroni_remove_data_directory_on_rewind_failure: false  # or 'true' (if use_pg_rewind: 'true')
# avoid removing the data directory on an unsuccessful rewind
# if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica.

patroni_remove_data_directory_on_diverged_timelines: false  # or 'true'
# if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica
# if it notices that timelines are diverging and the former master can not start streaming from the new master.

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#bootstrap
patroni_cluster_bootstrap_method: "initdb"  # or "wal-g", "pgbackrest", "pg_probackup"

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#building-replicas
patroni_create_replica_methods:
  - basebackup

pgbackrest:
  - { option: "command", value: "/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore" }
  - { option: "keep_data", value: "True" }
  - { option: "no_params", value: "True" }
wal_g:
  - { option: "command", value: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST" }
  - { option: "no_params", value: "True" }
basebackup:
  - { option: "max-rate", value: "100M" }
  - { option: "checkpoint", value: "fast" }
#  - { option: "waldir", value: "{{ postgresql_wal_dir }}" }
pg_probackup:
  - { option: "command", value: "{{ pg_probackup_restore_command }}" }
  - { option: "no_params", value: "true" }
```

```yaml

# "restore_command" written to recovery.conf when configuring follower (create replica)
postgresql_restore_command: ""

# pg_probackup
pg_probackup_install: false  # or 'true'
pg_probackup_install_from_postgrespro_repo: true  # or 'false'
pg_probackup_version: "{{ postgresql_version }}"
pg_probackup_instance: "pg_probackup_instance_name"
pg_probackup_dir: "/mnt/backup_dir"
pg_probackup_threads: "4"
pg_probackup_add_keys: "--recovery-target=latest --skip-external-dirs --no-validate"
# Ensure there is a space at the beginning of each part to prevent commands from concatenating.
pg_probackup_command_parts:
  - "pg_probackup-{{ pg_probackup_version }}"
  - " restore -B {{ pg_probackup_dir }}"
  - " --instance {{ pg_probackup_instance }}"
  - " -j {{ pg_probackup_threads }}"
  - " {{ pg_probackup_add_keys }}"
pg_probackup_restore_command: "{{ pg_probackup_command_parts | join('') }}"
pg_probackup_patroni_cluster_bootstrap_command: "{{ pg_probackup_command_parts | join('') }}"

# WAL-G
wal_g_install: false  # or 'true'
wal_g_version: "3.0.0"
wal_g_json:  # config https://github.com/wal-g/wal-g#configuration
  - { option: "AWS_ACCESS_KEY_ID", value: "{{ AWS_ACCESS_KEY_ID | default('') }}" }  # define values or pass via --extra-vars
  - { option: "AWS_SECRET_ACCESS_KEY", value: "{{ AWS_SECRET_ACCESS_KEY | default('') }}" }  # define values or pass via --extra-vars
  - { option: "WALG_S3_PREFIX", value: "{{ WALG_S3_PREFIX | default('') }}" } # define values or pass via --extra-vars
  - { option: "WALG_COMPRESSION_METHOD", value: "brotli" }  # or "lz4", "lzma", "zstd"
  - { option: "WALG_DELTA_MAX_STEPS", value: "6" }  # determines how many delta backups can be between full backups
  - { option: "PGDATA", value: "{{ postgresql_data_dir }}" }
  - { option: "PGHOST", value: "{{ postgresql_unix_socket_dir }}" }
  - { option: "PGPORT", value: "{{ postgresql_port }}" }
  - { option: "PGUSER", value: "{{ patroni_superuser_username }}" }
#  - { option: "AWS_S3_FORCE_PATH_STYLE", value: "true" }  # to use Minio.io S3-compatible storage
#  - { option: "AWS_ENDPOINT", value: "http://minio:9000" }  # to use Minio.io S3-compatible storage
#  - { option: "", value: "" }
wal_g_archive_command: "wal-g wal-push %p"
wal_g_patroni_cluster_bootstrap_command: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST"

# Define job_parts outside of wal_g_cron_jobs
# Ensure there is a space at the beginning of each part to prevent commands from concatenating.
wal_g_backup_command:
  - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
  - " && wal-g backup-push {{ postgresql_data_dir }} > {{ postgresql_log_dir }}/walg_backup.log 2>&1"
wal_g_delete_command:
  - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
  - " && wal-g delete retain FULL 4 --confirm > {{ postgresql_log_dir }}/walg_delete.log 2>&1"

wal_g_cron_jobs:
  - name: "WAL-G: Create daily backup"
    user: "postgres"
    file: /etc/cron.d/walg
```

```yaml
    minute: "30"
    hour: "3"
    day: "*"
    month: "*"
    weekday: "*"
    job: "{{ wal_g_backup_command | join('') }}"
  - name: "WAL-G: Delete old backups" # retain 4 full backups (adjust according to your company's backup retention policy)
    user: "postgres"
    file: /etc/cron.d/walg
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "*"
    job: "{{ wal_g_delete_command | join('') }}"

# pgBackRest
pgbackrest_install: false  # or 'true'
pgbackrest_install_from_pgdg_repo: false  # or 'false'
pgbackrest_stanza: "{{ patroni_cluster_name }}"  # specify your --stanza
pgbackrest_repo_type: "posix"  # or "s3", "gcs", "azure"
pgbackrest_repo_host: ""  # dedicated repository host (optional)
pgbackrest_repo_user: "postgres"
pgbackrest_conf_file: "/etc/pgbackrest/pgbackrest.conf"
# config https://pgbackrest.org/configuration.html
pgbackrest_conf:
  global:  # [global] section
    - { option: "log-level-file", value: "detail" }
    - { option: "log-path", value: "/var/log/pgbackrest" }
    - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
    - { option: "repo1-path", value: "/var/lib/pgbackrest" }
    - { option: "repo1-retention-full", value: "4" }
    - { option: "repo1-retention-archive", value: "4" }
    - { option: "start-fast", value: "y" }
    - { option: "stop-auto", value: "y" }
    - { option: "resume", value: "n" }
    - { option: "link-all", value: "y" }
    - { option: "spool-path", value: "/var/spool/pgbackrest" }
    - { option: "archive-async", value: "y" } # Enables asynchronous WAL archiving (details: https://pgbackrest.org/user-guide.html#async-archiving)
    - { option: "archive-get-queue-max", value: "1GiB" }
  stanza:  # [stanza_name] section
    - { option: "process-max", value: "4" }
    - { option: "log-level-console", value: "info" }
    - { option: "recovery-option", value: "recovery_target_action=promote" }
    - { option: "pg1-socket-path", value: "{{ postgresql_unix_socket_dir }}" }
    - { option: "pg1-path", value: "{{ postgresql_data_dir }}" }
#    - { option: "", value: "" }
# (optional) dedicated backup server config (if "repo_host" is set)
pgbackrest_server_conf:
  global:
    - { option: "log-level-file", value: "detail" }
    - { option: "log-level-console", value: "info" }
    - { option: "log-path", value: "/var/log/pgbackrest" }
    - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
```

```yaml
    - { option: "repo1-path", value: "/var/lib/pgbackrest" }
    - { option: "repo1-retention-full", value: "4" }
    - { option: "repo1-retention-archive", value: "4" }
    - { option: "repo1-bundle", value: "y" }
    - { option: "repo1-block", value: "y" }
    - { option: "start-fast", value: "y" }
    - { option: "stop-auto", value: "y" }
    - { option: "resume", value: "n" }
    - { option: "link-all", value: "y" }
    - { option: "archive-check", value: "y" }
    - { option: "archive-copy", value: "n" }
    - { option: "backup-standby", value: "y" }
#    - { option: "", value: "" }
# the stanza section will be generated automatically

pgbackrest_archive_command: "pgbackrest --stanza={{ pgbackrest_stanza }} archive-push %p"

pgbackrest_patroni_cluster_restore_command:
  '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore'  # restore from latest backup
#  '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --type=time "--target=2020-06-01 11:00:00+03" --delta restore'  # Point-in-Time Recovery (example)

# By default, the cron jobs is created on the database server.
# If 'repo_host' is defined, the cron jobs will be created on the pgbackrest server.
pgbackrest_cron_jobs:
  - name: "pgBackRest: Full Backup"
    file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
    user: "postgres"
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "0"
    job: "pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup"
    # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup; fi"
  - name: "pgBackRest: Diff Backup"
    file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
    user: "postgres"
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "1-6"
    job: "pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup"
    # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup; fi"

# PITR mode (if patroni_cluster_bootstrap_method: "pgbackrest" or "wal-g"):
# 1) The database cluster directory will be cleaned (for "wal-g") or overwritten (for "pgbackrest" --delta restore).
# 2) And also the patroni cluster "{{ patroni_cluster_name }}" will be removed from the DCS (if exist) before recovery.

disable_archive_command: true  # or 'false' to not disable archive_command after restore
keep_patroni_dynamic_json: true  # or 'false' to remove patroni.dynamic.json after restore (if exists)

# Netdata - https://github.com/netdata/netdata
netdata_install: false  # or 'true' for install Netdata on postgresql cluster nodes (with kickstart.sh)
```

```yaml
netdata_install_options: "--stable-channel --disable-telemetry --dont-wait"
netdata_conf:
  web_bind_to: "*"
  # https://learn.netdata.cloud/docs/store/change-metrics-storage
  memory_mode: "dbengine"  # The long-term metrics storage with efficient RAM and disk usage.
  page_cache_size: 64  # Determines the amount of RAM in MiB that is dedicated to caching Netdata metric values.
  dbengine_disk_space: 1024  # Determines the amount of disk space in MiB that is dedicated to storing Netdata metric values.

...


```

Listing 137: Testsystem - Deployment - main.yml

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

```
root@sks1000:~/postgresql_cluster# ansible all -m ping
10.0.22.176 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.177 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.172 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.170 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.171 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.175 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.173 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
10.0.22.174 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}

```

Listing 138: Deploy - Anhang - Ansible Ping

Deployt wird wie folgt:

```
ansible-playbook deploy_pgcluster.yml
```

Listing 139: Deploy - Anhang - deploy_pgcluster.yml

Nach dem deployment sollte folgende Ausgabe zu sehen sein:

```
root@sks1000:~/postgresql_cluster# ansible-playbook deploy_pgcluster.yml

PLAY [Deploy PostgreSQL HA Cluster (based on "Patroni")]
    ***********************************************************************************************************************************************

TASK [Gathering Facts]
    ***********************************************************************************************************************************************************************************************

ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.171]
ok: [10.0.22.176]
ok: [10.0.22.177]
ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

TASK [Include main variables]
    ***********************************************************************************************************************************************************************************************

ok: [10.0.22.170]
ok: [10.0.22.171]
ok: [10.0.22.172]
ok: [10.0.22.176]
ok: [10.0.22.177]
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include system variables]
    ***********************************************************************************************************************************************************************************************

ok: [10.0.22.171]
ok: [10.0.22.170]
ok: [10.0.22.176]
ok: [10.0.22.177]
ok: [10.0.22.172]
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include OS-specific variables]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.171]
ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.176]
ok: [10.0.22.177]
ok: [10.0.22.173]
```

```
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [System information]
    ****************************************************************************************************************************************************************************************************

ok: [10.0.22.170] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-18-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.171] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-18-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.176] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-18-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.177] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-18-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
```

```
ok: [10.0.22.172] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 4, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-18-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.173] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-20-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.174] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-20-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}
ok: [10.0.22.175] => {
    "system_info": {
        "Architecture": "x86_64",
        "CPU model": "Intel(R) Xeon(R) Gold 6142 CPU @ 2.60GHz, count: 8, cores: 1",
        "Disk space total": "45.14 GB",
        "Kernel": "6.1.0-20-amd64",
        "Memory": "7.71 GB",
        "OS": "Debian 12.5",
        "Product name": "VMware Virtual Platform",
        "Virtualization type": "VMware"
    }
}

TASK [pre-checks : PostgreSQL | check that data directory "/var/lib/postgresql/16/main" is not initialized]
    ***********************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]
```

```

TASK [Update apt cache]
    ***************************************************************************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.176]
ok: [10.0.22.172]
ok: [10.0.22.177]
ok: [10.0.22.171]
ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

TASK [Make sure the gnupg and apt-transport-https packages are present]
    ***********************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.176]
changed: [10.0.22.172]
changed: [10.0.22.171]
changed: [10.0.22.177]
changed: [10.0.22.173]
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [Make sure that the iproute is installed]
    ***************************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.176]
ok: [10.0.22.172]
ok: [10.0.22.171]
ok: [10.0.22.177]
ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

PLAY [etcd_cluster.yml | Deploy etcd Cluster]
    ***********************************************************************************************************************************************

TASK [Gathering Facts]
    ***************************************************************************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.171]

TASK [Include main variables]
    ***************************************************************************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.171]
ok: [10.0.22.172]

TASK [Include system variables]
    ***************************************************************************************************************************************************************
```

```
ok: [10.0.22.170]
ok: [10.0.22.171]
ok: [10.0.22.172]

TASK [Include OS-specific variables]
    ******************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.171]
ok: [10.0.22.172]

TASK [Update apt cache]
    ******************************************************************************************************************************************************************************

ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.171]

TASK [Make sure the gnupg and apt-transport-https packages are present]
    ******************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.171]

TASK [sysctl : Build a sysctl_conf dynamic variable]
    ******************************************************************************************************************************************************************
ok: [10.0.22.170] => (item=etcd_cluster)
ok: [10.0.22.171] => (item=etcd_cluster)
ok: [10.0.22.172] => (item=etcd_cluster)

TASK [etcd : Make sure the unzip/tar packages are present]
    ******************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.172]
changed: [10.0.22.171]

TASK [etcd : Download "etcd" package]
    ******************************************************************************************************************************************************************************
changed: [10.0.22.171] => (item=https://github.com/etcd-io/etcd/releases/download/v3.5.11/etcd-v3.5.11-linux-amd64.tar.gz)
[WARNING]: Module remote_tmp /tmp/root/ansible did not exist and was created with a mode of 0700, this may cause issues when running as another user. To avoid this, create the
    remote_tmp dir with the correct
permissions manually
changed: [10.0.22.172] => (item=https://github.com/etcd-io/etcd/releases/download/v3.5.11/etcd-v3.5.11-linux-amd64.tar.gz)
changed: [10.0.22.170] => (item=https://github.com/etcd-io/etcd/releases/download/v3.5.11/etcd-v3.5.11-linux-amd64.tar.gz)

TASK [etcd : Extract "etcd" into /tmp]
    ******************************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.172]
changed: [10.0.22.171]

TASK [etcd : Copy "etcd" and "etcdctl" binary files to /usr/local/bin/]
    ******************************************************************************************************************************************************************
changed: [10.0.22.170] => (item=etcd)
```

```
changed: [10.0.22.172] => (item=etcd)
changed: [10.0.22.171] => (item=etcd)
changed: [10.0.22.170] => (item=etcdctl)
changed: [10.0.22.172] => (item=etcdctl)
changed: [10.0.22.171] => (item=etcdctl)

TASK [etcd : Add etcd user]
    ************************************************************************************************************************************************************************************************************************************************************************************************************

changed: [10.0.22.170]
changed: [10.0.22.171]
changed: [10.0.22.172]

TASK [etcd : Create etcd conf directory]
    ************************************************************************************************************************************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.172]
changed: [10.0.22.171]

TASK [etcd : Create etcd data directory]
    ************************************************************************************************************************************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.172]
changed: [10.0.22.171]
[WARNING]: sftp transfer mechanism failed on [10.0.22.170]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: sftp transfer mechanism failed on [10.0.22.172]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: sftp transfer mechanism failed on [10.0.22.171]. Use ANSIBLE_DEBUG=1 to see detailed information

TASK [etcd : Generate conf file "/etc/etcd/etcd.conf"]
    ************************************************************************************************************************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.172]
changed: [10.0.22.171]

TASK [etcd : Copy systemd service file]
    ************************************************************************************************************************************************************************************************************************************************************************************************
changed: [10.0.22.170]
changed: [10.0.22.171]
changed: [10.0.22.172]

TASK [etcd : Enable and start etcd service]
    ************************************************************************************************************************************************************************************************************************************************************************************
changed: [10.0.22.172]
changed: [10.0.22.171]
changed: [10.0.22.170]

TASK [etcd : Wait for port 2379 to become open on the host]
    ************************************************************************************************************************************************************************************************************************************************************************
ok: [10.0.22.170]
ok: [10.0.22.172]
ok: [10.0.22.171]

TASK [etcd : Wait until the etcd cluster is healthy]
    ************************************************************************************************************************************************************************************************************************************************************************
```

```
ok: [10.0.22.172]
ok: [10.0.22.171]
ok: [10.0.22.170]

TASK [etcd : cluster health] ********************************************************************************************************************************************************************
ok: [10.0.22.170] => {
    "msg": "http://10.0.22.170:2379 is healthy: successfully committed proposal: took = 2.588786ms\n"
}
ok: [10.0.22.171] => {
    "msg": "http://10.0.22.171:2379 is healthy: successfully committed proposal: took = 5.296369ms\n"
}
ok: [10.0.22.172] => {
    "msg": "http://10.0.22.172:2379 is healthy: successfully committed proposal: took = 2.024139ms\n"
}

PLAY [consul.yml | Configure Consul instances] ******************************************************************************************************************************************
skipping: no hosts matched

PLAY [deploy_pgcluster.yml | Postgres Cluster Configuration] ****************************************************************************************************************************

TASK [Gathering Facts] ******************************************************************************************************************************************************************
ok: [10.0.22.174]
ok: [10.0.22.173]
ok: [10.0.22.175]

TASK [Include main variables] ***********************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include system variables] *********************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include OS-specific variables] ****************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [add-repository : Add repository apt-key] ******************************************************************************************************************************************
changed: [10.0.22.173] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.asc'})
```

```
changed: [10.0.22.175] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.asc'})
changed: [10.0.22.174] => (item={'key': 'https://www.postgresql.org/media/keys/ACCC4CF8.asc'})

TASK [add-repository : Add repository]
    ***********************************************************************************************************************************************************************************
changed: [10.0.22.175] => (item={'repo': 'deb https://apt.postgresql.org/pub/repos/apt/ bookworm-pgdg main'})
changed: [10.0.22.173] => (item={'repo': 'deb https://apt.postgresql.org/pub/repos/apt/ bookworm-pgdg main'})
changed: [10.0.22.174] => (item={'repo': 'deb https://apt.postgresql.org/pub/repos/apt/ bookworm-pgdg main'})

