

# 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

<b>Abkürzungen</b>	<b>4</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Ausgangslage und Problemstellung . . . . .	1
1.1.1 Das Kantonsspital Graubünden . . . . .	1
1.1.2 Die ICT des Kantonsspital Graubünden . . . . .	3
1.1.3 Rolle in der ICT vom Kantonsspital Graubünden . . . . .	5
1.1.4 Ausgangslage . . . . .	6
1.1.5 Problemstellung . . . . .	9
1.2 Zieldefinition . . . . .	13
1.3 Abgrenzungen . . . . .	16
1.4 Abhängigkeiten . . . . .	18
1.5 Risikomanagement . . . . .	19
1.6 Vorgehensweise und Methoden . . . . .	24
1.7 Projektmanagement . . . . .	24
1.7.1 Projektcontrolling . . . . .	25
1.7.2 GANTT-Diagramm . . . . .	26
1.8 Status-Reports . . . . .	28
1.8.1 Initialer Statusbericht . . . . .	28
1.8.2 Zweiter Statusbericht . . . . .	29
1.9 Expertengespräche . . . . .	30
<b>2 Umsetzung</b>	<b>31</b>
2.1 Evaluation . . . . .	31
2.1.1 Exkurs Architektur . . . . .	31
2.1.2 Erheben und Gewichten der Anforderungen . . . . .	35
2.1.3 Testziele erarbeiten . . . . .	48
2.1.4 PostgreSQL Benchmarking . . . . .	48
2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen . . . . .	48
2.1.6 Vorauswahl . . . . .	66
2.1.7 Installation verschiedener Lösungen . . . . .	67
2.1.8 Gegenüberstellung der Lösungen . . . . .	70
2.1.9 Entscheid . . . . .	70
2.2 Aufbau und Implementation Testsystem . . . . .	70
2.2.1 Bereitstellen der Grundinfrastruktur . . . . .	70

2.2.2	Installation und Konfiguration PostgreSQL HA Cluster . . . . .	70
2.2.3	Technical Review der Umgebung . . . . .	70
2.3	Testing . . . . .	71
2.3.1	Testing . . . . .	71
2.3.2	Protokollierung . . . . .	71
2.3.3	Review und Auswertung . . . . .	71
2.4	Troubleshooting und Lösungsfindung . . . . .	71
<b>3</b>	<b>Resultate</b>	<b>72</b>
3.1	Zielüberprüfung . . . . .	72
3.2	Schlussfolgerung . . . . .	72
3.3	Weiteres Vorgehen / offene Arbeiten . . . . .	72
3.4	Persönliches Fazit . . . . .	72
	<b>Abbildungsverzeichnis</b>	<b>73</b>
	<b>Tabellenverzeichnis</b>	<b>75</b>
	<b>Listings</b>	<b>76</b>
	<b>Literatur</b>	<b>77</b>
	<b>Glossar</b>	<b>81</b>
	<b>Anhang</b>	<b>i</b>
I	Statusbericht . . . . .	i
I.I	. . . . .	i
II	Arbeitsrapport . . . . .	ii
III	Protokoll - Fachgespräche . . . . .	iii
IV	Kommentare / Anmerkungen . . . . .	iv
V	rke2 . . . . .	v
V.I	Vorbereitung . . . . .	v
V.II	Installation . . . . .	v
V.III	Cluster Konfiguration . . . . .	vi
VI	pgpool-II . . . . .	viii
VI.I	PostgreSQL Cluster Installation . . . . .	viii
VI.II	yugabyteDB . . . . .	viii
VII	Stackgres mit Citus . . . . .	viii
VIII	zotero.py . . . . .	viii
IX	riskmatrix.py . . . . .	xiv

## Abkürzungen

ICT	information and communications technology
ibW	ibW Höhere Fachschule Südostschweiz
KSGR	Kantonsspital Graubünden
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

## 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

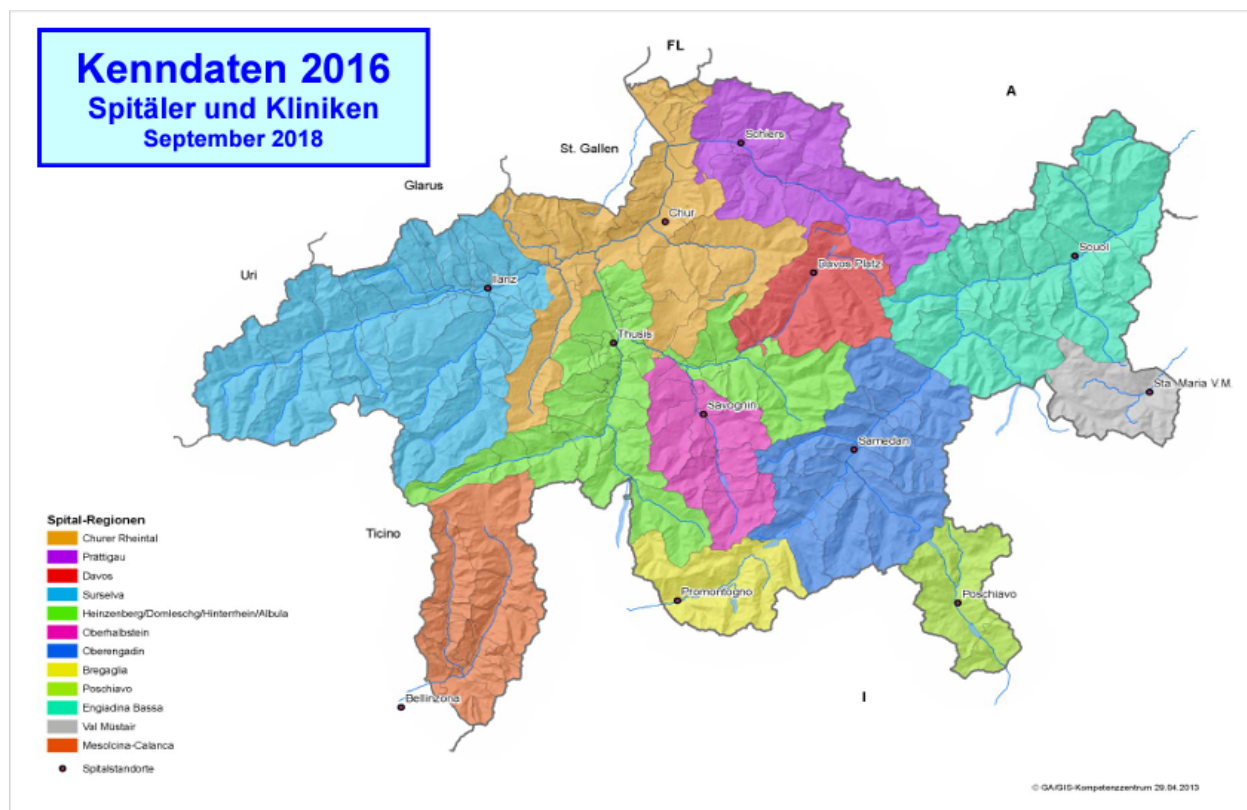


Abbildung 1.1: Spitalregionen Kanton Graubünden[30]

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[53]

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[24]

### 1.1.2 Die ICT des Kantonsspital Graubünden

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

## Organigramm des Kantonsspitals Graubünden

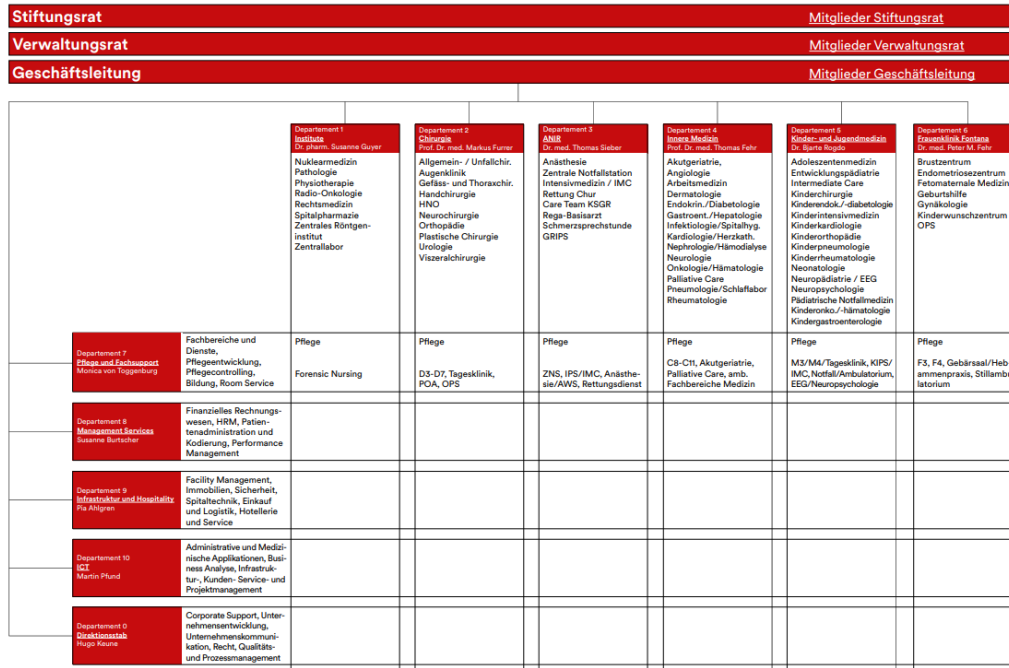
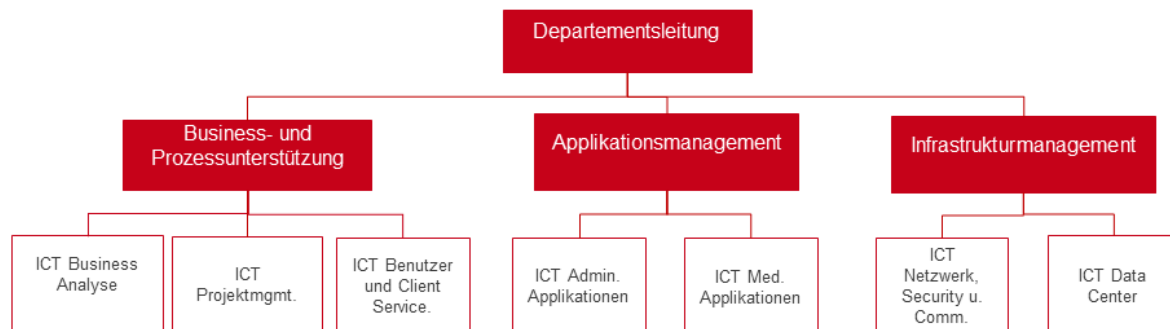


Abbildung 1.4: Organigramm Kantonsspital Graubünden

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



29.09.2023

3

Abbildung 1.5: Organigramm Departement 10 - ICT

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

### 1.1.3 Rolle in der ICT vom Kantonsspital Graubünden

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

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

### 1.1.4 Ausgangslage

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

Basierend auf der Liste *DB-Engines Ranking*[21] 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

Tabelle 1.2: Datenbankinventar

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

		Appliance	Datenbanken	Instanz
OS	RDBMS			
HP-UX	Oracle Database	0	24	21
Linux	MariaDB	0	2	2
	MySQL	3	36	14
	Oracle Database	0	1	1
	PostgreSQL	4	8	8
	Redis	0	1	1
Windows Server	MongoDB	0	2	2
	MySQL	0	14	14
	Oracle Database	0	5	5
	PostgreSQL	0	12	12
Gesamtergebnis		7	105	80

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

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

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

Aktuell werden die Daten des Zabbix der Netzwerktechniker auf eine MariaDB Datenbank gespeichert, dies soll sich aber ändern. Da das Zabbix alle Netzwerkgeräte überwacht, pro

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

### 1.1.5 Problemstellung

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

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

Dadurch ergeben sich nach BSI folgende Risiken:

Identifikation						Abschätzung		Behandlung		
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen ergreifen?	Zielwert WS SM	Massnahme
1	I	G0.22	Manipulation von Informationen	Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja	1 2	Best-Practice bei Härtung der Systeme. Redundanzen einführen
2	A	G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme 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. Aufgrund der sehr heterogenen Landschaft ist der Administrationsaufwand für die jetzigen Systeme sehr gross. Zu gross, als das für jede Datenbank und deren Betriebssystem die notwendige Zeit für eine bedarfsgerechte Administration erbracht werden kann.	Fehlfunktionen können innerhalb von Datenbanksystemen die Datenkonsistenz verletzen. Daten können verloren gehen oder ungewollt von Dritten und unberechtigten Personen eingesehen werden. Systeme könnten nicht mehr oder nur noch eingeschränkt verfügbar werden.  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)	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 Ressourcenwachstum zu spät bemerkt wird. So kann die CPU Usage oder das Memory Usage schnell anwachsen. Auch der Storage eines Betriebssystems kann nicht mehr ausreichend für ein System werden.	Wenn die CPU- und Memory-Usage über einen gewissen Schwellwert geht, fängt das Betriebssystem an zu Priorisieren. Dies wird primär der Endanwender in form von Performance Einbussen bemerken. Im schlimmsten Fall steht eine Anwendung nicht mehr zur Verfügung.  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.  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.	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	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.  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.	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	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



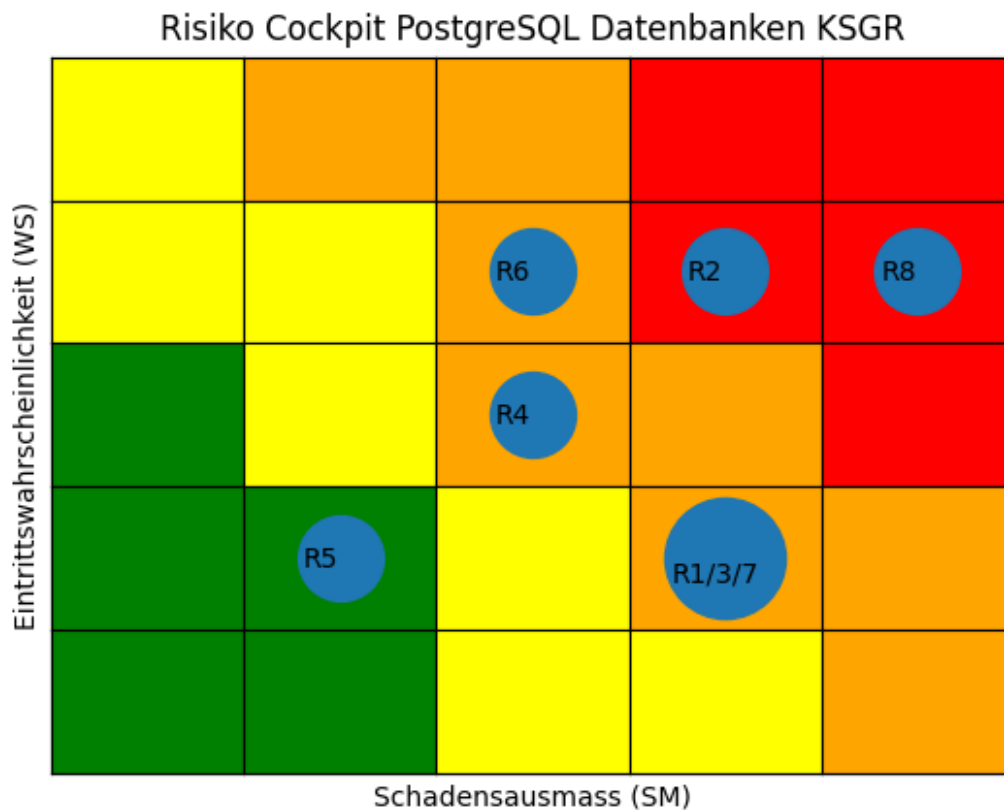


Abbildung 1.6: Risiken bestehende Lösung

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:

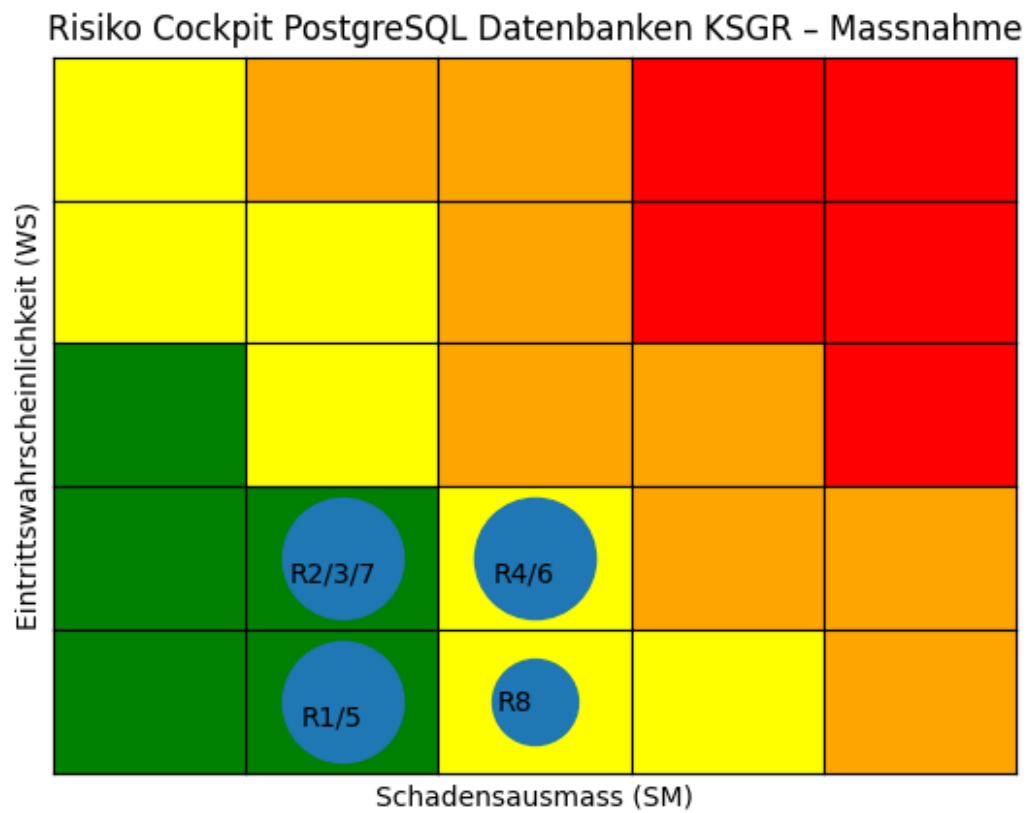


Abbildung 1.7: Risiken bestehende Lösung mit Massnahmen

## 1.2 Zieldefinition

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [39, 44] folgende zehn Tasks die zum täglichen Alltag gehören:

Nr.	Aufgabe	Beschreibung	Wichtigkeit
1	Failover	In einem Fehlerfall soll die DB-Node auf einen Standby-Node übergeben werden. Nach einem Failover muss der DB-Node wieder vom Standby-Node auf den Primären Node zurückgesetzt werden.	Hoch
2	Failover Restore	Dabei darf es zu keinem Datenverlust kommen, also alle Daten die auf dem Standby-Node erfasst wurden, müssen auf den Primären DB-Node zurückgeschrieben werden beim Failover Restore Die Datenmenge von Datenbanken wachsen in der Regel beständig.	Hoch
3	Filesystem Management	Die Belegung von Tablespace und Filesystem muss deshalb überwacht und ggf. erweitert werden. Läuft eine Disk voll kommt es im besten Fall zu einem Stillstand der DB, im schlimmsten Fall zu Inkonsistenzen und Datenverlust Nebst den allgemeinen Metriken wie CPU / Memory Usage und der Port Verfügbarkeit gibt es noch eine Reihe weiterer Aspekte die überwacht werden müssen.	Hoch
4	Monitoring	Zum Beispiel ob es zu Verzögerungen bei der Replikation kommt oder die Tablespace genügend Platz haben. Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall. PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. Zudem wird im Rahmen des gleichen Scheduled Tasks ein Cleanup vorgenommen,	Mittel
5	Statistiken / Cleanup Jobs justieren	so dass z.B. gelöschte Datensätze den Disk Space nicht sinnlos belegen. Die Konfiguration dieser Jobs muss an der Metrik der Datenbank angepasst werden, weil gewisse Tasks dann entweder viel zu oft oder viel zu wenig bis gar nicht mehr ausgeführt werden.	Mittel
6	SQL optimierungen	In PostgreSQL können inperfomante SQL Statements ausgelesen werden und zum Teil werden auch Informationen zum Tuning geliefert[18]. 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. Das Filesystem muss beim Erreichen von 95% Usage automatisiert vergrößert werden.	1	Muss	
3	Filesystem Management	Hoch	Die Vergrößerung muss anhand der Wachstumsrate (die mittels Linux Commands zu ermitteln ist), vergrößert werden	4	Kann	
4	Monitoring	Mittel	Der Status der Clusterumgebung und der Replikation muss im PRTG überwacht werden	2	Muss	
5	Statistiken / Cleanup Jobs justieren	Mittel	Regelmässig müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden Es gibt SQL Abfragen, mit dem fehlende Indizes ermittelt werden können. Diese Indizes sollen automatisiert erstellt werden.	2	Muss	
6	SQL optimierungen	Tief	Im gleichen Zug sollen aber auch Indizes, welche nicht verwendet werden, entfernt werden. Sie tragen nicht nur nichts zu performanteren Abfragen bei sondern beziehen unnötige Ressourcen bei Datenmanipulationen[18]. Tabellen und Indizes können sich aufblähen (bloated table / bloated index)	2	Kann	
7	Health Checks und Aktionen (Maintenance)	Hoch	Ist ein Index aufgebläht, kann dies mittels eines REINDEX mit geringem Impact auf die Datenbank gelöst werden[2].	2	Muss	
8	Housekeeping	Mittel	Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden.	3	Kann	
9	Verwalten von DB Objekten	Tief	Keine Automatisierung möglich	5		
10	User Management	Tief	Regelmässige Reports sollen User 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 evaluiert werden
- Resultate  
Aus den mindestens drei Methoden muss die optimale Methode ermittelt werden.  
Am Ende muss zudem ein Funktionierendes Testsystem bestehen.

