ibW Höhere Fachschule Südostschweiz

Diplomarbeit Technik und Wirtschaftsinformatik 2023-2024

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

Zusammenfassung

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

Management Summary

Diplomarbeit Michael Graber

Inhaltsverzeichnis

Abkürzungen **4**

1 Einleitung **1**

1.1 Ausgangslage und Problemstellung 1

1.1.1 Das Kantonsspital Graubünden 1

1.1.2 Die ICT des Kantonsspital Graubünden 3

1.1.3 Rolle in der ICT vom Kantonsspital Graubünden 5

1.1.4 Ausgangslange 6

1.1.5 Problemstellung 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 24

1.8 GANTT-Diagramm 24

1.9 Status-Reports 26

1.9.1 Status Reports 26

2 Umsetzung **28**

2.1 Evaluation 28

2.1.1 Erheben und Gewichten der Anforderungen 28

2.1.2 Exkurs Architektur 30

2.1.3 Testziele erarbeiten 35

2.1.4 PostgreSQL Benchmarking 35

2.1.5 Analyse gängiger PostgreSQL HA Cluster Lösungen 35

2.1.6 Installation verschiedener Lösungen 38

2.1.7 Gegenüberstellung der Lösungen 40

2.1.8 Entscheid 40

2.2 Aufbau und Implementation Testsystem 40

2.2.1 Bereitstellen der Grundinfrastruktur 40

2.2.2 Installation und Konfiguration PostgreSQL HA Cluster 40

2.2.3 Technical Review der Umgebung 40

2.3 Testing . . . 40
2.3.1 Testing . . . 40
2.3.2 Protokollierung . . . 40
2.3.3 Review und Auswertung . . . 41
2.4 Troubleshooting und Lösungsfindung . . . 41

3 Resultate **42**
3.1 Zielüberprüfung . . . 42
3.2 Schlussfolgerung . . . 42
3.3 Weiteres Vorgehen / offene Arbeiten . . . 42
3.4 Persönliches Fazit . . . 42

Abbildungsverzeichnis **43**

Tabellenverzeichnis **44**

Listings **45**

Literatur **46**

Glossar **48**

Anhang **i**
I Statusbericht . . . i
I.I . . . i
II Rapport . . . ii
III rke2 . . . iii
III.I Vorbereitung . . . iii
III.II Installation . . . iii
III.III Cluster Konfiguration . . . iv
IV pgpool-II . . . vi
IV.I PostgreSQL Cluster Installation . . . vi
IV.II yugabyteDB . . . vi
V Stackgres mit Citus . . . vi
VI zotero.py . . . vi
VII riskmatrix.py . . . xii
VIII Disposition . . . xviii

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

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[9]

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[22]

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[7]

1.1.2 Die ICT des Kantonsspital Graubünden

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

Abbildung 1.4: Organigramm Kantonsspital Graubünden

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

Abbildung 1.5: Organigramm Departement 10 - ICT

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

1.1.3 Rolle in der ICT vom Kantonsspital Graubünden

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

1.1.4 Ausgangslange

Die meisten der über 400 Applikationen, die das KSGR betreibt, haben in den allermeisten Fällen ihre Daten in Datenbanksysteme speichern. Entsprechend der Vielfalt der Applikationen existieren auch eine vielzahl an Datenbanksystemen und Versionen.
Basierend auf der Liste *DB-Engines Ranking*[6] 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:

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

Tabelle 1.3: Datenbankinventor - Nach Betriebssystemen aufgeschlüsselt

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

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

1.1.5 Problemstellung

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

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

Dadurch ergeben sich nach BSI folgende Risiken:

Identifikation						Abschätzung		Behandlung			
								Massnahmen ergreifen?	Zielwert		Massnahme
ID	Schutzziel	Referenz BSI 200-3	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM		WS	SM	
1	I	G0.22	Manipulation von Informationen	Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja	1	2	Best-Practice bei Härtung der Systeme. Redundanzen einführen
2	A	G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung.	4	4	Ja	2	2	Redundanzen einführen
3	C, I, A	G0.26	Fehlfunktion von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Fehlfunktionen erleiden. Allerdings versuchen Datenbanksysteme, die Auswirkungen so gering wie möglich zu halten.	Fehlfunktionen können innerhalb von Datenbanksystemen die Datenkonsistenz verletzen, Daten können verloren gehen oder ungewollt von Dritten und unberechtigten Personen eingesehen werden. Systeme könnten nicht mehr oder nur noch eingeschränkt verfügbar werden. Daher sind sowohl Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	2	4	Ja	2	2	Systeme zentralisieren Lifecycle etablieren
4	C, I, A	G0.27-1	Ressourcenmangel (personelle Ressourcen)	Aufgrund der sehr heterogenen Landschaft ist der Administrationsaufwand für die jetzigen Systeme sehr gross. Zu gross, als das für jede Datenbank und deren Betriebssystem die notwendige Zeit für eine bedarfsgerechte Administration erbracht werden kann. Dadurch bleiben Fehler länger unentdeckt, Hotfixes, Patches, Updates und Upgrades können nicht oder nicht zur richtigen Zeit eingespielt werden. Bei einem akuten Problemfall ist nicht garantiert, das die Leute erreichbar sind, die notwendig sind	Die Auswirkungen können vielfältig sein, abhängig davon welcher Aspekt unter dem Ressourcenmangel leidet. Grundsätzlich wird aber sowohl die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	3	3	Ja	2	3	Systeme zentralisieren
5	A	G0.27-2	Ressourcenmangel (technische Ressourcen)	Kann auftreten wenn Ressorucenwachstum zu spät bemerkt wird. So kann die CPU Usage oder das Memory Usage schnell anwachsen. Auch der Storage eines Betriebssystems kann nicht mehr ausreichend für ein System werden.	Wenn die CPU- und Memory-Usage über einen gewissen Schwellwert geht, fängt das Betriebssystem an zu Priorisieren. Dies wird primär der Endanwender in form von Performance Einbussen bemerken. Im schlimmsten Fall steht eine Anwendung nicht mehr zur Verfügung. Gefährlicher sind Storage Overflows, besonders wenn die Datenbank nicht mehr alle Informationen schreiben konnte, die sie für einen korrekten Neustart benötigte.	2	2	Ja	1	2	Monitoring verschärfen
6	C, I, A	G0.31	Fehlerhafte Nutzung oder Administration von Geräten und Systemen	Durch die Vielfalt an Datenbankversionen und Betriebssystemen und Plattformen worauf diese betrieben werden, besteht allen voran das Risiko einer Fehlerhafter Administration und Konfiguration.	Doch die folgen bleiben nichtsdesto trotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen. Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist.	4	3	Ja	2	3	Systeme zentralisieren
7	C, I, A	G0.32	Missbrauch von Berechtigungen	Obwohl das Microsoft Active Directory die Zentrale Benutzerverwaltung ist, sind die wenigsten Datenbanken an dieses angeschlossen. Hinzu kommt der umstand, das in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen. Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen Wissendlich oder Unwissendlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen.	Der Wissentliche oder Unwissentliche Missbrauch von Berechtigungen kann verheerende Auswirkungen haben. Unter anderem können Daten missbräuchlich abgezogen werden, Daten manipuliert oder das ganze System komplett zerstört werden.	2	4	Ja	2	2	Systeme zentralisieren Übergreifendes Berechtigungskonzept einführen Monitoring der Zugriffe
8	A, I	G0.45	Datenverlust	Zudem wurden die meisten Datenbanken nur mittels Snapshots oder einem Filesystem Backup gesichert, nicht über eine eigentliche Sicherung mittels WAL. Gerade die fehlende WAL-Archivierung führt im Backupfall dazu, das alle Transaktionen die zwischen dem letzten Backup nicht mehr vorhanden sind. Hinzu kommt, das für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde. Ein weiterer Aspekt des Risikos besteht in der tatsache, das aufgrund der grossen Anzahl Datenbanken und deren Heterogenität nur wenige Backups auch wirklich regelmässig geprüft werden.	Aus dem Risiko ergeben sich zwei Auswirkungen, die aber beide ein hohes Mass an Schaden verursachen können. Erstens könnten Backups gar nicht mehr Wiederhergestellt werden, dies hätte dann einen Totalen Datenverlust zur Folge. Die zweite Ursache erwächst auf der fehlenden WAL-Archivierung, dadurch können zwar die Daten bis zu einem Zeitpunkt X Wiederhergestellt werden allerdings sind diese dann nicht zwingend Konsistent.	4	5	Ja	1	3	Systeme zentralisieren Einheitliches Backupkonzept Regelmässige Restore-Tests

Tabelle 1.4: Risiko-Matrix aktuelle Situation PostgreSQL Datenbanken

Abbildung 1.6: 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. [15, 18] folgende zehn Tasks die zum täglichen Alltag gehören:

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

Tabelle 1.5: Administrative Aufgaben

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

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

Tabelle 1.6: Automatisierung Administrativer Aufgaben

Mit der Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

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

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

Daraus ergeben sich folgende Ziele:

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

Tabelle 1.7: Ziele

1.3 Abgrenzungen

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

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

Tabelle 1.8: Gegebene Systeme

Daraus ergeben sich nach nach Züst, Troxler 2002[24] 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 eintrifft. Dies wären folgende:

Objekt	Abhängigkeit	Beschreibung	Status	Risiko	Impact
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.
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
Log Management / SIEM System	Sichern der Logfiles für Log Rotation	Ein Log Management System / SIEM muss vorhanden sein, um Logs langfristig sichern zu können.	Die bestehenden Plattformen für das 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.
HP-UX Ablöseprojekt	Ressourcen	Das Projekt zur Ablösung der HP-UX Plattform für die Oracle Datenbanken geht in die Konzeptions- und Umsetzungsphase.	Das Projekt geht in die Konzeptions- und Umsetzungsphase.	Als Oracle DBA bin ich stark in das Projekt eingebunden. Es besteht, dass Risiko eines Ressourcenengpasses	Projekt kann nicht Zeitgemäss abgeschlossen werden
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.

