PURESTORAGE®

Service Discovery and Postgresql HA

Aliaksandr “Sasha” Aliashkevich

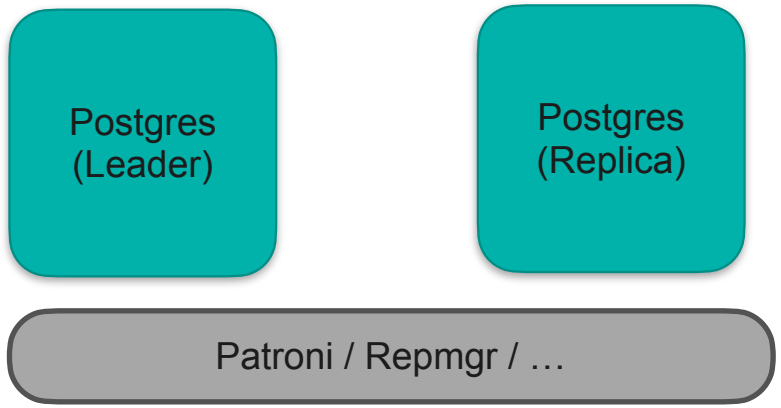
Yes

- Quick Overview of Postgresql HA
- Intro into Service Discovery
- Implementation Design

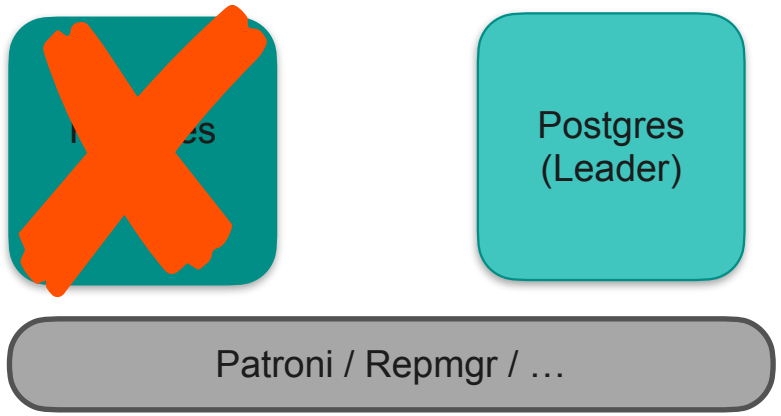
No

- Postgres
- Kubernetes
- Demo