Daraus ergeben sich folgende Ziele:

Nr.	Ziel	Beschreibung	Priorität
1	Evaluation	Am Ende der Evaluationsphase müssen mindestens drei Methoden für einen PostgreSQL HA Cluster müssen evaluiert werden. Innerhalb der evaluation muss analysiert werden, welche Methode oder welches Tool sich hierfür eignen würde.	Hoch
2	Testsystem	Am Ende der Diplomarbeit muss ein funktionierendes Testsystem Installiert sein.	Hoch
3	Automatisierter Failover	Ein PostgreSQL Cluster muss im Fehlerfall auf mindestens einen Standby-Node umschwenken. Dabei muss das Timeout so niedrig sein, dass Applikationen nicht auf ein Timeout laufen.	Hoch
4	Automatisierter Failover Restore	Nach einem Failover muss es zu einem Fallback oder Failover Restore kommen, sobald der Primary-Node wieder verfügbar ist.	Hoch
5	Monitoring - Cluster Healthcheck	Die wichtigsten Parameter für das Monitoring des PostgreSQL Clusters (isready, Locks, 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 Folgende Testziele müssen erreicht werden:	Hoch
13	Testziele	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	
	Rocky Linux	
Sekundäre Linux Distributionen	Oracle Linux	RedHat Enterprise Linux (RedHat Enterprise Linux (RHEL)), Rocky Linux oder Oracle Linux wird nur eingesetzt, wenn es nicht anders möglich ist
	RedHat Enterprise Linux (RedHat Enterprise Linux (RHEL))	
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[65] folgende Abgrenzungen:

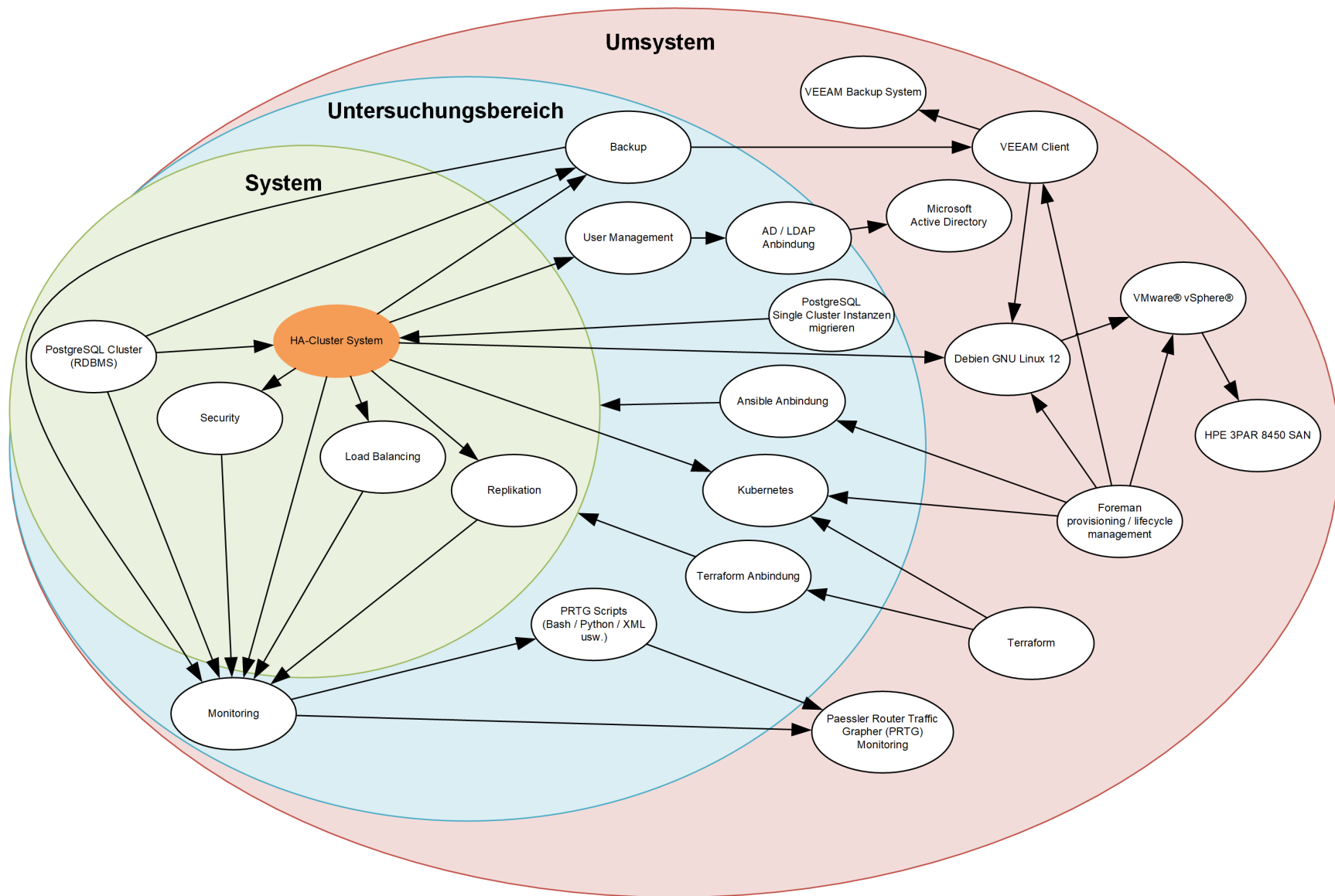


Abbildung 1.8: Systemabgrenzung

1.4 Abhängigkeiten

Es existieren Technische und Organisatorische Abhängigkeiten. Diese haben sowohl ein Risiko als auch einen Impact wenn das Risiko eintritt. 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 das Risiko eines Ressourcenengpasses	Projekt kann nicht Zeitgemäss abgeschlossen werden
5	GitLab	Sicherung	Sicherung von Konfigurationen, Scripts usw.	GitLab ist Implementiert und Betriebsbereit.	GitLab steht nicht mehr zur Verfügung	Keine Versionierung und 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.

Tabelle 1.9: Abhängigkeiten







Aus den Abhängigkeiten heraus wurden folgende Risiken identifiziert:

Identifikation				Abschätzung		Behandlung			
ID	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM	Massnahmen ergreifen?	Zielwert		Massnahme
							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. Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase.	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	Das 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

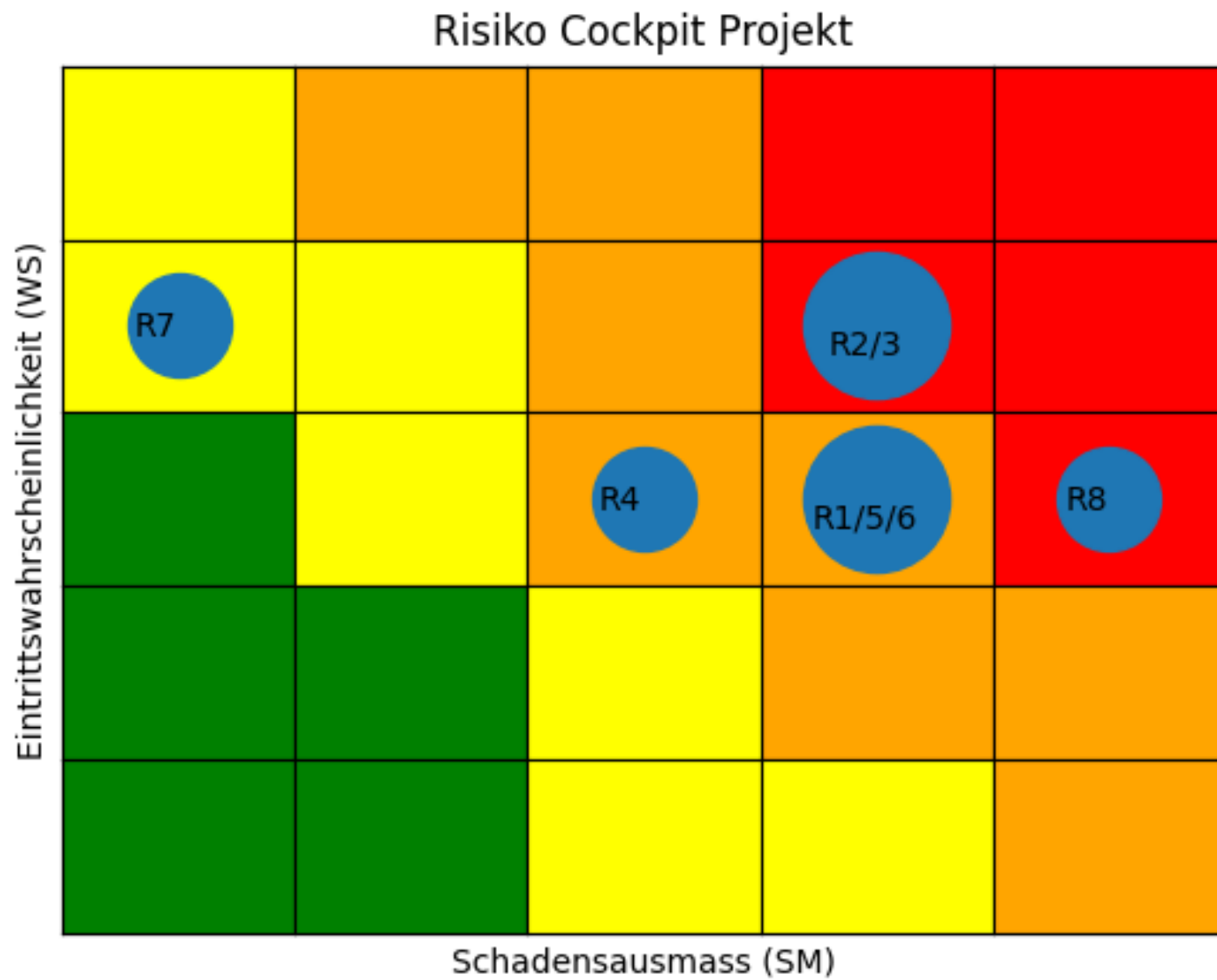


Abbildung 1.9: Projektrisiken

Mit den entsprechenden Massnahmen können die Risiken gesenkt werden:

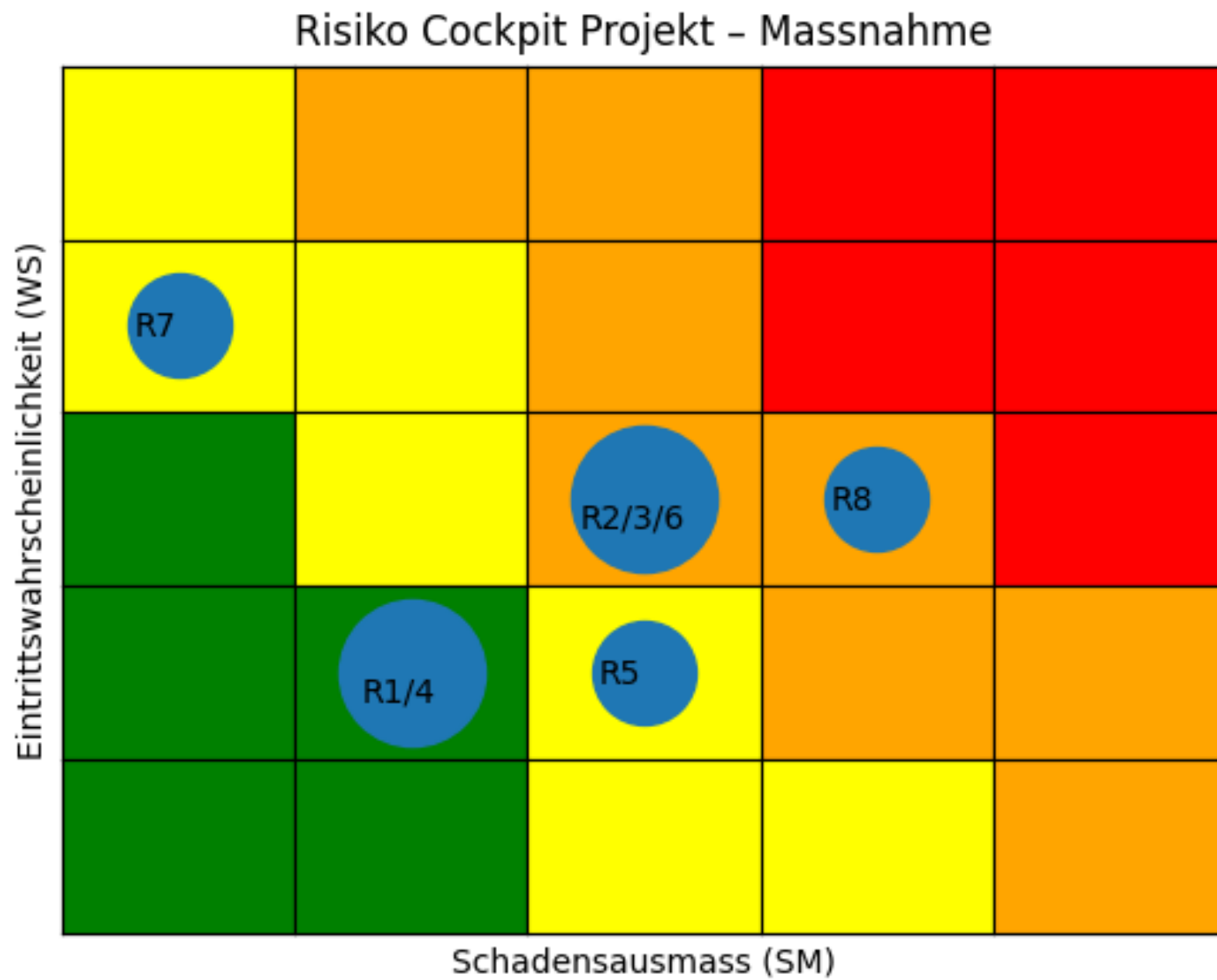


Abbildung 1.10: Projektrisiken mit Massnahmen

1.6 Vorgehensweise und Methoden

1.7 Projektmanagement

## 1.7.1 Projektcontrolling

	Phase	Subphase	Dauer [h]	Geplante Dauer [h]	Verbleibende Zeit [h]
0	Dokumentation	-	13.8	80	66.2
1	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	7.5	16	8.5
2	Evaluation	Anordnungskatalog	4.5	16	11.5
3	Evaluation	Vorbereitung Benchmarking	0.0	4	4.0

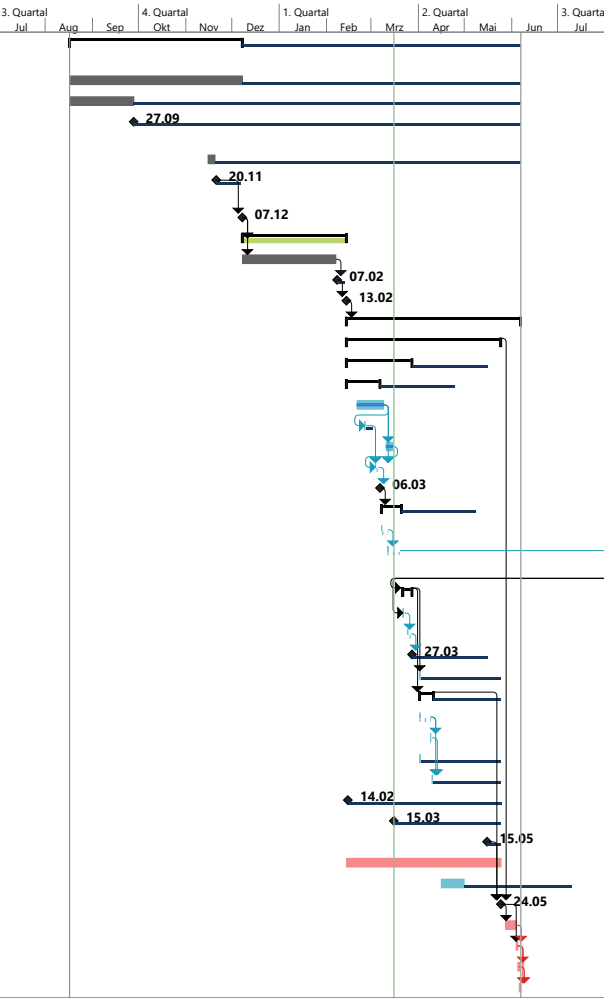
Tabelle 1.11: Projektcontrolling

## GANTT-Diagramm

### 1.7.2



Vorgangsm		Vorgangsname		Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang	Ist-Ende	Geplante Arbeit	Geleistete Arbeit	3. Quartal			4. Quartal			1. Quartal			2. Quartal			3. Quartal		
														Jul	Aug	Sep	Okt	Nov	Dez	Jan	Feb	Mrz	Apr	Mai	Jun	Jul	
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			Vorebereituungsmaßnahmen	43 Tage	7		Nein Fre 08.12.23	Die 06.02.24		NV	NV	20 Std.															
10			Abgabe Statusbericht	1 Tag	9		Ja Mit 07.02.24	Mit 07.02.24		NV	NV	1 Std.															
11			Vernissage	1 Tag?	10		Ja Die 13.02.24	Die 13.02.24		NV	NV	4 Std.															
12			Phase III: Hauptarbeiten	64.75 Tage?	11		Nein Mit 14.02.24	Don 06.06.24		NV	NV	0 Std.															
13			Hauptteil Diplomarbeit	55.75 Tage?			Nein Mit 14.02.24	Fre 24.05.24		NV	NV	200 Std.															
14			Umsetzung	25 Tage?			Nein Mit 14.02.24	Mit 27.03.24		NV	NV	0 Std.															
15			Evaluation	13 Tage?			Nein Mit 14.02.24	Mit 06.03.24		NV	NV	0 Std.															
16			Anforderungskatalog	10 Tage			Nein Mit 21.02.24	Fre 08.03.24	Mit 21.02.24	Fre 08.03.24		16 Std. 4.5															
17			Vorbereitung Benchmarking	1 Tag	16		Nein Mon 26.02.24	Mon 26.02.24		NV	NV	4 Std.															
18			Analyse PostgreSQL HA Cluster Lösungen	4 Tage	16		Nein Mon 11.03.24	Don 14.03.24	Mon 11.03.24	Don 14.03.24		16 Std. 7.5															
19			Gegenüberstellung	2 Tage	16;17;18		Nein Mon 04.03.24	Die 05.03.24		NV	NV	8 Std.															
20			Variantenentscheid	1 Tag?	19		Ja Mit 06.03.24	Mit 06.03.24		NV	NV	4 Std.															
21			Aufbau und Implementation Testsystem	8 Tage?	20		Nein Fre 08.03.24	Mit 20.03.24		NV	NV	27 Std.															
22			Basisinfrastruktur	2 Tage			Nein Fre 08.03.24	Mon 11.03.24		NV	NV	4 Std.															
23			Installation und Konfiguration PostgreSQL HA Cluster	5 Tage	22		Nein Die 12.03.24	Die 19.03.24		NV	NV	20 Std.															
24			Technical Review	1 Tag?	23		Ja Mit 20.03.24	Mon 16.09.24		NV	NV	3 Std.															
25			Testing	4 Tage?	24		Nein Fre 22.03.24	Mit 27.03.24		NV	NV	14 Std.															
26			Testing Testsystem	1 Tag	24		Nein Fre 22.03.24	Fre 22.03.24		NV	NV	8 Std.															
27			Protokollierung	2 Tage	26		Nein Mon 25.03.24	Die 26.03.24		NV	NV	4 Std.															
28			Review und Auswertung	1 Tag?	27		Ja Mit 27.03.24	Mit 27.03.24		NV	NV	2 Std.															
29			Troubleshooting und Lösungsfindung	1 Tag?	25		Nein Die 02.04.24	Die 02.04.24		NV	NV	8 Std.															
30			Resultate	6 Tage?	25		Nein Die 02.04.24	Mit 10.04.24		NV	NV	7 Std.															
31			Zielüberprüfung	4 Tage			Nein Die 02.04.24	Mon 08.04.24		NV	NV	2 Std.															
32			Schlussfolgerung	1 Tag?	31		Nein Die 09.04.24	Die 09.04.24		NV	NV	2 Std.															
33			Weiteres Vorgehen / offene Arbeiten	1 Tag?			Nein Die 02.04.24	Die 02.04.24		NV	NV	1 Std.															
34			Persönliches Fazit	1 Tag	31;32		Nein Mit 10.04.24	Mit 10.04.24		NV	NV	2 Std.															
35			1. Expertengespräch	1 Tag?			Ja Mit 14.02.24	Mit 14.02.24		NV	NV	0 Std.															
36			2. Expertengespräch	1 Tag?			Ja Fre 15.03.24	Fre 15.03.24		NV	NV	0 Std.															
37			Letztes Expertengespräch	1 Tag?			Ja Mit 15.05.24	Mit 15.05.24		NV	NV	0 Std.															
38			Dokumentation	55.75 Tage			Nein Mit 14.02.24	Fre 24.05.24		NV	NV	80 Std. 11.2															
39			Puffer	9 Tage			Nein Die 16.04.24	Die 30.04.24		NV	NV	16 Std.															
40			Abgabe Diplomarbeit	1 Tag?	13;30;37		Ja Fre 24.05.24	Fre 24.05.24		NV	NV	0 Std.															
41			Vorbereitung Präsentation	5 Tage	40		Nein Die 28.05.24	Mon 03.06.24		NV	NV	20 Std.															
42			Präsentation und Fachgespräch	1 Tag?	40;41		Nein Die 04.06.24	Die 04.06.24		NV	NV	1 Std.															
43			Diplomausstellung	1 Tag?	42		Nein Mit 05.06.24	Mit 05.06.24		NV	NV	0 Std.															
44			Diplomfeier	1 Tag?	42;43		Nein Don 06.06.24	Don 06.06.24		NV	NV	0 Std.															