Tabelle 1.9: Abhängigkeiten

1.5 Risikomanagement

Aus den Abhängigkeiten heraus wurden folgende Risiken identifiziert:

Identifikation				Abschätzung		Behandlung			
						Massnahmen ergreifen?	Zielwert		Massnahme
ID	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM		WS	SM	
1	Fehlende Ressourcen	Viele parallele Projekte, Aufträge und der Tagesbetrieb	Ressourcen während der Diplomarbeit sind knapp bemessen	3	4	Ja	2	2	Organisation und Selbstmanagement
2	HP-UX Ablöseprojekt	Das Projekt ist sehr Umfangreich und ist in die Konzeptions- und Umsetzungsphase gestartet	Das Projekt wird parallel zur Diplomarbeit sehr viele Ressourcen und Aufmerksamkeit binden	4	4	Ja	3	3	Ressourcen reservieren
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	HP-UX Plattform, DELL NetWorker / Data Domain Umgebung und HPE 3PAR SAN Storage Umgebung sind über dem Lifecycle und haben in den vergangenen Monaten immer wieder kritische Ausfälle erlebt	Bei einem Event, ausgelöst durch das Alter der HP-UX Plattform, der DELL NetWorker / Data Domain Umgebung oder dem SAN Storage, kann der ganze Betrieb zum erliegen kommen und entsprechend viele Ressourcen aufgrund der kritikalität binden	4	4	Ja	3	3	Monitoring vorgängig ausbauen und Massnahmen definieren
4	Schwächen beim Selbstmanagement und in der Selbstorganisation	Selbstmanagement und Organisation ist nicht meine Stärke	Das Projekt verzettelt sich, Zeit geht verloren. Auch eine folge könnte der Scope Verlust sein	3	3	Ja	2	2	Werkzeuge im Vorfeld definieren und bereitstellen
5	Scope verlust während des Projekts	Der Scope kann während des Projekts verloren gehen	Verzettelung und Zeitverlust bis hin zu scheitern	3	4	Ja	2	3	Ziele klar definieren
6	Scope Creep	Der Umfang kann stark steigen wenn Ziele nicht genau genug definiert wurden	Zeitverlust bis hin zu scheitern des Projekts	3	4	Ja	3	3	Ziele SMART definieren
7	SIEM / Log Plattform nicht betriebsbereit	Die öffentliche Ausschreibung für die neue / Log Plattform wurde erst am 23.10.2023 veröffentlicht. Bis zur Implementation kann noch Zeit vergehen.	Logs müssen länger auf dem System selber vorgehalten werden. Zudem müssen ggf. eigene Massnahmen zum Auslesen von Logs getroffen werden	4	1	Nein			
8	Foreman nicht betriebsbereit	Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase. Dadurch besteht das Risiko, dass sie nicht betriebsbereit zum Start der Diplomarbeit ist	Ms müssen von Hand provisioniert werden. Dies bedeutet einen massiven Mehraufwand und verzögert ggf. die Evaluationsphase und mit sicherheit die Installationsphase	3	5	Ja	3	4	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten.

Tabelle 1.10: Risiko-Matrix der Diplomarbeit

Daraus ergibt sich folgende Risikomatrix

Abbildung 1.9: 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	Dauer [h]	Geplante Dauer [h]	Verbleibende Zeit [h]
0	Dokumentation	6.500000	80	73.500000
1	Evaluation	2.500000	4	1.500000

Tabelle 1.11: Projektcontrolling

1.8 GANTT-Diagramm

Nr.	Vorgangsm	Vorgangsname	Dauer	Vorgänger	Meilenstein	Anfang	Ende	Ist-Anfang	Ist-Ende	Geplante Arbeit	Ist-Arbeit
1		**Phase I: Themenerarbeitung- und eingabe**	**81 Tage**		**Nein**	**Don 17.08.23**	**Don 07.12.23**	**Don 17.08.23**	**NV**	**48 Std.**	**3 Std.**
2		Einführung Diplomarbeit	81 Tage		Nein	Don 17.08.23	Don 07.12.23	Don 17.08.23	NV	20 Std.	3 Std.
3		Ideensammlung	30 Tage		Nein	Don 17.08.23	Mit 27.09.23	NV	NV	8 Std.	0 Std.
4		Abgabe Ideensammlung	1 Tag		Ja	Mit 27.09.23	Mit 27.09.23	NV	NV	0 Std.	0 Std.
5		Disposition erstellen	2.5 Tage		Nein	Mit 15.11.23	Son 19.11.23	NV	NV	20 Std.	0 Std.
6		Abgabe Disposition	1 Tag		Ja	Mon 20.11.23	Mon 20.11.23	NV	NV	0 Std.	0 Std.
7		Bewilligung Disposition durch FV	1 Tag	6	Ja	Don 07.12.23	Don 07.12.23	NV	NV	8 Std.	0 Std.
8		**Phase II: Vorarbeitungsarbeiten**	**48 Tage**	**7**	**Nein**	**Fre 08.12.23**	**Die 13.02.24**	**NV**	**NV**	**25 Std.**	**0 Std.**
9		Vorebereitungsmassnahmen	43 Tage	7	Nein	Fre 08.12.23	Die 06.02.24	NV	NV	20 Std.	0 Std.
10		Abgabe Statusbericht	1 Tag	9	Ja	Mit 07.02.24	Mit 07.02.24	NV	NV	1 Std.	0 Std.
11		Vernissage	1 Tag?	10	Ja	Die 13.02.24	Die 13.02.24	NV	NV	4 Std.	0 Std.
12		**Phase III: Hauptarbeiten**	**64.75 Tage?**	**11**	**Nein**	**Mit 14.02.24**	**Don 06.06.24**	**NV**	**NV**	**0 Std.**	**0 Std.**
13		**Haupteil Diplomarbeit**	**55.75 Tage?**		**Nein**	**Mit 14.02.24**	**Fre 24.05.24**	**NV**	**NV**	**200 Std.**	**0 Std.**
14		**Umsetzung**	**25 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 27.03.24**	**NV**	**NV**	**0 Std.**	**0 Std.**
15		**Evaluation**	**13 Tage?**		**Nein**	**Mit 14.02.24**	**Mit 06.03.24**	**NV**	**NV**	**0 Std.**	**0 Std.**
16		Anforderungskatalog	6 Tage		Nein	Mit 14.02.24	Fre 23.02.24	NV	NV	16 Std.	0 Std.
17		Vorbereitung Benchmarking	1 Tag	16	Nein	Mon 26.02.24	Mon 26.02.24	NV	NV	4 Std.	0 Std.
18		Analyse PostgreSQL HA Cluster Lösungen	4 Tage	16	Nein	Mon 26.02.24	Fre 01.03.24	NV	NV	16 Std.	0 Std.
19		Gegenüberstellung	2 Tage	16;17;18	Nein	Mon 04.03.24	Die 05.03.24	NV	NV	8 Std.	0 Std.
20		Variantenentscheid	1 Tag?	19	Ja	Mit 06.03.24	Mit 06.03.24	NV	NV	4 Std.	0 Std.
21		**Aufbau und Implementation Testsystem**	**8 Tage?**	**20**	**Nein**	**Fre 08.03.24**	**Mit 20.03.24**	**NV**	**NV**	**27 Std.**	**0 Std.**
22		Basisinfrastruktur	2 Tage		Nein	Fre 08.03.24	Mon 11.03.24	NV	NV	4 Std.	0 Std.
23		Installation und Konfiguration PostgreSQL HA Cluster	5 Tage	22	Nein	Die 12.03.24	Die 19.03.24	NV	NV	20 Std.	0 Std.
24		Technical Review	1 Tag?	23	Ja	Mit 20.03.24	Mon 16.09.24	NV	NV	3 Std.	0 Std.
25		**Testing**	**4 Tage?**	**24**	**Nein**	**Fre 22.03.24**	**Mit 27.03.24**	**NV**	**NV**	**14 Std.**	**0 Std.**
26		Testing Testsystem	1 Tag	24	Nein	Fre 22.03.24	Fre 22.03.24	NV	NV	8 Std.	0 Std.
27		Protokollierung	2 Tage	26	Nein	Mon 25.03.24	Die 26.03.24	NV	NV	4 Std.	0 Std.
28		Review und Auswertung	1 Tag?	27	Ja	Mit 27.03.24	Mit 27.03.24	NV	NV	2 Std.	0 Std.
29		Troubleshooting und Lösungsfindung	1 Tag?	25	Nein	Die 02.04.24	Die 02.04.24	NV	NV	8 Std.	0 Std.
30		**Resultate**	**6 Tage?**	**25**	**Nein**	**Die 02.04.24**	**Mit 10.04.24**	**NV**	**NV**	**7 Std.**	**0 Std.**
31		Zielüberprüfung	4 Tage		Nein	Die 02.04.24	Mon 08.04.24	NV	NV	2 Std.	0 Std.
32		Schlussfolgerung	1 Tag?	31	Nein	Die 09.04.24	Die 09.04.24	NV	NV	2 Std.	0 Std.
33		Weiteres Vorgehen / offene Arbeiten	1 Tag?		Nein	Die 02.04.24	Die 02.04.24	NV	NV	1 Std.	0 Std.
34		Persönliches Fazit	1 Tag	31;32	Nein	Mit 10.04.24	Mit 10.04.24	NV	NV	2 Std.	0 Std.
35		1. Expertengespräch	1 Tag?		Ja	Mit 14.02.24	Mit 14.02.24	NV	NV	0 Std.	0 Std.
36		2. Expertengespräch	1 Tag?		Ja	Fre 15.03.24	Fre 15.03.24	NV	NV	0 Std.	0 Std.
37		Letztes Expertengespräch	1 Tag?		Ja	Mit 15.05.24	Mit 15.05.24	NV	NV	0 Std.	0 Std.
38		Dokumentation	55.75 Tage		Nein	Mit 14.02.24	Fre 24.05.24	NV	NV	80 Std.	0 Std.
39		Puffer	9 Tage		Nein	Die 16.04.24	Die 30.04.24	NV	NV	16 Std.	0 Std.
40		Abgabe Diplomarbeit	1 Tag?	13;30;37	Ja	Fre 24.05.24	Fre 24.05.24	NV	NV	0 Std.	0 Std.
41		Vorbereitung Präsentation	5 Tage	40	Nein	Die 28.05.24	Mon 03.06.24	NV	NV	20 Std.	0 Std.
42		Präsentation und Fachgespräch	1 Tag?	40;41	Nein	Die 04.06.24	Die 04.06.24	NV	NV	1 Std.	0 Std.
43		Diplomausstellung	1 Tag?	42	Nein	Mit 05.06.24	Mit 05.06.24	NV	NV	0 Std.	0 Std.
44		Diplomfeier	1 Tag?	42;43	Nein	Don 06.06.24	Don 06.06.24	NV	NV	0 Std.	0 Std.

