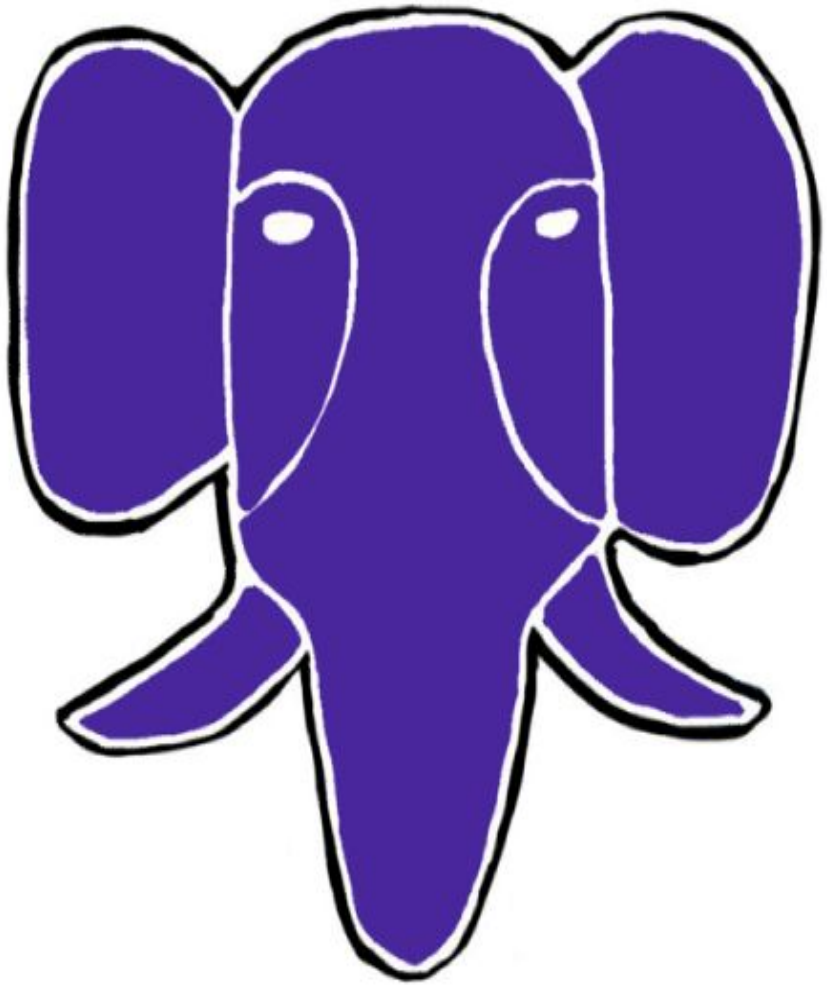
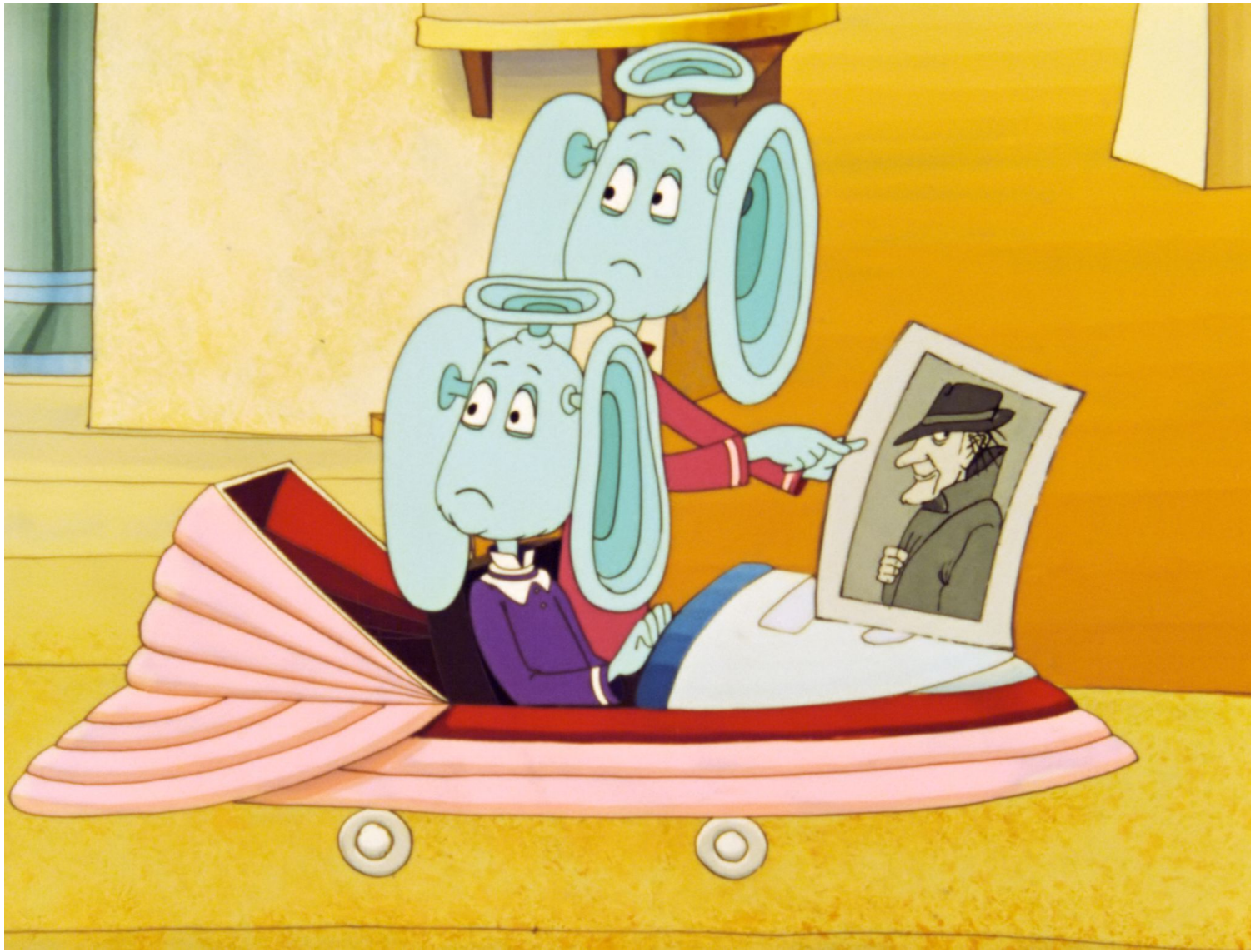
Евгений Аристов

Открытый урок Pgbouncer - connecting clients



https://aristov.tech



Аристов Евгений Николаевич

Founder & CEO aristov.tech
25 лет занимаюсь разработкой БД и ПО

Архитектор высоконагруженных баз данных и инфраструктуры

Спроектировал и разработал более ста проектов для финансового сектора, сетевых магазинов, фитнес-центров, отелей.

Сейчас решаю актуальные для бизнеса задачи: аудит и оптимизация БД и инфраструктуры, миграция на PostgreSQL, обучение сотрудников.

Автор более 10 практических курсов по PostgreSQL, MySQL, Mongo и др..

Автор книг по PostgreSQL. Новинка **PostgreSQL 16: лучшие практики оптимизации**

Правила вебинара

Задаем вопрос в чат

Вопросы вижу, отвечу в момент логической паузы

Если есть вопрос голосом - поставьте знак ? в чат

Если остались вопросы, можно написать мне через сайт aristov.tech

https://aristov.tech

Маршрут вебинара

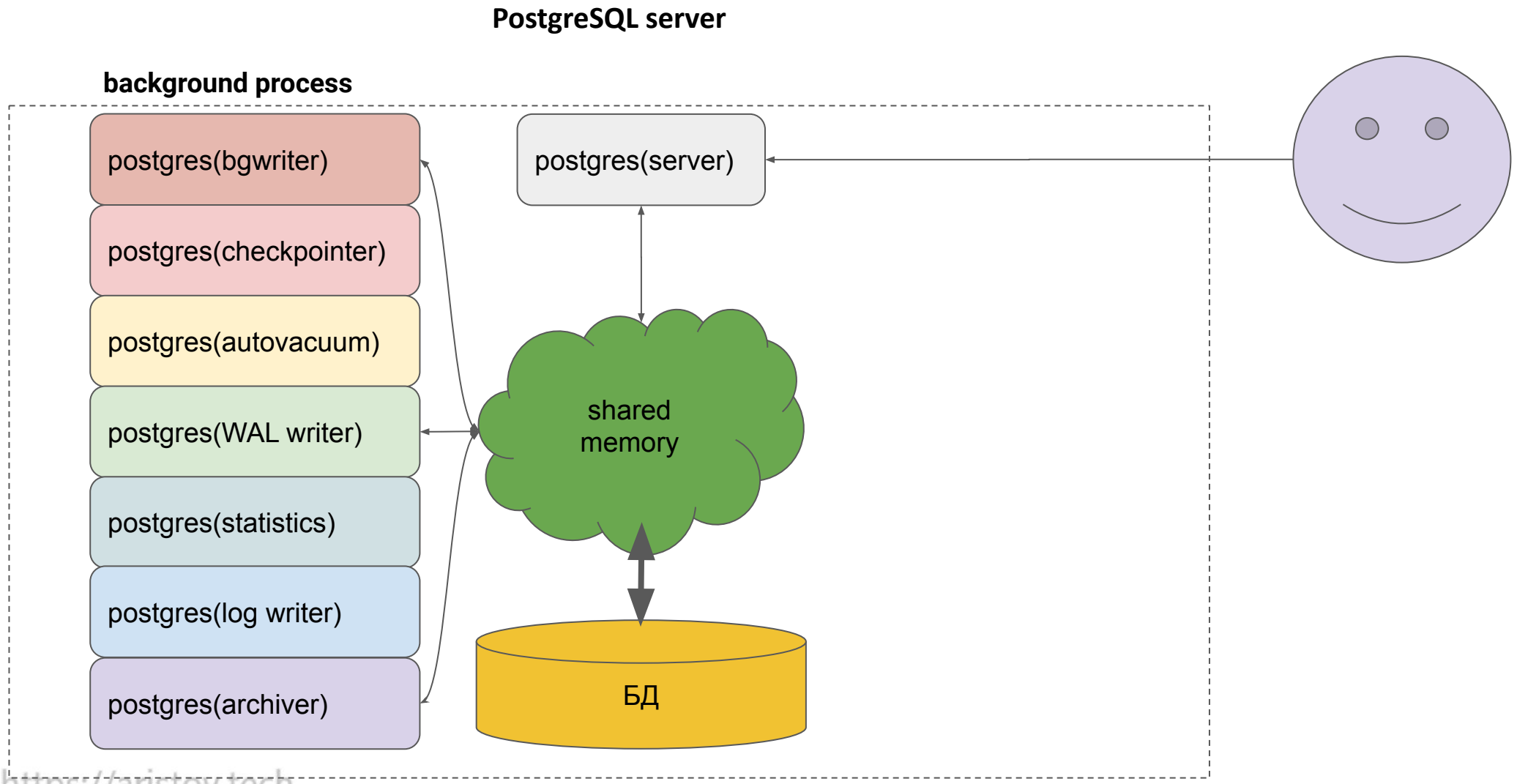
- Принципы подключения в PostgreSQL - pg_hba, fork backend process, work_mem, проблематика
- Назначение и виды пуллконнекторов
- Pgbouncer - принципы работы, настройка и возможный troubleshooting
- Проблематика idle и idle in transaction - исследование и рекомендации
- Коротко о проекте aristov.tech
- Розыгрыш скидки 30% на курс
- Ответ на вопросы