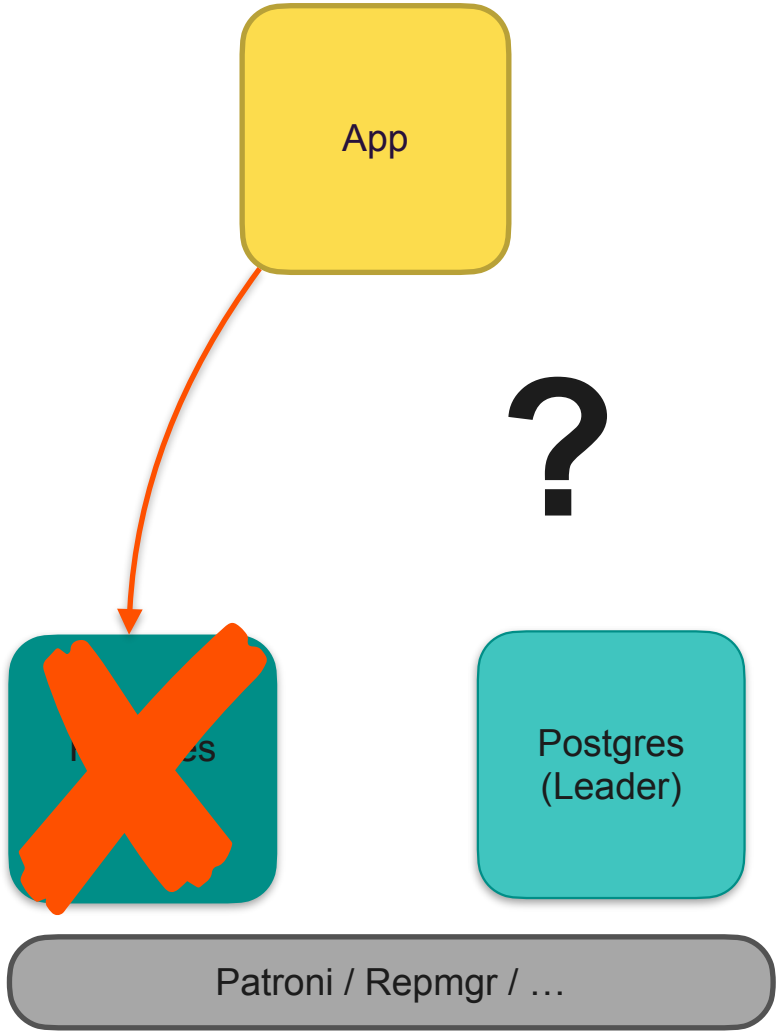
Postgresql HA / DBA View



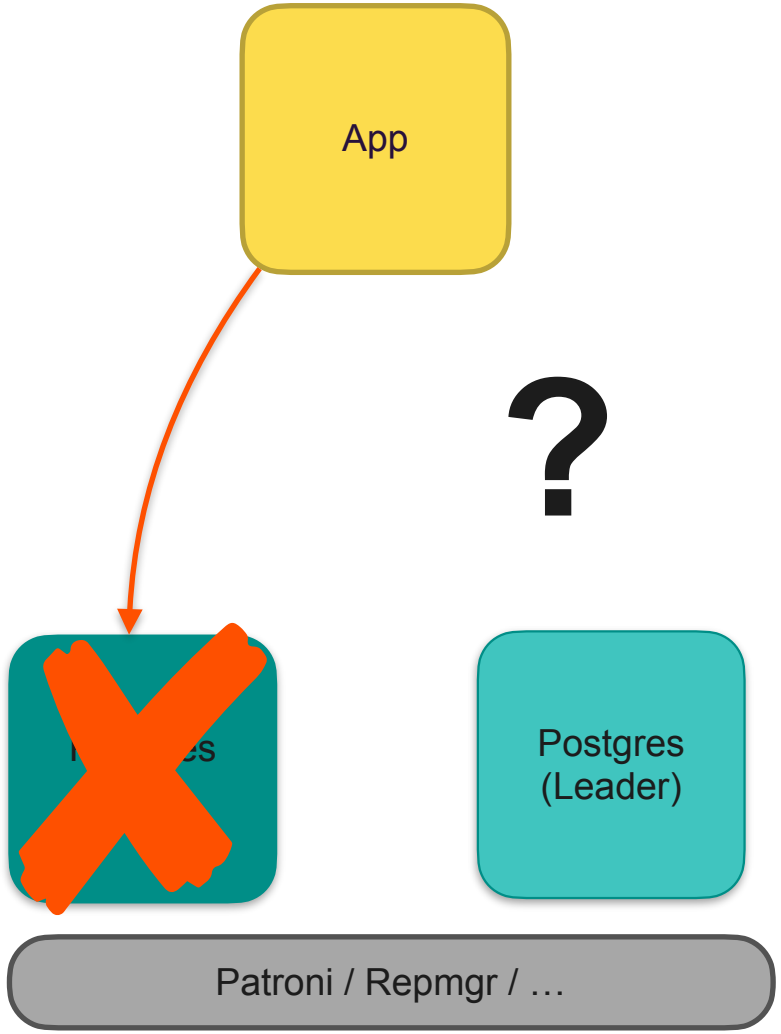
Postgresql HA / DBA View



Postgresql HA

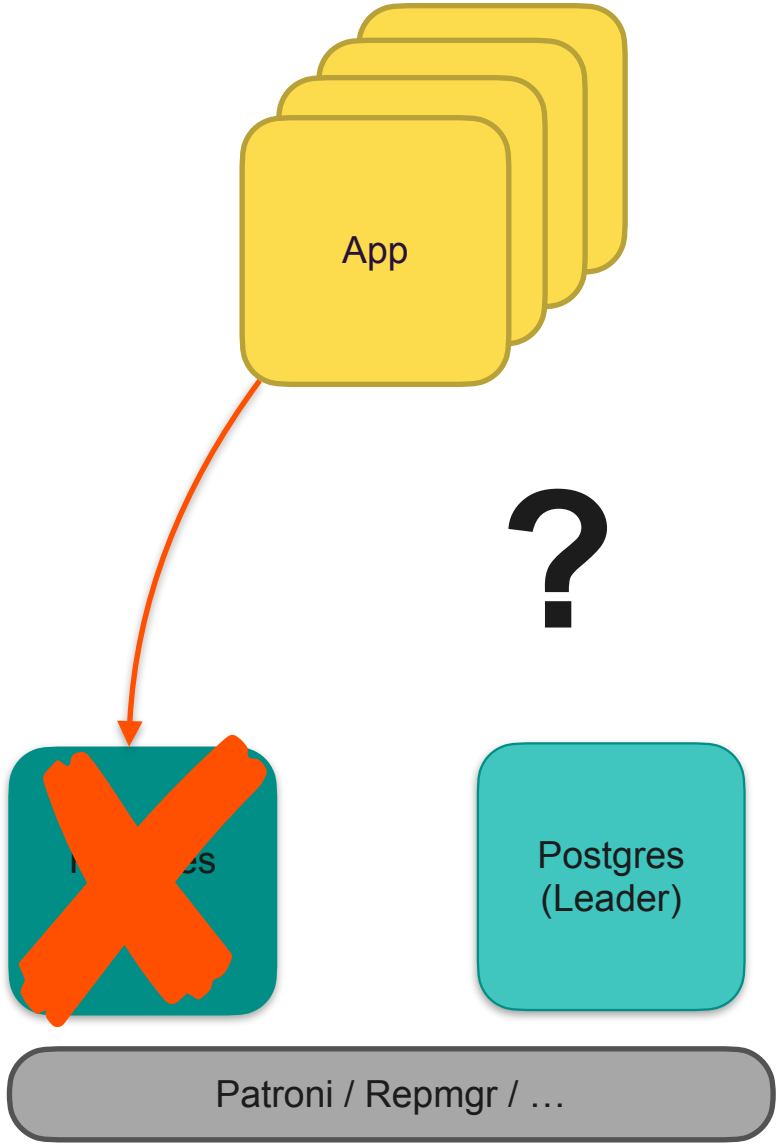


Postgresql HA



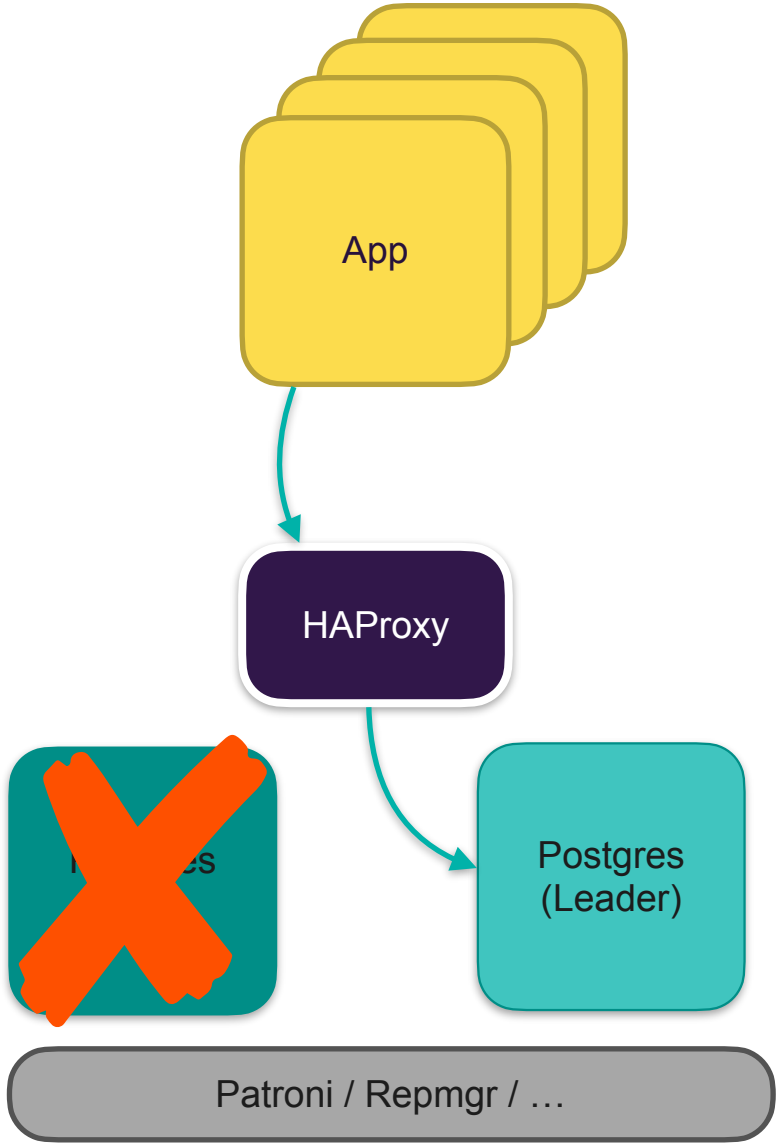
1. Reconfigure App

Postgresql HA



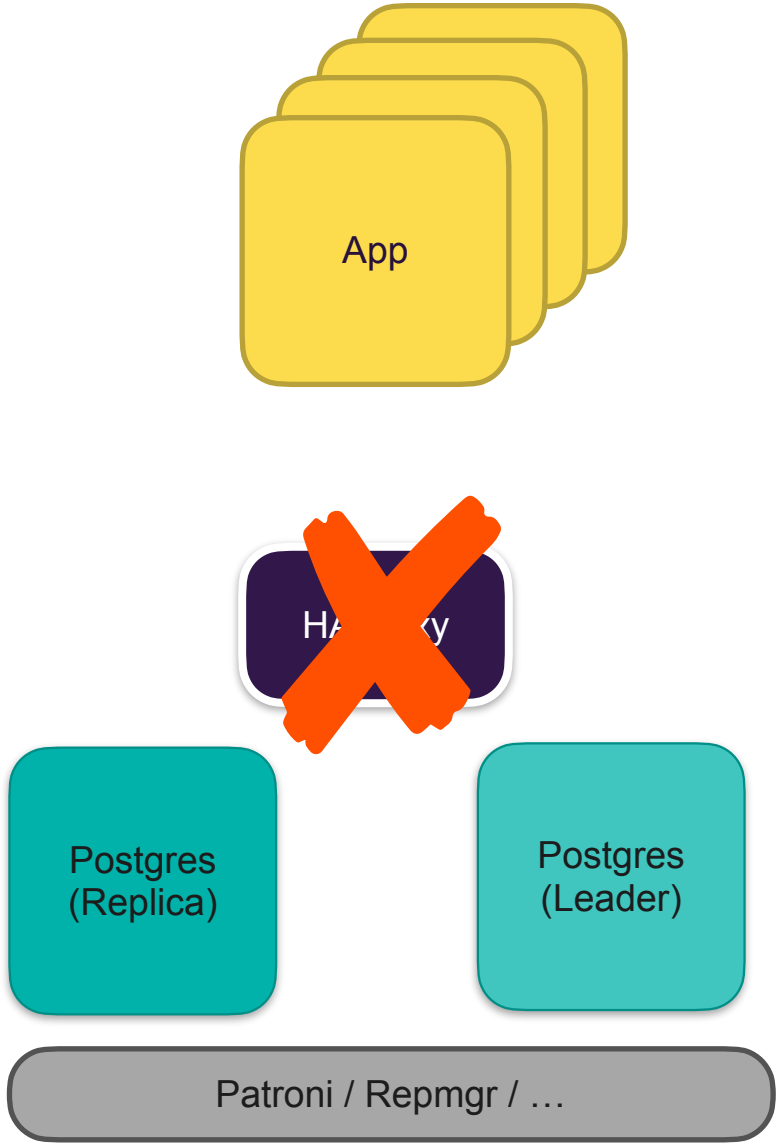
