Postgres Lifecycle Management Operators in Kubernetes

Miroslav Širina

Bachelor's thesis
2023

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miroslav Širina**
Osobní číslo:	**A20079**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Kombinovaná**
Téma práce:	**Operátoři pro správu životního cyklu databázového systému Postgres v Kubernetes**
Téma práce anglicky:	**Postgres Lifecycle Management Operators in Kubernetes**

Zásady pro vypracování

1. Seznamte se s operátory v Kubernetes a Postgres databázovým systémem.
2. Definujte životní cyklus databázového serveru.
3. Vyberte vhodné operátory a popište je.
4. Sestavte metodiku testování jednotlivých operátorů.
5. Na základě sestavené metodiky otestujte vybrané operátory.
6. Proveďte zhodnocení.

Forma zpracování bakalářské práce: **tištěná/elektronická**
Jazyk zpracování: **Angličtina**

Seznam doporučené literatury:

1. SAYFAN, Gigi, 2020. Mastering Kubernetes: level up your container orchestration skills with Kubernetes to build, run, secure, and observe large-scale distributed apps. Third edition. Birmingham: Packt Publishing. ISBN 9781839211256.
2. RIGGS, Simon a Gianni CIOLLI, 2022. PostgreSQL 14 Administration Cookbook. Birmingham: Packt publishing. ISBN 9781803248974.
3. DOBIES, Jason a Joshua WOOD. Kubernetes operators: automating the container orchestration platform. Sebastopol, CA: O'Reilly Media, 2020. ISBN 9781492048046.
4. FARLEY, David, 2021. Modern software engineering: doing what really works to build better software faster. Boston: Addison-Wesley. ISBN 9780137314911.
5. DAME, Michael, 2022. The Kubernetes Operator Framework Book. Birmingham: Packt Publishing. ISBN 9781803232850.

Vedoucí bakalářské práce: **Ing. Peter Janků, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

I hereby declare that:

- I understand that by submitting my Bachelor's thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Bachelor's Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Bachelor's Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlín.
- I am aware of the fact that my Bachelor's Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, Tomas Bata University in Zlín has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work – Bachelor's Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlín with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Bachelor's Thesis include the use of software provided by Tomas Bata University in Zlín or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Bachelor's Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Bachelor's Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- The submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlín; dated: .....

Student's Signature

ABSTRAKT

Text abstraktu česky

Klíčová slova: Přehled klíčových slov

ABSTRACT

Text of the abstract

Keywords: Some keywords

Zde je místo pro případné poděkování, motto, úryvky knih, básní atp.

TABLE OF CONTENTS

INTRODUCTION

The cloud has made our work easier. We no longer have to physically connect new machines to the network, configure network connections, add disks, or even plug in virtual ones. Kubernetes, together with the cloud, can automatically allocate new resources for our applications and, thanks to operators, can even create entire database clusters with high availability. Thanks to operators, it can also automatically set up scaling or backups. It can even restore an entire database system from a backup. This paper is focused specifically on Kubernetes operators for the popular Postgres database management system. Its goal is to find Operators for Postgres. To evaluate their pros and cons. To test them and recommend the best one.

I. THEORY

1 BACKGROUND

This chapter introduces the key technologies used in this thesis including Postgres, Kubernetes, and Kubernetes operators.

1.1 Postgres

PostgreSQL is a powerful object-relational database management system (ORDBMS) derived from the POSTGRES package written at the University of California at Berkeley. [?] [?] The first version of POSTGRES was released in June 1989. POSTGRES has been used in many applications, including financial data analysis systems, asteroid tracking databases, medical information database, and several geographic information systems. The size of external community users has nearly doubled by 1993. [?]

POSTGRES was using its POSTQUEL query language from version one until 1995, when Andrew Yu and Jolly Chen introduced SQL to POSTGRES. The name has changed to Postgres95. Postgres95 was completely ANSI C code reduced by 25 % and was 30 – 50 % faster than Postgres 4.2. [?]

It was clear by 1996 that the name would not stand the test of time therefore it has been renamed to PostgreSQL. As stated by PostgreSQL documentation [?]: "Many people continue to refer to PostgreSQL as "Postgres" (now rarely in all capital letters) because of tradition or because it is easier to pronounce. This usage is widely accepted as a nickname or alias." This thesis will use Postgres as an alias for PostgreSQL as well.