https://aristov.tech

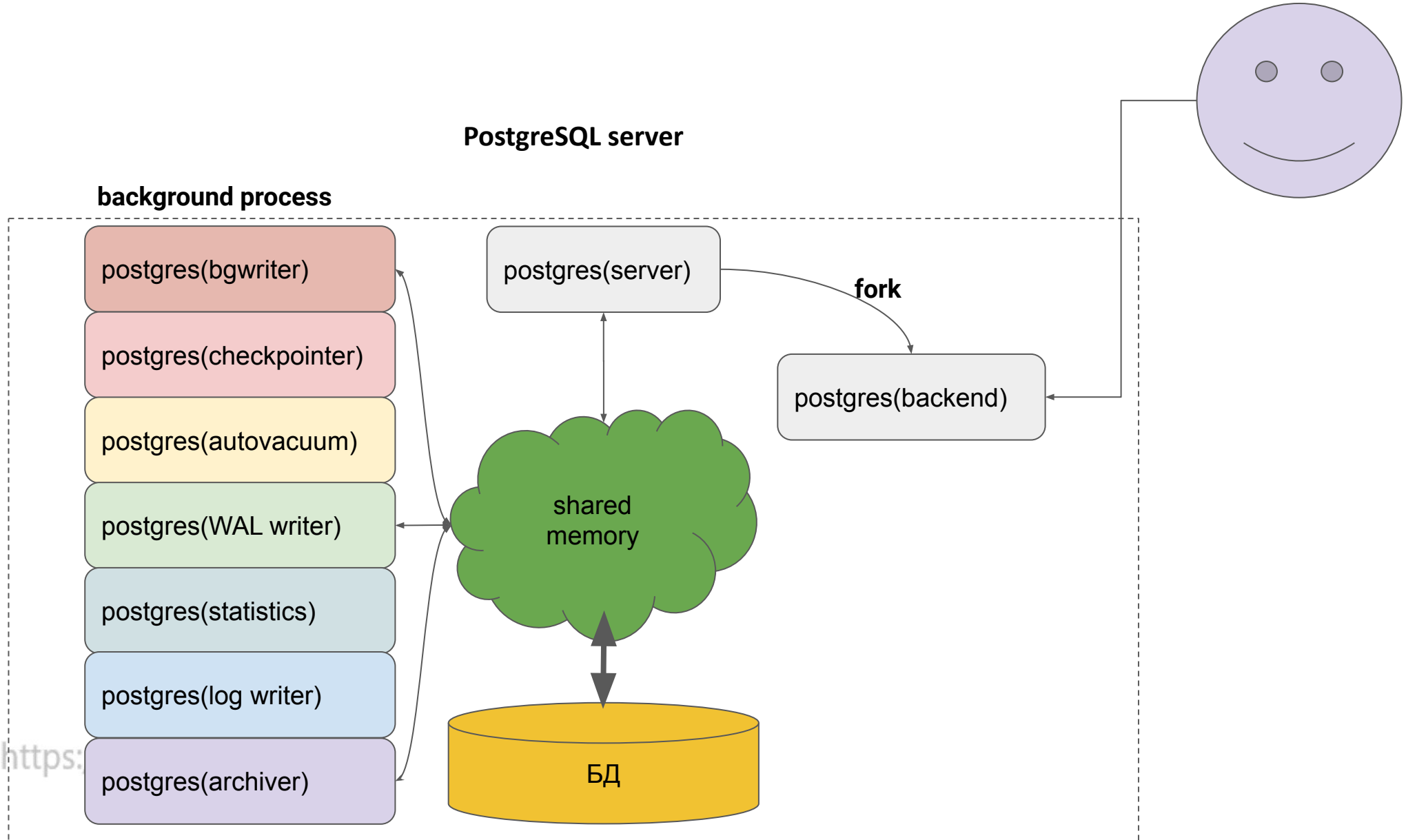
Проблематика

https://aristov.tech

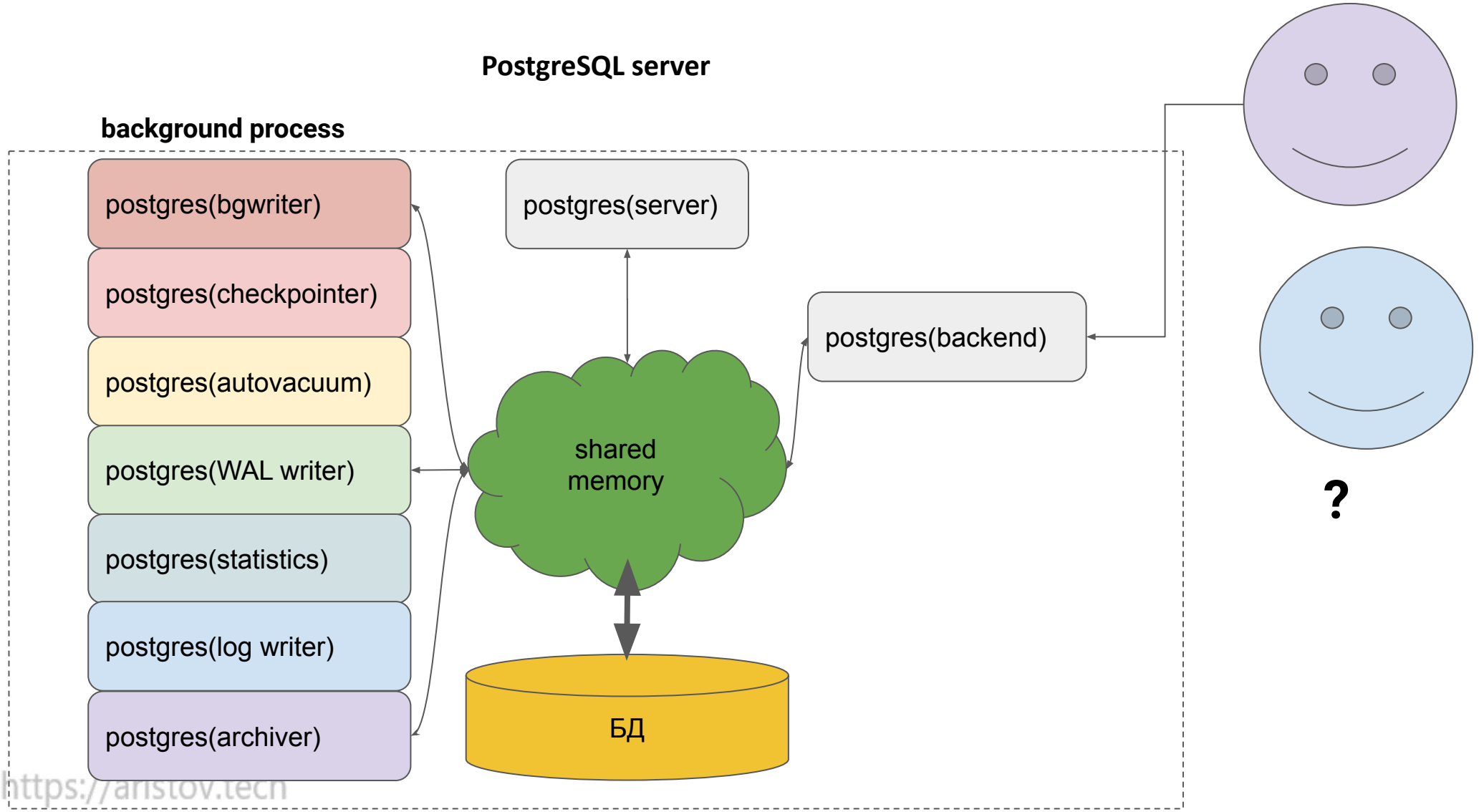
Серверные процессы и память



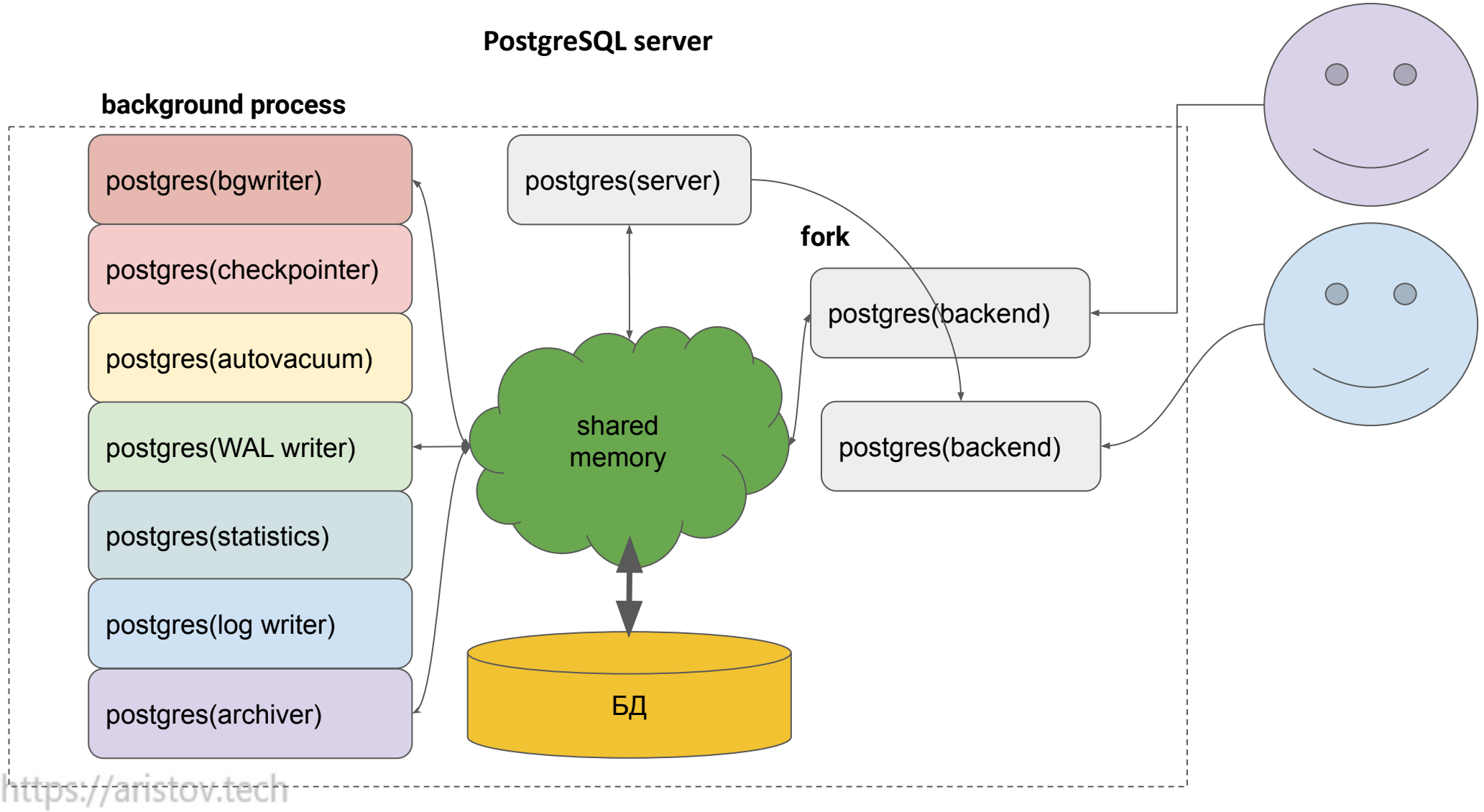
Серверные процессы и память



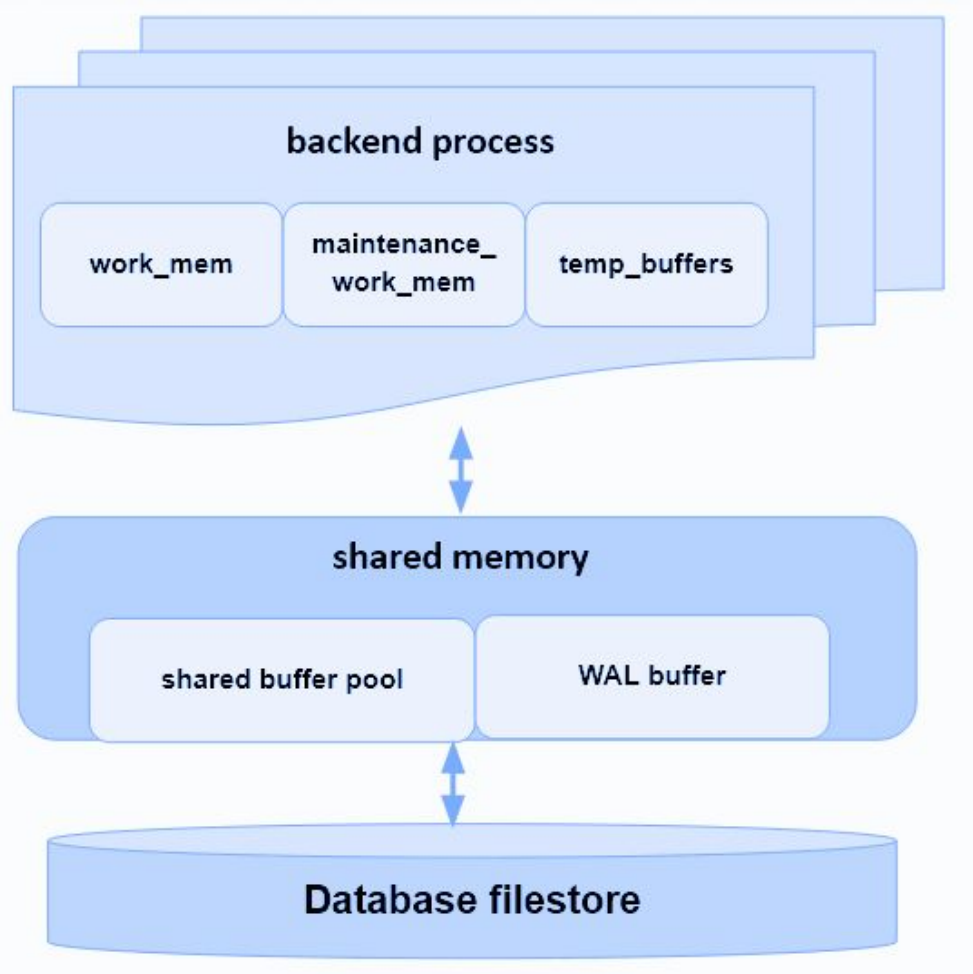
Серверные процессы и память



Серверные процессы и память



Кроме это выделяется память для каждой сессии



- принадлежит КАЖДОМУ backend процессу
- **work_mem (4 MB)**

  эта память используется на этапе выполнения запроса
- **maintenance_work_mem (64MB)**

  используется служебными операциями типа VACUUM и REINDEX
- **temp_buffers (8 MB)**

  используется на этапе выполнения для хранения временных таблиц

Нюансы

Принадлежит каждому backend процессу:

- **work_mem (4 MB)**

  эта память используется на этапе выполнения запроса для сортировок строк, например ORDER BY и DISTINCT - выделяться может неоднократно!!!

- **maintenance_work_mem (64MB)**

  используется служебными операциями типа VACUUM и REINDEX

- **выделяется только** при использовании команд обслуживания в сессии
- **temp_buffers (8 MB)** используется на этапе выполнения для хранения временных таблиц -

При превышении **work_mem** или **temp_buffers** - дальше идем в temp tablespace.

https://aristov.tech

Общая формула настройки памяти

Общая формула:

ОЗУ > 2Gb + Shared buffers + max_connections * (work_mem + temp_buffers) + maintance_parallel_workers * maintance_work_mem.

С учетом того, что work_mem может выделяться несколько раз.

Правда не весь объем сразу резервируется, но есть шанс OOM killer.

https://aristov.tech

Варианты доступа к кластеру

Подключение к PostgreSQL

На предыдущем занятии мы установили Постгрес на ВМ.

После установки Постгрес запускается с такими параметрами:

```
aeugene@Aeuge:/mnt/d/download/00 SQL s 0$ pg_lsclusters
Ver Cluster Port Status Owner    Data directory              Log file
16  main    5433 online postgres /var/lib/postgresql/16/main /var/log/postgresql/postgresql-16-main.log
```

Одна инсталяция инстанса Постгреса называется кластером

При этом при установке создается Linux пользователь postgres

Он предназначен для запуска кластера Постгреса и является владельцем всех файлов, относящихся к Постгресу - исполняемых файлов, файлов данных и логов

По умолчанию к этим файлам кроме него могут получить доступ только суперпользователи Линукс - root и группа пользователей, имеющая право на запуск утилиты sudo - позволяющей повысить свои личные права до прав суперпользователя

Подключение к PostgreSQL

Доступ к Постгресу после запуска возможен только через psql и Unix socket

Это означает, что пароль от единственного пользователя СУБД НЕ будет запрашиваться при входе в Постгрес, вместо этого Постгрес спросил у ОС - авторизован ли такой пользователь в ОС.

