ibW Höhere Fachschule Südostschweiz

Diplomarbeit Technik und Wirtschaftsinformatik 2023-2024

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

Zusammenfassung

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

Management Summary

Diplomarbeit Michael Graber

Inhaltsverzeichnis

Abkürzungen **5**

1 Einleitung **1**

1.1 Ausgangslage und Problemstellung . . . 1

1.1.1 Das Kantonsspital Graubünden . . . 1

1.1.2 Die ICT des Kantonsspital Graubünden . . . 3

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

1.1.4 Ausgangslange . . . 6

1.1.5 Problemstellung . . . 10

1.2 Zieldefinition . . . 14

1.3 Abgrenzungen . . . 17

1.4 Abhängigkeiten . . . 19

1.5 Risikomanagement . . . 20

1.5.1 Riskcontrolling . . . 23

1.6 Vorgehensweise und Methoden . . . 26

1.7 Projektmanagement . . . 26

1.7.1 Projektcontrolling . . . 27

1.7.2 GANTT-Diagramm . . . 29

1.8 Status-Reports . . . 31

1.8.1 Initialer Statusbericht . . . 31

1.8.2 Zweiter Statusbericht . . . 32

1.9 Expertengespräche . . . 33

2 Umsetzung **34**

2.1 Evaluation . . . 34

2.1.1 Exkurs Architektur . . . 34

2.1.2 Erheben und Gewichten der Anforderungen . . . 41

2.1.3 Testziele erarbeiten . . . 54

2.1.4 PostgreSQL Benchmarking . . . 54

2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen . . . 60

2.1.6 Vorauswahl . . . 99

2.1.7 Installation verschiedener Lösungen . . . 99

2.1.8 Gegenüberstellung der Lösungen . . . 107

2.1.9 Entscheid . . . 116

2.2 Aufbau und Implementation Testsystem . . . 116
2.2.1 Bereitstellen der Grundinfrastruktur . . . 116
2.2.2 Installation und Konfiguration PostgreSQL HA Cluster . . . 116
2.2.3 Technical Review der Umgebung . . . 116
2.3 Testing . . . 116
2.3.1 Testing . . . 116
2.3.2 Protokollierung . . . 116
2.3.3 Review und Auswertung . . . 117
2.4 Troubleshooting und Lösungsfindung . . . 117

3 Resultate **118**
3.1 Zielüberprüfung . . . 118
3.2 Schlussfolgerung . . . 118
3.3 Weiteres Vorgehen / offene Arbeiten . . . 118
3.4 Persönliches Fazit . . . 118

Abbildungsverzeichnis **119**

Tabellenverzeichnis **122**

Listings **123**

Literatur **125**

Glossar **131**

Anhang **i**
I Arbeitsrapport . . . i
II Protokoll - Fachgespräche . . . ii
III Kommentare / Anmerkungen . . . iii
IV rke2 . . . v
IV.I Vorbereitung . . . v
IV.II Installation . . . v
IV.III Cluster Konfiguration . . . vi
V yugabyteDB . . . x
V.I Prerequisites . . . x
V.II Installation . . . xi
V.III Rekonfiguration mit 300GiB Storage . . . xxiv
VI Stackgres mit Citus . . . xxiv
VI.I Prerequisites . . . xxiv

VII sks9016 - yugabyteDB . . . xxv
VII.I yugabyteDB - Download und Installation yugabyteDB . . . xxv
VIII Patroni . . . xxv
VIII.I Prerequisites . . . xxvi
VIII.II Installation . . . xxvii
VIII.III Konfiguration . . . xxvii
IX zotero.py . . . xxvii
X zotero_bibtex_configuration.yaml . . . xxxiii
XI zotero_biblatex_keystore.yaml . . . xxxiii
XII riskmatrix.py . . . xl
XIII riskmatrix_plotter_conf.yaml . . . xlv
XIV riskmatrix_xy_axis_tuple_matrix.yaml . . . xlviii
XV cost_benefit_diagram.py . . . xlix
XVI cost_benefit_diagram_plotter_conf.yaml . . . li
XVII pandas_dataframe_to_latex_table.py . . . lii
XVIII csv_to_latex_diplomarbeit.yaml . . . lxi
XIX pandas_data_chart_plotter.py . . . lxxxi
XX pandas_data_chart_plotter_conf.yaml . . . lxxxviii

Abkürzungen

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

PKI	Private Key Infrastructure
DCS	Distributed Configuration Store
DQL	Data Query Language
DML	Data Manipulation Language
ACID	Atomicity, Consistency, Isolation und Durability
EDB	EnterpriseDB
CRD	Custom Resource Definition
rke2	Ranger Kubernetes Engine 2
BGP	Border Gateway Protocol

1 Einleitung

1.1 Ausgangslage und Problemstellung

1.1.1 Das Kantonsspital Graubünden

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

Abbildung 1.1: Spitalregionen Kanton Graubünden[53]

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

Abbildung 1.2: Wahlkreise Kanton St. Gallen[78]

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[47]

1.1.2 Die ICT des Kantonsspital Graubünden

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

Abbildung 1.4: Organigramm Kantonsspital Graubünden

Die ICT betreibt über 400 Applikationen die auf mehr als 1055 physische und virtuelle Server und Appliances. Das Rückgrat der Infrastruktur ist dabei die Virtualisierungsplattformen VMware ESXi für Server und Citrix für die Thinclients der Enduser. Es werden aber auch Dienstleistungen für andere Spitäler und Kliniken oder andere Einrichtungen des Gesundheitswesens erbracht. Entsprechen wurde die ICT in ein Applikationsmanagement, ein Infrastrukturmanagement sowie einem unterstützenden Bereich aufgegliedert. Das Applikationsmanagement wurde in je einen Bereich für die Administrativen und Medizinischen Applikationen aufgeteilt. Das Infrastrukturmanagement wiederum wurde in den Bereich Netzwerk und Data Center, welcher für Server zuständig ist, aufgeteilt. Der Bereich Business- und Prozessunterstützung beinhaltet je eine Abteilung für die Businessanalyse, das Projektmanagement und Benutzer- und Clientservices in der auch der Service-Desk untergebracht ist.

Abbildung 1.5: Organigramm Departement 10 - ICT

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

1.1.3 Rolle in der ICT vom Kantonsspital Graubünden

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

1.1.4 Ausgangslange

Die meisten der über 400 Applikationen, die das KSGR betreibt, haben in den allermeisten Fällen ihre Daten in Datenbanksysteme speichern. Entsprechend der Vielfalt der Applikationen existieren auch eine vielzahl an Datenbanksystemen und Versionen.
Basierend auf der Liste *DB-Engines Ranking*[44] der Top-Datenbanksysteme . Allerdings werden nicht alle Datenbanksysteme berücksichtigt, entweder weil das Datenbanksystem keine Client/Server Architektur hat oder nicht im Scope der IT oder des Projekts ist.
Folgende Datenbanken sind inventarisiert:

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

Tabelle 1.1: Inventarisierte Datenbanksysteme

Folgende Datenbanksysteme sind demnach im KSGR im Einsatz:

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

Tabelle 1.2: Datenbankinventar

Abbildung 1.6: Datenbanken - Aufgeschlüsselt nach RDBMS

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

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

Abbildung 1.7: Datenbanken - Aufgeschlüsselt nach Betriebssystem

Die Kernsysteme des Spitals werden auf Oracle Datenbanken (Oracle Database)betrieben, die aktuell auf einer HP-UX betrieben werden. Stand heute gibt es kein Clustersystem für die Open-Source Datenbanken wie MariaDB/MySQL oder PostgreSQL.
Durch die Einführung von Kubernetes als Containerplattform wird der Bedarf an PostgreSQL Datenbanken immer grösser. Es werden in naher Zukunft auch verschiedene Oracle Datenbanken sowie MySQL Datenbanken auf PostgreSQL migriert werden.
Aktuell werden die Daten des Zabbix der Netzwerktechniker auf eine MariaDB Datenbank gespeichert, dies soll sich aber ändern. Da das Zabbix alle Netzwerkgeräte Überwacht, pro Sekunde werden im Moment 1’200 Datenpunkte abgefragt und xxx in die Datenbank und wird im Laufe der Zeit mehrere Terrabyte gross werden.

1.1.5 Problemstellung

Zusammen mit den bestehenden PostgreSQL-Datenbankinstanzen werden die PostgreSQL Datenbanken in der Art, wie sie bisher Betrieben werden, nicht mehr Betreibbar sein. Die bisherige Strategie erzeugt sehr viele Aufwände und provoziert Risiken, namentlich:

- dezentrale Backups und fragmentierte Backup-Strategien
  - Fehlende Kontrolle
  - Wiederherstellbarkeit nicht garantiert
- Verschiedene Betriebssysteme mit verschiedenen Versionen
  - Fehlernder Überblick
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand
- Uneinheitliche Absicherung und Härtung
  - Hohe Angreifbarkeit
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand
- Uneinheitliche HA-Fähigkeit
  - Hohe Angreifbarkeit
  - Veraltete Betriebssystem- und Datenbankversionen
  - Grosser Administrationsaufwand

Dadurch ergeben sich nach BSI folgende Risiken:

Identifikation						Abschätzung		Behandlung			
								Massnahmen ergreifen?	Zielwert		Massnahme
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM		WS	SM	
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 Fehlerhafter Administration und Konfiguration.	Doch die folgen bleiben nichtsdesto trotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen. Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist.	4	3	Ja	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, das in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen. Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen Wissendlich oder Unwissendlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen.	Der Wissentliche oder Unwissentliche Missbrauch von Berechtigungen kann verheerende Auswirkungen haben. Unter anderem können Daten missbräuchlich abgezogen werden, Daten manipuliert oder das ganze System komplett zerstört werden.	2	4	Ja	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

Abbildung 1.8: Risiken bestehende Lösung

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

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

Mit diesen Massnahmen lassen sich die Risiken gesenkt werden:

Abbildung 1.9: Risiken bestehende Lösung mit Massnahmen

1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [63, 69] folgende zehn Tasks die zum täglichen Alltag gehören:

Nr.	Aufgabe	Beschreibung	Wichtigkeit
1	Failover	In einem Fehlerfall soll die DB-Node auf einen Standby-Node übergeben werden.	Hoch
2	Failover Restore	Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den Primären Node zurückgesetzt werden. Dabei darf es zu keinem Datenverlust kommen, also alle Daten die auf dem Standby-Node erfasst wurden, müssen auf den Primären DB-Node zurückgeschrieben werden beim Failover Restore	Hoch
3	Filesystem Management	Die Datenmenge von Datenbanken wachsen in der Regel beständig. Die Belegung von Tablespaces und Filesystem muss deshalb Überwacht und ggf. erweitert werden. Läuft eine Disk voll kommt es im besten Fall zu einem Stillstand der DB, im schlimmsten Fall zu Inkonsistenzen und Datenverlust	Hoch
4	Monitoring	Nebst den allgemeinen Metriken wie CPU / Memory Usage und der Port Verfügbarkeit gibt es noch eine reihe weiterer Aspekte die Überwacht werden müssen. Zum Beispiel ob es zu verzögerungen bei der Replikation kommt oder die Tablespaces genügend Platz haben. Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall.	Mittel
5	Statistiken / Cleanup Jobs justieren	PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. Zudem wird im Rahmen des gleichen Sheduled Tasks ein Cleanup Vorgenommen, so dass z.B. gelöschte Datensätze den Disk Space nicht sinnlos belegen. Die Konfiguration dieser Jobs muss an der Metrik der Datenbank angepasst werden, weil gewisse Tasks dann entweder viel zu oft oder viel zu wenig bis gar nicht mehr ausgeführt werden.	Mittel
6	SQL optimierungen	In PostgreSQL können inperfomante SQL Statements ausgelesen werden und zum Teil werden auch informationen zum Tuning geliefert[42]. Diese müssen Regelmässig ausgelesen werden	Tief
7	Health Checks und Aktionen (Maintenance)	Regelmässig muss die Gesundheit der DBs überprüft werden, etwa ob Tabellen und/oder Indizes sich aufgebläht haben oder ob Locks vorhanden sind[2]. Während der Hauptarbeitszeit muss dies mindestens alle 90 Minuten geprüft und ggf. reagiert werden.	Hoch
8	Housekeeping	Mit Housekeeping Jobs werden regelmässig Trace- und Alertlogfiles aufgeräumt, um Platz auf den Disken zu sparen aber auch um die Übersichtlichkeit zu wahren.	Mittel
9	Verwalten von DB Objekten	Regelmässig müssen DB Objekte wie Datenbanken, Tabellen, Trigger, Views etc. angepasst oder erstellt werden. Dies richtet sich nach den Bedürfnis der Kunden resp. deren Applikationen.	Tief
10	User Management	Die Zugriffe der User müssen Überwacht, angepasst, erfasst oder gesperrt werden. Auch diese Aufgabe richtet sich nach den Bedürfnissen der Kunden.	Tief

Tabelle 1.5: Administrative Aufgaben

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

Nr.	Aufgabe	Wichtigkeit	Zu automatisierender Task	Priorität	Muss / Kann	Spätester Termin
1	Failover	Hoch	Automatisierter Failover auf mindestens einen Sekundären DB-Node	1	Muss	Abgabe
2	Failover Restore	Hoch	Sobald der Primäre DB-Node wieder vorhanden ist, muss automatisch auf den Primären DB-Node zurückgesetzt werden.	1	Muss	
3	Filesystem Management	Hoch	Das Filesystem muss beim erreichen von 95% Usage automatisiert vergrössert werden. Die Vergrösserung muss anhand der Wachstumsrate (die mittels Linux Commands zu ermitteln ist), vergrössert werden	4	Kann	
4	Monitoring	Mittel	Der Status der Clusterumgebung und der Replikation muss im PRTG überwacht werden	2	Muss	
5	Statistiken / Cleanup Jobs justieren	Mittel	Regelmässig müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden	2	Muss	
6	SQL optimierungen	Tief	Es gibt SQL Abfragen, mit dem fehlende Indizes ermittelt werden können. Diese Indizes sollen automatisiert erstellt werden. Im gleichen Zug sollen aber auch Indizes, welche nicht verwendet werden, entfernt werden. Sie tragen nicht nur nichts zu performanteren Abfragen bei sondern beziehen unnötige Ressourcen bei Datenmanipulationen[42].	2	Kann	
7	Health Checks und Aktionen (Maintenance)	Hoch	Tabellen und Indizes können sich aufblähen (bloaded table / bloaded index) Ist ein Index aufgebläht, kann dies mittels eines REINDEX mit geringem Impact auf die Datenbank gelöst werden[2].	2	Muss	
8	Housekeeping	Mittel	Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden.	3	Kann	
9	Verwalten von DB Objekten	Tief	Keine automatisierung möglich	5		
10	User Management	Tief	Regelmässige Reports sollen User aufzeigen, die seit mehr als einer Woche nicht mehr aktiv waren.	4	Kann	

Tabelle 1.6: Automatisierung Administrativer Aufgaben

Mit der Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

- Ergebnisse
  Mindestens drei Methoden einen PostgreSQL Cluster aufzubauen müssen analysiert und evaluirt werden

- Resultate
  Aus den mindestens drei Methoden muss die optimale Methode ermittelt werden.
  Am Ende muss zudem ein Funktionierendes Testsystem bestehen.