More than 30 years after the first version Postgres has been considered the most used ORDBMS for professional developers by Stack Overflow survey [?]. According to Riggs and Ciolli [?]: "The PostgreSQL feature set attracts serious users who have serious applications. Financial services companies may be PostgreSQL's largest user group, although governments, telecommunication companies, and many other segments are strong users as well." It is fully ACID compliant [?] and supports many kinds of data models such as relational, document, and key/value. [?]

1.2 Kubernetes

Kubernetes, also known as K8s, is an open-source platform for automating deployment, scaling, and management of containerized applications. It provides a way to manage and orchestrate containers, which are units of software that package up an application and its dependencies into a single, isolated package that can run consistently on any infrastructure. [?]

As described by Kubernetes Documentation [?] Kubernetes provides several key features, including:

- **Service discovery:** A container can be exposed by Kubernetes either through its DNS name or its own IP address.
- **Load balancing:** In the case of high traffic to a container, stability of the deployment can be ensured by Kubernetes load balancing and distributing the network traffic.
- **Storage Orchestration:** Storage orchestration in Kubernetes allows for the automatic mounting of a storage system of choice, including local storage, public cloud providers, and others.
- **Automated rollouts and rollbacks:** The desired state of deployed containers can be described using Kubernetes, and the actual state can be changed to the desired state at a controlled rate. For instance, the automation of Kubernetes can be utilized to create new containers for the deployment, remove existing containers, and transfer all their resources to the newly created container.
- **Automatic bin packing:** A cluster of nodes for running containerized tasks is provided to Kubernetes. The amount of CPU and memory required by each container is specified to Kubernetes. The optimal utilization of resources can be achieved by Kubernetes fitting the containers onto the nodes.
- **Self healing:** Containers that fail are restarted by Kubernetes, those that do not respond to the user-defined health check are replaced or killed, and they are not advertised to clients until they are deemed ready to serve.
- **Secret and configuration management:** Sensitive information, such as passwords, OAuth tokens, and SSH keys, can be stored and managed by Kubernetes. The deployment and updating of secrets and application configuration can be

done without the need to rebuild container images and without the exposure of secrets in the stack configuration.

1.2.1 Kubernetes Components

Kubernetes cluster is composed of a set of worker machines that run containerized applications called nodes. Each cluster must have at least one node. [?]

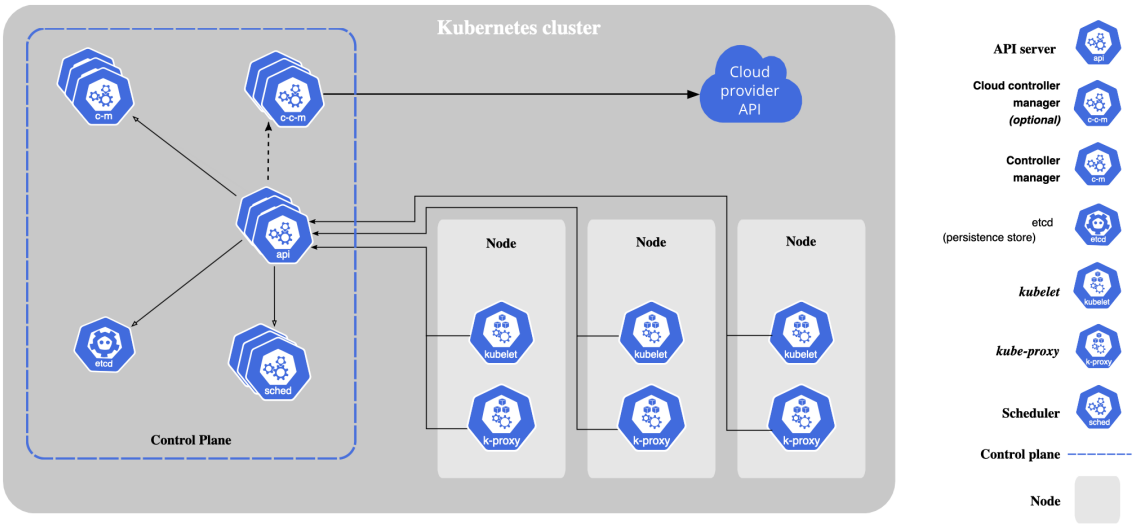

Figure 1.1 The components of a Kubernetes cluster [?]