В нашем случае суперпользователь БД также имеет имя postgres и если мы в Линукс перейдем от нашего пользователя aeugene к пользователю postgres, повысив свои права до суперпользователя Линукс и дав команду переключиться на пользователя postgres, то при запуске утилиты psql Постгрес спросит у Линукса авторизован ли пользователь postgres и пустит внутрь СУБД без пароля. Иначе доступ мы не получим.

https://aristov.tech

Подключение к PostgreSQL

Для проверки подключения необходимо выполнить 3 шага:

Набрать в консоли команду для перехода под пользователя postgres

sudo su postgres

2. Зайти в утилиту управления СУБД

psql

3. Посмотреть параметры подключения

\conninfo

Второй вариант - повысить свои права до суперюзера и подключиться:

sudo -u postgres psql

https://aristov.tech

Варианты аутентификации. подключение по сети (протокол TCP/IP)

Кроме подключения по Unix Socket стандартным подключением к СУБД является подключение по сети (протокол TCP/IP)

Для того, чтобы зайти по сети мы должны отредактировать два конфигурационных файла:

hba_file.conf - настройки встроенного в Постгрес файрвола

postgresql.conf - настройки Постгреса, в том числе подключений извне

Расположение этих файлов зависит от типа и варианта ОС. Посмотреть, где они расположены можно из утилиты psql:

show hba_file;

show config_file;

P.S. чтобы вставить в консоль скопированные команды в VirtualBox нужно нажать Ctrl+Shift+V

Варианты аутентификации

В Ubuntu данные файлы расположены:

/etc/postgresql/16/main/pg_hba.conf

/etc/postgresql/16/main/postgresql.conf

Посмотрим на настройки файрвола, для этого:

выйдем из psql

\q или exit

обратите разницу на строку подключения при нахождении в Линукс и в psql

postgres@postgres (Linux) VS

postgres=# (psql - пользователь СУБД postgres, не Линукса)

Команды Линукса в psql выполнятся не будут и наоборот

Варианты аутентификации

Используем утилиту просмотра файлов cat под текущим пользователем Линукс postgres