Projekt: Diplomarbeit - Postgre  
Datum: Sam 16.03.24

Vorgang

Unterbrechung

Meilenstein

Sammelvorgang

Projektsammelvorgang

Inaktiver Vorgang

Inaktiver Meilenstein

Inaktiver Sammelvorgang

Manueller Vorgang

Nur Dauer

Manueller Sammelrollup

Manueller Sammelvorgang

Nur Anfang

Nur Ende

Externe Vorgänge

Stichtag

Kritisch

Kritische Unterbrechung

Verspätet

Geplant

Geplanter Meilenstein

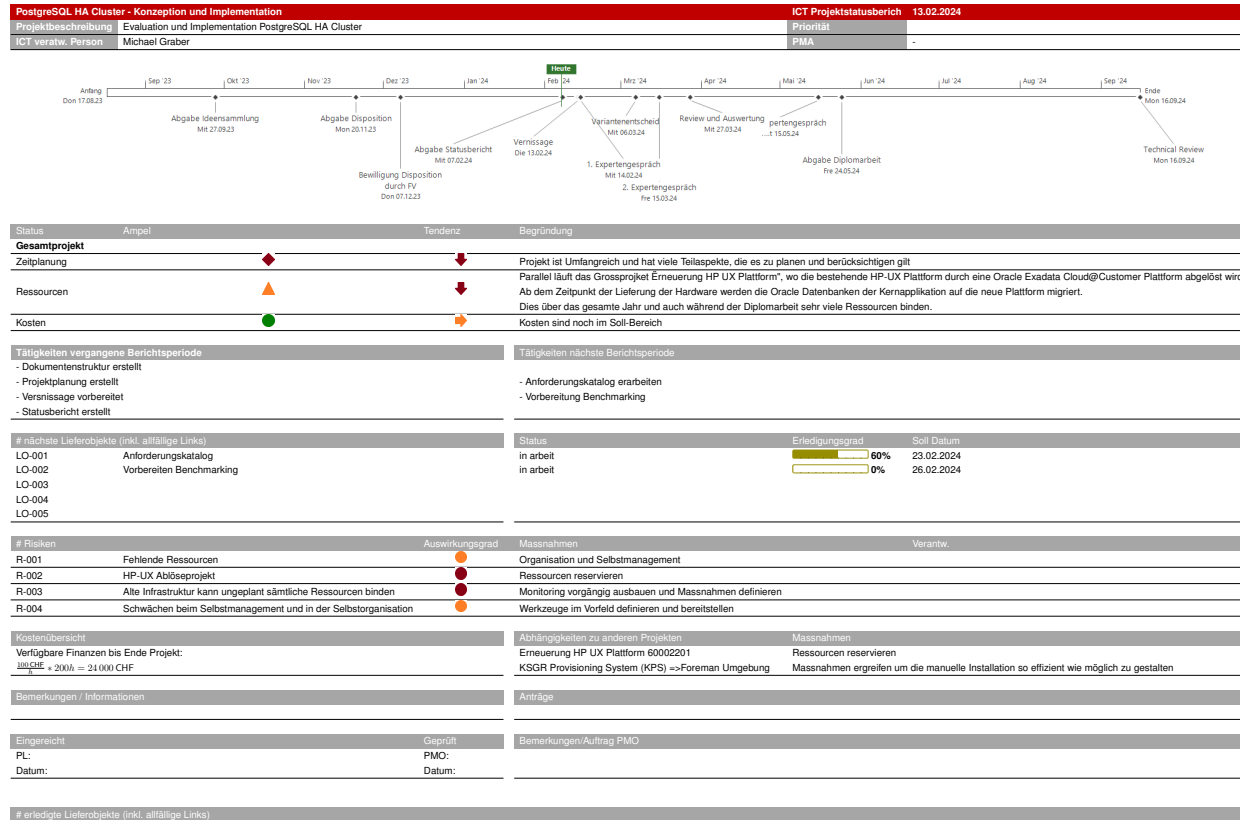
Geplanter Sammelvorgang

In Arbeit

Manueller Fortschritt

Puffer

## Initialer Statusbericht



### Tabelle 1.12: Initialer Statusbericht

1.8.2 Zweiter Statusbericht

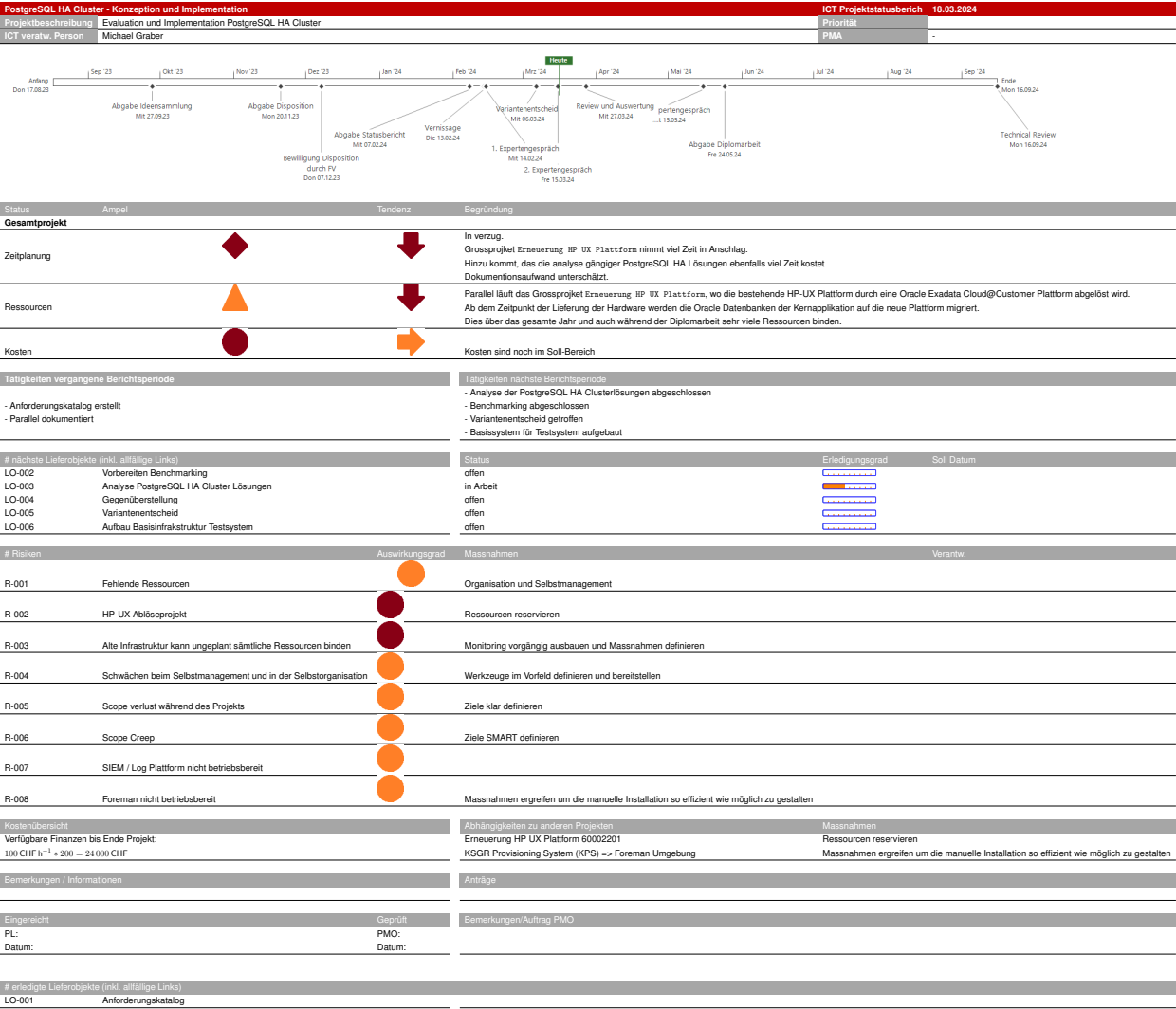


Tabelle 1.13: Zweiter Statusbericht

## 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		Norman Süsstrunk	-	Michael Graber	

Tabelle 1.14: Fachgespräche

Das Protokoll ist im Anhang zu finden.

## 2 Umsetzung

### 2.1 Evaluation

#### 2.1.1 Exkurs Architektur

##### 2.1.1.1 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

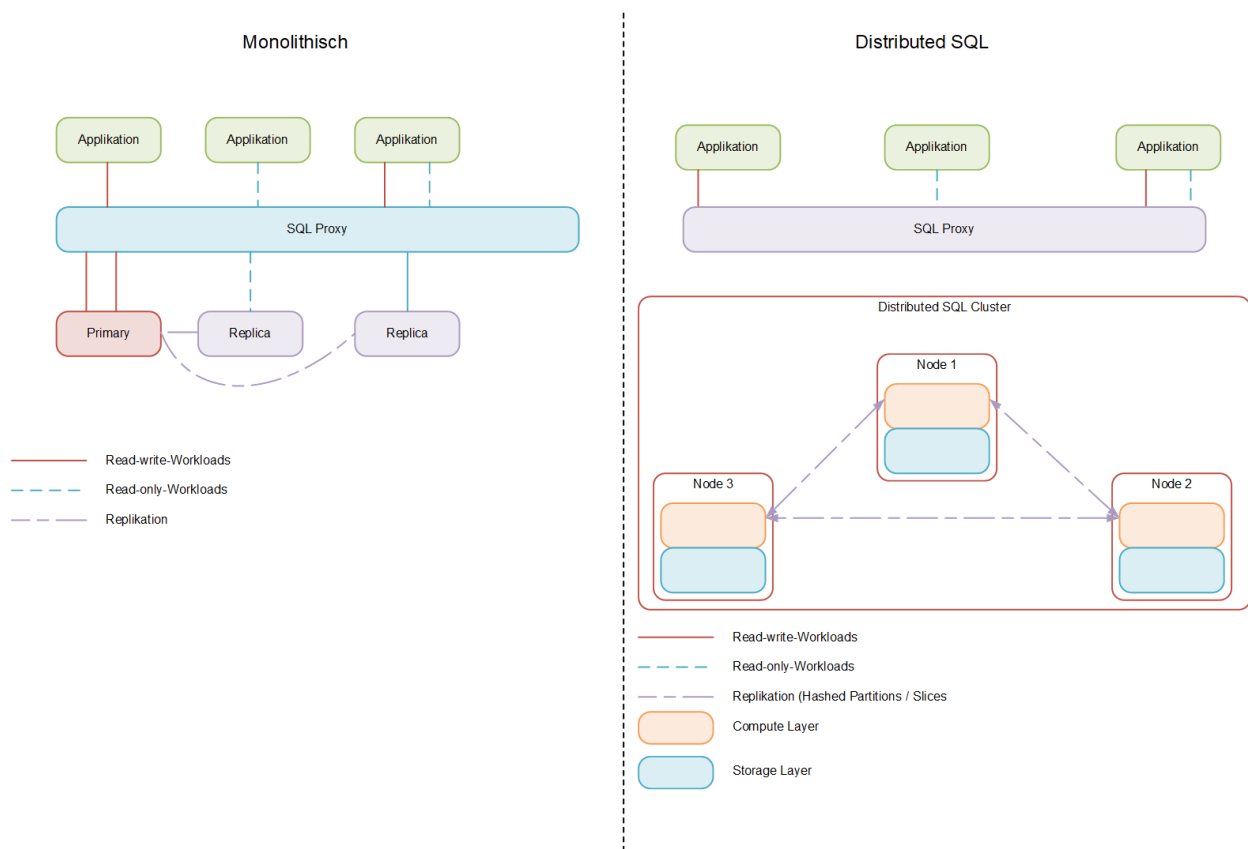


Abbildung 2.1: Monolithische vs. verteilte SQL Systeme

##### 2.1.1.2 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[42]. 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 mindestens 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 performant, allerdings kann es je nach Latenz und genereller Auslastung zu Datenverlusten kommen, wenn es zum Failover kommt.

#### **2.1.1.3 Quorum**

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

#### **Quorum**

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

#### **Throughput**

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

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

#### **Fehlertoleranz**

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

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

Hier ein Beispiel wie sie in den Artikeln [40, 51, 36] 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

#### 2.1.1.4 CAP Theorem

Das CAP Theorem besagt, dass nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[26].

##### Konsistenz - Consistency

Die Datenbank ist Konsistent, alle Clients sehen gleichzeitig die gleichen Daten unabhängig auf welchem Node Zugriffen wird. Hierzu muss eine Replikation der Daten an alle Nodes stattfinden und der Commit zurückgegeben werden, also eine Synchronre 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 2.2: CAP-Theorem

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

#### 2.1.1.5 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[38]:



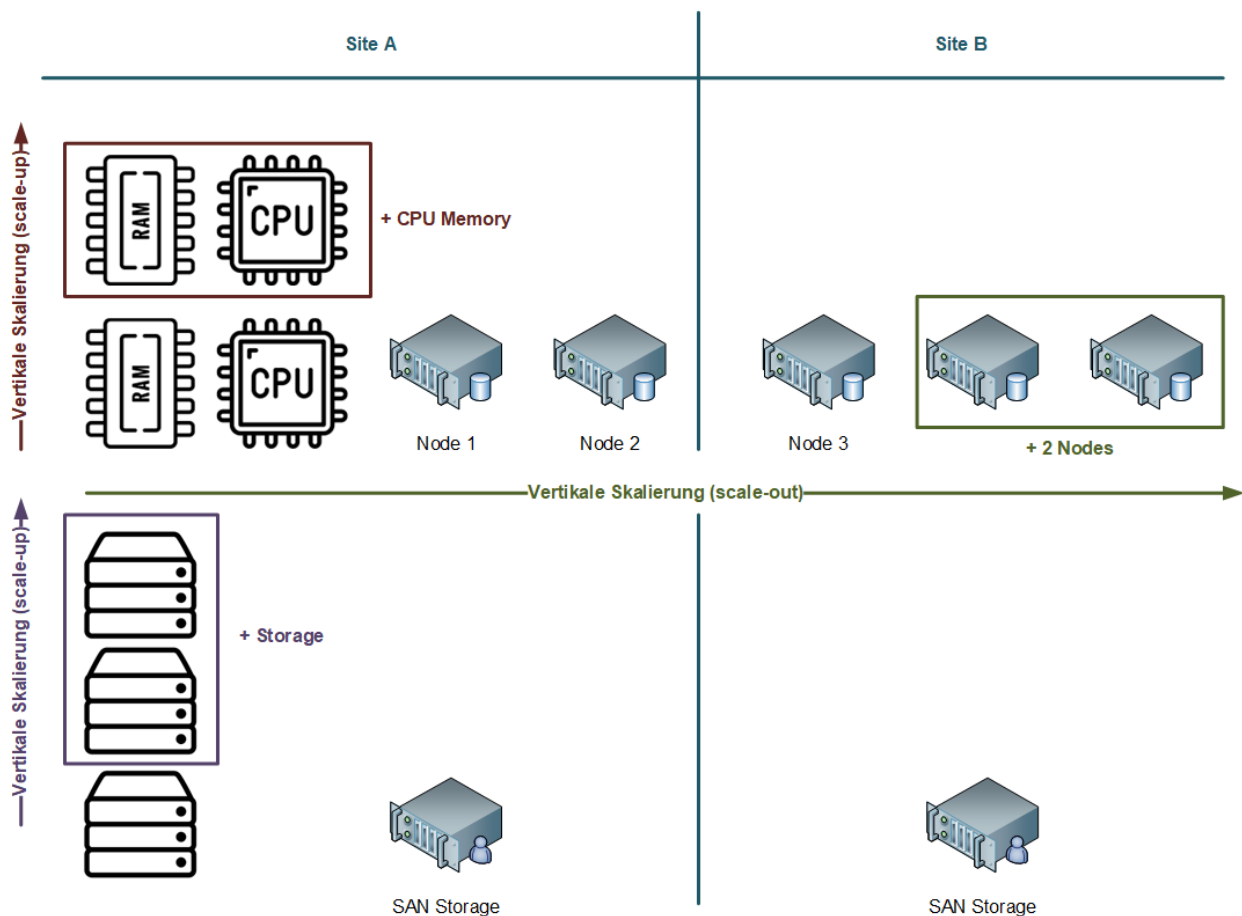


Abbildung 2.3: Datenbankskalierung

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	Systemvielfalt		Es muss mindestens eine Monolithisches und mindestens 2 zwei Distributed SQL Cluster ermittelt werden	Beides	MUSS
2	Synergien		Skripte und APIs des Monolithisches Systems müssen auch in einem Distributed SQL System verwendet werden können	Beides	MUSS
3	Failover	Automatismus	Das Clustersystem muss bei einem Nodeausfall automatisch auf einen anderen Node umstellt	Beides	MUSS
4	Failover	Connection - Stabilität	Beim Failover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
5	Failover	Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
6	Switchover	Skript / API	Das System muss ein Skript oder eine API liefern, welche einen geordneten Switchover auf einen anderen Node erlaubt	Beides	MUSS
7	Switchover	Connection - Stabilität	Beim Switchover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
8	Switchover	Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgeführt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
9	Restore	Skript / API	Das Clustersystem muss ein Skript oder eine API liefern, welche das einfache und ggf. automatisierte 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	Monolithisch	MUSS
14	Replikation	Failover / Switchover Garantie	Die Replikation muss sicherstellen, das es bei einem Failover/Switchover zu keinem Fehler kommt	Monolithisch	MUSS
15	Replikation	Throughput	Beschreibt, wie viele Transaktionen pro Zeiteinheit vom Primary an die Replikas gesendet und Committed werden. Dieser Wert ist bei 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	Robustheit	Das Quorum des Clustersystems muss robust genug sein, um eine Split-Brain-Situation zu verhindern	Beides	MUSS
20	Connection		Das Clustersystem muss sicherstellen, dass eine Applikation ohne Entwicklungsaufwand mittels dem PostgreSQL Wired Connector zugreifen kann	Beides	MUSS
21	Management-API	Management-API vorhanden	Das Clustersystem muss Skripte oder eine API liefern, mit dem das System zu konfigurieren, verwalten oder überwachen zu können. Zudem müssen mit geringen Arbeitsaufwand damit Nodes hinzugefügt oder entfernt werden können	Beides	MUSS
22	Management-API	Authentifizierung & Autorisierung	Es müssen gängige Standards für Authentifizierung und Autorisierung mitgebracht werden	Beides	MUSS
23	Management-API	Aufwand	Der Aufwand, der benötigt wird um die DB zu verwalten, Nodes hinzuzufügen oder zu entfernen usw. muss gegeneinander verglichen werden.	Beides	MUSS
24	Backup	Backup mit PostgreSQL Standards	Backups müssen mittels PostgreSQL Standards angezogen werden	Beides	MUSS
25	Backup	Restore mit PostgreSQL Standards	Backups müssen mittels PostgreSQL Standards restored werden können	Beides	MUSS
26	Housekeeping - Log Rotation		Das Clustersystem muss die möglichkeit zur Log Rotation bieten	Beides	MUSS
27	Self 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.	Beides	MUSS
30	Performance	tps - Read-Only	Als Basis dienen hier Informationen wie z.B. GitHub Insights. 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.
Stakeholder 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	2	3	4	5	6	7	8	9	10	11	12	13	14	15
					System vielfalt														
13	15	1	1	System vielfalt															
12	14	2	2	Synergien	1														
11	13	3	3	Fallover	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	16	16	16	14	16
100	120																		

## Legende

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.4: Präferenzmatrix

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

	Gewicht	Nennungen	Rang	Nr.	Ziele	1	2
						Automatismus	Connection-Stabilität
	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 2.5: Präferenzmatrix - Failover

Beim Switchover wurde die Gewichtung wie folgt aufgeteilt:

	Gewicht	Nennungen	Rang	Nr.	Ziele	1	2
						Skript / API	Connection - Stabilität
	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 2.6: Präferenzmatrix - Switchover

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

Gewicht	Nennungen	Rang	Nr.	Ziele	1	2	3
					Skript / API	Datensicherheit	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 2.7: Präferenzmatrix - Restore

Die Replikationsanforderungen resp. deren Gewichtung ist wie folgt aufgebaut:



	Gewicht	Nennungen	Rang	Nr.	Ziele	Synchrone Replikation	1
							Failover / Switchover Garantie 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 2.8: Präferenzmatrix - Replikation

Das Sharding setzt sich aus folgenden Teilen zusammen:

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

### Legende

- Eingabefelder
- Zellbezüge
- berechnete Felder

Abbildung 2.9: Präferenzmatrix - Sharding

Die Quorum-Anforderung ist folgendermassen zusammengesetzt:

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

**Legende**

Eingabefelder

Zellbezüge

berechnete Felder

Abbildung 2.10: Präferenzmatrix - Quorum

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

	Gewicht	Nennungen	Rang	Nr.	Ziele	Management-API vorhanden 1	Authentifizierung & Autorisierung 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 2.11: Präferenzmatrix - Management-API

Anforderungen zum Backup wurden nachfolgend aufgeteilt und gewichtet:

Gewicht	Nennungen	Rang	Nr.	Ziele	Backup mit PostgreSQL Standards
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 2.12: Präferenzmatrix - Backup

Performance-Benchmarking lässt sich in nachfolgende Teile unterteilen:

					1	2	3
Gewicht	Nennungen	Rang	Nr.	Ziele	tps - Read-Only	tps - Read-Writes	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 2.13: Präferenzmatrix - Performance

#### 2.1.3 Testziele erarbeiten

#### 2.1.4 PostgreSQL Benchmarking

PostgreSQL bietet ein Benchmarking-Tool,[34, 1] mit dem die DB Vermessen werden kann.

#### 2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen

##### 2.1.5.1 PostgreSQL Replikation

PostgreSQL bietet von Haus aus Möglichkeiten, um Replikationen durchzuführen. Dabei ist nicht jede gleich gut für jedes Szenario geeignet[33].

### Shared Disk Failover

**File System (Block Device) Replication**

**Write-Ahead Log Shipping**

**Logical Replication**

**Trigger-Based Primary-Standby Replication**

**Data Partitioning**

**Multiple-Server Parallel Query Execution**

#### 2.1.5.2 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[52]. 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.