The Kubernetes control plane is the management system of a Kubernetes cluster, responsible for maintaining the desired state of the cluster. It consists of multiple components that work together to manage the cluster and its resources, including pods, services, and volumes. The key components of control plane are [?]:

- **kube-apiserver:** Acts as the front-end for the Kubernetes API and exposes the API to other components. [?]
- **etcd:** Highly available distributed key-value store that serves as the backing store for the cluster's configuration data. [?]
- **kube-scheduler:** Assigns work to nodes in the cluster, such as scheduling pods to run on nodes. [?]
- **kube-controller-manager:** Monitors the cluster's state and makes adjustments as necessary to maintain the desired state. [?]

- **cloud-controller-manager:** Manages cloud-related tasks such as node creation and management, volume management, and load balancing, allowing the other components of the control plane to focus on their specific responsibilities. Cloud manager is optional. Can be avoided when Kubernetes not used in cloud. [?]

Node components: Node components in a Kubernetes cluster run on each node and provide crucial functionality for the operation of containers on that node. [?]

- **kubelet:** Is responsible for communicating with the control plane and ensuring that containers are running and healthy. [?]
- **kube-proxy:** Is responsible for maintaining network rules on the nodes, allowing network communication to the containers. It enables the containers in a pod to communicate with other containers and the outside world, and performs tasks such as load balancing and traffic routing. [?]
- **container runtime:** Is responsible for running containers. [?]

1.2.2 Kubernetes Concepts

Pod is the smallest deployable unit that can be created in Kubernetes. [?] A Pod in Kubernetes is comprised of multiple containers and storage volumes that are run together within the same execution environment. Pods, not individual containers, are considered to be the smallest unit that can be deployed in a cluster. As a result, all containers included in a single Pod will always run on the same machine. [?]

A Pod's specifications are outlined in a Pod manifest, which is simply a JSON or YAML text file that represents the Kubernetes API object. Kubernetes follows a declarative configuration approach, where the system's desired state is defined in a configuration file, and the service then implements the necessary changes to make the desired state a reality. [?]

ReplicaSet's purpose is to ensure a consistent number of replica Pods are running at all times. It is commonly used to guarantee a specified number of identical Pods are available. However, a Deployment is a more advanced concept that oversees ReplicaSets and provides a more streamlined way to make updates to Pods. It also offers additional features. As a result, it's advisable to use Deployments instead of directly utilizing ReplicaSets, unless you have specific update requirements or don't need to make updates at all. [?]

Service is an abstraction layer and defines a group of Pods and the method to access them (often referred to as a micro-service). The group of Pods targeted by a Service is usually specified through a selector. The Service abstraction makes this possible by enabling the decoupling of components. [?]

TBD service, service discover, volumes etc, stateful vs stateles

TBD - Read https://containerjournal.com/kubecon-cnc-eu-2022/why-run-postgres-in-kubernetes/

TBD - Read data on Kubernetes https://dok.community/dokc-2021-report/

1.3 Operators

TBD - operators capability levels TBD - where to find operators (best operator are from the developers of postgres) TBD - define Operator hub

1.4 Running Postgres in Kubernetes

TBD - statefull set, risk of loosing data without statefull set, high availability, scaling

2 DATABASE SYSTEM LIFECYCLE

The database system itself is a software like any other. It is therefore also subject to the same life cycle as software. As depicted in figure 2.1 application lifecycle consists of three main parts. It is the governance part, development, and operations. For this thesis, only the operations part is relevant because it is the only part we are able to control.

