MANNEM Solutions

1 Introduction

This whitepaper describes PgVillage[1], a solution blueprint designed by Mannem Solutions for easily deploying, running and managing Postgres in your estate. The added value of running this solution consists of:

- Completely Open Source and completely community driven
- Based on the standard PostgreSQL distribution
- Deployment at scale, to run 1 is as easy as to run 1000
- Security by design
- Design is based on 'Keep It Simple and Stupid'
- Loosely coupled and easy to integrate
- It is a blueprint, meaning you can leave out and combine as you see fit
- Forward thinking, new technology

The options to run this blueprint:

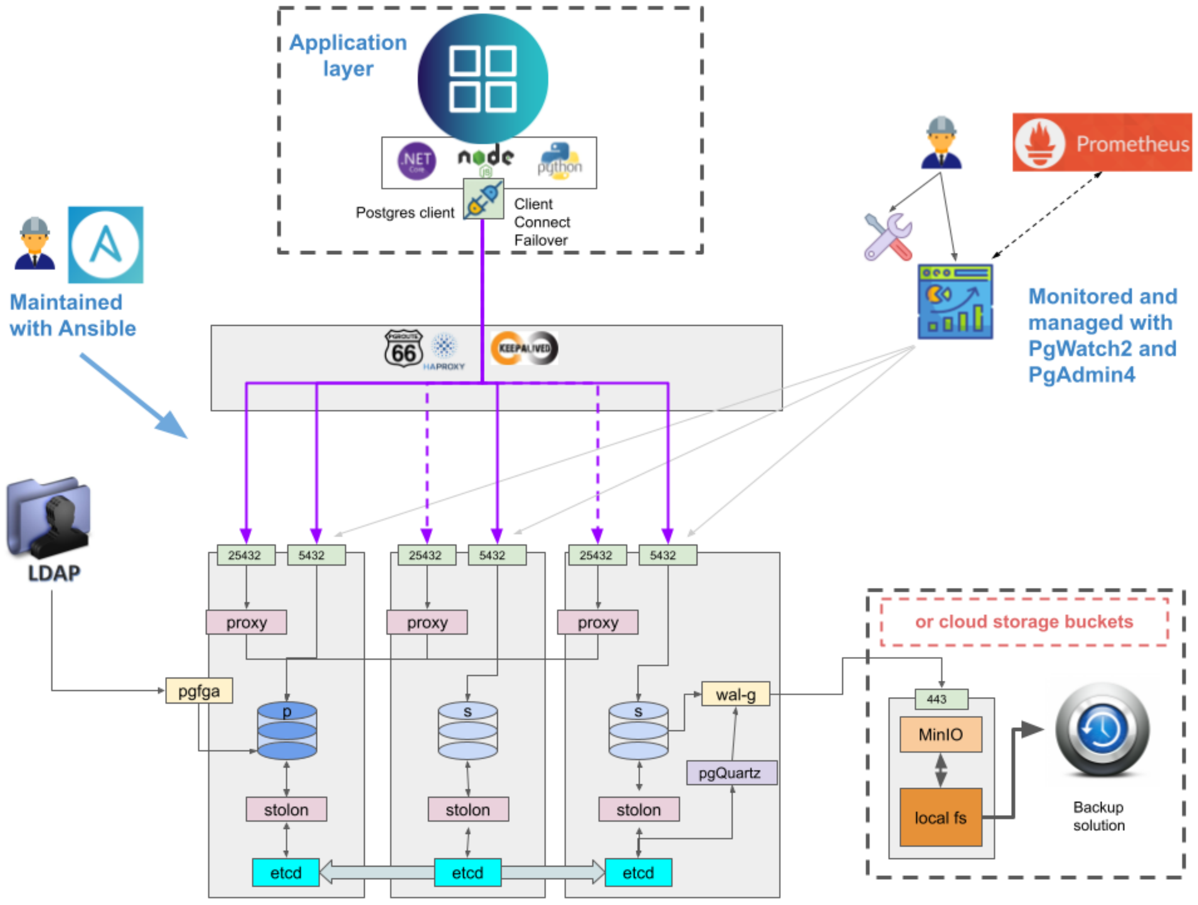
- Download, run and use it, it is for free
- Contribute to enhance the solutions
- Mannem Solutions can provide premiere support on the stack if needed

Important note: Many parts of this solution are readily available, but some still have a 'coming soon' status. For the current state and roadmap, please check with pgvillage on github: https://github.com/MannemSolutions/pgvillage/issues.