#### 2.1.5.3 pgpool-II

pgpool-II ist eine Middleware die zwischen einem PostgreSQL Cluster und einem PostgreSQL Client gesetzt wird. pgpool-II bietet folgende Funktionen[49, 31]:

##### **High Availability**

pgpool-II bietet einen automatic Failover genannten Service an, den Watchdog. Dieser schwenkt auf einen Standby-Server und entfernt den Defekten Server. Um false positive Events und Split-brains zu verhindern setzt pgpool-II auf einen eigens entwickelten Quorum-Algorithmus.

### **Connection Pooling**

Bestehende Connections werden wiederverwendet um die Anzahl gleichzeitig offener Connections zu reduzieren. Der Pool wird dabei anhand von Username, Database, Protocol und weiteren Verbindungsparametern zugeordnet.

### **Replikation**

Nebst dem Standard PostgreSQL bietet pgpool-II sein eigenes Replikationssystem an.

### **Load Balancing**

Ähnlich wie Oracle Active Data Guard [17] bietet auch pgpool-II die Möglichkeit, SELECT-Queries und Backup-Jobs auf die Secondary-Nodes umzuleiten um den Primary Node zu entlasten.

### **Limiting Exceeding Connections**

Die Anzahl an concurrent Connections, also gleichzeitiger Verbindungen, ist bei PostgreSQL begrenzt (Systemparameter wird dabei vom DBA gesetzt). pgpool-II speichert alle Connections, die über dem Limit sind, in einer Queue und somit nicht sofort fehlerhaft abgelehnt.

### **Watchdog**

Der Watchdog koordiniert mehrere pgpool-II Nodes und verhindert ein Split-brain.

### **In Memory Query Caching**

pgpool-II speichert SELECT-Queries in einem Cache und verwendet die ResultSets wieder, wenn eine identische Abfrage eingeht.

### **Online Recovery**

pgpool-II bietet die Möglichkeit, einen Online Recovery resp. eine Online Synchronisation eines Nodes durchzuführen, auch kann ein neuer Standby-Node synchronisiert werden. Dafür muss der Node aber im Detached Mode stehen, unabhängig ob der Detach manuell oder von pgpool-II ausgeführt wurde.

#### **2.1.5.4 pg\_auto\_failover**

##### **2.1.5.4.1 Replikation**

##### **2.1.5.4.2 Replikation**

##### **2.1.5.4.3 Proxy**

pg\_auto\_failover benötigt einen HAProxy, um Load Balancing usw. [9]



## 2.1.5.4.4 API / Skripte

## 2.1.5.4.5 Architektur

Die Dokumentation von pg\_auto\_failover [3] zeigt auf, wie der Failover funktioniert:

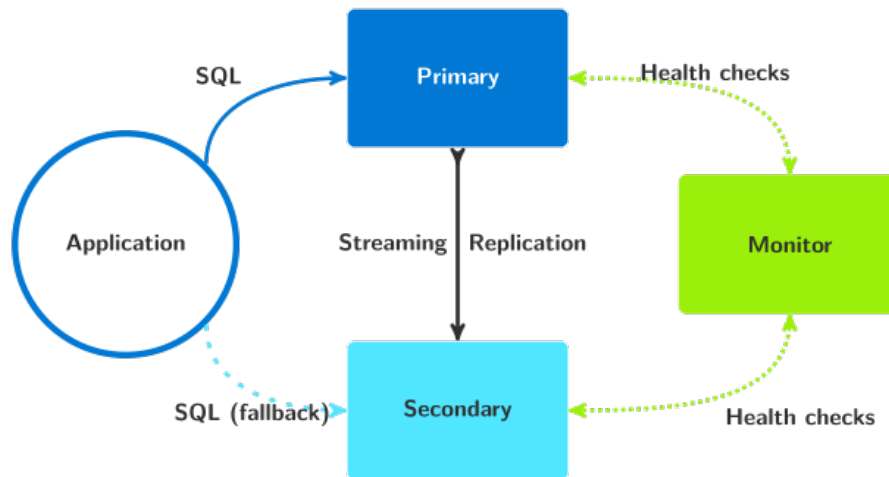


Abbildung 2.14: pg\_auto\_failover-Architektur - Single Standby

Aber auch Multi-Nodes können eingebunden werden[11]:

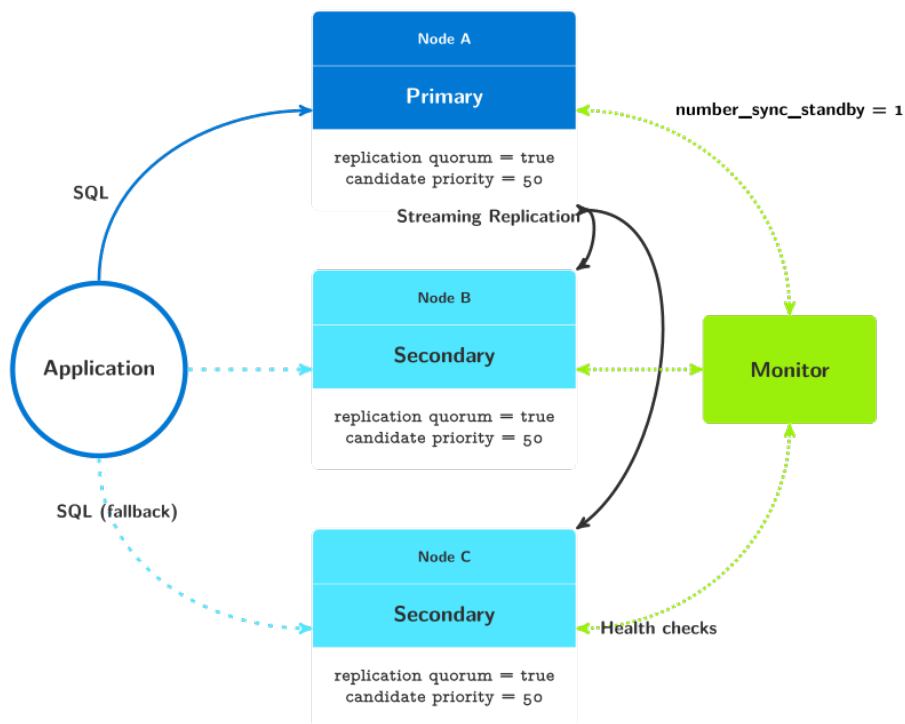


Abbildung 2.15: pg\_auto\_failover-Architektur - Multi-Node Standby

pg\_auto\_failover kann Citus einbinden[5]. Allerdings bleibt die Architektur im Kern immer Monolithisch.

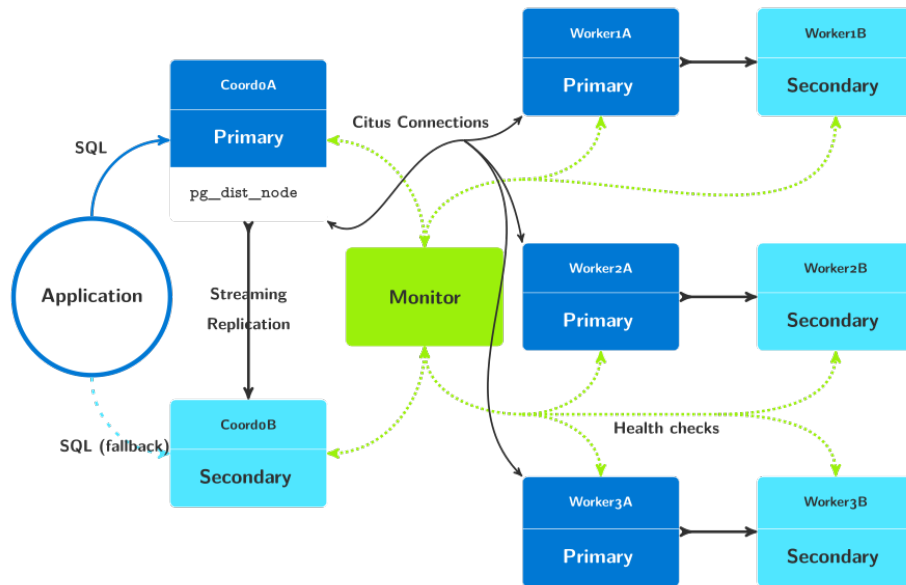


Abbildung 2.16: pg\_auto\_failover-Architektur - Citus

#### 2.1.5.5 Patroni

##### 2.1.5.5.1 Replikation

##### 2.1.5.5.2 Proxy

Patroni benötigt einen HAProxy, um Load Balancing usw. [9]

##### 2.1.5.5.3 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.

##### 2.1.5.5.4 etcd

Patroni benötigt etcd als key-value-store

## 2.1.5.5.5 Architektur

Das Architektur-Schaubild sieht folgendermassen aus:

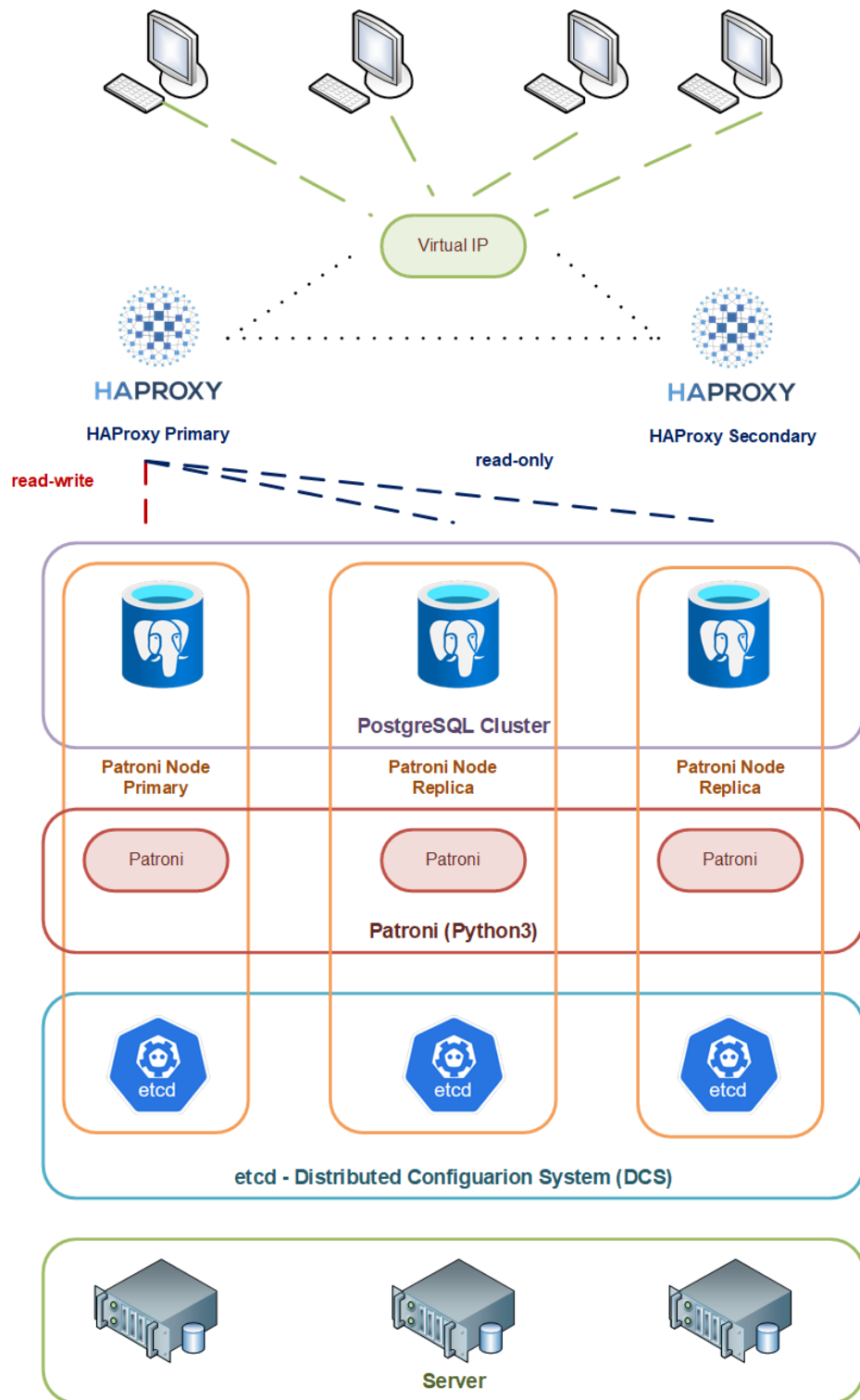


Abbildung 2.17: Patroni-Architektur

## 2.1.5.5.6 Maintenance

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

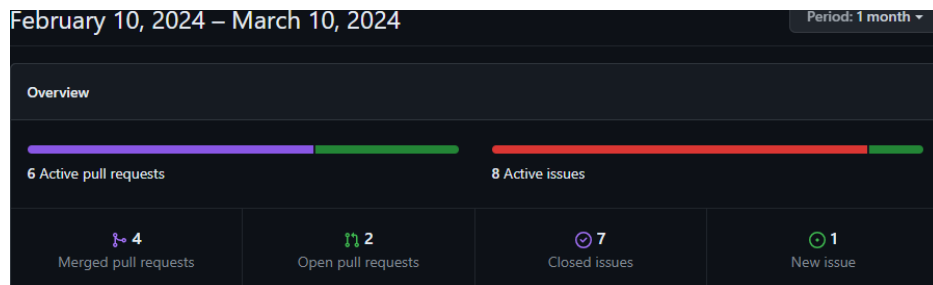


Abbildung 2.18: Patroni - Pulse

Code wird Regelmässig hinzugefügt und entfernt:

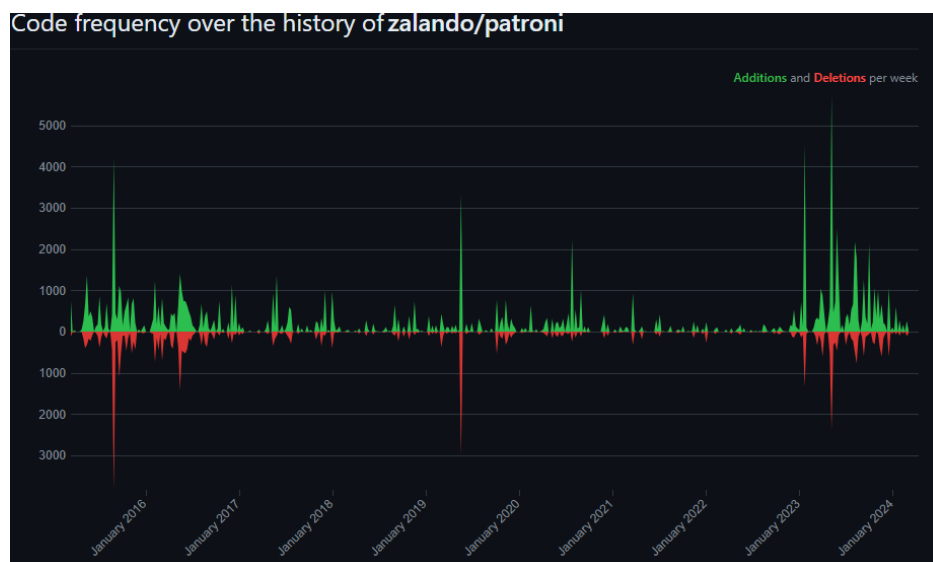


Abbildung 2.19: Patroni - Code Frequency

Das Projekt hält auch die gängigen Standards auf Github ein:

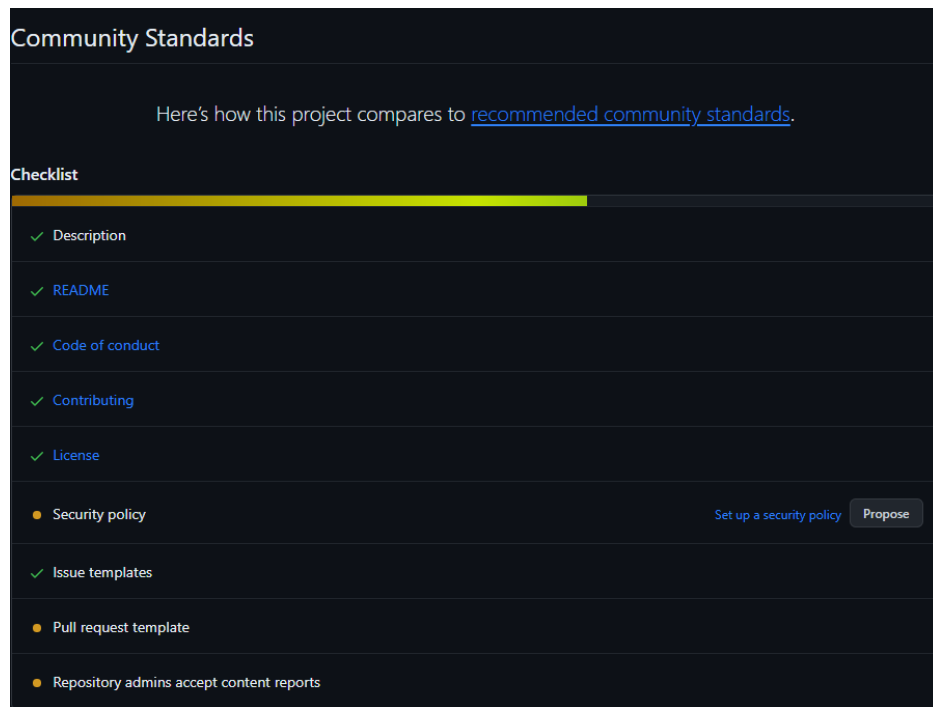


Abbildung 2.20: Patroni - Community Standards

Die Contributors commiten, löschen und erweitern Patroni Regelmässig:

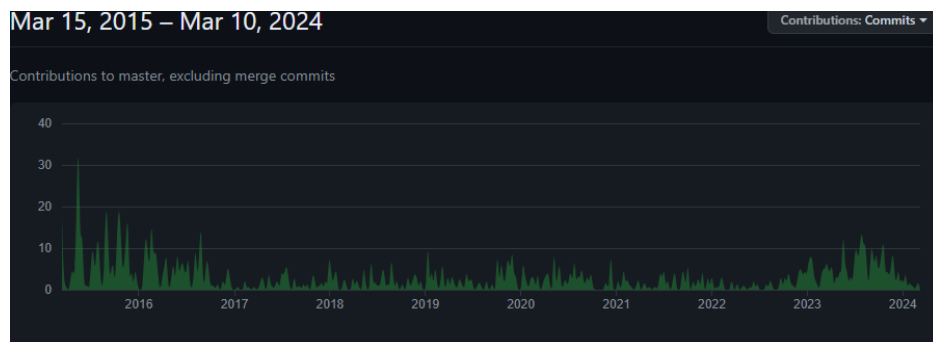


Abbildung 2.21: Patroni - Contributors Commits

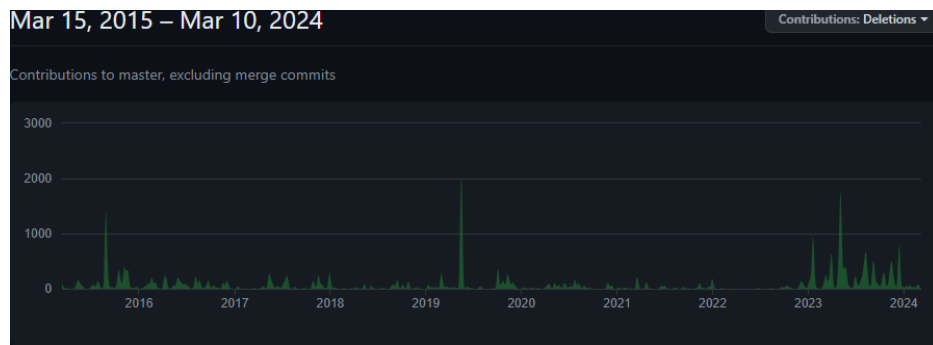


Abbildung 2.22: Patroni - Contributors Deletations

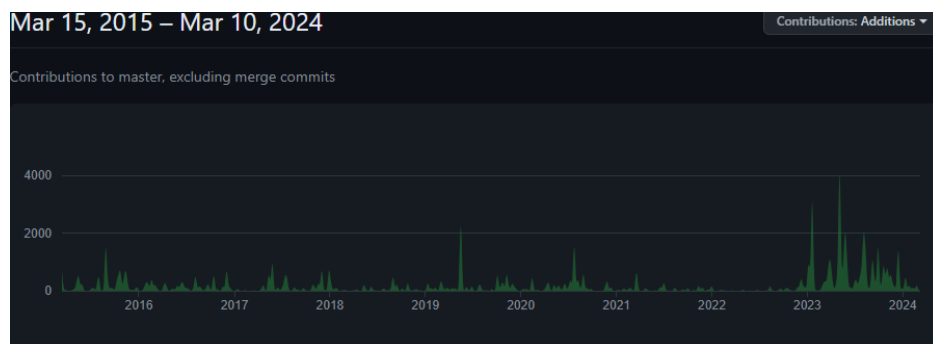


Abbildung 2.23: Patroni - Contributors Additions

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

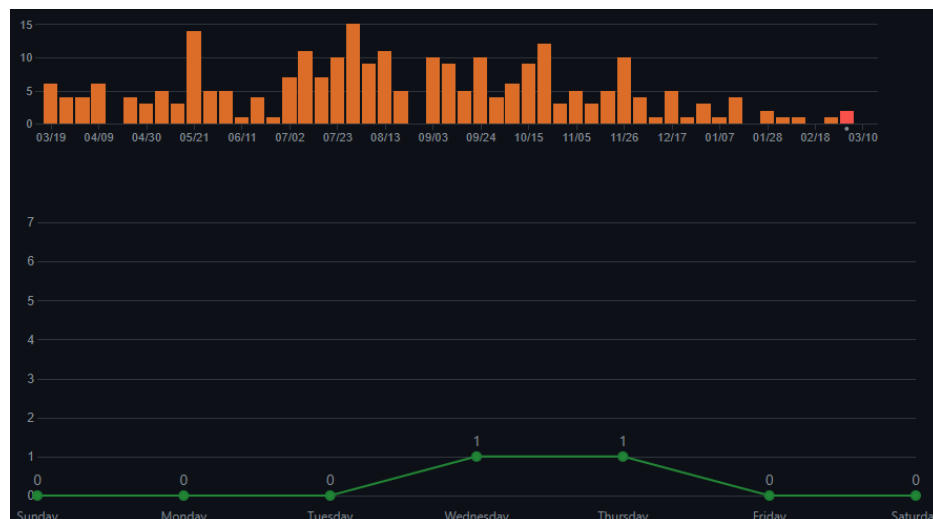


Abbildung 2.24: Patroni - Commit Activity

Nebst Zalando selbst, hat auch EnterpriseDB[19] ein grösseres Repository eingebunden. Dies weil EnterpriseDB stark auf Patroni setzt.