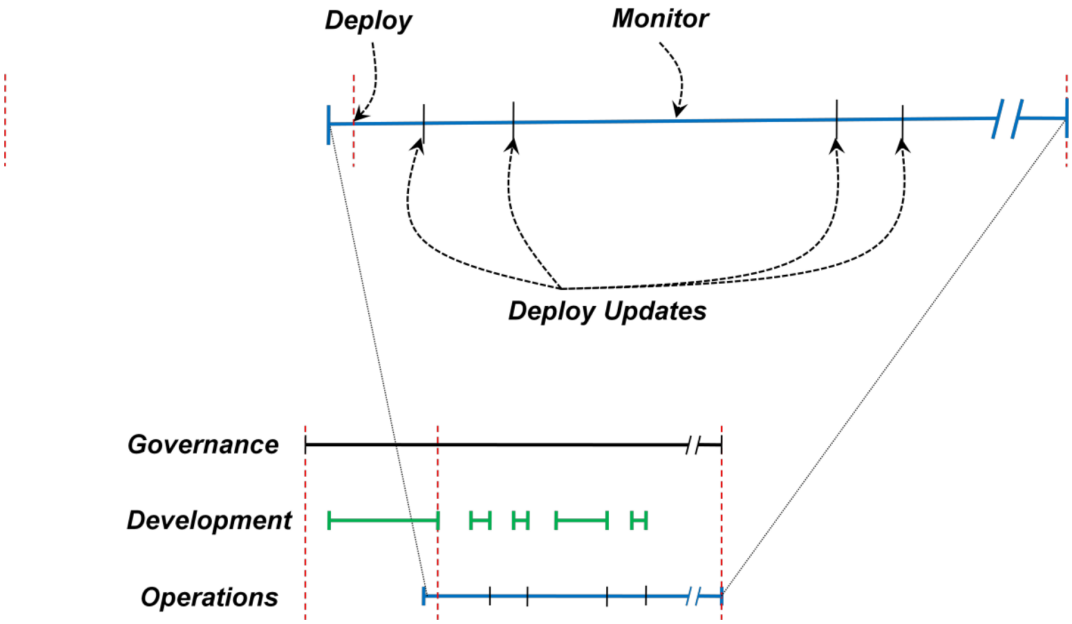

Figure 2.1 Application Life Cycle [?]

Operation is the process of running and managing the application, which starts with deployment and continues until the application is taken out of service. This aspect of the application lifecycle management covers the release of the application into production, ongoing monitoring, and other related tasks. [?]

TBD: connect to capability levels of Operator (define related tasks, chose the capability that would suit them all)

3 SEARCH FOR POSTGRES OPERATORS

TBD: connect Operators (where to find operators)

TBD: describe CloudNativePG and Patroni

TBD: describe EDB licence

TBD: each operator security Artifact hub https://artifacthub.io/packages/olm/community-operators/postgresql

TBD: What does Percona different to Crunchy to EDB?

TBD: Postgres update every three months https://access.crunchydata.com/documentation/postgr operator/5.3.0/tutorial/update-cluster/

TBD: Container types CloudNativePG is built on immutable application containers. What does it mean? https://cloudnative-pg.io/documentation/1.19/faq/

As previously stated in the section on Operators, the most suitable operators for a software are those developed by the software developers themselves, as they possess an in-depth understanding of the software. Regrettably, no such operator has been created for Postgres.

Operator Hub presents eight operators with varying levels of capabilities, including Crunchy Postgres for Kubernetes by Crunchy Data, EDB Postgres for Kubernetes by EnterpriseDB Corporation, Ext Postgres Operator by movetokube.com, Percona Operator for PostgreSQL by Percona, Postgres-Operator by Zalando SE, Postgresql Operator by Openlabs, and PostgreSQL Operator by Dev4Ddevs.com.

Deeper internet research revealed three more operators: CloudNativePG, StackGres, and Stolon. [**?**] [**?**]

Of the eleven operators available, only five meet our minimum capability requirement of Deep Insight, namely: Crunchy Postgres for Kubernetes, EDB Postgres for Kubernetes, Percona Operator for PostgreSQL, CloudNativePG Operator, and StackGres Operator. As a result, only these five will be subjected to deeper research, testing, and evaluation.

3.1 Crunchy Postgres for Kubernetes

Crunchy Postgres for Kubernetes (PGO) is a Postgres Operator provided by Crunchy Data, which offers a declarative solution for the management of PostgreSQL clusters, with a focus on automation. Crunchy Data is a company that specializes in providing open-source software solutions for Postgres. The company also provides a range of support, consulting, and training services to help organizations implement and optimize their Postgres deployment. [?]:

PGO's capabilities are the following:

- **Postgres Cluster Provisioning:** PGO is able to create [?], update [?] or delete Postgres cluster [?]
- **High Availability:** High availability is achieved by adding additional nodes. PGO uses a synchronous replication technique. [?]
- **Postgres updates:** PGO is able to apply minor patches [?], and major upgrades since version 5.1. [?]
- **Backups:** PGO backup capabilities features: automatic backup schedules, backup to multiple locations, backup to cloud providers (AWS S3, Google Cloud Storage, Azure Blob), ad hoc backups, and backup encryption. [?]
- **Disaster Recovery:** PGO is capable of Point In Time recovery, in place Point in Time Recovery, restore of an individual database. [?]
- **Cloning:** PGO is able to clone cluster. [?]
- **Monitoring:** Monitoring is provided by Prometheus, Grafana, and Alertmanager. [?]
- **Connection Pooling:** PgBouncer connection pooler from Postgres is part of PGO. [?]

