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

It is essential for the database server to be as close as possible to the applications that are using it. This reduces the number of men in the middle between the database and the application, which reduces database access latency and thus reduces overall application latency and increases security. The mass migration of applications to Kubernetes clusters implies a necessary shift of Postgres to Kubernetes. This thesis defines Postgres, Kubernetes, and their Operators. It then further describes the lifecycle of a Postgres cluster, and searches for Operators capable of managing this lifecycle. It establishes metrics by which it tests and evaluates these Operators. The result of this thesis is the recommendation of a suitable Operator based on the defined metrics.

TBD - remove

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 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.

TBD - remove end

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. [1] [2] 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. [3]

POSTGRES was using its POSTQUEL query language from version, until Andrew Yu and Jolly Chen introduced SQL to POSTGRES in 1995. The name has changed to Postgres95. Postgres95 was completely ANSI C code reduced by 25 % and was 30 – 50 % faster than Postgres 4.2. [3]

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 [3]: "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 [4]. According to Riggs and Ciolli [2]: "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 [5] and supports many kinds of data models such as relational, document, and key/value. [2]

1.1.1 Write Ahead Log

Write-ahead Logging (WAL) used by Postgres is a standard technique to ensure data integrity. Its main concept is that changes in data files (where tables and indexes are stored) must only be written after they are logged (saved to a log file). That means the database is updated after the changes are written to disk. In the event of a system crash, all transactions will be recovered from the disk. [6]

Although WAL is primarily designed for recovery after a database server crash, its design also allows any changes to the database server state to be replayed backward. A copy of the log is also a form of backup. Thus, for recovery to a point in time, only logs that have been saved to that point in time can be restored. This technique is called Point-In-Time Recovery (PITR). [7] These log files can also be streamed to other nodes to serve as a replica or remote backup. [8]

1.1.2 Backup and restore

A full set of backup commands is included in Postgres. Among the simple backup commands are pg_dump and pg_dumpall, which enable one or more databases to be saved in SQL format. A wide range of configuration options are available for these commands, including compression for large databases or exporting only the database schema. To restore a database from a file at a later time, the psql command can be used, which is capable of restoring a database from its dump. [9] These commands are also helpful with migration from one major Postges version to another because the dumped files are plain SQL commands.

The backup options in Postgres are quite limited. Postgres allows to set up of a backup command that runs after the next log file is created, database dumps, and log streaming. For more advanced backup techniques, additional software such as PgBackRest must be used. [7]

PgBackRest PgBackRest is a reliable and simple backup and restore solution that provides many features on top of classic Postgres backup and restore tools like parallel backup options with compression, local or remote backups, cloud backup (S3. Azure and Google Cloud), or backup encryption. Full, incremental, or differential backup is also supported. [10]

TBD: why is it here? Conect to crunhy and operatores

1.1.3 High Availability

The basic structure of a database cluster consists of one or more database servers, which can be called nodes. In Postgres there are two types of nodes, Primary node and Standby node. A Primary node is such a node that allows reading and writing information. The newly written information is then streamed to the Standby nodes. Standby nodes are read-only, they do not allow writing. [8]

Achieving high availability with Postgres is possible by using more than one node in the cluster. Two options are possible here. A single Primary node option, where the Primary node is read and write enabled, and the other nodes are Standby nodes. If the Primary node is unavailable, then the Standby node is promoted to the Primary node. This event is called failover. In this variant, the Primary node streams the logs to the Standby nodes. The second option is to use multiple Primary nodes. However, conflicts can occur because all Primary nodes allow concurrent writes. [11]

Patroni Since Postgres does not provide any software that can detect that a node is unavailable, it is necessary to use software outside of Postgres [12], such as Patroni. Patroni is a popular open-source tool created by Zalando to achieve high availability of Postgres clusters. Patroni uses a distributed configuration source such as ZooKeeper, Etcd, Consul, or Kubernetes for its operation. Patroni can automatically adjust the settings of all managed nodes, therefore it can automate failover and make it seamless. [13] [14]

1.1.4 Load Balancing and Connection Pooling

Using more than one node allows to direct traffic to a node that is less busy and thus achieve load balancing. Postgres doesn't come with any software that allows splitting the load on different nodes, so it is necessary to use an external load balancer such as HA Proxy or pgBouncer. The load balancer then acts as an intermediary between the database and the client and directs the traffic to the available nodes according to the set rules. These load balancers also enable connection pooling which is a technique for managing and reusing database connections to increase performance and reduce overhead. Connection pooling involves creating a pool of pre-created connections that can be shared and reused by multiple client requests, instead of creating a new connection

for each request. This removes the overhead of creating a new process each time a client connects to Postgres and allows the client to use resources that would otherwise be used to service multiple requests (or complete them faster). [15]

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. [16]

As described by Kubernetes Documentation [17] 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. [17]

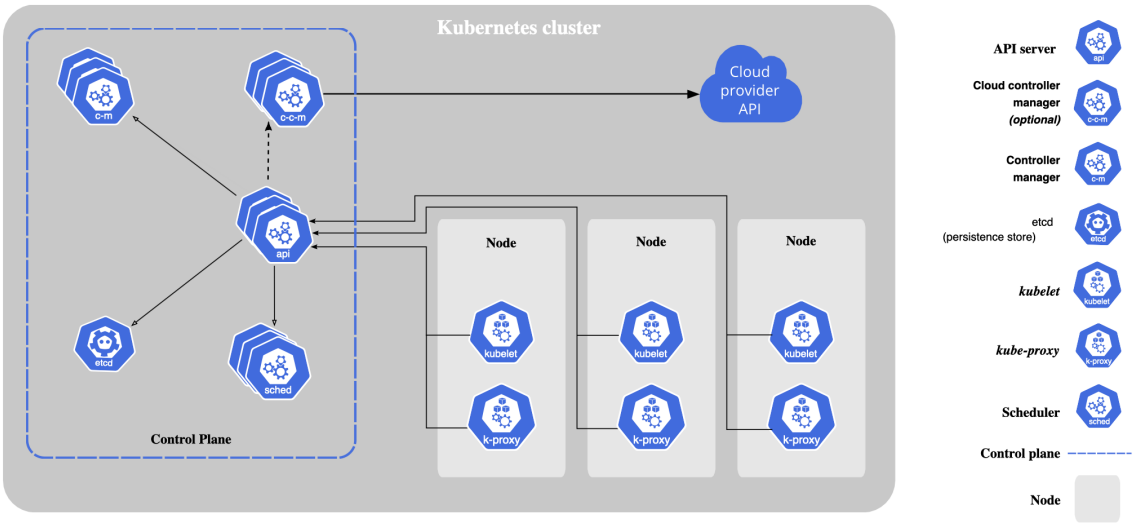

Figure 1.1 The components of a Kubernetes cluster [17]