1. Reconfigure App

Postgresql HA



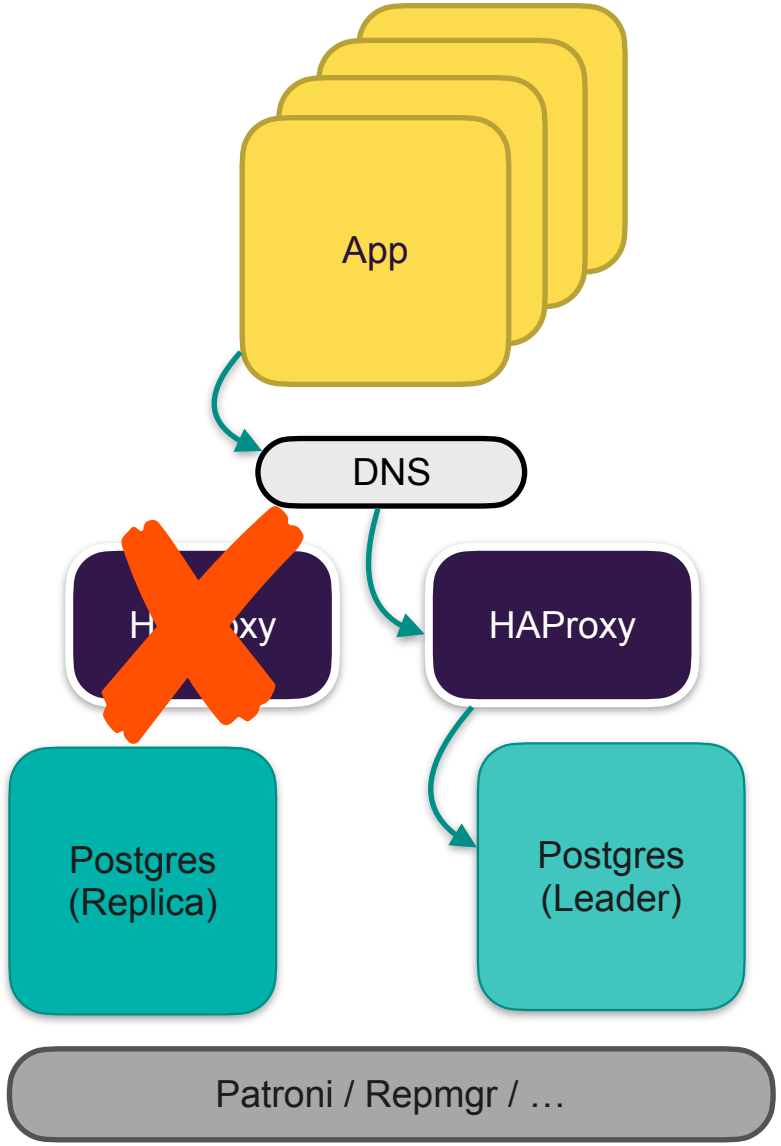
1. Reconfigure App
2. Load Balancer

Postgresql HA



1. Reconfigure App
2. Load Balancer

Postgresql HA



1. Reconfigure App
2. Load Balancer

Service Discovery

- Design pattern for the microservices architecture
- Mechanism for finding and connecting to the services available on a network
- Allows services to find each other and communicate dynamically, without hard-coding IP addresses or URLs
- Enables services to fail over to another instance if one instance goes down, improving reliability and availability of the overall system
- Kubernetes built-in

Service Discovery / Service Registry

- Central database for service discovery
- Keeps track of available services and their status
- Services register themselves in the registry on startup
- Clients query the registry to find services they need
- Examples: Etcd, Consul, Zookeeper