Projekt: Diplomarbeit - Postgre
Datum: Die 06.02.24

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

1.9 Status-Reports

1.9.1 Status Reports

1.9.1.1 Initialer Statusbericht

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

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

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

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

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

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

Bemerkungen / Informationen	Anträge

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

erledigte Lieferobjekte (inkl. allfällige Links)

Tabelle 1.12: Initialer Statusbericht

2 Umsetzung

2.1 Evaluation

2.1.1 Erheben und Gewichten der Anforderungen

2.1.1.1 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 evaluirten Lösungen Cloud Native Lösungen sein. Wenn der Zeitaufwand reicht, können auch eine Cloud Native und Monolithisches System aufgebaut werden.

Nr.	Anforderung	Beschreibung	System	Muss / Kann
1	Systemvielfallt	Es muss mindestens eine Monolitisches und mindestens 2 zwei Distributed SQL Cluster ermittelt werden	Beides	MUSS
2	Synergien	Skripte und APIs des Monolithisches Systems müssen auch in einem Distributed SQL System verwendet werden können	Beides	MUSS
3	Failover	Das Clustersystem muss bei einem Nodeausfall automatisch auf einen anderen Node umstellt	Beides	MUSS
4	Failover	Beim Failover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
5	Failover-Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgefühgrt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
6	Switchover	Das System muss ein Skript oder eine API liefern, welche einen geordeten Switchover auf einen anderen Node erlaubt	Beides	MUSS
7	Switchover	Beim Switchover dürfen bestehende Connections nicht getrennt werden oder sofort Wiederhergestellt werden	Beides	MUSS
8	Switchover-Geschwindigkeit	Das umstellen auf den nächsten Node muss so schnell ausgefühgrt werden, das ein Disconnect mittels Client-Konfiguration (Timeout) verhindert wird.	Beides	MUSS
9	Restore	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	Beim Wiederherstellen des Ursprungszustands darf es zu keinem Datenverlust kommen	Beides	MUSS
11	Restore	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	Es muss eine Synchrone Replikation sichergestellt werden	Monolitisch	MUSS
14	Replikation	Die Replikation muss sicherstellen, das es bei einem Failover/Switchover zu keinem Fehler kommt	Monolitisch	MUSS
15	Sharding	Die Datenkonsistenz und Datenintegrität auf den Shards muss sichergestellt werden	Distributed SQL	MUSS
16	Sharding	Die Synchronisation der Shards muss sicherstellen, dass es zu keinem Datenverlust kommt	Distributed SQL	MUSS
17	Quorum	Das Clustersystem muss über ein Quorum-System besitzen	Beides	MUSS
18	Quorum	Das Quorum des Clustersystems muss robust genug sein, um eine Split-Brain-Situation zu verhindern	Beides	MUSS
19	Connection	Das Clustersystem muss sicherstellen, dass eine Applikation ohne Entwicklungsaufwand mittels dem PostgreSQL Wired Connector zugreifen kann	Beides	MUSS
20	Management-API	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
21	Management-API	Es müssen gängige Standards für Authentifizierung und Autorisierung mitgebracht werden	Beides	MUSS
22	Backup	Backups müssen mittels PostgreSQL Standards angezogen werden	Beides	MUSS
23	Backup Restore	Backups müssen mittels PostgreSQL Standards restored werden können	Beides	MUSS
24	Housekeeping - Log Rotation	Das Clustersystem muss die möglichkeit zur Log Rotation bieten	Beides	MUSS
25	Self Heahling	Das Clustersystem muss im Fehlerfall Nodes selber wiederherstellen können	Beides	KANN
26	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

Tabelle 2.1: Anforderungskatalog

Kostenrechnung

Für die Kostenberechnung des Zeitaufwands wird im KSGR intern mit $120CHF/h$ gerechnet.
Jeder Arbeitstag hat dabei $8.4h$ und pro Jahr wird mit $220Tagen$ gerechnet.

Messung des Zeitaufwands

Der Zeitaufwand in der Evaluationsphase kann nur mit manueller Ausführung gemessen werden, da die Automatisierung nicht in der Evaluationsphase umgesetzt werden kann.
In die Evaluation einfliessen wird aber die Schätzung, wie viel Aufwand betrieben werden muss um die wichtigsten Tasks automatisieren zu können.

Folgende Messgrössen werden gestellt:

Quorum

Zeitaufwand Quorum erweitern

Bemessen wird, wie lange man braucht um einen neuen Node dem Quorum hinzuzufügen.

Zeitaufwand Failover und Recovery

Bemessen wird, wie lange ein Failover und ein anschliessender Recover auf den normalen Zustand dauert.

Failover Funktionsfähigkeit

Misst, ob der Failover bei korrekter Konfiguration funktionsfähig ist wie er vom entsprechenden System spezifiziert wurde.

Failover Reaktionszeit

Gemessen und bemessen wird, wie lange es im Failoverszenario dauert, bis auf einen Standby-Node umgeschaltet wird und wie lange es dauert bis offene Connections wieder voll funktionsfähig sind.

Recoverydauer

Bemisst, wie lange es nach einem Failover-Szenario dauert, bis der Normalzustand Widerhergestellt werden kann.

2.1.1.2 Gewichtung

2.1.2 Exkurs Architektur

2.1.2.1 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

2.1.2.2 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[17]. Dabei wird zwischen den folgenden beiden Replikationen unterschieden:

Synchrone Replikation
Wenn bei einer Synchronen Replikation eine Transaktion abgesetzt wird, wird der Commit auf der primären Seite erst gesetzt, wenn die Änderung auf der sekundären Seite oder den

sekundären Seiten ebenfalls eingetragen und Committed ist. Bis zu diesem Moment ist die Transaktion nicht als Committed.
Dies wird dann zum Problem, wenn keine Verbindung mehr zu mindesten einer sekundären Seite vorhanden ist. Zudem wird die Synchrone Replikation bei hohen Latenzen zum Bottleneck der Datenbank.

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

2.1.2.3 Quorum

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

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

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

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

Hier ein Beispiel wie sie in den Artikeln [16, 20, 13] 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 2.2: Quorum Beispiele

2.1.2.4 CAP Theorem

Das CAP Theorem besagt, das nur zwei der drei folgenden drei Merkmale von verteilten Systeme gewährleistet werden können[8].

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

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

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

Abbildung 2.2: CAP-Theorem

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

2.1.2.5 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[14]:

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.

2.1.3 Testziele erarbeiten

2.1.4 PostgreSQL Benchmarking

PostgreSQL bietet ein Benchmarking-Tool,[12, 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[11].

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[21]. 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[19, 10]:

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 [4] 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

2.1.5.5 Patroni

Replikation

Proxy Patroni benötigt einen HAProxy, um Load Balancing usw. [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.

etcd Patroni benötigt etcd als key-value-store

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[23].

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

2.1.5.8.2

2.1.6 Installation verschiedener Lösungen

2.1.6.1 Monolitische Umgebung

2.1.6.2 Kubernetes

Um ein minimales, dem Produktiven möglichst nahes Enviroment für die evaluation zu erhalten, wurde folgendes Setting ausgewählt:

Abbildung 2.4: Kubernetes - Evaluations-Enviroment

Die genauen Spezifikationen sind wie folgt:

Linux Distribution	Debian 11
Kubernetes Runtime	rke2
Container-Enviroment	containerd
Container Network Interface (CNI)	cilium
Cloud Native Storage (CNS)	local-path-provisioner

Tabelle 2.3: Evaluations settings

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 2.4: Evaluationssyssteme

2.1.7 Gegenüberstellung der Lösungen

Abbildung 2.5: Kosten-Nutzen-Analyse

2.1.8 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[9] 1
1.2 Wahlkreise Kanton St. Gallen[22] 2
1.3 Spitalregionen / Spitalstrategie Kanton St. Gallen[7] 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 Kubernetes - Evaluations-Enviroment 39
2.5 Kosten-Nutzen-Analyse 40

1 Disposition lxiv

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 24
1.12 Initialer Statusbericht 27

2.1 Anforderungskatalog 29
2.2 Quorum Beispiele 33
2.3 Evaluations settings 39
2.4 Evaluationssyssteme 39

I Arbeitsrapport ii

Listings

1 Proxy Settings . . . iii
2 Downlaod rke2 server . . . iii
3 rke2 server installieren . . . iii
4 Downlaod rke2 agent . . . iii
5 rke2 agent aktivieren . . . iii
6 rke2 server proxy . . . iv
7 rke2 server proxy kopieren . . . iv
8 rke2 server cilium installieren . . . iv
9 rke2 server cilium aktivieren . . . iv
10 rke2 server starten . . . iv
11 iptables entries server . . . v
12 rke2 server token . . . vi
13 Python LaTex - zotero - Zotero BibLaTex Importer . . . vi
14 Python LaTex - riskmatrix - Risxikomatrizen . . . xii

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. *HAProxy Documentation Converter*. https://docs.haproxy.org/.

[4] ORACLE CORPORATION. „Oracle (Active) Data Guard 19c“. In: (2019), S. 14.

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

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

[7] 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.

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

[9] 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.

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

[11] The PostgreSQL Global Development Group. *27.1. Comparison of Different Solutions*. https://www.postgresql.org/docs/16/different-replication-solutions.html. 2023.

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

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

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

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

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

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

[18] 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.

[19] SRA OSS LLC. *pgpool Wiki*. https://www.pgpool.net/mediawiki/index.php/Main_Page. 2023.

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

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

[22] 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.