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 [18]:

- **kube-APIserver:** Acts as the front-end for the Kubernetes API and exposes the API to other components. [17]
- **Etcd:** Highly available distributed key-value store that serves as the backing store for the cluster's configuration data. [19]
- **kube-scheduler:** Assigns work to nodes in the cluster, such as scheduling pods to run on nodes. [20]

- **kube-controller-manager:** Monitors the cluster's state and makes adjustments as necessary to maintain the desired state. [18]
- **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. [17]

Node components: Node components in a Kubernetes cluster run on each node and provide crucial functionality for the operation of containers on that node. [17]

- **kubelet:** Is responsible for communicating with the control plane and ensuring that containers are running and healthy. [21]
- **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. [21]
- **container runtime:** Is responsible for running containers. [17]

1.2.2 Kubernetes Concepts

RepliceSet extension - Operators p. 28 (Replica is general and appllication agnostic)

Pod is the smallest deployable unit that can be created in Kubernetes. [22] A Pod in Kubernetes is comprised of multiple containers and storage volumes that are run together within the same execution environment. As a result, all containers included in a single Pod will always run on the same machine. [20] 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. [23]

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. [24]

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. [25] Kubernetes includes built-in service discovery mechanisms. When a service is created in Kubernetes, it is automatically assigned an IP address and DNS name. Clients and other services can use this name or address to access the service within the Kubernetes cluster. [25]

Containers and pods in Kubernetes are ephemeral. When a container is terminated, any data it has written to its own filesystem is lost. In Kubernetes, storage is represented by a basic abstraction called "volumes". Containers use these volumes by binding them to their respective pods, and can then access the storage regardless of its physical location as if it were a part of their local filesystem. [26]

Kubernetes version 1.5 came with a new object called StatefulSet that allows a set of stateful pods to be deployed and managed. Each pod has a unique, stable network identity and a persistent storage volume. This enables stateful applications like databases to be run on Kubernetes. Advantages of using StatefulSets include predictable naming schemes, ordered pod creation and deletion, and unique persistent storage. [27] [28]

In version 1.7, Kubernetes introduced the Custom Resources extension to its API. [29] This extension allows Kubernetes to use user-defined resources that are not native to Kubernetes as if they were native. [30] Custom resources (CR) is an extension to the Kubernetes API that extends the deployment with additional parameters that are not part of it. CR stores these parameters and allows the API server to access them just like the native Kubernetes parts. CR is created in the Kubernetes cluster using a definition called Custom Resource Definition (CRD). [31]

Kuberentes controllers are control loops[1)] that constantly check the state of their controlled objects. If the controlled objects are not in the desired state, the controller performs actions to get the controlled objects into that state. For example, restart a crashed node, add a new replica, modify settings, etc. [32] However, to work with CR, custom controllers that can work with these resources must be created, these controllers are called Custom Controllers. [33]

[1)] A control loop is a process that continuously monitors the state of a system, compares it to a desired state, and makes adjustments to bring the system closer to the desired state.

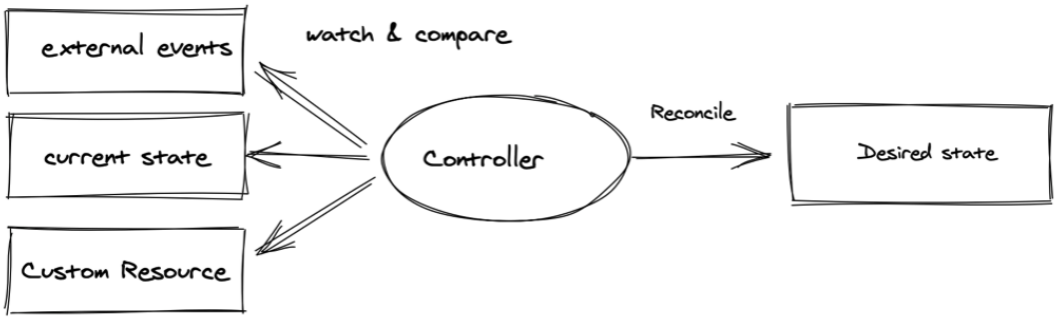

Figure 1.2 Kubernetes controller [34]

TBD - show that Kubernetes can run stateless very well, maybe from Operator book

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 Running Postgres in Kubernetes

Kubernetes cannot know all complex stateful applications, which can contain a large number of nodes and have a wide range of uses while remaining general-purpose. The goal of Kubernetes is to provide an abstraction covering basic application concepts and providing options for extensions for more complex applications and their specific operations. Kubernetes cannot and should not know all the possible settings and operations that, for example, a Postgres cluster needs to run. [35]

The easiest way to run Postgres in Kubernetes is through the StatefulSet just mentioned. This StatefulSet can start a Postgres pod, create a persistent volume, and connect this volume to the pod. A stateful set can do this for all replicas set in its configuration. It can also scale up or down. Unfortunately, however, all independent Postgres instances created by StatefulSet controller are not synchronized in any manner.

This basic setup may be sufficient for running a single node, but it is no longer sufficient fro managing the whole Postgres lifecycle. For managing whole Postgres lifecycle it is necessary to install other applications in the Kubernetes cluster and then configure the entire Postgres cluster to work with them. This represents a large amount of work and subsequent maintenance that Kubernetes Operators can facilitate.

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.