TASK [packages : Install system packages]
    ***********************************************************************************************************************************************************************************
ok: [10.0.22.173] => (item=['python3'])
ok: [10.0.22.175] => (item=['python3'])
ok: [10.0.22.174] => (item=['python3'])
ok: [10.0.22.173] => (item=python3)
ok: [10.0.22.175] => (item=python3)
ok: [10.0.22.174] => (item=python3)
changed: [10.0.22.175] => (item=python3-dev)
changed: [10.0.22.173] => (item=python3-dev)
changed: [10.0.22.174] => (item=python3-dev)
changed: [10.0.22.175] => (item=python3-psycopg2)
changed: [10.0.22.173] => (item=python3-psycopg2)
changed: [10.0.22.175] => (item=python3-setuptools)
changed: [10.0.22.174] => (item=python3-psycopg2)
changed: [10.0.22.173] => (item=python3-setuptools)
changed: [10.0.22.174] => (item=python3-setuptools)
changed: [10.0.22.175] => (item=python3-pip)
changed: [10.0.22.173] => (item=python3-pip)
ok: [10.0.22.175] => (item=curl)
ok: [10.0.22.173] => (item=curl)
ok: [10.0.22.175] => (item=less)
ok: [10.0.22.173] => (item=less)
ok: [10.0.22.175] => (item=sudo)
ok: [10.0.22.173] => (item=sudo)
changed: [10.0.22.174] => (item=python3-pip)
ok: [10.0.22.174] => (item=curl)
ok: [10.0.22.174] => (item=less)
ok: [10.0.22.174] => (item=sudo)
changed: [10.0.22.175] => (item=vim)
changed: [10.0.22.173] => (item=vim)
ok: [10.0.22.175] => (item=gcc)
ok: [10.0.22.173] => (item=gcc)
changed: [10.0.22.175] => (item=jq)
changed: [10.0.22.173] => (item=jq)
ok: [10.0.22.175] => (item=iptables)
ok: [10.0.22.173] => (item=iptables)
changed: [10.0.22.174] => (item=vim)
ok: [10.0.22.174] => (item=gcc)
changed: [10.0.22.175] => (item=acl)
changed: [10.0.22.173] => (item=acl)
changed: [10.0.22.175] => (item=dnsutils)
changed: [10.0.22.174] => (item=jq)
changed: [10.0.22.173] => (item=dnsutils)
ok: [10.0.22.174] => (item=iptables)
```

```
changed: [10.0.22.174] => (item=acl)
changed: [10.0.22.174] => (item=dnsutils)

TASK [packages : PostgreSQL | ensure postgresql database-cluster manager package]
    ************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [packages : PostgreSQL | disable initializing of a default postgresql cluster]
    ************************************************************************************************************************
changed: [10.0.22.175]
changed: [10.0.22.173]
changed: [10.0.22.174]

TASK [packages : PostgreSQL | disable log rotation with logrotate for postgresql]
    ************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [packages : Install PostgreSQL packages]
    ************************************************************************************************************************
changed: [10.0.22.174] => (item=postgresql-16)
changed: [10.0.22.173] => (item=postgresql-16)
changed: [10.0.22.175] => (item=postgresql-16)
ok: [10.0.22.173] => (item=postgresql-client-16)
ok: [10.0.22.174] => (item=postgresql-client-16)
ok: [10.0.22.175] => (item=postgresql-client-16)
ok: [10.0.22.173] => (item=postgresql-contrib-16)
ok: [10.0.22.175] => (item=postgresql-contrib-16)
ok: [10.0.22.174] => (item=postgresql-contrib-16)
changed: [10.0.22.174] => (item=postgresql-server-dev-16)
changed: [10.0.22.175] => (item=postgresql-server-dev-16)
changed: [10.0.22.173] => (item=postgresql-server-dev-16)
[WARNING]: sftp transfer mechanism failed on [10.0.22.174]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: sftp transfer mechanism failed on [10.0.22.173]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: sftp transfer mechanism failed on [10.0.22.175]. Use ANSIBLE_DEBUG=1 to see detailed information

TASK [sudo : Add user to /etc/sudoers.d/]
    ************************************************************************************************************************
changed: [10.0.22.174] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL'})
changed: [10.0.22.173] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL'})
changed: [10.0.22.175] => (item={'name': 'postgres', 'nopasswd': 'yes', 'commands': 'ALL'})

TASK [swap : Ensure swap exists]
    ************************************************************************************************************************

ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

TASK [swap : Swap exists]
    ************************************************************************************************************************
```

```
ok: [10.0.22.173] => {
    "msg": "swap_size_mb: 3904"
}
ok: [10.0.22.174] => {
    "msg": "swap_size_mb: 3904"
}
ok: [10.0.22.175] => {
    "msg": "swap_size_mb: 3904"
}

TASK [swap : Disable all existing swaps]
    ************************************************************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [swap : Remove swap from /etc/fstab]
    ************************************************************************************************************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [swap : Remove swap file (if exists)]
    ************************************************************************************************************************************************************************************************************
ok: [10.0.22.174]
ok: [10.0.22.175]
ok: [10.0.22.173]

TASK [swap : Create swap file]
    ************************************************************************************************************************************************************************************************************

changed: [10.0.22.175]
changed: [10.0.22.173]
changed: [10.0.22.174]

TASK [swap : Set permissions on swap file]
    ************************************************************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [swap : Make swap file if necessary]
    ************************************************************************************************************************************************************************************************************
changed: [10.0.22.175]
changed: [10.0.22.173]
changed: [10.0.22.174]

TASK [swap : Run swapon on the swap file]
    ************************************************************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]
```

```
TASK [swap : Manage swap file entry in fstab]
    ****************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [sysctl : Build a sysctl_conf dynamic variable]
    ****************************************************************************************************************************************************************
ok: [10.0.22.173] => (item=master)
ok: [10.0.22.173] => (item=postgres_cluster)
ok: [10.0.22.174] => (item=postgres_cluster)
ok: [10.0.22.174] => (item=replica)
ok: [10.0.22.175] => (item=postgres_cluster)
ok: [10.0.22.175] => (item=replica)

TASK [sysctl : Setting kernel parameters]
    ****************************************************************************************************************************************************************
changed: [10.0.22.173] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
changed: [10.0.22.175] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
changed: [10.0.22.173] => (item={'name': 'vm.swappiness', 'value': '1'})
changed: [10.0.22.175] => (item={'name': 'vm.swappiness', 'value': '1'})
changed: [10.0.22.173] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
changed: [10.0.22.175] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
changed: [10.0.22.173] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
changed: [10.0.22.175] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
changed: [10.0.22.175] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864'})
changed: [10.0.22.174] => (item={'name': 'vm.overcommit_memory', 'value': '2'})
changed: [10.0.22.175] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})
changed: [10.0.22.174] => (item={'name': 'vm.swappiness', 'value': '1'})
changed: [10.0.22.175] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
changed: [10.0.22.174] => (item={'name': 'vm.min_free_kbytes', 'value': '102400'})
changed: [10.0.22.175] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
changed: [10.0.22.174] => (item={'name': 'vm.dirty_expire_centisecs', 'value': '1000'})
changed: [10.0.22.175] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
changed: [10.0.22.174] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864'})
changed: [10.0.22.173] => (item={'name': 'vm.dirty_background_bytes', 'value': '67108864'})
changed: [10.0.22.175] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
changed: [10.0.22.174] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})
changed: [10.0.22.175] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
changed: [10.0.22.174] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
changed: [10.0.22.175] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000 65535'})
changed: [10.0.22.174] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
changed: [10.0.22.175] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000'})
changed: [10.0.22.174] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
changed: [10.0.22.173] => (item={'name': 'vm.dirty_bytes', 'value': '536870912'})
changed: [10.0.22.173] => (item={'name': 'vm.zone_reclaim_mode', 'value': '0'})
changed: [10.0.22.173] => (item={'name': 'kernel.numa_balancing', 'value': '0'})
changed: [10.0.22.173] => (item={'name': 'kernel.sched_autogroup_enabled', 'value': '0'})
changed: [10.0.22.175] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192'})
changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
changed: [10.0.22.175] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
```

```
changed: [10.0.22.174] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000 65535'})
changed: [10.0.22.174] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000'})
changed: [10.0.22.175] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
changed: [10.0.22.173] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000 65535'})
changed: [10.0.22.173] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000'})
changed: [10.0.22.174] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192'})
changed: [10.0.22.173] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192'})
changed: [10.0.22.174] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
changed: [10.0.22.173] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
changed: [10.0.22.174] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
changed: [10.0.22.173] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})

TASK [transparent_huge_pages : Create systemd service "disable-transparent-huge-pages.service"]
    ****************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

RUNNING HANDLER [transparent_huge_pages : Start disable-transparent-huge-pages service]
    ****************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [pam_limits : Linux PAM limits | add or modify nofile limits]
    ****************************************************************************************************************************
changed: [10.0.22.174] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value': 65536})
changed: [10.0.22.175] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value': 65536})
changed: [10.0.22.173] => (item={'limit_type': 'soft', 'limit_item': 'nofile', 'value': 65536})
changed: [10.0.22.174] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value': 200000})
changed: [10.0.22.175] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value': 200000})
changed: [10.0.22.173] => (item={'limit_type': 'hard', 'limit_item': 'nofile', 'value': 200000})

TASK [locales : Generate locales]
    ****************************************************************************************************************************

ok: [10.0.22.174] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})
ok: [10.0.22.175] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})
ok: [10.0.22.173] => (item={'language_country': 'en_US', 'encoding': 'UTF-8'})

TASK [locales : Set locale "en_US.utf-8" into /etc/default/locale]
    ****************************************************************************************************************************
changed: [10.0.22.173] => (item=LANG=en_US.utf-8)
changed: [10.0.22.174] => (item=LANG=en_US.utf-8)
changed: [10.0.22.175] => (item=LANG=en_US.utf-8)
changed: [10.0.22.173] => (item=LANGUAGE=en_US.utf-8)
changed: [10.0.22.174] => (item=LANGUAGE=en_US.utf-8)
changed: [10.0.22.175] => (item=LANGUAGE=en_US.utf-8)
changed: [10.0.22.173] => (item=LC_ALL=en_US.utf-8)
changed: [10.0.22.174] => (item=LC_ALL=en_US.utf-8)
changed: [10.0.22.175] => (item=LC_ALL=en_US.utf-8)

PLAY [balancers.yml | Configure HAProxy load balancers]
    ****************************************************************************************************************************
```

```

TASK [Gathering Facts] ********************************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [Include main variables] *************************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [Include system variables] ***********************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [Include OS-specific variables] ******************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [Update apt cache] *******************************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [Make sure the gnupg and apt-transport-https packages are present] *******************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [sysctl : Build a sysctl_conf dynamic variable] **************************************************************************************************
ok: [10.0.22.176] => (item=balancers)
ok: [10.0.22.177] => (item=balancers)

TASK [sysctl : Setting kernel parameters] *************************************************************************************************************
changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_nonlocal_bind', 'value': '1'})
changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_forward', 'value': '1'})
changed: [10.0.22.176] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000 65535'})
changed: [10.0.22.177] => (item={'name': 'net.ipv4.ip_local_port_range', 'value': '10000 65535'})
changed: [10.0.22.176] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000'})
changed: [10.0.22.177] => (item={'name': 'net.core.netdev_max_backlog', 'value': '10000'})
changed: [10.0.22.177] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192'})
changed: [10.0.22.176] => (item={'name': 'net.ipv4.tcp_max_syn_backlog', 'value': '8192'})
changed: [10.0.22.176] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
changed: [10.0.22.177] => (item={'name': 'net.core.somaxconn', 'value': '65535'})
```

```
changed: [10.0.22.176] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})
changed: [10.0.22.177] => (item={'name': 'net.ipv4.tcp_tw_reuse', 'value': '1'})

TASK [haproxy : Install HAProxy package]
    ************************************************************************************************************************************************************
changed: [10.0.22.176]
changed: [10.0.22.177]

TASK [haproxy : Make sure the kernel parameter "net.ipv4.ip_nonlocal_bind" are enabled]
    ************************************************************************************************************************************
ok: [10.0.22.177]
ok: [10.0.22.176]

TASK [haproxy : Add haproxy group]
    ************************************************************************************************************************************************************************

ok: [10.0.22.176]
ok: [10.0.22.177]

TASK [haproxy : Add haproxy user]
    ************************************************************************************************************************************************************************************

changed: [10.0.22.176]
changed: [10.0.22.177]

TASK [haproxy : Create directories]
    ************************************************************************************************************************************************************************************
changed: [10.0.22.177] => (item=/etc/haproxy)
changed: [10.0.22.176] => (item=/etc/haproxy)
ok: [10.0.22.177] => (item=/run/haproxy)
ok: [10.0.22.176] => (item=/run/haproxy)
changed: [10.0.22.177] => (item=/var/lib/haproxy/dev)
changed: [10.0.22.176] => (item=/var/lib/haproxy/dev)
[WARNING]: sftp transfer mechanism failed on [10.0.22.177]. Use ANSIBLE_DEBUG=1 to see detailed information
[WARNING]: sftp transfer mechanism failed on [10.0.22.176]. Use ANSIBLE_DEBUG=1 to see detailed information

TASK [haproxy : Generate conf file "/etc/haproxy/haproxy.cfg"]
    ************************************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [haproxy : Generate systemd service file "/etc/systemd/system/haproxy.service"]
    ************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [confd : Download and copy "confd" binary file to /usr/local/bin/]
    ************************************************************************************************************************************************************
changed: [10.0.22.177] => (item=https://github.com/kelseyhightower/confd/releases/download/v0.16.0/confd-0.16.0-linux-amd64)
changed: [10.0.22.176] => (item=https://github.com/kelseyhightower/confd/releases/download/v0.16.0/confd-0.16.0-linux-amd64)

TASK [confd : Create conf directories]
    ************************************************************************************************************************************************************************************
changed: [10.0.22.176] => (item=/etc/confd/conf.d)
```

```
changed: [10.0.22.177] => (item=/etc/confd/conf.d)
changed: [10.0.22.176] => (item=/etc/confd/templates)
changed: [10.0.22.177] => (item=/etc/confd/templates)

TASK [confd : Generate conf file "/etc/confd/confd.toml"]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [confd : Generate conf file "/etc/confd/conf.d/haproxy.toml"]
    ****************************************************************************************************************************************
changed: [10.0.22.176]
changed: [10.0.22.177]

TASK [confd : Generate template "/etc/confd/templates/haproxy.tmpl"]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [confd : Copy systemd service file]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [keepalived : Install keepalived packages]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [keepalived : Make sure the kernel parameters "net.ipv4.ip_nonlocal_bind", "net.ipv4.ip_forward" are enabled]
    ****************************************************************************************
ok: [10.0.22.176] => (item=net.ipv4.ip_nonlocal_bind)
ok: [10.0.22.177] => (item=net.ipv4.ip_nonlocal_bind)
ok: [10.0.22.177] => (item=net.ipv4.ip_forward)
ok: [10.0.22.176] => (item=net.ipv4.ip_forward)

TASK [keepalived : Make sure the "/usr/libexec/keepalived" directory exists]
    ****************************************************************************************************************************************
changed: [10.0.22.176]
changed: [10.0.22.177]

TASK [keepalived : Create vrrp_script "/usr/libexec/keepalived/haproxy_check.sh"]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

TASK [keepalived : Generate conf file "/etc/keepalived/keepalived.conf"]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

RUNNING HANDLER [haproxy : Restart haproxy service]
    ****************************************************************************************************************************************
changed: [10.0.22.177]
```

```
changed: [10.0.22.176]

RUNNING HANDLER [haproxy : Check HAProxy is started and accepting connections]
    ****************************************************************************************************************
ok: [10.0.22.177]
ok: [10.0.22.176]

RUNNING HANDLER [confd : Restart confd service]
    ********************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

RUNNING HANDLER [keepalived : Restart keepalived service]
    ******************************************************************************************************************************************
changed: [10.0.22.177]
changed: [10.0.22.176]

RUNNING HANDLER [keepalived : Wait for the cluster ip address (VIP) "10.0.22.178" is running]
    ****************************************************************************************************************
ok: [10.0.22.176]
ok: [10.0.22.177]

PLAY [deploy_pgcluster.yml | Install and configure pgBackRest]
    ******************************************************************************************************************************************

TASK [Gathering Facts]
    ******************************************************************************************************************************************************************************************************************

ok: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

TASK [Include main variables]
    ******************************************************************************************************************************************************************************************************************

ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include OS-specific variables]
    ******************************************************************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

PLAY [deploy_pgcluster.yml | PostgreSQL Cluster Deployment]
    ******************************************************************************************************************************************

TASK [Gathering Facts]
    ******************************************************************************************************************************************************************************************************************

ok: [10.0.22.175]
ok: [10.0.22.173]
ok: [10.0.22.174]
```

```

TASK [Include main variables]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include system variables]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [Include OS-specific variables]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [cron : Make sure that the cron package is installed]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.175]
ok: [10.0.22.174]
ok: [10.0.22.173]

TASK [pgbouncer : Install pgbouncer package]
    ***********************************************************************************************************************************************************************************************
changed: [10.0.22.175]
changed: [10.0.22.173]
changed: [10.0.22.174]

TASK [pgbouncer : Ensure config directory "/etc/pgbouncer" exist]
    ***********************************************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [pgbouncer : Ensure log directory "/var/log/pgbouncer" exist]
    ***********************************************************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [pgbouncer : Check if pgbouncer systemd service file exists]
    ***********************************************************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [pgbouncer : Stop and disable standard init script]
    ***********************************************************************************************************************************************************************************************
changed: [10.0.22.174]
```

```
changed: [10.0.22.175]
changed: [10.0.22.173]

TASK [pgbouncer : Configure pgbouncer systemd service file] ********************************************************************************************************
changed: [10.0.22.175] => (item=pgbouncer)
changed: [10.0.22.173] => (item=pgbouncer)
changed: [10.0.22.174] => (item=pgbouncer)

TASK [pgbouncer : Ensure pgbouncer service is enabled] ************************************************************************************************************
changed: [10.0.22.173] => (item=pgbouncer)
changed: [10.0.22.174] => (item=pgbouncer)
changed: [10.0.22.175] => (item=pgbouncer)

TASK [pgbouncer : Enable log rotation with logrotate] **************************************************************************************************************
changed: [10.0.22.175] => (item=pgbouncer)
changed: [10.0.22.174] => (item=pgbouncer)
changed: [10.0.22.173] => (item=pgbouncer)