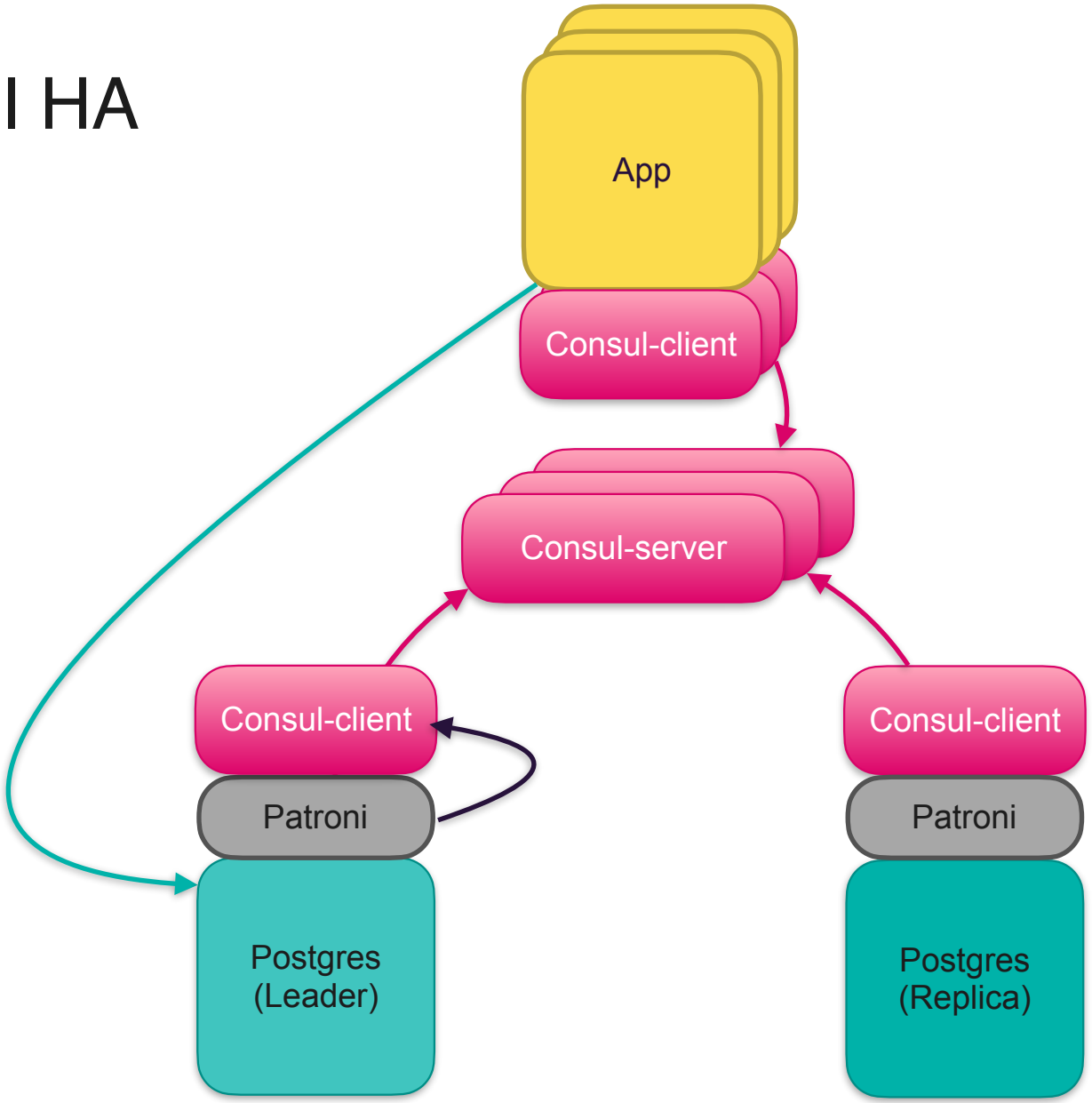
Service Discovery / Sidecar Pattern

- Another design pattern for the microservices architecture
- A separate process running alongside the main application to provide some extra functionality
- Enables adding features and functionality without changing the main application code
- Examples: PGBouncer, Patroni, Prometheus Exporter