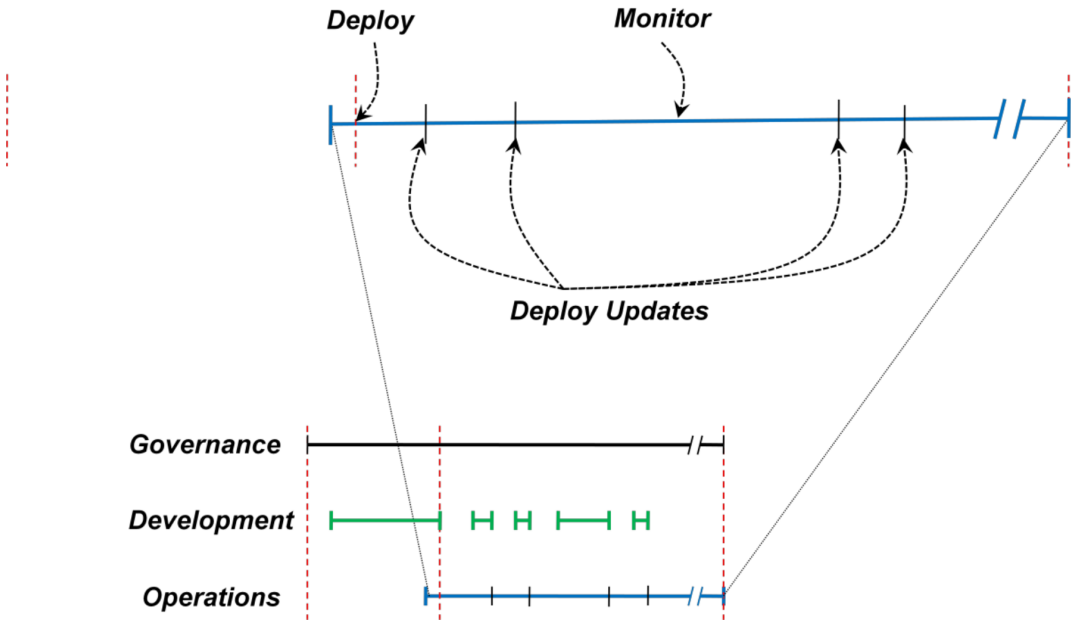

Figure 2.1 Application Life Cycle [36]

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. [36]

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

2.1 Operators

TBD - define Operator hub

As described in the previous chapters, Kubernetes can run stateless applications very well. But its general purpose makes running complex stateful applications on top of it quite challenging.

This has changed in 2016 when CoreOS came up with Operators as a way to deploy complex applications with state such as databases, caches, or monitoring systems. [37]

An operator is a special kind of software that extends the Kubernetes API and has a particular knowledge of managed resource that Kubernetes does not have. The Operator also serves as a packaging mechanism for distributing applications including their dependencies in Kubernetes. The Operator can manage, restore, update or monitor the resource. It can also manage very complex applications. The Kubernetes Operator thus replaces the human operator after which it is named, who would otherwise take care of these tasks. [38] [37]

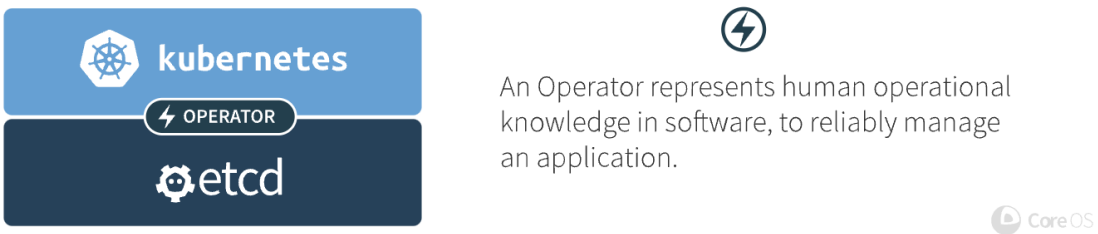

Figure 2.2 Definition of Kubernetes Operator [37]

CoreOS demonstrated the use of its Operator on Etcd (described in the Kubernetes Components chapter). When new Etcd nodes are created, it is necessary to give them a DNS names and use the Etcd cluster management tools to add the new nodes to an existing cluster. CoreOS has automated these tasks with the Etcd Operator so that all that is required is to increase the number of replicas in the Operator CRD and the Etcd Operator will perform these tasks instead of a human operator. [37] By embedding the human operator's operational knowledge into the code, this ensures that these tasks are repeatable, testable and upgradable. It also ensures that the necessary operations are always performed, executed in the order in which they are supposed to be performed, and none are skipped. This reduces the number of hours spent on dull but essential work such as backups. [34]

As described by Operator White Paper [34] and depicted in figure 2.3, Operator consists of the following parts

- The managed application or infrastructure
- Software that has some specific knowledge of the managed application or infrastructure and allows the user to declaratively set the desired state
- Custom Controller, which is responsible for achieving the desired state

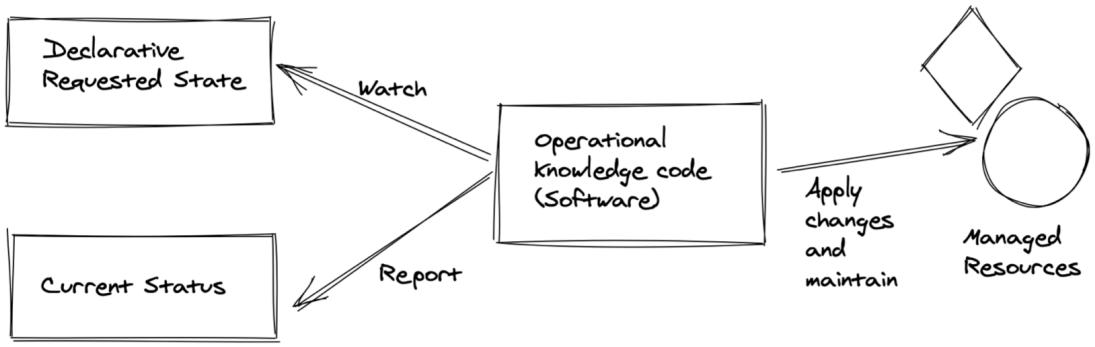

Figure 2.3 Operator pattern [34]

Like human operators, Kubernetes Operators can have a level of manual skill ranging from basic software installation and setup skills to a high level where they can scale software vertically or horizontally to automatically change the configuration or detect abnormalities. All Operator maturity levels are depicted in the figure 2.4. The highest level can only be reached by programming the Operator in the GO programming language or by using the Ansible automation tool. [39]