TASK [pgbouncer : Configure pgbouncer.ini] *************************************************************************************************************************
changed: [10.0.22.175] => (item=pgbouncer)
changed: [10.0.22.173] => (item=pgbouncer)
changed: [10.0.22.174] => (item=pgbouncer)

TASK [pgpass : Configure a password file (/var/lib/postgresql/.pgpass)] ********************************************************************************************
changed: [10.0.22.175]
changed: [10.0.22.174]
changed: [10.0.22.173]

RUNNING HANDLER [Restart pgbouncer service] ************************************************************************************************************************
changed: [10.0.22.174] => (item=pgbouncer)
changed: [10.0.22.173] => (item=pgbouncer)
changed: [10.0.22.175] => (item=pgbouncer)

RUNNING HANDLER [Wait for port "6432" to become open on the host] **************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [patroni : Copy patroni requirements.txt file] ****************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [patroni : Install setuptools] ********************************************************************************************************************************
changed: [10.0.22.175]
```

```
changed: [10.0.22.173]
changed: [10.0.22.174]

TASK [patroni : Install requirements] ************************************************************************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [patroni : Install patroni 3.3.0] ***********************************************************************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [patroni : Create conf directory] ***********************************************************************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [patroni : Generate conf file "/etc/patroni/patroni.yml"] ***********************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [patroni : Copy systemd service file "/etc/systemd/system/patroni.service"] *****************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [patroni : Prepare PostgreSQL | make sure the postgresql log directory "/var/log/postgresql" exists] ****************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [patroni : Prepare PostgreSQL | make sure PostgreSQL data directory "/var/lib/postgresql/16/main" exists] ************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.173]
changed: [10.0.22.175]

TASK [patroni : Prepare PostgreSQL | make sure the postgresql config files exists] ***********************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [patroni : Prepare PostgreSQL | generate default postgresql config files] ***************************************************************************************************************
changed: [10.0.22.173]
```

```
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [patroni : Prepare PostgreSQL | make sure the data directory "/var/lib/postgresql/16/main" is empty]
    ******************************************************************************************************
changed: [10.0.22.174] => (item=absent)
changed: [10.0.22.173] => (item=absent)
changed: [10.0.22.175] => (item=absent)
changed: [10.0.22.173] => (item=directory)
changed: [10.0.22.174] => (item=directory)
changed: [10.0.22.175] => (item=directory)

TASK [patroni : Start patroni service on the Master server]
    ******************************************************************************************************************************************************
changed: [10.0.22.173]

TASK [patroni : Wait for port 8008 to become open on the host]
    ******************************************************************************************************************************************************
ok: [10.0.22.173]

TASK [patroni : Check PostgreSQL is started and accepting connections on Master]
    ******************************************************************************************************************************************************
ok: [10.0.22.173]

TASK [patroni : Wait for the cluster to initialize (master is the leader with the lock)]
    ******************************************************************************************************************************************************
ok: [10.0.22.173]

TASK [patroni : Prepare PostgreSQL | generate pg_hba.conf]
    ******************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [patroni : Prepare PostgreSQL | reload for apply the pg_hba.conf]
    ******************************************************************************************************************************************************
changed: [10.0.22.173]
ok: [10.0.22.175]
ok: [10.0.22.174]

TASK [patroni : Make sure the postgresql users are present, and password does not differ from the specified]
    ******************************************************************************************************
failed: [10.0.22.173] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and (pgbackrest_install|bool
     or wal_g_install|bool) and (existing_pgcluster is not defined or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template loop_control.label: 'None' has no
    attribute 'name'. 'None' has no attribute 'name'", "skip_reason": "Conditional result was False"}
...ignoring
failed: [10.0.22.174] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and (pgbackrest_install|bool
     or wal_g_install|bool) and (existing_pgcluster is not defined or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template loop_control.label: 'None' has no
    attribute 'name'. 'None' has no attribute 'name'", "skip_reason": "Conditional result was False"}
...ignoring
failed: [10.0.22.175] (item=None) => {"ansible_loop_var": "item", "changed": false, "false_condition": "patroni_cluster_bootstrap_method != \"initdb\" and (pgbackrest_install|bool
     or wal_g_install|bool) and (existing_pgcluster is not defined or not existing_pgcluster|bool)", "item": null, "msg": "Failed to template loop_control.label: 'None' has no
    attribute 'name'. 'None' has no attribute 'name'", "skip_reason": "Conditional result was False"}
...ignoring
```

```

TASK [patroni : Start patroni service on Replica servers]
    ************************************************************************************************************************************************
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [patroni : Wait for port 8008 to become open on the host]
    ************************************************************************************************************************************************
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [patroni : Check that the patroni is healthy on the replica server]
    ************************************************************************************************************************************************
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [patroni : Turning off postgresql autostart from config "start.conf" (will be managed by patroni)]
    ************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.174]
changed: [10.0.22.175]

TASK [patroni : Disable "postgresql@16-main" service]
    ************************************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [patroni : Add PATRONICTL_CONFIG_FILE environment variable into /etc/environment]
    ************************************************************************************************************************************************
changed: [10.0.22.173]
changed: [10.0.22.175]
changed: [10.0.22.174]

TASK [postgresql-users : Make sure the PostgreSQL users are present]
    ************************************************************************************************************************************************
changed: [10.0.22.173] => (item=pgbouncer)
changed: [10.0.22.173] => (item=xksgr_sks1160_gitlab)
changed: [10.0.22.173] => (item=xksgr_sks1172_harbor)
changed: [10.0.22.173] => (item=xksgr_sks1195_kcsso)
changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_monitor)
changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_backup)

TASK [postgresql-users : Grant roles to users]
    ************************************************************************************************************************************************
changed: [10.0.22.173] => (item=xksgr_k8s_core_psql_monitor)

TASK [postgresql-databases : Make sure the PostgreSQL databases are present]
    ************************************************************************************************************************************************
changed: [10.0.22.173] => (item={'db': 'k8s_core_gitlab_prod', 'encoding': 'UTF8', 'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1160_gitlab'})
changed: [10.0.22.173] => (item={'db': 'k8s_core_harbor_prod', 'encoding': 'UTF8', 'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1172_harbor'})
changed: [10.0.22.173] => (item={'db': 'k8s_core_keycloak_prod', 'encoding': 'UTF8', 'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'xksgr_sks1195_kcsso'})
changed: [10.0.22.173] => (item={'db': 'gramic_test', 'encoding': 'UTF8', 'lc_collate': 'en_US.UTF-8', 'lc_ctype': 'en_US.UTF-8', 'owner': 'postgres'})

```

```
TASK [postgresql-extensions : Add extensions to the databases]
    ***********************************************************************************************************************************
changed: [10.0.22.173] => (item={'ext': 'pgstattuple', 'db': 'postgres'})
changed: [10.0.22.173] => (item={'ext': 'pgstattuple', 'db': 'k8s_core_gitlab_prod'})
changed: [10.0.22.173] => (item={'ext': 'pgstattuple', 'db': 'k8s_core_harbor_prod'})
changed: [10.0.22.173] => (item={'ext': 'pgstattuple', 'db': 'k8s_core_keycloak_prod'})
changed: [10.0.22.173] => (item={'ext': 'pgstattuple', 'db': 'gramic_test'})

TASK [pgbouncer/config : Ensure config directory "/etc/pgbouncer" exist]
    ***********************************************************************************************************************************
ok: [10.0.22.173]
ok: [10.0.22.174]
ok: [10.0.22.175]

TASK [pgbouncer/config : Update pgbouncer.ini]
    ***********************************************************************************************************************************
ok: [10.0.22.173] => (item=pgbouncer)
ok: [10.0.22.175] => (item=pgbouncer)
ok: [10.0.22.174] => (item=pgbouncer)

TASK [pgbouncer/config : Check if 'user_search' function exists]
    ***********************************************************************************************************************************
ok: [10.0.22.173]

TASK [pgbouncer/config : Create 'user_search' function for pgbouncer 'auth_query' option]
    ***********************************************************************************************************************************
changed: [10.0.22.173]

TASK [deploy-finish : Get postgresql users list]
    ***********************************************************************************************************************************
ok: [10.0.22.173]

TASK [deploy-finish : PostgreSQL list of users]
    ***********************************************************************************************************************************
ok: [10.0.22.173] => {
    "users_result.stdout_lines": [
        "                                   List of roles",
        "          Role name          |                         Attributes                         ",
        "-----------------------------+------------------------------------------------------------",
        " pgbouncer                    | ",
        " postgres                     | Superuser, Create role, Create DB, Replication, Bypass RLS",
        " replicator                   | Replication",
        " xksgr_k8s_core_psql_backup   | Superuser",
        " xksgr_k8s_core_psql_monitor  | ",
        " xksgr_sks1160_gitlab         | Superuser",
        " xksgr_sks1172_harbor         | Superuser",
        " xksgr_sks1195_kcsso          | Superuser"
    ]
}

TASK [deploy-finish : Get postgresql database list]
    ***********************************************************************************************************************************
ok: [10.0.22.173]

```

```
TASK [deploy-finish : PostgreSQL list of databases]
    ***********************************************************************************************************************************************
ok: [10.0.22.173] => {
    "dbs_result.stdout_lines": [
        "          name          |        owner         | encoding |   collate   |    ctype    |  size   | tablespace ",
        "------------------------+----------------------+----------+-------------+-------------+---------+------------",
        " gramic_test            | postgres             | UTF8     | en_US.UTF-8 | en_US.UTF-8 | 7492 kB | pg_default",
        " k8s_core_gitlab_prod   | xksgr_sks1160_gitlab | UTF8     | en_US.UTF-8 | en_US.UTF-8 | 7492 kB | pg_default",
        " k8s_core_harbor_prod   | xksgr_sks1172_harbor | UTF8     | en_US.UTF-8 | en_US.UTF-8 | 7492 kB | pg_default",
        " k8s_core_keycloak_prod | xksgr_sks1195_kcsso  | UTF8     | en_US.UTF-8 | en_US.UTF-8 | 7492 kB | pg_default",
        " postgres               | postgres             | UTF8     | en_US.UTF-8 | en_US.UTF-8 | 7492 kB | pg_default",
        "(5 rows)"
    ]
}

TASK [deploy-finish : Check postgresql cluster health]
    ***********************************************************************************************************************************************
ok: [10.0.22.173]

TASK [deploy-finish : PostgreSQL Cluster health]
    ***********************************************************************************************************************************************
ok: [10.0.22.173] => {
    "patronictl_result.stdout_lines": [
        "+ Cluster: k8s-core-psql (7367246950910762117) ----+----+-----------+",
        "| Member  | Host        | Role         | State     | TL | Lag in MB |",
        "+---------+-------------+--------------+-----------+----+-----------+",
        "| sks1263 | 10.0.22.173 | Leader       | running   |  1 |           |",
        "| sks1264 | 10.0.22.174 | Sync Standby | streaming |  1 |         0 |",
        "| sks1265 | 10.0.22.175 | Sync Standby | streaming |  1 |         0 |",
        "+---------+-------------+--------------+-----------+----+-----------+"
    ]
}

TASK [deploy-finish : PostgreSQL Cluster connection info]
    ***********************************************************************************************************************************************
ok: [10.0.22.173] => {
    "msg": [
        "+-----------------------------------------------+",
        "address (VIP) 10.0.22.178",
        "port 5000 (read/write) master",
        "port 5001 (read only)  all replicas",
        "port 5002 (read only)  synchronous replica only",
        "port 5003 (read only)  asynchronous replicas only",
        "+-----------------------------------------------+"
    ]
}

PLAY RECAP
    ***********************************************************************************************************************************************

10.0.22.170                : ok=28   changed=11   unreachable=0    failed=0    skipped=56   rescued=0    ignored=0
10.0.22.171                : ok=28   changed=11   unreachable=0    failed=0    skipped=49   rescued=0    ignored=0
10.0.22.172                : ok=28   changed=11   unreachable=0    failed=0    skipped=49   rescued=0    ignored=0
10.0.22.173                : ok=95   changed=52   unreachable=0    failed=0    skipped=308  rescued=0    ignored=1
```

```
10.0.22.174                : ok=81   changed=46   unreachable=0    failed=0    skipped=294  rescued=0    ignored=1
10.0.22.175                : ok=81   changed=46   unreachable=0    failed=0    skipped=294  rescued=0    ignored=1
10.0.22.176                : ok=39   changed=20   unreachable=0    failed=0    skipped=89   rescued=0    ignored=0
10.0.22.177                : ok=39   changed=20   unreachable=0    failed=0    skipped=83   rescued=0    ignored=0


```

Listing 140: Deploy - Anhang - Deployt

X.III Maintenance

Das `Inventory` bleibt gleich.

Geändert wurde nur das `main.yml`:

```yaml
---
#--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
# Proxy variables (optional) for download packages using a proxy server
proxy_env:
  http_proxy: http://sproxy.sivc.first-it.ch:8080
  https_proxy: http://sproxy.sivc.first-it.ch:8080

# Cluster variables
#cluster_vip: ""  # IP address for client access to the databases in the cluster (optional).
cluster_vip: "10.0.22.178"  # IP address for client access to the databases in the cluster (optional).
vip_interface: "{{ ansible_default_ipv4.interface }}"  # interface name (e.g., "ens32").
# Note: VIP-based solutions such as keepalived or vip-manager may not function correctly in cloud environments like AWS.

#patroni_cluster_name: "postgres-cluster"  # the cluster name (must be unique for each cluster)
patroni_cluster_name: "k8s-core-psql"  # the cluster name (must be unique for each cluster)
patroni_install_version: "3.3.0"  # or 'latest'

patroni_superuser_username: "postgres"
patroni_superuser_password: "<Password Secure / Safe>"
patroni_replication_username: "replicator"
patroni_replication_password: "<Password Secure / Safe>"

#synchronous_mode: false  # or 'true' for enable synchronous database replication
synchronous_mode: true  # or 'true' for enable synchronous database replication
synchronous_mode_strict: false  # if 'true' then block all client writes to the master, when a synchronous replica is not available
#synchronous_node_count: 1  # number of synchronous standby databases
synchronous_node_count: 2  # number of synchronous standby databases

# Load Balancing
with_haproxy_load_balancing: true  # or 'true' if you want to install and configure the load-balancing
haproxy_listen_port:
  master: 5000
  replicas: 5001
  replicas_sync: 5002
  replicas_async: 5003
# The following ('_direct') ports are used for direct connections to the PostgreSQL database,
# bypassing the PgBouncer connection pool (if 'pgbouncer_install' is 'true').
# Uncomment the relevant lines if you need to set up direct connections.
  stats: 7000
haproxy_maxconn:
```

```yaml
  global: 100000
  master: 10000
  replica: 10000
haproxy_timeout:
  client: "60m"
  server: "60m"

# keepalived (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'true')
keepalived_virtual_router_id: "{{ cluster_vip.split('.')[3] | int }}" # The last octet of 'cluster_vip' IP address is used by default.
# virtual_router_id - must be unique in the network (available values are 0..255).

# vip-manager (if 'cluster_vip' is specified and 'with_haproxy_load_balancing' is 'false')
vip_manager_version: "2.4.0"  # version to install
vip_manager_conf: "/etc/patroni/vip-manager.yml"
vip_manager_interval: "1000"  # time (in milliseconds) after which vip-manager wakes up and checks if it needs to register or release ip addresses.
vip_manager_iface: "{{ vip_interface }}"  # interface to which the virtual ip will be added
vip_manager_ip: "{{ cluster_vip }}"  # the virtual ip address to manage
vip_manager_mask: "24"  # netmask for the virtual ip


# DCS (Distributed Consensus Store)
dcs_exists: false  # or 'true' if you don't want to deploy a new etcd cluster
dcs_type: "etcd"  # or 'consul'

# if dcs_type: "etcd" and dcs_exists: false
etcd_version: "3.5.11"  # version for deploy etcd cluster
etcd_data_dir: "/var/lib/etcd"
etcd_cluster_name: "etcd-{{ patroni_cluster_name }}"  # ETCD_INITIAL_CLUSTER_TOKEN

# if dcs_type: "etcd" and dcs_exists: true
patroni_etcd_hosts: []  # list of servers of an existing etcd cluster
patroni_etcd_namespace: "service"  # (optional) etcd namespace (prefix)
patroni_etcd_username: "" # (optional) username for etcd authentication
patroni_etcd_password: "" # (optional) password for etcd authentication
patroni_etcd_protocol: "" # (optional) http or https, if not specified http is used

# if dcs_type: "consul"
consul_version: "1.15.8"
consul_config_path: "/etc/consul"
consul_configd_path: "{{ consul_config_path }}/conf.d"
consul_data_path: "/var/lib/consul"
consul_domain: "consul"  # Consul domain name
consul_datacenter: "dc1"  # Datacenter label (can be specified for each host in the inventory)
consul_disable_update_check: true  # Disables automatic checking for security bulletins and new version releases
consul_enable_script_checks: true  # This controls whether health checks that execute scripts are enabled on this agent
consul_enable_local_script_checks: true  # Enable them when they are defined in the local configuration files
consul_ui: false  # Enable the consul UI?
consul_syslog_enable: true  # Enable logging to syslog
consul_iface: "{{ ansible_default_ipv4.interface }}"  # specify the interface name with a Private IP (ex. "enp7s0")
# TLS
# You can enable TLS encryption by dropping a CA certificate, server certificate, and server key in roles/consul/files/
consul_tls_enable: false
consul_tls_ca_crt: "ca.crt"
consul_tls_server_crt: "server.crt"
```

```yaml
consul_tls_server_key: "server.key"
# DNS
consul_recursors: []  # List of upstream DNS servers
consul_dnsmasq_enable: true  # Enable DNS forwarding with Dnsmasq
consul_dnsmasq_cache: 0  # dnsmasq cache-size (0 - disable caching)
consul_dnsmasq_servers: "{{ nameservers }}" # Upstream DNS servers used by dnsmasq
consul_join: []  # List of LAN servers of an existing consul cluster, to join.