The current stable version of PGO is 5.3.0 was released on 13th January 2023 and has one critical vulnerability, 7 highly critical vulnerabilities, 31 medium, and 31 low. [?] PGO is compatible with following platforms: Kubernetes 1.22-1.25, OpenShift 4.8-4.11, Rancher, Google Kubernetes Engine (GKE), including Anthos, Amazon EKS, Microsoft AKS, VMware Tanzu. [?]

PGO is distributed under the Apache License 2.0, an open-source license that allows for both commercial and non-commercial use. With regards to capability, PGO is considered to have the highest capability level, labeled as Autopilot. [?]

PGO consists of the following key components [?]:

- High Availability: Patroni
- Backups: PgBackRest
- Connection Pooler: PgBouncer
- Monitoring: PgMonitor, Prometheus, Grafana, and Alertmanager

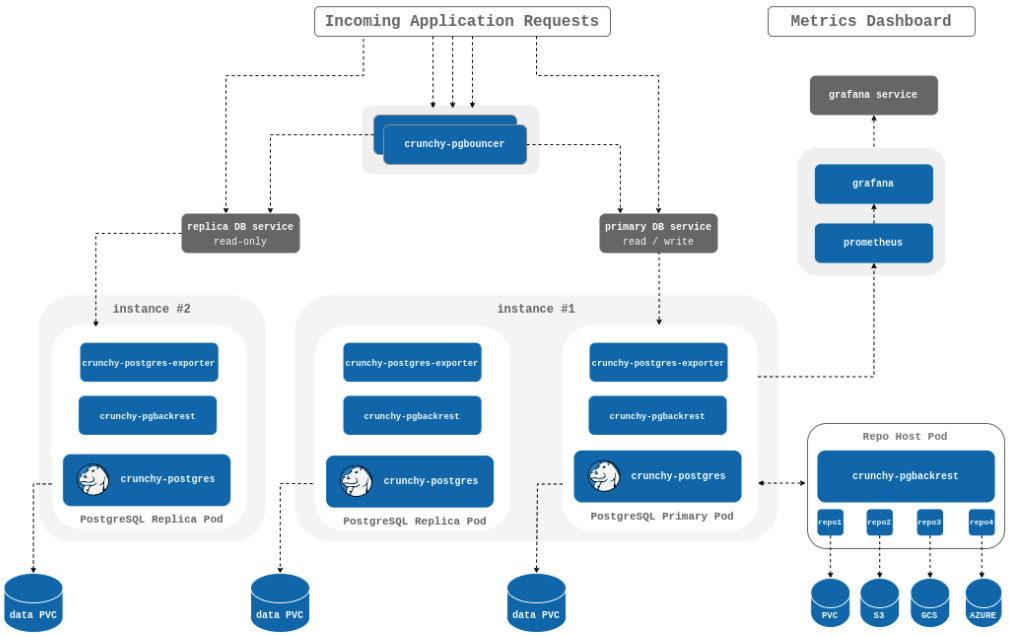

Figure 3.1 PGO's architecture [?]

3.2 EDB Postgres for Kubernetes

The EDB Postgres for Kubernetes (EDBO) is a fully supported operator that has been designed, developed, and maintained by EnterpriseDB Corporation. It provides comprehensive coverage of the entire lifecycle of highly available PostgreSQL database clusters with a primary/standby architecture, utilizing native streaming replication. The operator is based on the open-source CloudNativePG operator and offers additional benefits. [**?**]

EnterpriseDB (EDB) is a software company that provides enterprise-class PostgreSQL software and services. EDB is a leading provider of PostgreSQL technology, offering a range of products and services designed to help organizations adopt, deploy, and manage PostgreSQL databases. [**?**]

As stated previously EDBO is based on open-source CloudNativePG operator. The operator that is not listed on Operator Hub. EDBO provides additional features on top of CloudNativePG operator such as support for Oracle compatibility using EDB Postgres Advanced Server, support for additional platforms such as IBM Power, and additional enterprise-grade security features. [**?**]

EDBO is distributed under the EDB Limited Usage License Agreement, a proprietary license that is specific to software provided by EnterpriseDB Corporation. A license key is always required for the operator to work longer than 30 days. [**?**] Due to the restrictive nature of the license EDBO will no longer be subject to testing and evaluation but CloudNativePG will.