Abbildung 2.25: Patroni - Network Graph

#### 2.1.5.6 CloudNativePG

#### 2.1.5.7 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[62].

#### 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 wiederum ist in den Microsoft Konzern eingebettet

##### 2.1.5.8.1 Architektur

##### 2.1.5.8.1.1 Citus Coordinator und Workers

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

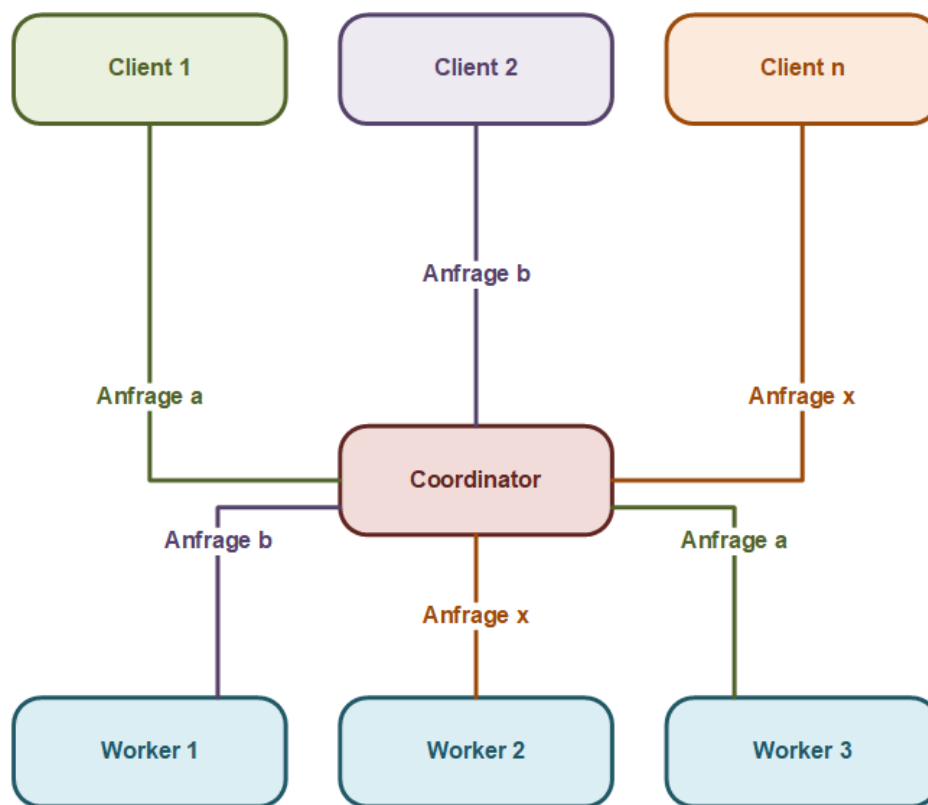


Abbildung 2.26: Citus - Coordinator und Workers

#### 2.1.5.8.1.2 Citus Sharding

Citus bietet zwei Sharding-Modelle an.

**Row-based sharding** Beim diesen sharding werden Tabellen anhand einer Distribution Column aufgeteilt. [7, 4]

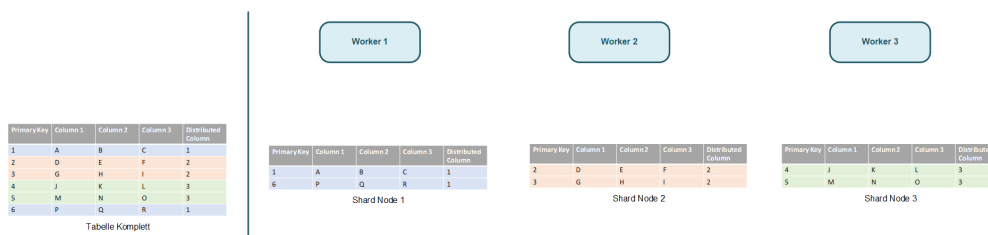


Abbildung 2.27: Citus - Row-Based-Sharding

#### Schema-based sharding



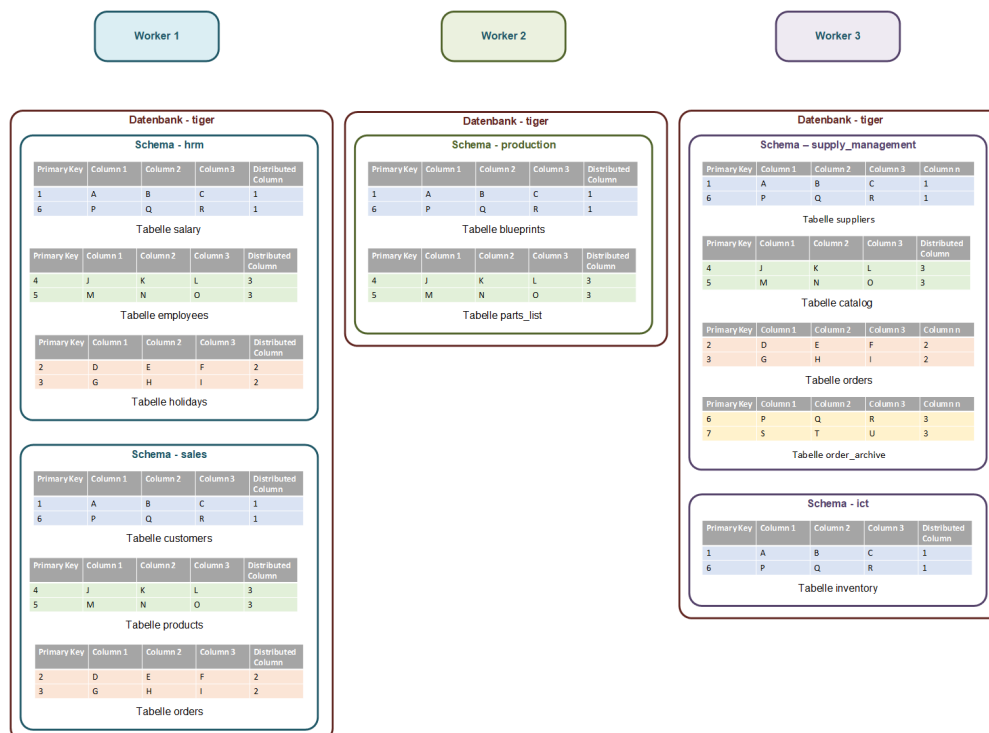


Abbildung 2.28: Citus - Schema-Based-Sharding

**Schlussfolgerung** Beide Sharding-Methoden haben eine grosse Schwäche. Sie sind nicht vollständig ACID-Konform (??) da Datenverlust entstehen kann, wenn ein Node wegfällt. Die Shards müssen mit entsprechenden mit Replikation gesichert werden[6]. Dies muss aber bei der evaluation mittels Tests noch bestätigt werden.

#### 2.1.5.8.2 Maintenance

Bei Stackgres gab es im letzten Monat keine wirkliche Bewegung:

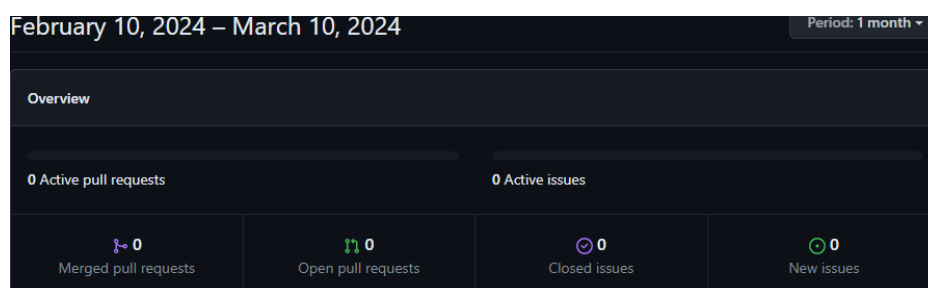


Abbildung 2.29: Stackgres - Pulse

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

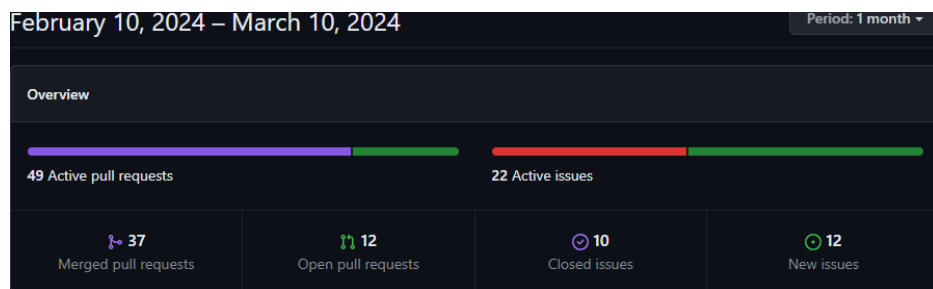


Abbildung 2.30: Citus - Pulse

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

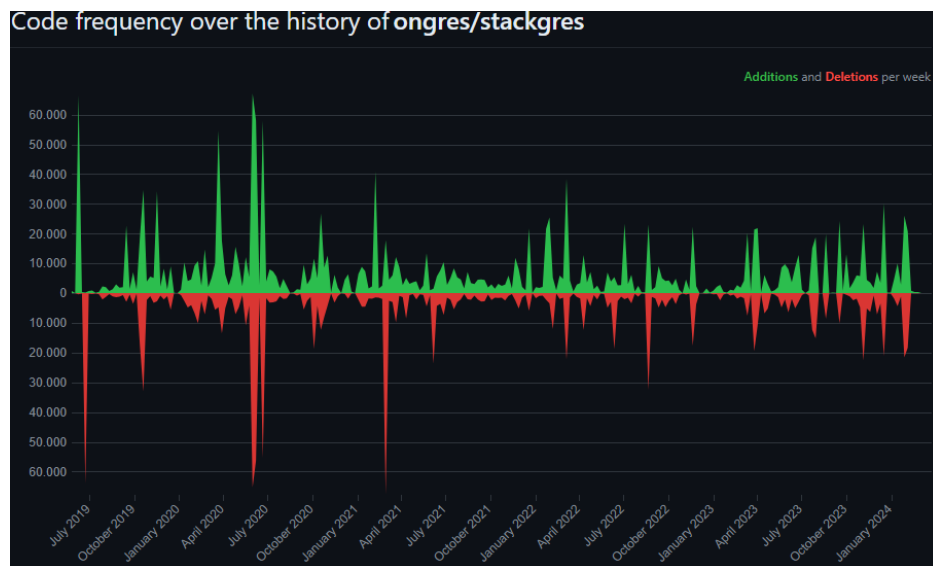


Abbildung 2.31: Stackgres - Code Frequency

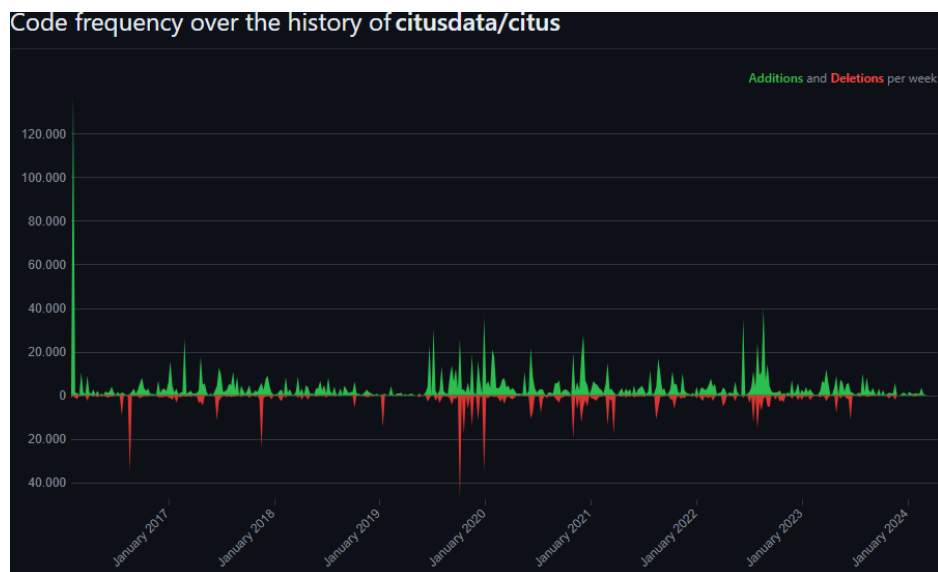


Abbildung 2.32: Citus - Code Frequency

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

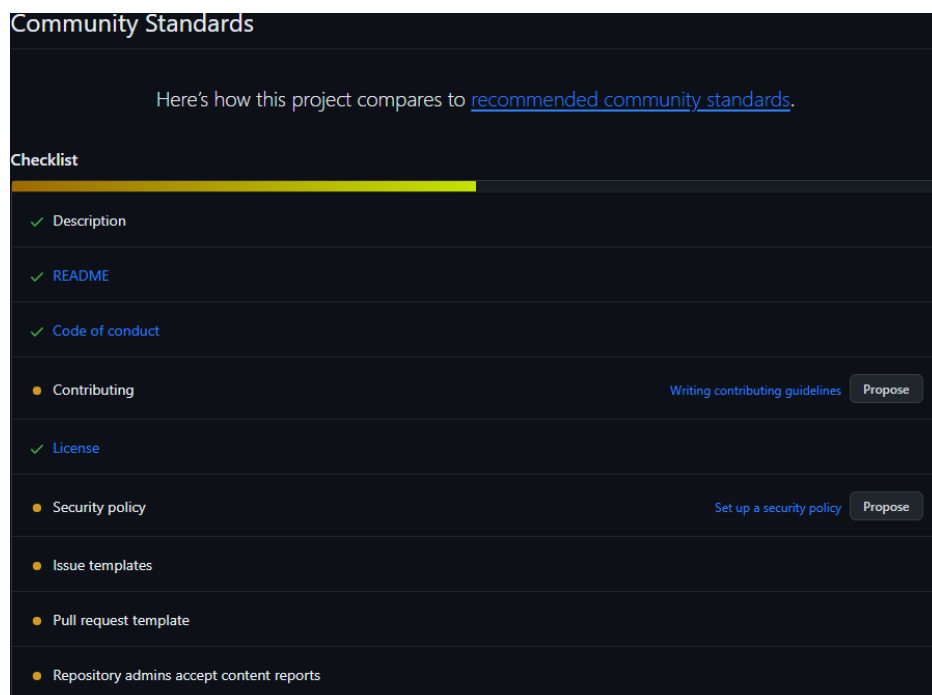


Abbildung 2.33: Stackgres - Community Standards

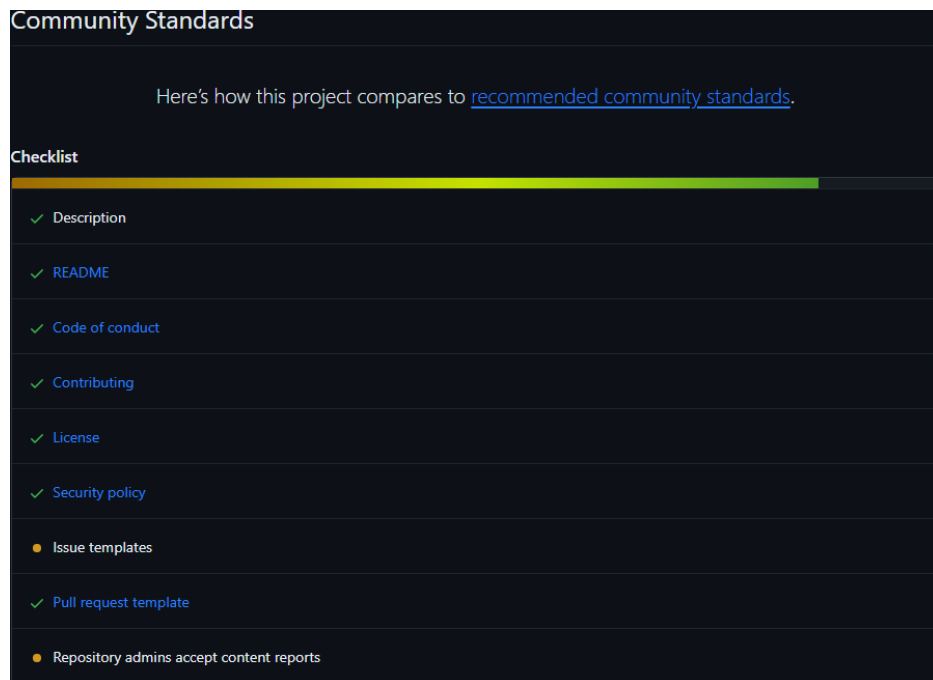


Abbildung 2.34: Citus - Community Standards

Die Stackgres Contributors pflegen aktiv Additions ein, löschen Regelmässig und Committen ebenfalls auf die main-Branch. Citus, dessen Repository länger Committed wird, hat weniger bewegung auf die main-Branch.

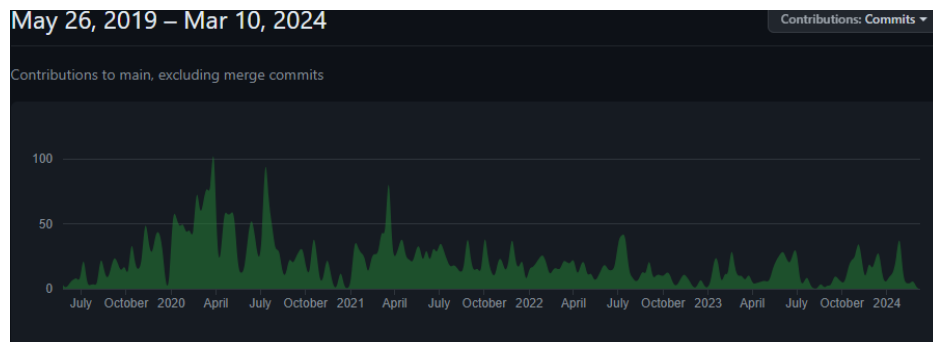


Abbildung 2.35: Stackgres - Contributors Commits

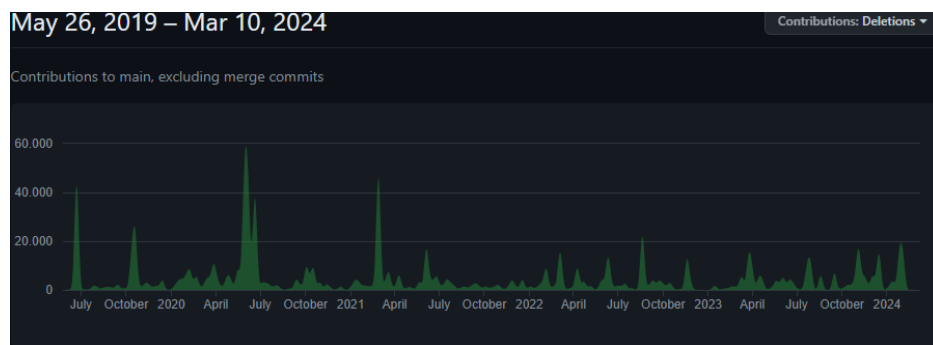


Abbildung 2.36: Stackgres - Contributors Deletations

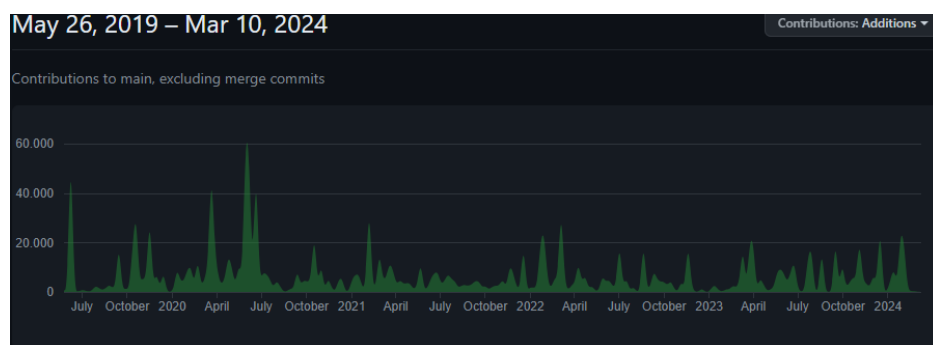


Abbildung 2.37: Stackgres - Contributors Additions

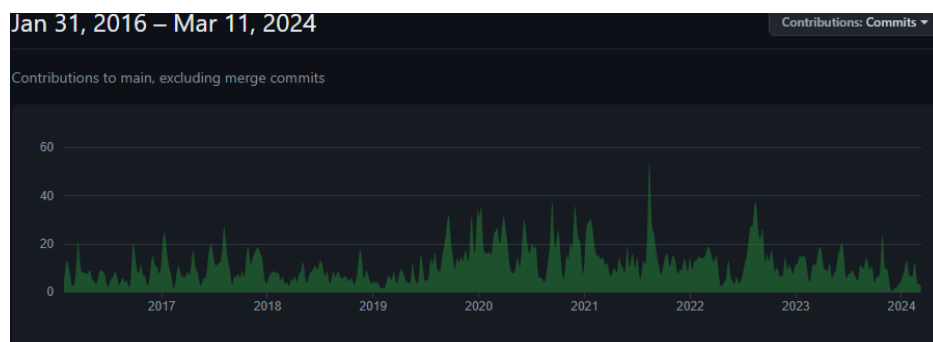


Abbildung 2.38: Citrus - Contributors Commits

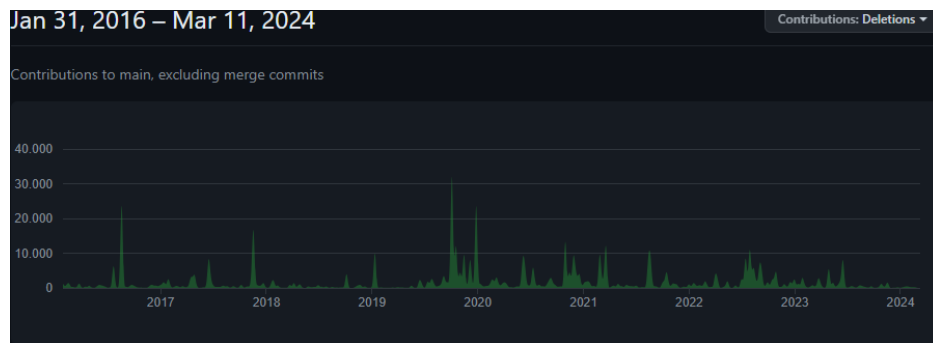


Abbildung 2.39: Citus - Contributors Deletations

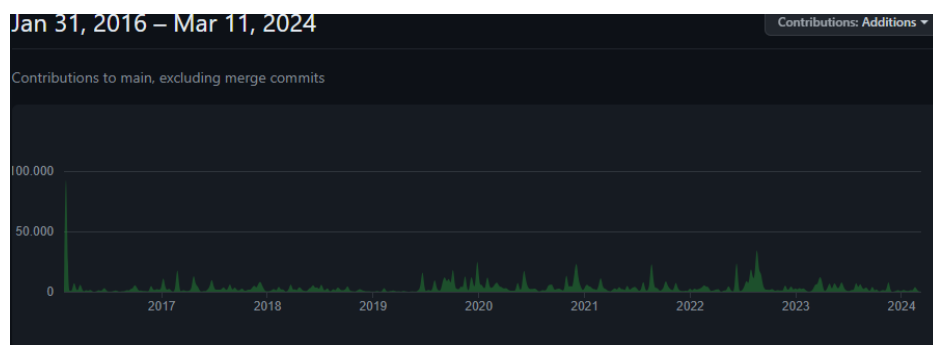


Abbildung 2.40: Citus - Contributors Additions

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:

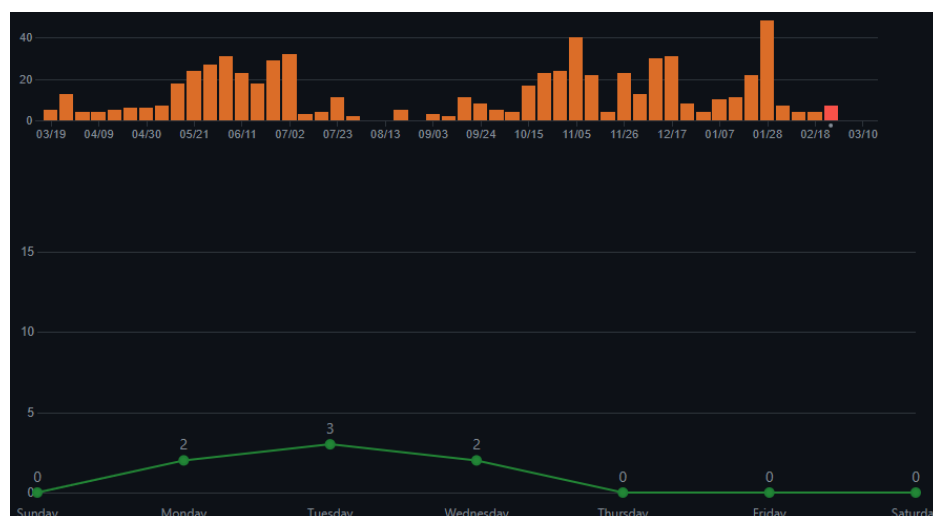


Abbildung 2.41: Stackgres - Commit Activity

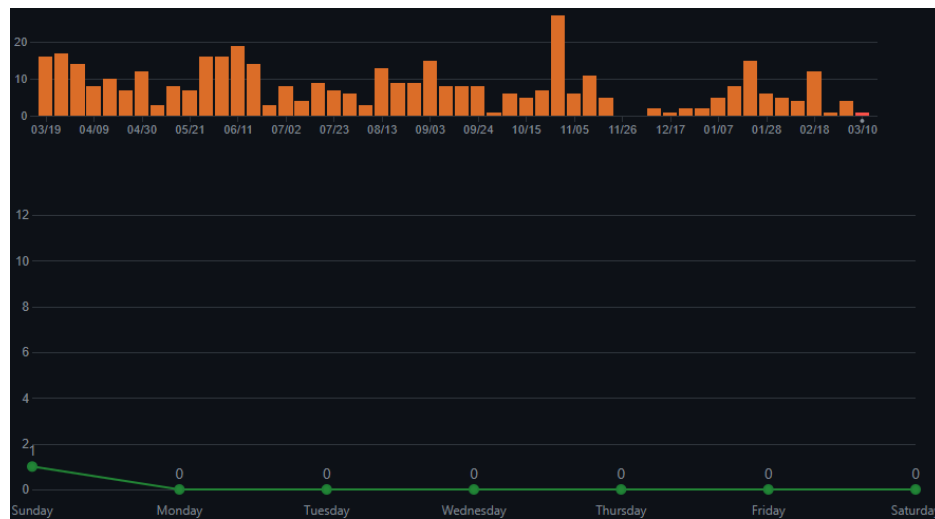


Abbildung 2.42: Citus - Commit Activity

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.

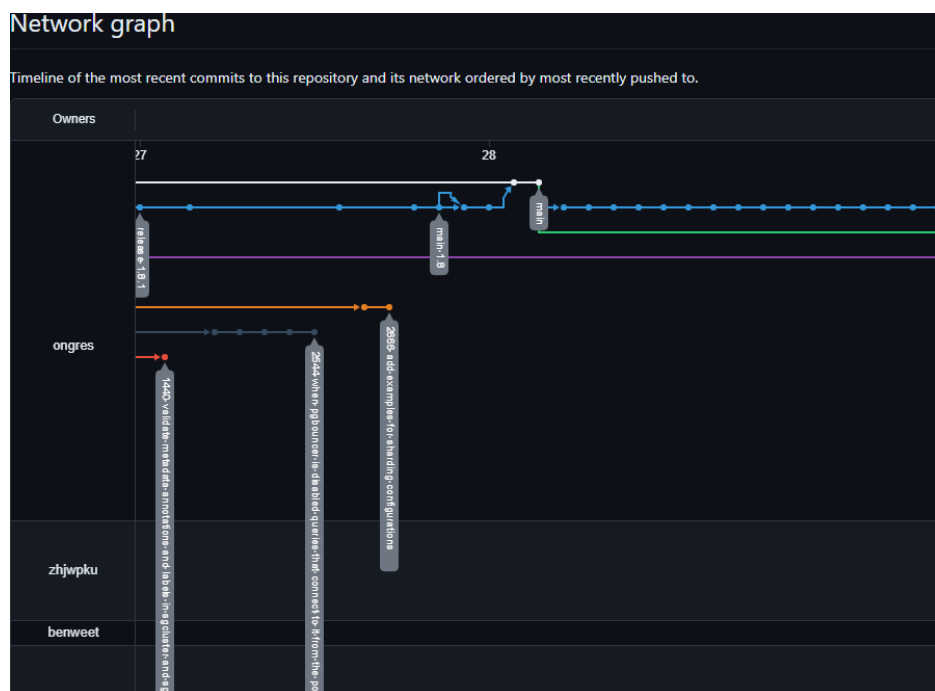


Abbildung 2.43: Stackgres - Network Graph



Abbildung 2.44: Citus - Network Graph

### 2.1.6 Vorauswahl

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

Nr.	Lösung	Status	Begründung
1	KSGR-Lösung	Vorausgeschieden	Hat nur einen Standby / Replika-Node.
2	pgpool-II	Vorausgeschieden	Failover Funktioniert nur bei kleineren Datenmengen wirklich in einer vernünftigen Zeit.
3	pg_auto_failover	Vorausgeschieden	pgpool-II hat kein GitHub-Repository und bietet daher keine vergleichswerte mittels Github Insights. pg_auto_failover würde zwar Citus-Support bieten, allerdings gibt es keine gut dokumentierte Implementation für Kubernetes.
4	CloudNativePG	Vorausgeschieden	Erfüllt daher das Kriterium für die Synergien nicht 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 2.4: Vorauswahl - Ausgeschieden

Entsprechend werden nur noch nachfolgende Lösungen genauer betrachtet:



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

Tabelle 2.5: 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
sks1185	Distributed SQL	Agent	sks1185.ksgr.ch	10.0.20.105
vks0032	Distributed SQL	Virteulle IP	vks0032.ksgr.ch	10.0.20.106

Tabelle 2.6: Evaluationssysteme

#### 2.1.7.1 rke2 - Evaluationsplattform

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

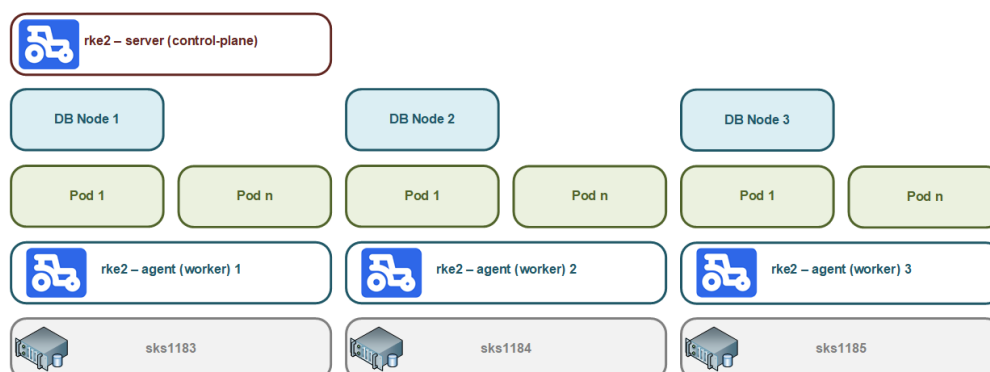


Abbildung 2.45: Evaluationssystem - Distributed SQL / Shards

Die Konfiguration der rke2-Nodes sieht folgendermassen aus:

<b>Kubernetes Runtime</b>	rke2
<b>Container-Environment</b>	containerd
<b>Container Network Interface (CNI)</b>	cilium
<b>Cloud Native Storage (CNS)</b>	local-path-provisioner

Tabelle 2.7: Evaluationssystem - Distributed SQL / Sharding

#### 2.1.7.2 Patroni

#### 2.1.7.3 StackGres - Citus

### 2.1.7.3.1 Architektur

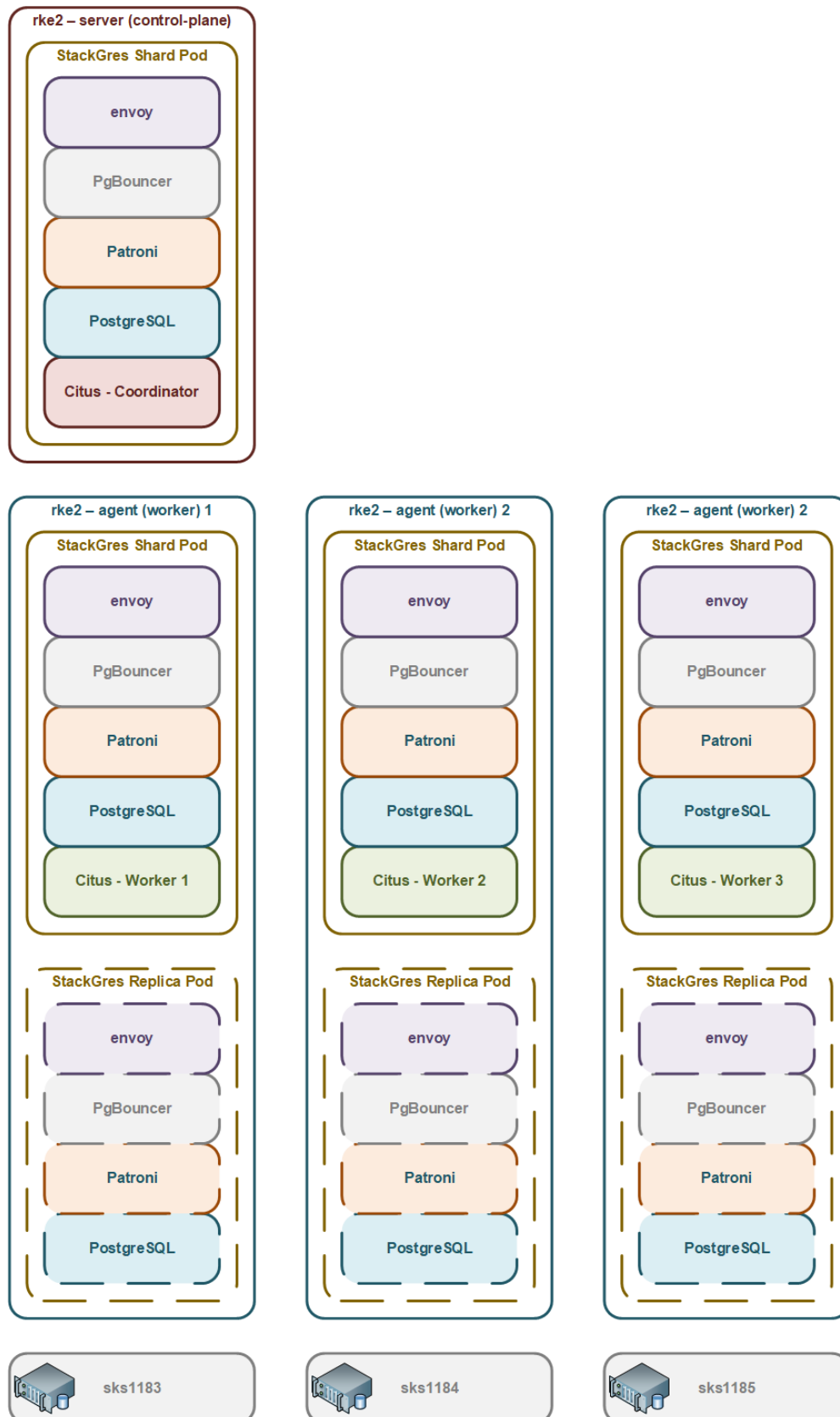


Abbildung 2.46: Stackgres - Citus - Evaluationsarchitektur

#### 2.1.7.4 yugabyteDB

#### 2.1.8 Gegenüberstellung der Lösungen

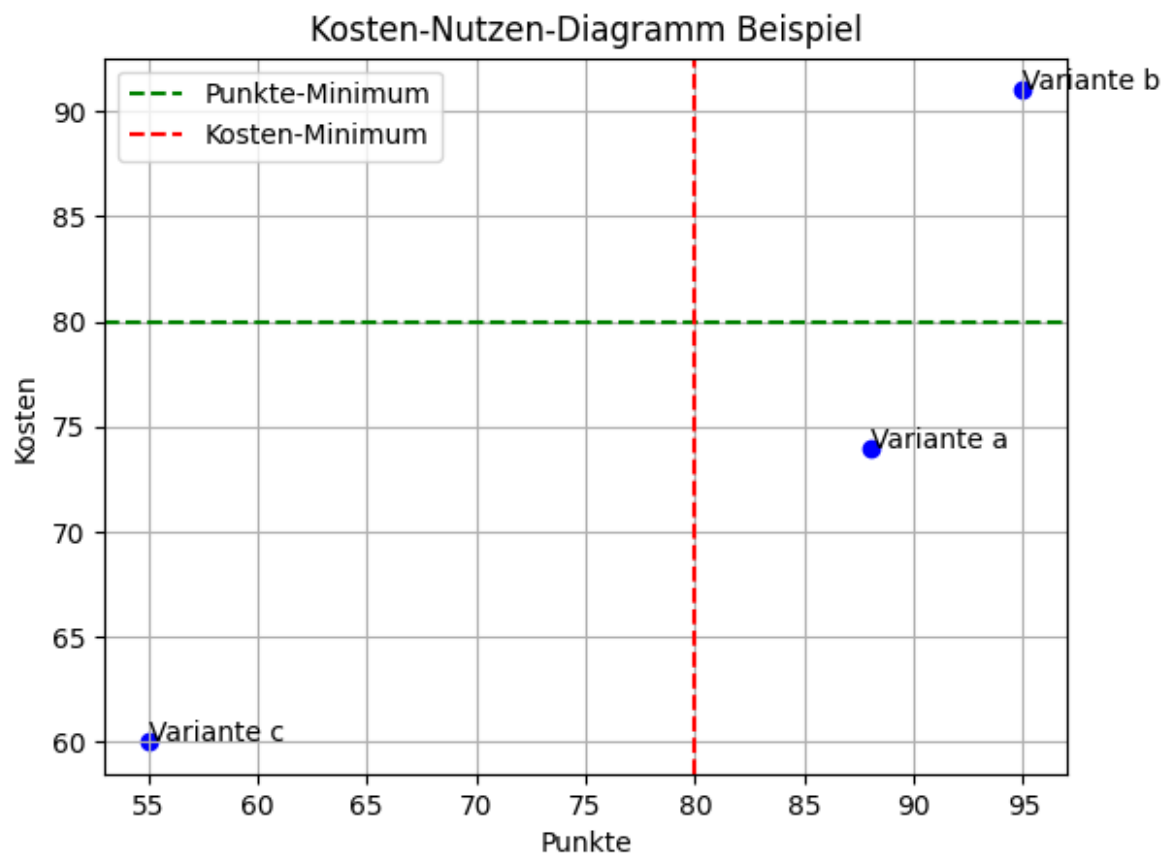


Abbildung 2.47: Kosten-Nutzen-Analyse

#### 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[30]	1
1.2	Wahlkreise Kanton St. Gallen[53]	2
1.3	Spitalregionen / Spitalstrategie Kanton St. Gallen[24]	3
1.4	Organigramm Kantonsspital Graubünden	4
1.5	Organigramm Departement 10 - ICT	5
1.6	Risiken bestehende Lösung	11
1.7	Risiken bestehende Lösung mit Massnahmen	12
1.8	Systemabgrenzung	17
1.9	Projektrisiken	21
1.10	Projektrisiken mit Massnahmen	23
2.1	Monolithische vs. verteilte SQL Systeme	31
2.2	CAP-Theorem	34
2.3	Datenbankskalierung	35
2.4	Präferenzmatrix	39
2.5	Präferenzmatrix - Failover	40
2.6	Präferenzmatrix - Switchover	41
2.7	Präferenzmatrix - Restore	42
2.8	Präferenzmatrix - Replikation	43
2.9	Präferenzmatrix - Sharding	44
2.10	Präferenzmatrix - Quorum	45
2.11	Präferenzmatrix - Management-API	46
2.12	Präferenzmatrix - Backup	47
2.13	Präferenzmatrix - Performance	48
2.14	pg_auto_failover-Architektur - Single Standby	51
2.15	pg_auto_failover-Architektur - Multi-Node Standby	51
2.16	pg_auto_failover-Architektur - Citus	52
2.17	Patroni-Architektur	53
2.18	Patroni - Pulse	54
2.19	Patroni - Code Frequency	54
2.20	Patroni - Community Standards	55
2.21	Patroni - Contributors Commits	55
2.22	Patroni - Contributors Deletations	56
2.23	Patroni - Contributors Additions	56
2.24	Patroni - Commit Activity	56

2.25 Patroni - Network Graph . . . . .	57
2.26 Citus - Coordinator und Workers . . . . .	58
2.27 Citus - Row-Based-Sharding . . . . .	58
2.28 Citus - Schema-Based-Sharding . . . . .	59
2.29 Stackgres - Pulse . . . . .	59
2.30 Citus - Pulse . . . . .	60
2.31 Stackgres - Code Frequency . . . . .	60
2.32 Citus - Code Frequency . . . . .	61
2.33 Stackgres - Community Standards . . . . .	61
2.34 Citus - Community Standards . . . . .	62
2.35 Stackgres - Contributors Commits . . . . .	62
2.36 Stackgres - Contributors Deletations . . . . .	63
2.37 Stackgres - Contributors Additions . . . . .	63
2.38 Citus - Contributors Commits . . . . .	63
2.39 Citus - Contributors Deletations . . . . .	64
2.40 Citus - Contributors Additions . . . . .	64
2.41 Stackgres - Commit Activity . . . . .	64
2.42 Citus - Commit Activity . . . . .	65
2.43 Stackgres - Network Graph . . . . .	65
2.44 Citus - Network Graph . . . . .	66
2.45 Evaluationssystem - Distributed SQL / Shards . . . . .	67
2.46 Stackgres - Citus - Evaluationsarchitektur . . . . .	69
2.47 Kosten-Nutzen-Analyse . . . . .	70



## Tabellenverzeichnis

1.1	Inventarisierte Datenbanksysteme . . . . .	7
1.2	Datenbankinventar . . . . .	8
1.3	Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt . . . . .	8
1.4	Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken . . . . .	10
1.5	Administrative Aufgaben . . . . .	13
1.6	Automatisierung Administrativer Aufgaben . . . . .	14
1.7	Ziele . . . . .	15
1.8	Gegebene Systeme . . . . .	16
1.9	Abhängigkeiten . . . . .	18
1.10	Risiko-Matrix der Diplomarbeit . . . . .	20
1.11	Projektcontrolling . . . . .	25
1.12	Initialer Statusbericht . . . . .	28
1.13	Zweiter Statusbericht . . . . .	29
1.14	Fachgespräche . . . . .	30
2.1	Quorum Beispiele . . . . .	33
2.2	Anforderungskatalog . . . . .	37
2.3	Stakeholder . . . . .	38
2.4	Vorauswahl - Ausgeschieden . . . . .	66
2.5	Vorauswahl - Evaluation . . . . .	67
2.6	Evaluationssysteme . . . . .	67
2.7	Evaluationssystem - Distributed SQL / Sharding . . . . .	68
I	Arbeitsrapport . . . . .	ii
II	Fachgespräche - Protokoll . . . . .	iii
III	Kommentare - Anmerkung . . . . .	iv