# https://developer.hashicorp.com/consul/docs/discovery/services
consul_services:
  - name: "{{ patroni_cluster_name }}"
    id: "{{ patroni_cluster_name }}-master"
    tags: ['master', 'primary']
    port: "{{ pgbouncer_listen_port }}"  # or "{{ postgresql_port }}" if pgbouncer_install: false
    checks:
      - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/primary", interval: "2s" }
      - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }  # comment out this check if pgbouncer_install: false
  - name: "{{ patroni_cluster_name }}"
    id: "{{ patroni_cluster_name }}-replica"
    tags: ['replica']
    port: "{{ pgbouncer_listen_port }}"
    checks:
      - { http: "http://{{ inventory_hostname }}:{{ patroni_restapi_port }}/replica?lag={{ patroni_maximum_lag_on_replica }}", interval: "2s" }
      - { args: ["systemctl", "status", "pgbouncer"], interval: "5s" }

# PostgreSQL variables
postgresql_version: "16"
# postgresql_data_dir: see vars/Debian.yml or vars/RedHat.yml
postgresql_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }},127.0.0.1" to listen on a specific IP address.
postgresql_port: "5432"
postgresql_encoding: "UTF8"  # for bootstrap only (initdb)
postgresql_locale: "en_US.UTF-8"  # for bootstrap only (initdb)
postgresql_data_checksums: true  # for bootstrap only (initdb)
postgresql_password_encryption_algorithm: "scram-sha-256"  # or "md5" if your clients do not work with passwords encrypted with SCRAM-SHA-256

# (optional) list of users to be created (if not already exists)
postgresql_users:
  - { name: "{{ pgbouncer_auth_username }}", password: "{{ pgbouncer_auth_password }}", flags: "LOGIN", role: "" }
  - { name: "xksgr_sks1160_gitlab", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_sks1172_harbor", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_sks1195_kcsso", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
  - { name: "xksgr_k8s_core_psql_monitor", password: "<Password Secure / Safe>", flags: "LOGIN", role: "pg_monitor" }
  - { name: "xksgr_k8s_core_psql_backup", password: "<Password Secure / Safe>", flags: "SUPERUSER" }
# (optional) list of databases to be created (if not already exists)
#postgresql_databases: []
postgresql_databases:
  - { db: "k8s_core_gitlab_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1160_gitlab" }
  - { db: "k8s_core_harbor_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1172_harbor" }
  - { db: "k8s_core_keycloak_prod", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "xksgr_sks1195_kcsso" }
  - { db: "gramic_test", encoding: "UTF8", lc_collate: "en_US.UTF-8", lc_ctype: "en_US.UTF-8", owner: "postgres" }

# (optional) list of schemas to be created (if not already exists)
postgresql_schemas: []
#  - { schema: "myschema", db: "mydatabase", owner: "mydb-user" }
```

```yaml

# (optional) list of database extensions to be created (if not already exists)
#postgresql_extensions: []
postgresql_extensions:
  - { ext: "pgstattuple", db: "postgres" }
  - { ext: "pgstattuple", db: "k8s_core_gitlab_prod" }
  - { ext: "pgstattuple", db: "k8s_core_harbor_prod" }
  - { ext: "pgstattuple", db: "k8s_core_keycloak_prod" }
  - { ext: "pgstattuple", db: "gramic_test" }

# postgresql parameters to bootstrap dcs (are parameters for example)
postgresql_parameters:
  - { option: "max_connections", value: "500" }
  - { option: "superuser_reserved_connections", value: "5" }
  - { option: "password_encryption", value: "{{ postgresql_password_encryption_algorithm }}" }
  - { option: "max_locks_per_transaction", value: "512" }
  - { option: "max_prepared_transactions", value: "0" }
  - { option: "huge_pages", value: "try" }  # or "on" if you set "vm_nr_hugepages" in kernel parameters
  - { option: "shared_buffers", value: "{{ (ansible_memtotal_mb * 0.25) | int }}MB" }  # by default, 25% of RAM
  - { option: "effective_cache_size", value: "{{ (ansible_memtotal_mb * 0.75) | int }}MB" }  # by default, 75% of RAM
  - { option: "work_mem", value: "128MB" }  # please change this value
  - { option: "maintenance_work_mem", value: "256MB" }  # please change this value
  - { option: "checkpoint_timeout", value: "15min" }
  - { option: "checkpoint_completion_target", value: "0.9" }
  - { option: "min_wal_size", value: "2GB" }
  - { option: "max_wal_size", value: "8GB" }  # or 16GB/32GB
  - { option: "wal_buffers", value: "32MB" }
  - { option: "default_statistics_target", value: "1000" }
  - { option: "seq_page_cost", value: "1" }
  - { option: "random_page_cost", value: "1.1" }  # or "4" for HDDs with slower random access
  - { option: "effective_io_concurrency", value: "200" }  # or "2" for traditional HDDs with lower I/O parallelism
  - { option: "synchronous_commit", value: "on" }  # or 'off' if you can you lose single transactions in case of a crash
  - { option: "autovacuum", value: "on" }  # never turn off the autovacuum!
  - { option: "autovacuum_max_workers", value: "5" }
  - { option: "autovacuum_vacuum_scale_factor", value: "0.01" }  # or 0.005/0.001
  - { option: "autovacuum_analyze_scale_factor", value: "0.01" }
  - { option: "autovacuum_vacuum_cost_limit", value: "500" }  # or 1000/5000
  - { option: "autovacuum_vacuum_cost_delay", value: "2" }
  - { option: "autovacuum_naptime", value: "1s" }
  - { option: "max_files_per_process", value: "4096" }
  - { option: "archive_mode", value: "on" }
  - { option: "archive_timeout", value: "1800s" }
  - { option: "archive_command", value: "cd ." }  # not doing anything yet with WAL-s
  - { option: "wal_level", value: "logical" }
  - { option: "wal_keep_size", value: "2GB" }
  - { option: "max_wal_senders", value: "10" }
  - { option: "max_replication_slots", value: "10" }
  - { option: "hot_standby", value: "on" }
  - { option: "wal_log_hints", value: "on" }
  - { option: "wal_compression", value: "on" }
  - { option: "shared_preload_libraries", value: "pg_stat_statements,auto_explain" }
  - { option: "pg_stat_statements.max", value: "10000" }
  - { option: "pg_stat_statements.track", value: "all" }
  - { option: "pg_stat_statements.track_utility", value: "false" }
```

```yaml
  - { option: "pg_stat_statements.save", value: "true" }
  - { option: "auto_explain.log_min_duration", value: "10s" }  # enable auto_explain for 10-second logging threshold. Decrease this value if necessary
  - { option: "auto_explain.log_analyze", value: "true" }
  - { option: "auto_explain.log_buffers", value: "true" }
  - { option: "auto_explain.log_timing", value: "false" }
  - { option: "auto_explain.log_triggers", value: "true" }
  - { option: "auto_explain.log_verbose", value: "true" }
  - { option: "auto_explain.log_nested_statements", value: "true" }
  - { option: "auto_explain.sample_rate", value: "0.01" }  # enable auto_explain for 1% of queries logging threshold
  - { option: "track_io_timing", value: "on" }
  - { option: "log_lock_waits", value: "on" }
  - { option: "log_temp_files", value: "0" }
  - { option: "track_activities", value: "on" }
  - { option: "track_activity_query_size", value: "4096" }
  - { option: "track_counts", value: "on" }
  - { option: "track_functions", value: "all" }
  - { option: "log_checkpoints", value: "on" }
  - { option: "logging_collector", value: "on" }
  - { option: "log_truncate_on_rotation", value: "on" }
  - { option: "log_rotation_age", value: "1d" }
  - { option: "log_rotation_size", value: "0" }
  - { option: "log_line_prefix", value: "'%t [%p-%l] %r %q%u@%d '" }
  - { option: "log_filename", value: "postgresql-%a.log" }
  - { option: "log_directory", value: "{{ postgresql_log_dir }}" }
  - { option: "hot_standby_feedback", value: "on" }  # allows feedback from a hot standby to the primary that will avoid query conflicts
  - { option: "max_standby_streaming_delay", value: "30s" }
  - { option: "wal_receiver_status_interval", value: "10s" }
  - { option: "idle_in_transaction_session_timeout", value: "10min" }  # reduce this timeout if possible
  - { option: "jit", value: "off" }
  - { option: "max_worker_processes", value: "24" }
  - { option: "max_parallel_workers", value: "8" }
  - { option: "max_parallel_workers_per_gather", value: "2" }
  - { option: "max_parallel_maintenance_workers", value: "2" }
  - { option: "tcp_keepalives_count", value: "10" }
  - { option: "tcp_keepalives_idle", value: "300" }
  - { option: "tcp_keepalives_interval", value: "30" }

# Set this variable to 'true' if you want the cluster to be automatically restarted
# after changing the 'postgresql_parameters' variable that requires a restart in the 'config_pgcluster.yml' playbook.
# By default, the cluster will not be automatically restarted.
#pending_restart: false
pending_restart: true # gramic, maintenance test

# specify additional hosts that will be added to the pg_hba.conf
postgresql_pg_hba:
  - { type: "local", database: "all", user: "{{ patroni_superuser_username }}", address: "", method: "trust" }
  - { type: "local", database: "all", user: "{{ pgbouncer_auth_username }}", address: "", method: "trust" } # required for pgbouncer auth_user
  - { type: "local", database: "replication", user: "all", address: "", method: "trust" }
  - { type: "host", database: "replication", user: "all", address: "127.0.0.1/32", method: "trust" }
  - { type: "host", database: "replication", user: "all", address: "::1/128", method: "trust" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.173/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.174/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.22.175/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "replication", user: "{{ patroni_replication_username }}", address: "10.0.28.16/24", method: "{{ postgresql_password_encryption_algorithm }}" } #
```

```
    gramic, maintenance test
  - { type: "local", database: "all", user: "all", address: "", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "all", address: "127.0.0.1/32", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "all", address: "::1/128", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_gitlab_prod", user: "xksgr_sks1160_gitlab", address: "10.0.20.88/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_harbor_prod", user: "xksgr_sks1172_harbor", address: "10.0.20.92/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "k8s_core_keycloak_prod", user: "xksgr_sks1195_kcsso", address: "10.0.20.98/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.63/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.43/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.20.77/24", method: "{{ postgresql_password_encryption_algorithm }}" }
  - { type: "host", database: "all", user: "{{ patroni_superuser_username }}", address: "10.0.28.16/24", method: "{{ postgresql_password_encryption_algorithm }}" } # gramic,
    maintenance test

# list of lines that Patroni will use to generate pg_ident.conf
postgresql_pg_ident: []

# the password file (~/.pgpass)
postgresql_pgpass:
  - "localhost:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "{{ inventory_hostname }}:{{ postgresql_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "*:{{ pgbouncer_listen_port }}:*:{{ patroni_superuser_username }}:{{ patroni_superuser_password }}"
  - "*:{{ postgresql_port }}:*:{{ patroni_replication_username }}:{{ patroni_replication_password }}"
  - "10.0.20.88:5432:k8s_core_gitlab_prod:xksgr_sks1160_gitlab:<Password Secure / Safe>"
  - "10.0.20.92:5432:k8s_core_harbor_prod:xksgr_sks1172_harbor:<Password Secure / Safe>"
  - "10.0.20.98:5432:k8s_core_keycloak_prod:xksgr_sks1195_kcsso:<Password Secure / Safe>"

# PgBouncer parameters
pgbouncer_install: true  # or 'false' if you do not want to install and configure the pgbouncer service
pgbouncer_processes: 1  # Number of pgbouncer processes to be used. Multiple processes use the so_reuseport option for better performance.
pgbouncer_conf_dir: "/etc/pgbouncer"
pgbouncer_log_dir: "/var/log/pgbouncer"
pgbouncer_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
pgbouncer_listen_port: 6432
pgbouncer_max_client_conn: 10000
pgbouncer_max_db_connections: 1000
pgbouncer_max_prepared_statements: 1024
pgbouncer_default_pool_size: 20
pgbouncer_query_wait_timeout: 120
pgbouncer_default_pool_mode: "session"
pgbouncer_admin_users: "{{ patroni_superuser_username }}"  # comma-separated list of users, who are allowed to change settings
pgbouncer_stats_users: "{{ patroni_superuser_username }}"  # comma-separated list of users who are just allowed to use SHOW command
pgbouncer_ignore_startup_parameters: "extra_float_digits,geqo,search_path"
pgbouncer_auth_type: "{{ postgresql_password_encryption_algorithm }}"
pgbouncer_auth_user: true # or 'false' if you want to manage the list of users for authentication in the database via userlist.txt
pgbouncer_auth_username: pgbouncer # user who can query the database via the user_search function
pgbouncer_auth_password: "<Password Secure / Safe>"
pgbouncer_auth_dbname: "postgres"
pgbouncer_client_tls_sslmode: "disable"
pgbouncer_client_tls_key_file: ""
pgbouncer_client_tls_cert_file: ""
pgbouncer_client_tls_ca_file: ""
pgbouncer_client_tls_protocols: "secure" # allowed values: tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3, all, secure (tlsv1.2,tlsv1.3)
pgbouncer_client_tls_ciphers: "default" # allowed values: default, secure, fast, normal, all (not recommended)

```

```yaml
pgbouncer_pools:
  - { name: "postgres", dbname: "postgres", pool_parameters: "" }
  - { name: "k8s_core_gitlab_prod", dbname: "k8s_core_gitlab_prod", pool_parameters: "" }
  - { name: "k8s_core_harbor_prod", dbname: "k8s_core_harbor_prod", pool_parameters: "" }
  - { name: "k8s_core_keycloak_prod", dbname: "k8s_core_keycloak_prod", pool_parameters: "" }
  - { name: "gramic_test", dbname: "gramic_test", pool_parameters: "" }

# Extended variables (optional)
#patroni_restapi_listen_addr: "0.0.0.0" # Listen on all interfaces. Or use "{{ inventory_hostname }}" to listen on a specific IP address.
patroni_restapi_listen_addr: "{{ inventory_hostname }}" # gramic, maintenance test
patroni_restapi_port: 8008
patroni_ttl: 30
patroni_loop_wait: 10
patroni_retry_timeout: 10
patroni_master_start_timeout: 300
patroni_maximum_lag_on_failover: 1048576 # (1MB) the maximum bytes a follower may lag to be able to participate in leader election.
patroni_maximum_lag_on_replica: "100MB" # the maximum of lag that replica can be in order to be available for read-only queries.

# https://patroni.readthedocs.io/en/latest/yaml_configuration.html#postgresql
patroni_callbacks: []

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#standby-cluster
# Requirements:
# 1. the cluster name for Standby Cluster must be unique ('patroni_cluster_name' variable)
# 2. the IP addresses (or network) of the Standby Cluster servers must be added to the pg_hba.conf of the Main Cluster ('postgresql_pg_hba' variable).
patroni_standby_cluster:
  host: ""  # an address of remote master
  port: "5432"  # a port of remote master
#  primary_slot_name: ""  # which slot on the remote master to use for replication (optional)
#  restore_command: ""  # command to restore WAL records from the remote master to standby leader (optional)
#  recovery_min_apply_delay: ""  # how long to wait before actually apply WAL records on a standby leader (optional)

# Permanent replication slots.
# These slots will be preserved during switchover/failover.
# https://patroni.readthedocs.io/en/latest/dynamic_configuration.html
patroni_slots: []
#  - slot: "logical_replication_slot" # the name of the permanent replication slot.
#    type: "logical" # the type of slot. Could be 'physical' or 'logical' (if the slot is logical, you have to define 'database' and 'plugin').
#    plugin: "pgoutput" # the plugin name for the logical slot.
#    database: "postgres" # the database name where logical slots should be created.
#  - slot: "test_logical_replication_slot"
#    type: "logical"
#    plugin: "pgoutput"
#    database: "test"

patroni_log_destination: stderr  # or 'logfile'
# if patroni_log_destination: logfile
patroni_log_dir: /var/log/patroni
patroni_log_level: info
patroni_log_traceback_level: error
patroni_log_format: "%(asctime)s %(levelname)s: %(message)s"
patroni_log_dateformat: ""
patroni_log_max_queue_size: 1000
patroni_log_file_num: 4
```

```yaml
patroni_log_file_size: 25000000  # bytes
patroni_log_loggers_patroni_postmaster: warning
patroni_log_loggers_urllib3: warning  # or 'debug'

patroni_watchdog_mode: automatic  # or 'off', 'required'
patroni_watchdog_device: /dev/watchdog

patroni_postgresql_use_pg_rewind: true  # or 'false'
# try to use pg_rewind on the former leader when it joins cluster as a replica.

patroni_remove_data_directory_on_rewind_failure: false  # or 'true' (if use_pg_rewind: 'true')
# avoid removing the data directory on an unsuccessful rewind
# if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica.

patroni_remove_data_directory_on_diverged_timelines: false  # or 'true'
# if 'true', Patroni will remove the PostgreSQL data directory and recreate the replica
# if it notices that timelines are diverging and the former master can not start streaming from the new master.

