PostgreSQL and Kubernetes

Database as a Service without a Vendor Lock-in

Oleksii Kliukin

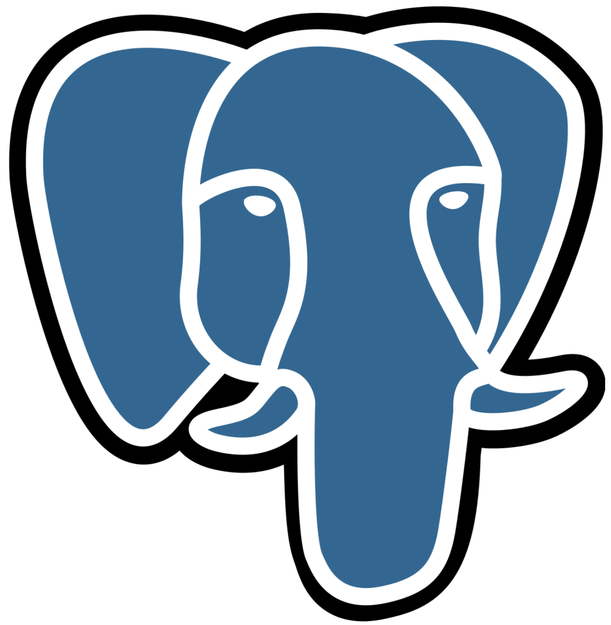
PostgreSQL Sessions 10
Paris, France

About me

- PostgreSQL Engineer @ Adjust
- PostgreSQL Contributor
- Organizer of PostgreSQL Meetup Group in Berlin
- Worked on Patroni, Postgres Operator, Spilo and other Zalando projects.

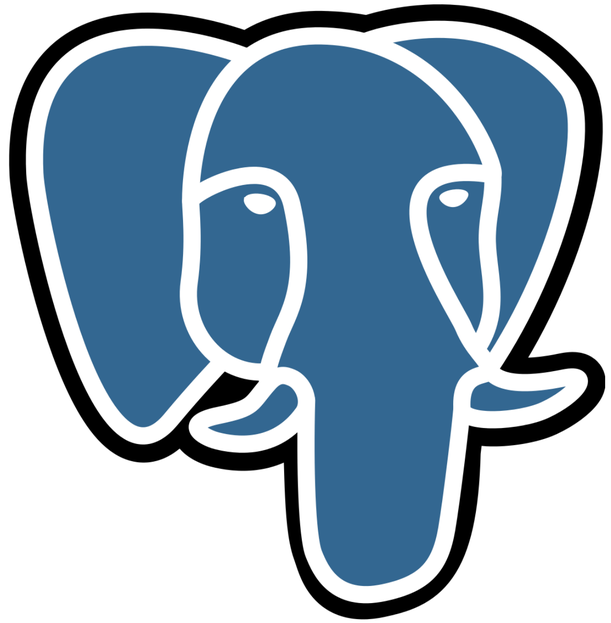
PostgreSQL advantages

- Designed for reliability
- SQL Standard Conformance
- Actively developed by the community
- Scalable (physical/logical replication, sharding)
- Performant
- Extensible (custom types, indexes, wal records, background workers, planner/executor hooks)



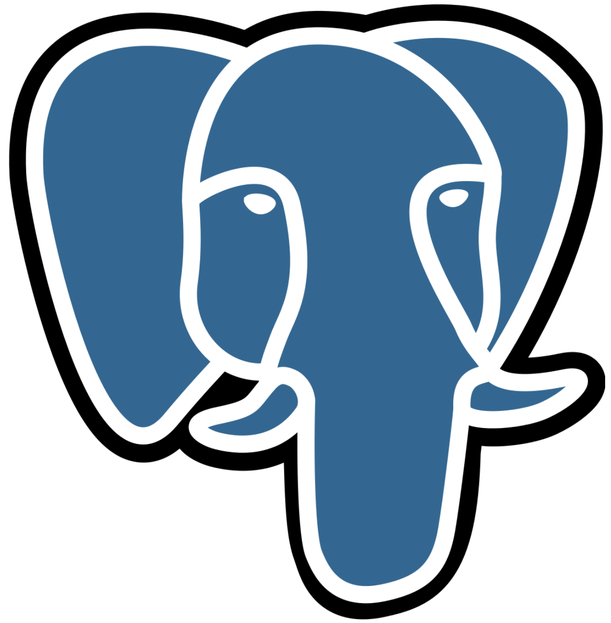
PostgreSQL is open-source

- Source code is available in git
- Learn how your database works
- Implement new features (or pay someone to do it)
- Fix bugs and test fixes without waiting for new release
- No license costs, no price per core or per server