cat /etc/postgresql/16/main/pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            scram-sha-256
```

Важно! Открывать только сетевое подключение в доверенной локальной сети!

НЕ рекомендуется открывать доступ в интернет!

P.S. если комбинация Alt+Tab для переключения окон не выпускает нас из ВМ в VirtualBox - необходимо сначала нажать кнопку правый Ctrl для освобождения удерживаемого фокуса

Варианты систем шифрования паролей

с 14 версии используется система шифрования паролей SCRAM-SHA-256,

в более ранних версиях - система MD5

! они не совместимы, при обновлении кластера с 13 на 14+ версию нужно это иметь ввиду.

! не стоит использовать систему PASSWORD - Пароль передается в открытом виде!

Подключение к PostgreSQL

Чтобы задать пароль необходимо выполнить:

Зайти в консоль

psql

2. Установить пароль для текущего пользователя

а) команда psql

\password

или

б) SQL команда

ALTER USER postgres PASSWORD '123';

Подключение к PostgreSQL

Теперь мы можем зайти по сети на localhost (127.0.0.1).

Для этого необходимо выполнить 3 шага:

выйти из psql

\q

2. подключиться указав хост для подключения

psql -h localhost

3. посмотреть статус подключения

\conninfo

Видим, что теперь мы вместо Unix Socket подключились по сети с локалхоста

P.S. psql мы запускаем из линукса из под пользователя postgres, а НЕ student. Если закрыли консоль, переключиться на пользователя postgres можно выполнив sudo su postgres

Подключение извне

Подключение к PostgreSQL ИЗВНЕ

Для подключения извне ВМ, нам необходимо сделать несколько больше шагов:

включаем listener в postgresql.conf (обычно 2 сети - внутренняя и внешняя - интернет. Подключение через интернет категорически не рекомендуется)

listen_addresses = '*' # IP адреса, на которых принимает подключения Постгрес, например localhost, 10.*.*.*;

второй вариант

alter system set listen_addresses = ‘*’;

включаем вход по паролю в pg_hba.conf и меняем маску подсети

hostall all 0.0.0.0/0 scram-sha-256

в 13 и меньше версии scram-sha-256 -> md5

добавляем порт во внешний файрвол, используемый у вас в организации

https://aristov.tech

Подключение к PostgreSQL ИЗВНЕ

4. задаем пароль юзеру postgres

ALTER USER postgres PASSWORD '123';

5. Перегружаем сервер - обратите внимание, что при установке PostgreSQL запущен от имени root и рестарт от пользователя postgres мы сделать не можем (только stop и потом start). Правильно потом переконфигурировать пользователя на postgres при старте ВМ.

pg_ctlcluster 16 main restart

Вуаля

psql -h 104.197.151.20 -U postgres

Мы оказались внутри нашего кластера и можем посмотреть параметры подключения выполнив

\conninfo

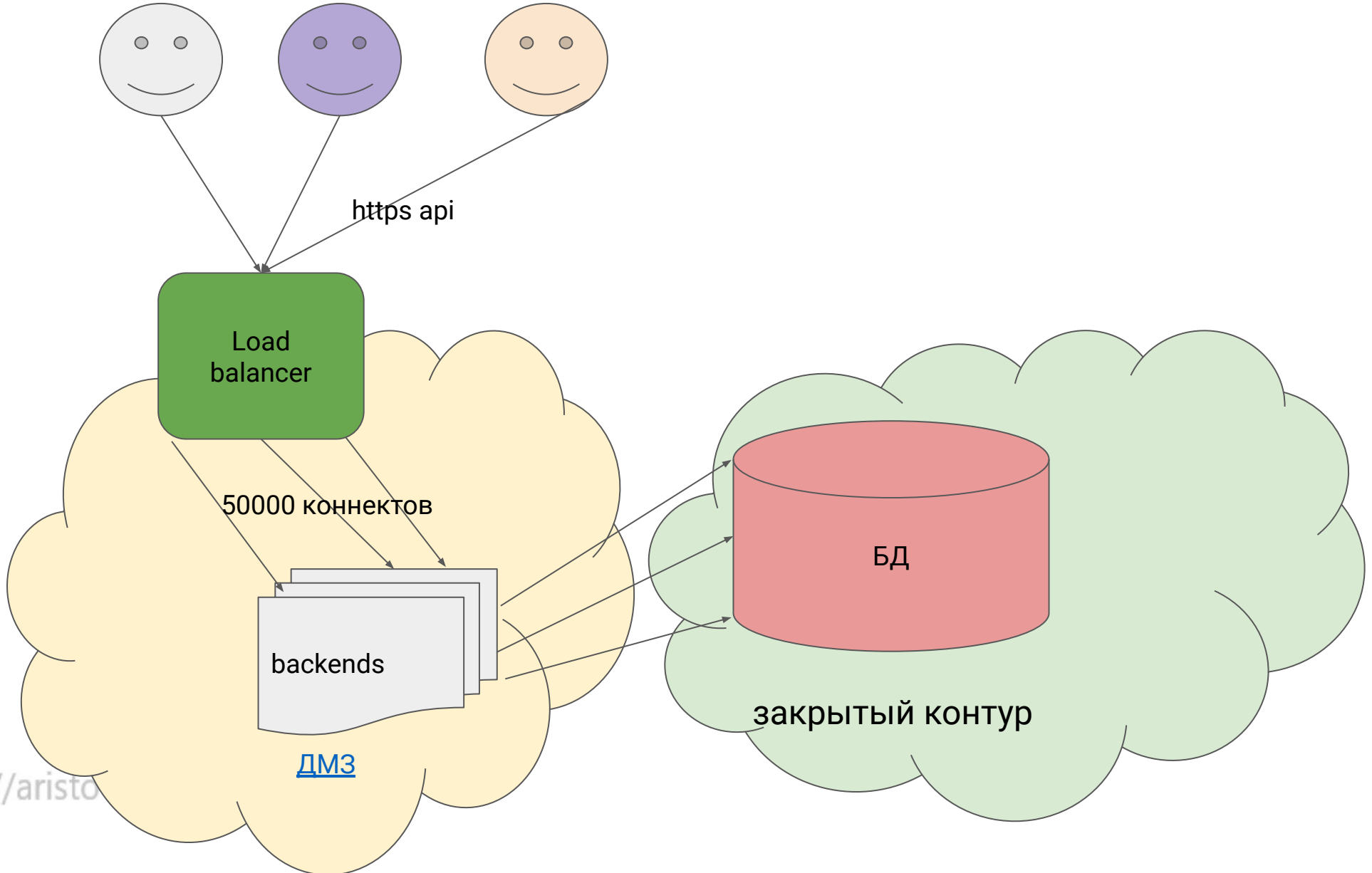
Подключение из других версий

Подключение к PostgreSQL другой версии

Так как у PostgreSQL открытый протокол обмена между кластерами и версиями, то никаких проблем при совместимости, начиная с версии 9.6, не наблюдается. Естественно, если вы используете psql 16 версии и подключаетесь к 10, то функционал будет доступен только 10 версии)