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#bootstrap
patroni_cluster_bootstrap_method: "initdb"  # or "wal-g", "pgbackrest", "pg_probackup"

# https://patroni.readthedocs.io/en/latest/replica_bootstrap.html#building-replicas
patroni_create_replica_methods:
#  - pgbackrest
#  - wal_g
#  - pg_probackup
  - basebackup

pgbackrest:
  - { option: "command", value: "/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore" }
  - { option: "keep_data", value: "True" }
  - { option: "no_params", value: "True" }
wal_g:
  - { option: "command", value: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST" }
  - { option: "no_params", value: "True" }
basebackup:
  - { option: "max-rate", value: "100M" }
  - { option: "checkpoint", value: "fast" }
#  - { option: "waldir", value: "{{ postgresql_wal_dir }}" }
pg_probackup:
  - { option: "command", value: "{{ pg_probackup_restore_command }}" }
  - { option: "no_params", value: "true" }

# "restore_command" written to recovery.conf when configuring follower (create replica)
postgresql_restore_command: ""

# pg_probackup
pg_probackup_install: false  # or 'true'
pg_probackup_install_from_postgrespro_repo: true  # or 'false'
pg_probackup_version: "{{ postgresql_version }}"
pg_probackup_instance: "pg_probackup_instance_name"
pg_probackup_dir: "/mnt/backup_dir"
pg_probackup_threads: "4"
pg_probackup_add_keys: "--recovery-target=latest --skip-external-dirs --no-validate"
```

```yaml
# Ensure there is a space at the beginning of each part to prevent commands from concatenating.
pg_probackup_command_parts:
  - "pg_probackup-{{ pg_probackup_version }}"
  - " restore -B {{ pg_probackup_dir }}"
  - " --instance {{ pg_probackup_instance }}"
  - " -j {{ pg_probackup_threads }}"
  - " {{ pg_probackup_add_keys }}"
pg_probackup_restore_command: "{{ pg_probackup_command_parts | join('') }}"
pg_probackup_patroni_cluster_bootstrap_command: "{{ pg_probackup_command_parts | join('') }}"

# WAL-G
wal_g_install: false  # or 'true'
wal_g_version: "3.0.0"
wal_g_json:  # config https://github.com/wal-g/wal-g#configuration
  - { option: "AWS_ACCESS_KEY_ID", value: "{{ AWS_ACCESS_KEY_ID | default('') }}" }  # define values or pass via --extra-vars
  - { option: "AWS_SECRET_ACCESS_KEY", value: "{{ AWS_SECRET_ACCESS_KEY | default('') }}" }  # define values or pass via --extra-vars
  - { option: "WALG_S3_PREFIX", value: "{{ WALG_S3_PREFIX | default('') }}" } # define values or pass via --extra-vars
  - { option: "WALG_COMPRESSION_METHOD", value: "brotli" }  # or "lz4", "lzma", "zstd"
  - { option: "WALG_DELTA_MAX_STEPS", value: "6" }  # determines how many delta backups can be between full backups
  - { option: "PGDATA", value: "{{ postgresql_data_dir }}" }
  - { option: "PGHOST", value: "{{ postgresql_unix_socket_dir }}" }
  - { option: "PGPORT", value: "{{ postgresql_port }}" }
  - { option: "PGUSER", value: "{{ patroni_superuser_username }}" }
wal_g_archive_command: "wal-g wal-push %p"
wal_g_patroni_cluster_bootstrap_command: "wal-g backup-fetch {{ postgresql_data_dir }} LATEST"

# Define job_parts outside of wal_g_cron_jobs
# Ensure there is a space at the beginning of each part to prevent commands from concatenating.
wal_g_backup_command:
  - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
  - " && wal-g backup-push {{ postgresql_data_dir }} > {{ postgresql_log_dir }}/walg_backup.log 2>&1"
wal_g_delete_command:
  - "[ $(curl -s -o /dev/null -w '%{http_code}' http://{{ inventory_hostname }}:{{ patroni_restapi_port }}) = '200' ]"
  - " && wal-g delete retain FULL 4 --confirm > {{ postgresql_log_dir }}/walg_delete.log 2>&1"

wal_g_cron_jobs:
  - name: "WAL-G: Create daily backup"
    user: "postgres"
    file: /etc/cron.d/walg
    minute: "30"
    hour: "3"
    day: "*"
    month: "*"
    weekday: "*"
    job: "{{ wal_g_backup_command | join('') }}"
  - name: "WAL-G: Delete old backups" # retain 4 full backups (adjust according to your company's backup retention policy)
    user: "postgres"
    file: /etc/cron.d/walg
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "*"
    job: "{{ wal_g_delete_command | join('') }}"
```

```yaml

# pgBackRest
pgbackrest_install: false  # or 'true'
pgbackrest_install_from_pgdg_repo: false  # or 'false'
pgbackrest_stanza: "{{ patroni_cluster_name }}"  # specify your --stanza
pgbackrest_repo_type: "posix"  # or "s3", "gcs", "azure"
pgbackrest_repo_host: ""  # dedicated repository host (optional)
pgbackrest_repo_user: "postgres"
pgbackrest_conf_file: "/etc/pgbackrest/pgbackrest.conf"
# config https://pgbackrest.org/configuration.html
pgbackrest_conf:
  global:  # [global] section
    - { option: "log-level-file", value: "detail" }
    - { option: "log-path", value: "/var/log/pgbackrest" }
    - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
    - { option: "repo1-path", value: "/var/lib/pgbackrest" }
    - { option: "repo1-retention-full", value: "4" }
    - { option: "repo1-retention-archive", value: "4" }
    - { option: "start-fast", value: "y" }
    - { option: "stop-auto", value: "y" }
    - { option: "resume", value: "n" }
    - { option: "link-all", value: "y" }
    - { option: "spool-path", value: "/var/spool/pgbackrest" }
    - { option: "archive-async", value: "y" } # Enables asynchronous WAL archiving (details: https://pgbackrest.org/user-guide.html#async-archiving)
    - { option: "archive-get-queue-max", value: "1GiB" }
  stanza:  # [stanza_name] section
    - { option: "process-max", value: "4" }
    - { option: "log-level-console", value: "info" }
    - { option: "recovery-option", value: "recovery_target_action=promote" }
    - { option: "pg1-socket-path", value: "{{ postgresql_unix_socket_dir }}" }
    - { option: "pg1-path", value: "{{ postgresql_data_dir }}" }
#    - { option: "", value: "" }
# (optional) dedicated backup server config (if "repo_host" is set)
pgbackrest_server_conf:
  global:
    - { option: "log-level-file", value: "detail" }
    - { option: "log-level-console", value: "info" }
    - { option: "log-path", value: "/var/log/pgbackrest" }
    - { option: "repo1-type", value: "{{ pgbackrest_repo_type | lower }}" }
    - { option: "repo1-path", value: "/var/lib/pgbackrest" }
    - { option: "repo1-retention-full", value: "4" }
    - { option: "repo1-retention-archive", value: "4" }
    - { option: "repo1-bundle", value: "y" }
    - { option: "repo1-block", value: "y" }
    - { option: "start-fast", value: "y" }
    - { option: "stop-auto", value: "y" }
    - { option: "resume", value: "n" }
    - { option: "link-all", value: "y" }
    - { option: "archive-check", value: "y" }
    - { option: "archive-copy", value: "n" }
    - { option: "backup-standby", value: "y" }
#    - { option: "", value: "" }
# the stanza section will be generated automatically

```

```yaml
pgbackrest_archive_command: "pgbackrest --stanza={{ pgbackrest_stanza }} archive-push %p"

pgbackrest_patroni_cluster_restore_command:
  '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --delta restore'  # restore from latest backup
#  '/usr/bin/pgbackrest --stanza={{ pgbackrest_stanza }} --type=time "--target=2020-06-01 11:00:00+03" --delta restore'  # Point-in-Time Recovery (example)

# By default, the cron jobs is created on the database server.
# If 'repo_host' is defined, the cron jobs will be created on the pgbackrest server.
pgbackrest_cron_jobs:
  - name: "pgBackRest: Full Backup"
    file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
    user: "postgres"
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "0"
    job: "pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup"
    # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --type=full --stanza={{ pgbackrest_stanza }} backup; fi"
  - name: "pgBackRest: Diff Backup"
    file: "/etc/cron.d/pgbackrest-{{ patroni_cluster_name }}"
    user: "postgres"
    minute: "30"
    hour: "6"
    day: "*"
    month: "*"
    weekday: "1-6"
    job: "pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup"
    # job: "if [ $(psql -tAXc 'select pg_is_in_recovery()') = 'f' ]; then pgbackrest --type=diff --stanza={{ pgbackrest_stanza }} backup; fi"

# PITR mode (if patroni_cluster_bootstrap_method: "pgbackrest" or "wal-g"):
# 1) The database cluster directory will be cleaned (for "wal-g") or overwritten (for "pgbackrest" --delta restore).
# 2) And also the patroni cluster "{{ patroni_cluster_name }}" will be removed from the DCS (if exist) before recovery.

disable_archive_command: true  # or 'false' to not disable archive_command after restore
keep_patroni_dynamic_json: true  # or 'false' to remove patroni.dynamic.json after restore (if exists)


# Netdata - https://github.com/netdata/netdata
netdata_install: false  # or 'true' for install Netdata on postgresql cluster nodes (with kickstart.sh)
netdata_install_options: "--stable-channel --disable-telemetry --dont-wait"
netdata_conf:
  web_bind_to: "*"
  # https://learn.netdata.cloud/docs/store/change-metrics-storage
  memory_mode: "dbengine"  # The long-term metrics storage with efficient RAM and disk usage.
  page_cache_size: 64  # Determines the amount of RAM in MiB that is dedicated to caching Netdata metric values.
  dbengine_disk_space: 1024  # Determines the amount of disk space in MiB that is dedicated to storing Netdata metric values.

...
```

Listing 141: Testsystem - Anhang - Maintenance - main.yml

Nun muss ebenfalls das Playbook ausgeführt werden:

```
ansible-playbook config_pgcluster.yml

```

Listing 142: Testsystem - Anhang - Maintenance - config_pgcluster.yml

X.IV Prequenteries - Erweiterungstests

Auf dem `sks9016` muss zuerst der apt-Proxy gesetzt werden:

```
# 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 143: Testsystem - sks9016 - apt-Proxy Settings

Auch der Proxy muss gesetzt werden: Als nächstes muss der Proxy gesetzt werden:

```
# 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 144: Testsystem - sks9016 - Proxy Settings

Firewall war keine auf dem Host installiert

X.V Haproxy erweitern

Zuerst musste das `Inventory` um den entsprechenden Node erweitert werden:

```
[etcd_cluster]  # recommendation: 3, or 5-7 nodes
10.0.22.170
10.0.22.171
10.0.22.172

# if dcs_exists: false and dcs_type: "consul"
[consul_instances]  # recommendation: 3 or 5-7 nodes

# if with_haproxy_load_balancing: true
[balancers]
10.0.22.176
10.0.22.177
10.0.18.16 new_node=true

# PostgreSQL nodes
[master]
10.0.22.173 postgresql_exists=false

```

```
[replica]
10.0.22.174 postgresql_exists=false
10.0.22.175 postgresql_exists=false

[postgres_cluster:children]
master
replica

# if pgbackrest_install: true and "repo_host" is set
[pgbackrest]  # optional (Dedicated Repository Host)

# Connection settings
[all:vars]
ansible_connection='ssh'
ansible_ssh_port='22'
ansible_user='itgramic'
ansible_ssh_pass='<Secret ;-)>'  # "sshpass" package is required for use "ansible_ssh_pass"
#ansible_ssh_private_key_file=
ansible_python_interpreter='/usr/bin/env python3'

[pgbackrest:vars]


```

Listing 145: Testsystem - Anhang - add_balancer.yml - inventory

Nun muss das Playbook add_balancer.yml ausgeführt werden:

```
ansible-playbook add_balancer.yml

```

Listing 146: Testsystem - Anhang - add_balancer.yml

X.VI Patroni Node

Zuerst musste das `Inventory` um den entsprechenden Node erweitert werden:

```
[etcd_cluster]  # recommendation: 3, or 5-7 nodes
10.0.22.170
10.0.22.171
10.0.22.172

# if dcs_exists: false and dcs_type: "consul"
[consul_instances]  # recommendation: 3 or 5-7 nodes

# if with_haproxy_load_balancing: true
[balancers]
10.0.22.176
10.0.22.177

# PostgreSQL nodes
[master]
10.0.22.173 postgresql_exists=false

[replica]
```

```
10.0.22.174 postgresql_exists=false
10.0.22.175 postgresql_exists=false
10.0.28.16 postgresql_exists=false new_node=true

[postgres_cluster:children]
master
replica

# if pgbackrest_install: true and "repo_host" is set
[pgbackrest]  # optional (Dedicated Repository Host)

# Connection settings
[all:vars]
ansible_connection='ssh'
ansible_ssh_port='22'
ansible_user='itgramic'
ansible_ssh_pass='<Secret ;-)>'  # "sshpass" package is required for use "ansible_ssh_pass"
#ansible_ssh_private_key_file=
ansible_python_interpreter='/usr/bin/env python3'

[pgbackrest:vars]

```

Listing 147: Testsystem - Anhang - add_pgnode.yml - inventory

Das Deployment erfolgt nun ebenfalls via Playbook add_pgnode.yml:

```
ansible-playbook add_pgnode.yml

```

Listing 148: Testsystem - Anhang - add_pgnode.yml

XI Testsystem - Testing

XII Maintenance-Tool

XII.I Maintenance-Tool - Bloated Tables

Das Python-Skript:

```
import os
import psycopg2
from psycopg2.extensions import ISOLATION_LEVEL_AUTOCOMMIT
import yaml
from kubernetes import client, config
import base64
import sys

#   Pass Excption Class
class OneException(Exception):
    pass

#   Read the ConfigMap
#
#   Get the ConfigMap from the Namespace
```

```python
def read_config(confimap):
    # config = os.environ['config-map-connections.yaml']
    script_dir = os.path.dirname(os.path.abspath(__file__))
    configfile = os.path.join(script_dir, confimap)
    config = dict()
    with open(configfile, "r") as file:
        config = yaml.load(file, Loader=yaml.FullLoader)
    print(config)
    return config

#   Open Database Connection
#
#   Opens a Database Connection to the over given Database.
#   Returns the opened Connection.
def database_connection(config, database, user, password):
    # user = config['connectiondata']['username']
    host = config['host']
    port = config['port']
    conn = psycopg2.connect(database=database,
                            user=user,
                            host=host,
                            password=password,
                            port=port)
    return conn

#   Get the Secret
#
#   Get the Secret from the Namespace
def get_secret(secret_name):
    secret = open(secret_name, "r").read()
    print("secret",secret)
    return secret

#   Get the Username
#
#   Get the Username from the Secret from the Namespace
def get_username(secret_name):
    username = open(secret_name, "r").read()
    print("username", username)
    return username

#   Cleanup Bloated Database
#
#   Connected to the Database, check the AUTOVACUUM and VACUUM the Table and
#   REINDEX all Indices of the Table
def cleanup_bloated_database(config):
    #   Get Configuration
    dead_tuple_percent = config['dead_tuple_percent']
    database = config['database']
    secret_username = config['secret_username']
    secret_password = config['secret_password']

    #   Get Secret of the Database
    secret = get_secret(secret_password)
```

```python

    #   Get Username for the Database
    user = get_username(secret_username)

    #   Open Database Connection
    connection = database_connection(config, database, user, secret)

    #   Get the AUTOVACUUM State
    active_autovaacum = check_autovacuum(connection)

    #   Ckeck AUTOVACUUM State
    #
    #   Run Maintenance Job if no AUTOVACUUM is running
    if active_autovaacum == 0:
        #   Get bloated Tables
        bloated_tables = get_bloated_tables(connection, dead_tuple_percent)
        #   Cleanup bloated Tables
        cleanup_bloated_tables(connection, bloated_tables)
    else:
        print('autovacuum maintenance active!')

    #   Close Database Connection
    connection.close()

#   Check AUTOVACUUM
#
#   Check if an AUTOVACUUM Job is running.
#   For this, a Select on the Table pg_stat_activity on all Queries like autovacuum:% is used.
def check_autovacuum(connection):
    cur = connection.cursor()
    sql = "select count(*) as active from pg_stat_activity where query like 'autovacuum:%';"
    cur.execute(sql)
    active_autovaacum = cur.fetchone()
    active_autovaacum = active_autovaacum[0]

    return active_autovaacum

#   Get Bloated Tables
#
#   Select the Bloated Tables.
def get_bloated_tables(connection, dead_tuple_percent):
    cur = connection.cursor()
    sql = """select
    relname as table,
    (pgstattuple(oid)).dead_tuple_percent
from pg_class
where
    relkind = 'r' and
    (pgstattuple(oid)).dead_tuple_percent > %s
order by dead_tuple_percent desc;
        """
    cur.execute(sql, (dead_tuple_percent,))
    return cur

```

```python
#   Cleanup Bloated Tables
#
#   Run through the Bloated Tables and VACUUM and REINDEX
def cleanup_bloated_tables(connection, cur):
    bloated_tables = cur.fetchall()
    for bloated_table in bloated_tables:
        table = bloated_table[0]
        vacuum_table(connection, table)
        reindex_table(connection, table)
    print("finished")

#   VACUUM Table
#
#   VACUUM the Table
def vacuum_table(connection, table):
    old_isolation = connection.isolation_level
    connection.set_isolation_level(ISOLATION_LEVEL_AUTOCOMMIT)
    cur = connection.cursor()
    sql = "vacuum " + table + ";"
    cur.execute(sql)
    connection.set_isolation_level(old_isolation)
    print("vaccum table " + table + " successfully")

#   REINDEX
#
#   REINDEX the whole Table
def reindex_table(connection, table):
    cur = connection.cursor()
    sql = "reindex table " + table + ";"
    cur.execute(sql)
    connection.commit()
    print("reindex table " + table + " successfully")

#   Main Method
def maintenance_bloated_databases(confimap):
    # confimap = confimap_Arg[0]
    try:
        # print(confimap)
        config = read_config(confimap)
        cleanup_bloated_database(config)
    except OneException as e:
        print(repr(e))

arg1 = sys.argv[1]
maintenance_bloated_databases( arg1 )

```

Listing 149: Maintenance-Tool - Bloated Tables / Indices - ksgr_postgresql_maintenance_bloated_tables.py

Zuerst muss der Namespace `ksgr-postgresql-maintenance` erstellt werden:

```
kubectl create namespace ksgr-postgresql-maintenance

```

Listing 150: Maintenance-Tool - Bloated Tables / Indices - Namespace

Das Python-Skript `ksgr_postgresql_maintenance_bloated_tables.py` kann in ein ConfigMap umgewandelt werden:

```
kubectl create configmap ksgr-postgresql-maintenance-bloated-tables --from-file /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/
    ksgr_postgresql_maintenance_bloated_tables.py --dry-run=client -o yaml > /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/
    ksgr_postgresql_maintenance_bloated_tables.yml -n ksgr-postgresql-maintenance

```

Listing 151: Maintenance-Tool - Bloated Tables / Indices - Python > ConfigMap

Entsprechend lässt es sich deployen:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/ksgr_postgresql_maintenance_bloated_tables.yml -n ksgr-postgresql-maintenance

```

Listing 152: Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy

Der Username und das Passwort für den Cluster wird als Secret gespeichert.
Dazu werden erst die Daten in Base64 Strings umgewandelt:

```
USER=postgres
PASSWORD=<Password Secure / Safe>
echo -n ${USER} | base64
echo -n ${PASSWORD} | base64

```

Listing 153: Maintenance-Tool - Bloated Tables / Indices - Base64

Diese werden nun in ein Secret gegeben:

```
apiVersion: v1
kind: Secret
metadata:
  name: secret-vks0041
  namespace: ksgr-postgresql-maintenance
type: Opaque
data:
  user: <Base64 User>
  password: <Base64 Password>

```

Listing 154: Maintenance-Tool - Bloated Tables / Indices - Secret

Das Secret kann deployt werden:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/secret_vks0041.yaml

```

Listing 155: Maintenance-Tool - Bloated Tables / Indices - Secret Deploy

Die ConfigMap `configmap-vks0041-gramic-test` beinhaltet den Hostname, den Port, den Datenbanknamen sowie den Pfad zu den Secrets:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-vks0041-gramic-test
  namespace: ksgr-postgresql-maintenance
data:
  configmap-vks0041-gramic-test.yaml: |-
    host: "10.0.22.178"
```

```
    port: "5000"
    database: "gramic_test"
    secret_username: "/mnt/etc/secret-volume/secret-vks0041/user"
    secret_password: "/mnt/etc/secret-volume/secret-vks0041/password"
    dead_tuple_percent: "1.5"