## Listings

1	Proxy Settings . . . . .	v
2	Downlaod rke2 server . . . . .	v
3	rke2 server installieren . . . . .	v
4	Downlaod rke2 agent . . . . .	v
5	rke2 agent aktivieren . . . . .	v
6	rke2 server proxy . . . . .	vi
7	rke2 server proxy kopieren . . . . .	vi
8	rke2 server cilium installieren . . . . .	vi
9	rke2 server cilium aktivieren . . . . .	vi
10	rke2 server starten . . . . .	vi
11	iptables entries server . . . . .	vii
12	rke2 server token . . . . .	viii
13	Python LaTeX - zotero - Zotero BibLaTeX Importer . . . . .	viii
14	Python LaTeX - riskmatrix - Risxikomatrizen . . . . .	xiv

## 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. *Architecture Basics — pg\_auto\_failover 2.0 documentation*. <https://pg-auto-failover.readthedocs.io/en/main/architecture.html>.
- [4] unknown author. *Choosing Distribution Column - Citus 12.1 documentation*. [https://docs.citusdata.com/en/v12.1/sharding/data\\_modeling.html#distributed-data-modeling](https://docs.citusdata.com/en/v12.1/sharding/data_modeling.html#distributed-data-modeling).
- [5] unknown author. *Citus Support — pg\_auto\_failover 2.0 documentation*. <https://pg-auto-failover.readthedocs.io/en/main/citus.html>.
- [6] unknown author. *Cluster Management - Citus 12.1 documentation - worker-node-failure*. [https://docs.citusdata.com/en/v12.1/admin\\_guide/cluster\\_management.html#worker-node-failure](https://docs.citusdata.com/en/v12.1/admin_guide/cluster_management.html#worker-node-failure).
- [7] unknown author. *Concepts - Citus 12.1 documentation - row-based-sharding*. [https://docs.citusdata.com/en/v12.1/get\\_started/concepts.html#row-based-sharding](https://docs.citusdata.com/en/v12.1/get_started/concepts.html#row-based-sharding).
- [8] unknown author. *etcd*. <https://etcd.io/>.
- [9] unknown author. *HAProxy Documentation Converter*. <https://docs.haproxy.org/>.
- [10] unknown author. *HAProxy version 2.9.6 - Starter Guide*. <https://docs.haproxy.org/2.9/intro.html#3.2>.
- [11] unknown author. *Multi-node Architectures — pg\_auto\_failover 2.0 documentation*. <https://pg-auto-failover.readthedocs.io/en/main/architecture-multi-standby.html>.
- [12] GitLab B.V. und GitLab Inc. *The DevSecOps Platform | GitLab*. <https://about.gitlab.com/>.
- [13] Alexandre Cassen und Read the Docs. *Introduction — Keepalived 1.2.15 documentation*. <https://keepalived.readthedocs.io/en/latest/introduction.html>. 2017.
- [14] Microsoft Corporation. *Azure SQL-Datenbank – ein verwalteter Clouddatenbankdienst | Microsoft Azure*. <https://azure.microsoft.com/de-de/products/azure-sql/database>. 2023.
- [15] Microsoft Corporation. *Datenbank-Software und Datenbankanwendungen | Microsoft Access*. <https://www.microsoft.com/de-de/microsoft-365/access>. 2023.
- [16] Microsoft Corporation. *Microsoft Data Platform | Microsoft*. <https://www.microsoft.com/de-ch/sql-server>.

- [17] ORACLE CORPORATION. „Oracle (Active) Data Guard 19c“. In: (2019), S. 14.
- [18] Varun Dhawan und data-nerd.blog. *PostgreSQL-Diagnostic-Queries* – data-nerd.blog. <https://data-nerd.blog/2018/12/30/postgresql-diagnostic-queries/>.
- [19] EDB: Open-Source, Enterprise Postgres Database Management. <https://www.enterprisedb.com/>.
- [20] Elektronik-Kompendium.de und Schnabel Schnabel. *SAN - Storage Area Network*. <https://www.elektronik-kompendium.de/sites/net/0906071.htm>. 2023.
- [21] DB-Engines und solidIT consulting & software development gmbh. *DB-Engines Ranking*. <https://db-engines.com/en/ranking>.
- [22] DB-Engines und solidIT consulting & software development gmbh. *relationale Datenbanken - DB-Engines Enzyklopädie*. <https://db-engines.com/de/article/relationale+Datenbanken?ref=RDBMS>.
- [23] The Linux Foundation. *Harbor*. <https://goharbor.io/>. 2023.
- [24] 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>.
- [25] Git. *About - Git*. <https://git-scm.com/about>.
- [26] IBM Deutschland GmbH. *Was ist das CAP-Theorem? | IBM*. <https://www.ibm.com/de-de/topics/cap-theorem>. 2023.
- [27] IBM Deutschland GmbH. *Was ist OLAP? | IBM*. <https://www.ibm.com/de-de/topics/olap>.
- [28] Jedox GmbH. *Was ist OLAP? Online Analytical Processing im Überblick*. <https://www.jedox.com/de/blog/was-ist-olap/>. Section: Knowledge.
- [29] 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>.
- [30] 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>.
- [31] The Pgpool Global Development Group. *What is Pgpool-II?* <https://www.pgpool.net/docs/44/en/html/intro-what-is.html>. 2023.
- [32] The PostgreSQL Global Development Group. *25.1. Routine Vacuuming*. <https://www.postgresql.org/docs/16/routine-vacuuming.html>. 2023.
- [33] The PostgreSQL Global Development Group. *27.1. Comparison of Different Solutions*. <https://www.postgresql.org/docs/16/different-replication-solutions.html>. 2023.

- [34] The PostgreSQL Global Development Group. *pgbench*. <https://www.postgresql.org/docs/16/pgbench.html>. 2023.
- [35] Inc. HashiCorp. *Terraform by HashiCorp*. <https://www.terraform.io/>.
- [36] Patrick Hunt, Mahadev Konar, Flavio P Junqueira und Benjamin Reed. „ZooKeeper: Wait-free coordination for Internet-scale systems“. In: (2010).
- [37] Splunk Inc. *Splunk | Der Schlüssel zu einem resilienten Unternehmen*. [https://www.splunk.com/de\\_de](https://www.splunk.com/de_de). 2023.
- [38] Sebastian Insausti. *Scaling PostgreSQL for Large Amounts of Data*. <https://severalnines.com/blog/scaling-postgresql-large-amounts-data/>. 2019.
- [39] Shiv Iyer und MinervaDB. *PostgreSQL DBA Daily Checklist*. <https://minervadb.xyz/postgresql-dba-d>. 2020.
- [40] Unmesh Joshi. *Quorum*. <https://martinfowler.com/articles/patterns-of-distributed-systems/quorum.html>. 2020.
- [41] Martin Keen und IBM Deutschland GmbH. *IBM Db2*. <https://www.ibm.com/de-de/products/db2>.
- [42] Pasha Kostohrys. *Database replication — an overview*. <https://medium.com/@pkostohrys/database-replication-an-overview-f7ade110477>. 2020.
- [43] Anatoli Kreyman. *Was ist eigentlich Splunk?* <https://www.kreyman.de/index.php/splunk/76-was-ist-eigentlich-splunk-big-data-platform-monitoring-security>.
- [44] 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>.
- [45] Red Hat Limited. *Was ist Ansible?* <https://www.redhat.com/de/technologies/management/ansible/what-is-ansible>.
- [46] 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>.
- [47] Switzerland Linuxfabrik GmbH Zurich. *Keepalived — Open Source Admin-Handbuch der Linuxfabrik*. <https://docs.linuxfabrik.ch/software/keepalived.html>. 2023.
- [48] Nico Litzel, Stefan Luber und Vogel IT-Medien GmbH. *Was ist Elasticsearch?* <https://www.bigdata-insider.de/was-ist-elasticsearch-a-939625/>. 2020.
- [49] SRA OSS LLC. *pgpool Wiki*. [https://www.pgpool.net/mediawiki/index.php/Main\\_Page](https://www.pgpool.net/mediawiki/index.php/Main_Page). 2023.
- [50] Hewlett Packard Enterprise Development LP. *Was ist SAN-Speicher? | Glossar*. <https://www.hpe.com/ch/de/what-is/san-storage.html>.
- [51] Diego Ongaro. „Consensus: Bridging Theory and Practice“. In: (2014).

- [52] Bruno Queirós und LinkedIn Ireland Unlimited Company. *Postgresql replication with automatic failover*. <https://www.linkedin.com/pulse/postgresql-replication-automatic-failover-bruno-queiros>. 2020.
- [53] 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>.
- [54] Ed Reckers und SnapLogic Inc. *Was ist die Snowflake-Datenplattform?* <https://www.snaplogic.com/de/blog/snowflake-data-platform>. 2023.
- [55] IONOS SE. *Apache Cassandra: Verteilte Verwaltung großer Datenbanken*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/apache-cassandra-vorgestellt/>. 2021.
- [56] IONOS SE. *Datenbankmanagementsystem (DBMS) erklärt*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/datenbankmanagementsystem-dbms-erklart/>. 2020.
- [57] IONOS SE. *MongoDB – die flexible und skalierbare NoSQL-Datenbank*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/mongodb-vorstellung-und-vergleich-mit-mysql/>. 2019.
- [58] IONOS SE. *SQLite: Die bekannte Programmbibliothek im Detail vorgestellt*. <https://www.ionos.de/digitalguide/websites/web-entwicklung/sqlite/>. 2023.
- [59] IONOS SE. *Terraform*. <https://www.ionos.de/digitalguide/server/tools/was-ist-terraform/>. 2020.
- [60] IONOS SE. *Was ist Redis? Die Datenbank vorgestellt*. <https://www.ionos.de/digitalguide/hosting/hosting-technik/was-ist-redis/>. 2020.
- [61] IONOS SE. *Was ist SIEM (Security Information and Event Management)?* <https://www.ionos.de/digitalguide/server/sicherheit/was-ist-siem/>. 2020.
- [62] Sami Ahmed Siddiqui. *Distributed SQL 101*. <https://www.yugabyte.com/distributed-sql/>.
- [63] Inc. Snowflake. *Datenbanken, Tabellen und Ansichten – Überblick | Snowflake Documentation*. <https://docs.snowflake.com/de/guides-overview-db>.
- [64] Thomas-Krenn.AG. *Git Grundlagen – Thomas-Krenn-Wiki*. [https://www.thomas-krenn.com/de/wiki/Git\\_Grundlagen](https://www.thomas-krenn.com/de/wiki/Git_Grundlagen).
- [65] Rainer Züst. „Einstieg ins Systems Engineering“. In: (2002).

## Glossar

**Ansible** Ansible ist ein Open-Source Automatisierungstool zur Provisionierung, Konfiguration, Deployment und Orchestrierung. Ansible verbindet sich auf die Zielgeräte und führt dort die hinterlegten Module aus. Oft werden die verschiedenen Aufgaben in einem Skript, in einem sogenannten Playbook geschrieben werden[45].. 16

**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[32].. 14, 15

**Cassandra** Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[55] wurde.. 7

**CI/CD** Continuous Integration/Continuous Delivery bedeutet, dass Anpassungen kontinuierlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[46].. 4

**DBMS** Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[56].. 4

**Debian** Debian gehört neben Slackware Linux zu den ältesten Linux Distributionen die noch immer gepflegt und eingesetzt werden. Sie wurde im August 1993 gestartet und brachte im Laufe der Zeit einige der beliebtesten Distributionen wie Ubuntu hervor.. 16

**Elasticsearch** Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[48].. 7

**etcd** etcd ist [8]. 52

**Failover** In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary umgeplant geschaltet.. 15, 31, 32, 49, 82

**Foreman** Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basiert der Red Hat Satellite auf Foreman. 16, 20

**Git** Git ist eine Versionsierungsssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral sondern dezentral organisiert und arbeiten daher mit Working Copies von Repositories[25, 64].. 82

**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[12].. 15, 49

**HAProxy** HAProxy [10]. 50, 52

**Harbor** Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[23]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 15, 49

**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.. 4, 5, 8, 20

**IBM DB2** IBM DB2 ist eine Relationale Datenbank[41] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7, 34

**keepalived** keepalived nutzt VRRP um eine leichtgewichtige Lösung für ein HA-Failover zu realisieren. keepalived benötigt dazu keinen dritten Node, also einen Quorum-Node. Wenn die definierte sekundärseite keine Antwort mehr von der primären Seite nach einer definierten Anzahl versuchen in einem bestimmten Interval mehr bekommt, oder ein per Skript definiertes Event auf der primären Seite eintrifft, wird ein Failover auf die sekundäre Seite ausgeführt. Je nach Konfiguration kann der Restore auf die primäre Seite eingeleitet werden wenn diese wieder verfügbar ist oder der Restore unterbunden werden[47, 13].. 49

**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.. 4, 8, 16, 82

**Linux** Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und lose vom UNIX Derivat MINIX inspiriert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 83

**MariaDB** MariaDB ist ein MySQL Fork des ehemaligen MySQL Mitbegründers Michael Widenius, wobei sich der Name Maria aus dem Vornamen einer seiner Töchter ableitet. Nach dem Fork 2009 blieb MariaDB für eine Zeitlang sehr ähnlich mit MySQL und behielt ein ähnliches Versionierungsschema bei. Dies änderte sich 2012 wo dann direkt mit der Version 10 weitergefahren wurde. Beide Datenbanken entfernen sich im Lauf der Zeit immer mehr voneinander und sind nicht mehr in jedem Fall kompatibel oder beliebig austauschbar. Auf



den Linux Distributionen tritt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 8

**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[14].. 7

**Microsoft Access** Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[15].. 7

**Microsoft SQL Server** MS SQL Server ist das RDBMS von Microsoft[16]. Neben 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, 83

**MongoDB** MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zum ersten Mal 2007 veröffentlicht wurde[57].. 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 Microsystems 2008 MySQL übernahm, hielt sich die Option frei, bei einem Kauf von Sun Microsystems durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystems 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, 8

**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, 81, 83, 85

**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[28, 27]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 4, 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 kommt auf den Oracle Eigenen Appliances ODA und Exadata zum Einsatz. Für den Zweck als DB Plattform kann ein für Oracle Datenbanken optimierter Kernel verwendet werden. Zu

Oracle Linux kann ein kostenpflichtiger Support bezogen werden, allerdings ist die Distribution anders als RHEL auch ohne Lizenz erhältlich.. 16

**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 erschienenen Optimizer und mit der Version 9i 2001 erschienenen Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangenheit ermöglichen) konnte Oracle sich stark von der Konkurrenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Performance oder grosse Datenmengen geht.. 5, 7, 8, 34, 83

**PKI** . 4

**PostgreSQL** Die OpenSource Datenbank PostgreSQL wurde in Form von POSTGRES zum ersten Mal 1986 von der University of California at Berkeley veröffentlicht. und zählt zu den beliebtesten OpenSource Datenbanken. Zudem besteht in vielen Bereichen eine gewisse Ähnlichkeit zu Oracles Oracle Database.. 5, 7, 8, 9, 13, 34, 48, 49, 50, 57, viii

**PostgreSQL HA Cluster** Der HA Cluster des PostgreSQL Clusters. 15

**PostgreSQL Cluster** Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wie Oracle.. 3, 14, 15, 49, 84, viii

**PRTG** Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum ersten Mal veröffentlicht und war ebenfalls als Netzwerkmonitoring System konzipiert. Wie bei Zabbix lässt sich heute damit ebenfalls fast jedes IT-System überwachen. Reichen die zahlreichen vorhandenen Standard Sensoren nicht, können eigene Sensoren geschrieben werden. PRTG ist nicht Open-Source, man bezahlt anhand gewisser Sensor Packages.. 4, 5, 14, 16

**Quorum** In verteilten Systemen resp. Cluster muss sichergestellt werden, dass bei einem Ausfall oder einer 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 Mehrheit aller Nodes vereint. Daraus ergeben sich bestimmte Größen, mit 5 Nodes braucht es 3 Nodes um aktiv zu bleiben und mit 3 Nodes deren 2. Bei diesen Konstellationen wird daher darauf geachtet, eine ungerade Anzahl Nodes im Cluster zu halten um keine Partition-Situation zu provozieren. Im Kapitel [Unterunterabschnitt 2.1.1.3](#) wird genauer auf die Mechanik eines Quorums eingegangen. . 49, 82

**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[22].. 4

**RedHat Enterprise Linux (RHEL)** RHEL wurde in seiner Ursprünglichen Form Red Hat Linux (RHL) bis in den Oktober 1994 zurück, wobei die erste Version von RHEL wie es heute existiert im Jahr 2002 erfolgte. RHEL ist auf lange Wartungszyklen von fünf Jahren und grosskunden ausgelegt. Ohne entsprechenden Supportvertrag kann keine ISO-Datei bezogen werden. Somit hebt sich RHEL stark von anderen Linux Distributionen ab.. 16

**Redis** Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[60]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7

**Rocky Linux** Rocky Linux basierte auf der offen zugänglichen Linux Distribution CentOS welche RHEL Binärkompatibel war und gilt als inoffizieller Nachfolger von CentOS.. 16

**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 LUNs, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[20, 50, 29]. 4, 5, 16, 20

**SIEM** Ein sammelt Daten aus verschiedenen Netzwerkkomponenten oder Geräten von Agents oder Logs. Diese Daten werden permanent analysiert und mit einem definierten Regelwerk gegengeprüft. Ziel ist es, verdächtige Events zu erkennen und einem Angriff zuvorzukommen oder ihn möglichst früh zu unterbinden[61].. 4, 16

**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[63, 54]. 7

**Split-brain** Im Kapitel ?? werden die Ursachen und Folgen eines Split-brains genauer besprochen. . 32, 84

**Splunk** Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[37, 43]. . 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[58].. 7

**Switchover** In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geschwitched.. 15

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

4

**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[59, 35].. 16

**UNIX** Die erste Version von UNIX wurde im Jahr 1969 in den Bell Labs entwickelt und übernahm viele Komponenten aus dem gescheiterten Multics-Projekt. Aus dem Ursprünglichen UNIX entstanden im Laufe der Zeit viele offene und Proprioritäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 4

**VRRP** VRRP . 4, 82

**Zabbix** Das 2001 veröffentlichte Open-Source Monitoring System Zabbix gilt zwar als Netzwerk-Monitoring System, allerdings kann heute nahezu jedes IT-System damit überwacht werden. Zabbix speichert die Metriken und nicht die Auswertungen, das heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert werden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge Abschliessen.. 8, 16

### 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                    Statusbericht**

**I.I**

II                      Arbeitsrapport

Datum	Von	Bis	Dauer [h]	Phase	Subphase	Tätigkeit	Bemerkung	Schwierigkeit	Lösungen
21.02.2024	15:00	16:00	1.0	Evaluation	Anordnungskatalog	Anordnungskatalog erarbeiten			
22.02.2024	16:00	17:30	1.5	Evaluation	Anordnungskatalog	Anordnungskatalog erarbeiten			
27.02.2024	10:00	11:30	1.5	Dokumentation	-	Dokumentation erweitern			
27.02.2024	13:00	16:00	3.0	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 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 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	Anordnungskatalog	Anordnungskatalog 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 erweitern			
11.03.2024	16:45	17:30	0.5	Dokumentation	-	Dokumentation Stakeholder			
13.03.2024	17:45	19:45	2.0	Evaluation	Analyse PostgreSQL HA Cluster Lösungen	Stackres 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	-	Citus row-based-sharding Dokumentieren			
16.03.2024	17:45	18:30	0.8	Dokumentation	-	Projektkontrollieren Arbeiten			
17.03.2024	14:45	16:30	1.8	Dokumentation	-	Zweiter Statusbericht verfassen			

TABLE I: Arbeitsrapport





III 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		Norman Süsstrunk	-	Michael Graber	- Muss das Protokoll des Fachgesprächs jeweils Zeilnah 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	

TABLE II: Fachgespräche - Protokoll



#### IV Kommentare / Anmerkungen

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

Woche	Beschreibung / Event / Problem
	Vier ganze Tage war ich in Thalwil für die Oracle Multitenant-Schulung für das ExaCC Projekt (Ablösung HP-UX).
KW10	Am Freitag 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.