Текущая конфигурация

Текущая конфигурация



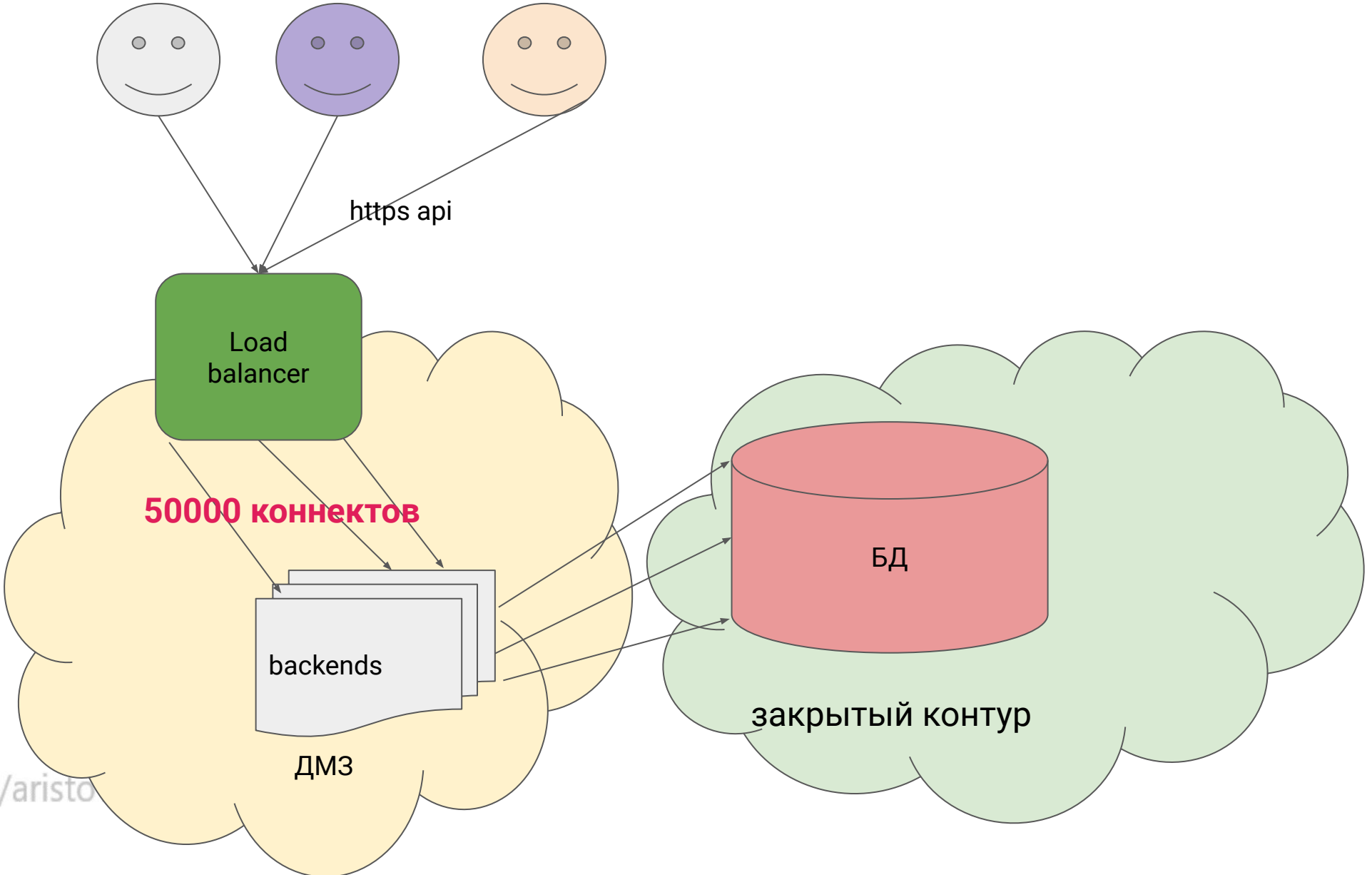
Проблематика

Несмотря на встроенный файрвол - Постгрес не средство защиты от DDoS и других атак:

- желательно работать только в доверенной зоне
- меняем стандартный порт
- аккуратнее поднимаем докер контейнеры с портом наружу -p 5432:5432 - лучше докер сеть и по ней доступ к бэкендам (касается и k8s и port-forward как вариант)
- сложные пароли - ибо никто ботов не отменял

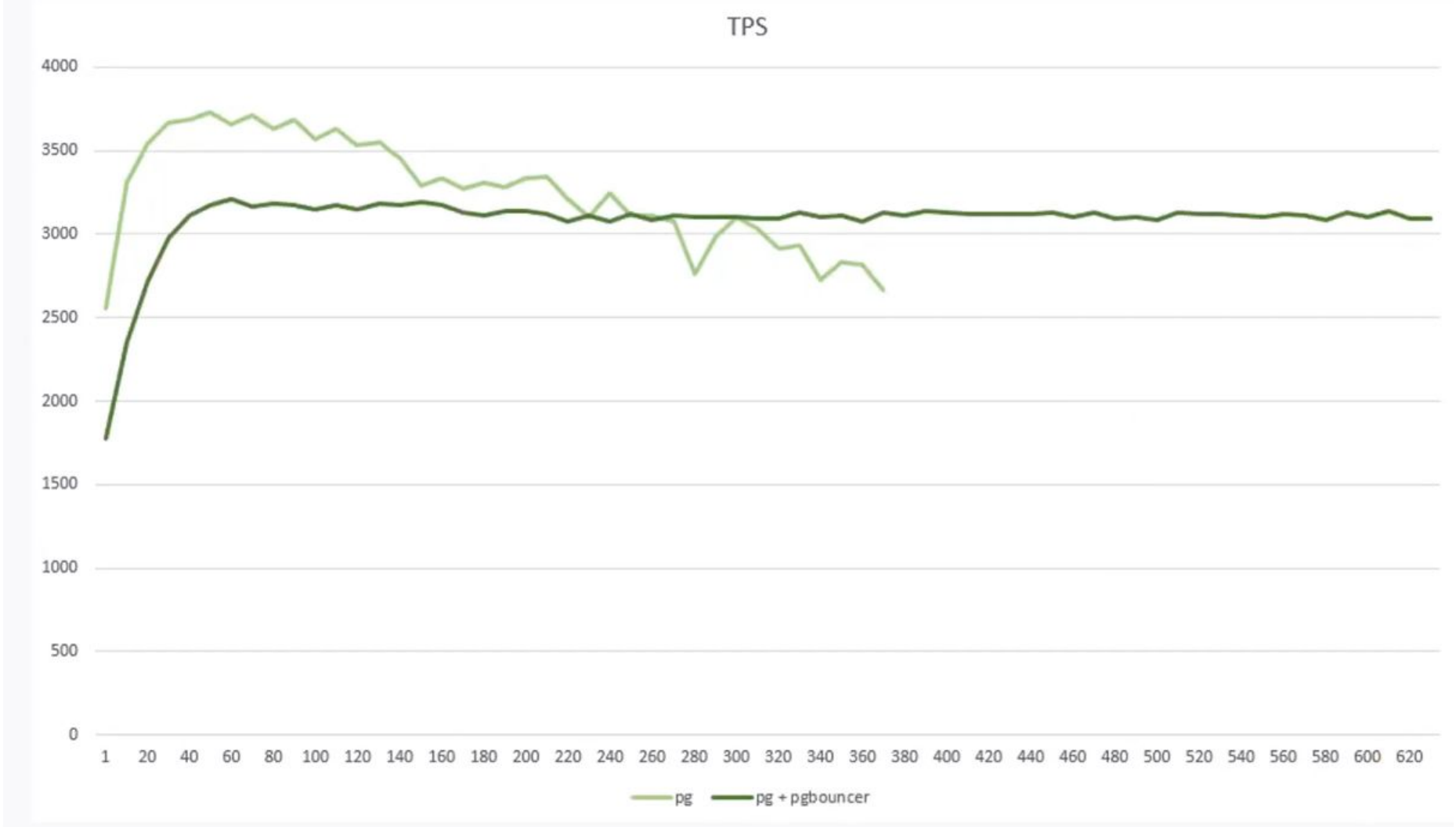
https://aristov.tech

Чем плохо много коннектов?



600++ conn

600++ conn



https://aristov.tech

Классические варианты пулеров

Вспоминаем, что каждый коннект это клон процесса + выделение памяти.

Поэтому используем пулер (pooler). Легковесные постоянные коннекты от бэкендов и постоянные соединения в БД.

- **Pgpool II**
- **pg_bouncer + haproxy**
- **Odyssey**

Ну или использовать готовые кластера

https://aristov.tech

Классические варианты

Pgpool II

- используется встроенный механизм репликации
- есть пул соединений
- балансировщик нагрузки
- высокая доступность (наблюдатель с виртуальным IP, автоматическое переключение мастера)

Как раз излишек функционала и есть основная проблема производительности Pgpool2

https://severalnines.com/database-blog/guide-pgpool-postgresql-part-one

https://aristov.tech

Классические варианты

PgBouncer

PgBouncer - пул соединений:

- легковесный - 2 Кб на соединение
- можно выбрать тип соединения: на сессию, транзакцию или каждую операцию
- онлайн-реконфигурация без сброса подключений

PgBouncer спроектирован однопоточным. Он сделан максимально простым и в этой простоте масштабируемость не присутствует как класс. Вот в CPU мы видим 97 % загрузки. База при этом не так, чтобы занята, но Bouncer не успевает передавать байтики туда и сюда.

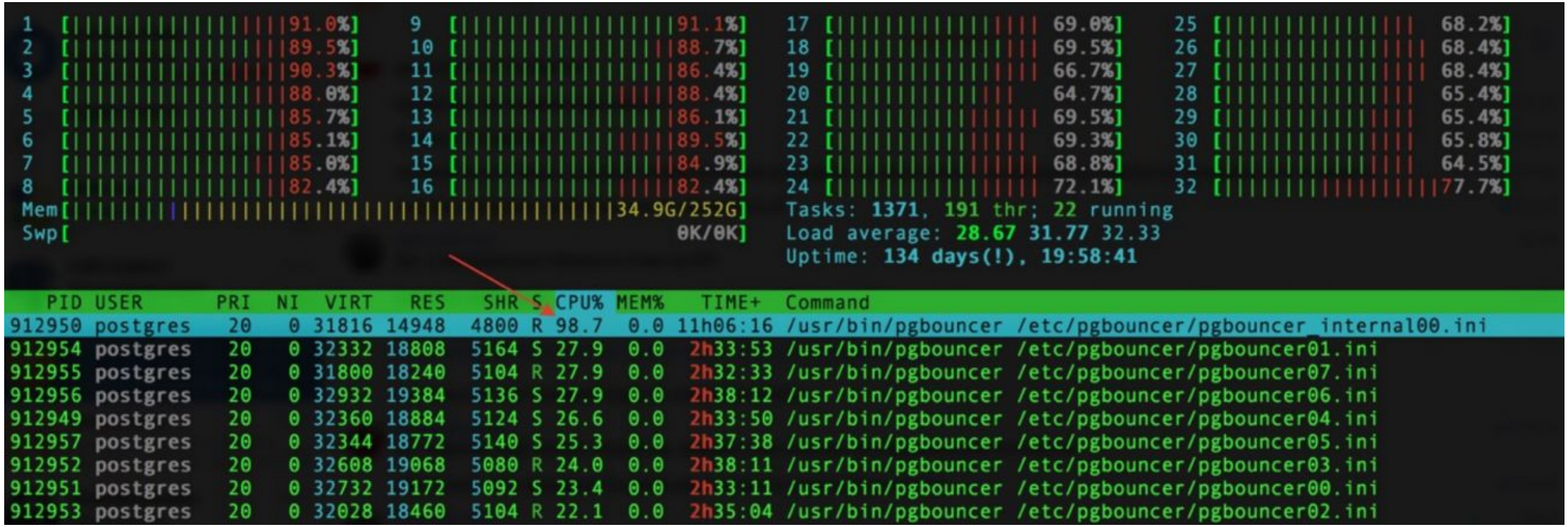
Сравнение

https://scalegrid.io/blog/postgresql-connection-pooling-part-4-pgbouncer-vs-pgpool/

https://aristov.tech

Классические варианты. Pgbouncer

Добавили reuseport и теперь можно в несколько копий, но..



пруф - https://habr.com/ru/articles/498250/

Классические варианты. Pgbouncer

Видим поднятый каскадом PgBouncer, где у нас есть внутренний Bouncer, который по-прежнему уперт в одно ядро.

Вы можете в каскаде обойтись без внутреннего PgBouncer, но тогда у вас будет connection pool внутри каждого процесса PgBouncer. И снова приходим к той же проблеме, что у нас много коннектов.

Внешний слой PgBouncer обычно используется для приема волны TLS-соединений. TLS-соединения – это операция, которая требует участие центрального процессора значительно больше, чем типичное перекладывание байтов из сокета в сокет. Т. е. для центрального процессора потоки данных, измеренные в байтах в секунду, они не заметны. Но необходимость выполнять криптографию при TLS handshake существенна.

https://aristov.tech

Классические варианты. Pgbouncer

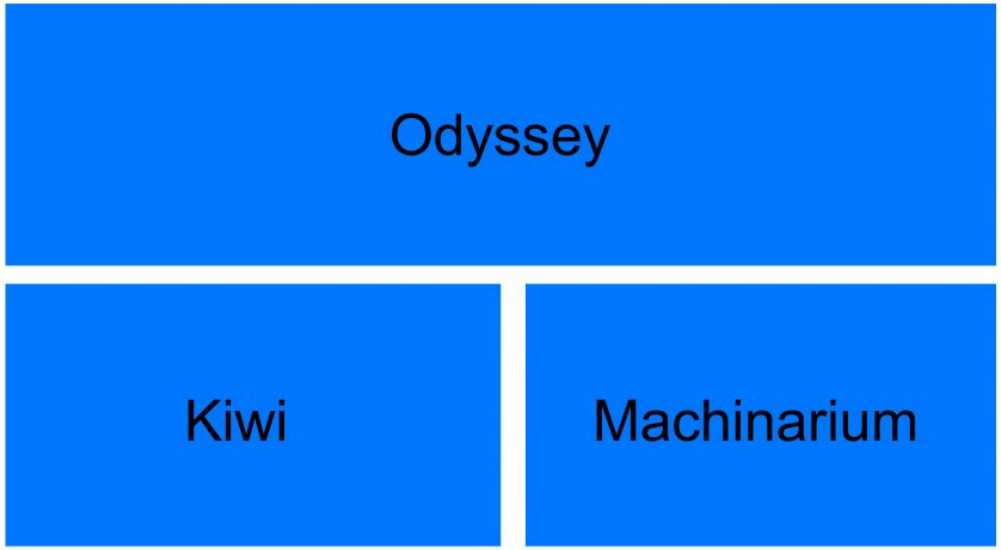