[1] https://github.com/MannemSolutions/pgvillage

2 Generic blueprint design

2.1 Architecture design



2.2 Architecture description

The architecture consists of the following parts:

- By default 3 or more Postgres database servers run with Streaming Replication
    - Running a single instance deployment without cluster management is an alternative option
    - Running as a single primary/standby setup by deploying an extra etcd daemon on another server is another option
- Deployment with Ansible (VM)
- Backup:
    - wal-g on the database server, backup from standby
    - Minio with filesystem backend and filesystem backup, or alternatively backup to Cloud Storage buckets

Future:

- Batch and maintenance with pgQuartz, which allows for easy job definition, cluster aware scheduling, and end result monitoring
- For connectivity the blueprint primarily relies on Client Connect Failover[2].
    - Option to run with Keepalived managed VIP For legacy applications without Client Connect Failover
    - Option to run with HAProxy and PgRoute66 for ReadOnly scalability
- Deployment with HELM on Kubernetes[3]
- Authorization and fine grained access
    - Manage authorization and fine grained access with an LDAP using PgFga
- Management and monitoring optional with PgWatch2 and pgAdmin4

[2] With client connect failover, the client will manage connection failovers. The client is aware of all host names and connects to the next host when the connection to a host fails. The client can be configured to only connect to a master, only to a standby, etc.

[3] Operator on the roadmap

3 Background on tools selection

3.1 stolon

TL;DR: Proper HA management without the risk for split brain issues is key with every PostgreSQL HA design, and stolon brings key capabilities that are crucial on this front, while remaining a simple tool to deploy and operate.

There are many different HA tools out there, but there is only one like stolon.
Stolon brings:

- Consensus mechanism based on etcd, which
  - means that creating a split brain scenario is very, very hard.
- Traffic routing is part of the base solution (stolon proxy), which
  - makes a split brain scenario virtually impossible
  - enables faster failover
  - easy to service single vip applications if needed
- Very fast failover with little to no impact on applications
  - tested to be sub-second (very low, and only transactional workload, faster than Patroni)
  - only impact is a reconnect and a resubmit of the transaction if needed
- Meets KISS design
  - GoLang singleton binary. Easy deployment and works everywhere
  - only one config file is all that is required
  - Manages hba requirements for stolon directly, no manual intervention required
  - Meant for one purpose, only for HA management, nothing else

3.2 Wal-g

TL;DR: Backup should be very easy and robust. Decoupling and simple tooling without any unnecessary bells and whistles help with that. And Wal-g meets all of these requirements.

There are many reasons why wal-g is such a great fit:

- Supports backup to buckets
  - optimal decoupling. Backup server runs minio, database servers run wal-g and communication is Cloud Storage protocol
  - Simple for integration. In the cloud, just use buckets. non-cloud, just run Minio and backup the storage behind it
- GoLang sourcecode
  - Fast, optimized for parallel execution
  - Singleton binary
    - Easy deployment
    - No module versioning hazard
  - Lean and mean
    - No unnecessary complexity, bells and whistles
    - Simple integration interface, only small bash script is all that is required
  - New technology
  - A lot of options for encryption to meet any security requirement

3.3 Chainsmith[4]

In simple terms: PgVillega uses TLS for encryption in transit and Client certificate authentication. But requesting or generating the correct certificates is a very complex task. ChainSmith is a templating tool to be used to request the correct certificates. optionally, ChainSMith can also be used to generate the certificates if external Certificate Authorities are not an option or not at your disposal.

3.4 Client Connection Failover

TL;DR: By leveraging Client connection capabilities which are part of the standard connection libraries since Postgres 10, the client directly connects to the correct postgres, which improves performance and security features and is used as the default blueprint. Optionally, the design can be expanded with keepalived, HAProxy and PgRoute66, which can expose the cluster through a single VIP, expanding capabilities with `follow the master` and `Read Only scalability` capabilities.