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

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

Glossar

Ansible Ansible ist ein Open-Soure Automatisierungstool zur Provisionierung, Konfiguration, Deployment und Orchestrierung. Ansible verbindet sich auf auf die Zielgeräte und führt dort die Hinterlegten Module aus. Oft werden die verschieden Aufgaben in einem Skript, in einem sogenannten Playbook geschrieben werden[**7SPK583Y**].. 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[**9EUWGEF8**].. 14, 15

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[**KA934RSV**] wurde.. 7

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[**I65F7WAQ**].. 4

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[**8XWD67EM**] 4

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

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

etcd etcd ist [**8A4R4E2D**]. 38

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 15, 31, 32, 36, 49

Foreman Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 16, 18, 20

Git Git ist eine Versionierungssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral sondern dezentral organsiert und arbeiten daher mit Working Copies von Repositories[**33C32L9G**, **SCE846MP**].. 48

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[**MPSC6ELK**].. 15, 18, 36

HAProxy HAProxy [**U6F2DWTQ**]. 37

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[**PV**
Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 15, 36

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, 18, 20

IBM DB2 IBM DB2 ist eine Relationale Datenbank[**DJX54K3M**] 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 realsieren. keepalived benötigt dazu keinen dritten Node, also einen Quorum-Node. Wenn die definierte sekundärseite keine Antwort mehr von der primären Seite nach einer definierten Anzahl versuchen in einem bestimmten Interval mehr bekommt, oder ein per Skript definiertes Event auf der primären Seite eintrifft, wird ein Failover auf die sekundäre Seite ausgeführt. Je nach Konfiguration kann der Restore auf die primäre Seite eingeleitet werden wenn diese wieder verfügbar ist oder der Restore unterbunden werden[**5IP362SV**, **ZW4PA3EQ**].. 36

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, 49

Linux Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und lose vom UNIX Derivat MINIX inspiert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 50

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

den Linux Distributionen tratt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 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[**QVZZTCG6**].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[**44L9YDSR**].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[**6LRCXMLC**]. Nebst Microsoft Windows und Windows Server lässt es sich seit Version 2014 ebenfalls auf Linux Betreiben. In der Wirtschaft ist die primäre Plattform aber Windows Server.. 5, 7, 50

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

MySQL Die Datenbank MySQL wurde Ursprünglich als reine Relationale Open-Source Datenbank von Firma MySQL AB 1994 Entwickelt. Der Name My leitet sich vom Namen My der Tochter des Mitbegründers Michael Widenius ab. Als Sun Microsystem 2008 MySQL übernahm, hielt sich die Option frei, bei einem Kauf von Sun Microsyszem durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 5, 7, 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, 48, 50, 52

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[**W5LMN5ZM**, **5D3IPPGJ**]. 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 komnt auf den Oracle Eigenenen Appliances ODA und Exadata zum Einsatz. Für den Zweck als DB Plattform kann ein für Oracle Datenbanken optimimierter Kernel verwendet werden. Zu

Oracle Linux kann ein kostenpflichtiger Support bezogen werden, allerdings ist die Distribution anders als RHEL auch ohne Lizenz erhältlich.. 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 erschienen Optimizer und mit der Version 9i 2001 erschienenn Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangheit ermöglichen) konnte Oracle sich stark von der Konkurenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Perforamce oder grosse Datenmengen geht.. 5, 7, 8, 34, 50

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 beliebstesten OpenSource Datenbanken. Zudem besteht in vielen bereichen eine gewisse ähnlichkeit zu Oracles Oracle Database.. 5, 7, 8, 9, 13, 34, 35, 36, 37, 38, vi

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 wei Oracle.. 3, 14, 15, 36, 51, vi

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

Quorum In verteilten Systemen resp. Cluster muss sichergestellt werden, das bei einem Ausfall oder ein Netzwerktrennung zwischen den Nodes es zu keiner Split-brain-Situation kommt. Hierzu wird i.d.R. ein Quorum verwendet. I.d.R. wird jener Teil des Quorums zum Primary oder alleinigen Node, der mit der die Mehrheit aller Nodes vereint. Daraus ergeben sich bestimmte grössen, mit 5 Nodes braucht es 3 Nodes um aktiv zu bleiben und mit 3 Nodes deren 2. Bei diesen Konstelationen wird daher darauf geachtet, eine ungerade Anzahl Nodes im Cluster zu halten um keine Pat-Situation zu provozieren. Im Kapitel Unterunterabschnitt 2.1.2.3 wird genauer auf die Mechanik eines Quorums eingegangen. . 36, 49

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[**Z9WAAQ2U**].. 4

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

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[**57XLMIRR**]. 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 Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[**ZRRXBFRA**, **7ZTCYW5G**, **JWVC9B7L**]. 4, 5, 16, 18, 20

SIEM Ein sammelt Daten aus verschieden Netzwerkkomponenten oder Geräten von Agents oder Logs. Diese Daten werden permanent analysiert und mit einem definierten Regelwerk gegengeprüft. Ziel ist es, verdächtige Events zu erkennen und einem Angriff zuvorzukommen oder ihn möglichst früh zu unterbinden[**78JPTB5R**].. 4, 16, 18

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[**QCM8CD5A**, **7VWNV2V4**]. 7

Split-brain Im Kapitel **??** werden die ursachen und folgenden eines Split-brains genauer besprochen. . 32, 51

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[**GUPY8F7E**, **PH3HQUCR**]. . 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[**JCLXWZSR**].. 7

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

SWOT-Analyse Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzueigen. Anhand einer SWOT-Analyse werden i.d.R. anschliessend Strategien abgeleitet um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder anzumildern.. 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[**FMIBZY6N**, **U29WWCXR**].. 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 enstanden im Laufe der Zeit viele offene und Proprioritäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 4

VRRP VRRP . 4, 49

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

Datum	Von	Bis	Dauer [h]	Phase	Tätigkeit	Bemerkung	Schwierigkeit	Lösungen
21.02.2024	15:00	16:00	1.000000	Evaluation	Anorderungskatalog erarbeiten	nan	nan	nan
22.02.2024	16:00	17:30	1.500000	Evaluation	Anorderungskatalog erarbeiten	nan	nan	nan
27.02.2024	10:00	11:30	1.500000	Dokumentation	Dokumentation erweitern	nan	nan	nan
27.02.2024	13:00	16:00	3.000000	Dokumentation	Dokumentation erweitern	nan	Viele LaTEX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken
28.02.2024	09:00	11:00	2.000000	Dokumentation	Dokumentation erweitern	nan	Viele LaTEX Tabellen.	Generator mit python pandas gebaut für alle möglichen Tabellen. Inkl. Aggregation und Pivot-Mechaniken

TABLE I: Arbeitsrapport

=:

III rke2

III.I Vorbereitung

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

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

export https_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
export http_proxy=http://sproxy.sivc.first-it.ch:8080
export HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
export no_proxy=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
export NO_PROXY=localhost,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16

source /etc/profile.d/proxy.sh
```

Listing 1: Proxy Settings

III.II Installation

III.II.I server

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

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

Listing 2: Downlaod rke2 server

Anschliessend muss das Package installiert werden:

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

Listing 3: rke2 server installieren

III.II.II agents

Der Agent muss direkt heruntergeladen werden:

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

Listing 4: Downlaod rke2 agent

Anschliessend muss der Dienst aktiviert werden:

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

Listing 5: rke2 agent aktivieren

III.III Cluster Konfiguration

III.III.I server

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

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

CONTAINERD_HTTPS_PROXY=http://sproxy.sivc.first-it.ch:8080
CONTAINERD_HTTP_PROXY=http://sproxy.sivc.first-it.ch:8080
CONTAINERD_NO_PROXY=localhost
    ,127.0.0.0/8,::1,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
```

Listing 6: rke2 server proxy

Dieses File muss entsprechend in das Homeverzeichnis gespeichert werden:

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

Listing 7: rke2 server proxy kopieren

Für den Netzwerkteil muss nun Cilium installiert werden:

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

Listing 8: rke2 server cilium installieren

Cilium muss nun aktiviert werden:

```
/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:

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

```
cni:
- cilium

systemctl restart rke2-server.service
```

Listing 10: rke2 server starten

Entsprechend muss die Firewall gesetzt werden:

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

# Generated by iptables-save v1.8.9 (nf_tables)
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -s 10.0.9.115/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.115" -j ACCEPT
-A INPUT -s 10.0.9.76/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.76" -j ACCEPT
-A INPUT -s 10.0.36.147/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.36.147" -j ACCEPT
-A INPUT -s 10.0.9.35/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.35" -j ACCEPT
-A INPUT -s 10.0.9.37/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.37" -j ACCEPT
-A INPUT -s 10.0.9.74/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.74" -j ACCEPT
-A INPUT -s 10.0.9.75/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.75" -j ACCEPT
-A INPUT -s 10.0.9.36/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.36" -j ACCEPT
-A INPUT -s 10.0.9.14/32 -p udp -m udp --dport 161 -m comment --comment "Allow
    SNMP for probe 10.0.9.14" -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p icmp -m icmp --icmp-type 8 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 6443 -j ACCEPT
-A INPUT -s 10.0.0.0/8 -p tcp -m tcp --dport 9345 -j ACCEPT
COMMIT
# Completed

systemctl restart iptables
```

Listing 11: iptables entries server

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

```
cni:
- cilium
token: <password safe>
```

Listing 12: rke2 server token

III.III.II agents

IV pgpool-II

IV.I PostgreSQL Cluster Installation

PostgreSQL Package Repository in Debian einbinden

IV.II yugabyteDB

IV.II.I minikube

IV.II.II yugabyteDB Konfiguration

V Stackgres mit Citus

VI zotero.py

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

def load_configuration():
    zotero_bibtex_config = dict()
    zotero_conf_filename = 'zotero_bibtex_configuration.json'
    zotero_conf_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', '
   configuration')
    # zotero_conf_dir = os.path.join(os.getcwd(), 'src', 'content')
    json_path = os.path.join(zotero_conf_dir, zotero_conf_filename)

    with open(json_path) as json_string:
        zotero_bibtex_config = json.load(json_string)

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

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

def get_data(zotero_bibtex_config):
    # result_limit = 100
    # access_type = 'groups'
    # zotero_access_id = '5245833'
    # collection_id = 'USSFDCEH'
    result_limit = int(zotero_bibtex_config.get('result_limit'))
    access_type = zotero_bibtex_config.get('access_type')
    zotero_access_id = zotero_bibtex_config.get('zotero_access_id')
    collection_id = zotero_bibtex_config.get('collection_id')
    API_KEY = zotero_bibtex_config.get('api_key')
    zotero_data = list()
    URL = 'https://api.zotero.org/' + str(access_type) + '/' + str(zotero_access_id) + '/collections/' + str(
        collection_id) + '/items?limit=1?format=json?sort=dateAdded?direction=asc'