```

Listing 156: Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test

Das deployment sieht entsprechend aus:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/configmap-vks0041-gramic_test.yaml

```

Listing 157: Maintenance-Tool - Bloated Tables / Indices - configmap-vks0041-gramic-test Deploy

Das finale Deployment erfolgt mittels einem CronJob.
Jeden Tag um Mitternacht wird ein Pod deployt, welcher Skript ausführt.
Dabei müssen erst folgende Dependencis installiert werden:

- psycopg2-binary
- PyYAML
- kubernetes

Normalerweise würden diese PyPi-Packages in einem helm Chart als Ressource mitgegeben werden.
Dann wird das Skript mit dem ConfigMap ausgeführt.
Dazu wird das Secret und das ConfigMap in das Pod-Filesystem gemouted.

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: ksgr-maintenance-bloating-vks0041-gramic-test
  namespace: ksgr-postgresql-maintenance
spec:
  schedule: "0 0 * * *"
  jobTemplate:
    spec:
      template:
        metadata:
          labels:
            workload: cronjob
            type: ksgr-postgresql-maintenance
        spec:
          containers:
            - name: ksgr-maintenance-bloating-vks0041-gramic-test-container
              image: python:3.11-bookworm
              command: ["sh", "-c"]
              args:
                [
                  "python -m pip install --proxy http://sproxy.sivc.first-it.ch:8080 psycopg2-binary PyYAML kubernetes;
                  python /tmp/python/ksgr_postgresql_maintenance_bloated_tables.py /tmp/conf/configmap-vks0041-gramic-test.yaml"
                ]
              volumeMounts:
```

```
            - name: ksgr-postgresql-maintenance-bloated-tables
              mountPath: /tmp/python
            - name: configmap-vks0041-gramic-test
              mountPath: /tmp/conf
            - name: secret-volume
              readOnly: true
              mountPath: /mnt/etc/secret-volume/secret-vks0041

        restartPolicy: Never
        volumes:
          - name: sproxy
            configMap:
              name: sproxy
          - name: ksgr-postgresql-maintenance-bloated-tables
            configMap:
              name: ksgr-postgresql-maintenance-bloated-tables
          - name: configmap-vks0041-gramic-test
            configMap:
              name: configmap-vks0041-gramic-test
          - name: secret-volume
            secret:
              secretName: secret-vks0041

```

Listing 158: Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-test

Das Deployment:

```
kubectl apply -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/cleaned/ksgr-maintenance-bloating-vks0041-gramic_test-cronjob.yaml

```

Listing 159: Maintenance-Tool - Bloated Tables / Indices - ksgr-maintenance-bloating-vks0041-gramic-test Deploy

XII.II Maintenance-Tool - AUTOVACUUM

Das Python-Skript:

```
import os
import sys
from datetime import date

import pandas as pd
import psycopg2
import yaml


#   Pass Excption Class
class OneException(Exception):
    pass

#   Read the ConfigMap
#
#   Get the ConfigMap from the Namespace
def read_config(confimap):
    # config = os.environ['config-map-connections.yaml']
```

```
    script_dir = os.path.dirname(os.path.abspath(__file__))
    configfile = os.path.join(script_dir, confimap)
    config = dict()
    with open(configfile, "r") as file:
        config = yaml.load(file, Loader=yaml.FullLoader)
    print(config)
    return config


#   Open Database Connection
#
#   Opens a Database Connection to the over given Database.
#   Returns the opened Connection.
def database_connection(config, database, user, password):
    # user = config['connectiondata']['username']
    host = config['host']
    port = config['port_ro']
    conn = psycopg2.connect(database=database,
                            user=user,
                            host=host,
                            password=password,
                            port=port)
    return conn


#   Get the Secret
#
#   Get the Secret from the Namespace
def get_secret(secret_name):
    secret = open(secret_name, "r").read()
    return secret

#   Get the Username
#
#   Get the Username from the Secret from the Namespace
def get_username(secret_name):
    username = open(secret_name, "r").read()
    print("username", username)
    return username

#   Get PostgreSQL Parameters
#
#   Get autovacuum_vacuum_scale_factor and autovacuum_vacuum_threshold from pg_settings
#   Get sum of reltuples from pg_class and get the sender_host (Primary) IP Address from pg_stat_wal_receiver
def get_paramaters(connection):
    cur = connection.cursor()
    sql = """select
    (
        select
            setting as autovacuum_vacuum_scale_factor
        from pg_settings
        where
            name = 'autovacuum_vacuum_scale_factor'
    ) as autovacuum_vacuum_scale_factor,
```

```
    (
        select
            setting as autovacuum_vacuum_threshold
        from pg_settings
        where
            name = 'autovacuum_vacuum_threshold'
    ) as autovacuum_vacuum_threshold,
    (
        select
            sum(reltuples) as reltuples
        from pg_class
    ) as reltuples,
    (
        select
            sender_host
        from pg_stat_wal_receiver
    ) as sender_host
;"""
    cur.execute(sql)
    postgresql_parameters = cur.fetchone()
    autovacuum_vacuum_scale_factor = postgresql_parameters[0]
    autovacuum_vacuum_threshold = postgresql_parameters[1]
    reltuples = postgresql_parameters[2]
    sender_host = postgresql_parameters[3]

    postgresql_parameter = dict(autovacuum_vacuum_scale_factor=autovacuum_vacuum_scale_factor,
                                autovacuum_vacuum_threshold=autovacuum_vacuum_threshold, reltuples=reltuples,
                                sender_host=sender_host)

    return postgresql_parameter

#   Calculate and Check AUTOVACUUM
#
#   Calculate needed autovacuum_vacuum_scale_factor and check with the configured ones.
#   Create HTML Table if values are different
def calculate_check_autovacuum(postgres_params, configmap_autovacuum):
    autovacuum_vacuum_scale_factor_actual = float(postgres_params.get('autovacuum_vacuum_scale_factor'))
    autovacuum_vacuum_threshold = float(postgres_params.get('autovacuum_vacuum_threshold'))
    reltuples = float(postgres_params.get('reltuples'))
    sender_host = postgres_params.get('sender_host')
    max_death_tuples = configmap_autovacuum.get('max_death_tuples')
    autovacuum_vacuum_scale_factor = round(((max_death_tuples - autovacuum_vacuum_threshold) / reltuples), 2)

    if autovacuum_vacuum_scale_factor != autovacuum_vacuum_scale_factor_actual:
        result_table = create_pandas_dataframe(autovacuum_vacuum_scale_factor_actual, autovacuum_vacuum_scale_factor, ['autovacuum_vacuum_scale_factor'])
        result_path = configmap_autovacuum.get('calc_result')
        filename_prefix = configmap_autovacuum.get('filename_prefix')
        filename = filename_prefix + str(date.today()) + '.html'
        write_calculating_result(result_table, result_path, filename)

#   Pandas DataFrame
#
#   Create a Pandas DataFrame from the overgiven Parameters
def create_pandas_dataframe(autovacuum_vacuum_scale_factor_actual, autovacuum_vacuum_scale_factor, index):
```

```
    data = {'Bestehend': autovacuum_vacuum_scale_factor_actual, 'Berechnet': autovacuum_vacuum_scale_factor}
    result_df = pd.DataFrame(data, index=index)
    return result_df


#   Write HTML
#
#   Write the Result Pandas DataFrame to HTML Table
def write_calculating_result(result_table, path, filename):
    # html = result_table.to_html(classes='table table-stripped')
    html = result_table.to_html()
    filepath = path + '/' + filename
    text_file = open(filepath, "w")
    text_file.write(html)
    text_file.close()

#   Maintenance Main
#
#   Calculate the AUTOVACUUM Parameters
def maintenance_autovacuum_calculation(confimap_conn, configmap_autovacuum):
    try:
        #   Read the Configs
        config = read_config(confimap_conn)
        config_autovac = read_config(configmap_autovacuum)

        #   Get Configuration
        database = config['database']
        secret_username = config['secret_username']
        secret_password = config['secret_password']

        #   Get Secret of the Database
        secret = get_secret(secret_password)

        #   Get Username for the Database
        user = get_username(secret_username)

        #   Open Database Connection
        connection = database_connection(config, database, user, secret)

        #   Get the PostgreSQL Parameters
        postgres_params = get_paramaters(connection)

        #   Calculate and Check the Autovacuum Vacuum Scale Factor
        calculate_check_autovacuum(postgres_params, config_autovac)

        #   Close Database Connection
        connection.close()

        # cleanup_bloated_database(config)
    except OneException as e:
        print(repr(e))

arg1 = sys.argv[1]
arg2 = sys.argv[2]
```

```
maintenance_autovacuum_calculation(arg1, arg2)

```

Listing 160: Maintenance-Tool - AUTOVACUUM - ksgr_postgresql_maintenance_autovacuum_calculation.py

Das Python-Skript `ksgr_postgresql_maintenance_autovacuum_calculation.py` kann in ein ConfigMap umgewandelt werden:

```
kubectl create configmap ksgr-postgresql-maintenance-autovacuum-calculation --from-file /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/
    ksgr_postgresql_maintenance_autovacuum_calculation.py --dry-run=client -o yaml > /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/
    ksgr_postgresql_maintenance_autovacuum_calculation.yml -n ksgr-postgresql-maintenance

```

Listing 161: Maintenance-Tool - AUTOVACUUM - Python > ConfigMap

Entsprechend lässt es sich deployen:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/ksgr_postgresql_maintenance_autovacuum_calculation.yml -n ksgr-postgresql-maintenance

```

Listing 162: Maintenance-Tool - Bloated Tables / Indices - Python - ConfigMap Deploy

Ein neues Secret muss nicht deployt werden.

Im ConfigMap `configmap-vks0041-postgres` wird der Hostname, den Read-Write Port, der Read-Only Port, die Datenbank und die Secret-Pfade gespeichert:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-vks0041-postgres
  namespace: ksgr-postgresql-maintenance
data:
  configmap-vks0041-postgres.yaml: |-
    host: "10.0.22.178"
    port_rw: "5000"
    port_ro: "5001"
    database: "postgres"
    secret_username: "/mnt/etc/secret-volume/secret-vks0041/user"
    secret_password: "/mnt/etc/secret-volume/secret-vks0041/password"

```

Listing 163: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres

Das deployment sieht entsprechend aus:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/configmap-vks0041-postgres.yaml

```

Listing 164: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-postgres Deploy

Für das AUTOVACUUM wird zudem die ConfigMap `configmap-vks0041-autovacuum` erstellt.
In diesem werden der Pfad zum Node (Host) Filesystem, die Anzahl Toter Tupels und dem Filenamen-Präfix:

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: configmap-vks0041-autovacuum
  namespace: ksgr-postgresql-maintenance
data:
```

```yaml
  configmap-vks0041-autovacuum.yaml: |-
    calc_result: "/mnt/data"
    max_death_tuples: 1000
    filename_prefix: "autovacuum-parameter-"

```

Listing 165: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum

Das deployment sieht entsprechend aus:

```
kubectl create -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/configmap-vks0041-autovacuum.yaml

```

Listing 166: Maintenance-Tool - AUTOVACUUM - configmap-vks0041-autovacuum Deploy

Das finale Deployment erfolgt mittels einem CronJob.
Jeden Tag um Mitternacht wird ein Pod deployt, welcher Skript ausführt.
Dabei müssen erst folgende Dependencis installiert werden:

- psycopg2-binary
- PyYAML
- kubernetes
- pandas

Normalerweise würden diese PyPi-Packages in einem helm Chart als Ressource mitgegeben werden.
Dann wird das Skript mit dem beiden ConfigMaps ausgeführt.
Dazu wird das Secret und die ConfigMaps in das Pod-Filesystem gemouted.

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: ksgr-maintenance-autovacuum-caluclation
  namespace: ksgr-postgresql-maintenance
spec:
  schedule: "0 0 * * *"
  jobTemplate:
    spec:
      template:
        metadata:
          labels:
            workload: cronjob
            type: ksgr-postgresql-maintenance
        spec:
          containers:
            - name: ksgr-maintenance-autovacuum-caluclation-container
              image: python:3.11-bookworm
              command: ["sh", "-c"]
              args:
                [
                  "python -m pip install --proxy http://sproxy.sivc.first-it.ch:8080 psycopg2-binary PyYAML kubernetes pandas;
                  python /tmp/python/ksgr_postgresql_maintenance_autovacuum_calculation.py /tmp/conf/postgres/configmap-vks0041-postgres.yaml /tmp/conf/autovacuum/configmap-
    vks0041-autovacuum.yaml"
```

```
              ]
            volumeMounts:
            - name: ksgr-postgresql-maintenance-autovacuum-calculation
              mountPath: /tmp/python
            - name: configmap-vks0041-postgres
              mountPath: /tmp/conf/postgres
            - name: configmap-vks0041-autovacuum
              mountPath: /tmp/conf/autovacuum
            - name: secret-volume
              readOnly: true
              mountPath: /mnt/etc/secret-volume/secret-vks0041
            - name: ksgr-data
              mountPath: /mnt/data
          restartPolicy: Never
          volumes:
            - name: ksgr-postgresql-maintenance-autovacuum-calculation
              configMap:
                name: ksgr-postgresql-maintenance-autovacuum-calculation
            - name: configmap-vks0041-postgres
              configMap:
                name: configmap-vks0041-postgres
            - name: configmap-vks0041-autovacuum
              configMap:
                name: configmap-vks0041-autovacuum
            - name: ksgr-data
              hostPath:
                path: /mnt/data
            - name: secret-volume
              secret:
                secretName: secret-vks0041

```

Listing 167: Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation

Das Deployment sieht entsprechend aus:

```
kubectl apply -f /home/gramic/PycharmProjects/ksgr_postgresql_maintenance/ksgr_postgresql_maintenance_autovacuum_calculation_cronjob.yaml

```

Listing 168: Maintenance-Tool - AUTOVACUUM - ksgr-maintenance-autovacuum-caluclation Deploy

XIII Monitoring - PRTG

Bevor das Python-Skript abgelegt werden kann, müssen einige PyPI-Packages auf allen Probe-Server installiert werden:

- json
- socket
- requests

☞ Bei PRTG sind die Status folgendermassen aufgeteilt:

0 = OK, 1 = Warning und 2 = Error.
Dies gilt es zu beachten.

Das Python-Skript `KSGR - Patroni - Healthcheck.py` sieht wie folgt aus:

```python
# -*- coding: utf-8 -*-

"""
Created:    2024-05-05

@author:    Michael Graber, Kantonsspital Graubuenden, Switzerland
"""
import json
import socket
import sys

import requests
from paesslerag_prtg_sensor_api.sensor.result import CustomSensorResult
from paesslerag_prtg_sensor_api.sensor.units import ValueUnit


#   node_state = 0  # ok
#   node_state = 1  # warning
#   node_state = 2  # error

def check_patroni_cluster(host, replication_mode, lag_warning, lag_error):
    #   Get the IP
    resolution = socket.gethostbyname_ex(host)
    ip_address_block = resolution[2]
    ip_address = ip_address_block[0]
    print(str(ip_address))

    #   Variable declaration
    patroni_comment = ''
    cluster_state = 2
    cluster_unlocked = 2
    failsafe_mode_is_active = 2
    sync_replication_state = 2
    replication_lage_state = 2

    #   Get the Cluster State
    patroni_cluster_state_url = 'http://' + ip_address + ':8008/patroni'
    patroni_cluster_state_response = requests.get(patroni_cluster_state_url)
    patroni_cluster_state_response_json = patroni_cluster_state_response.json()
    patroni_cluster_state_response_code = patroni_cluster_state_response.status_code

    patroni_cluster_name = patroni_cluster_state_response_json.get('patroni').get('name')
    patroni_cluster_state = patroni_cluster_state_response_json.get('state')
    patroni_role = patroni_cluster_state_response_json.get('role')

    if 'paused' in patroni_cluster_state_response_json:
        patroni_cluster_paused = patroni_cluster_state_response_json.get('paused')
    else:
        patroni_cluster_paused = False
```

```
    if patroni_cluster_state == 'running' and patroni_cluster_paused:
        cluster_state = 1
        patroni_comment = '(Cluster ' + patroni_cluster_name + ' paused)'
    elif not(patroni_cluster_state):
        cluster_state = 2
        patroni_comment = '(Cluster ' + patroni_cluster_name + ' not running)'
    else:
        cluster_state = 0

    if 'cluster_unlocked' in patroni_cluster_state_response_json:
        patroni_cluster_unlocked = patroni_cluster_state_response_json.get('cluster_unlocked')
        if 'failsafe_mode_is_active' in patroni_cluster_state_response_json:
            patroni_failsafe_mode_is_active = patroni_cluster_state_response_json.get('failsafe_mode_is_active')
            if patroni_cluster_unlocked == True and patroni_failsafe_mode_is_active == True:
                patroni_comment += ' (DCS Failsafe Mode)'
                cluster_unlocked = 2
                failsafe_mode_is_active = 2
        else:
                cluster_unlocked = 2
                failsafe_mode_is_active = 0
    else:
        cluster_unlocked = 0
        failsafe_mode_is_active = 0

    #   Get the Cluster Node States
    patroni_nodes_state_url = 'http://' + ip_address + ':8008/cluster'
    patroni_cluster_nodes_response = requests.get(patroni_nodes_state_url)
    patroni_cluster_nodes_response_json = patroni_cluster_nodes_response.json()
    patroni_cluster_nodes_response_code = patroni_cluster_nodes_response.status_code

    replica_count = 0
    sync_replica_count = 0
    if patroni_role == 'master':
        patroni_replication = patroni_cluster_state_response_json.get('replication')
        for patroni_replication_member in patroni_replication:
            patroni_sync_state = patroni_replication_member.get('sync_state')
            if patroni_sync_state == replication_mode:
                sync_replica_count = sync_replica_count + 1
                replica_count = replica_count + 1
    else:
        cluster_members = patroni_cluster_nodes_response_json.get('members')
        for cluster_node in cluster_members:
            node_role = cluster_node.get('role')
            if node_role == 'leader':
                leader_api = cluster_node.get('api_url')
                patroni_leader_state_response = requests.get(leader_api)
                patroni_leader_state_response_json = patroni_leader_state_response.json()
                patroni_leader_state_response_code = patroni_leader_state_response.status_code

                patroni_replication = patroni_leader_state_response_json.get('replication')
                for patroni_replication_member in patroni_replication:
                    patroni_sync_state = patroni_replication_member.get('sync_state')
                    if patroni_sync_state == replication_mode:
```