3.3 CloudNativePG

The CloudNativePG operator (CNPGO) is an operator that is available as an open-source solution and aims to manage PostgreSQL workloads across various Kubernetes clusters running in private, public, hybrid, or multi-cloud environments. The operator aligns with DevOps principles and concepts like immutable infrastructure and declarative configuration. [?]

Initially developed by EDB, CNPGO was later made available to the public as an open-source software under the Apache License 2.0. In April 2022, the project was submitted to CNCF Sandbox for further development and community engagement. [?]

CNPGO's capabilities are the following:

- **Postgres Cluster Provisioning:** CNPGO is able to create, update or delete Postgres cluster. [?]
- **High Availability:** High availability is achieved by adding additional nodes. PGO uses a synchronous replication technique. [?]
- **Direct database imports:** CNPGO provides direct database import from remote Postgres server by using pg_dump and pg_restore even on different Postgres versions. [?]
- **Postgres updates:** CNPGO is able to apply minor patches. [?] Major updates are possible by previously mentioned Direct database imports.
- **Backups:** CNPGO backup capabilities features: automatic backup schedules, backup to multiple locations, backup to cloud providers (AWS S3, Google Cloud Storage, Azure Blob), on-demand backups, and backup encryption [?][?]. Due to EDB's backup software Barman backup compression is available also. [?]
- **Disaster Recovery:** CNPGO is capable of Point In Time recovery. Database recovery was not mentioned in the documentation. [?]
- **Cloning:** CNPGO is able to create cluster replicas. [?]
- **Monitoring:** Monitoring can be provided by the additional installation of Prometheus, and Grafana, and Alertmanager. [?]
- **Connection Pooling:** Is provided by native Postgres pooler PgBouncer. [?]

- **Customization:** CNPGO provides a wide area of Postgres customization such as max parallel workers tuning or WAL configuration [?]

The current major stable version of CNPGO is 1.19 was released on 14th February 2023. CNPGO is distributed under the Apache License 2.0 open-source license. CNPGO is considered to have the highest capability level, labeled as Autopilot. [?]

CNPGO consists of the following key components [?] [?]:

- High Availability: Postgres instance manager
- Backups: Barman
- Connection Pooler: PgBouncer
- Monitoring: Prometheus, Grafana, and Alertmanager