Daraus ergeben sich folgende Ziele:

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

Tabelle 1.7: Ziele

1.3 Abgrenzungen

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

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

Tabelle 1.8: Gegebene Systeme

Daraus ergeben sich nach nach Züst, Troxler 2002[93] folgende Abgrenzungen:

Abbildung 1.10: Systemabgrenzung

1.4 Abhängigkeiten

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

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

Tabelle 1.9: Abhängigkeiten

1.5 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 ergibt sich folgende Risikomatrix

Abbildung 1.11: Projektrisiken

Mit den entsprechenden Massnahmen können die Risiken gesenkt werden:

Abbildung 1.12: Projektrisiken mit Massnahmen

1.5.1 Riskcontrolling

1.5.1.1 Neu erfasste Risiken

	ID	Definiert / Erkannt	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen notwendig	WS.1	SM.1	Massnahmen
	9	19.03.2024	Veeam Kasten K10[3] nicht betriebsbereit	Abhängigkeit zum KSGR k8s Projekt.							
Ohne Kubernetes kein Veeam Kasten.		Keine Kubernetes-Gerechten Sicherungen von Pods usw. Sicherungen Inkonsistent o.ä. Ohne Backup kann das Ziel des Projekts nicht erreicht werden	5	5	Ja	5	2	Backup auf einem nfs-Share ablegen. So können Test-DBs wenigstens gesichert werden.			

Tabelle 1.11: Neu Erkannte / Erfasste Risiken

1.5.1.2 Assessment 21.03.2024

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

Tabelle 1.12: Risiko-Assessment 21.03.2024

Abbildung 1.13: Riskikomatrix - Assessment 21.03.2024

1.6 Vorgehensweise und Methoden

1.7 Projektmanagement

1.7.1 Projektcontrolling

	Phase	Subphase	Dauer [h]	Geplante Dauer [h]	Verbleibende Zeit [h]
0	1. Expertengespräch	1. Expertengespräch	1.0	1.0	0.0
1	2. Expertengespräch	2. Expertengespräch	0.2	1.0	0.8
2	Aufbau und Implementation Testsystem	Basisinfrastruktur	0.0	4.0	4.0
3	Aufbau und Implementation Testsystem	Installation und Konfiguration PostgreSQL HA Cluster	0.0	20.0	20.0
4	Aufbau und Implementation Testsystem	Technical Review	0.0	3.0	3.0
5	Dokumentation	Dokumentation	31.0	80.0	49.0
6	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	17.8	16.0	-1.8
7	Evaluation	Anorderungskatalog	4.5	16.0	11.5
8	Evaluation	Gegenüberstellung	0.0	8.0	8.0
9	Evaluation	Variantenentscheid	0.0	4.0	4.0
10	Evaluation	Vorbereitung Benchmarking	5.0	4.0	-1.0
11	Letztes Expertengespräch	Letztes Expertengespräch	0.0	1.0	1.0
12	Puffer	Puffer	0.0	16.0	16.0
13	Resultate	Persönliches Fazit	0.0	2.0	2.0
14	Resultate	Schlussfolgerung	0.0	2.0	2.0
15	Resultate	Weiteres Vorgehen / offene Arbeiten	0.0	1.0	1.0
16	Resultate	Zielüberprüfung	0.0	2.0	2.0
17	Testing	Protokollierung	0.0	4.0	4.0
18	Testing	Review und Auswertung	0.0	2.0	2.0
19	Testing	Testing Testsystem	0.0	8.0	8.0
20	Troubleshooting und Lösungsfindung	Troubleshooting und Lösungsfindung	0.0	8.0	8.0
Total			59.5	203.0	143.5

Tabelle 1.13: Projektcontrolling

Abbildung 1.14: Projektcontrolling

1.7.2 GANTT-Diagramm

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

Projekt: Diplomarbeit - Postgre
Datum: Sam 16.03.24

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

1.8 Status-Reports

1.8.1 Initialer Statusbericht

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusberich	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	- Anforderungskatalog erarbeiten
- Versnissage vorbereitet	- Vorbereitung Benchmarking
- Statusbericht erstellt	

# 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:	Erneuerung HP UX Plattform 60002201	Ressourcen reservieren
$\frac{100\,\text{CHF}}{h} * 200h = 24\,000$ CHF	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)

Tabelle 1.14: Initialer Statusbericht

1.8.2 Zweiter Statusbericht

PostgreSQL HA Cluster - Konzeption und Implementation		ICT Projektstatusbericht	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 Plmtform`, 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: $100\,\mathrm{CHF\,h^{-1}} * 200 = 24\,000\,\mathrm{CHF}$	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:	PMO:	
Datum:	Datum:	

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

Tabelle 1.15: Zweiter Statusbericht

1.9 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 1.16: Fachgespräche

Das Protokoll ist im Anhang zu finden.

2 Umsetzung

2.1 Evaluation

2.1.1 Exkurs Architektur

2.1.1.1 Sharding

2.1.1.1.1 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 2.1: 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 2.2: Sharding - Horizontales Partitionierung

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

2.1.1.1.2 Key Based Sharding

Hierbei wird das sharding anhand eines oder mehreren Keys ausgeführt.

2.1.1.1.3 Range Based Sharding

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

2.1.1.1.4 Directory Based Sharding

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

2.1.1.1.5 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.

2.1.1.2 Monolithische vs. verteilte SQL Systeme

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

Nur verteilte Systeme, sogenannte Distributed SQL wiederum sind Cloud Native

Abbildung 2.3: Monolithische vs. verteilte SQL Systeme

2.1.1.3 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[67]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

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

Asynchrone Replikation

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

2.1.1.4 Quorum

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

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

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

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

Hier ein Beispiel wie sie in den Artikeln [65, 76, 60] beschrieben werden. Es zeigt auf, ab wie vielen Nodes die Fehlertoleranz erhöht wird und wie sich der Representative Throughput verhält.

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

Tabelle 2.1: Quorum Beispiele

2.1.1.5 CAP Theorem

Das CAP Theorem besagt, das nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[49].

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

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

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

Abbildung 2.4: CAP-Theorem

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

2.1.1.6 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[62]:

Abbildung 2.5: 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.

2.1.2 Erheben und Gewichten der Anforderungen

2.1.2.1 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 ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
6	Switchover	Skript / API	Das System muss ein Skript oder eine API liefern, welche einen 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 ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
9	Restore	Skript / API	Das Clustersystem muss ein Skript oder eine API liefern, welche das einfache und ggf. automatisierte 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 2.2: Anforderungskatalog

2.1.2.2 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 2.3: Stakeholder

2.1.2.3 Gewichtung

Die Gewichtung wurde mittels einer Präferenzmatrix ermittelt.
Dabei wurden folgende Anforderungen aus übersichtsgründe in Sub-Matrizen aufgeteilt:

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

Die Grundlegende Gewichtung wurde folgendermassen vorgenommen:

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

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.6: Präferenzmatrix

Die Gewichtung der Failover-Anforderungen setzt sich wie folgt zusammen:

Gewicht	Nennungen	Rang	Nr.	Ziele	Automatismus 1	Connection-Stabilität 2
5	3	1	1	Automatismus		
4	2	2	2	Connection-Stabilität	1	
2	1	3	3	Geschwindigkeit	1	2
11	6					

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.7: Präferenzmatrix - Failover

Beim Switchover wurde die Gewichtung wie folgt aufgeteilt:

Gewicht	Nennungen	Rang	Nr.	Ziele	Skript / API (1)	Connection - Stabilität (2)
5	3	1	1	Skript / API		
3	2	2	2	Connection - Stabilität	1	
2	1	3	3	Geschwindigkeit	1	2
10	6					

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.8: Präferenzmatrix - Switchover

Die Gewichtung und Aufteilung der Restore-Anforderungen sieht wie folgt aus:

Gewicht	Nennungen	Rang	Nr.	Ziele	1 Skript / API	2 Datensicherheit	3 Connection - Stabilität
5	3	1	1	Skript / API			
2	1	2	2	Datensicherheit	1		
2	1	2	3	Connection - Stabilität	1	2	
2	1	2	4	Geschwindigkeit	1	4	3
9	6						

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.9: Präferenzmatrix - Restore

Die Replikationsanforderungen resp. deren Gewichtung ist wie folgt aufgebaut:

Gewicht	Nennungen	Rang	Nr.	Ziele	Synchrone Replikation 1	Failover / Switchover Garant 2
4	3	1	1	Synchrone Replikation		
3	2	2	2	Failover / Switchover Garantie	1	
1	1	3	3	Throughput	1	2
8	6					

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.10: Präferenzmatrix - Replikation

Das Sharding setzt sich aus folgenden Teilen zusammen:

Gewicht	Nennungen	Rang	Nr.	Ziele	Datenkonsistenz- und Integr 1
2	2	1	1	Datenkonsistenz- und Integrität	
1	1	2	2	Schutz vor Datenverlust	1
3	3				

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.11: Präferenzmatrix - Sharding

Die Quorum-Anforderung ist folgendermassen zusammengesetzt:

Gewicht	Nennungen	Rang	Nr.	Ziele	Quorum-System vorhanden 1
4	2	1	1	Quorum-System vorhanden	
2	1	2	2	Robustheit	1
7	3				

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.12: Präferenzmatrix - Quorum

Bei der Management-API gibt es mehrere Sub-Anforderungen:

Gewicht	Nennungen	Rang	Nr.	Ziele	Management-API vorhanden 1	Authentifizierung & Autorisie 2
1	2	1	1	Management-API vorhanden		
1	2	1	2	Authentifizierung & Autorisierung	1	
1	2	1	3	Aufwand	3	2
3	6					

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.13: Präferenzmatrix - Management-API

Anforderungen zum Backup wurden nachfolgend aufgeteilt und gewichtet:

Gewicht	Nennungen	Rang	Nr.	Ziele	Backup mit PostgreSQL Stan 1
4	2	1	1	Backup mit PostgreSQL Standards	
2	1	2	2	Restore mit PostgreSQL Standards	1
6	3				

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.14: Präferenzmatrix - Backup

Performance-Benchmarking lässt sich in nachfolgende Teile unterteilen:

Gewicht	Nennungen	Rang	Nr.	Ziele	1 tps - Read-Only	2 tps - Read-Writes	3 Ø Latenz - Read-Only
2	4	1	1	tps - Read-Only			
2	3	2	2	tps - Read-Writes	1		
1	2	3	3	Ø Latenz - Read-Only	1	2	
1	1	4	4	Ø Latenz - Read-Write	1	2	3
6	10						

Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.15: Präferenzmatrix - Performance

2.1.3 Testziele erarbeiten

2.1.4 PostgreSQL Benchmarking

2.1.4.1 pgBench - Basis-Benchmarking

PostgreSQL bietet ein Benchmarking-Tool,[56, 1] mit dem die DB Vermessen werden kann.

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

Es gibt einiges zu beachten, wenn PostgreSQL einem Benchmarking unterzogen wird. Aus diversen Quellen [37, 34, 91, 1] sind dies folgende Anforderungen:

pgGather
Mit pgGather [64] müssen vorgängig alle Probleme analysiert und beseitigt werden.

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

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

Non-Default Konfiguration
Die PostgreSQL DB sollte nicht mit der Default Konfiguration betrieben werden.
Die Konfiguration sollte anhand der zu erwartenden Workloads und Sessions konfiguriert werden.

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

Benchmark Dauer
Die Zeit zwischen den Transaktionen und die Dauer an sich sollten über einen längeren Zeitraum stattfinden.
Nur so, kann ein echtes verhalten gemessen werden.

Störfaktoren
Störfaktoren, etwa Netzwerklatenzen [87] usw., müssen ebenfalls bemessen werden.
Nur so kann sichergestellt werden, dass die Umgebung sauber ist.

2.1.4.2 Replication Throughput Benchmarking

Etwas Komplexer wird es beim Benchmark des Throughput. Den nebst den DB-Latenzen usw. kommt nun noch die Netzwerklatenz usw. hinzu [75].

2.1.4.3 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 2.16: 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 2.17: 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 2.18: Benchmark Settings - Zabbix - Queries per Seconds

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

Abbildung 2.19: 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 2.20: 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 2.4: 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 2.5: 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[57]:

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 2.6: 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 2.7: 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 2.8: 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 2.21: Benchmark Settings - Anzahl Records / Skalierungsfaktor

2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen

2.1.5.1 Percona

Percona bietet nebst dem bekannteren Galera-Cluster und MySQL- und MariaDB auch Dienstleistungen [33] aller Art für PostgreSQL an.

Percona arbeitet oft auf Basis von Patroni,
bietet aber auch eigene Erweiterungen und eigene Software an[30].

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.

2.1.5.2 EnterpriseDB

EnterpriseDB, oder EDB, ist ein führendes Software- und Servicehaus für PostgreSQL[18].
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.

2.1.5.3 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[77]. 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.

2.1.5.4 pgpool-II

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

2.1.5.4.1 Core-Features

pgpool-II bietet folgende Core-Feature[54]:

- 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

2.1.5.4.2 Replikation

pgpool-II bietet ein eigenes Replikationssystem an.

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

2.1.5.4.3 Proxy

pgpool-II hat bereits einen intergrierten Load Balancer.
Einen zusätzlichen Proxy wird daher nicht benötigt.

2.1.5.4.4 Pooling

pgpool-II bietet ebenfalls von Haus aus einen Pooler.

2.1.5.4.5 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.

2.1.5.4.6 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.

2.1.5.5 pg_auto_failover

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

2.1.5.5.1 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

2.1.5.5.2 Replikation

pg_auto_failover bietet die Standard PostgreSQL Replikationen.

2.1.5.5.3 Proxy

pg_auto_failover benötigt einen HAProxy, um Load Balancing usw. [22]

2.1.5.5.4 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[26].

2.1.5.5.5 Architektur

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

Abbildung 2.22: pg_auto_failover-Architektur - Single Standby

Aber auch Multi-Nodes können eingebunden werden[29]:

Abbildung 2.23: pg_auto_failover-Architektur - Multi-Node Standby

pg_auto_failover kann Citus einbinden[11]. Allerdings bleibt die Architektur im Kern immer Monolothisch.

Abbildung 2.24: pg_auto_failover-Architektur - Citus

2.1.5.5.6 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.

2.1.5.6 CloudNativePG

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

2.1.5.6.1 Core-Features

Die wichtigsten Features von CloudNativePG sind[13]:

- 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

2.1.5.6.2 Replikation

CloudNativePG bietet die üblichen PostgreSQL Replikaionen an.

2.1.5.6.3 Proxy

CloudNativePG benötigt keinen zusätzlichen Proxy.

2.1.5.6.4 Pooling

CloudNativePG unterstützt pgBouncer als Pooler.

2.1.5.6.5 API / Skripte

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

2.1.5.6.6 Architektur

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

Abbildung 2.25: CloudNativePG - Kubernetes - PostgreSQL

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

Abbildung 2.26: CloudNativePG - Kubernetes - Read-write workloads

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

Abbildung 2.27: CloudNativePG - Kubernetes - Read-only workloads

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

2.1.5.6.7 Maintenance

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

Abbildung 2.28: 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 2.29: CloudNativePG - Code Frequency

Das Projekt hält die meisten Standards von GitHub ein:

Abbildung 2.30: 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 2.31: CloudNativePG - Contributors Commits

Abbildung 2.32: CloudNativePG - Contributors Deletations

Abbildung 2.33: CloudNativePG - Contributors Additions

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

Abbildung 2.34: CloudNativePG - Commit Activity

Nebst dem Projekt cloudnative-pg der «© The CloudNativePG Contributors»ist CloudNativePG-Gründer EDB noch immer ein grosser Contributor.

Abbildung 2.35: CloudNativePG - Network Graph

2.1.5.6.8 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.

2.1.5.7 Patroni

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

2.1.5.7.1 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

2.1.5.7.2 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.

2.1.5.7.3 Proxy

Patroni benötigt einen HAProxy, um Load Balancing betreiben zu können[22].

2.1.5.7.4 Pooling

Patroni benötigt einen externen Pooler.
Hier wird oft PgBouncer [31] verwendet.

2.1.5.7.5 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.

2.1.5.7.6 etcd

Patroni benötigt etcd als key-value-store

2.1.5.7.7 Architektur

Das Architektur-Schaubild sieht folgendermassen aus:

Abbildung 2.36: Patroni-Architektur

2.1.5.7.8 Maintenance

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

Abbildung 2.37: Patroni - Pulse

Code wird Regelmässig hinzugefügt und entfernt:

Abbildung 2.38: Patroni - Code Frequency

Das Projekt hält auch die gängigen Standards auf Github ein:

Abbildung 2.39: Patroni - Community Standards

Die Contributors commiten, löschen und erweitern Patroni Regelmässig:

Abbildung 2.40: Patroni - Contributors Commits

Abbildung 2.41: Patroni - Contributors Deletations

Abbildung 2.42: Patroni - Contributors Additions

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

Abbildung 2.43: Patroni - Commit Activity

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

Abbildung 2.44: Patroni - Network Graph

2.1.5.7.9 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`[92] bieten zudem die vorlage für eine noch stärkere automatisierung.