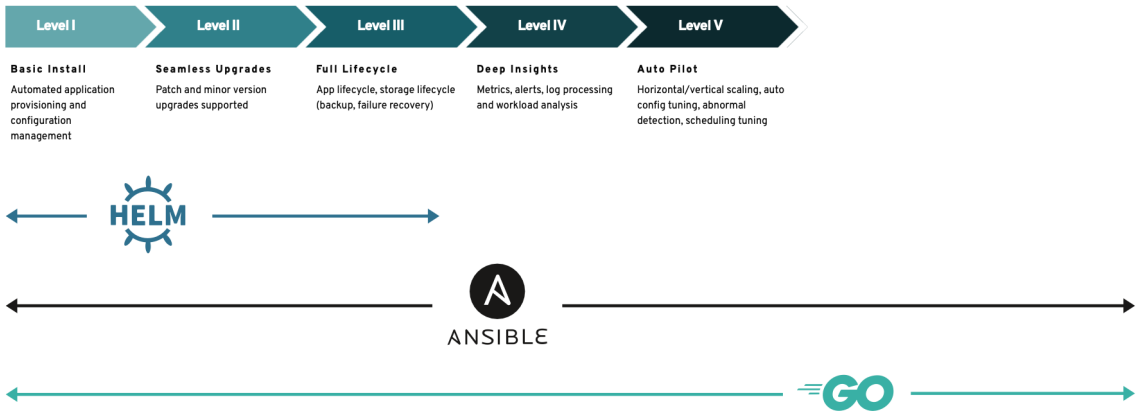

Figure 2.4 Operator maturity levels described by Operator Framework [40]

The Kubernetes cluster is divided into individual namespaces that separate the objects and names in the cluster and can have constraints applied to them. This partitioning

makes it easier to share the cluster between users or entire teams. The object name must be unique within a namespace, but not between namespaces. An operator usually operates in its own namespace so it has a Namespace Scope, but it can also operate in the whole cluster in which case it will be a Cluster Scope Operator. Namespace Scope Operators are more flexible and easier to upgrade due to their independence from the rest of the cluster. Operator rights are further restricted by the so-called Role-Based Acceess Control (RBAC), which grants the rights assigned to the Operator. [31]

The following options are advised by the Operator white paper [39] in case the Operator is to be used for controlling the resource:

- Consultation with the creator of the resource to be controlled about the possibilities of using the Operator.
- The search for public Operator registries that provide a platform for publishing Operators and the underlying documentation.
- The creation of own Operator.

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 recommended in the previous section. The choice of Operator should first be consulted with the manufacturer of the controlled source. Postgres offers the following Kubernetes Operators in its software catalog [41]: CloudNativePG, EDB Postgres for Kubernetes a Kubegres.

The next step involved a search of the Operators' registers. In particular the Operator Hub. [42] Operator Hub presents nine 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, PostgreSQL Operator by Dev4Ddevs.com and StackGres by OnGres.

A deeper internet search revealed Stolon Operator. [43]

Of the thirteen 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. [44]:

PGO's capabilities are the following:

- **Postgres Cluster Provisioning:** PGO is able to create [45], update [46] or delete Postgres cluster [47]
- **High Availability:** High availability is achieved by adding additional nodes. PGO uses a synchronous replication technique. [48]
- **Postgres updates:** PGO is able to apply minor patches [49], and major upgrades since version 5.1. [50]
- **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. [51]
- **Disaster Recovery:** PGO is capable of Point In Time recovery, in place Point in Time Recovery, restore of an individual database. [52]
- **Cloning:** PGO is able to clone cluster. [52]
- **Monitoring:** Monitoring is provided by Prometheus, Grafana, and Alertmanager. [53]
- **Connection Pooling:** PgBouncer connection pooler from Postgres is part of PGO. [54]

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. [55] 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. [56]

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. [57]

PGO consists of the following key components [58]:

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

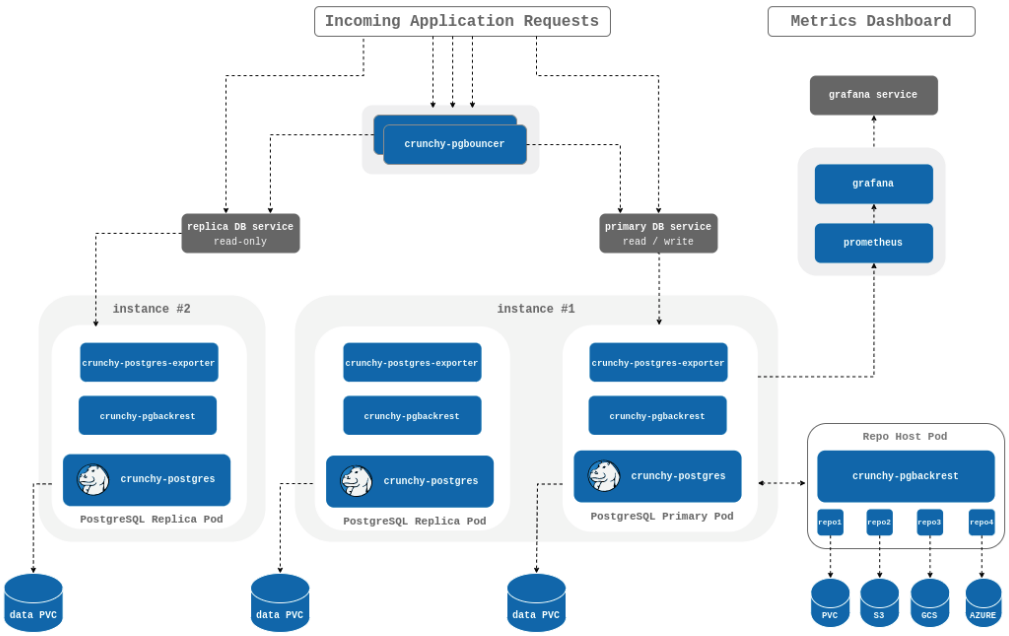

Figure 3.1 PGO's architecture [59]

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. [60]

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. [61]

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. [62]

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. [63] 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. [64]

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. [64]

CNPGO's capabilities are the following:

- **Postgres Cluster Provisioning:** CNPGO is able to create, update or delete Postgres cluster. [65]
- **High Availability:** High availability is achieved by adding additional nodes. PGO uses a synchronous replication technique. [66]
- **Direct database imports:** CNPGO provides direct database import from remote Postgres server by using pg_dump and pg_restore even on different Postgres versions. [67]
- **Postgres updates:** CNPGO is able to apply minor patches. [68] 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 [69][70]. Due to EDB's backup software Barman backup compression is available also. [69]
- **Disaster Recovery:** CNPGO is capable of Point In Time recovery. Database recovery was not mentioned in the documentation. [69]
- **Cloning:** CNPGO is able to create cluster replicas. [66]
- **Monitoring:** Monitoring can be provided by the additional installation of Prometheus, and Grafana, and Alertmanager. [71]
- **Connection Pooling:** Is provided by native Postgres pooler PgBouncer. [72]