```
                        sync_replica_count = sync_replica_count + 1
                        replica_count = replica_count + 1

    if sync_replica_count == sync_replica_count:
        sync_replication_state = 0
    elif sync_replica_count > 0:
        sync_replication_state = 1
        patroni_comment += ' (Not all Replicas run in sync standby mode)'
    else:
        sync_replication_state = 2
        patroni_comment += ' (None of the Replicas run in sync standby mode)'

    #   Get the Replication lag warning
    lag_warning_url = 'http://' + ip_address + ':8008//replica?lag=' + lag_warning
    lag_warning_response = requests.get(lag_warning_url)
    lag_warning_response_code = lag_warning_response.status_code

    #   Get the Replication lag error
    lag_error_url = 'http://' + ip_address + ':8008//replica?lag=' + lag_error
    lag_error_response = requests.get(lag_error_url)
    lag_error_response_code = lag_error_response.status_code

    if patroni_role == 'master':
        replication_lage_state = 0
    else:
        if lag_warning_response_code != 200:
            replication_lage_state = 1
            patroni_comment += ' (replication lag over ' + lag_warning + ')'
        elif lag_error_response_code != 200:
            replication_lage_state = 2
            patroni_comment += ' (replication lag over ' + lag_error + ')'
        else:
            replication_lage_state = 0

    csr = CustomSensorResult(text="This sensor runs on %s" % data["host"])

    #   Patroni Cluster State
    channel_name = "PATRONI CLUSTER - " + patroni_cluster_name
    csr.add_primary_channel(name=patroni_cluster_name,
                            value=cluster_state,
                            unit=ValueUnit.CUSTOM,
                            is_float=False,
                            is_limit_mode=True,
                            limit_max_warning=0.75,
                            limit_max_error=1.5,
                            limit_warning_msg=channel_name + " Warning",
                            limit_error_msg=channel_name + " Error")

    channel_name = "CLUSTER UNLOCK"
    csr.add_channel(name=channel_name,
                    value=cluster_unlocked,
                    unit=ValueUnit.CUSTOM,
                    is_float=False,
                    is_limit_mode=True,
```

```
                    limit_max_warning=0.75,
                    limit_max_error=1.5,
                    limit_warning_msg=channel_name + " Warning",
                    limit_error_msg=channel_name + " Error")

    channel_name = "FAILSAFE MODE ACTIVITY"
    csr.add_channel(name=channel_name,
                    value=failsafe_mode_is_active,
                    unit=ValueUnit.CUSTOM,
                    is_float=False,
                    is_limit_mode=True,
                    limit_max_warning=0.75,
                    limit_max_error=1.5,
                    limit_warning_msg=channel_name + " Warning",
                    limit_error_msg=channel_name + " Error")

    channel_name = "REPLICATION STATE"
    csr.add_channel(name=channel_name,
                    value=sync_replication_state,
                    unit=ValueUnit.CUSTOM,
                    is_float=False,
                    is_limit_mode=True,
                    limit_max_warning=0.75,
                    limit_max_error=1.5,
                    limit_warning_msg=channel_name + " Warning",
                    limit_error_msg=channel_name + " Error")

    channel_name = "REPLICATION LAG"
    csr.add_channel(name=channel_name,
                    value=replication_lage_state,
                    unit=ValueUnit.CUSTOM,
                    is_float=False,
                    is_limit_mode=True,
                    limit_max_warning=0.75,
                    limit_max_error=1.5,
                    limit_warning_msg=channel_name + " Warning",
                    limit_error_msg=channel_name + " Error")

    #   Overwrite the Sensor Text and create a custom Warning/Error Message
    patroni_comment = patroni_comment.strip()
    csr.text = "This sensor runs on %s" % data["host"] + ' ' + patroni_comment

    return csr

if __name__ == "__main__":
    try:
        #   Get Parameters
        data1 = sys.argv[1]
        data = json.loads(data1)
        data_overgiven = json.loads(data1)

        #   Get the overgiven Parameter String and cast to a Dictionary
        params = data_overgiven["params"]
        perams = dict(param.split('=') for param in params.split(','))
```

```python

        #   Get the used Binary
        replication_mode = perams["replication_mode"]
        lag_warning = perams["lag_warning"]
        lag_error = perams["lag_error"]

        #   Call Function / Method
        csr = check_patroni_cluster(data["host"], replication_mode, lag_warning, lag_error)

        #   Return the Value
        print(csr.json_result)

    except Exception as e:
        csr = CustomSensorResult(text="Python Script execution error")
        csr.error = "Python Script execution error: %s" % str(e)
        print(csr.json_result)

```

Listing 169: Monitoring - KSGR - Patroni - Healthcheck.py

Das Python-Skript muss auf allen PRTG Probes, welches die entsprechenden Sensoren ausführt, in folgendes Verzeichnis angelegt werden:

XIV Exkurs Architekturen - Umsysteme und Prinzipien

XIV.I Raft-Consensus

Raft ist eine kombination aus einer Log Replikation und einer State Machine[92].
Ein Node kann dabei einen von drei States einnehmen:

Leader
: Empfängt als einziger Requests von den Clients und hat die Hoheit über die Transaktionen im Log

Canditate
: Wenn ein Leader ausgefallen ist oder für einen Follower nicht mehr innerhalb eines zufälligen Timeouts erreichbar ist,
ernennt dieser sich zum Kandidaten als neuer Leader und wählt sich selbst.
Dazu versendet er alle seine Informationen inklusive seinem Log.
Gewählt wird dann jener Candidate, der ein konsistentes, vollständiges und jüngste Log besitzt.

Follower
: Follower können nur Commands empfangen und Response Messages senden.

Das Prozedere für die Bestimmung eines Leaders kann viele Ursachen haben.
Etwa weil ein System neu gestartet wird oder ein Leader nicht mehr verfügbar oder erreichbar ist.
Dann werden folgende Schritte vorgenommen:

1. Alle Nodes starten als Follower und warten auf einen Leader
2. Wenn kein Leader vorhanden ist, startet mindestens ein Follower die elektion eines Leaders.
   Mindetsens eine Request Vote wird versendet.
3. Die Follower wählen dann ihren neuen Leader aus.

4. Der neue Leader sendet nun Append Entries als Heartbeat-Message und anweisungen an die Follower, Log Einträge zu replizieren.

Das zufällige Heartbeat-Timeout der Follower verhindert dabei,
dass sich alle Follower gleichzeitig als Candidate ausrufen und sich selber Wählen.
Sonst hätte man eine Split-Brain Situation.

Ist ein Leader aktiv, wird der Log konsens wie folgt ausgeführt:

1. Der Leader erhält einen Requet von einem Client, Daten zu schreiben
2. DFer Leader legt dieses als neuen Eintrag ein
3. Der Leader sendet Append Entry Messages zu den Followern
4. Eine Mehrheit der Follower müssen nun bestätigen,
   das sie diese erhalten und in ihr Log geschrieben haben.
5. Der Leader schreibt nun ein Commit in seine State Machine
6. Der Leader sendet nun Append Entry Messages, dass er die Einträge commited hat.
   Der Leader befiehlt den Followern, es ihm gleichzutun
7. Nun ist eine Konsistente Transaktion auf den Cluster geschrieben

Dem Log wird immer der Index des vorangegangenen Eintrags gesendet.
Fällt ein Follower aus und wird später wieder dem Quorum hinzugefügt,
meldet er einen Fehler wenn er neue Daten erhält, ihm fehlen ja Einträge.
Der Leader sendet dem Follower also den Eintrag.
Fehlen weitere vorangehende Einträge, wird der Vorgang solange ausgeführt,
bis wieder ein Konsistenter Stand erreicht ist.
Das System ist so Selbstheilend.

XIV.II local-path-provisioner

local-path-provisioner wartet auf einen PVC (Persistence Volume Claim).
Sobald ein Pod existiert, wird auf dem Worker Node ein Volume erzeugt, welches Normalerweise unter dem Pfad `var/volumes` auf dem Node zu finden ist.
Anschliessend wird ein PV (Pesistence Volume) und dem Volume auf dem Node angefügt.
Anschliessend wird das PV an den PVC gebunden:

Abbildung L: local-path-provisioner[36]

XV Python Utils

XV.I zotero.py

```python
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

        if zotero_item_type == 'thesis':
            master_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('masterthesis')
            phd_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('phdthesis')

            # First Master thesis
            if any(item in bibtex_item_titel for item in master_list):
                bibtex_item_key = 'masterthesis'
            # Second PHD Thesis
            elif any(item in bibtex_item_titel for item in phd_list):
                bibtex_item_key = 'phdthesis'
            else:
                bibtex_item_key = 'masterthesis'
        else:
            if zotero_bibtex_keys.get(zotero_item_type).get('key'):
                bibtex_item_key = zotero_bibtex_keys.get(zotero_item_type).get('key')
            else:
                bibtex_item_key = 'misc'

        # get all Keys for the zotero item type
        entryset = '\n'
        entry = ''

        zotero_item_attributes = zotero_bibtex_keys.get(zotero_item_type).get('attributes').keys()
        item_attributes = sorted(zotero_item_attributes, reverse=True)

        for index, item_attribute in enumerate(item_attributes):
            bibtex_item_attribute = zotero_bibtex_keys.get(zotero_item_type).get('attributes').get(item_attribute)
            zotero_item_value = biblio_item.get(item_attribute)
            zotero_item_value_extra = ''
            bibtex_item_attribute_extra = ''

            # Special Cases
            if bibtex_item_attribute == 'SPECIALCHECK' and zotero_item_value not in ['', None]:
                bibtex_special_attribute = zotero_bibtex_attributes_special.get(item_attribute)

                match bibtex_special_attribute:
                    case 'get_dates':
                        zotero_item_value = get_dates(zotero_item_value, bibtex_item_key, bibtex_month_attributes)
                        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 170: Python LaTex - zotero.py - Zotero BibLaTex Importer

XV.II zotero_bibtex_configuration.yaml

```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 171: Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer

XV.III zotero_biblatex_keystore.yaml

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

```yaml
    publisher: publisher
    place: address
bookSection:
  key: inbook
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    pages: pages
    publisher: publisher
    place: address
    bookTitle: booktitle
case:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
conferencePaper:
  key: inproceedings
  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
```

```yaml
    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
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
```

```yaml
    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
```

```yaml
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
podcast:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
presentation:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
radioBroadcast:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
report:
  techreport: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
software:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
computerProgram:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
statute:
  key: misc
```

```yaml
  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:
  key: misc
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    url: howpublished
    extra: note
```

```yaml
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 172: Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer

XV.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

#   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)

    # This example is for a 5 * 5 matrix
    nrows = 5
    ncols = 5
    axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0, nrows) for c in range(0, ncols)]

    # remove the x and y ticks
    for ax in axes:
        ax.set_xticks([])
        ax.set_yticks([])
        ax.set_xlim(0, 5)
        ax.set_ylim(0, 5)

    # Add background colors
    # This has been done manually for more fine-grained control
    # Run the loop below to identify the indice of the axes
    for _ in green:
        axes[_].set_facecolor('green')

    for _ in yellow:
        axes[_].set_facecolor('yellow')

    for _ in orange:
        axes[_].set_facecolor('orange')

    for _ in red:
        axes[_].set_facecolor('red')

    # run throuh datas and generate axis datas
    dict_bubble_axis = dict()
    bubble_axis = list()
    for datasets in datas:
        # get the datas
        riskid = datas.get(datasets).get('risk-id')
        x_axis = int(datas.get(datasets).get('x-axis'))
        y_axis = int(datas.get(datasets).get('y-axis'))
```

```python
            axis_point = matrix.get((x_axis, y_axis))
            x_axis_text = float(datas.get(datasets).get('x-axis-text'))
            y_axis_text = float(datas.get(datasets).get('y-axis-text'))
            x_axis_bubble = float(datas.get(datasets).get('x-axis-bubble'))
            y_axis_bubble = float(datas.get(datasets).get('y-axis-bubble'))
            bubble_axis.append(axis_point)

            # merge riks if two or more risks share the same axispoint
            if dict_bubble_axis.get(axis_point):
                risktag = dict_bubble_axis.get(axis_point).get('risk')
                risktag = risktag + '/' + riskid
                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 173: Python LaTex - riskmatrix.py - Risxikomatrizen

XV.V riskmatrix_plotter_conf.yaml

```yaml
risk_inventory:
  - "postgresql"
  - "project"
```

```yaml
  - "Postgresql-massnahme"
  - "Project-massnahme"
riskmatrix:
  startpath: "parentdir"
  configfile_path: "source/configuration"
  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"
    imagename: "riskmatrix-project.png"
```

```yaml
    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)"
    title: "Risiko Cockpit PostgreSQL Datenbanken KSGR - Massnahme"
    bubble-standard-size: 1000
    settings:
      green-boxes:
        - 10
        - 15
        - 16
```

```yaml
        - 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
        - 16
        - 20
        - 21
      yellow-boxes:
        - 0
        - 5
        - 6
        - 11
        - 17
        - 22
        - 23
      orange-boxes:
        - 1
```

```yaml
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
```

Listing 174: Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen

XV.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 175: Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples

XV.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
```

```python
    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 176: Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm

XV.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 177: Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm

XV.IX pandas_dataframe_to_latex_table.py

```python
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:
    #         reader = csv.reader(file)
    #         data = list(reader)
    #
    #     # panda_table_data = pd.DataFrame(data, columns=data[0])
    #     # panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding, lineterminator='\n', engine='python')
    #     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding, lineterminator='\n')
    #     df_dtype = {
    #         "Nr.": int,
    #         "Anforderung": str,
    #         "Beschreibung": str,
    #         "System": str,
    #         "Muss / Kann": str
    #     }
    #     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, lineterminator='\n', dtype=df_dtype)
    #     # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding)
    # else:
    #     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, low_memory=False, engine='python')
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, engine='python', dtype='unicode')
    # readed = open(data_path, 'r', encoding=encoding)
    # panda_table_data = pd.read_csv(open(data_path, 'r', encoding=encoding), sep=",", decimal=".", encoding=encoding)
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding = "ISO-8859-1")
    # panda_table_data = pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, chunksize=10)

    # for chunk in pd.read_csv(data_path, sep=",", decimal=".", encoding=encoding, chunksize=5):
    #     print(chunk)
    # panda_table_data = pd.DataFrame()
    # temp = pd.read_csv(data_path, iterator=True, sep=",", decimal=".", encoding=encoding, chunksize=1000)
    # panda_table_data = pd.concat(temp, ignore_index=True)

    df_dtype = {
        "Nr.": int,
        "Anforderung": str,
        "Beschreibung": str,
        "System": str,
        "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)
        panda_table_data = get_data(startpath, destination, tablefilename, datafile_path, datafile, alternative_cvs_load, separator, decimal)

        # filter by where clausel
        if where_clausel:
            panda_table_data = panda_table_data.query(where_clausel)

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

        # set aggregation functions
        # if groupby_values and not agg_funtion and not pivot_column and not pivot_table_column:
        if groupby_values and not (pivot_column or (pivot_table_column or pivot_table_value or pivot_table_indizes)):
            match group_by_function:
                case 'max':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).max()
                case 'min':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).min()
                case 'head':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).head()
                case 'sum':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).sum()
                case 'mean':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).mean()
```

```
        else:
            panda_table_data = panda_table_data

        # pivot if pivot is selected
        if pivot_table_column or pivot_table_value or pivot_table_indizes:
            if type(pivot_table_agg_function) is list:
                agg_tuple = tuple(pivot_table_agg_function)
                panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
                                                  columns=pivot_table_column, values=pivot_table_value,
                                                  aggfunc=agg_tuple, margins=margin, margins_name=margin_name)
            elif type(pivot_table_agg_function) is dict:
                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)

```

```python
        # 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 178: Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle

XV.X csv_to_latex_diplomarbeit.yaml

```yaml
tables_inventory:
 - "db_inventory"
 - "db_inventory_per_rdbms"
 - "db_inventory_per_os"
 - "anforderungskatalog"
 - "arbeitsrapport"
 - "projektcontrolling"
```

```yaml
 - "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: ","
  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"
```

```yaml
  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"
```

```yaml
  - "eol"
  - "age"
  - "eol_since"
  - "comment"
 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:
  rdbms: "RDBMS"
  instance : "Instanz"
  databases : "Datenbanken"
  appliance: "Appliance"
 where_clausel:
 sorting:
  order_by:
   - "rdbms"
  sort_acending: True
  sort_inplace: True
 margins:
  margin: True
  margin_name: "Gesamtergebnis"
 table_styles:
  selector: "caption"
  props:
   caption-side: "below"
   position: "H"
   sparse_columns: True
   longtable: False
   resize_textwidth: False
   linebreak_columns:
   table_header: True
db_inventory_per_os:
 id: "db_inventory_per_os"
 isbigfile:
 has_longtexts: False
 separator: ","
 decimal: "."
 caption: "Datenbankinventor - Nach Betriebssystemen üaufgeschlsselt"
 label: "db_inventory_per_os"
 startpath: "parentdir"
```

```
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "inventory.csv"
  tablefilename: "db_inventory_per_os.tex"
  decimal_format:
  group_by:
   - "rdbms"
   - "os"
  group_by_function: "sum"
  agg_funtion:
  agg_colums:
   - "os"
  drop_columns:
   - "server"
   - "version"
   - "releasedate"
   - "eol"
   - "age"
   - "eol_since"
   - "comment"
#   - "appliance"
  column_operations:
   datas:
   operations:
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
    pivot_indizes:
     - "os"
     - "rdbms"
    pivot_indizes_visible: True
    pivot_rename_indizes:
     os: "OS"
     rdbms: "RDBMS"
   pivot_columns:
   pivot_values:
   pivot_agg_func:
    instance: "sum"
    databases: "sum"
    appliance: "sum"
   transpose: True
  rename_columns:
   rdbms: "RDBMS"
   instance : "Instanz"
   databases : "Datenbanken"
   os : "OS"
   appliance: "Appliance"
  where_clausel:
```