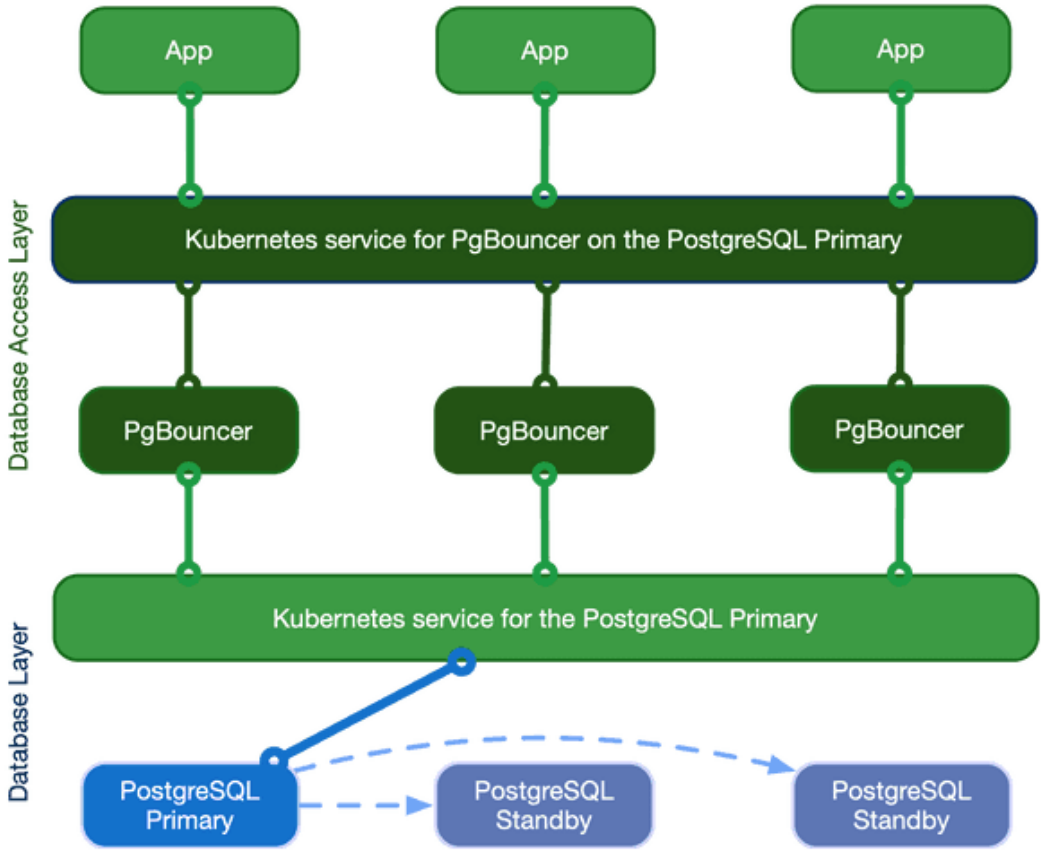

Figure 3.2 CNPGO's architecture [?]

3.4 StackGres Operator

StackGres (SPGO) is a comprehensive distribution of PostgreSQL for Kubernetes, delivered in a user-friendly deployment package. The distribution includes a set of PostgreSQL components that have been carefully selected and optimized to work seamlessly with each other. [**?**]

SPGO is developed by OnGres that was established as a result of years of experience in working with and creating products based on Postgres and supporting clients with their Postgres infrastructures. Postgres databases are at the heart of the company's business, as the name suggests. [**?**]

SPGO's capabilities are the following [**?**]:

- **Postgres Cluster Provisioning:** SPGO is able to create, update or delete Postgres cluster.
- **High Availability:** High availability is achieved by adding additional nodes.
- **Postgres updates:** SPGO is able to apply minor patches. Major updates are possible by SGDbOps [**?**].
- **Backups:** SPGO backup capabilities features: automatic backup schedules, backup to multiple locations, backup to cloud providers (AWS S3, Google Cloud Storage, Azure Blob)
- **Disaster Recovery:** SPGO is capable of Point In Time recovery. Database recovery was not mentioned in the documentation.
- **Cloning:** SPGO is able to create cluster replicas.
- **Monitoring:** Monitoring is provided by Prometheus, Grafana, and Alertmanager.
- **Connection Pooling:** Is provided by native Postgres pooler PgBouncer.
- **Customization:** SPGO provides a wide area of Postgres customization such as WAL configuration, archive mode, vacuum, etc. [**?**]
- **Mamagement Console:** SPGO provides a fully featured management web console.

The current stable version of SPGO is 1.4.2 was released on 24th January 2022. [?] SPGO is distributed under the AGPL3 open-source license. [?] SPGO capability level hasn't been mentioned in the documentation or its web pages.

SPGO consists of the following key components [?]:

- High Availability: Patroni
- Backups: WAL-G
- Connection Pooler: PgBouncer
- Monitoring: Prometheus, Grafana, and Alertmanager.