2.1.5.8 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

2.1.5.8.1 Core-Features

Die wichtigsten Features von Stackgres sind[21]:

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

2.1.5.8.2 Replikation

Stackgres bietet Asynchrone und Synchrone Replikation, Gruppenreplikation sowie kaskadierende Replikation an.

Citus bietet sein eigenes Modell mit dem Sharding an.

2.1.5.8.3 Proxy

Stackgres hat den Proxy bereits mit envoy[19] implementiert.

2.1.5.8.4 Pooling

PgBounder[31] ist bereits integriert, es braucht also keinen weiteren Pooler.

2.1.5.8.5 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.

2.1.5.8.6 Architektur

2.1.5.8.6.1 StackGres

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

Abbildung 2.45: Stackgres - Grundarchitektur

2.1.5.8.6.2 Citus Coordinator und Workers

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

Abbildung 2.46: Citus - Coordinator und Workers

2.1.5.8.6.3 Citus Sharding

Citus bietet zwei Sharding-Modelle an.

Row-based sharding Beim diesen sharding werden Tabellen anhand einer Distribution Column aufgeteilt. [16, 8]

Abbildung 2.47: Citus - Row-Based-Sharding

Schema-based sharding

Abbildung 2.48: Citus - Schema-Based-Sharding

Schlussfolgerung Beide Sharding-Methoden haben eine grosse Schwäche. In Version 7.2 konnte noch ein Replikationsfaktor angegeben werden[15], ab Version 11 wurde auch diese Variante gestrichen und man konnte noch eine 1:1 Repliacation auf einen Worker fahren[10]. 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[14]. 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 2.49: 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.

2.1.5.8.7 Maintenance

Bei Stackgres gab es im letzten Monat keine wirkliche Bewegung:

Abbildung 2.50: 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 2.51: Citus - Pulse

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

Abbildung 2.52: Stackgres - Code Frequency

Abbildung 2.53: Citus - Code Frequency

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

Abbildung 2.54: Stackgres - Community Standards

Abbildung 2.55: 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 2.56: Stackgres - Contributors Commits

Abbildung 2.57: Stackgres - Contributors Deletations

Abbildung 2.58: Stackgres - Contributors Additions

Abbildung 2.59: Citus - Contributors Commits

Abbildung 2.60: Citus - Contributors Deletations

Abbildung 2.61: 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 2.62: Stackgres - Commit Activity

Abbildung 2.63: 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 2.64: Stackgres - Network Graph

Abbildung 2.65: Citus - Network Graph

2.1.5.9 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[88].

2.1.5.9.1 Core-Features

Die wichtigsten Features von YugabyteDB sind[7]:

- PostgreSQL Kombatibel
- Cassandra-Kompatibilität
- Horizontal skalierbarkeit
- Global verteilbar
- Cloud Native

2.1.5.9.2 Replikation

2.1.5.9.3 Proxy

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

2.1.5.9.4 Pooling

YugabyteDB hat ein Connection Pooling mit dem YSQL Connection Manager integriert[58].

2.1.5.9.5 API / Skripte

YugabyteDB bietet eigene APIs[5] und CLIs[12] für das Verwalten an.

Diese bieten auch die möglichkeit, abgesichert zu werden.

2.1.5.9.6 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 2.66: yugabyteDB - Grundkonzept

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

Abbildung 2.67: yugabyteDB - Architektur

2.1.5.9.6.1 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 2.68: yugabyteDB - Sharding

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

Abbildung 2.69: 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 [35] geregelt:

Abbildung 2.70: yugabyteDB - Tablet - Replikationsfaktor

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

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

Abbildung 2.71: 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`[32].

Abbildung 2.72: yugabyteDB - Zone outage Tolerance

2.1.5.9.7 Maintenance

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

Abbildung 2.73: YugabyteDB - Pulse

Die Code Frequency kann nicht ausgegeben werden, es gab zu viele Commits:

Abbildung 2.74: YugabyteDB - Code Frequency

Das Projekt hält nur die wichtigsten Community SStandards ein:

Abbildung 2.75: YugabyteDB - Community Standards

Es werden immer wieder Commits abgesetzt, allerdings sind diese nicht weiter aufgeteilt in Commits, Additions und Deletations:

Abbildung 2.76: YugabyteDB - Contributors

Die Commits wiederum werden Regelmässig ausgeführt, es wird scheinbar in kurzen Sprints gearbeitet:

Abbildung 2.77: YugabyteDB - Commit Activity

YugabyteDB ist der Maintainer seines Produkts.
Es gibt keine anderen Grossen Contributors:

Abbildung 2.78: YugabyteDB - Network Graph

2.1.5.9.8 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.

2.1.6 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 2.9: 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 2.10: Vorauswahl - Evaluation

2.1.7 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

Continued on next page

Tabelle 2.11: Evaluationssyssteme

Server	Typ	Funktion	Full Qualified Device Name	IP
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 2.11: Evaluationssyssteme

2.1.7.1 rke2 - Evaluationsplattform

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

Abbildung 2.79: 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 2.12: Evaluationssysstem - Distributed SQL / Sharding

2.1.7.2 Patroni

2.1.7.2.1 Architektur

Abbildung 2.80: Patroni - Evaluationsarchitektur

2.1.7.3 StackGres - Citus

2.1.7.3.1 Architektur

Abbildung 2.81: Stackgres - Citus - Evaluationsarchitektur

2.1.7.4 yugabyteDB

2.1.7.4.1 Installation

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

(Repository yugawre):

```
gramic@cks4040: ~
Context: default ✏️                    <c>       Copy
Cluster: default                        <e>       Edit
User:    default                        <n>       Next Match
K9s Rev: v0.31.8 ⚡v0.32.4              <shift-n> Prev Match
K8s Rev: v1.29.0+rke2r1                 <r>       Toggle Auto-Refresh
CPU:     1%                             <f>       Toggle FullScreen
MEM:     38%
                                        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 2.82: 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 2.1: 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[27], wo oben genannter Ansatz empfohlen wurde.

2.1.7.4.2 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 2.2: 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 2.3: 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"
```

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

Listing 2.4: yugabyteDB - Helm Chart Manifest - Detail Resources

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

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

Listing 2.5: yugabyteDB - Helm Chart Manifest - Detail Replika

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

```
1 ...
2 # Disable the YSQL
```

```
disableYsql: false
...

```

Listing 2.6: yugabyteDB - Helm Chart Manifest - Detail Disable YSQL

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

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

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

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

```

Listing 2.7: yugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints

2.1.7.4.3 Connection

Auch mit einem PostgreSQL-

2.1.8 Gegenüberstellung der Lösungen

2.1.8.0.1 yugabyteDB

Zuerst wurde für die Annährend 5GiB-DB initialisiert:

```
yugabyte=# create database pgbench_eval_bench;
CREATE DATABASE
```

Listing 2.8: yugabyteDB - Benchmarking - DB erstellen

2.1.8.1 Benchmarks

Der vergleich zwischen den verschiedenen Varianten.

> ☞ **YugabyteDB**
> 4GiB Memory für die `tserver` waren offensichtlich zu knapp bemessen. Zumindest wenn die Tabelle 120'000'000 Rows hat und ein mixed Benchmark abgesetzt wird.
>
> Dies äusserte sich in einem Fehler (Absturz)auf zwei von drei `tserver`-Nodes sowie einer hohen Anzahl an Fehlern bei den mixed-Benchmarks. Daher wurde das Memory auf 8GiB erhöht und die komplette Testreihe erneut gestartet. Zudem wurden die Anzahl Fehler ebenfalls in die Benchmark-Auswertung einbezogen.

Bei den Transaktionen pro Sekunden gilt, je höher der Wert, umso besser das Ergebnis.
Zuerst die Ergebnisse mit den mixed-Transaktionen:

Abbildung 2.83: Benchmarks - tps mixed

Folgend die reinen Select-Transaktionen.

Abbildung 2.84: Benchmarks - tps dql

Bei der Latenz ist es genau andersrum, je höher der Wert desto schlechter schnitt die Variante ab.
Auch hier zuerst wieder die mixed-Transaktionen:

Abbildung 2.85: Benchmarks - latency mixed

Folgend die Select-Transaktionen:

Abbildung 2.86: Benchmarks - latency dql

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

(a) absolute

(b) percentage

Abbildung 2.87: Benchmarks - Fehler bei mixed-Transaktionen

2.1.8.2 Gemessene Zeiten und Kosten

2.1.8.2.1 Gegebene Parameter und Annahmen

2.1.8.2.2 Gemessene Zeiten

2.1.8.2.3 Zeitvergleiche

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

Tabelle 2.13: Gemessene und Extrapolierte Aufwände Bsp.

Abbildung 2.88: Zeitaufwände

2.1.8.2.4 Kostenvergleiche

Abbildung 2.89: Kostenaufwände

2.1.8.3 Kosten-Nutzen

Abbildung 2.90: Kosten-Nutzen-Analyse

2.1.9 Entscheid

2.2 Aufbau und Implementation Testsystem

2.2.1 Bereitstellen der Grundinfrastruktur

2.2.2 Installation und Konfiguration PostgreSQL HA Cluster

2.2.3 Technical Review der Umgebung

2.3 Testing

2.3.1 Testing

2.3.2 Protokollierung

2.3.3 Review und Auswertung

2.4 Troubleshooting und Lösungsfindung

3 Resultate

3.1 Zielüberprüfung

3.2 Schlussfolgerung

3.3 Weiteres Vorgehen / offene Arbeiten

3.4 Persönliches Fazit

Abbildungsverzeichnis

1.1 Spitalregionen Kanton Graubünden[53] . . . 1
1.2 Wahlkreise Kanton St. Gallen[78] . . . 2
1.3 Spitalregionen / Spitalstrategie Kanton St. Gallen[47] . . . 3
1.4 Organigramm Kantonsspital Graubünden . . . 4
1.5 Organigramm Departement 10 - ICT . . . 5
1.6 Datenbanken - Aufgeschlüsselt nach RDBMS . . . 8
1.7 Datenbanken - Aufgeschlüsselt nach Betriebssystem . . . 9
1.8 Risiken bestehende Lösung . . . 12
1.9 Risiken bestehende Lösung mit Massnahmen . . . 13
1.10 Systemabgrenzung . . . 18
1.11 Projektrisiken . . . 21
1.12 Projektrisiken mit Massnahmen . . . 22
1.13 Riskikomatrix - Assessment 21.03.2024 . . . 26
1.14 Projektcontrolling . . . 28

2.1 Sharding - Vertikale Partitionierung . . . 34
2.2 Sharding - Horizontales Partitionierung . . . 35
2.3 Monolithische vs. verteilte SQL Systeme . . . 37
2.4 CAP-Theorem . . . 40
2.5 Datenbankskalierung . . . 41
2.6 Präferenzmatrix . . . 45
2.7 Präferenzmatrix - Failover . . . 46
2.8 Präferenzmatrix - Switchover . . . 47
2.9 Präferenzmatrix - Restore . . . 48
2.10 Präferenzmatrix - Replikation . . . 49
2.11 Präferenzmatrix - Sharding . . . 50
2.12 Präferenzmatrix - Quorum . . . 51
2.13 Präferenzmatrix - Management-API . . . 52
2.14 Präferenzmatrix - Backup . . . 53
2.15 Präferenzmatrix - Performance . . . 54
2.16 Benchmark Settings - Zabbix - Systeminformationen . . . 56
2.17 Benchmark Settings - Zabbix - Connections per Seconds . . . 56
2.18 Benchmark Settings - Zabbix - Queries per Seconds . . . 56
2.19 Benchmark Settings - Zabbix - Client Queries per Seconds . . . 57
2.20 Benchmark Settings - Zabbix - DB Size . . . 57

2.21 Benchmark Settings - Anzahl Records / Skalierungsfaktor 60
2.22 pg_auto_failover-Architektur - Single Standby 64
2.23 pg_auto_failover-Architektur - Multi-Node Standby 64
2.24 pg_auto_failover-Architektur - Citus 65
2.25 CloudNativePG - Kubernetes - PostgreSQL 66
2.26 CloudNativePG - Kubernetes - Read-write workloads 67
2.27 CloudNativePG - Kubernetes - Read-only workloads 67
2.28 CloudNativePG - Pulse 68
2.29 CloudNativePG - Code Frequency 68
2.30 CloudNativePG - Community Standards 69
2.31 CloudNativePG - Contributors Commits 69
2.32 CloudNativePG - Contributors Deletations 70
2.33 CloudNativePG - Contributors Additions 70
2.34 CloudNativePG - Commit Activity 70
2.35 CloudNativePG - Network Graph 71
2.36 Patroni-Architektur 73
2.37 Patroni - Pulse 74
2.38 Patroni - Code Frequency 74
2.39 Patroni - Community Standards 75
2.40 Patroni - Contributors Commits 75
2.41 Patroni - Contributors Deletations 76
2.42 Patroni - Contributors Additions 76
2.43 Patroni - Commit Activity 76
2.44 Patroni - Network Graph 77
2.45 Stackgres - Grundarchitektur 79
2.46 Citus - Coordinator und Workers 80
2.47 Citus - Row-Based-Sharding 80
2.48 Citus - Schema-Based-Sharding 81
2.49 StackGres-Citus - Shard-Replikation 82
2.50 Stackgres - Pulse 82
2.51 Citus - Pulse 83
2.52 Stackgres - Code Frequency 83
2.53 Citus - Code Frequency 84
2.54 Stackgres - Community Standards 84
2.55 Citus - Community Standards 85
2.56 Stackgres - Contributors Commits 85
2.57 Stackgres - Contributors Deletations 86
2.58 Stackgres - Contributors Additions 86
2.59 Citus - Contributors Commits 86

2.60 Citus - Contributors Deletations 87
2.61 Citus - Contributors Additions 87
2.62 Stackgres - Commit Activity 87
2.63 Citus - Commit Activity 88
2.64 Stackgres - Network Graph 88
2.65 Citus - Network Graph 89
2.66 yugabyteDB - Grundkonzept 90
2.67 yugabyteDB - Architektur 91
2.68 yugabyteDB - Sharding 91
2.69 yugabyteDB - Tablet - Leader und Follower 92
2.70 yugabyteDB - Tablet - Replikationsfaktor 93
2.71 yugabyteDB - Zonen 94
2.72 yugabyteDB - Zone outage Tolerance 95
2.73 YugabyteDB - Pulse 96
2.74 YugabyteDB - Code Frequency 96
2.75 YugabyteDB - Community Standards 97
2.76 YugabyteDB - Contributors 97
2.77 YugabyteDB - Commit Activity 98
2.78 YugabyteDB - Network Graph 98
2.79 Evaluationssystem - Distributed SQL / Shards 100
2.80 Patroni - Evaluationsarchitektur 101
2.81 Stackgres - Citus - Evaluationsarchitektur 102
2.82 yugabyteDB - Susbsription yugawre 103
2.83 Benchmarks - tps mixed 108
2.84 Benchmarks - tps dql 109
2.85 Benchmarks - latency mixed 110
2.86 Benchmarks - latency dql 111
2.87 Benchmarks - Fehler bei mixed-Transaktionen 112
2.88 Zeitaufwände 114
2.89 Kostenaufwände 115
2.90 Kosten-Nutzen-Analyse 116

Tabellenverzeichnis

1.1 Inventarisierte Datenbanksysteme . . . 7
1.2 Datenbankinventar . . . 8
1.3 Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt . . . 9
1.4 Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken . . . 11
1.5 Administrative Aufgaben . . . 14
1.6 Automatisierung Administrativer Aufgaben . . . 15
1.7 Ziele . . . 16
1.8 Gegebene Systeme . . . 17
1.9 Abhängigkeiten . . . 19
1.10 Risiko-Matrix der Diplomarbeit . . . 20
1.11 Neu Erkannte / Erfasste Risiken . . . 23
1.12 Risiko-Assessment 21.03.2024 . . . 25
1.13 Projektcontrolling . . . 27
1.14 Initialer Statusbericht . . . 31
1.15 Zweiter Statusbericht . . . 32
1.16 Fachgespräche . . . 33

2.1 Quorum Beispiele . . . 38
2.2 Anforderungskatalog . . . 43
2.3 Stakeholder . . . 44
2.4 Benchmark Settings - Mixed Transaktionen . . . 57
2.5 Benchmark Settings - DQL Transaktionen . . . 58
2.6 Benchmark Settings - Skalierungsfaktoren . . . 58
2.7 Benchmark Settings - Datenbankgrössen / Skalierungsfaktor . . . 58
2.8 Benchmark Settings - Tabellengrössen / Skalierungsfaktor . . . 59
2.9 Vorauswahl - Ausgeschieden . . . 99
2.10 Vorauswahl - Evaluation . . . 99
2.11 Evaluationssyssteme . . . 100
2.12 Evaluationssysstem - Distributed SQL / Sharding . . . 100
2.13 Gemessene und Extrapolierte Aufwände Bsp. . . . 113

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

Listings

2.1 metallb - Konfig YAML - Detail L2Advertisement . . . 103
2.2 yugabyteDB - Helm Chart Manifest - Detail Image . . . 104
2.3 yugabyteDB - Helm Chart Manifest - Detail StorageClass . . . 104
2.4 yugabyteDB - Helm Chart Manifest - Detail Resources . . . 104
2.5 yugabyteDB - Helm Chart Manifest - Detail Replika . . . 105
2.6 yugabyteDB - Helm Chart Manifest - Detail Disable YSQL . . . 105
2.7 yugabyteDB - Helm Chart Manifest - Detail Domainname und Service-Endpoints . 106
2.8 yugabyteDB - Benchmarking - DB erstellen . . . 107

1 Proxy Settings . . . v
2 rke2 server - Verzeichnis erstellen . . . v
3 rke2 server - config.yaml . . . v
4 rke2 server - cilium-config.yaml . . . v
5 rke2 server installieren . . . vi
6 rke2 agenten installieren . . . vi
7 rke2 agent - config.yaml . . . vi
8 -rke2 agent service restart . . . vi
9 rke2 server proxy . . . vi
10 rke2 server proxy kopieren . . . vii
11 rke2 server cilium installieren . . . vii
12 rke2 server cilium aktivieren . . . vii
13 rke2 server starten . . . vii
14 iptables entries server . . . vii
15 rke2 server token . . . viii
16 local-path-storage auf Linux Bereitstellen . . . viii
17 local-path-provisioner definieren . . . viii
18 local-path-storage aktualisieren . . . ix
19 MetallB installieren . . . ix
20 MetallB konfigurieren . . . ix
21 MetallB Konfiguration einspielen . . . x
22 yugabyteDB - StorageClass setzen . . . x
23 yugabyteDB - StorageClass / PersistentVolume aktivieren . . . xi
24 yugabyteDB - Namespace . . . xi
25 yugabyteDB - Helm Chart Manifest . . . xi
26 yugabyteDB - Installation . . . xxiii

27 StackGres-Citus - StorageClass setzen xxiv
28 yugabyteDB - StorageClass / PersistentVolume aktivieren xxiv
29 sks9016 - Download yugabyteDB On-Premise xxv
30 sks9016 - Installation yugabyteDB On-Premise xxv
31 sks9016 - Check yugabyteDB On-Premise xxv
32 Patroni - Proxy Settings xxvi
33 Patroni - apt-Proxy Settings xxvi
34 Patroni - PostgreSQL einbinden xxvi
35 Python LaTex - zotero.py - Zotero BibLaTex Importer xxvii
36 Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer xxxiii
37 Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer xxxiii
38 Python LaTex - riskmatrix.py - Risxikomatrizen xli
39 Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen xlv
40 Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples xlviii
41 Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm xlix
42 Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm li
43 Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle lii
44 Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle lxi
45 Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm lxxxi
46 Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme lxxxviii

Literatur

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[27] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[47] Kanton St. Gallen - Amt für Gesundheitsversorgung und Staatskanzlei Kanton St. Gallen - Dienststelle Kommunikation. *Weiterentwicklung der Strategie der St.Galler Spitalverbunde | sg.ch*. https://www.sg.ch/gesundheit-soziales/gesundheit/gesundheitsversorgung--spitaeler-s spitaeler-kliniken/spitalzukunft.html.

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

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

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

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

[52] 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.

[53] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[69] 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.

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

[71] 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.

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

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

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

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

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

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

[78] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[93] Rainer Züst. „Einstieg ins Systems Engineering“. In: (2002).

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[70].. 17

AUTOVACUUM Der AUTOVACUUM Job räumt die Tablespaces und Data Files innerhalb von PostgreSQL sowie auf dem Filesystem nach Lösch- und Manipulations-Transaktionen auf, aktualisiert Datenbank interne Statistiken und verhindert Datenverlust von selten genutzten Datensätzen[55].. 15, 16, 55

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[80] wurde.. 7, 89, 90, 98

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[71].. 5

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[9, 25].. 103, 104

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[81].. 5

DCS Der DSC ist eine Kernkomponente von Patroni [17]. Realisiert wird der DCS bei Patroni mit Etcd.. 6

Debian Debian gehört nebst Slackware Linux zu den ältesten Linux Distribution die noch immer gepflegt und eingesetzt werden. Sie wurde im August 1993 gestartet und brachte im Laufe der Zeit einige der beliebstesten Distributionen wie Ubuntu hervor.. 17

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[73].. 7

etcd etcd ist [20]. 72, 131, xxvii

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 16, 37, 38, 61, 132

Foreman Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 17, 20, xxvi

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[48, 90].. 132

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**].. xxvii

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[36].. 16, 61

HAProxy HAProxy [23]. 63, 72

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[46]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 16, 61

HP-UX Dieses UNIX-Derivat ist ein abkömmling von System III, System V R3 und System V R4 und wurde von HP zum ersten Mal 1982 veröffentlicht.. 5, 9, 20

IBM DB2 IBM DB2 ist eine Relationale Datenbank[66] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 40

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[72, 38].. 61

Key-Value-orientierte Siehe Key-Value-Datenbank. 135

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. 132, 133, 134

Key-Value-Store Siehe Key-Value-Datenbank. 58, 90

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.. 5, 9, 17, 132

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, 133

MariaDB MariaDB ist ein MySQL Fork des ehemaligen MySQL Mitbegründers Michael Widenius, wobei sich der Name Maria aus dem VOrnamen einer seiner Töchter ableitet. NAch dem Fork 2009 blieb MariaDB für eine Zeitlang sehr ähnlich mit MySQL und behielt ein ähnliches Versionierungsschema bei. Dies änderte sich 2012 wo dann direkt mit der Version 10 weitergefahren wurde. Beide Datenbanken entfernen sich im Lauf eder Zeit immer mehr voneinander undf sind nicht mehr in jdem Fall kompatibel oder beliebig austauschbar. Auf den Linux Distributionen tratt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 9, 60

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[28].. 103

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[39].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[40].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[41]. 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, 134

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zurm ersten Mal 2007 veröffentlicht wurde[82].. 7

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

zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 5, 7, 9, 60

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, 131, 132, 133, 135, 136

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[51, 50]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 5, 7

Oracle Linux Oracle Linux ist eine RHEL-Distribution der Firma Oracle und ist mit RHL Binärkompatibel. Sie wird primär für den Betrieb von Oracle Datenbanken verwendet und komnt auf den Oracle Eigenenen Appliances ODA und Exadata zum Einsatz. Für den Zweck als DB Plattform kann ein für Oracle Datenbanken optimimierter Kernel verwendet werden. Zu Oracle Linux kann ein kostenpflichtiger Support bezogen werden, allerdings ist die Distribution anders als RHEL auch ohne Lizenz erhältlich.. 17

Oracle Database Die erste verfügbare Version der Oracle Datenbank kam im Jahr 1979 mit Version 2 (statt Version 1) heraus, damals allerdings nur mit den Basisfunktionen. Im Laufe der Zeit wuchs der Funktionsumfang sehr stark an, die Grundlage des Client-Server-Designs kam erstmals im Jahr 1985 mit Version auf den Markt und hat sich im Prinzip bis heute gehalten. Mit der mit Version 8/8i 1997 erschienen Optimizer und mit der Version 9i 2001 erschienenn Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangheit ermöglichen) konnte Oracle sich stark von der Konkurenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Perforamce oder grosse Datenmengen geht.. 5, 7, 9, 40, 134

PKI . 6

PostgreSQL Die OpenSource Datenbank PostgreSQL wurde in Form von POSTGRES zum ersten Mal 1986 von der University of California at Berkeley veröffentlicht. und zählt zu den beliebstesten OpenSource Datenbanken. Zudem besteht in vielen bereichen eine gewisse ähnlichkeit zu Oracles Oracle Database.. 5, 7, 9, 10, 14, 40, 61, 89

PostgreSQL HA Cluster Der HA Cluster des PostgreSQL Clusters. 16

PostgreSQL Cluster Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wei Oracle.. 15, 16, 61, 134

PRTG Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum erstmals veröffentlicht und war ebenfalls als Netzwerkmonitoring System konzipiert. Wie bei Zabbix lässt sich heute damit ebenfalls fast jedes IT-System damit Überwachen. Reichen die Zahlreich vorhanden Standard Sensoren nicht, können eigene Sensoren geschrieben werden. PRTG ist nicht Open-Source, man bezahlt anhand gewisser Sensor Packages.. 5, 15, 17

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 2.1.1.4 wird genauer auf die Mechanik eines Quorums eingegangen. . 61, 132

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[45].. 5, 90

RedHat Enterpise Linux (RHEL) RHEL wurde in seiner Ursprüglichen Form Red Hat Linux (RHL) bis in den Oktober 1994 zurück, wobei die erste Version von RHEL wie es heute existiert im Jahr 2002 erfolgte. RHEL ist auf lange Wartungszyklen von fünf Jahren und grosskunden ausgelegt. Ohne entsprechenden Supportvertrag kann keine ISO-Datei bezogen werden. Somit hebt sich RHEL stark von aderen Linux Distributionen ab.. 17

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[85]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 132

rke2 rke2 ist eine Leichgewichtige Kubernetes Distribution, die alles notwendige mitbringt, um einen k8s Cluster zu betreiben[24].. 100, 103

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

SAN Ein Storage Area Network ist ein dediziertes Netzwerk aus Storage Komponenten. SAN Systeme bieten redundante Pools an Speicher. Die Physischen Festplatten werden zu Virtuellen Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[43, 74, 52]. 5, 17, 20

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[86].. 5, 17

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[89, 79]. 7

Split-brain Im Kapitel **??** werden die ursachen und folgenden eines Split-brains genauer besprochen. . 38, 135

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[61, 68]. . 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[83].. 7

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

SWOT-Analyse Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzueigen. Anhand einer SWOT-Analyse werden i.d.R. anschliessend Strategien abgeleitet um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder anzumildern.. 5

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[84, 59].. 17

Transaktion Eine Transaktion ist beinhaltet Schreib-, Lese-, Mutatations- oder Löschoperatione auf Daten.. 55, 57

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.. 5

VRRP VRRP . 5, 132

Zabbix Das 2001 veröffentlichte Open-Source Monitoring System Zabbix gilt zwar als Netzwerk-Monitoring System, allerdings kann heute nahezu jedes IT-System damit überwacht werden. Zabbrix speichert die Metriken und nicht die Auswertungen, das heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert worden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge Abschliessen.. 9, 17

Selbstständigkeitserklärung

Ich versichere, dass die vorliegende Arbeit von den Autoren selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Alle Inhalte dieser Arbeit, dazu gehören neben Texten auch Grafiken, Programmcode, etc., die wörtlich oder sinngemäss aus anderen Quellen stammen, sind als solche eindeutig kenntlich gemacht und korrekt im Quellenverzeichnis gelistet. Dies gilt auch für einzelne Auszüge aus fremden Quellen.

Die Arbeit ist in gleicher oder ähnlicher Form noch nicht veröffentlicht und noch keiner Prüfungsbehörde vorgelegt worden.

Ort, Datum, Unterschrift

Haftungsausschluss

Der vorliegende Bericht wurde von Studierenden im Rahmen einer Diplomarbeit erarbeitet. Es muss an dieser Stelle darauf hingewiesen werden, dass die Arbeit nicht im Rahmen eines Auftragsverhältnisses erstellt wurde. Weder der Ersteller noch die ibW Höhere Fachhochschule Südostschweiz können deshalb für Aktivitäten auf der Basis dieser Diplomarbeit eine Haftung übernehmen.

I Arbeitsrapport

Datum	Von	Bis	Dauer [h]	Phase	Subphase	Tätigkeit	Bemerkung	Schwierigkeit	Lösungen
14.02.2024	19:00	20:00	1.0	1. Expertengespräch	1. Expertengespräch				
21.02.2024	15:00	16:00	1.0	Evaluation	Anorderungskatalog	Anorderungskatalog erarbeiten			
22.02.2024	16:00	17:30	1.5	Evaluation	Anorderungskatalog	Anorderungskatalog erarbeiten			
27.02.2024	10:00	11:30	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
27.02.2024	13:00	16:00	3.0	Dokumentation	Dokumentation	Dokumentation erweitern		Viele LaTEX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken
28.02.2024	09:00	11:00	2.0	Dokumentation	Dokumentation	Dokumentation erweitern		Viele LaTEX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken
01.03.2024	07:00	09:00	2.0	Dokumentation	Dokumentation	Dokumentation Exkurs Architektur	Um Entscheidungen Transparent zu machen, müssen Grundlegende Konzepte aufgezeigt werden. Nicht alle Konzepte wie z.B. Distributed SQL sind bekannt resp. das zusammenspiel mit Kubernetes.	Konzepte wie Distributed SQL sind nicht einfach zu erklären.	
08.03.2024	07:00	09:00	2.0	Evaluation	Anorderungskatalog	Anorderungskatalog erarbeiten			
11.03.2024	07:00	11:30	4.5	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Informationen Sammeln	pgpool II	pgpool II hat kein GitHub Repository. Das macht es unmöglich, diese Lösung mit all den anderen zu vergleichen.	pgpool II fällt somit direkt aus der betrachtung raus, da kein vergleich möglich ist.
11.03.2024	12:00	13:30	1.5	Dokumentation	Dokumentation	Dokumentation erweitern			
11.03.2024	16:45	17:30	0.5	Dokumentation	Dokumentation	Dokumentation erweitern	Stakeholder erfassen		
13.03.2024	17:45	19:45	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Stackgres und Citus analysieren	Citus row-based-sharding	Citus Dokumentation stark Textlastig. Wenig Abbildungen, vieles muss selber gezeichnet werden.	
14.03.2024	19:45	20:45	1.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen		Citus row-based-sharding		
14.03.2024	20:45	21:30	0.8	Dokumentation	Dokumentation	Projektcontrolling Arbeiten	Citus row-based-sharding Dokumentieren		
16.03.2024	17:45	18:30	0.8	Dokumentation	Dokumentation	Projektcontrolling Arbeiten			
17.03.2024	14:45	16:30	1.8	Dokumentation	Dokumentation	Dokumentation erweitern	Zweiter Statusbericht verfassen		
17.03.2024	19:30	20:00	0.5	Dokumentation	Dokumentation	Dokumentation erweitern	ACID Exkurs erfassen		
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

Tabelle I: Arbeitsrapport

II Protokoll - Fachgespräche

Fachgespräch	Datum	Fachexperte	Nebenexperte	Studenten	Fragen	Antworten	Sonstige Themen	Bemerkungen
1	14.02.2024	Norman Süsstrunk	-	Michael Graber Curdin Roffler	- Darf eine Vorauswahl stattfinden, um den Aufwand zur reduzieren?	- Eine Vorauswahl ist Sinnvoll und in diesem Rahmen fast zwingend Notwendig, da sonst viel zuviel Zeit investiert werden müsste	- Vorstellung Norman Süsstrunk, Curdin Roffler und Michael Graber - Kontaktdaten shared - Bei Fragen jederzeit an Norman wenden - Norman braucht aber mindestens 1. Woche vorlaufzeit - Norman wird sich spätestens zur Halbzeit melden. - Norman wird sic	- Es wurden zwar für alle Studenten von Norman Süsstrunk Zoom-Räume bereitgestellt, aus effizienzgründen nahmen Curdin Roffler und ich beide am selben Meeting teil
2	26.03.2024	Norman Süsstrunk	-	Michael Graber	- Muss das Protokoll des Fachgesprächs jeweils Zeitnah freigegeben werden? - Hat Norman ggf. noch vorschläge zu PostgreSQL Clustern gefunden? - Soll ich die Gewichtung mit 100 Punkten machen oder 1000? Im Moment haben diverse Punkte eine sehr kleine Punktzahl - Soll die Disposition in den Anhang? Diese ist 50 Seiten lang		- Protokoll genehmigen	

Tabelle II: Fachgespräche - Protokoll

=:

III Kommentare / Anmerkungen

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

Woche	Beschreibung / Event / Problem
KW10	Vier ganze Tage war ich in Thalwil für die Oracle Multitenant-Schulung für das ExaCC Projekt (Ablösung HP-UX). Am Freiutag war ich ebenfalls fast den ganzen Tag dran. Weitere Termine werden folgen, das Risiko durch das Projekt tritt langsam ein. Projekt Zeitlich im Verzug.
KW11	Nebst dem HP-UX Ablösungsprojekt schlagen auch diverse Betriebsthemen ein. Die analyse der PostgreSQL HA Cluster nimmt ebenfalls mehr Zeit in Anspruch, als erwartet.
KW12	- HP-UX Probleme am Montag. Backups sind über das Weekend nicht durchgelaufen. Die ganze Montagsplanung wurde über den Haufen geworfen. - Besprechung bezüglich Backup. Veeam Kasten steht noch nicht zur Verfügung.
KW12	- Mittwochvormittag in Zürich, am Nachmittag Probleme mit dfs-Shares. So wenig Zeit. - 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. Später bemerkte ich, dass man das Repository yugabyte verwendet muss.
KW13	- Bereits jetzt viel gerlernt über Kubernetes, Ranger (rke2) und Helm

Tabelle III: Kommentare - Anmerkung

IV rke2

IV.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

IV.II Installation

IV.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

IV.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

IV.III Cluster Konfiguration

IV.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

IV.III.II agents

IV.III.III local-path-provisioner

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

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

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

root@sks1185:~# mkdir /var/local-path-provisioner
root@sks1185:~# chmod -R 777 /var/local-path-provisioner/
```

