

PostgreSQL cluster высокой доступности под управлением Patroni для 1С

Единая точка входа организована Consul DNS
на Windows.



Немного о себе

- Работаю в крупной международной компании
- Мы используем 1С для бухгалтерского аутсорсинга в России и не только
- У нас около 200 баз самого разного размера
- Мы используем PostgreSQL и стремимся что бы наши сервисы работали стабильно и быстро

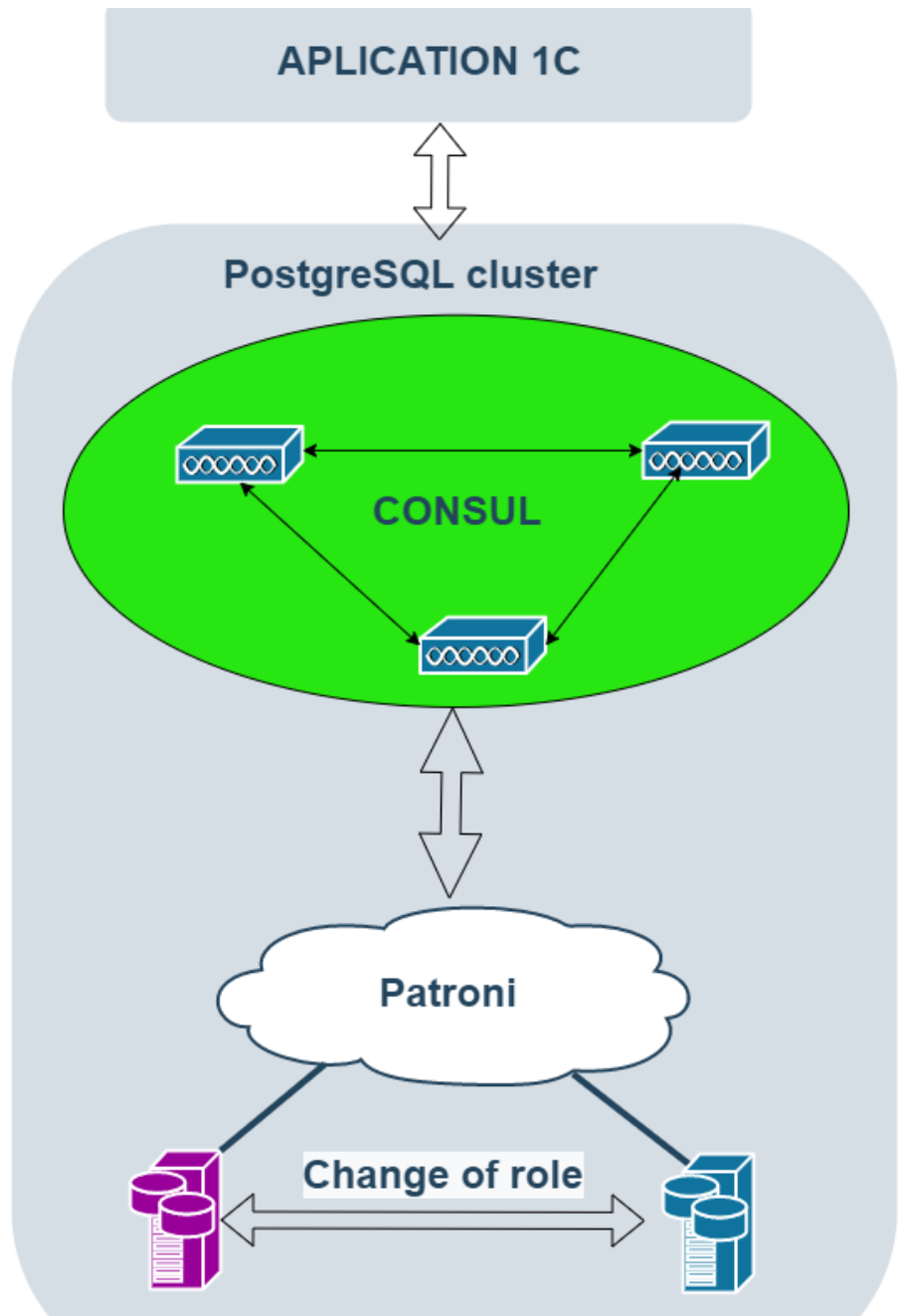
Мы являемся международной, независимой и интегрированной компанией, которая предоставляет услуги в сфере **аудита, бухгалтерского учёта, оценки, финансового, налогового, юридического и других видов консалтинга.**

Mazars является надёжным партнером в развитии бизнеса для клиентов различных отраслей и различного уровня – от крупных транснациональных компаний до предприятий малого и среднего бизнеса, частных инвесторов и государственных органов на каждом этапе их развития.