    # API_KEY = '6Xgb3XhGjQXwA8NuZgu3bw3s'
    response = requests.get(URL, headers={'Zotero-API-Key': API_KEY})

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

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

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

            zotero_data.extend(zotero_result)

```

```
            start_index += result_limit
            index += 1

    return zotero_data

def convert_to_datetime(input_str, parserinfo=None):
    return parse(input_str, parserinfo=parserinfo)
def get_dates(date, bibtex_item_type, bibtex_month_attributes):
    dated_date = convert_to_datetime(date)
    return_value = dict()
    if bibtex_item_type in bibtex_month_attributes:
        year = dated_date.year
        month = dated_date.month
        return_value = {'year': year, 'month':month}
    else:
        year = dated_date.year
        return_value = {'year': year}

    return return_value

def split_creators(creators):
    if creators != []:

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

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

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

    return bib_entry


def write_bibliography(zotero_data, zotero_bibtex_config):
```

```
    # file_json = 'keystore.json'
    file_json = zotero_bibtex_config.get('keystore_file')
    keystore_path = zotero_bibtex_config.get('keystore_filepath')
    # tex_dir = os.path.join(os.path.dirname(os.getcwd()), 'source', 'configuration')
    tex_dir = os.path.join(os.path.dirname(os.getcwd()), keystore_path)
    # tex_dir = os.path.join(os.getcwd(), 'src', 'content')
    json_path = os.path.join(tex_dir, file_json)

    with open(json_path) as json_string:
        zotero_bibtex_keys = json.load(json_string)

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

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

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

        # some item types have no titles
```

```
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:
```

```
                    entryset = entryset + str(bibtex_item_attribute) + '=' + str(zotero_item_value)

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

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

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


zotero_bibtex_config = load_configuration()
zotero_data = get_data(zotero_bibtex_config)
write_bibliography(zotero_data, zotero_bibtex_config)
```

Listing 13: Python LaTex - zotero - Zotero BibLaTex Importer

VII riskmatrix.py

```
import matplotlib.pyplot as plt
import numpy as np
import pip as pd
import os
import csv
import pandas as pd

def riskmatrix(risk, conf, matrix):
    # get the risk datas
    risk_conf = conf.get(risk)
    startpath = risk_conf.get('startpath')
    destination = risk_conf.get('destinatination')
    imagename = risk_conf.get('imagename')
    datafilename = risk_conf.get('datafilename')
```

```
    itemname = risk_conf.get('itemname')
    x_axis_title = risk_conf.get('x-axis-title')
    y_axis_title = risk_conf.get('y-axis-title')
    title = risk_conf.get('title')
    bubble_standard_size = int(risk_conf.get('bubble-standard-size'))

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

    print(directory)

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

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

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

    # fig_dir = os.path.join(os.path.dirname(os.getcwd()), 'src', 'source')
    fig = plt.figure()
    plt.subplots_adjust(wspace=0, hspace=0)
    plt.xticks([])
    plt.yticks([])
    plt.xlim(0, 5)
    plt.ylim(0, 5)
    plt.xlabel(x_axis_title)
    plt.ylabel(y_axis_title)
    plt.title(title)

    #This example is for a 5 * 5 matrix
    nrows=5
    ncols=5
    axes = [fig.add_subplot(nrows, ncols, r * ncols + c + 1) for r in range(0,
   nrows) for c in range(0, ncols) ]

    # remove the x and y ticks
    for ax in axes:
        ax.set_xticks([])
        ax.set_yticks([])
```

```
        ax.set_xlim(0,5)
        ax.set_ylim(0,5)

    #Add background colors
    #This has been done manually for more fine-grained control
    #Run the loop below to identify the indice of the axes

    #Identify the index of the axes
    green = [10, 15, 16, 20 , 21] #Green boxes
    yellow = [0, 5, 6, 11, 17, 22, 23] #yellow boxes
    orange = [1 , 2, 7, 12, 13, 18, 19, 24] # orange boxes
    red = [3, 4, 8, 9, 14] #red boxes

    for _ in green:
        axes[_].set_facecolor('green')

    for _ in yellow:
        axes[_].set_facecolor('yellow')

    for _ in orange:
        axes[_].set_facecolor('orange')

    for _ in red:
        axes[_].set_facecolor('red')


    #Add labels to the Green boxes
    # axes[10].text(0.1,0.8, '4')
    # axes[15].text(0.1,0.8, '2')
    # axes[20].text(0.1,0.8, '1')
    # axes[16].text(0.1,0.8, '5')
    # axes[21].text(0.1,0.8, '3')

    #Add labels to the Yellow boxes
    # axes[0].text(0.1,0.8, '11')
    # axes[5].text(0.1,0.8, '7')
    # axes[6].text(0.1,0.8, '12')
    # axes[11].text(0.1,0.8, '8')
    # axes[17].text(0.1,0.8, '9')
    # axes[22].text(0.1,0.8, '6')
    # axes[23].text(0.1,0.8, '10')

    #Add lables to the Orange boxes
    # axes[1].text(0.1,0.8, '16')
    # axes[2].text(0.1,0.8, '20')
    # axes[7].text(0.1,0.8, '17')
    # axes[12].text(0.1,0.8, '13')
    # axes[13].text(0.1,0.8, '18')
```

```
    # axes[18].text(0.1,0.8, '14')
    # axes[19].text(0.1,0.8, '19')
    # axes[24].text(0.1,0.8, '15')

    #Add lables to the Red Boxes
    # axes[3].text(0.1,0.8, '23')
    # axes[8].text(0.1,0.8, '21')
    # axes[4].text(0.1,0.8, '25')
    # axes[9].text(0.1,0.8, '24')
    # axes[14].text(0.1,0.8, '22')

    # run throuh datas and generate axis datas
    dict_bubble_axis = dict()
    bubble_axis = list()
    for datasets in datas:
        # get the datas
        riskid = datas.get(datasets).get('risk-id')
        x_axis = int(datas.get(datasets).get('x-axis'))
        y_axis = int(datas.get(datasets).get('y-axis'))
        axis_point = matrix.get((x_axis, y_axis))
        x_axis_text = float(datas.get(datasets).get('x-axis-text'))
        y_axis_text = float(datas.get(datasets).get('y-axis-text'))
        x_axis_bubble = float(datas.get(datasets).get('x-axis-bubble'))
        y_axis_bubble = float(datas.get(datasets).get('y-axis-bubble'))
        bubble_axis.append(axis_point)

        # merge riks if two or more risks share the same axispoint
        if dict_bubble_axis.get(axis_point):
            risktag = dict_bubble_axis.get(axis_point).get('risk')
            risktag = risktag + '/' + riskid
            x_axis_text = x_axis_text + 0.25
            y_axis_text = y_axis_text - 0.5
            bubble_size = bubble_standard_size * 2
        else:
            risktag = itemname + riskid
            bubble_size = bubble_standard_size
        dict_axis_value = dict()

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

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

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

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

"""
Config File:
    1.  Name
    2.  Startpoint Directory
    3.  Destination Dir
    4.  Alternate Path
    5.  Data File Name
Data File:
    1.  Spalte:  Nummer
    2.  x-achse
    3.  x-achse
"""

"""
    Matrix
    This Matrix translate the x/y axis from a given risk matrix csv to the axispoint.

    The key of each axispoint is an integer tupel (x, y)
    So, you can access the axis point this way:
    <axispoint> = matrix.get((<x_axis>, <y_axis>))
"""
matrix = {
     # first column
     (1, 1):20,
     (1, 2):15,
     (1, 3):10,
     (1, 4):5,
     (1, 5):0,
     # second column
     (2, 1):21,
     (2, 2):16,
     (2, 3):11,
     (2, 4):6,
     (2, 5):1,
```

```
    # third column
    (3, 1): 22,
    (3, 2): 17,
    (3, 3): 12,
    (3, 4): 7,
    (3, 5): 2,
    # fourth column
    (4, 1): 23,
    (4, 2): 18,
    (4, 3): 13,
    (4, 4): 8,
    (4, 5): 3,
    # fifth column
    (5, 1): 24,
    (5, 2): 19,
    (5, 3): 14,
    (5, 4): 9,
    (5, 5): 4
}

# load the configuration file
riskmatrix_conf_filename = 'conf.csv'
riskmatrix_conf_dir = 'source/configuration/'
conf_riskmatrix_path = os.path.join(os.path.dirname(os.getcwd()),
    riskmatrix_conf_dir)
conf_csv_path = os.path.join(conf_riskmatrix_path, riskmatrix_conf_filename)
with open(conf_csv_path) as f:
    csv_list = [[val.strip() for val in r.split(",")] for r in f.readlines()]