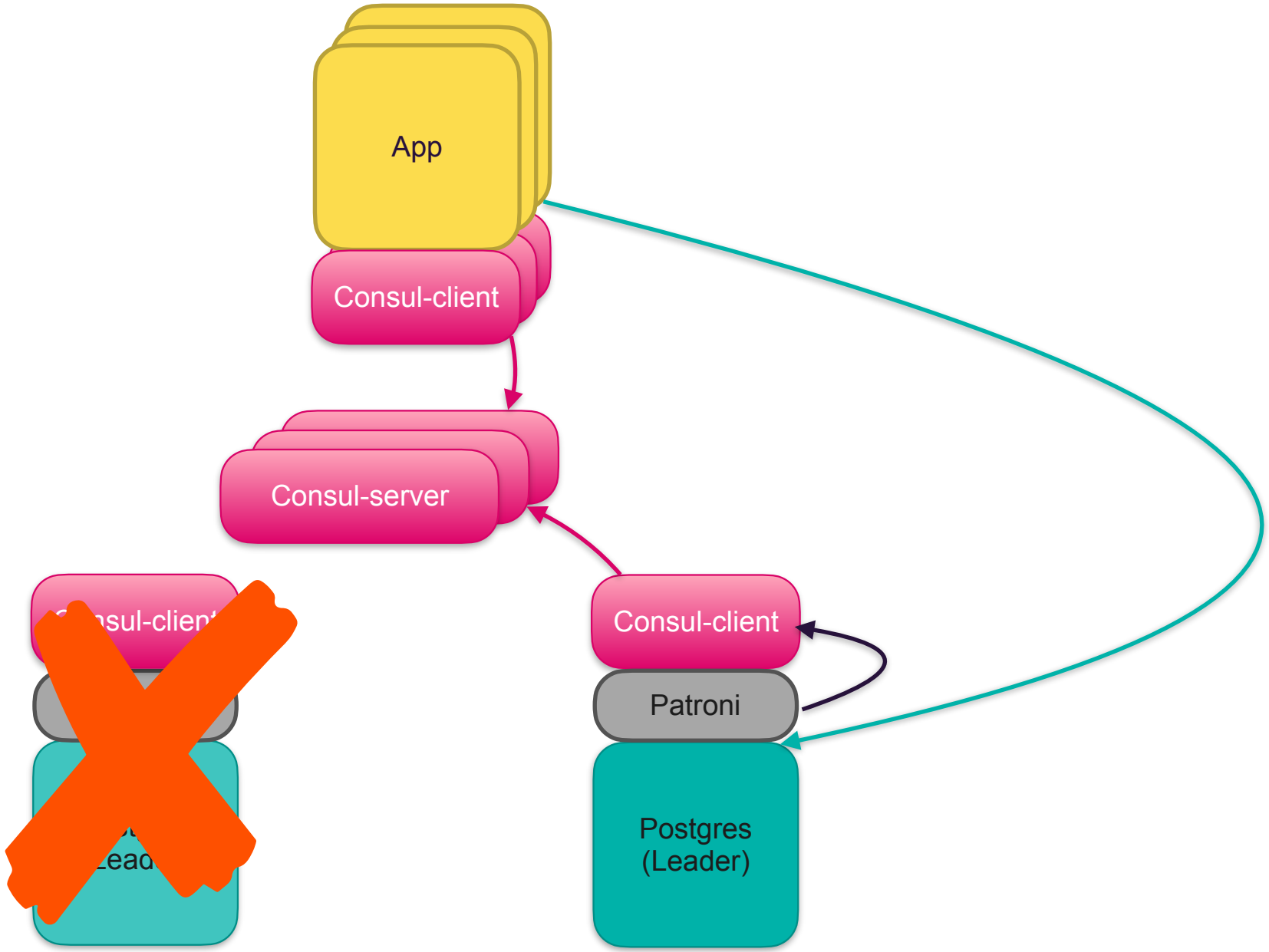
Service Discovery / Consul

- Service registry and service discovery framework
- Developed by HashiCorp and written in Go
- Consul-servers are responsible for voting and storing the data (3-5 nodes is optimal)
- Consul-clients run as a sidecar alongside the main process and responsible for communication with consul-servers
- Applications communicate only with local consul-clients
- Consul agents have a DNS interface

Postgresql HA



Postgresql HA



Postgresql HA / Consul server config

```
{
    "node_name": "consul01",
    "server": true,
    "ui_config": {
        "enabled" : true
    },
    "data_dir": "/consul/data",
    "addresses": {
        "http" : "0.0.0.0"
    },
    "retry_join":[
        "consul01",
        "consul02",
        "consul03"
    ]
}
```

Postgresql HA / Consul client config

```
{
    "node_name": "${HOSTNAME}",
    "data_dir": "/consul/data",
    "retry_join":[
        "consul01",
        "consul02",
        "consul03"
     ],
    "recursors": ["127.0.0.11"],
    "addresses": {
        "dns": "127.0.0.1",
        "http": "127.0.0.1"
    }
}
```

This flag provides addresses of upstream DNS servers used for recursive resolving queries if they are not inside the Consul service domain

Postgresql HA / Patroni config

```
scope: pglab
consul:
  url: http://127.0.0.1:8500
  register_service: true
```

Patroni doesn't register services in Consul by default, but turning it on creates a service with the <scope> name and two tags: "master" and "replica".