Listing 16: local-path-storage auf Linux Bereitstellen

Anschliessend musste rke2 entsprechend angepasst werden. Damit Automatisch der local-path auf das Verzeichnis `/var/local-path-provisioner/` geht, muss dies in einem entsprechenden Manifest geschrieben werden:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: local-path-config
  namespace: local-path-storage
data:
  config.json: |-
        {
                "nodePathMap":[
```

```
                {
                        "node":"DEFAULT_PATH_FOR_NON_LISTED_NODES",
                        "paths":["/var/local-path-provisioner"]
                }
                ]
        }
  setup: |-
        #!/bin/sh
        set -eu
        mkdir -m 0777 -p "$VOL_DIR"
  teardown: |-
        #!/bin/sh
        set -eu
        rm -rf "$VOL_DIR"
  helperPod.yaml: |-
        apiVersion: v1
        kind: Pod
        metadata:
          name: helper-pod
        spec:
          priorityClassName: system-node-critical
          tolerations:
            - key: node.kubernetes.io/disk-pressure
              operator: Exists
              effect: NoSchedule
          containers:
          - name: helper-pod
            image: busybox
```

Listing 17: local-path-provisioner definieren

Zuerst mussten auf den drei Servern der Storage bereitgestellt werden:

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

Listing 18: local-path-storage aktualisieren

IV.III.IV MetallB - Proxy / Load Balancer

MetallB musste installiert werden:

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

Listing 19: MetallB installieren

Das Konfigurationsmanifest wurde eingespielt:

```
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

V yugabyteDB

V.I Prerequisites

V.I.I Prometheus Daten

V.I.II 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 22: yugabyteDB - StorageClass setzen

Die Storage Class und das PercistenVolume 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 23: yugabyteDB - StorageClass / PersistentVolume aktivieren

V.II Installation

Zuerst muss ein Namespace erstellt werden:

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

Listing 24: yugabyteDB - Namespace

```
# 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"
```

```
21   tserver:
22     count: 1
23     size: 3Gi
24     storageClass: "yb-storage"
25
26 resource:
27   master:
28     requests:
29       cpu: "1"
30       memory: 2Gi
31     limits:
32       cpu: "1"
33       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating
     from these formats
34       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes
     ' flag.
35       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so
     may lead to
36       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects
     integer values.
37       memory: 2Gi
38   tserver:
39     requests:
40       cpu: "1"
41       memory: 4Gi
42     limits:
43       cpu: "1"
44       ## Ensure the 'memory' value is strictly in 'Gi' or 'G' format. Deviating
     from these formats
45       ## may result in setting an incorrect value for the 'memory_limit_hard_bytes
     ' flag.
46       ## Avoid using floating numbers for the numeric part of 'memory'. Doing so
     may lead to
47       ## the 'memory_limit_hard_bytes' being set to 0, as the function expects
     integer values.
48       memory: 4Gi
49
50 replicas:
51   master: 3
52   tserver: 3
53   ## Used to set replication factor when isMultiAz is set to true
54   totalMasters: 3
55
56 partition:
57   master: 0
58   tserver: 0
59
60 updateStrategy:
```

```yaml
  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.
  ## When certManager.enabled=true and boostrapSelfsigned=true, rootCA is ignored.
  ## When certManager.enabled=true and bootstrapSelfsigned=false, only rootCA.cert is used
  ## to verify TLS certs generated and signed by the external provider.
```

```
105   rootCA:
106     cert: "
      LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM2VENDQWRHZ0F3SUJBZ0lCQVRBTkJna3Foa2lH0XcwQkFRc0
      ="
107     key: "
      LS0tLS1CRUdJTiBSU0EgUFJJVkFURSBLRVktLS0tLQpNSUlFcEFJQkFBS0NBUUVBdU4xYXVpZzhvalUwczQ5cXdBeG
      ="
108   ## When tls.certManager.enabled=false
109   ## nodeCert and clientCert will be used only when rootCA.key is empty.
110   ## Will be ignored and genSignedCert will be used to generate
111   ## node and client certs if rootCA.key is provided.
112   ## cert and key are base64 encoded content of certificate and key.
113   nodeCert:
114     cert: ""
115     key: ""
116   clientCert:
117     cert: ""
118     key: ""
119
120 gflags:
121   master:
122     default_memory_limit_to_ram_ratio: 0.85
123   tserver: {}
124 #   use_cassandra_authentication: false
125
126 PodManagementPolicy: Parallel
127
128 enableLoadBalancer: true
129
130 ybc:
131   enabled: false
132   ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers
      /#resource-requests-and-limits-of-pod-and-container
133   ## Use the above link to learn more about Kubernetes resources configuration.
134   # resources:
135   #   requests:
136   #     cpu: "1"
137   #     memory: 1Gi
138   #   limits:
139   #     cpu: "1"
140   #     memory: 1Gi
141
142 ybCleanup: {}
143   ## https://kubernetes.io/docs/concepts/configuration/manage-resources-containers
      /#resource-requests-and-limits-of-pod-and-container
144   ## Use the above link to learn more about Kubernetes resources configuration.
145   # resources:
146   #   requests:
```

```
147   #       cpu: "1"
148   #       memory: 1Gi
149   #     limits:
150   #       cpu: "1"
151   #       memory: 1Gi
152
153 domainName: "cluster.local"
154
155 serviceEndpoints:
156   - name: "yb-master-ui"
157     type: LoadBalancer
158     annotations: {}
159     clusterIP: ""
160     ## Sets the Service's externalTrafficPolicy
161     externalTrafficPolicy: ""
162     app: "yb-master"
163     loadBalancerIP: ""
164     ports:
165       http-ui: "7000"
166
167   - name: "yb-tserver-service"
168     type: LoadBalancer
169     annotations:
170       metallb.universe.tf/loadBalancerIPs: 10.0.20.106
171     clusterIP: ""
172     ## Sets the Service's externalTrafficPolicy
173     externalTrafficPolicy: ""
174     app: "yb-tserver"
175     loadBalancerIP: ""
176     ports:
177       tcp-yql-port: "9042"
178       tcp-yedis-port: "6379"
179       tcp-ysql-port: "5433"
180
181 Services:
182   - name: "yb-masters"
183     label: "yb-master"
184     skipHealthChecks: false
185     memory_limit_to_ram_ratio: 0.85
186     ports:
187       http-ui: "7000"
188       tcp-rpc-port: "7100"
189
190   - name: "yb-tservers"
191     label: "yb-tserver"
192     skipHealthChecks: false
193     ports:
194       http-ui: "9000"
```

```
195       tcp-rpc-port: "9100"
196       tcp-yql-port: "9042"
197       tcp-yedis-port: "6379"
198       tcp-ysql-port: "5433"
199       http-ycql-met: "12000"
200       http-yedis-met: "11000"
201       http-ysql-met: "13000"
202       grpc-ybc-port: "18018"
203
204
205 ## Should be set to true only if Istio is being used. This also adds
206 ## the Istio sidecar injection labels to the pods.
207 ## TODO: remove this once
208 ## https://github.com/yugabyte/yugabyte-db/issues/5641 is fixed.
209 ##
210 istioCompatibility:
211   enabled: false
212
213 ## Settings required when using multicluster environment.
214 multicluster:
215   ## Creates a ClusterIP service for each yb-master and yb-tserver
216   ## pod.
217   createServicePerPod: false
218   ## creates a ClusterIP service whos name does not have release name
219   ## in it. A common service across different clusters for automatic
220   ## failover. Useful when using new naming style.
221   createCommonTserverService: false
222
223   ## Enable it to deploy YugabyteDB in a multi-cluster services enabled
224   ## Kubernetes cluster (KEP-1645). This will create ServiceExport.
225   ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-
      cluster-services#registering_a_service_for_export
226   ## You can use this gist for the reference to deploy the YugabyteDB in a multi-
      cluster scenario.
227   ## Gist - https://gist.github.com/baba230896/78cc9bb6f4ba0b3d0e611cd49ed201bf
228   createServiceExports: false
229
230   ## Mandatory variable when createServiceExports is set to true.
231   ## Use: In case of GKE, you need to pass GKE Hub Membership Name.
232   ## GKE Ref - https://cloud.google.com/kubernetes-engine/docs/how-to/multi-
      cluster-services#enabling
233   kubernetesClusterId: ""
234
235   ## mcsApiVersion is used for the MCS resources created by the
236   ## chart. Set to net.gke.io/v1 when using GKE MCS.
237   mcsApiVersion: "multicluster.x-k8s.io/v1alpha1"
238
239 serviceMonitor:
```

```
240 ## If true, two ServiceMonitor CRs are created. One for yb-master
241 ## and one for yb-tserver
242 ## https://github.com/coreos/prometheus-operator/blob/master/Documentation/api.
      md#servicemonitor
243 ##
244 enabled: false
245 ## interval is the default scrape_interval for all the endpoints
246 interval: 30s
247 ## extraLabels can be used to add labels to the ServiceMonitors
248 ## being created
249 extraLabels: {}
250   # release: prom
251
252 ## Configurations of ServiceMonitor for yb-master
253 master:
254   enabled: true
255   port: "http-ui"
256   interval: ""
257   path: "/prometheus-metrics"
258
259 ## Configurations of ServiceMonitor for yb-tserver
260 tserver:
261   enabled: true
262   port: "http-ui"
263   interval: ""
264   path: "/prometheus-metrics"
265 ycql:
266   enabled: true
267   port: "http-ycql-met"
268   interval: ""
269   path: "/prometheus-metrics"
270 ysql:
271   enabled: true
272   port: "http-ysql-met"
273   interval: ""
274   path: "/prometheus-metrics"
275 yedis:
276   enabled: true
277   port: "http-yedis-met"
278   interval: ""
279   path: "/prometheus-metrics"
280
281 commonMetricRelabelings:
282 # https://git.io/JJW5p
283 # Save the name of the metric so we can group_by since we cannot by __name__
      directly...
284 - sourceLabels: ["__name__"]
285   regex: "(.*)"
```

```
286     targetLabel: "saved_name"
287     replacement: "$1"
288   # The following basically retrofit the handler_latency_* metrics to label format
      .
289   - sourceLabels: ["__name__"]
290     regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(.*)"
291     targetLabel: "server_type"
292     replacement: "$1"
293   - sourceLabels: ["__name__"]
294     regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(.*)"
295     targetLabel: "service_type"
296     replacement: "$2"
297   - sourceLabels: ["__name__"]
298     regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(_sum|_count)?"
299     targetLabel: "service_method"
300     replacement: "$3"
301   - sourceLabels: ["__name__"]
302     regex: "handler_latency_(yb_[^_]*)_([^_]*)_([^_]*)(_sum|_count)?"
303     targetLabel: "__name__"
304     replacement: "rpc_latency$4"
305
306 resources: {}
307
308 nodeSelector: {}
309
310 affinity: {}
311
312 statefulSetAnnotations: {}
313
314 networkAnnotation: {}
315
316 commonLabels: {}
317
318 master:
319   ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
      affinity-v1-core
320   ## This might override the default affinity from service.yaml
321   # To successfully merge, we need to follow rules for merging nodeSelectorTerms
      that kubernentes
322   # has. Each new node selector term is ORed together, and each match expression
      or match field in
323   # a single selector is ANDed together.
324   # This means, if a pod needs to be scheduled on a label 'custom_label_1' with a
      value
325   # 'custom_value_1', we need to add this 'subterm' to each of our pre-defined
      node affinity
326   # terms.
327   #
```

```
328  # Pod anti affinity is a simpler merge. Each term is applied separately, and the
       weight is tracked.
329  # The pod that achieves the highest weight is selected.
330  ## Example.
331  # affinity:
332  #   podAntiAffinity:
333  #     requiredDuringSchedulingIgnoredDuringExecution:
334  #     - labelSelector:
335  #         matchExpressions:
336  #         - key: app
337  #           operator: In
338  #           values:
339  #           - "yb-master"
340  #       topologyKey: kubernetes.io/hostname
341  #
342  # For further examples, see examples/yugabyte/affinity_overrides.yaml
343  affinity: {}
344
345  ## Extra environment variables passed to the Master pods.
346  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
       envvar-v1-core
347  ## Example:
348  # extraEnv:
349  # - name: NODE_IP
350  #   valueFrom:
351  #     fieldRef:
352  #       fieldPath: status.hostIP
353  extraEnv: []
354
355  # secretEnv variables are used to expose secrets data as env variables in the
       master pod.
356  # TODO Add namespace also to support copying secrets from other namespace.
357  # secretEnv:
358  # - name: MYSQL_LDAP_PASSWORD
359  #   valueFrom:
360  #     secretKeyRef:
361  #       name: secretName
362  #       key: password
363  secretEnv: []
364
365  ## Annotations to be added to the Master pods.
366  podAnnotations: {}
367
368  ## Labels to be added to the Master pods.
369  podLabels: {}
370
371  ## Tolerations to be added to the Master pods.
372  ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
```

```
    toleration-v1-core
  ## 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:
#     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: {}
```

```
458
459 ## Labels to be added to the TServer pods.
460 podLabels: {}
461
462 ## Tolerations to be added to the TServer pods.
463 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
      toleration-v1-core
464 ## Example:
465 # tolerations:
466 # - key: dedicated
467 #   operator: Equal
468 #   value: experimental
469 #   effect: NoSchedule
470 tolerations: []
471
472 ## Sets the --server_broadcast_addresses flag on the TServer, no
473 ## preflight checks are done for this address. You might need to add
474 ## `use_private_ip: cloud` to the gflags.master and gflags.tserver.
475 serverBroadcastAddress: ""
476
477 ## Extra volumes
478 ## extraVolumesMounts are mandatory for each extraVolumes.
479 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
      volume-v1-core
480 ## Example:
481 # extraVolumes:
482 # - name: custom-nfs-vol
483 #   persistentVolumeClaim:
484 #     claimName: some-nfs-claim
485 extraVolumes: []
486
487 ## Extra volume mounts
488 ## Ref: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22/#
      volumemount-v1-core
489 ## Example:
490 # extraVolumeMounts:
491 # - name: custom-nfs-vol
492 #   path: /home/yugabyte/nfs-backup
493 extraVolumeMounts: []
494
495 ## Set service account for tserver DB pods. The service account
496 ## should exist in the namespace where the tserver DB pods are brought up.
497 serviceAccount: ""
498
499 ## Memory limit hard % (between 1-100) of the memory limit.
500 memoryLimitHardPercentage: 85
501
502
```

```
503 helm2Legacy: false
504
505 ip_version_support: "v4_only" # v4_only, v6_only are the only supported values at
    the moment
506
507 # For more https://docs.yugabyte.com/latest/reference/configuration/yugabyted/#
    environment-variables
508 authCredentials:
509   ysql:
510     user: "yadmin"
511     password: "TES2&Daggerfall"
512     database: ""
513   ycql:
514     user: ""
515     password: ""
516     keyspace: ""
517
518 oldNamingStyle: true
519
520 preflight:
521   # Set to true to skip disk IO check, DNS address resolution, and
522   # port bind checks
523   skipAll: false
524   # Set to true to skip port bind checks
525   skipBind: false
526
527   ## Set to true to skip ulimit verification
528   ## SkipAll has higher priority
529   skipUlimit: false
530
531 ## Pod securityContext
532 ## Ref: https://kubernetes.io/docs/reference/kubernetes-api/workload-resources/pod
    -v1/#security-context
533 ## The following configuration runs YB-Master and YB-TServer as a non-root user
534 podSecurityContext:
535   enabled: false
536   ## Mark it false, if you want to stop the non root user validation
537   runAsNonRoot: true
538   fsGroup: 10001
539   runAsUser: 10001
540   runAsGroup: 10001
541
542 ## Added to handle old universe which has volume annotations
543 ## K8s universe <= 2.5 to >= 2.6
544 legacyVolumeClaimAnnotations: false
```