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

for risks in conf:
    riskmatrix(risks, conf, matrix)
# data = pd.read_csv('/home/itgramic/LaTex/riskmatrix/src/source/riskmatrixproblem
    .csv', header=None, dtype={0: str}).set_index(0).squeeze().to_dict()
```

Listing 14: Python LaTex - riskmatrix - Risxikomatrizen

VIII Disposition

ibW Höhere Fachschule Südostschweiz

Disposition Diplomarbeit Technik und Wirtschafts-informatik 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

Disposition Diplomarbeit

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.

Inhaltsverzeichnis

Abkürzungen **3**

1 Ausgangslage und Problemstellung **1**
1.1 Das Kantonsspital Graubünden 1
1.2 Die ICT des Kantonsspital Graubünden 3
1.3 Rolle in der ICT vom Kantonsspital Graubünden 5
1.4 Ausgangslange 5
1.5 Problemstellung 9

2 Ziele und Nutzen **13**

3 Abgrenzungen **19**

4 Abhängigkeiten und Risiken **22**
4.1 Abhänigkeiten 22
4.2 Risiken 24

5 Geplante Vorgehensweise **28**
5.1 Arbeitsmethodik 28
5.2 Arbeitsschritte 28
5.3 Instrumente und Werkzeuge 28

6 Provisorisches Inhaltsverzeichnis der DA **29**

7 Bewilligung (wird durch die Schulleitung ausgefüllt) **31**

Abbildungsverzeichnis **32**

Tabellenverzeichnis **33**

Literatur **34**

Glossar **37**

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

1 Ausgangslage und Problemstellung

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[16]

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[28]

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[11]

1.2 Die ICT des Kantonsspital Graubünden

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

Abbildung 1.4: Organigramm Kantonsspital Graubünden

Die ICT betreibt über 400 Applikationen die auf mehr als 1055 physische und virtuelle Server und Appliances. Das Rückgrat der Infrastruktur ist dabei die Virtualisierungsplattformen VMware ESXi für Server und Citrix für die Thinclients der Enduser. Es werden aber auch Dienstleistungen für andere Spitäler und Kliniken oder andere Einrichtungen des Gesundheitswesens erbracht.

Entsprechen wurde die ICT in ein Applikationsmanagement, ein Infrastrukturmanagement sowie einem unterstützenden Bereich aufgegliedert. Das Applikationsmanagement wurde in je einen Bereich für die Administrativen und Medizinischen Applikationen aufgeteilt. Das Infrastrukturmanagement wiederum wurde in den Bereich Netzwerk und Data Center, welcher für Server zuständig ist, aufgeteilt. Der Bereich Business- und Prozessunterstützung beinhaltet je eine Abteilung für die Businessanalyse, das Projektmanagement und Benutzer- und Clientservices in der auch der Service-Desk untergebracht ist.

Abbildung 1.5: Organigramm Departement 10 - ICT

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

1.3 Rolle in der ICT vom Kantonsspital Graubünden

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

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

1.4 Ausgangslange

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

Basierend auf der Liste *DB-Engines Ranking*[8] 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	Summe RDBMS / Cluster / CDB / Instance	Summe Databases
MariaDB	2	2
MongoDB	2	2
MySQL	28	50
Oracle Database	27	30
PostgreSQL	20	20
Redis	1	1
Gesamtergebnis	**80**	**105**

Tabelle 1.2: Datenbankinventar

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

OS / RDBMS	Summe RDBMS / Cluster / CDB / Instance	Summe Databases
HP-UX	**21**	**24**
Oracle Databases	21	24
Linux	**26**	**48**
MariaDB	2	2
MySQL	14	36
Oracle Database	1	1
PostgreSQL	8	8
Redis	1	1
Windows Server	**33**	**33**
MongoDB	2	2
MySQL	14	14
Oracle Databases	5	5
PostgreSQL	12	12
Gesamtergebnis	**80**	**105**

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.

1.5 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	Zielwert SM	Massnahme
1	I	G0.22	Manipulation von Informationen	Durch veraltete Systeme die zudem unterschiedlich gut gehärtet und gesichert sind (z.B. durch Verschlüsselung des Verkehrs oder der Daten auf dem Storage), besteht das Risiko das Daten manipuliert werden	Die Auswirkungen reichen von einer Fehlfunktion des Systems bis hin zum vollständigen Verlust der Integrität der Daten	2	4	Ja	1	2	Best-Practice bei Härtung der Systeme. Redundanzen einführen
2	A	G0.25	Ausfall von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Ausfallen	Sofern keine HA-Architektur aufgebaut wurde, ist die Verfügbarkeit ernsthaft gefährdet resp. die Applikation steht nicht mehr zur Verfügung.	4	4	Ja	2	2	Redundanzen einführen
3	C, I, A	G0.26	Fehlfunktion von Geräten oder Systemen	Manche Datenbanken und deren Betriebssysteme sind sehr alt und sehr lange im Einsatz. Einige dieser Systeme sind schon so alt, das keine Hotfixes, Patches und Updates mehr erhältlich sind. Hierdurch entsteht das Risiko, das Systeme Fehlfunktionen erleiden.	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.	2	4	Ja	2	2	Systeme zentralisieren Lifecycle etablieren
4	C, I, A	G0.27-1	Ressourcenmangel (personelle Ressourcen)	Allerdings versuchen Datenbanksysteme, die Auswirkungen so gering wie möglich zu halten. Aufgrund der sehr heterogenen Landschaft ist der Administrationsaufwand für die jetzigen Systeme sehr gross. Zu gross, als das für jede Datenbank und deren Betriebssystem die notwendige Zeit für eine bedarfsgerechte Administration erbracht werden kann.	Daher sind sowohl Vertraulichkeit, Integrität und Verfügbarkeit gefährdet. Die Auswirkungen können vielfältig sein, abhängig davon welcher Aspekt unter dem Ressourcenmangel leidet.	3	3	Ja	2	3	Systeme zentralisieren
5	A	G0.27-2	Ressourcenmangel (technische 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	Grundsätzlich wird aber sowohl die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet.	2	2	Ja	1	2	Monitoring verschärfen
6	C, I, A	G0.31	Fehlerhafte Nutzung oder Administration von Geräten und Systemen	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.	4	3	Ja	2	3	Systeme zentralisieren
7	C, I, A	G0.32	Missbrauch von Berechtigungen	Durch die Vielfalt an Datenbankversionen und Betriebssystemen und Plattformen wonach diese betrieben werden, besteht allen voran das Risiko einer Fehlerhafter Administration und Konfiguration.	Doch die folgen bleiben nichtsdesto trotz überschaubar. Abhängig davon, welche Fehler gemacht wurden können die Auswirkungen auch stark variieren. Sie reichen von fehlender Verschlüsselung bis hin zu nicht vorhandenem Backup mit nicht mehr gesicherter Wiederherstellbarkeit von Systemen. Daraus erschliesst sich das auch bei diesem Risiko die Vertraulichkeit, Integrität und Verfügbarkeit gefährdet ist.	2	4	Ja	2	2	Systeme zentralisieren Übergreifendes Berechtigungskonzept einführen Monitoring der Zugriffe
8	A, I	G0.45	Datenverlust	Obwohl das Microsoft Active Directory die Zentrale Benutzerverwaltung ist, sind die wenigsten Datenbanken an dieses angeschlossen. Hinzu kommt der umstand, das in der Vergangenheit jeder Softwarelieferant sein eigenes Benutzerkonzept mitgebracht hat, auch bei den Datenbankzugängen. Multipliziert mit der Anzahl der unterschiedlichsten Datenbanken, Betriebssystemen und Applikationen entsteht das Risiko, das Berechtigungen Wissentlich oder Unwissendlich missbraucht werden. Verschiedene Datenbanken sind Standalone Cluster (Instanzen) welche über keinen Failover-Mechanismus verfügen.	Der Wissentliche oder Unwissentliche Missbrauch von Berechtigungen kann verheerende Auswirkungen haben. Unter anderem können Daten missbräuchlich abgezogen werden, Daten manipuliert oder das ganze System komplett zerstört werden.	2	4	Ja	2	2	Systeme zentralisieren Einheitliches Backupkonzept Regelmässige Restore-Tests
				Zudem wurden die meisten Datenbanken nur mittels Snapshots oder einem Filesystem Backup gesichert, nicht über eine eigentliche Sicherung mittels WAL. Gerade die fehlende WAL-Archivierung führt im Backupfall dazu, das alle Transaktionen die zwischen dem letzten Backup nicht mehr vorhanden sind. Hinzu kommt, das für die meisten Datenbanken hohe Sicherungsintervalle von einmal pro Stunde oder gar nur einmal am Tag gewählt wurde. Ein weiterer Aspekt des Risikos besteht in der tatsache, das aufgrund der grossen Anzahl Datenbanken und deren Heterogenität nur wenige Backups auch wirklich regelmässig geprüft werden.	Aus dem Risiko ergeben sich zwei Auswirkungen, die aber beide ein hohes Mass an Schaden verursachen können. Ersteres 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 zweigend Konsistent.	4	5	Ja	1	3	

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

2 Ziele und Nutzen

Das administrieren einer PostgreSQL Datenbank umfasst i.d.R. [20, 23] 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 Tablespaces und Filesystem muss deshalb Überwacht und ggf. erweitert werden. Läuft eine Disk voll kommt es im besten Fall zu einem Stillstand der DB, im schlimmsten Fall zu Inkonsistenzen und Datenverlust 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 Tablespaces genügend Platz haben. Dazu gehört auch das Überwachen des Logs und entsprechende Schritte im Fehlerfall.	Mittel
5	Statistiken / Cleanup Jobs justieren	PostgreSQL sammelt Statistiken um SQL Queries optimaler ausführen zu können. Zudem wird im Rahmen des gleichen Sheduled Tasks ein Cleanup Vorgenommen, so dass z.B. gelöschte Datensätze den Disk Space nicht sinnlos belegen. Die Konfiguration dieser Jobs muss an der Metrik der Datenbank angepasst werden, weil gewisse Tasks dann entweder viel zu oft oder viel zu wenig bis gar nicht mehr ausgeführt werden.	Mittel
6	SQL optimierungen	In PostgreSQL können inperfomante SQL Statements ausgelesen werden und zum Teil werden auch informationen zum Tuning geliefert[6]. 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[1]. 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 2.1: Administrative Aufgaben

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

Nr.	Aufgabe	Wichtigkeit	Zu automatisierender Task	Priorität	Muss / Kann	Spätester Termin
1	Failover	Hoch	Automatisierter Failover auf mindestens einen Sekundären DB-Node	1	Muss	Abgabe
2	Failover Restore	Hoch	Sobald der Primäre DB-Node wieder vorhanden ist, muss automatisch auf den Primären DB-Node zurückgesetzt werden.	1	Muss	
3	Filesystem Management	Hoch	Das Filesystem muss beim erreichen von 95% Usage automatisiert vergrössert werden. Die Vergrösserung muss anhand der Wachstumsrate (die mittels Linux Commands zu ermitteln ist), vergrössert werden	4	Kann	