- **Customization:** CNPGO provides a wide area of Postgres customization such as max parallel workers tuning or WAL configuration [73]

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. [64]

CNPGO consists of the following key components [74] [71]:

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

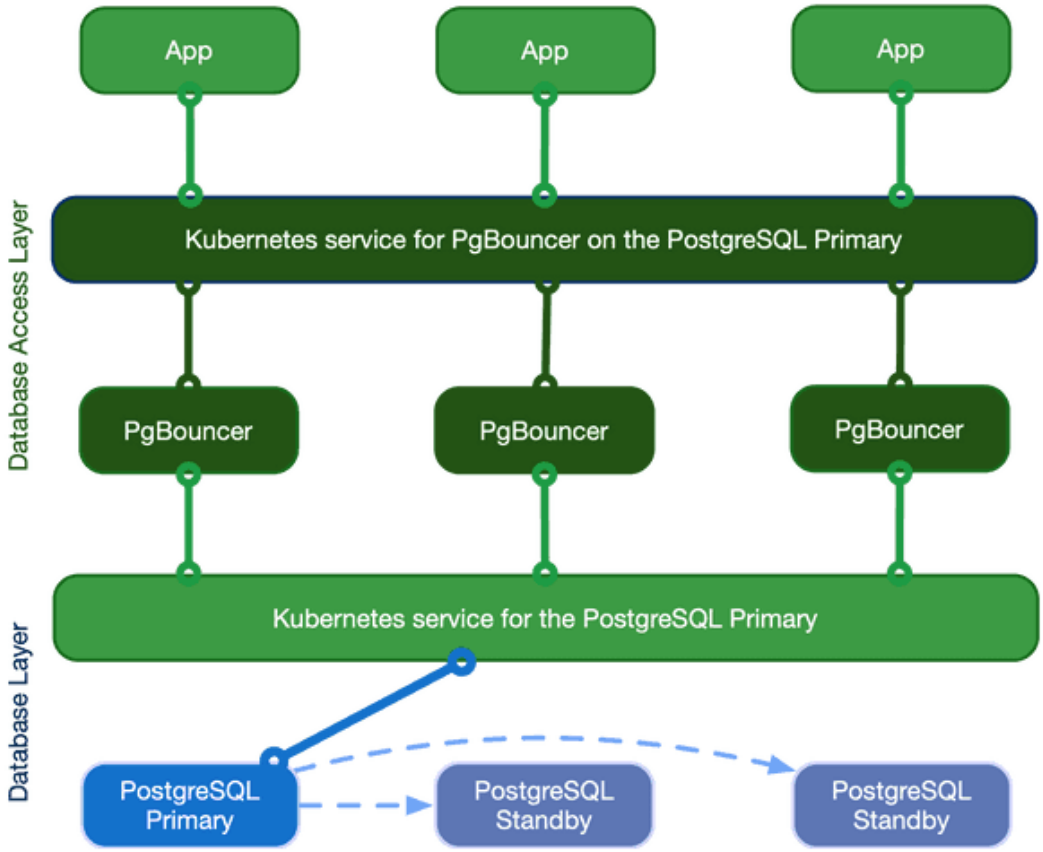

Figure 3.2 CNPGO's architecture [72]

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. [75]

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. [76]

SPGO's capabilities are the following [76]:

- **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 [77].
- **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. [78]
- **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. [79] SPGO is distributed under the AGPL3 open-source license. [80] SPGO capability level hasn't been mentioned in the documentation or its web pages.

SPGO consists of the following key components [74]:

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

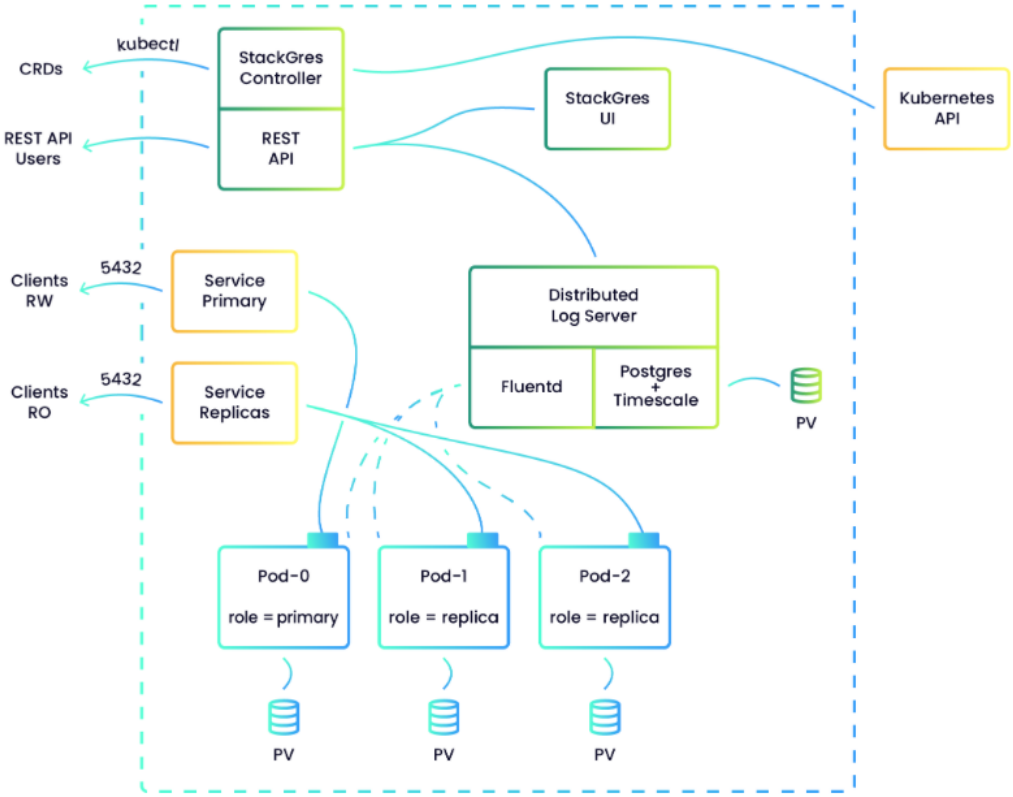

Figure 3.3 SPGO's architecture [81]