```yaml
  sorting:
   order_by:
   sort_acending: False
   sort_inplace: True
  margins:
   margin: True
   margin_name: "Gesamtergebnis"
  table_styles:
   selector: "caption"
   props:
    caption-side: "below"
    position: "H"
    sparse_columns: True
    longtable: True
    resize_textwidth: False
    linebreak_columns:
    table_header: True
 anforderungskatalog:
  id: "anforderungskatalog"
  isbigfile:
  has_longtexts: True
  separator: ";"
  decimal: "."
  caption: "Anforderungskatalog"
  label: "anforderungskatalog"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "anforderungskatalog.CSV"
  tablefilename: "anforderungskatalog.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
  column_operations:
   datas:
   operations:
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
    pivot_indizes_visible: False
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
```

```yaml
  where_clausel:
  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: False
    longtable: False
    resize_textwidth: True
    linebreak_columns:
     - "Beschreibung"
    table_header: True
 arbeitsrapport:
  id: "arbeitsrapport"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "Arbeitsrapport"
  label: "arbeitsrapport"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "arbeitsrapport.CSV"
  tablefilename: "arbeitsrapport.tex"
  decimal_format: "{:0.1f}"
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
   - "Hide"
   - "Geplante Dauer [h]"
   - "dauer_summe"
  column_operations:
   datas:
   operations:
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
```

```yaml
    pivot_indizes_visible: False
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
  where_clausel: "Hide == 0"
  sorting:
   order_by:
    - "Datum"
    - "Von"
   sort_acending: False
   sort_inplace: False
  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:
     - "äTtigkeit"
     - "Bemerkung"
     - "Schwierigkeit"
     - "öLsungen"
    table_header: True
 projektcontrolling:
  id: "projektcontrolling"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "Projektcontrolling"
  label: "projektcontrolling"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "arbeitsrapport.CSV"
  tablefilename: "projektcontrolling.tex"
  decimal_format: "{:0.1f}"
  group_by:
   - "Phase"
   - "Subphase"
  group_by_function: "sum"
  agg_funtion:
  agg_colums:
     - "Dauer [h]"
     - "Geplante Dauer [h]"
     - "dauer_summe"
  drop_columns:
```

```yaml
  - "Datum"
  - "Von"
  - "Bis"
  - "Hide"
  - "äTtigkeit"
  - "Bemerkung"
  - "Schwierigkeit"
  - "öLsungen"
 column_operations:
  datas:
   - "dauer_summe"
  operations:
   dauer_summe:
    operation_function: "diff"
    axis_number: 1
    columns:
    - "Dauer [h]"
    - "Geplante Dauer [h]"
 pivot:
  pivot_columns:
  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
```

```yaml
 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:
 column_operations:
  datas:
  operations:
   dauer_summe:
    operation_function:
    axis_number:
    columns:
 pivot:
  pivot_columns:
  pivot_values:
 pivot_table:
  pivot_index:
   pivot_indizes_visible: False
   pivot_rename_indizes:
  pivot_columns:
  pivot_values:
  pivot_agg_func:
 rename_columns:
 where_clausel:
 sorting:
  order_by:
   - "Server"
   - "Typ"
  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
dependencis:
```

```yaml
id: "dependencis"
isbigfile:
has_longtexts: False
separator: ";"
decimal: "."
caption: "äAbhngigkeiten"
label: "dependencis"
startpath: "parentdir"
destination_path: "content/latex_tables"
datafile_path: "source/tables"
datafile: "dependencis.csv"
tablefilename: "dependencis.tex"
decimal_format:
group_by:
group_by_function:
agg_funtion:
agg_colums:
drop_columns:
column_operations:
 datas:
 operations:
  dauer_summe:
   operation_function:
   axis_number:
   columns:
pivot:
 pivot_columns:
 pivot_values:
pivot_table:
 pivot_index:
  pivot_indizes_visible: False
  pivot_rename_indizes:
 pivot_columns:
 pivot_values:
 pivot_agg_func:
rename_columns:
where_clausel:
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:
```

```yaml
    - "äAbhngigkeit"
    - "Beschreibung"
    - "Status"
    - "Risiko"
    - "Impact"
   table_header: True
predecision_out:
 id: "predecision_out"
 isbigfile:
 has_longtexts: False
 separator: ";"
 decimal: "."
 caption: "Vorauswahl - Ausgeschieden"
 label: "predecision_out"
 startpath: "parentdir"
 destination_path: "content/latex_tables"
 datafile_path: "source/tables"
 datafile: "pre-decision.csv"
 tablefilename: "pre-decision-out.tex"
 decimal_format:
 group_by:
 group_by_function:
 agg_funtion:
 agg_colums:
 drop_columns:
  - "hide_state"
 column_operations:
  datas:
  operations:
   dauer_summe:
    operation_function:
    axis_number:
    columns:
 pivot:
  pivot_columns:
  pivot_values:
 pivot_table:
  pivot_index:
   pivot_indizes_visible: False
   pivot_rename_indizes:
  pivot_columns:
  pivot_values:
  pivot_agg_func:
 rename_columns:
 where_clausel: "hide_state == 1"
 sorting:
  order_by:
   - "Nr."
  sort_acending: True
  sort_inplace: True
 margins:
  margin: False
  margin_name:
 table_styles:
```

```yaml
  selector: "caption"
  props:
   caption-side: "below"
   position: "H"
   sparse_columns: True
   longtable: False
   resize_textwidth: True
   linebreak_columns:
    - "üBegrndung"
   table_header: True
predecision_in:
 id: "predecision_in"
 isbigfile:
 has_longtexts: False
 separator: ";"
 decimal: "."
 caption: "Vorauswahl - Evaluation"
 label: "predecision_in"
 startpath: "parentdir"
 destination_path: "content/latex_tables"
 datafile_path: "source/tables"
 datafile: "pre-decision.csv"
 tablefilename: "pre-decision-in.tex"
 decimal_format:
 group_by:
 group_by_function:
 agg_funtion:
 agg_colums:
 drop_columns:
  - "hide_state"
 column_operations:
  datas:
  operations:
   dauer_summe:
    operation_function:
    axis_number:
    columns:
 pivot:
  pivot_columns:
  pivot_values:
 pivot_table:
  pivot_index:
   pivot_indizes_visible: False
   pivot_rename_indizes:
  pivot_columns:
  pivot_values:
  pivot_agg_func:
 rename_columns:
 where_clausel: "hide_state == 2"
 sorting:
  order_by:
   - "Nr."
  sort_acending: True
  sort_inplace: True
```

```yaml
  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:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
    pivot_indizes_visible: False
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
  where_clausel:
  sorting:
   order_by:
```

```yaml
    - "Woche"
   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:
     - "Beschreibung / Event / Problem"
    table_header: True
evaluation_distributed_sql:
 id: "evaluation_distributed_sql"
 isbigfile:
 has_longtexts: False
 separator: ";"
 decimal: "."
 caption: "Evaluationssysstem - Distributed SQL / Sharding"
 label: "evaluation_distributed_sql"
 startpath: "parentdir"
 destination_path: "content/latex_tables"
 datafile_path: "source/tables"
 datafile: "evaluation_platform_distributed_sql.csv"
 tablefilename: "evaluation_platform_distributed_sql.tex"
 decimal_format:
 group_by:
 group_by_function:
 agg_funtion:
 agg_colums:
 drop_columns:
 column_operations:
  datas:
  operations:
   dauer_summe:
    operation_function:
    axis_number:
    columns:
 pivot:
  pivot_columns:
  pivot_values:
 pivot_table:
  pivot_index:
   pivot_indizes_visible: False
   pivot_rename_indizes:
  pivot_columns:
  pivot_values:
  pivot_agg_func:
 rename_columns:
```

```yaml
 where_clausel:
 sorting:
  order_by:
  sort_acending: False
  sort_inplace: False
 margins:
  margin: False
  margin_name:
 table_styles:
  selector: "caption"
  props:
   caption-side: "below"
   position: "H"
   sparse_columns: True
   longtable: False
   resize_textwidth: False
   linebreak_columns:
   table_header: False
expert_discussions_overview:
  id: "expert_discussions_overview"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "äFachgesprche"
  label: "expert_discussions_overview"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "expert_discussions.csv"
  tablefilename: "expert_discussions_overview.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
     - "Fragen"
     - "Antworten"
     - "Sonstige Themen"
  column_operations:
   datas:
   operations:
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   pivot_index:
    pivot_indizes_visible: False
    pivot_rename_indizes:
```

```yaml
    pivot_columns:
    pivot_values:
    pivot_agg_func:
  rename_columns:
  where_clausel:
  sorting:
    order_by:
      - "äFachgesprch"
    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:
        - "Studenten"
        - "Bemerkungen"
      table_header: True
expert_discussions_full_list:
  id: "expert_discussions_full_list"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "äFachgesprche - Protokoll"
  label: "expert_discussions_full_list"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "expert_discussions.csv"
  tablefilename: "expert_discussions_full_list.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
  column_operations:
    datas:
    operations:
      dauer_summe:
        operation_function:
        axis_number:
        columns:
  pivot:
    pivot_columns:
    pivot_values:
```

```yaml
  pivot_table:
   pivot_index:
    pivot_indizes_visible: False
    pivot_rename_indizes:
   pivot_columns:
   pivot_values:
   pivot_agg_func:
  rename_columns:
  where_clausel:
  sorting:
   order_by:
    - "äFachgesprch"
   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:
     - "Studenten"
     - "Fragen"
     - "Antworten"
     - "Sonstige Themen"
     - "Bemerkungen"
    table_header: True
stakeholder:
  id: "stakeholder"
  isbigfile:
  has_longtexts: False
  separator: ";"
  decimal: "."
  caption: "Stakeholder"
  label: "stakeholder"
  startpath: "parentdir"
  destination_path: "content/latex_tables"
  datafile_path: "source/tables"
  datafile: "stakeholder.csv"
  tablefilename: "stakeholder.tex"
  decimal_format:
  group_by:
  group_by_function:
  agg_funtion:
  agg_colums:
  drop_columns:
  column_operations:
   datas:
   operations:
```

```
    dauer_summe:
     operation_function:
     axis_number:
     columns:
  pivot:
   pivot_columns:
   pivot_values:
  pivot_table:
   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 179: Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle

XV.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)
```

```python

    return panda_diagram_plotter_config

def get_data(startpath, destination, tablefilename, datafile_path, datafile, 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")

    print(datafile, ':', encoding)
    panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal, encoding=encoding)
    # return data
    return panda_table_data
def create_panda_diagram_plotter(panda_diagram_plotter_config):
    pdp_tables = panda_diagram_plotter_config.get('diagram_inventory')
    for table_item in pdp_tables:
        print(table_item)
        startpath = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('startpath')
        destination = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('destination_path')
        imagename = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('imagename')
        datafile_path = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('datafile_path')
        datafile = panda_diagram_plotter_config.get('panda_diagram_plotter').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)
        image_path = os.path.join(directory, imagename)
        separator = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('separator')
        decimal = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('decimal')

        # column operations
        column_operations = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('datas')

        # group by / aggregation
        groupby_values = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('group_by')
        group_by_function = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('group_by_function')

        agg_funtion = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('agg_funtion')
        agg_colums = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('agg_colums')
        # dropping and renaming columns
        drop_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('drop_columns')
        rename_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('rename_columns')

        # table filtering and sorting
        where_clausel = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('where_clausel')
        order_by = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('sorting').get('order_by')
```

```
        sort_acending = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('sorting').get('sort_acending')
        sort_inplace = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('sorting').get('sort_inplace')

        # pivot settings
        pivot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot')
        pivot_column = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_columns')
        pivot_value = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_values')

        # pivot_table settings
        pivot_table = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table')
        pivot_table_column = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_columns')
        pivot_table_value = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_values')
        pivot_table_agg_function = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_agg_func')
        pivot_table_indizes = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_indizes')
        pivot_table_indizes_visible = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_indizes_visible')
        pivot_table_rename_indizes = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('pivot_table').get(
            'pivot_index').get('pivot_rename_indizes')

        # margins (subtotals)
        margin = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('margins').get('margin')
        margin_name = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('margins').get('margin_name')

        # chart settings
        chart = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-kind')
        title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('title')
        x_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('x-axis-title')
        y_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('y-axis-title')
        x_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('x-axis-columns')
        y_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('y-axis-columns')
        index = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-index')

        # chart styles
        grid = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('grid')
        legend = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('legend')
        rot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('rot')
        fontsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('fontsize')
        figsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('figsize')
        stacked = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('stacked')
        secondary_y = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('secondary_y')
        stylelist = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('stylelist')
        subplots = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('subplots')
        autopct = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('autopct')
        loc = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('loc')
        bbox_to_anchor = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('chart-designs').get('bbox_to_anchor')

        # get the pandas (panda data)
        panda_table_data = get_data(startpath, destination, imagename, datafile_path, datafile, separator, decimal)

        # filter by where clausel
```

```
        if where_clausel:
            panda_table_data = panda_table_data.query(where_clausel)

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

        # set aggregation functions
        # if groupby_values and not agg_funtion and not pivot_column and not pivot_table_column:
        if groupby_values and not (pivot_column or (pivot_table_column or pivot_table_value or pivot_table_indizes)):
            match group_by_function:
                case 'max':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).max()
                case 'min':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).min()
                case 'head':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).head()
                case 'sum':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).sum()
                case 'mean':
                    panda_table_data = panda_table_data.groupby(groupby_values, as_index=False).mean()
        else:
            panda_table_data = panda_table_data

        # pivot if pivot is selected
        if pivot_table_column or pivot_table_value or pivot_table_indizes:
            if type(pivot_table_agg_function) is list:
                agg_tuple = tuple(pivot_table_agg_function)
                panda_table_data = pd.pivot_table(panda_table_data, index=pivot_table_indizes,
                                                  columns=pivot_table_column, values=pivot_table_value,
                                                  aggfunc=agg_tuple, margins=margin, margins_name=margin_name)
            elif type(pivot_table_agg_function) is dict:
                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_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('operations').get(column_ops).get('operation_function')
                operation_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').get(table_item).get('column_operations').get('operations').get(column_ops).get('columns')
                operation_axis = panda_diagram_plotter_config.get('panda_diagram_plotter').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)

        # 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 180: Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm

XV.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
      grid: True
      legend: True
      rot:
      fontsize:
      stacked: False
      secondary_y: False
      stylelist:
      figsize:
      autopct:
      loc:
      bbox_to_anchor:
    group_by:
    group_by_function:
    agg_funtion:
    agg_colums:
    drop_columns:
      - "tps_1_iteration"
      - "tps_typ"
    column_operations:
      datas:
```

```yaml
    pivot:
      pivot_columns:
      pivot_values:
    pivot_table:
      pivot_index:
       pivot_indizes_visible: False
       pivot_rename_indizes:
      pivot_columns:
      pivot_values:
      pivot_agg_func:
    rename_columns:
      variante: "Varianten"
      tps_2_iteration: "2. Iteration"
      tps_3_iteration: "3. Iteration"
      tps_4_iteration: "4. Iteration"
    where_clausel: "tps_typ == 'mixed'"
    sorting:
      order_by:
      sort_acending: True
      sort_inplace: True
    margins:
      margin: False
      margin_name:
  db_inventory_per_rdbms:
    id: "db_inventory_per_rdbms"
    startpath: "parentdir"
    destination_path: "source/pandas_data_chart_plotter"
    imagename: "db_inventory_per_rdbms.png"
    datafile_path: "source/tables"
    datafile: "inventory.csv"
    separator: ","
    decimal: "."
    x-axis-columns: "RDBMS"
    y-axis-columns:
    x-axis-title:
    y-axis-title:
    title: "Datenbankinventor - Pro RDBMS"
    chart-index: "RDBMS"
    chart-kind: "pie"
    chart-designs:
      subplots: True
      grid: False
      legend: True
      rot:
      fontsize:
      stacked: False
      secondary_y: False
      stylelist:
      figsize: !!python/tuple [25,10]
      autopct: '%1.0f%%'
      loc: "best"
      bbox_to_anchor:
    group_by:
      - "rdbms"
```

```yaml
    group_by_function: "sum"
    agg_funtion:
    agg_colums:
      - "rdbms"
    drop_columns:
     - "server"
     - "os"
     - "version"
     - "releasedate"
     - "eol"
     - "age"
     - "eol_since"
     - "comment"
     - "appliance"
    column_operations:
      datas:
    pivot:
      pivot_columns:
      pivot_values:
    pivot_table:
      pivot_index:
       pivot_indizes_visible: False
       pivot_rename_indizes:
      pivot_columns:
      pivot_values:
      pivot_agg_func:
    rename_columns:
     rdbms: "RDBMS"
     instance : "Instanz"
     databases : "Datenbanken"
     appliance: "Appliance"
    where_clausel:
    sorting:
      order_by:
        - "rdbms"
      sort_acending: True
      sort_inplace: True
    margins:
      margin: False
      margin_name:
  db_inventory_per_os:
    id: "db_inventory_per_os"
    separator: ","
    decimal: "."
    startpath: "parentdir"
    destination_path: "source/pandas_data_chart_plotter"
    datafile_path: "source/tables"
    datafile: "inventory.csv"
    imagename: "db_inventory_per_os.png"
    decimal_format:
    x-axis-columns: "RDBMS"
    y-axis-columns:
    x-axis-title:
    y-axis-title:
```

```yaml
    title: "Datenbankinventor - Pro OS"
    chart-index:
    chart-kind: "pie"
    chart-designs:
      subplots: True
      grid: False
      legend: False
      rot:
      fontsize:
      stacked: False
      secondary_y: False
      stylelist:
      figsize: !!python/tuple [25,10]
      autopct: '%1.0f%%'
      loc: "upper center"
      bbox_to_anchor: !!python/tuple [0,0]
    group_by:
     - "rdbms"
     - "os"
    group_by_function: "sum"
    agg_funtion:
    agg_colums:
     - "os"
    drop_columns:
     - "server"
     - "version"
     - "releasedate"
     - "eol"
     - "age"
     - "eol_since"
     - "comment"
#    - "appliance"
   column_operations:
    datas:
    operations:
     dauer_summe:
      operation_function:
      axis_number:
      columns:
   pivot:
    pivot_columns:
    pivot_values:
   pivot_table:
    pivot_index:
     pivot_indizes:
      - "os"
      - "rdbms"
     pivot_indizes_visible: True
     pivot_rename_indizes:
      os: "OS"
      rdbms: "RDBMS"
    pivot_columns:
    pivot_values:
    pivot_agg_func:
```

```
      instance: "sum"
      databases: "sum"
      appliance: "sum"
     transpose: True
    rename_columns:
     rdbms: "RDBMS"
     instance : "Instanz"
     databases : "Datenbanken"
     os : "OS"
     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 181: Python LaTex - pandas_data_chart_plotter_conf.ya