Мы работаем в 91 стране мира

О чем сегодня буду рассказывать

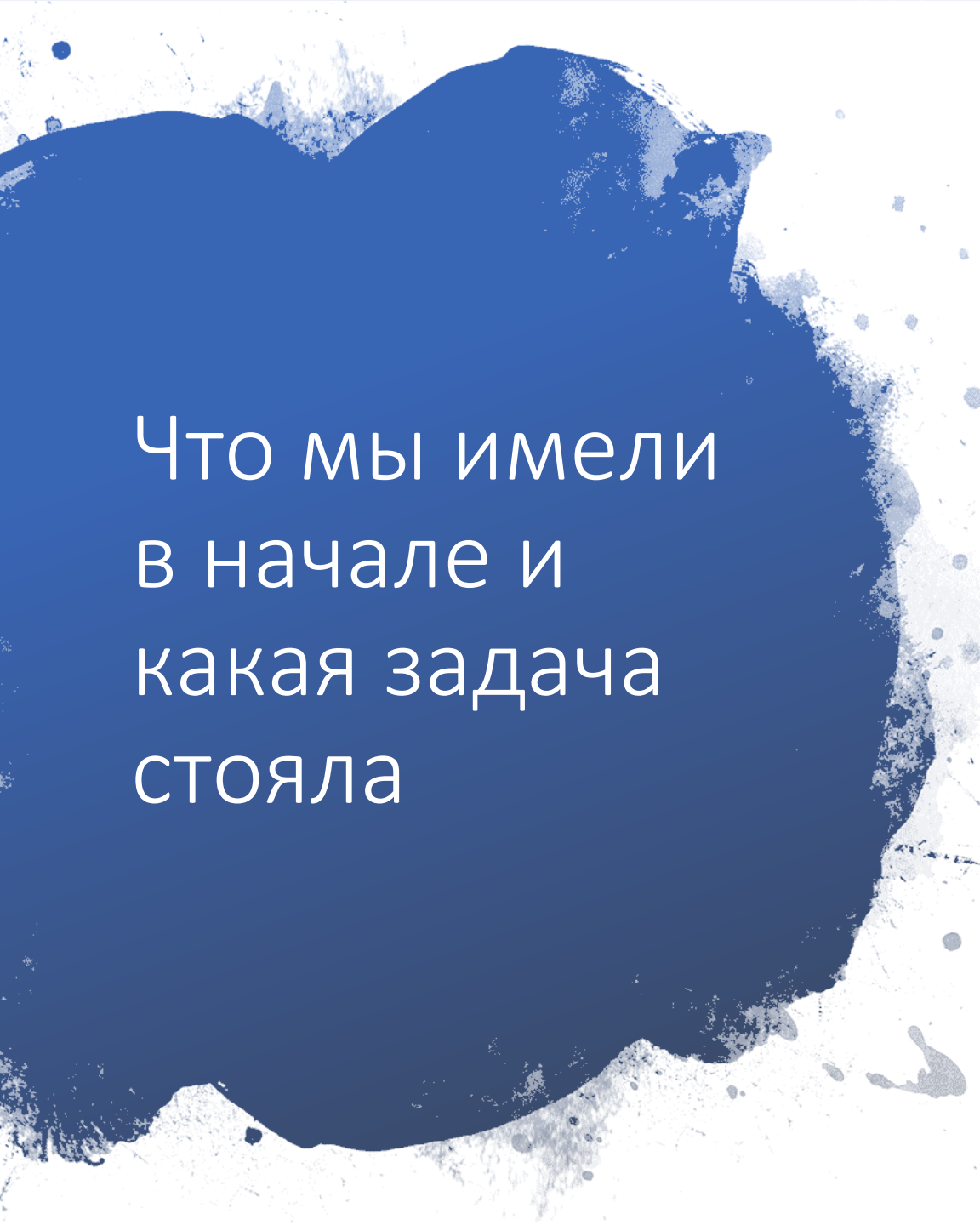
- Немного о Cluster
- Какие задачи решаем
- Patroni
- Единая точка входа в кластер
- Почему мы выбрали такое решение
- Ansible