Listing 25: 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 26: yugabyteDB - Installation

V.III Rekonfiguration mit 300GiB Storage

VI Stackgres mit Citus

VI.I Prerequisites

VI.I.I StorageClass setzen

Zuerst muss die StorageClass und das PersistentVolume gesetzt werden:

```
# https://docs.yugabyte.com/preview/yugabyte-platform/install-yugabyte-platform/
    prepare-environment/kubernetes/#configure-storage-class
# https://github.com/rancher/local-path-provisioner
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: stackgres-storage
provisioner: rancher.io/local-path
parameters:
  nodePath: /var/local-path-provisioner
volumeBindingMode: WaitForFirstConsumer
reclaimPolicy: Delete
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: stackgres-storage-pv
  labels:
    type: local
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 3Gi
  storageClassName: "stackgres-storage"
  hostPath:
    path: /var/local-path-provisioner
```

Listing 27: 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 28: yugabyteDB - StorageClass / PersistentVolume aktivieren

VII sks9016 - yugabyteDB

VII.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 29: 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 30: 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 31: sks9016 - Check yugabyteDB On-Premise

Damit ist der Benchmarking-Server ready.

VIII Patroni

VIII.I Prerequisites

Zuerst muss der Proxy gesetzt werden:

```
# sks1232 / sks1233 / sks1234
# Proxy setzen
# nano /etc/profile.d/proxy.sh
export https_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
export http_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
# source /etc/profile.d/proxy.sh
```

Listing 32: 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 33: 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 34: Patroni - PostgreSQL einbinden

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

VIII.II Installation

VIII.III Konfiguration

IX zotero.py

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

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

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

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

    return zotero_bibtex_config
def downlaod_zotero_datas(URL, API_KEY):
    zotero_result = list
    response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
    response = response.json()
```

```
    zotero_raw = json.dumps(response, ensure_ascii=False)  # json.loads(response)
    zotero_result = json.loads(zotero_raw)
    return zotero_result

#   Get the bibtex Datas from Zotero
def get_data(zotero_bibtex_config):
    result_limit = int(zotero_bibtex_config.get('result_limit'))
    access_type = zotero_bibtex_config.get('access_type')
    zotero_access_id = zotero_bibtex_config.get('zotero_access_id')
    collection_id = zotero_bibtex_config.get('collection_id')
    API_KEY = zotero_bibtex_config.get('api_key')
    zotero_data = list()
    URL = 'https://api.zotero.org/' + str(access_type) + '/' + str(zotero_access_id) + '/collections/' + str(
        collection_id) + '/items?limit=1?format=json?sort=dateAdded?direction=asc'

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

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

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

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

            zotero_data.extend(zotero_result)

            start_index += result_limit
            index += 1

    return zotero_data
```

```python

#   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
```

```
162         else:
163             if zotero_item_type == 'statute':
164                 biblio_item['title'] = zotero_item_nameofact
165                 bibtex_item_titel = zotero_item_nameofact
166             elif zotero_item_type == 'case':
167                 biblio_item['title'] = zotero_item_nameofcase
168                 bibtex_item_titel = zotero_item_nameofcase
169             elif zotero_item_type == 'email':
170                 biblio_item['title'] = zotero_item_subject
171                 bibtex_item_titel = zotero_item_subject
172 
173         if zotero_item_type == 'thesis':
174             master_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('
    masterthesis')
175             phd_list = zotero_bibtex_keys_specials.get(zotero_item_type).get('
    phdthesis')
176 
177             # First Master thesis
178             if any(item in bibtex_item_titel for item in master_list):
179                 bibtex_item_key = 'masterthesis'
180             # Second PHD Thesis
181             elif any(item in bibtex_item_titel for item in phd_list):
182                 bibtex_item_key = 'phdthesis'
183             else:
184                 bibtex_item_key = 'masterthesis'
185         else:
186             if zotero_bibtex_keys.get(zotero_item_type).get('key'):
187                 bibtex_item_key = zotero_bibtex_keys.get(zotero_item_type).get('
    key')
188             else:
189                 bibtex_item_key = 'misc'
190 
191         # get all Keys for the zotero item type
192         entryset = '\n'
193         entry = ''
194 
195         zotero_item_attributes = zotero_bibtex_keys.get(zotero_item_type).get('
    attributes').keys()
196         item_attributes = sorted(zotero_item_attributes, reverse=True)
197 
198         for index, item_attribute in enumerate(item_attributes):
199             bibtex_item_attribute = zotero_bibtex_keys.get(zotero_item_type).get('
    attributes').get(item_attribute)
200             zotero_item_value = biblio_item.get(item_attribute)
201             zotero_item_value_extra = ''
202             bibtex_item_attribute_extra = ''
203 
204             # Special Cases
```

```
205             if bibtex_item_attribute == 'SPECIALCHECK' and zotero_item_value not
    in ['', None]:
206                 bibtex_special_attribute = zotero_bibtex_attributes_special.get(
    item_attribute)
207
208                 match bibtex_special_attribute:
209                     case 'get_dates':
210                         zotero_item_value = get_dates(zotero_item_value,
    bibtex_item_key, bibtex_month_attributes)
211                         if zotero_item_value.get('month'):
212                             zotero_item_value_extra = zotero_item_value.get('month
    ')
213                             bibtex_item_attribute_extra = 'month'
214
215                         zotero_item_value = zotero_item_value.get('year')
216                         bibtex_item_attribute = 'year'
217                     case 'split_creators':
218                         authors = split_creators(zotero_item_value)
219                         entryset = entryset + authors
220             elif bibtex_item_attribute == 'howpublished':
221                 if zotero_item_value not in ['', None, []]:
222                     zotero_item_value = '\\url{' + zotero_item_value + '}'
223
224             if bibtex_item_attribute not in ['', 'None', 'author', 'SPECIALCHECK']
     and zotero_item_value not in ['', None, []]:
225                 if zotero_item_value_extra:
226
227                     if type(zotero_item_value_extra) == "string":
228                         entryset = entryset + str(bibtex_item_attribute_extra) + '
    =\"' + str(zotero_item_value_extra) + '\"'
229                     else:
230                         entryset = entryset + str(bibtex_item_attribute_extra) + '
    =' + str(zotero_item_value_extra)
231
232                     if index != len(item_attributes) - 1:
233                         entryset = entryset + ',\n'
234                     else:
235                         entryset = entryset + '\n'
236
237                 if type(zotero_item_value) == str and not zotero_item_value.
    isnumeric():
238                     entryset = entryset + str(bibtex_item_attribute) + '=\"' + str
    (zotero_item_value) + '\"'
239                 else:
240                     entryset = entryset + str(bibtex_item_attribute) + '=' + str(
    zotero_item_value)
241
242                 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 35: Python LaTex - zotero.py - Zotero BibLaTex Importer

X 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 36: Python LaTex - zotero_bibtex_configuration.yaml - Konfigurationsdatei - Zotero BibLaTex Importer

XI 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
    publisher: publisher
    place: address
bookSection:
  key: inbook
  attributes:
    title: title
    date: SPECIALCHECK
    creators: SPECIALCHECK
    pages: pages
    publisher: publisher
    place: address
    bookTitle: booktitle
case:
```

```
51   key: misc
52   attributes:
53     title: title
54     date: SPECIALCHECK
55     creators: SPECIALCHECK
56     url: howpublished
57     extra: note
58 conferencePaper:
59   key: inproceedings
60   attributes:
61     title: title
62     date: SPECIALCHECK
63     creators: SPECIALCHECK
64     series: series
65     proceedingsTitle: booktitle
66     publisher: publisher
67     pages: pages
68     place: address
69 dictionaryEntry:
70   key: misc
71   attributes:
72     title: title
73     date: SPECIALCHECK
74     creators: SPECIALCHECK
75     url: howpublished
76     extra: note
77 document:
78   key: misc
79   attributes:
80     title: title
81     date: SPECIALCHECK
82     creators: SPECIALCHECK
83     url: howpublished
84     extra: note
85 email:
86   key: misc
87   attributes:
88     title: title
89     date: SPECIALCHECK
90     creators: SPECIALCHECK
91     url: howpublished
92     extra: note
93 encyclopediaArticle:
94   key: misc
95   attributes:
96     title: title
97     date: SPECIALCHECK
98     creators: SPECIALCHECK
```

```
    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
    creators: SPECIALCHECK
```

```
147     volume: volume
148     pages: pages
149     seriesNumber: number
150     seriesTitle: journal
151     url: url
152 letter:
153   key: misc
154   attributes:
155     title: title
156     date: SPECIALCHECK
157     creators: SPECIALCHECK
158     url: howpublished
159     extra: note
160 magazineArticle:
161   key: article
162   attributes:
163     title: title
164     date: SPECIALCHECK
165     creators: SPECIALCHECK
166     volume: volume
167     pages: pages
168     seriesNumber: number
169     seriesTitle: journal
170     url: url
171 manuscript:
172   key: unpublished
173   attributes:
174     title: title
175     date: SPECIALCHECK
176     creators: SPECIALCHECK
177 map:
178   key: misc
179   attributes:
180     title: title
181     date: SPECIALCHECK
182     creators: SPECIALCHECK
183     url: howpublished
184     extra: note
185 newspaperArticle:
186   key: article
187   attributes:
188     title: title
189     date: SPECIALCHECK
190     creators: SPECIALCHECK
191     volume: volume
192     pages: pages
193     seriesNumber: number
194     seriesTitle: journal
```

```
195     url: url
196 patent:
197   key: misc
198   attributes:
199     title: title
200     date: SPECIALCHECK
201     creators: SPECIALCHECK
202     url: howpublished
203     extra: note
204 podcast:
205   key: misc
206   attributes:
207     title: title
208     date: SPECIALCHECK
209     creators: SPECIALCHECK
210     url: howpublished
211     extra: note
212 presentation:
213   key: misc
214   attributes:
215     title: title
216     date: SPECIALCHECK
217     creators: SPECIALCHECK
218     url: howpublished
219     extra: note
220 radioBroadcast:
221   key: misc
222   attributes:
223     title: title
224     date: SPECIALCHECK
225     creators: SPECIALCHECK
226     url: howpublished
227     extra: note
228 report:
229   techreport: misc
230   attributes:
231     title: title
232     date: SPECIALCHECK
233     creators: SPECIALCHECK
234     url: howpublished
235     extra: note
236 software:
237   key: misc
238   attributes:
239     title: title
240     date: SPECIALCHECK
241     creators: SPECIALCHECK
242     url: howpublished
```

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

```yaml
    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
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 37: Python LaTex - zotero_biblatex_keystore.yaml - x-y-Achse Konfigurationsdatei - Zotero BibLaTex Importer

XII 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)

```

```
88          fig = plt.figure()
89          plt.subplots_adjust(wspace=0, hspace=0)
90          plt.xticks([])
91          plt.yticks([])
92          plt.xlim(0, 5)
93          plt.ylim(0, 5)
94          plt.xlabel(x_axis_title)
95          plt.ylabel(y_axis_title)
96          plt.title(title)
97
98          # This example is for a 5 * 5 matrix
99          nrows = 5
100         ncols = 5
101         axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0,
    nrows) for c in range(0, ncols)]
102
103         # remove the x and y ticks
104         for ax in axes:
105             ax.set_xticks([])
106             ax.set_yticks([])
107             ax.set_xlim(0, 5)
108             ax.set_ylim(0, 5)
109
110         # Add background colors
111         # This has been done manually for more fine-grained control
112         # Run the loop below to identify the indice of the axes
113         for _ in green:
114             axes[_].set_facecolor('green')
115
116         for _ in yellow:
117             axes[_].set_facecolor('yellow')
118
119         for _ in orange:
120             axes[_].set_facecolor('orange')
121
122         for _ in red:
123             axes[_].set_facecolor('red')
124
125         # run throuh datas and generate axis datas
126         dict_bubble_axis = dict()
127         bubble_axis = list()
128         for datasets in datas:
129             # get the datas
130             riskid = datas.get(datasets).get('risk-id')
131             x_axis = int(datas.get(datasets).get('x-axis'))
132             y_axis = int(datas.get(datasets).get('y-axis'))
133             axis_point = matrix.get((x_axis, y_axis))
134             x_axis_text = float(datas.get(datasets).get('x-axis-text'))
```

```
            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 38: Python LaTex - riskmatrix.py - Risxikomatrizen

XIII riskmatrix_plotter_conf.yaml

```yaml
risk_inventory:
  - "postgresql"
  - "project"
  - "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
```

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

```yaml
  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
        - 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"
```

```yaml
    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
        - 2
        - 7
        - 12
        - 13
        - 18
        - 19
        - 24
      red-boxes:
        - 3
        - 4
        - 8
        - 9
        - 14
```

Listing 39: Python LaTex - riskmatrix_plotter_conf.yaml - Konfigurationsdatei - Risxikomatrizen

XIV 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 40: Python LaTex - riskmatrix_xy_axis_tuple_matrix.yaml - Konfigurationsdatei - Risikomatrizen - X-Y-Achsen Tuples

XV 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
```

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

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

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

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

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

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

    return cost_benefit_data

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

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

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