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. [82]

Percona Operator for PostgreSQL (PPO) is based on Crunchy Postgres for Kubernetes. Percona forked PGO v 4.7 and has added enhancements for monitoring, upgradability, and flexibility. [83]

Differences between PGO and PPO are the following:

- **Postgres updates:** PPO provides automatic Postgres updates for minor and major versions of Postgres. [84]
- **Backups:** PPO is not able to back up to Azure. [85]
- **Disaster Recovery:** PPO documentation does not mention the possibility of restoring a single database from a backup. [86]
- **Monitoring:** PPO is not using the usual monitoring stack consisting of Prometheus and Grafana but their own Percona Monitoring and Management. [87]

The current stable version of PPO is 1.3.0 was released on 4th August 2022. The upcoming version 2.0.0 is not yet production ready therefore will not be subject to testing and evaluation. [88] 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. [89]

PPO consists of the following key components [74]:

- High Availability: Patroni
- Backups: PgBackRest
- Connection Pooler: PgBouncer
- Monitoring: Percona Monitoring and Management

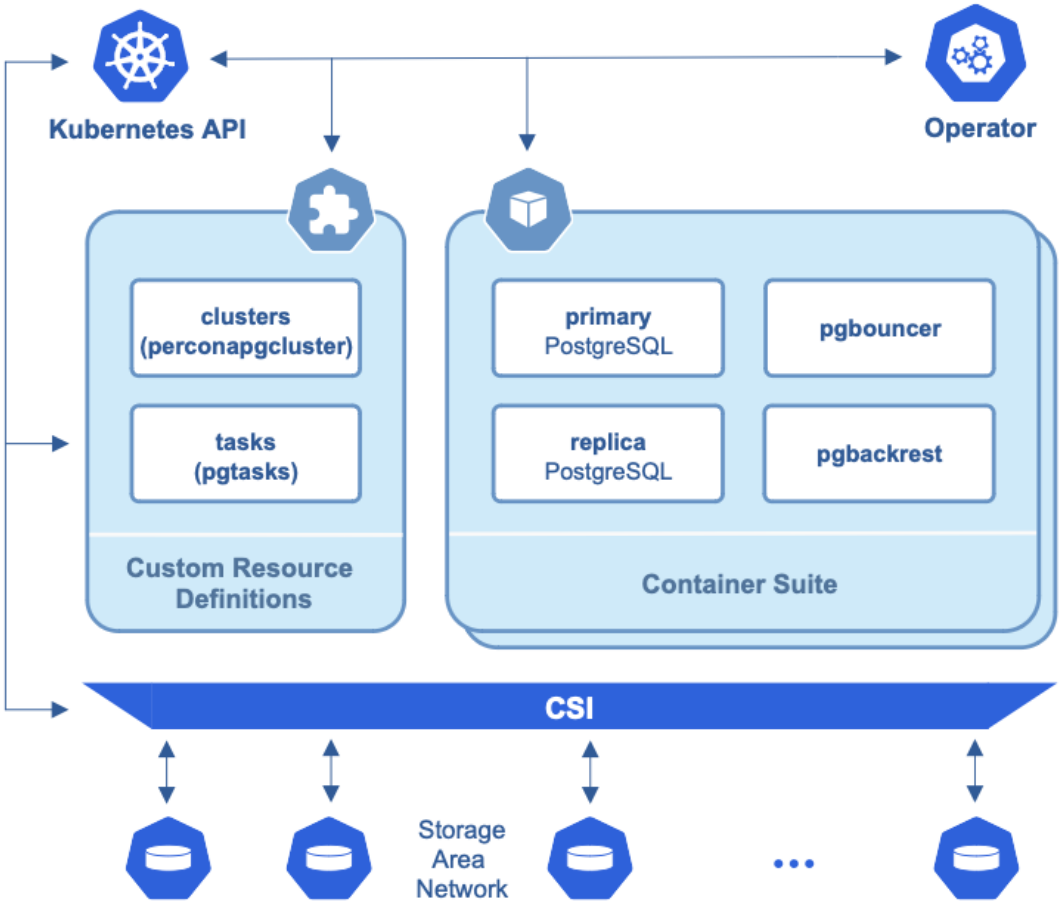

Figure 3.4 PPO's architecture [59]

Table 3.1 Comparison of selected Operators

	PGO 5.3.0	CNPGO 1.18.1	SPGO 1.4.2	PPO 1.3.0
Programing language	GO	GO	JAVA	GO
Supported Postgres versions	12, 13, 14, 15	11, 12, 13, 14, 15	12, 13, 14, 15	12, 13, 14
Major version update	Manual	Manual	Manual	Automatic
Licence	Apache 2.0	Apache 2.0	Apache 2.0	AGPL 3
User interface	No	No	Yes	No

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 6.1 prezentuje logo Fakulty aplikované informatiky.

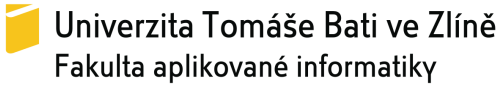

Figure 6.1 Popisek obrázku

6.2 Tabulka

Tabulka 6.1 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.

REFERENCES

[1] Group, T. P. G. D.: PostgreSQL 15.1 Documentation - What Is PostgreSQL? [visited 2023-02-08].
URL `https://www.postgresql.org/docs/15/intro-whatis.html`

[2] Riggs, S.; Ciolli, G.: *PostgreSQL 14 Administration Cookbook*, chapter Introducing PostgreSQL 14. Birmingham: Packt publishing, 1st edition, 2022, ISBN 9781803248974, pp. 2–8.

[3] Group, T. P. G. D.: PostgreSQL 15.1 Documentation - History. [visited 2023-02-08].
URL `https://www.postgresql.org/docs/15/history.html`

[4] Inc., S. E.: Stack Overflow 2022 Developer Survey. [visited 2023-02-07].
URL `https://survey.stackoverflow.co/2022/`

[5] Juba, S.; Vannahme, A.; Volkov, A.: *Learning PostgreSQL*, chapter Relational databases. Packt Publishing Ltd, 1st edition, 2015, ISBN 9781783989188, pp. 1–4.