Немного о том как устроен PostgreSQL кластер высокой доступности

- На уровне PostgreSQL мы имеем MASTER и SLAVE
- Patroni инициализирует, управляет, мониторит и хранит данные о кластере Postgres
- Consul является key-value хранилищем для Patroni
- Consul DNS является единой точкой входа в кластер

- Для PostgreSQL минимум нужно два сервера под Master и Slave
- Для Consul минимум 3, чтобы собрался кворум



Что мы имели
в начале и
какая задача
стояла

- Развернуть на основной рабочий контур Postgres
- Два хороших железных сервера
- По максимуму использовать ресурсы серверов
- Смена администраторов не приводила к последствиям. Быстрый ввод администратора в строй, без права на ошибку. Что бы смог управлять администратор 1с
- Простота в работе которая снизит человеческий фактор к минимуму.
- Минимальное количество компонентов кластера для снижения риска выхода из строя кластера

Почему именно patroni

- ПРОСТОТА, которая снижает стоимость владения кластером
- Patroni автоматически обрабатывает отказы кластера
- Прост в конфигурировании как следствие снижение человеческого фактора
- Работает стабильно
- Имеет удобный rest api
- Может управлять админ 1С, для нас это ключевое
- Не надо писать скрипты для управления кластером

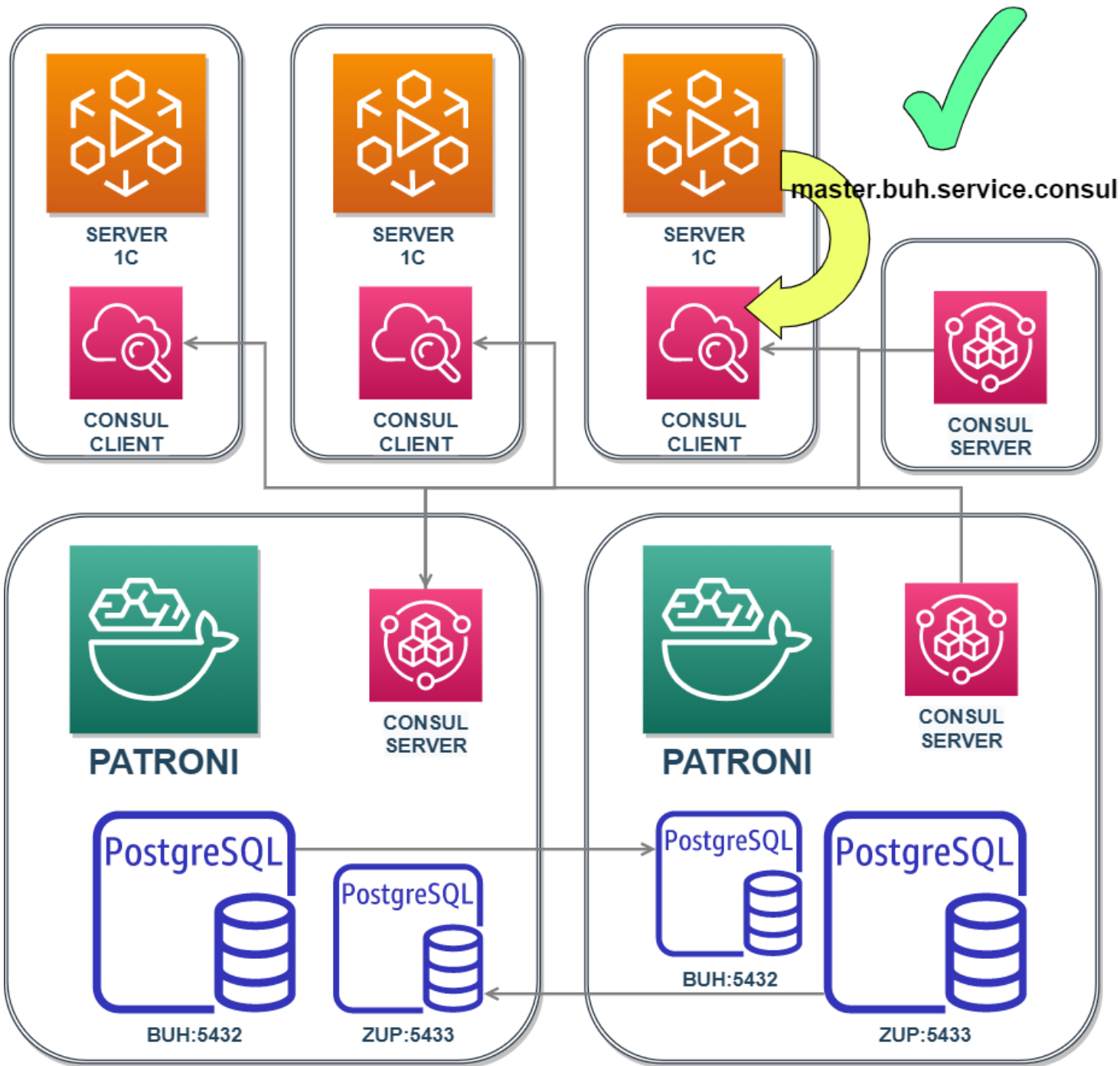


Схема работы

Использование виртуальных ip, использование vrrp, HAProxy, повышает количество компонентов кластера, а соответственно увеличивает риски отказа кластера.

Мы выбрали Consul DNS как единую точку входа в кластер.

При выходе из строя Consul, Patroni останавливает кластер