TABLE III: Kommentare - Anmerkung

## V rke2

### V.I Vorbereitung

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

```
1 sudo nano /etc/profile.d/proxy.sh
2
3 export https_proxy=http://sproxy.sivc.first-it.ch:8080
4 export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
5 export http_proxy=http://sproxy.sivc.first-it.ch:8080
6 export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
7 export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
8 export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
9
10 source /etc/profile.d/proxy.sh
```

Listing 1: Proxy Settings

### V.II Installation

#### V.II.I server

Es gibt kein apt-Package. Daher muss zuerst das tarball-Package heruntergeladen werden:

```
1 sudo curl -sL https://get.rke2.io | sh -
```

Listing 2: Downlaod rke2 server

Anschliessend muss das Package installiert werden:

```
1 sudo curl -sL https://get.rke2.io | sh -
```

Listing 3: rke2 server installieren

#### V.II.II agents

Der Agent muss direkt heruntergeladen werden:

```
1 curl -sL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
```

Listing 4: Downlaod rke2 agent

Anschliessend muss der Dienst aktiviert werden:

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

Listing 5: rke2 agent aktivieren

### V.III Cluster Konfiguration

#### V.III.I server

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

```
1 nano /etc/default/rke2-server
2 HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
3 HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
4 NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
5
6 CONTAINERD_HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
7 CONTAINERD_HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
8 CONTAINERD_NO_PROXY=localhost
   ,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
```

Listing 6: rke2 server proxy

Dieses File muss entsprechend in das Homeverzeichnis gespeichert werden:

```
1 mkdir /home/itgramic/.kube
2 cp /etc/rancher/rke2/rke2.yaml /home/itgramic/.kube/config
```

Listing 7: rke2 server proxy kopieren

Für den Netzwerkteil muss nun Cilium installiert werden:

```
1 nano /var/lib/rancher/rke2/server/manifests/rke2-cilium-config.yaml
2 ---
3 apiVersion: helm.cattle.io/v1
4 kind: HelmChartConfig
5 metadata:
6   name: rke2-cilium
7   namespace: kube-system
8 spec:
9   valuesContent: |-
10     eni:
11       enabled: true
```

Listing 8: rke2 server cilium installieren

Cilium muss nun aktiviert werden:

```
1 /var/lib/rancher/rke2/bin/kubectl apply -f /var/lib/rancher/rke2/server/manifests/
   rke2-cilium-config.yaml
```

Listing 9: 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:

```
1 /var/lib/rancher/rke2/bin/kubectl cluster-info --kubeconfig /etc/rancher/rke2/rke2
   .yaml
2 nano /etc/rancher/rke2/config.yaml
```

```

3 cni:
4 - cilium
5
6 systemctl restart rke2-server.service

```

Listing 10: rke2 server starten

Entsprechend muss die Firewall gesetzt werden:

```

1 nano /etc/iptables/rules.v4
2
3 # Generated by iptables-save v1.8.9 (nf_tables)
4 *filter
5 :INPUT DROP [0:0]
6 :FORWARD ACCEPT [0:0]
7 :OUTPUT ACCEPT [0:0]
8 -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
9 -A INPUT -p udp -m udp --sport 53 -j ACCEPT
10 -A INPUT -p icmp -j ACCEPT
11 -A INPUT -i lo -j ACCEPT
12 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
13 -A INPUT -s 10.0.9.115/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.115" -j ACCEPT
14 -A INPUT -s 10.0.9.76/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.76" -j ACCEPT
15 -A INPUT -s 10.0.36.147/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.36.147" -j ACCEPT
16 -A INPUT -s 10.0.9.35/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.35" -j ACCEPT
17 -A INPUT -s 10.0.9.37/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.37" -j ACCEPT
18 -A INPUT -s 10.0.9.74/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.74" -j ACCEPT
19 -A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.75" -j ACCEPT
20 -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
21 -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
22 -A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
23 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6443 -j ACCEPT
24 -A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 9345 -j ACCEPT
25 COMMIT
26 # Completed
27
28 systemctl restart iptables

```

Listing 11: iptables entries server

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

```

1 cni:
2 - cilium
3 token: <password safe>

```

Listing 12: rke2 server token

V.III.II agents

VI pgpool-II

VI.I PostgreSQL Cluster Installation

## PostgreSQL Package Repository in Debian einbinden

VI.II yugabyteDB

VI.II.I minikube

VI.II.II yugabyteDB Konfiguration

VII Stackgres mit Citus

VIII zotero.py

```

1 import json
2 import pybtex
3 import requests
4 import os
5 from pybtex.database import BibliographyData, Entry, Person
6 from dateutil.parser import parse
7 import math
8
9 def load_configuration():
10     zotero_bibtex_config = dict()
11     zotero_conf_filename = 'zotero_bibtex_configuration.json'
12     zotero_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
configuration')
13     # zotero_conf_dir = os.path.join(os.getcwd(), 'src', 'content')
14     json_path = os.path.join(zotero_conf_dir, zotero_conf_filename)
15
16     with open(json_path) as json_string:
17         zotero_bibtex_config = json.load(json_string)
18
19     return zotero_bibtex_config
20 def downlaod_zotero_datas(URL, API_KEY):
21     zotero_result = list
22     response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
23     response = response.json()

```

```

24     zotero_raw = json.dumps(response, ensure_ascii=False) # json.loads(response)
25     zotero_result = json.loads(zotero_raw)
26     return zotero_result
27
28 def get_data(zotero_bibtex_config):
29     # result_limit = 100
30     # access_type = 'groups'
31     # zotero_access_id = '5245833'
32     # collection_id = 'USSFDCEH'
33     result_limit = int(zotero_bibtex_config.get('result_limit'))
34     access_type = zotero_bibtex_config.get('access_type')
35     zotero_access_id = zotero_bibtex_config.get('zotero_access_id')
36     collection_id = zotero_bibtex_config.get('collection_id')
37     API_KEY = zotero_bibtex_config.get('api_key')
38     zotero_data = list()
39     URL = 'https://api.zotero.org/' + str(access_type) + '/' + str(
40         zotero_access_id) + '/collections/' + str(
41         collection_id) + '/items?limit=1?format=json?sort=dateAdded?direction=asc'
42
43     # API_KEY = '6Xgb3XhGjQXwA8NuZgu3bw3s'
44     response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})
45
46     header_dict = response.headers
47     total_elems = int(header_dict.get('Total-Results'), 0)
48
49     if total_elems < result_limit:
50         URL_ALL_ITEMS = 'https://api.zotero.org/' + str(access_type) + '/' + str(
51             zotero_access_id) + '/collections/' + str(collection_id) + '/items?
52         limit=' + str(
53             result_limit) + '?format=json?sort=dateAdded?direction=asc'
54         zotero_result = download_zotero_datas(URL_ALL_ITEMS, API_KEY)
55
56         zotero_data.extend(zotero_result)
57     else:
58         runs = int(math.ceil(total_elems / result_limit))
59         index = 0
60         start_index = 0
61         while index < runs:
62             URL_Separated = 'https://api.zotero.org/' + str(access_type) + '/' +
63             str(
64                 zotero_access_id) + '/collections/' + str(collection_id) + '/items
65             ?limit=' + str(
66                 result_limit) + '?format=json?sort=dateAdded?direction=asc' + '&
67             start=' + str(start_index)
68             zotero_result = download_zotero_datas(URL_Separated, API_KEY)
69
70             zotero_data.extend(zotero_result)

```

```

67         start_index += result_limit
68         index += 1
69
70     return zotero_data
71
72 def convert_to_datetime(input_str, parserinfo=None):
73     return parse(input_str, parserinfo=parserinfo)
74 def get_dates(date, bibtex_item_type, bibtex_month_attributes):
75     dated_date = convert_to_datetime(date)
76     return_value = dict()
77     if bibtex_item_type in bibtex_month_attributes:
78         year = dated_date.year
79         month = dated_date.month
80         return_value = {'year': year, 'month': month}
81     else:
82         year = dated_date.year
83         return_value = {'year': year}
84
85     return return_value
86
87 def split_creators(creators):
88     if creators != []:
89
90         creatorlist = ''
91         for index, creator in enumerate(creators):
92             type = creator.get('creatorType')
93             firstname = creator.get('firstName')
94             lastname = creator.get('lastName')
95             name = creator.get('name')
96             if type == 'author':
97
98                 if name and not (firstname or lastname):
99                     creatorlist = creatorlist + name
100                 if index != len(creators) - 1:
101                     creatorlist = creatorlist + ' and '
102                 else:
103                     creatorlist = creatorlist + lastname + ', ' + firstname
104                 if index != len(creators) - 1:
105                     creatorlist = creatorlist + ' and '
106             else:
107                 creatorlist = 'unknown author'
108
109     bib_entry = 'author=' + '\"' + creatorlist + '\"'
110
111     return bib_entry
112
113
114 def write_bibliography(zotero_data, zotero_bibtex_config):

```



```

115 # file_json = 'keystore.json'
116 file_json = zotero_bibtex_config.get('keystore_file')
117 keystore_path = zotero_bibtex_config.get('keystore_filepath')
118 # tex_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
configuration')
119 tex_dir = os.path.join(os.path.dirname(os.getcwd()), keystore_path)
120 # tex_dir = os.path.join(os.getcwd(), 'src', 'content')
121 json_path = os.path.join(tex_dir, file_json)
122
123 with open(json_path) as json_string:
124     zotero_bibtex_keys = json.load(json_string)
125
126 zotero_bibtex_keys_specials = {
127     'thesis': {'phdthesis': ['dissertation', 'phd', 'doctorial', 'doctor', '
doktor', 'doktorarbeit'],
128               'masterthesis': ['ma', 'master', 'masters']}
129 }
130 zotero_bibtex_attributes_special = {
131     'date': 'get_dates',
132     'creators': 'split_creators'
133 }
134 bibtex_month_attributes = ['booklet', 'mastersthesis', 'phdthesis', '
techreport']
135 # Bibliography
136 # tex_dir = os.path.join(os.path.dirname(os.getcwd()), 'source')
137 bibtex_path = zotero_bibtex_config.get('bibtex_filepath')
138 tex_dir = os.path.join(os.path.dirname(os.getcwd()), bibtex_path)
139 # tex_dir = os.path.join(os.getcwd(), 'src', 'content')
140 # file_name = 'Datenbank_Projektauftrag_Michael_Graber.bib'
141 file_name = zotero_bibtex_config.get('bibtex_filename')
142
143 file_path = os.path.join(tex_dir, file_name)
144
145 # bib_datas = BibliographyData()
146 listKeys = list()
147 bib_data = ''
148 for zotero_items in zotero_data:
149     biblio_item = zotero_items.get('data')
150     itemkeys = biblio_item.keys()
151     listKeys.extend(biblio_item.keys())
152     zotero_item_key = biblio_item.get('key')
153     zotero_item_title = biblio_item.get('title')
154     zotero_item_nameofact = biblio_item.get('nameOfAct')
155     zotero_item_nameofcase = biblio_item.get('caseName')
156     zotero_item_subject = biblio_item.get('subject')
157     zotero_item_type = biblio_item.get('itemType')
158
159     # some item types have no titles

```

```

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

```

```

203     bibtex_item_attribute_extra = ''
204
205     # Special Cases
206     if bibtex_item_attribute == 'SPECIALCHECK' and zotero_item_value not
in ['', None]:
207         bibtex_special_attribute = zotero_bibtex_attributes_special.get(
item_attribute)
208
209         match bibtex_special_attribute:
210             case 'get_dates':
211                 zotero_item_value = get_dates(zotero_item_value,
bibtex_item_key, bibtex_month_attributes)
212                 if zotero_item_value.get('month'):
213                     zotero_item_value_extra = zotero_item_value.get('month
')
214                     bibtex_item_attribute_extra = 'month'
215
216                     zotero_item_value = zotero_item_value.get('year')
217                     bibtex_item_attribute = 'year'
218             case 'split_creators':
219                 authors = split_creators(zotero_item_value)
220                 entryset = entryset + authors
221             elif bibtex_item_attribute == 'howpublished':
222                 if zotero_item_value not in ['', None, []]:
223                     zotero_item_value = '\\url{' + zotero_item_value + '}'
224
225                 if bibtex_item_attribute not in ['', 'None', 'author', 'SPECIALCHECK']
and zotero_item_value not in ['', None, []]:
226                     if zotero_item_value_extra:
227
228                         if type(zotero_item_value_extra) == "string":
229                             entryset = entryset + str(bibtex_item_attribute_extra) + '
=\\' + str(zotero_item_value_extra) + '\\\"
230                         else:
231                             entryset = entryset + str(bibtex_item_attribute_extra) + '
=' + str(zotero_item_value_extra)
232
233                         if index != len(item_attributes) - 1:
234                             entryset = entryset + ',\\n'
235                         else:
236                             entryset = entryset + '\\n'
237
238                 if type(zotero_item_value) == str and not zotero_item_value.
isnumeric():
239                     entryset = entryset + str(bibtex_item_attribute) + '=\\' + str
(zotero_item_value) + '\\\"
240                 else:

```

```

241         entryset = entryset + str(bibtex_item_attribute) + '=' + str(
            zotero_item_value)
242
243         if index != len(item_attributes) - 1:
244             entryset = entryset + ',\n'
245         else:
246             entryset = entryset + '\n'
247
248         # create the Entry
249         entry = '@' + bibtex_item_key + '{' + zotero_item_key + ',\n'
250         entry = entry + entryset + '}'
251         bib_data = bib_data + '\n' + entry
252
253         # parse String to pybtex.database Object
254         # bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex",
            encoding='ISO-8859-1')
255         bib_datas = pybtex.database.parse_string(bib_data, bib_format="bibtex",
            encoding='Utf-8')
256         # Save pybtex.database to file
257         # BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding
            ='ISO-8859-1')
258         BibliographyData.to_file(bib_datas, file_path, bib_format="bibtex", encoding='
            utf-8')
259
260
261 zotero_bibtex_config = load_configuration()
262 zotero_data = get_data(zotero_bibtex_config)
263 write_bibliography(zotero_data, zotero_bibtex_config)

```

Listing 13: Python LaTeX - zotero - Zotero BibLaTeX Importer

## IX riskmatrix.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import pip as pd
4 import os
5 import csv
6 import pandas as pd
7
8 def riskmatrix(risk, conf, matrix):
9     # get the risk datas
10    risk_conf = conf.get(risk)
11    startpath = risk_conf.get('startpath')
12    destination = risk_conf.get('destination')
13    imagename = risk_conf.get('imagename')
14    datafilename = risk_conf.get('datafilename')

```

```

15 itemname = risk_conf.get('itemname')
16 x_axis_title = risk_conf.get('x-axis-title')
17 y_axis_title = risk_conf.get('y-axis-title')
18 title = risk_conf.get('title')
19 bubble_standard_size = int(risk_conf.get('bubble-standard-size'))
20
21 if startpath == 'homedir':
22     directory = os.path.join(os.getcwd(), destination)
23 else: # parentdir
24     directory = os.path.join(os.path.dirname(os.getcwd()), destination)
25
26 print(directory)
27
28 # get the Datas as dict
29 data_path = os.path.join(directory, datafilename)
30 image_path = os.path.join(directory, imagename)
31
32 # load datas from csv into dict
33 with open(data_path) as f:
34     csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]
35
36 (_, *header), *data = csv_list
37 datas = {}
38 for row in data:
39     key, *values = row
40     datas[key] = {key: value for key, value in zip(header, values)}
41
42 # fig_dir = os.path.join(os.path.dirname(os.getcwd()), 'src', 'source')
43 fig = plt.figure()
44 plt.subplots_adjust(wspace=0, hspace=0)
45 plt.xticks([])
46 plt.yticks([])
47 plt.xlim(0, 5)
48 plt.ylim(0, 5)
49 plt.xlabel(x_axis_title)
50 plt.ylabel(y_axis_title)
51 plt.title(title)
52
53 #This example is for a 5 * 5 matrix
54 nrows=5
55 ncols=5
56 axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0,
nrows) for c in range(0, ncols) ]
57
58 # remove the x and y ticks
59 for ax in axes:
60     ax.set_xticks([])
61     ax.set_yticks([])

```

```
62     ax.set_xlim(0,5)
63     ax.set_ylim(0,5)
64
65     #Add background colors
66     #This has been done manually for more fine-grained control
67     #Run the loop below to identify the indice of the axes
68
69     #Identify the index of the axes
70     green = [10, 15, 16, 20 , 21] #Green boxes
71     yellow = [0, 5, 6, 11, 17, 22, 23] #yellow boxes
72     orange = [1 , 2, 7, 12, 13, 18, 19, 24] # orange boxes
73     red = [3, 4, 8, 9, 14] #red boxes
74
75     for _ in green:
76         axes[_].set_facecolor('green')
77
78     for _ in yellow:
79         axes[_].set_facecolor('yellow')
80
81     for _ in orange:
82         axes[_].set_facecolor('orange')
83
84     for _ in red:
85         axes[_].set_facecolor('red')
86
87
88     #Add labels to the Green boxes
89     # axes[10].text(0.1,0.8, '4')
90     # axes[15].text(0.1,0.8, '2')
91     # axes[20].text(0.1,0.8, '1')
92     # axes[16].text(0.1,0.8, '5')
93     # axes[21].text(0.1,0.8, '3')
94
95     #Add labels to the Yellow boxes
96     # axes[0].text(0.1,0.8, '11')
97     # axes[5].text(0.1,0.8, '7')
98     # axes[6].text(0.1,0.8, '12')
99     # axes[11].text(0.1,0.8, '8')
100    # axes[17].text(0.1,0.8, '9')
101    # axes[22].text(0.1,0.8, '6')
102    # axes[23].text(0.1,0.8, '10')
103
104    #Add lables to the Orange boxes
105    # axes[1].text(0.1,0.8, '16')
106    # axes[2].text(0.1,0.8, '20')
107    # axes[7].text(0.1,0.8, '17')
108    # axes[12].text(0.1,0.8, '13')
109    # axes[13].text(0.1,0.8, '18')
```

```

110 # axes[18].text(0.1,0.8, '14')
111 # axes[19].text(0.1,0.8, '19')
112 # axes[24].text(0.1,0.8, '15')
113
114 #Add lables to the Red Boxes
115 # axes[3].text(0.1,0.8, '23')
116 # axes[8].text(0.1,0.8, '21')
117 # axes[4].text(0.1,0.8, '25')
118 # axes[9].text(0.1,0.8, '24')
119 # axes[14].text(0.1,0.8, '22')
120
121 # run throuh datas and generate axis datas
122 dict_bubble_axis = dict()
123 bubble_axis = list()
124 for datasets in datas:
125     # get the datas
126     riskid = datas.get(datasets).get('risk-id')
127     x_axis = int(datas.get(datasets).get('x-axis'))
128     y_axis = int(datas.get(datasets).get('y-axis'))
129     axis_point = matrix.get((x_axis, y_axis))
130     x_axis_text = float(datas.get(datasets).get('x-axis-text'))
131     y_axis_text = float(datas.get(datasets).get('y-axis-text'))
132     x_axis_bubble = float(datas.get(datasets).get('x-axis-bubble'))
133     y_axis_bubble = float(datas.get(datasets).get('y-axis-bubble'))
134     bubble_axis.append(axis_point)
135
136 # merge riks if two or more risks share the same axispoint
137 if dict_bubble_axis.get(axis_point):
138     risktag = dict_bubble_axis.get(axis_point).get('risk')
139     risktag = risktag + '/' + riskid
140     x_axis_text = x_axis_text + 0.25
141     y_axis_text = y_axis_text - 0.5
142     bubble_size = bubble_standard_size * 2
143 else:
144     risktag = itemname + riskid
145     bubble_size = bubble_standard_size
146 dict_axis_value = dict()
147
148 dict_axis_value['risk'] = risktag
149 dict_axis_value['x-axis-text'] = x_axis_text
150 dict_axis_value['y-axis-text'] = y_axis_text
151 dict_axis_value['x-axis-bubble'] = x_axis_bubble
152 dict_axis_value['y-axis-bubble'] = y_axis_bubble
153 dict_axis_value['size'] = bubble_size
154 dict_bubble_axis[axis_point] = dict_axis_value
155
156 # cleanup the list, remove duplicated entries
157 bubble_axis = set(bubble_axis)

```

```

158
159     # plot the bubbles and texts in the bubbles
160     for axispoint in bubble_axis:
161         axes[axispoint].scatter(dict_bubble_axis[axispoint]['x-axis-bubble'],
162                                dict_bubble_axis[axispoint]['y-axis-bubble'], dict_bubble_axis[axispoint]['
163                                size'], alpha=1)
164         axes[axispoint].text(dict_bubble_axis[axispoint]['x-axis-text'],
165                              dict_bubble_axis[axispoint]['y-axis-text'], s=dict_bubble_axis[axispoint]['
166                              risk'], va='bottom', ha='center')
167
168     # save the plot as image
169     plt.savefig(image_path)
170
171 """
172 Config File:
173     1. Name
174     2. Startpoint Directory
175     3. Destination Dir
176     4. Alternate Path
177     5. Data File Name
178
179 Data File:
180     1. Spalte: Nummer
181     2. x-achse
182     3. x-achse
183
184 """
185
186 """
187 Matrix
188 This Matrix translate the x/y axis from a given risk matrix csv to the
189 axispoint.
190
191 The key of each axispoint is an integer tuple (x, y)
192 So, you can access the axis point this way:
193 <axispoint> = matrix.get((<x_axis>, <y_axis>))
194
195 """
196 matrix = {
197     # first column
198     (1, 1):20,
199     (1, 2):15,
200     (1, 3):10,
201     (1, 4):5,
202     (1, 5):0,
203     # second column
204     (2, 1):21,
205     (2, 2):16,
206     (2, 3):11,
207     (2, 4):6,
208     (2, 5):1,

```



```
201     # third column
202     (3, 1): 22,
203     (3, 2): 17,
204     (3, 3): 12,
205     (3, 4): 7,
206     (3, 5): 2,
207     # fourth column
208     (4, 1): 23,
209     (4, 2): 18,
210     (4, 3): 13,
211     (4, 4): 8,
212     (4, 5): 3,
213     # fifth column
214     (5, 1): 24,
215     (5, 2): 19,
216     (5, 3): 14,
217     (5, 4): 9,
218     (5, 5): 4
219 }
220
221 # load the configuration file
222 riskmatrix_conf_filename = 'conf.csv'
223 riskmatrix_conf_dir = 'source/configuration/'
224 conf_riskmatrix_path = os.path.join(os.path.dirname(os.getcwd()),
225                                     riskmatrix_conf_dir)
226 conf_csv_path = os.path.join(conf_riskmatrix_path, riskmatrix_conf_filename)
227 with open(conf_csv_path) as f:
228     csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]
229
230 (_, *header), *data = csv_list
231 conf = {}
232 for row in data:
233     key, *values = row
234     conf[key] = {key: value for key, value in zip(header, values)}
235
236 for risks in conf:
237     riskmatrix(risks, conf, matrix)
238 # data = pd.read_csv('/home/itgramic/LaTeX/riskmatrix/src/source/riskmatrixproblem
239 .csv', header=None, dtype={0: str}).set_index(0).squeeze().to_dict()
```

Listing 14: Python LaTeX - riskmatrix - Risikomatrizen