```python

    # 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 41: Python LaTex - cost_benefit_diagram.py - Kosten-Nutzen-Diagramm

XVI cost_benefit_diagram_plotter_conf.yaml

```yaml
startpath: "parentdir"
desitination_path: "source/cost_benefit_diagram"
datafile: "cost_benefit_diagram.csv"
```

```yaml
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 42: Python LaTex - cost_benefit_diagram_plotter_conf.yaml - Konfigurationsdatei - Kosten-Nutzen-Diagramm

XVII 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')
```

```
101         datafile = panda_latex_tables_config.get('tables').get(table_item).get('
    datafile')
102         if startpath == 'homedir':
103             directory = os.path.join(os.getcwd(), destination)
104         else:  # parentdir
105             directory = os.path.join(os.path.dirname(os.getcwd()), destination)
106         tablefile = os.path.join(directory, tablefilename)
107         separator = panda_latex_tables_config.get('tables').get(table_item).get('
    separator')
108         decimal = panda_latex_tables_config.get('tables').get(table_item).get('
    decimal')
109 
110         # column operations
111         column_operations = panda_latex_tables_config.get('tables').get(table_item
    ).get('column_operations').get('datas')
112 
113         # group by / aggregation
114         groupby_values = panda_latex_tables_config.get('tables').get(table_item).
    get('group_by')
115         group_by_function = panda_latex_tables_config.get('tables').get(table_item
    ).get('group_by_function')
116         # selected_rows = panda_latex_tables_config.get('tables').get(table_item).
    get('selected_rows')
117         agg_funtion = panda_latex_tables_config.get('tables').get(table_item).get(
    'agg_funtion')
118         agg_colums = panda_latex_tables_config.get('tables').get(table_item).get('
    agg_colums')
119         # dropping and renaming columns
120         drop_columns = panda_latex_tables_config.get('tables').get(table_item).get
    ('drop_columns')
121         rename_columns = panda_latex_tables_config.get('tables').get(table_item).
    get('rename_columns')
122 
123         # table filtering and sorting
124         where_clausel = panda_latex_tables_config.get('tables').get(table_item).
    get('where_clausel')
125         order_by = panda_latex_tables_config.get('tables').get(table_item).get('
    sorting').get('order_by')
126         sort_acending = panda_latex_tables_config.get('tables').get(table_item).
    get('sorting').get('sort_acending')
127         sort_inplace = panda_latex_tables_config.get('tables').get(table_item).get
    ('sorting').get('sort_inplace')
128 
129         # pivot settings
130         pivot = panda_latex_tables_config.get('tables').get(table_item).get('pivot
    ')
131         pivot_column = panda_latex_tables_config.get('tables').get(table_item).get
    ('pivot_columns')
```

```
132         pivot_value = panda_latex_tables_config.get('tables').get(table_item).get(
    'pivot_values')
133
134         # pivot_table settings
135         pivot_table = panda_latex_tables_config.get('tables').get(table_item).get(
    'pivot_table')
136         pivot_table_column = panda_latex_tables_config.get('tables').get(
    table_item).get('pivot_table').get(
137             'pivot_columns')
138         pivot_table_value = panda_latex_tables_config.get('tables').get(table_item
    ).get('pivot_table').get(
139             'pivot_values')
140         pivot_table_agg_function = panda_latex_tables_config.get('tables').get(
    table_item).get('pivot_table').get(
141             'pivot_agg_func')
142         pivot_table_indizes = panda_latex_tables_config.get('tables').get(
    table_item).get('pivot_table').get(
143             'pivot_index').get('pivot_indizes')
144         pivot_table_indizes_visible = panda_latex_tables_config.get('tables').get(
    table_item).get('pivot_table').get(
145             'pivot_index').get('pivot_indizes_visible')
146         pivot_table_rename_indizes = panda_latex_tables_config.get('tables').get(
    table_item).get('pivot_table').get(
147             'pivot_index').get('pivot_rename_indizes')
148
149         # margins (subtotals)
150         margin = panda_latex_tables_config.get('tables').get(table_item).get('
    margins').get('margin')
151         margin_name = panda_latex_tables_config.get('tables').get(table_item).get(
    'margins').get('margin_name')
152
153         # table settings
154         table_caption = panda_latex_tables_config.get('tables').get(table_item).
    get('caption')
155         table_label = panda_latex_tables_config.get('tables').get(table_item).get(
    'label')
156         table_style = panda_latex_tables_config.get('tables').get(table_item).get(
    'table_styles')
157         sparse_columns = panda_latex_tables_config.get('tables').get(table_item).
    get('table_styles').get(
158             'sparse_columns')
159         table_caption_position = panda_latex_tables_config.get('tables').get(
    table_item).get('table_styles').get(
160             'props').get('caption-side')
161         table_position = panda_latex_tables_config.get('tables').get(table_item).
    get('table_styles').get('props').get(
162             'position')
163         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)

    # 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 43: Python LaTex - pandas_dataframe_to_latex_table.py CSV - LaTex Tabelle

XVIII csv_to_latex_diplomarbeit.yaml

```
tables_inventory:
 - "db_inventory"
 - "db_inventory_per_rdbms"
 - "db_inventory_per_os"
 - "anforderungskatalog"
 - "arbeitsrapport"
 - "projektcontrolling"
 - "evaluation_inventory"
 - "dependencis"
 - "predecision_out"
 - "predecision_in"
 - "project_comments"
```

```yaml
 - "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"
    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:
```

```
109  agg_colums:
110   - "rdbms"
111  drop_columns:
112   - "server"
113   - "os"
114   - "version"
115   - "releasedate"
116   - "eol"
117   - "age"
118   - "eol_since"
119   - "comment"
120  column_operations:
121   datas:
122   operations:
123    dauer_summe:
124     operation_function:
125     axis_number:
126     columns:
127  pivot:
128   pivot_columns:
129   pivot_values:
130  pivot_table:
131   pivot_index:
132    pivot_indizes_visible:
133    pivot_rename_indizes:
134   pivot_columns:
135   pivot_values:
136   pivot_agg_func:
137  rename_columns:
138   rdbms: "RDBMS"
139   instance : "Instanz"
140   databases : "Datenbanken"
141   appliance: "Appliance"
142  where_clausel:
143  sorting:
144   order_by:
145    - "rdbms"
146   sort_acending: True
147   sort_inplace: True
148  margins:
149   margin: True
150   margin_name: "Gesamtergebnis"
151  table_styles:
152   selector: "caption"
153   props:
154    caption-side: "below"
155    position: "H"
156    sparse_columns: True
```

```
157     longtable: False
158     resize_textwidth: False
159     linebreak_columns:
160     table_header: True
161  db_inventory_per_os:
162   id: "db_inventory_per_os"
163   isbigfile:
164   has_longtexts: False
165   separator: ","
166   decimal: "."
167   caption: "Datenbankinventor - Nach Betriebssystemen üaufgeschlsselt"
168   label: "db_inventory_per_os"
169   startpath: "parentdir"
170   destination_path: "content/latex_tables"
171   datafile_path: "source/tables"
172   datafile: "inventory.csv"
173   tablefilename: "db_inventory_per_os.tex"
174   decimal_format:
175   group_by:
176    - "rdbms"
177    - "os"
178   group_by_function: "sum"
179   agg_funtion:
180   agg_colums:
181    - "os"
182   drop_columns:
183    - "server"
184    - "version"
185    - "releasedate"
186    - "eol"
187    - "age"
188    - "eol_since"
189    - "comment"
190 #    - "appliance"
191   column_operations:
192    datas:
193    operations:
194     dauer_summe:
195      operation_function:
196      axis_number:
197      columns:
198   pivot:
199    pivot_columns:
200    pivot_values:
201   pivot_table:
202    pivot_index:
203     pivot_indizes:
204      - "os"
```

```
205      - "rdbms"
206     pivot_indizes_visible: True
207     pivot_rename_indizes:
208      os: "OS"
209      rdbms: "RDBMS"
210    pivot_columns:
211    pivot_values:
212    pivot_agg_func:
213     instance: "sum"
214     databases: "sum"
215     appliance: "sum"
216    transpose: True
217   rename_columns:
218    rdbms: "RDBMS"
219    instance : "Instanz"
220    databases : "Datenbanken"
221    os : "OS"
222    appliance: "Appliance"
223   where_clausel:
224   sorting:
225    order_by:
226    sort_acending: False
227    sort_inplace: True
228   margins:
229    margin: True
230    margin_name: "Gesamtergebnis"
231   table_styles:
232    selector: "caption"
233    props:
234     caption-side: "below"
235     position: "H"
236     sparse_columns: True
237     longtable: True
238     resize_textwidth: False
239     linebreak_columns:
240     table_header: True
241 anforderungskatalog:
242  id: "anforderungskatalog"
243  isbigfile:
244  has_longtexts: True
245  separator: ";"
246  decimal: "."
247  caption: "Anforderungskatalog"
248  label: "anforderungskatalog"
249  startpath: "parentdir"
250  destination_path: "content/latex_tables"
251  datafile_path: "source/tables"
252  datafile: "anforderungskatalog.CSV"
```

```
253   tablefilename: "anforderungskatalog.tex"
254   decimal_format:
255   group_by:
256   group_by_function:
257   agg_funtion:
258   agg_colums:
259   drop_columns:
260   column_operations:
261    datas:
262    operations:
263     dauer_summe:
264      operation_function:
265      axis_number:
266      columns:
267   pivot:
268    pivot_columns:
269    pivot_values:
270   pivot_table:
271    pivot_index:
272     pivot_indizes_visible: False
273     pivot_rename_indizes:
274    pivot_columns:
275    pivot_values:
276    pivot_agg_func:
277   rename_columns:
278   where_clausel:
279   sorting:
280    order_by:
281     - "Nr."
282    sort_acending: True
283    sort_inplace: True
284   margins:
285    margin: False
286    margin_name:
287   table_styles:
288    selector: "caption"
289    props:
290     caption-side: "below"
291     position: "H"
292     sparse_columns: False
293     longtable: False
294     resize_textwidth: True
295     linebreak_columns:
296      - "Beschreibung"
297     table_header: True
298 arbeitsrapport:
299   id: "arbeitsrapport"
300   isbigfile:
```

```yaml
301  has_longtexts: False
302  separator: ";"
303  decimal: "."
304  caption: "Arbeitsrapport"
305  label: "arbeitsrapport"
306  startpath: "parentdir"
307  destination_path: "content/latex_tables"
308  datafile_path: "source/tables"
309  datafile: "arbeitsrapport.CSV"
310  tablefilename: "arbeitsrapport.tex"
311  decimal_format: "{:0.1f}"
312  group_by:
313  group_by_function:
314  agg_funtion:
315  agg_colums:
316  drop_columns:
317   - "Hide"
318   - "Geplante Dauer [h]"
319   - "dauer_summe"
320  column_operations:
321   datas:
322   operations:
323    dauer_summe:
324     operation_function:
325     axis_number:
326     columns:
327  pivot:
328   pivot_columns:
329   pivot_values:
330  pivot_table:
331   pivot_index:
332    pivot_indizes_visible: False
333    pivot_rename_indizes:
334   pivot_columns:
335   pivot_values:
336   pivot_agg_func:
337  rename_columns:
338  where_clausel: "Hide == 0"
339  sorting:
340   order_by:
341    - "Datum"
342    - "Von"
343   sort_acending: False
344   sort_inplace: False
345  margins:
346   margin: False
347   margin_name:
348  table_styles:
```

```
349    selector: "caption"
350    props:
351      caption-side: "below"
352      position: "H"
353      sparse_columns: True
354      longtable: False
355      resize_textwidth: True
356      linebreak_columns:
357        - "äTtigkeit"
358        - "Bemerkung"
359        - "Schwierigkeit"
360        - "öLsungen"
361      table_header: True
362  projektcontrolling:
363   id: "projektcontrolling"
364   isbigfile:
365   has_longtexts: False
366   separator: ";"
367   decimal: "."
368   caption: "Projektcontrolling"
369   label: "projektcontrolling"
370   startpath: "parentdir"
371   destination_path: "content/latex_tables"
372   datafile_path: "source/tables"
373   datafile: "arbeitsrapport.CSV"
374   tablefilename: "projektcontrolling.tex"
375   decimal_format: "{:0.1f}"
376   group_by:
377    - "Phase"
378    - "Subphase"
379   group_by_function: "sum"
380   agg_funtion:
381   agg_colums:
382      - "Dauer [h]"
383      - "Geplante Dauer [h]"
384      - "dauer_summe"
385   drop_columns:
386    - "Datum"
387    - "Von"
388    - "Bis"
389    - "Hide"
390    - "äTtigkeit"
391    - "Bemerkung"
392    - "Schwierigkeit"
393    - "öLsungen"
394   column_operations:
395    datas:
396     - "dauer_summe"
```

```
397     operations:
398      dauer_summe:
399       operation_function: "diff"
400       axis_number: 1
401       columns:
402       - "Dauer [h]"
403       - "Geplante Dauer [h]"
404    pivot:
405     pivot_columns:
406     pivot_values:
407    pivot_table:
408     pivot_index:
409      pivot_indizes_visible:
410      pivot_rename_indizes:
411     pivot_columns:
412     pivot_values:
413     pivot_agg_func:
414    rename_columns:
415     dauer_summe: "Verbleibende Zeit [h]"
416    where_clausel:
417    sorting:
418     order_by:
419      - "Phase"
420      - "Subphase"
421     sort_acending: True
422     sort_inplace: True
423    margins:
424     margin: True
425     margin_name: "Total"
426    table_styles:
427     selector: "caption"
428     props:
429      caption-side: "below"
430      position: "H"
431      sparse_columns: True
432      longtable: False
433      resize_textwidth: True
434      linebreak_columns:
435      table_header: True
436  evaluation_inventory:
437    id: "evaluation_inventory"
438    isbigfile:
439    has_longtexts: False
440    separator: ";"
441    decimal: "."
442    caption: "Evaluationssysteme"
443    label: "evaluation_inventory"
444    startpath: "parentdir"
```

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

```yaml
dependencis:
 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"
```

```
541   props:
542    caption-side: "below"
543    position: "H"
544    sparse_columns: True
545    longtable: False
546    resize_textwidth: True
547    linebreak_columns:
548     - "äAbhngigkeit"
549     - "Beschreibung"
550     - "Status"
551     - "Risiko"
552     - "Impact"
553    table_header: True
554 predecision_out:
555  id: "predecision_out"
556  isbigfile:
557  has_longtexts: False
558  separator: ";"
559  decimal: "."
560  caption: "Vorauswahl - Ausgeschieden"
561  label: "predecision_out"
562  startpath: "parentdir"
563  destination_path: "content/latex_tables"
564  datafile_path: "source/tables"
565  datafile: "pre-decision.csv"
566  tablefilename: "pre-decision-out.tex"
567  decimal_format:
568  group_by:
569  group_by_function:
570  agg_funtion:
571  agg_colums:
572  drop_columns:
573   - "hide_state"
574  column_operations:
575   datas:
576   operations:
577    dauer_summe:
578     operation_function:
579     axis_number:
580     columns:
581  pivot:
582   pivot_columns:
583   pivot_values:
584  pivot_table:
585   pivot_index:
586    pivot_indizes_visible: False
587    pivot_rename_indizes:
588   pivot_columns:
```

```
589     pivot_values:
590     pivot_agg_func:
591   rename_columns:
592   where_clausel: "hide_state == 1"
593   sorting:
594     order_by:
595       - "Nr."
596     sort_acending: True
597     sort_inplace: True
598   margins:
599     margin: False
600     margin_name:
601   table_styles:
602     selector: "caption"
603     props:
604       caption-side: "below"
605       position: "H"
606       sparse_columns: True
607       longtable: False
608       resize_textwidth: True
609       linebreak_columns:
610         - "üBegrndung"
611       table_header: True
612 predecision_in:
613   id: "predecision_in"
614   isbigfile:
615   has_longtexts: False
616   separator: ";"
617   decimal: "."
618   caption: "Vorauswahl - Evaluation"
619   label: "predecision_in"
620   startpath: "parentdir"
621   destination_path: "content/latex_tables"
622   datafile_path: "source/tables"
623   datafile: "pre-decision.csv"
624   tablefilename: "pre-decision-in.tex"
625   decimal_format:
626   group_by:
627   group_by_function:
628   agg_funtion:
629   agg_colums:
630   drop_columns:
631     - "hide_state"
632   column_operations:
633     datas:
634     operations:
635       dauer_summe:
636         operation_function:
```