Единая точка входа Consul DNS на windows

- У consul есть несколько ролей server, client
- Как минимум должно быть три ноды с ролью server или больше, но нечетное количество
- Роль client ставится на сервер приложений 1С, количество не ограничено. Именно эта роль в нашем случае и организует единую точку входа.
- Сервер 1С обращается по адресу master.buh.service.consul и consul DNS возвращает ip мастера, при переключении на реплику consul DNS возвращает адрес нового мастера
- Consul DNS отвечает по порту 8600, а локальный DNS на 53 порту. Для устранения этого разрыва используется еще один DNS server, который пересылает пакеты между портами
- Для работы этой схемы нужно на локальном компьютере поставить DNS server 127.0.0.1 (localhost)
- Zabbix у нас проверяет resolve адреса master.buh.service.consul и в случае ошибки посылает команду на сервер ipconfig /flushdns – эта команда чистит локальный кэш DNS
- TTL = 0, время жизни записи в кэше DNS

Чем мы пользуемся в patroni

- Управление в patroni происходит через утилиту patronictl
- `patronictl -c /etc/patroni/postgresql.yml list buh`
- `buh zup` – имена кластеров
- `edit-config` – редактирование конфигурации PostgreSQL
- `pause/resume` – ручное обслуживание, например остановить кластер без failover
- `restart` – перезагрузка кластера
- `reload` – перечитать конфиги без перезагрузки
- `switchover` – переключение на реплику в ручном режиме
- `reinit` – пересоздание реплики, мастер не даст (просто умница)

```
-bash-4.2$ patronictl -c /etc/patroni/... yml list ...
```