[6] Group, T. P. G. D.: PostgreSQL: Documentation: 15: 30.3. Write-Ahead Logging (WAL). [visited 2023-02-21].
URL `https://www.postgresql.org/docs/current/wal-intro.html`

[7] Group, T. P. G. D.: PostgreSQL: Documentation: 15: 26.3. Continuous Archiving and Point-in-Time Recovery (PITR). [visited 2023-02-23].
URL `https://www.postgresql.org/docs/15/continuous-archiving.html`

[8] Riggs, S.; Ciolli, G.: *PostgreSQL 14 Administration Cookbook*, chapter Replication and Upgrades. Birmingham: Packt publishing, 1st edition, 2022, ISBN 9781803248974, pp. 499–557.

[9] Group, T. P. G. D.: PostgreSQL: Documentation: 15: 26.1. SQL Dump. [visited 2023-02-23].
URL `https://www.postgresql.org/docs/15/backup-dump.html`

[10] Group, T. P. G. D.: Reliable PostgreSQL Backup and Restore. [visited 2023-02-23].
URL `https://pgbackrest.org`

[11] Group, T. P. G. D.: PostgreSQL: Documentation: 15: 27. High Availability, Load Balancing, and Replication. [visited 2023-02-23].
URL `https://www.postgresql.org/docs/15/high-availability.html`

[12] Group, T. P. G. D.: PostgreSQL: Documentation: 15: 27.3. Failover. [visited 2023-02-23].
URL https://www.postgresql.org/docs/15/warm-standby-failover.html

[13] Stratnev, P.: Migrating a PostgreSQL database cluster managed by Patroni. [visited 2023-02-24].
URL https://blog.palark.com/migrating-a-postgresql-cluster-managed-by-patron

[14] SE, Z.: Introduction — Patroni 3.0.1 documentation. [visited 2023-02-24].
URL https://patroni.readthedocs.io/en/latest/

[15] Avinash Vallarapu, F. L. C., Jobin Augustine: Scaling PostgreSQL using Connection Poolers and Load Balancers for an Enterprise Grade environment. [visited 2023-02-24].
URL https://www.percona.com/blog/scaling-postgresql-using-connection-poolers

[16] Vayghan, L. A.; Saied, M. A.; Toeroe, M.; et al.: Kubernetes as an availability manager for microservice applications. *arXiv preprint arXiv:1901.04946*, 2019.

[17] Authors, T. K.: Kubernetes Components. [visited 2023-02-07].
URL https://kubernetes.io/docs/concepts/overview/components/

[18] Sayfan, G.: *Mastering Kubernetes*, chapter Kubernetes concepts. Birmingham: Packt Publishing, third edition edition, [2020]., ISBN 978-183-9211-256.

[19] Dobies, J.; Wood, J.: *Kubernetes operators*. Sebastopol, CA: O'Reilly Media, 2020, ISBN 978-149-2048-046.

[20] Burns, B.; Beda, J.; Hightower, K.; et al.: *Kubernetes: up and running*, chapter Pods. " O'Reilly Media, Inc.", third edition edition, 2022.

[21] Burns, B.; Beda, J.; Hightower, K.; et al.: *Kubernetes: up and running*. " O'Reilly Media, Inc.", third edition edition, 2022.

[22] Authors, T. K.: Pods. [visited 2023-02-07].
URL https://kubernetes.io/docs/concepts/workloads/pods/

[23] Authors, T. K.: Create static Pods. [visited 2023-02-07].
URL https://kubernetes.io/docs/tasks/configure-pod-container/static-pod/

[24] Authors, T. K.: ReplicaSet. [visited 2023-02-07].
URL https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

[25] Authors, T. K.: Service. [visited 2023-02-07].
URL https://kubernetes.io/docs/concepts/services-networking/service/

[26] Sayfan, G.: *Mastering Kubernetes*, chapter Managing Storage. Birmingham: Packt Publishing, third edition edition, [2020]., ISBN 978-183-9211-256, pp. 175–219.

[27] Authors, T. K.: StatefulSets. [visited 2023-02-24].
URL https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

[28] Foundaution, C. N.: kubernetes/CHANGELOG-1.5.md at master · a0x8o/kubernetes. [visited 2023-02-24].
URL https://github.com/a0x8o/kubernetes/blob/master/CHANGELOG-1.5.md

[29] Foundaution, C. N.: Custom Resource (CR). [visited 2023-03-02].
URL https://ibm.github.io/kubernetes-operators/lab1/

[30] Ziolkowski, D.: Kubernetes CRDs: What They Are and Why They Are Useful. [visited 2023-03-01].
URL https://thenewstack.io/kubernetes-crds-what-they-are-and-why-they-are-us

[31] Dobies, J.; Wood, J.: *Kubernetes operators*. Sebastopol, CA: O'Reilly Media, 2020, ISBN 978-149-2048-046, pp. 27–32.

[32] Authors, T. K.: Controllers. [visited 2023-03-02].
URL https://kubernetes.io/docs/concepts/architecture/controller/

[33] Authors, T. K.: Custom Resources. [visited 2023-03-02].
URL https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/

[34] Kahani, O.; Strejevitch, J.; Schuetz, T.; et al.: CNCF Operator White Paper - Final Version. 2023.
URL https://github.com/cncf/tag-app-delivery/blob/main/operator-wg/whitepaper/Operator-WhitePaper_v1-0.md

[35] Dobies, J.; Wood, J.: *Kubernetes operators*. Sebastopol, CA: O'Reilly Media, 2020, ISBN 978-149-2048-046, pp. 4–8.

[36] David, C.: WHAT IS APPLICATION LIFECYCLE MANAGEMENT? [visited 2023-02-07].

URL http://davidchappell.com/writing/white_papers/What_is_ALM_v2.0--Chappell.pdf

[37] Philips, B.: Introducing Operators: Putting Operational Knowledge into Software. [visited 2023-03-01].
URL https://web.archive.org/web/20170129131616/https://coreos.com/blog/introducing-operators.html

[38] Dobies, J.; Wood, J.: *Kubernetes operators*. Sebastopol, CA: O'Reilly Media, 2020, ISBN 978-149-2048-046, pp. XIII–XVI.

[39] Dobies, J.; Wood, J.: *Kubernetes operators*. Sebastopol, CA: O'Reilly Media, 2020, ISBN 978-149-2048-046, pp. 27–32.