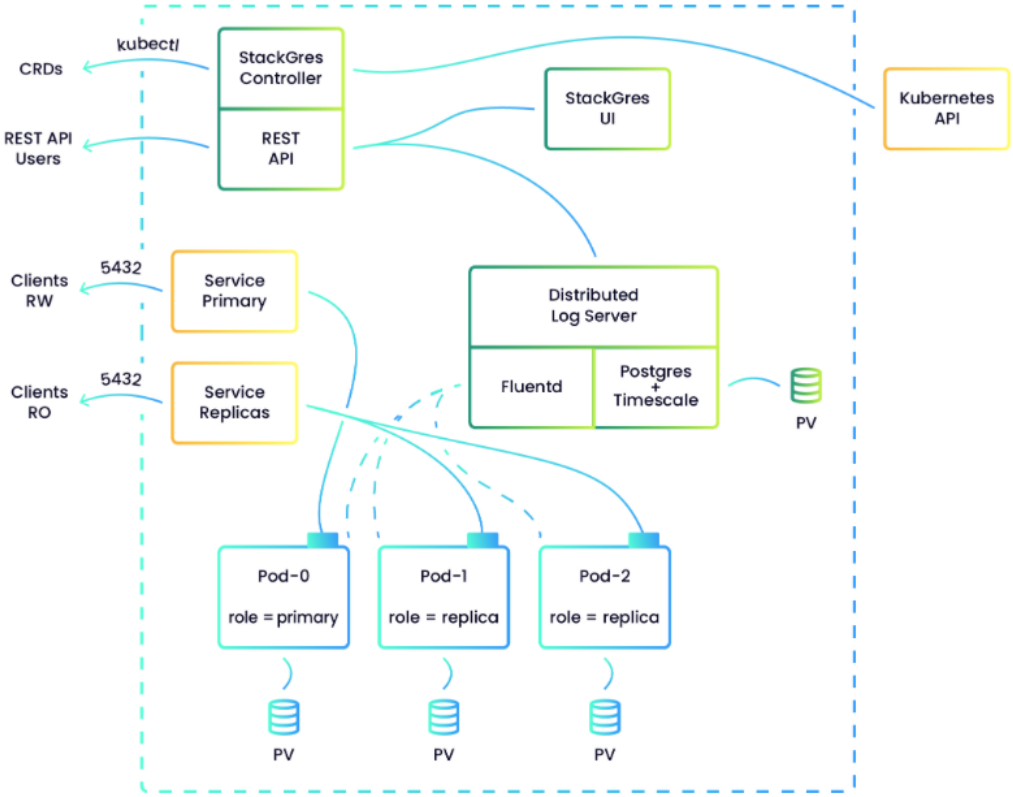

Figure 3.3 SPGO's architecture [?]

3.5 Percona Operator for PostgreSQL

Percona is a company that provides services and solutions for open-source database technologies. It offers expertise, support, and software for MySQL, MongoDB, and PostgreSQL. The company's offerings help organizations manage their open-source databases and ensure optimal performance, security, and scalability. [**?**]

The current stable version of PPO is 1.3.0 and it is compatible with a minimum Kubernetes version of 1.11.0. The upcoming version 2.0.0 is not yet production ready but will be included in the evaluation. PPO is distributed under the Apache License 2.0, an open-source license that allows for both commercial and non-commercial use. With regards to capability, PPO is considered to have the second highest capability level, labeled as Deep Insights. [**?**]

PPO's main features are the same as PGO's [**?**]

Table 3.1 Comparison of selected Operators

	PGO 5.3.0	EDBO 1.18.1	PPO 1.3.0	PPO 2.0.0
Release date	1.22.0	unknown	1.21.0	1.22.0
Minimal K8s version	1.22.0	unknown	1.21.0	1.22.0
Postgres versions supported	15	EDBO 1.18.1	PPO 1.12.0	
Licence	PGO 5.3.0	EDBO 1.18.1	PPO 1.12.0	
Capability	Autopilot	Autopilot	Deep Insights	Deep Insights
Security	PGO 5.3.0	EDBO 1.18.1	PPO 1.12.0	

4 ARCHITECTURE

5 NADPISY A PODNADPISY

Na této stránce je k vidění způsob tvorby různých úrovní nadpisů.

5.1 Podnadpis A

Text

5.2 Podnadpis B

Text

5.3 Podnadpis C

Text

5.3.1 Podpodnadpis alfa

Text

5.3.2 Podpodnadpis beta

Text

5.3.3 Podpodnadpis gama

Text

5.4 Podnadpis D

Text

6 VKLÁDÁNÍ OBRÁZKŮ, TABULEK A CITACÍ

Níže následují ukázky vložení obrázku, tabulky a různorodých citací.

6.1 Obrázek

Obrázek **??** prezentuje logo Fakulty aplikované informatiky.

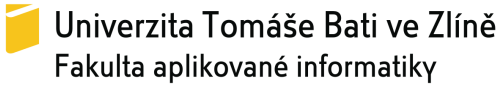

Figure 6.1 Popisek obrázku

6.2 Tabulka

Tabulka **??** obsahuje dva řádky a celkem 7 sloupců.

Table 6.1 Popisek tabulky

	1	2	3	4	5	Cena [Kč]
F	(jedna)	(dva)	(tři)	(čtyři)	(pět)	300

II. PRAKTICKÁ ČÁST

7 NADPIS PRVNÍ KAPITOLY PRAKTICKÉ ČÁSTI

Text je text

ZÁVĚR

Text závěru.

LIST OF ABBREVIATIONS

ORDBMS	Object-relational Database Management System
ACID	Atomicity, Consistency, Isolation, Durability
WAL	Write Ahead Log
K8s	Kubernetes
PGO	Crunchy Postgres for Kubernetes
EDBO	EDB Postgres for Kubernetes Operator
CNPGO	CloudNativePG
SPGO	StackGres Operator
PPO	Percona Operator for PostgreSQL
CPU	Central Processing Unit
PTFE	Polytetrafluoroethylene
VNA	Vector Network Analyser

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

APPENDIX A I. NÁZEV PŘÍLOHY

Obsah přílohy