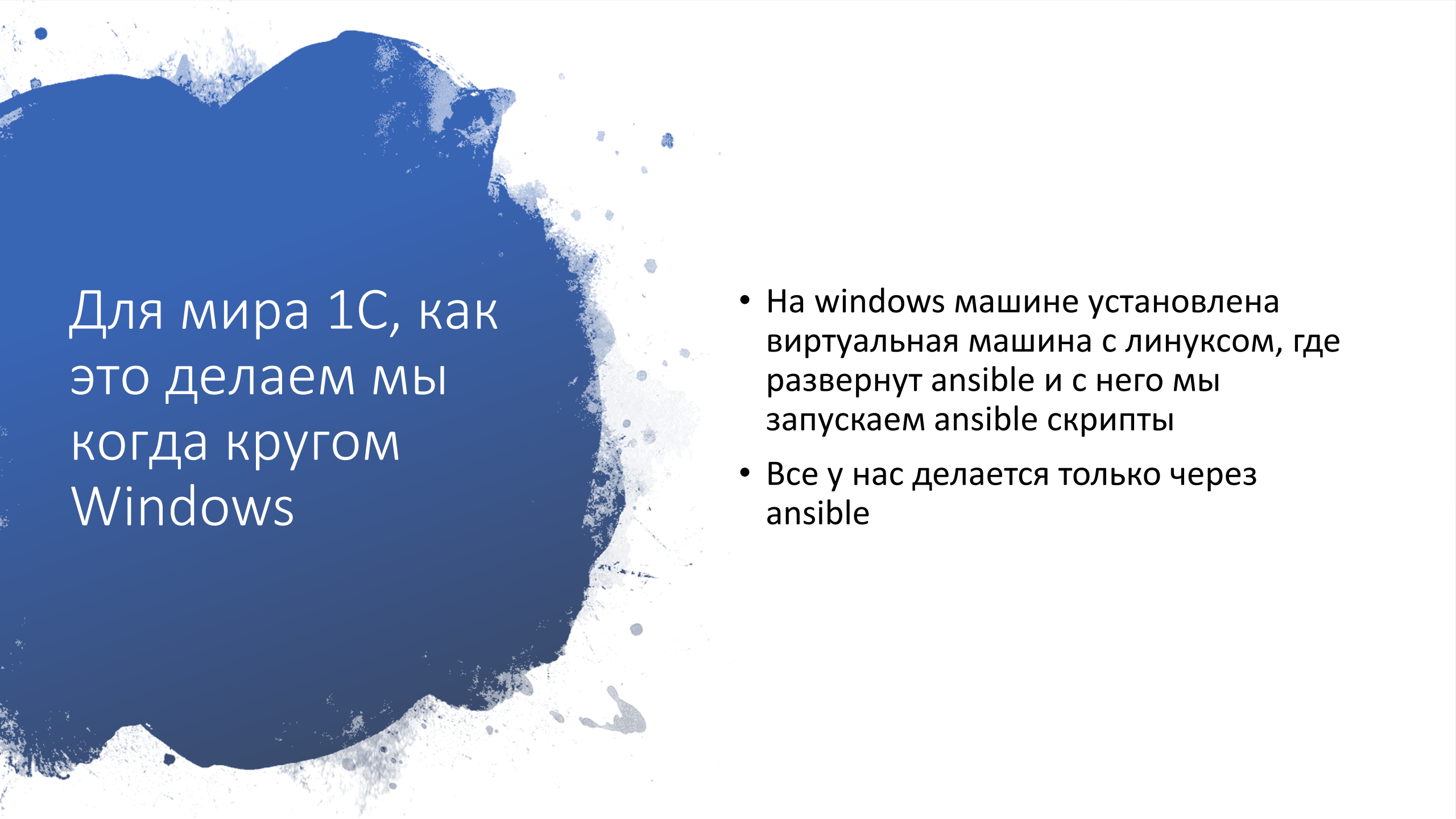
Cluster	Member	Host	Role	State	TL	Lag in MB
...	pg	Leader	running	...	0.0
...	pg		running	...	0.0

```
Maintenance mode: on
```