[40] Framework, T. O.: Welcome to Operator framework. [visited 2023-03-03].
URL https://operatorframework.io/operator-capabilities/

[41] Group, T. P. G. D.: PostgreSQL: Software Catalogue - Clustering/replication. [visited 2023-03-10].
URL https://www.postgresql.org/docs/15/warm-standby-failover.html

[42] OperatorHub.io: OperatorHub.io | The registry for Kubernetes Operators. [visited 2023-03-10].
URL https://operatorhub.io/?keyword=postgres

[43] Bogdanov, N.: Comparing Kubernetes operators for PostgreSQL. [visited 2023-02-20].
URL https://blog.palark.com/comparing-kubernetes-operators-for-postgresql/

[44] Crunchy Data Solutions, I.: Trusted Open Source PostgreSQL and Commercial Support for the Enterprise. [visited 2023-02-07].
URL https://www.crunchydata.com

[45] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/create-cluster/

[46] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/customize-cluster/

[47] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/delete-cluster/

[48] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/high-availability/

[49] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/update-cluster/

[50] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-17].
URL https://www.crunchydata.com/blog/easy-major-postgresql-upgrades-using-pg

[51] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/backups/

[52] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/disaster-recovery/

[53] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/monitoring/

[54] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/tutorial/connection-pooling/

[55] Foundation, T. L.: postgresql 5.3.0 · operator-framework/community-operators. [visited 2023-02-15].

URL https://artifacthub.io/packages/olm/community-operators/postgresql

[56] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-14].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0

[57] OperatorHub.io: OperatorHub.io | The registry for Kubernetes Operators. [visited 2023-02-07].
URL https://operatorhub.io/operator/postgresql

[58] Crunchy Data Solutions, I.: PGO, the Postgres Operator from Crunchy Data. [visited 2023-02-15].
URL https://github.com/CrunchyData/postgres-operator

[59] L'Ecuyer, A.: Easy Postgres Major Version Upgrades Using PGO v5.1. [visited 2023-02-15].
URL https://access.crunchydata.com/documentation/postgres-operator/5.3.0/architecture/overview/

[60] OperatorHub.io: OperatorHub.io | The registry for Kubernetes Operators. [visited 2023-02-13].
URL https://operatorhub.io/operator/cloud-native-postgresql

[61] Corporation, E.: About EDB. [visited 2023-02-13].
URL https://www.enterprisedb.com/company

[62] Corporation, E.: EDB Postgres for Kubernetes v1. [visited 2023-02-13].
URL https://www.enterprisedb.com/docs/postgres_for_kubernetes/latest/

[63] Corporation, E.: EDB Postgres for Kubernetes v1. [visited 2023-02-17].
URL https://www.enterprisedb.com/docs/postgres_for_kubernetes/latest/license_keys

[64] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/

[65] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/postgres_for_kubernetes/latest/replica_cluster/

[66] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/replication/

[67] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/database_import/

[68] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/rolling_update/#rolling-updates

[69] Corporation, E.: EDB Postgres for Kubernetes v1. [visited 2023-02-16].
URL https://cloudnative-pg.io/documentation/1.19/postgres_for_kubernetes/latest/backup_recovery/

[70] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/postgres_for_kubernetes/latest/tde/

[71] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/quickstart/

[72] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/postgres_for_kubernetes/latest/connection_pooling/

[73] Contributors, T. C.: CloudNativePG. [visited 2023-02-17].
URL https://cloudnative-pg.io/documentation/1.19/postgres_for_kubernetes/latest/postgresql_conf/

[74] Stölting, S. J.: PostgreSQL On Kubernetes Experiences. [visited 2023-02-20].
URL https://proopensource.it/blog/postgresql-on-k8s-experiences

[75] Inc., O.: OnGres Inc. / StackGres. [visited 2023-02-20].
URL https://gitlab.com/ongresinc/stackgres

[76] Inc., O.: About. [visited 2023-02-20].
URL https://www.ongres.com/about-us/

[77] Inc., O.: SGDbOps. [visited 2023-02-20].
URL https://stackgres.io/doc/latest/reference/crd/sgdbops/#major-version-upgrade

[78] Inc., O.: SGPostgresConfig. [visited 2023-02-20].
URL https://stackgres.io/doc/latest/reference/crd/sgpgconfig/

[79] Inc., O.: CHANGELOG.md · main · OnGres Inc. / StackGres. [visited 2023-02-20].
URL https://gitlab.com/ongresinc/stackgres/-/blob/main/CHANGELOG.md

[80] Inc., O.: Licensing. [visited 2023-02-20].
URL https://stackgres.io/doc/latest/intro/license/

[81] Inc., O.: Architecture. [visited 2023-02-20].
URL https://stackgres.io/doc/latest/intro/architecture/

[82] LLC, P.: About Percona - Percona. [visited 2023-02-14].
URL https://www.percona.com/about

[83] Pronin, S.: Run PostgreSQL in Kubernetes: Solutions, Pros and Cons. [visited 2023-02-20].
URL https://www.percona.com/blog/run-postgresql-in-kubernetes-solutions-pros

[84] LLC, P.: Update Percona Operator for PostgreSQL. [visited 2023-02-21].
URL https://docs.percona.com/percona-operator-for-postgresql/update.html

[85] LLC, P.: Comparison with other solutions. [visited 2023-02-20].
URL https://docs.percona.com/percona-operator-for-postgresql/compare.html

[86] LLC, P.: Providing Backups. [visited 2023-02-21].
URL https://docs.percona.com/percona-operator-for-postgresql/backups.html

[87] LLC, P.: Monitor with Percona Monitoring and Management (PMM). [visited 2023-02-2o].
URL https://docs.percona.com/percona-operator-for-postgresql/monitoring.html

[88] LLC, P.: Percona Operator for PostgreSQL. [visited 2023-02-21].
URL https://docs.percona.com/percona-operator-for-postgresql/2.0/index.html

[89] OperatorHub.io: OperatorHub.io | The registry for Kubernetes Operators. [visited 2023-02-14].
URL https://operatorhub.io/operator/percona-postgresql-operator

LIST OF ABBREVIATIONS

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
CNPGO	CloudNativePG
CR	Custom Resource
CRD	Custom Resource Definition
EDBO	EDB Postgres for Kubernetes Operator
K8s	Kubernetes
ORDBMS	Object-relational Database Management System
PGO	Crunchy Postgres for Kubernetes
PPO	Percona Operator for PostgreSQL
RBAC	Role-Based Acceess Control
SPGO	StackGres Operator
WAL	Write Ahead Log

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

APPENDIX A I. NÁZEV PŘÍLOHY

Obsah přílohy