Если у вас приложение по какой-то причине бросило соединение с базой данных, то PgBouncer продолжит выполнять тот запрос, который выполнялся. На эту проблему обращали внимание много раз. Авторы Odyssey написали pull request. Он принят не был. Его рассматривают до сих пор. И это еще одна проблема, что PgBouncer не очень хорошо саппортился.

Пришлось изобретать свой велосипед.

Классические варианты

Odyssey

https://github.com/yandex/odyssey



разбор архитектуры - https://habr.com/ru/articles/498250/

https://aristov.tech

Другие варианты:

Новичок нашего пула:

pg agroal is a high-performance protocol-native connection pool

- High performance
- Connection pool
- Limit connections for users and databases
- Prefill support
- Remove idle connections
- Perform connection validation

etc..

https://github.com/agroal/pgagroal

https://aristov.tech

Другие варианты:

Суперновичок нашего пула:

Supavisor - https://github.com/supabase/supavisor

- Apache 2.0
- Highly available
- Connection buffering
- Cloud-native
- Multi-tenant
- Scalable
- Load balancing

1 млн коннектов:

https://supabase.com/blog/supavisor-1-million

https://aristov.tech

Пулеры ЯП

У каждого ЯП свои пулеры.

Java - Hikari - периодические проблемы

etc..

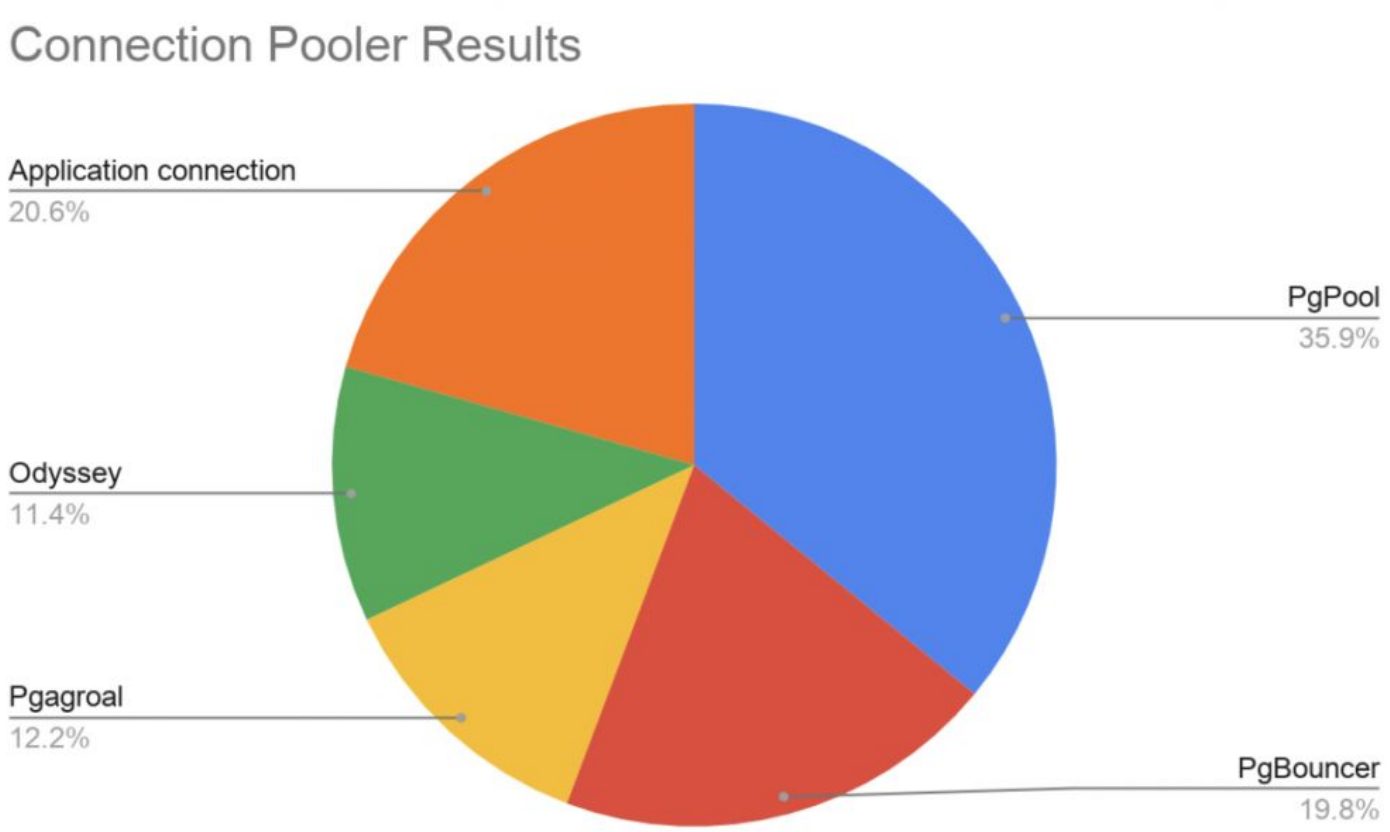
Общие проблемы:

- свой зоопарк
- держат коннект с конкретным инстансом - нет учета особенностей переключения между бэкендами
- сложность равномерно распределить нагрузку

https://aristov.tech

Опрос 3000 ДБА

https://www.enterprisedb.com/blog/what-3000-users-say-about-postgresql-tools-they-use



https://aristov.tech

Best practice

В зависимости от задачи и профиля нагрузки тестируем на дев стенде, помня об ограничениях, плюсах и минусах.

На что стоит обратить внимание:

- количество коннектов
- нагрузка на сеть/проц
- минимизация обмена данными по сети
- делать меньше запросов
- читать меньше данных
- обновлять меньше данных (несколько update insert в одну транзакцию)
- уменьшать обработку на лету - ХП
- проанализировать передачу данных
  - сколько данных было просканировано - сколько отослано
  - сколько было отослано - сколько использовано приложением

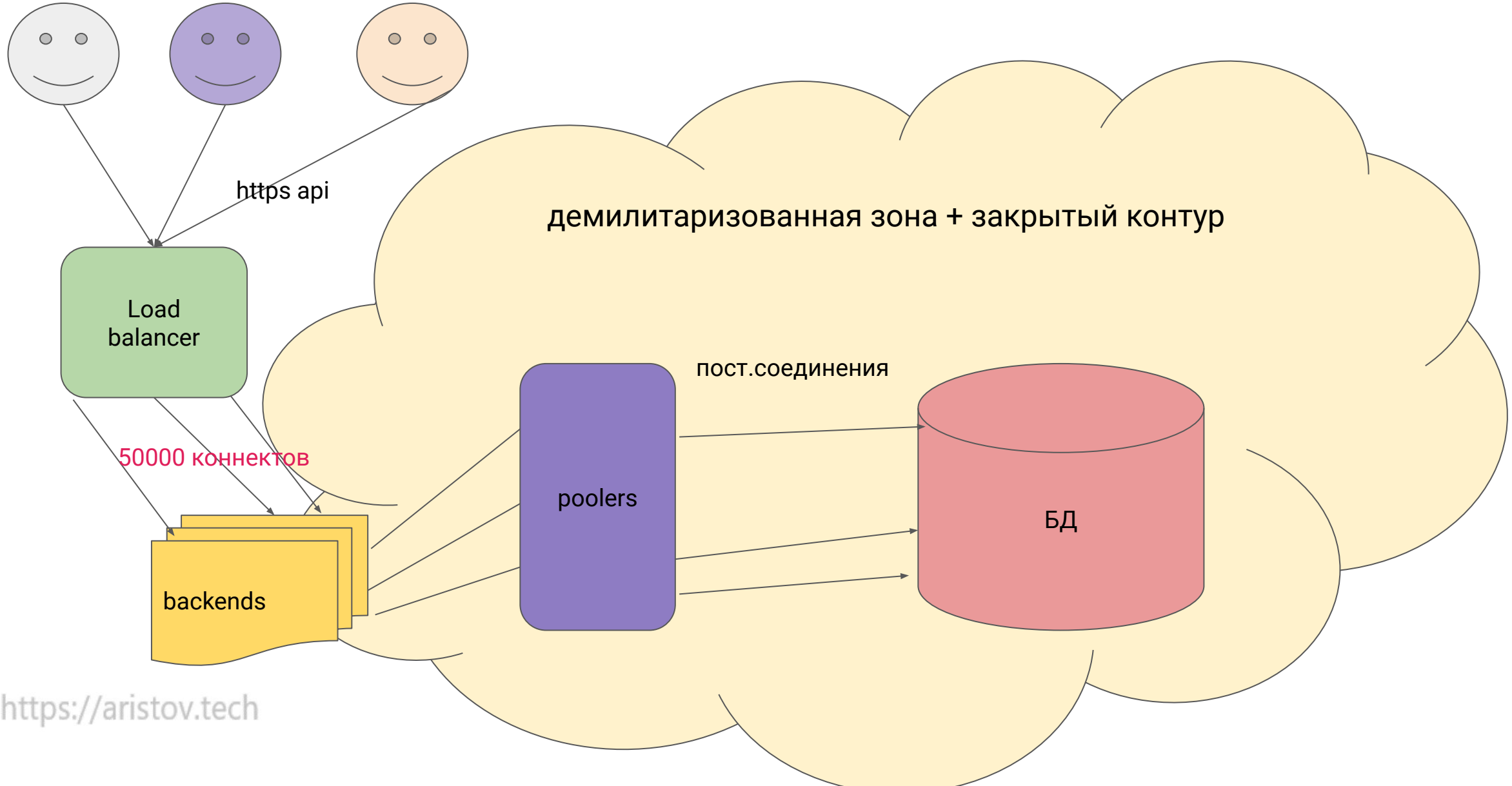
https://aristov.tech

Мое мнение

- бэкап снимаем с secondary
- возможно есть смысл в каскадной репликации при проблемах с производительностью сети
- не забываем про OLAP - делаем специальную реплику, возможно например подключить clickhouse
- георасределение нагрузки
- не забываем про LT-стенды (Стенд для нагрузочного тестирования: от DEV до PROD)

https://aristov.tech

Теперь все хорошо?



SSL и терминация трафика

Использование TLS/SSL (Secure Socket Layer) для защищенного соединения с PostgreSQL внутри закрытой периметрии (например, внутри безопасной сети или сети виртуальных машин) имеет свои плюсы и минусы:

Плюсы использования SSL:

- **Шифрование данных**: SSL обеспечивает шифрование данных между клиентом и сервером, что делает перехват и утечку информации более сложными для злоумышленников, даже если они имеют доступ к внутренней сети.