Most if not all current versions of postgres clients natively support a feature called Client Connection Failover, which allows for configuring multiple hosts, and a role (primary, standby, preferred_standby, etc.) to connect to. Configuring Client Connection Failover makes the client loop over the hosts until a host with the expected role is detected. For initial connections, and during connection issues / failover.

Leveraging these capabilities adds some benefits to the solution, mainly :

- A single solution to meet requirements for both cloud[5] and on-premise[6]
- client has a direct connection to postgres, which delivers
  - A higher performance: No extra hops, no extra kernel or application logic to redirect network packets
  - No need for SAN certificates

- Option to run with or without stolon-proxy[7]:
  - With stolon-proxy there is absolute protection against split brain issues, and more direct traffic routing during switchover
  - Without stolon-proxy the correct source address is known to Postgres which allows for narrower host-based access control, and correct representation of the source address in logging, pg_stat_activity, etc.
- Option to run with a routing cluster consisting of Keepalived, HAProxy and PgRoute66[8] for VIP, `follow the master` and `Read Only Scalability` capabilities.

3.5 PgQuartz[9]

TL;DR: Neither cron or bash are bad by nature, but they remain limited in the clusterwide options, and open a door to spaghetti code and error-prone issues. PgQuartz is closer to the nature of scheduling requirements for postgres and delivers a very clean and sustainable solution for scheduling Postgres jobs, both on servers and Kubernetes.

---

[4] https://github.com/MannemSolutions/chainsmith
[5] Don't manage VIP's on the cloud
[6] NLB's are extra components, especially on-premise)
[7] Even with stolon-proxy, the requirement for Client Connection Failover remains. The client still needs to be able to redirect to another host.
[8] https://github.com/MannemSolutions/pgroute66
[9] https://github.com/MannemSolutions/PgQuartz

Many (basically all) solutions use cron as their main weapon of choice for scheduling jobs on servers. Cron was initially developed in 1975, and it still makes a lot of sense. But cron is a very generic scheduling tool and as such relies heavily on (bash) scripting for capabilities required by a specific job. Which is not bad in the 90's, but it is severely limited against use cases in the year 2022 and beyond. PgQuartz is built to fix that.

Highlights:

- Generic capabilities required to schedule a Postgres job, such as 'only run on a master', and 'only run once' should be available at the job definition level (configure time) instead of scripted for every job.
- Furthermore, psql is great, but parsing a query from bash to psql, and starting a new psql process for every query while parsing the output in bash is very error prone on it's own. This should be a basic capability of a scheduling tool, connect once, and leverage that connection to run queries, process output and run some more.
- Adding proper error handling, and reporting back if errors occur should be part of a well written bash script, but this too is cumbersome, error prone and there are just too many bad implementations of this. Meanwhile it is very straight forward. The job is meant to do something, so defining a simple check if it has actually succeeded should be very straight forward too.

3.6 PgFga[10]

PgFGA is an agent that manages PostgreSQL objects such as users, databases and extensions. The most crucial part of PgFga is that is manages Fine Grained Authentication, where AD groups can be granted permissions on databases. PgFga uses 2 sources of information, one is the configuration file, which allows for defining objects that should exist, and the other is for an LDAP which allows for defining groups, users, roles and permissions. PgFga runs periodically and combines these 2 sources into a 'configured state' after which it will change the running state to reflect that configured state. This allows for the following features:

- You can define and stage the initial state to allow the Applicational operators to take it from there (e.a. create schema, load initial data, etc.)
- You can define RBAC in AD and PostgreSQL users and permissions will follow
  - New users automatically added
  - Old users automatically cleaned

PgFga makes PgVIllage into a self-service solution without the requirement to grant permissions that your customer should no have.

3.7 PgWatch2

TL;DR: PgWatch is based on a solid foundation (Grafana) and a flexible architecture. By enabling PgWatch2, the power of the Trend Analysis and alerting capabilities of Grafana bring deep insight into the operations of the Postgres database. PgVillage allows to run PgWatch2 next to the database, or as a separate monitoring system.