4	Monitoring	Mittel	Der Status der Clusterumgebung und der Replikation muss im PRTG überwacht werden	2	Muss	
5	Statistiken / Cleanup Jobs justieren	Mittel	Regelmässig müssen die Parameter für den AUTOVACUUM Job berechnet werden und das Configfile postgresql.conf automatisch angepasst werden 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[6].	2	Muss	
7	Health Checks und Aktionen (Maintenance)	Hoch	Tabellen und Indizes können sich aufblähen (bloaded table / bloaded index) Ist ein Index aufgebläht, kann dies mittels eines REINDEX mit geringem Impact auf die Datenbank gelöst werden[1].	2	Muss	
8	Housekeeping	Mittel	Log Rotation muss aktiviert werden und alte Logs regelmässig gelöscht werden.	3	Muss	
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 2.2: Automatisierung Administrativer Aufgaben

Mit der Arbeit sollen folgende Ergebnisse und Resultate erzielt werden:

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

Daraus ergeben sich folgende Ziele:

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

Tabelle 2.3: Ziele

3 Abgrenzungen

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

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

Tabelle 3.1: Gegebene Systeme

Daraus ergeben sich nach nach Züst, Troxler 2002[39] folgende Abgrenzungen:

Abbildung 3.1: Systemabgrenzung

4 Abhängigkeiten und Risiken

4.1 Abhänigkeiten

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

Objekt	Abhängigkeit	Beschreibung	Status	Risiko	Impact
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.
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
Log Management / SIEM System	Sichern der Logfiles für Log Rotation	Ein Log Management System / SIEM muss vorhanden sein, um Logs langfristig sichern zu können.	Die bestehenden Plattformen für das 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.
HP-UX Ablösеprojekt	Ressourcen	Das Projekt zur Ablösung der HP-UX Plattform für die Oracle Datenbanken geht in die Konzeptions- und Umsetzungsphase.	Das Projekt geht in die Konzeptions- und Umsetzungsphase.	Als Oracle DBA bin ich stark in das Projekt eingebunden. Es besteht, dass Risiko eines Ressourcenengpasses	Projekt kann nicht Zeitgemäss abgeschlossen werden
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.

Tabelle 4.1: Abhängigkeiten

4.2 Risiken

Zusätzlich wurde eine SWOT-Analyse-Analyse für das Projekt erstellt, um weitere Risiken und Gefahren für das Projekt aufzudecken. Dabei bezieht sich die Externe Betrachtung auf die Umsysteme und die ICT des KSGR und die Interne Betrachtung auf mich und das Team um mich herum.

Abbildung 4.1: SWOT-Analyse Projekt

Aus den Abhängigkeiten und der SWOT-Analyse-Analyse wurden folgende Risiken identifiziert:

Identifikation				Abschätzung		Behandlung			
						Massnahmen ergreifen?	Zielwert		Massnahme
ID	Risiko	Beschreibung / Ursache	Auswirkung	WS	SM		WS	SM	
1	Fehlende Ressourcen	Viele parallele Projekte, Aufträge und der Tagesbetrieb	Ressourcen während der Diplomarbeit sind knapp bemessen	3	4	Ja	2	2	Organisation und Selbstmanagement
2	HP-UX Ablösеprojekt	Das Projekt ist sehr Umfangreich und ist in die Konzeptions- und Umsetzungsphase gestartet	Das Projekt wird parallel zur Diplomarbeit sehr viele Ressourcen und Aufmerksamkeit binden	4	4	Ja	3	3	Ressourcen reservieren
3	Alte Infrastruktur kann ungeplant sämtliche Ressourcen binden	HP-UX Plattform, DELL NetWorker / Data Domain Umgebung und HPE 3PAR SAN Storage Umgebung sind über dem Lifecycle und haben in den vergangenen Monaten immer wieder kritische Ausfälle erlebt	Bei einem Event, ausgelöst durch das Alter der HP-UX Plattform, der DELL NetWorker / Data Domain Umgebung oder dem SAN Storage, kann der ganze Betrieb zum erliegen kommen und entsprechend viele Ressourcen aufgrund der kritikalität binden	4	4	Ja	3	3	Monitoring vorgängig ausbauen und Massnahmen definieren
4	Schwächen beim Selbstmanagement und in der Selbstorganisation	Selbstmanagement und Organisation ist nicht meine Stärke	Das Projekt verzettelt sich, Zeit geht verloren. Auch eine folge könnte der Scope Verlust sein	3	3	Ja	2	2	Werkzeuge im Vorfeld definieren und bereitstellen
5	Scope verlust während des Projekts	Der Scope kann während des Projekts verloren gehen	Verzettelung und Zeitverlust bis hin zu scheitern	3	4	Ja	2	3	Ziele klar definieren
6	Scope Creep	Der Umfang kann stark steigen wenn Ziele nicht genau genug definiert wurden	Zeitverlust bis hin zu scheitern des Projekts	3	4	Ja	3	3	Ziele SMART definieren
7	SIEM / Log Plattform nicht betriebsbereit	Die öffentliche Ausschreibung für die neue / Log Plattform wurde erst am 23.10.2023 veröffentlicht. Bis zur Implementation kann noch Zeit vergehen.	Logs müssen länger auf dem System selber vorgehalten werden. Zudem müssen ggf. eigene Massnahmen zum Auslesen von Logs getroffen werden	4	1	Nein			
8	Foreman nicht betriebsbereit	Die Foreman Provisioning- und Lifecycle Plattform befindet sich aktuell erst in der Proof of Concept Phase. Dadurch besteht das Risiko, dass sie nicht betriebsbereit zum Start der Diplomarbeit ist	Ms müssen von Hand provisioniert werden. Dies bedeutet einen massiven Mehraufwand und verzögert ggf. die Evaluationsphase und mit sicherheit die Installationsphase	3	5	Ja	3	4	Massnahmen ergreifen um die manuelle Installation so effizient wie möglich zu gestalten.

Tabelle 4.2: Risiko-Matrix der Diplomarbeit

Daraus ergibt sich folgende Risikomatrix

Abbildung 4.2: Projektrisiken

Mit den entsprechenden Massnahmen können die Risiken gesenkt werden:

Abbildung 4.3: Projektrisiken mit Massnahmen

5 Geplante Vorgehensweise

5.1 Arbeitsmethodik

Als Arbeitsmethodik soll IPERKA verwendet werden.

5.2 Arbeitsschritte

Die Umsetzung des Projekts sind die Evaluation, Installation einer Testumgebung und anschliessend das Testing zentrale bestandteile. Innerhalb der Evaluation sollen die Lösungen analysiert und anschliessend auch in Development-Umgebungen installiert werden, sofern sich die Eignung nicht bereits bei der analyse ausschliessen lässt, um Erfahrungen zu sammeln um Vergleichswerte sammeln zu können.

Die Testumgebung muss anschliessend installiert und konfiguriert werden. Vor dem Testing muss die Konfiguration reviewed werden.

In der Testphase muss die Lösung anschliessend anhand der Anforderungen und Testszenarien getestet werden.

5.3 Instrumente und Werkzeuge

Für die Zeitplanung muss ein Gannt-Diagramm mit Microsoft Project verwendet werden. Anhand des Gannt-Diagramms muss ein Netzplan mit einem kritischen Pfad erstellt und geführt werden, um das Zeitmanagement führen zu können. Ein Arbeitsfortschrittsdiagramm kann geführt werden.

Für die Bewertungen der Lösungen ist eine Nutzwertanalyse angedacht. Die Gewichtung muss mittels der Präferenzmatrix ermittelt werden.

6 Provisorisches Inhaltsverzeichnis der DA

Die Diplomarbeit soll in folgende Teile gegliedert werden:

- Titelblatt
- Management Summary
- Inhaltsverzeichnis
- Abkürzungen

1. Einleitung
   - 1.1. Ausgangslage und Problemstellung
   - 1.2. Zieldefinition
   - 1.3. Abgrenzung
   - 1.4. Vorgehensweise und Methoden
   - 1.5. Risikomanagement
   - 1.6. Projektmanagement
2. Umsetzung
   - 2.1. Evaluation
     - 2.1.1. Erheben und Gewichten der Anforderungen
     - 2.1.2. Testziele erarbeiten
     - 2.1.3. Analyse gängiger PostgreSQL HA Cluster Lösungen
     - 2.1.4. Installation verschiedener Lösungen
     - 2.1.5. Gegenüberstellung der Lösungen
     - 2.1.6. 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

i. Verzeichnisse

i.i. Literatur- und Quellverzeichnis

i.ii. Tabellenverzeichnis

i.iii. Abbildungsverzeichnis

i.iv. Glossar

i.v. Index

ii. Anhang

7 Bewilligung (wird durch die Schulleitung ausgefüllt)

Das Thema dieser Diplomarbeitsdisposition ist bewilligt.

Ort und Datum	Unterschrift Fachvorsteher
Sargans, 4.12.2023	

Anmerkungen und Hinweise zur Disposition:

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

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

2 Ziele und Nutzen

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

3 Abrenzungen

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

4 Abhängigkeiten und Risiken

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

Abbildungsverzeichnis

1.1 Spitalregionen Kanton Graubünden[16] 1
1.2 Wahlkreise Kanton St. Gallen[28] 2
1.3 Spitalregionen / Spitalstrategie Kanton St. Gallen[11] 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

3.1 Systemabgrenzung 21

4.1 SWOT-Analyse Projekt 24
4.2 Projektrisiken 26
4.3 Projektrisiken mit Massnahmen 27

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

2.1 Administrative Aufgaben 14
2.2 Automatisierung Administrativer Aufgaben 16
2.3 Ziele 18

3.1 Gegebene Systeme 20

4.1 Abhängigkeiten 23
4.2 Risiko-Matrix der Diplomarbeit 25

Literatur

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

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

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

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

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

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

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

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

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

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

[11] 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-spi spitaeler-kliniken/spitalzukunft.html.

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

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

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

[15] 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.

[16] 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.

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

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

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

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

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

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

[23] 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.

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

[25] 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.

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

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

[28] 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.

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