- **Доверие и безопасность**: Использование SSL помогает подтвердить подлинность сервера перед клиентом и создает доверительный канал для обмена данными. Это защищает от атак "человек посередине" (Man-in-the-Middle) и поддерживает целостность данных.
- **Соответствие стандартам и нормам**: В зависимости от вашей отрасли и законодательства, вам может потребоваться шифрование данных, даже если они передаются внутри закрытой сети. Использование SSL позволяет соответствовать стандартам безопасности данных.
- **Защита от внутренних угроз**: Внутренние угрозы, такие как злоумышленники внутри сети, могут попытаться перехватить данные или осуществить атаки на базу данных. SSL помогает уменьшить риски таких атак.

SSL и терминация трафика

Минусы использования SSL внутри закрытой периметрии:

- **Дополнительная нагрузка** на производительность: Шифрование и расшифровка данных может вызвать некоторую нагрузку на производительность сервера PostgreSQL. В закрытой сети эта нагрузка может быть излишней.
- **Сложность настройки**: Настройка SSL требует наличия корректных сертификатов, и это может быть сложно в случае внутренних сетей. Неправильная настройка может привести к проблемам соединения.
- **Увеличение сложности обслуживания**: Внедрение SSL увеличивает сложность конфигурации и обслуживания базы данных. Это может потребовать дополнительных шагов при развертывании и обновлении.
- **Затраты на ресурсы**: Создание и управление сертификатами требует времени и ресурсов. Возможно, вам потребуется наличие собственной инфраструктуры управления сертификатами.

В целом, использование SSL внутри закрытой периметрии имеет смысл, если вы стремитесь к обеспечению дополнительного уровня безопасности, защиты данных и соответствия стандартам. Однако следует тщательно взвесить плюсы и минусы в соответствии с требованиями вашей организации и сетевой инфраструктурой.

SSL и терминация трафика

Безопасность и СУБД: о чём надо помнить, подбирая средства защиты

Результаты тестирования:

	NO SSL	SSL
Устанавливается соединение при каждой транзакции		
latency average	171.915 ms	187.695 ms
tps including connections establishing	58.168112	53.278062
tps excluding connections establishing	64.084546	58.725846
CPU	24%	28%
Все транзакции выполняются в одно соединение		
latency average	6.722 ms	6.342 ms
tps including connections establishing	1587.657278	1576.792883
tps excluding connections establishing	1588.380574	1577.694766
CPU	17%	21%

- У нас дыра в безопасности!
- Ну хоть что-то у нас в безопасности

Обеспечение безопасности базы данных PostgreSQL

- Безопасность на сетевом уровне
- Безопасность на транспортном уровне
- Безопасность на уровне базы данных

https://aristov.tech

Балансируем нагрузку

Неплохо бы еще добавить HAPROXY для балансинга нагрузки

https://www.percona.com/blog/2018/10/02/scaling-postgresql-using-connection-poolers-and-load-balancers-for-an-enterprise-grade-environment/