```
      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
  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:
```

```
685   group_by_function:
686   agg_funtion:
687   agg_colums:
688   drop_columns:
689   column_operations:
690    datas:
691    operations:
692     dauer_summe:
693      operation_function:
694      axis_number:
695      columns:
696   pivot:
697    pivot_columns:
698    pivot_values:
699   pivot_table:
700    pivot_index:
701     pivot_indizes_visible: False
702     pivot_rename_indizes:
703    pivot_columns:
704    pivot_values:
705    pivot_agg_func:
706   rename_columns:
707   where_clausel:
708   sorting:
709    order_by:
710     - "Woche"
711    sort_acending: True
712    sort_inplace: True
713   margins:
714    margin: False
715    margin_name:
716   table_styles:
717    selector: "caption"
718    props:
719     caption-side: "below"
720     position: "H"
721     sparse_columns: True
722     longtable: False
723     resize_textwidth: True
724     linebreak_columns:
725      - "Beschreibung / Event / Problem"
726     table_header: True
727 evaluation_distributed_sql:
728   id: "evaluation_distributed_sql"
729   isbigfile:
730   has_longtexts: False
731   separator: ";"
732   decimal: "."
```

```
733  caption: "Evaluationssysstem - Distributed SQL / Sharding"
734  label: "evaluation_distributed_sql"
735  startpath: "parentdir"
736  destination_path: "content/latex_tables"
737  datafile_path: "source/tables"
738  datafile: "evaluation_platform_distributed_sql.csv"
739  tablefilename: "evaluation_platform_distributed_sql.tex"
740  decimal_format:
741  group_by:
742  group_by_function:
743  agg_funtion:
744  agg_colums:
745  drop_columns:
746  column_operations:
747   datas:
748   operations:
749    dauer_summe:
750     operation_function:
751     axis_number:
752     columns:
753  pivot:
754   pivot_columns:
755   pivot_values:
756  pivot_table:
757   pivot_index:
758    pivot_indizes_visible: False
759    pivot_rename_indizes:
760   pivot_columns:
761   pivot_values:
762   pivot_agg_func:
763  rename_columns:
764  where_clausel:
765  sorting:
766   order_by:
767   sort_acending: False
768   sort_inplace: False
769  margins:
770   margin: False
771   margin_name:
772  table_styles:
773   selector: "caption"
774   props:
775    caption-side: "below"
776    position: "H"
777    sparse_columns: True
778    longtable: False
779    resize_textwidth: False
780    linebreak_columns:
```

```
    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:
   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:
  pivot_table:
    pivot_index:
      pivot_indizes_visible: False
      pivot_rename_indizes:
    pivot_columns:
```

```
877     pivot_values:
878     pivot_agg_func:
879   rename_columns:
880   where_clausel:
881   sorting:
882     order_by:
883       - "äFachgesprch"
884     sort_acending: True
885     sort_inplace: True
886   margins:
887     margin: False
888     margin_name:
889   table_styles:
890     selector: "caption"
891     props:
892       caption-side: "below"
893       position: "H"
894       sparse_columns: True
895       longtable: False
896       resize_textwidth: True
897       linebreak_columns:
898         - "Studenten"
899         - "Fragen"
900         - "Antworten"
901         - "Sonstige Themen"
902         - "Bemerkungen"
903       table_header: True
904 stakeholder:
905   id: "stakeholder"
906   isbigfile:
907   has_longtexts: False
908   separator: ";"
909   decimal: "."
910   caption: "Stakeholder"
911   label: "stakeholder"
912   startpath: "parentdir"
913   destination_path: "content/latex_tables"
914   datafile_path: "source/tables"
915   datafile: "stakeholder.csv"
916   tablefilename: "stakeholder.tex"
917   decimal_format:
918   group_by:
919   group_by_function:
920   agg_funtion:
921   agg_colums:
922   drop_columns:
923   column_operations:
924     datas:
```

```
    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 44: Python LaTex - csv_to_latex_diplomarbeit.yaml - Konfigurationsdatei - CSV - LaTex-Tabelle

XIX pandas_data_chart_plotter.py

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


```

```
8  def load_configuration(panda_diagram_plotter_conf_filename):
9      panda_diagram_plotter_config = dict()
10
11     riskmatrix_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
       configuration')
12     yaml_path = os.path.join(riskmatrix_conf_dir,
       panda_diagram_plotter_conf_filename)
13
14     with open(yaml_path, "r") as file:
15         panda_diagram_plotter_config = yaml.load(file, Loader=yaml.FullLoader)
16
17     return panda_diagram_plotter_config
18
19 def get_data(startpath, destination, tablefilename, datafile_path, datafile,
       separator, decimal):
20     # Config Variables
21     if startpath == 'homedir':
22         directory = os.path.join(os.getcwd(), datafile_path)
23     else:  # parentdir
24         directory = os.path.join(os.path.dirname(os.getcwd()), datafile_path)
25
26     # get the Datas as dirct
27     data_path = os.path.join(directory, datafile)
28
29     # load datas from csv into dict
30     detected = chardet.detect(Path(data_path).read_bytes())
31     encoding = detected.get("encoding")
32
33     print(datafile, ':', encoding)
34     panda_table_data = pd.read_csv(data_path, sep=separator, decimal=decimal,
       encoding=encoding)
35     # return data
36     return panda_table_data
37 def create_panda_diagram_plotter(panda_diagram_plotter_config):
38     pdp_tables = panda_diagram_plotter_config.get('diagram_inventory')
39     for table_item in pdp_tables:
40         print(table_item)
41         startpath = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
       table_item).get('startpath')
42         destination = panda_diagram_plotter_config.get('panda_diagram_plotter').
       get(table_item).get('destination_path')
43         imagename = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
       table_item).get('imagename')
44         datafile_path = panda_diagram_plotter_config.get('panda_diagram_plotter').
       get(table_item).get('datafile_path')
45         datafile = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
       table_item).get('datafile')
46         if startpath == 'homedir':
```

```
47                directory = os.path.join(os.getcwd(), destination)
48            else:  # parentdir
49                directory = os.path.join(os.path.dirname(os.getcwd()), destination)
50            image_path = os.path.join(directory, imagename)
51            separator = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('separator')
52            decimal = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('decimal')
53
54            # column operations
55            column_operations = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('datas')
56
57            # group by / aggregation
58            groupby_values = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('group_by')
59            group_by_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('group_by_function')
60
61            agg_funtion = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('agg_funtion')
62            agg_colums = panda_diagram_plotter_config.get('panda_diagram_plotter').get
    (table_item).get('agg_colums')
63            # dropping and renaming columns
64            drop_columns = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('drop_columns')
65            rename_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('rename_columns')
66
67            # table filtering and sorting
68            where_clausel = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('where_clausel')
69            order_by = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('sorting').get('order_by')
70            sort_acending = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('sorting').get('sort_acending')
71            sort_inplace = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('sorting').get('sort_inplace')
72
73            # pivot settings
74            pivot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('pivot')
75            pivot_column = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_columns')
76            pivot_value = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_values')
77
78            # pivot_table settings
```

```
79          pivot_table = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('pivot_table')
80          pivot_table_column = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
81              'pivot_columns')
82          pivot_table_value = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
83              'pivot_values')
84          pivot_table_agg_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
85              'pivot_agg_func')
86          pivot_table_indizes = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
87              'pivot_index').get('pivot_indizes')
88          pivot_table_indizes_visible = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
89              'pivot_index').get('pivot_indizes_visible')
90          pivot_table_rename_indizes = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('pivot_table').get(
91              'pivot_index').get('pivot_rename_indizes')
92
93          # margins (subtotals)
94          margin = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('margins').get('margin')
95          margin_name = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('margins').get('margin_name')
96
97          # chart settings
98          chart = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-kind')
99          title = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('title')
100         x_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('x-axis-title')
101         y_axis_title = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('y-axis-title')
102         x_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('x-axis-columns')
103         y_axis_columns = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('y-axis-columns')
104         index = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-index')
105
106         # chart styles
107         grid = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('grid')
108         legend = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('legend')
```

```
109         rot = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('rot')
110         fontsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('fontsize')
111         figsize = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('figsize')
112         stacked = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('stacked')
113         secondary_y = panda_diagram_plotter_config.get('panda_diagram_plotter').
    get(table_item).get('chart-designs').get('secondary_y')
114         stylelist = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('stylelist')
115         subplots = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('subplots')
116         autopct = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('autopct')
117         loc = panda_diagram_plotter_config.get('panda_diagram_plotter').get(
    table_item).get('chart-designs').get('loc')
118         bbox_to_anchor = panda_diagram_plotter_config.get('panda_diagram_plotter')
    .get(table_item).get('chart-designs').get('bbox_to_anchor')
119
120         # get the pandas (panda data)
121         panda_table_data = get_data(startpath, destination, imagename,
    datafile_path, datafile, separator, decimal)
122
123         # filter by where clausel
124         if where_clausel:
125             panda_table_data = panda_table_data.query(where_clausel)
126
127         # Drop unused columns
128         if drop_columns:
129             panda_table_data = panda_table_data.drop(columns=drop_columns)
130
131         # set aggregation functions
132         # if groupby_values and not agg_funtion and not pivot_column and not
    pivot_table_column:
133         if groupby_values and not (pivot_column or (pivot_table_column or
    pivot_table_value or pivot_table_indizes)):
134             match group_by_function:
135                 case 'max':
136                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).max()
137                 case 'min':
138                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).min()
139                 case 'head':
140                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).head()
```

```
141                 case 'sum':
142                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).sum()
143                 case 'mean':
144                     panda_table_data = panda_table_data.groupby(groupby_values,
    as_index=False).mean()
145         else:
146             panda_table_data = panda_table_data
147
148         # pivot if pivot is selected
149         if pivot_table_column or pivot_table_value or pivot_table_indizes:
150             if type(pivot_table_agg_function) is list:
151                 agg_tuple = tuple(pivot_table_agg_function)
152                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
153                                                   columns=pivot_table_column,
    values=pivot_table_value,
154                                                   aggfunc=agg_tuple, margins=
    margin, margins_name=margin_name)
155             elif type(pivot_table_agg_function) is dict:
156                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
157                                                   columns=pivot_table_column,
158                                                   values=pivot_table_value,
    aggfunc=pivot_table_agg_function,
159                                                   margins=margin, margins_name=
    margin_name)
160             else:
161                 panda_table_data = pd.pivot_table(panda_table_data, index=
    pivot_table_indizes,
162                                                   columns=pivot_table_column,
    values=pivot_table_value,
163                                                   aggfunc=pivot_table_agg_function
    , margins=margin,
164                                                   margins_name=margin_name)
165
166         # set column operations
167         if column_operations:
168             for column_ops in column_operations:
169                 operation_function = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('operation_function')
170                 operation_columns = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('columns')
171                 operation_axis = panda_diagram_plotter_config.get('
    panda_diagram_plotter').get(table_item).get('column_operations').get('
    operations').get(column_ops).get('axis_number')
```

```
172                 match operation_function:
173                     case 'max':
174                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].max()
175                     case 'min':
176                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].min()
177                     case 'head':
178                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].head()
179                     case 'sum':
180                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].sum(axis=operation_axis)
181                     case 'mean':
182                         panda_table_data[column_ops] = panda_table_data[
    operation_columns].mean()
183                     case 'diff':
184                         panda_table_data[column_ops] = panda_table_data[
    operation_columns[1]] - panda_table_data[operation_columns[0]]
185
186
187         # order by
188         if order_by:
189             panda_table_data.sort_values(by=order_by, inplace=sort_inplace,
    ascending=sort_acending)
190
191         # rename columns
192         if rename_columns:
193             panda_table_data = panda_table_data.rename(columns=rename_columns)
194
195         # set indices
196         if index:
197             index_values = panda_table_data.get(index)
198             #panda_table_data.set_index(index_values)
199             panda_table_data = panda_table_data.set_index(index)
200
201         # rename indices
202         if pivot_table_rename_indizes:
203             panda_table_data = panda_table_data.rename_axis(index=
    pivot_table_rename_indizes)
204
205         #   Plotter
206         #   Plotter Process starts here!
207         if autopct:
208             panda_chart_plot = panda_table_data.plot(kind=chart, title=title, y=
    y_axis_columns, x=x_axis_columns, xlabel=x_axis_title,
209                                                 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 45: Python LaTex - pandas_data_chart_plotter.py CSV - Diagramm

XX 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:
```

```
17        - "2. Iteration"
18        - "3. Iteration"
19        - "4. Iteration"
20      x-axis-title: "Varianten"
21      y-axis-title: "Transaktionen pro Sekunde (tps) Bsp."
22      title: "Transaktionen pro Sekunden - mixed"
23      chart-index:
24      chart-kind: "bar"
25      chart-designs:
26        subplots: False
27        grid: True
28        legend: True
29        rot:
30        fontsize:
31        stacked: False
32        secondary_y: False
33        stylelist:
34        figsize:
35        autopct:
36        loc:
37        bbox_to_anchor:
38      group_by:
39      group_by_function:
40      agg_funtion:
41      agg_colums:
42      drop_columns:
43        - "tps_1_iteration"
44        - "tps_typ"
45      column_operations:
46        datas:
47      pivot:
48        pivot_columns:
49        pivot_values:
50      pivot_table:
51        pivot_index:
52         pivot_indizes_visible: False
53         pivot_rename_indizes:
54        pivot_columns:
55        pivot_values:
56        pivot_agg_func:
57      rename_columns:
58        variante: "Varianten"
59        tps_2_iteration: "2. Iteration"
60        tps_3_iteration: "3. Iteration"
61        tps_4_iteration: "4. Iteration"
62      where_clausel: "tps_typ == 'mixed'"
63      sorting:
64        order_by:
```

```
      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"
    group_by_function: "sum"
    agg_funtion:
    agg_colums:
      - "rdbms"
    drop_columns:
     - "server"
     - "os"
     - "version"
     - "releasedate"
     - "eol"
     - "age"
     - "eol_since"
```

```
113      - "comment"
114      - "appliance"
115    column_operations:
116      datas:
117    pivot:
118      pivot_columns:
119      pivot_values:
120    pivot_table:
121      pivot_index:
122       pivot_indizes_visible: False
123       pivot_rename_indizes:
124      pivot_columns:
125      pivot_values:
126      pivot_agg_func:
127    rename_columns:
128     rdbms: "RDBMS"
129     instance : "Instanz"
130     databases : "Datenbanken"
131     appliance: "Appliance"
132    where_clausel:
133    sorting:
134      order_by:
135        - "rdbms"
136      sort_acending: True
137      sort_inplace: True
138    margins:
139      margin: False
140      margin_name:
141  db_inventory_per_os:
142    id: "db_inventory_per_os"
143    separator: ","
144    decimal: "."
145    startpath: "parentdir"
146    destination_path: "source/pandas_data_chart_plotter"
147    datafile_path: "source/tables"
148    datafile: "inventory.csv"
149    imagename: "db_inventory_per_os.png"
150    decimal_format:
151    x-axis-columns: "RDBMS"
152    y-axis-columns:
153    x-axis-title:
154    y-axis-title:
155    title: "Datenbankinventor - Pro OS"
156    chart-index:
157    chart-kind: "pie"
158    chart-designs:
159      subplots: True
160      grid: False
```

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

```
      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 46: Python LaTex - pandas_data_chart_plotter_conf.yaml - Konfigurationsdatei - CSV - Diagramme