Ansible + Git

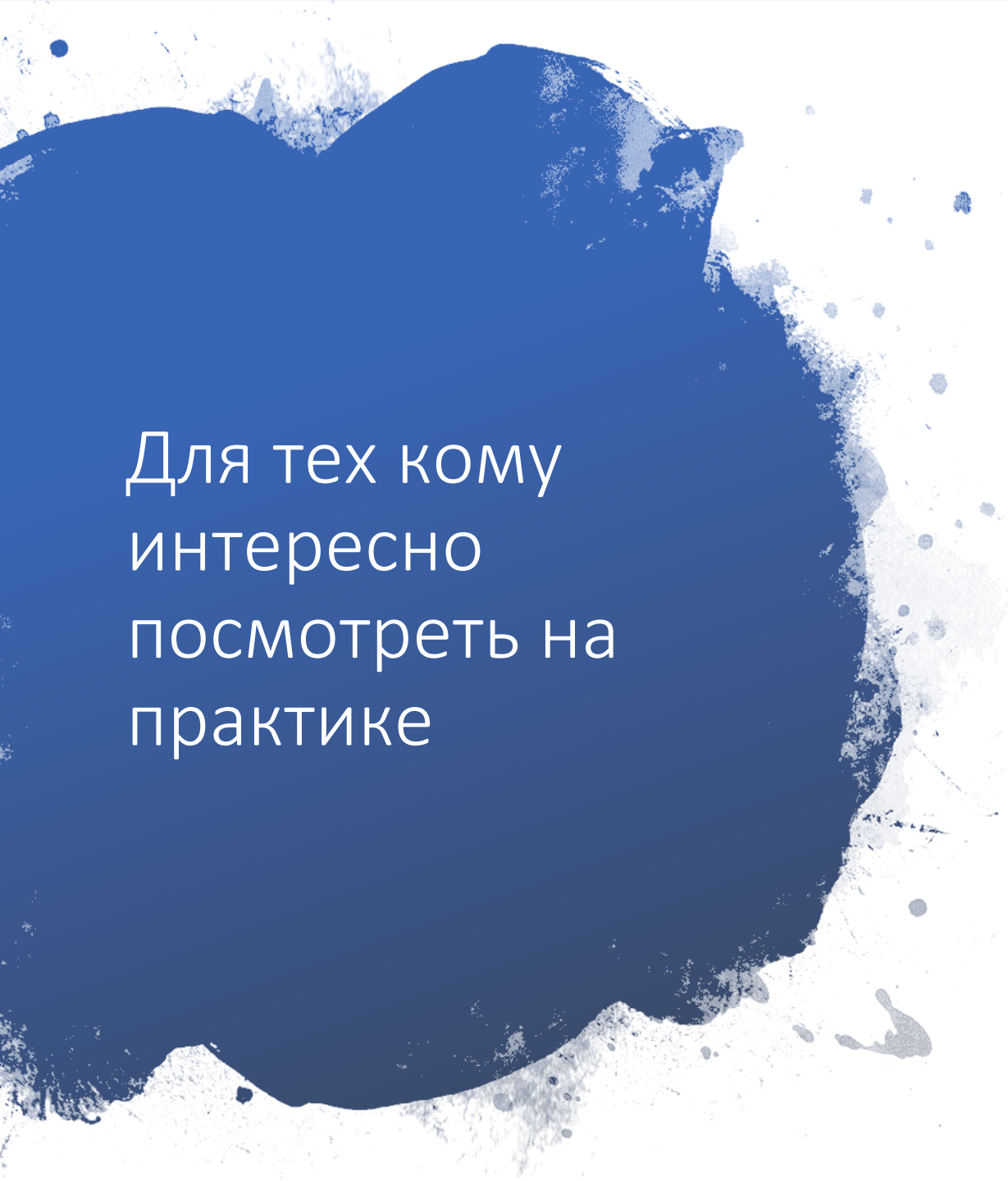
```
18
19 # tasks file for patroni
20
21 - name: Install dependansi for Patroni
22   yum:
23     name:
24       - python2-pip
25       - python-devel
26       - gcc
27       - python-virtualenv
28
29 - name: install pip upgrade
30   pip:
31     name:
32       - pip
33       - setuptools
34     state: latest
35
36 #pip install -U pip setuptools
37 - name: install psycopg2==2.7
38   pip:
39     name:
40       - psycopg2==2.7
```

- Мы используем ansible что бы в случае болезней, отпусков, увольнений администраторов, вводом в строй новых при тотальных сбоях это не приводило к простоям
- Передача настроек сервера от администратора к администратору без нюансов
- Ansible как и Patroni прост и удобен в работе
- Например нужно открыть новый порт на сервер добавляешь несколько строк пишешь для чего запускаешь playbook и готово. Все скрипты идемпотентны то есть при многократном запуске дает тот же результат
- Делаем почти все через ansible кроме подготовки железа не те объемы (установка системы, диски)
- Если вы не хотите использовать ansible то можете открыть текст роли и прочитать как это сделать, он очень прост
- Все храним в git



Для мира 1С, как
это делаем мы
когда кругом
Windows

- На windows машине установлена виртуальная машина с линуксом, где развернут ansible и с него мы запускаем ansible скрипты
- Все у нас делается только через ansible



Для тех кому
интересно
посмотреть на
практике

- Vagrant file
- Ansible script
- Скрипты обслуживания, стартовый набор
- Настройки ядра, которые мы подбирали для своих серверов

Доклад
окончен
спасибо за
внимание

Semyon Troshkin

SemyonAT@mail.ru

+7 (495) 792 52 45 /46

+7 (926) 313 99 18

<https://github.com/SemyonAT>



MAZARS