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

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

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

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

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

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

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

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

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

[39] Rainer Züst. "Einstieg ins Systems Engineering". In: (2002).

Glossar

Ansible Ansible ist ein Open-Soure Automatisierungstool zur Provisionierung, Konfiguration, Deployment und Orchestrierung. Ansible verbindet sich auf auf die Zielgeräte und führt dort die Hinterlegten Module aus. Oft werden die verschieden Aufgaben in einem Skript, in einem sogenannten Playbook geschrieben werden[24].. 20

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[17].. 16, 18

Cassandra Cassandra ist eine Spaltenorganisierte NoSQL-Datenbank die 2008 veröffentlicht[30] wurde.. 7

CI/CD Continuous Integration/Continuous Delivery bedeutet, das Anpassungen kontinuirlich in die Entwicklungsumgebungen integriert und auf die Zielplattformen verteilt werden[25].. 3

DBMS Ein Database Management System regelt und organisiert die Datenbasis einer Datenbank[31].. 3

Debian Debian gehört nebst Slackware Linux zu den ältesten Linux Distribution die noch immer gepflegt und eingesetzt werden. Sie wurde im August 1993 gestartet und brachte im Laufe der Zeit einige der beliebstesten Distributionen wie Ubuntu hervor.. 20

Elasticsearch Elasticsearch ist eine 2010 veröffentlichte Open-Source Suchmaschine die auf Basis von JSON-Dokumenten und einer NoSQL-Datenbank arbeitet[26].. 7

Failover In einem Fehlerfall wird in einem HA-System meist ein Primary Node auf den Secondary ungeplant geswitched.. 18

Foreman Foreman ist ein Lifecycle Management und Provisioning System für Virtuelle und Physische Server. Ab Version 6 basierte der Red Hat Satellite auf Foreman. 20, 23, 25

Git Git ist eine Versionierungssoftware und bietet die Möglichkeit, Repositories erstellen zu können. Die Repositories sind dabei nicht zentral sondern dezentral organsiert und arbeiten daher mit Working Copies von Repositories[12, 38].. 37

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[2].. 18, 23

Harbor Harbor ist ein Open-Source-Tool zur Registrierung von Richtlinien rollenbasierten Zugriffssteuerung[10]. Harbor wird beim KSGR zur Verwaltung der Kubernetes-Plattform verwendet.. 18

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.. 3, 5, 8, 23, 25

IBM DB2 IBM DB2 ist eine Relationale Datenbank[21] deren Vorläufer System-R von IBM zwischen 1975 und 1979 entwickelt wurde. DB2 selber wurde 1983 von IBM veröffentlicht.. 7

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.. 3, 8, 20, 38

Linux Linux ist ein Open-Source Betriebssystem, welches von Linus Torvalds 1991 in seiner frühesten Form entwickelt wurde und lose vom UNIX Derivat MINIX inspiert war. Linux besteht heute aus einer enorm grossen Anzahl an Distributionen und läuft auf einer grossen Anzahl von Plattformen.. 5, 8, 38

MariaDB MariaDB ist ein MySQL Fork des ehemaligen MySQL Mitbegründers Michael Widenius, wobei sich der Name Maria aus dem VOrnamen einer seiner Töchter ableitet. NAch dem Fork 2009 blieb MariaDB für eine Zeitlang sehr ähnlich mit MySQL und behielt ein ähnliches Versionierungsschema bei. Dies änderte sich 2012 wo dann direkt mit der Version 10 weitergefahren wurde. Beide Datenbanken entfernen sich im Lauf eder Zeit immer mehr voneinander undf sind nicht mehr in jdem Fall kompatibel oder beliebig austauschbar. Auf den Linux Distributionen tratt MariaDB die Nachfolge von MySQL als Standard Datenbank an.. 5, 7, 8, 9

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[3].. 7

Microsoft Access Access wurde 1992 veröffentlicht und ist Entwicklungsumgebung, Front- und Backend-Software und Relationale Datenbank in einem[4].. 7

Microsoft SQL Server MS SQL Server ist das RDBMS von Microsoft[5]. Nebst Microsoft Windows und Windows Server lässt es sich seit Version 2014 ebenfalls auf Linux Betreiben. In der Wirtschaft ist die primäre Plattform aber Windows Server.. 5, 7, 39

MongoDB MongoDB ist eine dokumentenorientierte NoSQL-Datenbank, die zurm ersten Mal 2007 veröffentlicht wurde[32].. 7, 8

MySQL Die Datenbank MySQL wurde Ursprünglich als reine Relationale Open-Source Datenbank von Firma MySQL AB 1994 Entwickelt. Der Name My leitet sich vom Namen My der Tochter

des Mitbegründers Michael Widenius ab. Als Sun Microsystem 2008 MySQL übernahm, hielt sich die Option frei, bei einem Kauf von Sun Microsyszem durch Oracle gründen zu dürfen. Seit Oracle Sun Microsystem 2010 gekauft hat, wurden immer mehr Funktionalitäten von der Community Edition zu der Enterprise Edition verschoben worden. Aus diesem Grund hat heute der MySQL Fork MariaDB MySQL mehrheitlich aus allen Linux Distributionen als Standard Datenbank verdrängt.. 5, 7, 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, 37, 38, 40

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[14, 13]. OLAP-Systeme bieten eine Hochperformante Analyse grosser Datenmengen und sind oftmals zentraler Teil eines Data-Warehouses.. 3, 7

Oracle Linux Oracle Linux ist eine RHEL-Distribution der Firma Oracle und ist mit RHL Binärkompatibel. Sie wird primär für den Betrieb von Oracle Datenbanken verwendet und komnt auf den Oracle Eigenenen Appliances ODA und Exadata zum Einsatz. Für den Zweck als DB Plattform kann ein für Oracle Datenbanken optimimierter Kernel verwendet werden. Zu Oracle Linux kann ein kostenpflichtiger Support bezogen werden, allerdings ist die Distribution anders als RHEL auch ohne Lizenz erhältlich.. 20

Oracle Database Die erste verfügbare Version der Oracle Datenbank kam im Jahr 1979 mit Version 2 (statt Version 1) heraus, damals allerdings nur mit den Basisfunktionen. Im Laufe der Zeit wuchs der Funktionsumfang sehr stark an, die Grundlage des Client-Server-Designs kam erstmals im Jahr 1985 mit Version auf den Markt und hat sich im Prinzip bis heute gehalten. Mit der mit Version 8/8i 1997 erschienen Optimizer und mit der Version 9i 2001 erschienenn Flashback-Funktionalität (die ein schnelles Online Recovery sowie einen Blick in die Vergangheit ermöglichen) konnte Oracle sich stark von der Konkurenz absetzen. Heute gilt die Datenbank als erste Wahl, wenn es um Hochverfügbare Systeme, hohe Perforamce oder grosse Datenmengen geht.. 5, 7, 8, 39

PostgreSQL Die OpenSource Datenbank PostgreSQL wurde in Form von POSTGRES zum ersten Mal 1986 von der University of California at Berkeley veröffentlicht. und zählt zu den beliebstesten OpenSource Datenbanken. Zudem besteht in vielen bereichen eine gewisse ähnlichkeit zu Oracles Oracle Database.. 5, 7, 8, 9, 13, 14

PostgreSQL HA Cluster Der HA Cluster des PostgreSQL Clusters. 18, 29

PostgreSQL Cluster Ein PostgreSQL Cluster entspricht einer Instanz bei MS SQL oder einer Container Database wei Oracle.. 17, 18, 39

PRTG Das Monitoring System Paessler Router Traffic Grapher der Firma Paessler wurde 2003 zum erstmals veröffentlicht und war ebenfalls als Netzwerkmonitoring System konzipiert. Wie bei Zabbix lässt sich heute damit ebenfalls fast jedes IT-System damit Überwachen. Reichen die Zahlreich vorhanden Standard Sensoren nicht, können eigene Sensoren geschrieben werden. PRTG ist nicht Open-Source, man bezahlt anhand gewisser Sensor Packages.. 3, 5, 16, 20

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[9].. 3, 8

RedHat Enterpise Linux (RHEL) RHEL wurde in seiner Ursprüglichen Form Red Hat Linux (RHL) bis in den Oktober 1994 zurück, wobei die erste Version von RHEL wie es heute existiert im Jahr 2002 erfolgte. RHEL ist auf lange Wartungszyklen von fünf Jahren und grosskunden ausgelegt. Ohne entsprechenden Supportvertrag kann keine ISO-Datei bezogen werden. Somit hebt sich RHEL stark von aderen Linux Distributionen ab.. 20

Redis Redis ist eine Key-Value-orientierte NoSQL In-Memory-Datenbank, dh. die Daten liegen Primär im Memory und nicht auf dem Storage[35]. Redis wurde 2009 zum ersten Mal veröffentlicht.. 7, 8

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

SAN Ein Storage Area Network ist ein dediziertes Netzwerk aus Storage Komponenten. SAN Systeme bieten redundante Pools an Speicher. Die Physischen Festplatten werden zu Virtuellen Lunes, also logischen Einheiten, zusammengefasst. Dies werden nach aussen den Konsumenten präsentiert[7, 27, 15]. 3, 5, 20, 23, 25

SIEM Ein sammelt Daten aus verschieden Netzwerkkomponenten oder Geräten von Agents oder Logs. Diese Daten werden permanent analysiert und mit einem definierten Regelwerk gegengeprüft. Ziel ist es, verdächtige Events zu erkennen und einem Angriff zuvorzukommen oder ihn möglichst früh zu unterbinden[36].. 3, 20, 23

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[37, 29]. 7

Splunk Splunk ist Big Data Plattform, Monitoring- und Security-Tool in einem[19, 22]. . 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[33].. 7

Switchover In einem Maintenance-Fall in einem HA-System meist ein Primary Node auf den Secondary geplant geswitched.. 18

SWOT-Analyse Eine SWOT-Analyse soll die Stärken (Strengths), Schwächen (Weaknesses), Chancen (Opportunities) und Risiken (Threads) für ein Unternehmen oder ein Projekt aufzueigen. Anhand einer SWOT-Analyse werden i.d.R. anschliessend Strategien abgeleitet um mit den Stärken und Chancen die Schwächen und Risiken abzufangen oder anzumildern.. 3, 24

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[34, 18].. 20

UNIX Die erste Version von UNIX wurde im Jahr 1969 in den Bell Labs entwickelt und übernahm viele Komponenten aus dem gescheiterten Multics-Projekt. Aus dem Ursprünglichen UNIX enstanden im Laufe der Zeit viele offene und Proprioritäre Derivate deren Einfluss weit über die Welt der Informatik reicht.. 3

Zabbix Das 2001 veröffentlichte Open-Source Monitoring System Zabbix gilt zwar als Netzwerk-Monitoring System, allerdings kann heute nahezu jedes IT-System damit überwacht werden. Zabbrix speichert die Metriken und nicht die Auswertungen, das heisst, solange die Daten vorhanden sind können Grafiken zu jedem Zeitpunkt generiert worden. Zabbix ist grundsätzlich Open-Source, man kann allerdings Supportverträge Abschliessen.. 9, 20

/

Abbildung 1: Disposition