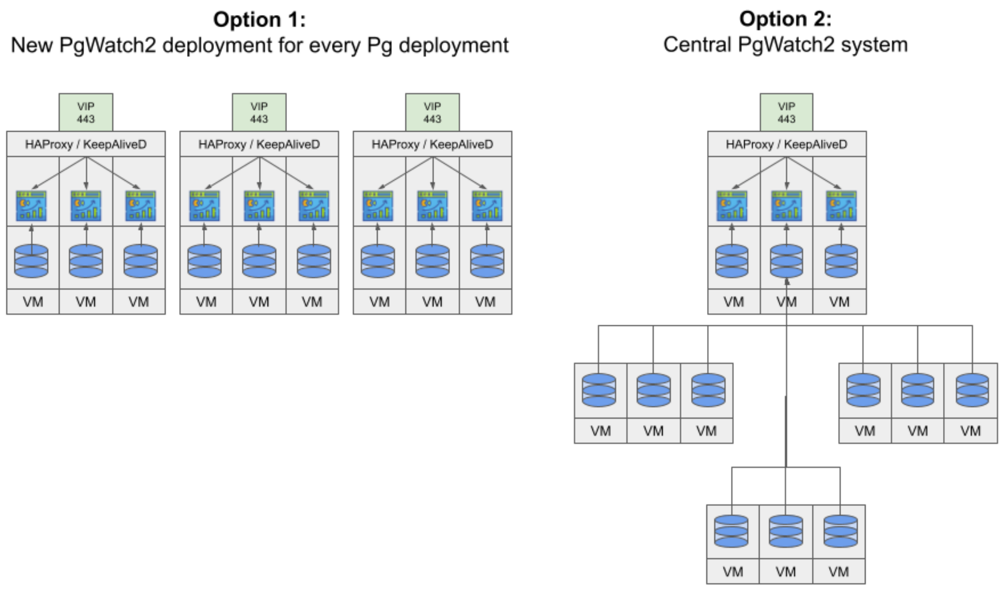
A proper monitoring solution should bring insights into the operation of the monitored systems through both trend analysis and alerting. Furthermore, Prometheus and Grafana are primary tools

[10] https://github.com/MannemSolutions/pgfga

leveraged by organizations for monitoring their landscape, and as such, monitoring for specific tools like Postgres should integrate well with these tools.
PgWatch2 checks all of these required boxes, and is easy to deploy next to your Postgres. The tool has a flexible architecture and can be easily integrated in your monitoring solution in any way you please.

Our blueprint supports 2 options to run PgWatch2:

1. Next to your database (1-on-1 deployment)
2. As a central monitoring system for your Postgres estate



3.8 PgAdmin4

As a web based management tool, PgAdmin4 complements PgWatch2 into a full featured Management and Monitoring solution enabling DBA’s proper and easy administration of the Postgres estate.

4 Deployment options

4.1 Compute

The blueprint design can be deployed in 2 distinct ways:

- On VM's, deployed with Ansible
- On Kubernetes, deployed using a helm chart (See Roadmap)

4.2 Security

The default deployment uses a self generated certificate chain which enables mTLS providing both encryption in transit and secure authentication for all internal communication of the cluster and application traffic. If required, the Certificate SIgning Requests are available and can be used to alternatively request server and client certificates signed by a Certificate Authority, and once available the self generated chain can be replaced.
For operators, Postgres can be integrated with Active Directory or OpenLDAP to leave authentication and authorization with the federated solution.
If required, Postgres can be deployed on servers with LUKS disk encryption[11]

4.3 Download and support

PgVillage is delivered as a combination of a lot of Open Source projects and as such it can be downloaded, used, changed at will, etc. Should you change the code please also contribute your ideas and changes upstream, so that the PgVillage ecosystem can benefit from them too.

Alternatively you can request support.
To bring the premiere support that our customers would require, Mannem Solutions collaborates with our partner ecosystem with contributors in many of the used Open Source projects, including PostgreSQL, WAL-G and Stolon.

And finally, we can also enable you to get started with a high pace. We can bring in the best expertise to enable you in any way necessary.

Our sole mission is to make you successful with running PostgreSQL in your estate. And for us both to have fun while doing so.

[11] I would like to see if we could use Cybertec's version with TDE…