Access the master node using "**master.pglab.service.consul**"

Postgresql HA / DNS settings

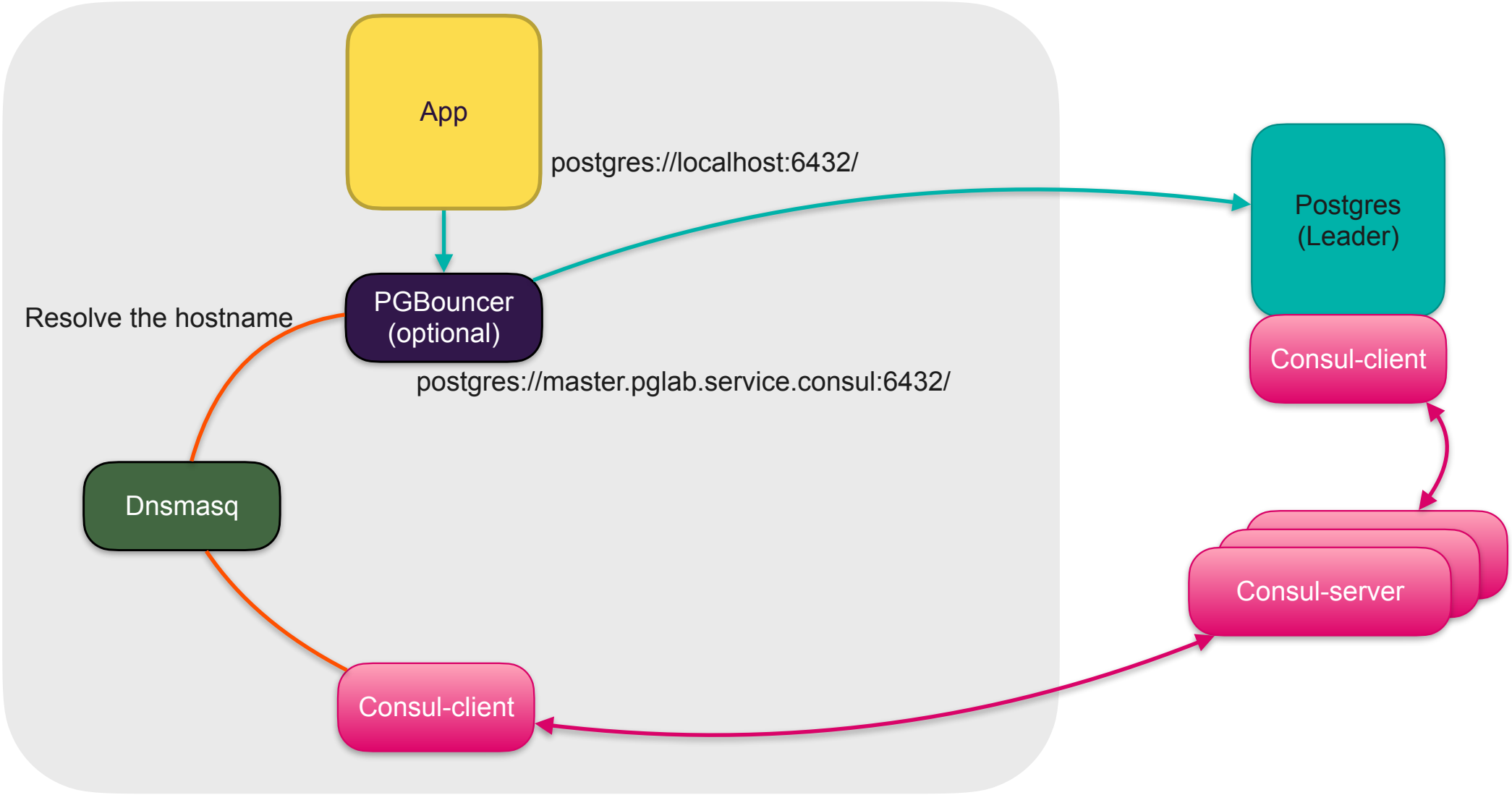
resolv.conf

```
nameserver 127.0.0.1
nameserver 127.0.0.11
options ndots:0
```

dnsmasq.conf

```
server=/consul/127.0.0.1#8600
```

Postgresql HA / PGBouncer



Postgresql HA / PGBouncer

```
[databases]
postgres = host=master.pglab.service.consul port=5432 dbname=postgres

[pgbouncer]
listen_port = 6432
listen_addr = localhost
auth_type = md5
auth_file = /etc/pgbouncer/userlist.txt
logfile = /var/log/pgbouncer/pgbouncer.log
pidfile = /home/pgbouncer/pgbouncer.pid
admin_users = app_user
pool_mode = session
dns_max_ttl = 0
server_login_retry = 0
```

Don’t cache DNS records internally

Retry to login immediately after failing connection attempt

Thank you

Question?

https://github.com/sasha-alias/postgresql-consul-demo