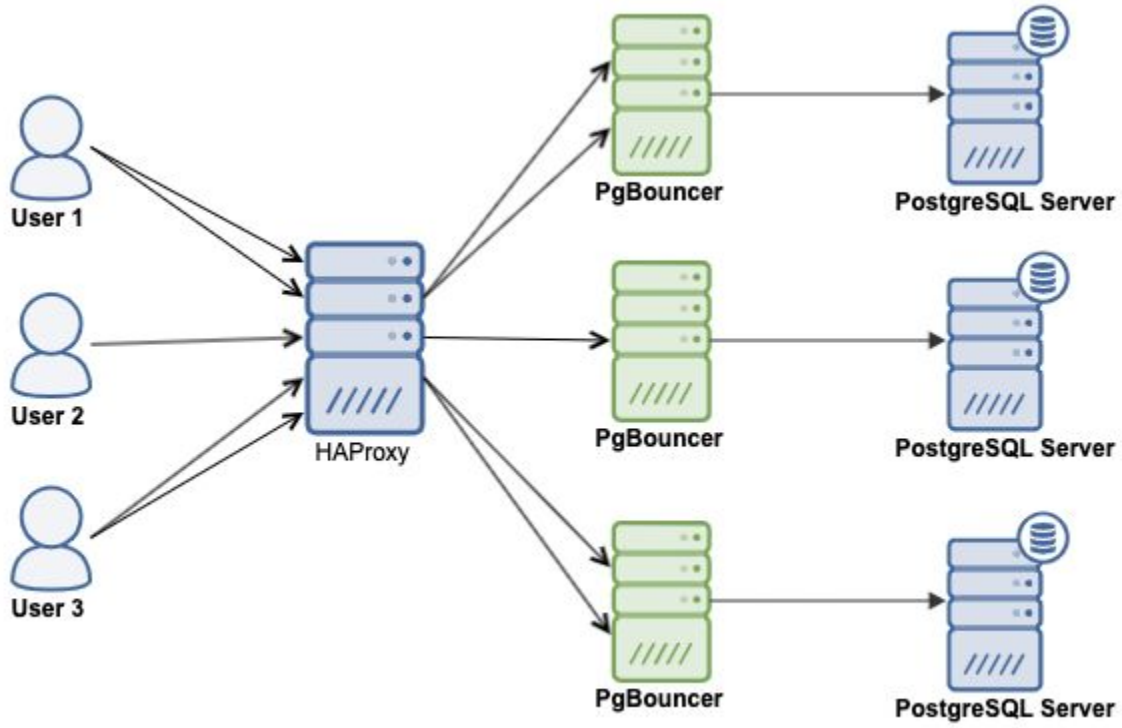
Еще варианты балансеров:

- используем или встроенный GLB в облако или варианты:
- NGINX https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/

3 варианта со своими + и -:

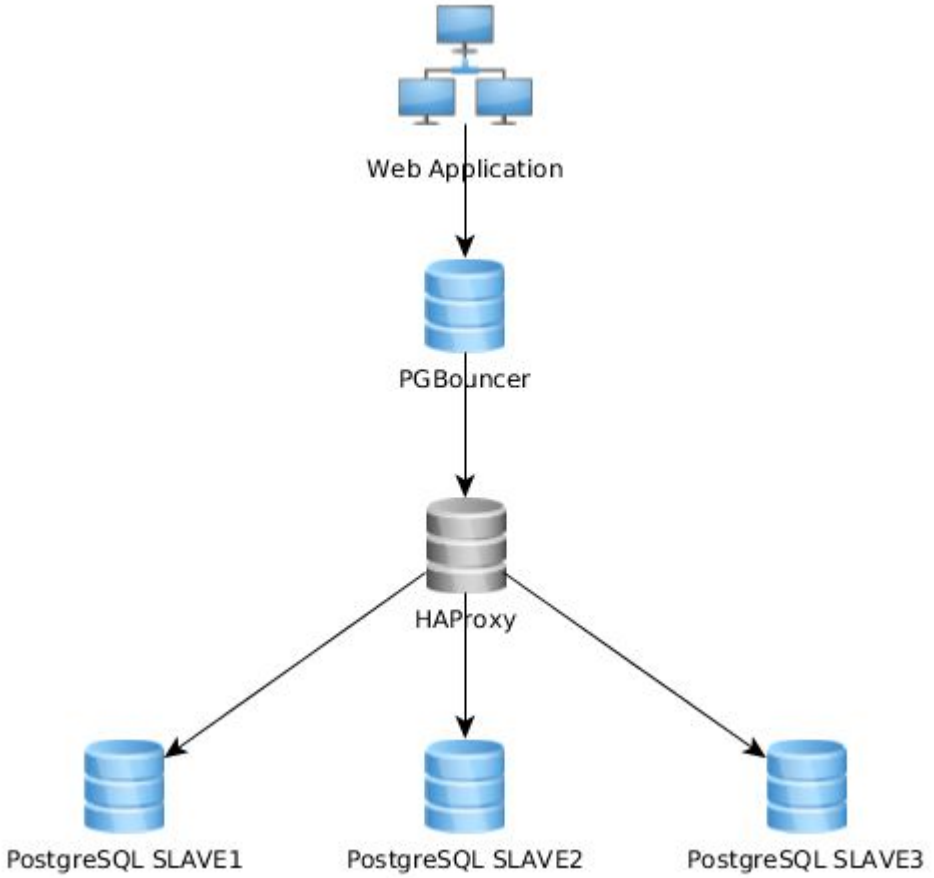
https://aristov.tech

Балансируем нагрузку



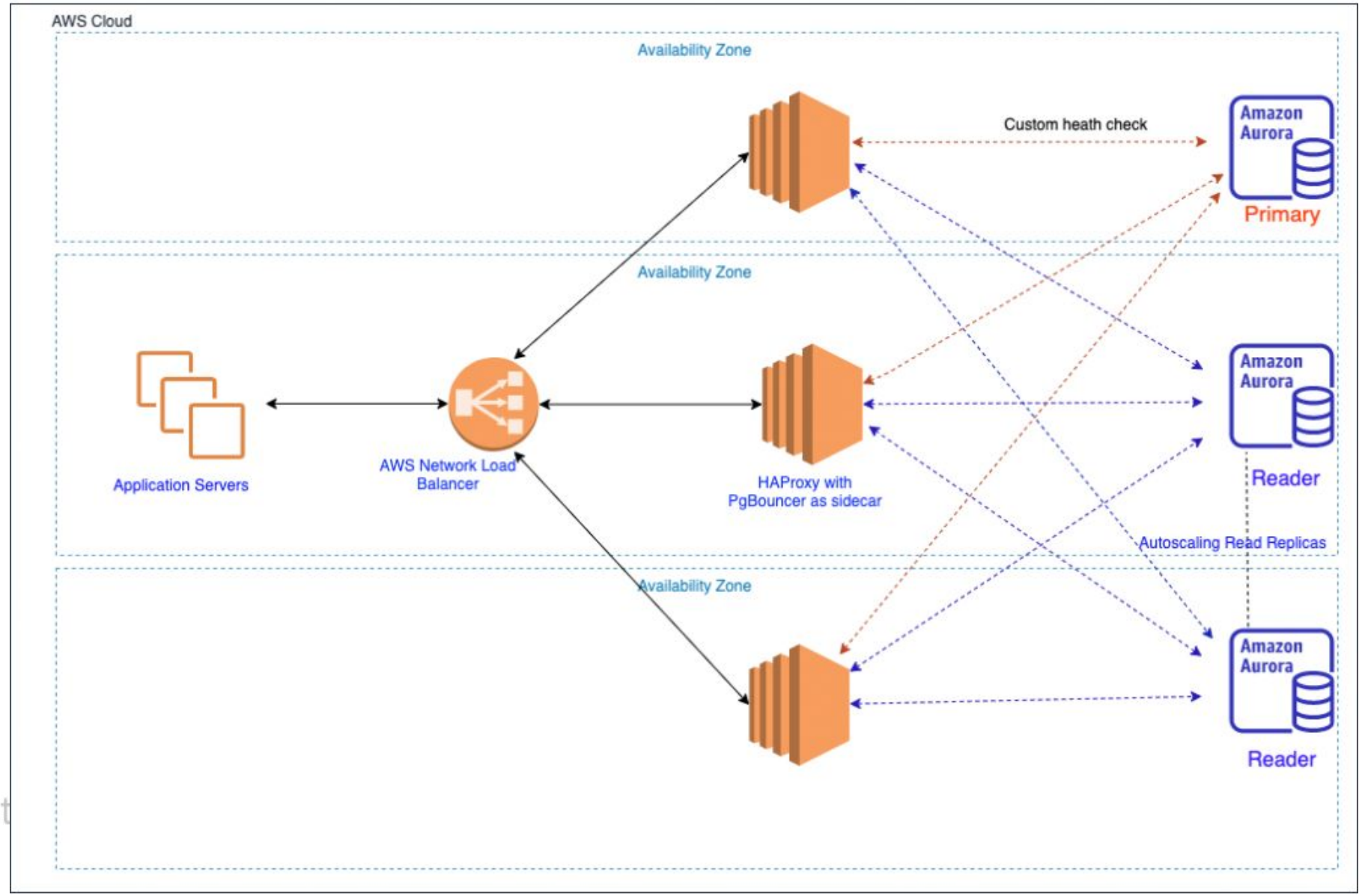
https://aristov.tech

Балансируем нагрузку



https://aristov.tech

Балансируем нагрузку



Балансируем нагрузку

А куда балансировать нагрузку?

У нас ведь 1 БД…

Разберем варианты на следующем вебинаре

https://aristov.tech

Практика

Тестирование производительности pgbouncer

А если под нагрузкой?

idle VS idle in transaction

https://aristov.tech

idle, idle in transaction - насколько плохо?

Запись видео: https://www.youtube.com/watch?v=wK8VSeDG_6I
В виде статьи: https://aristov.tech/blog/idle-vs-idle-in-transaction/
Скрипты на Питоне выложены на гитхаб: https://github.com/aeuge/aristov_tech

https://aristov.tech

Итоговый чеклист

https://aristov.tech

Checklist

- учитываем количество коннектов
- если больше 500 - выбираем пулер
- обязательно тестирование на вашем железе, объеме и профиле нагрузки
- учитываем рекомендации по сетевому обмену
- совместно с СБ выбираем модель SSL и терминации трафика
- выбираем модель балансинга в соответствии с вашим железом и бизнес задачами
- помним о проблемах долгих соединений

https://aristov.tech

Проект aristov.tech

https://aristov.tech

aristov.tech

Книга по 14 Постгресу (Архитектуре/16 Оптимизации) https://aristov.tech/#orderbook
Эксклюзивный **курс** по Оптимизации Постгреса 5 поток
https://aristov.tech/blog/kurs-po-optimizaczii-postgresql/
Со всеми **отзывами** без цензуры можно ознакомится по ссылке
https://aristov.tech/blog/otzivi-kurs/
Блог с популярными темами https://aristov.tech/blog
ТГ **канал** с новостями блога и проектаhttps://t.me/aristov_tech
Ютуб канал с интересными видео https://www.youtube.com/@aristovtech
Моя **группа** ДБА https://t.me/dbaristov - уже 350+ экспертов

Также за прошедшее время был запущен проект **менторинга** в котором уже участвует 15 лучших экспертов в отрасли ДБА/DevOps и разработки.
Ознакомится с преподавателями и проектом https://aristov.tech/mentorship
В направлении крутых **вакансий** уже 5 предложений от партнёров
https://aristov.tech/blog/vakansii-ot-partnerov-aristov-tech/
Всегда можно со мной связаться через сайт для консультация, аудита вашего проекта, менторинга, обучения и многого другого

aristov.tech

Розыгрыш книг и скидки 30% на курс (только очные варианты)

регистрируемся по форме:

https://forms.gle/9i8eRcSHGLnMXi3B7

Сам розыгрыш в db-fiddle

https://www.db-fiddle.com/f/43wbuUzv7YUAcS3aypkvvC/1

https://aristov.tech

ДЗ

ДЗ

1. Подписаться на канал
2. Подписаться на Ютуб
3. Вступить в мою группу ДБА
4. Посещать открытые уроки
5. Прийти на курс %)
6. Прокачаться с помощью менторинга
7. Устроиться на одну из вакансий 450+

Спасибо за внимание!

Следующий ОУ в январе 2025

Аристов Евгений