PostgreSQL is open-source

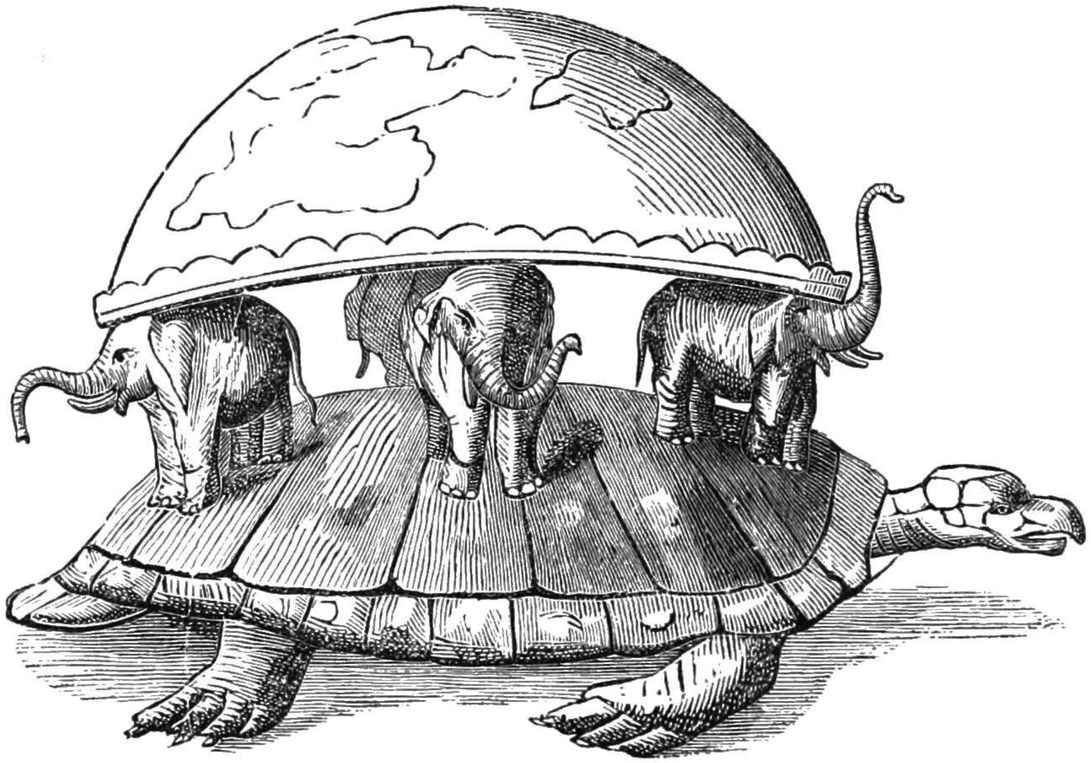
- Source code is available in git
- Learn how your database works
- Implement new features (or pay someone to do it)
- Fix bugs and test fixes without waiting for new release
- **No license costs, no price per core or per server**



From 1 to 1001 PostgreSQL clusters

Multiple PostgreSQL clusters

- Smaller databases
- Simpler maintenance
- Simpler security model
- One database per application
- Hundreds of smaller databases with microservices



Managing multiple PostgreSQLs

- Manual way: DBAs do everything by themselves (using shell scripts, ssh, …)
- Semi-automated way. DBAs run Ansible/Rex/Puppet/… scenarios to converge the cluster/clusters to the desired state
- Automated way: End-users create new clusters directly using Database as a Service (DBaaS)

Database as a Service

- End-user initiated:
  - Create cluster
  - Update database configuration
  - Add resources to the cluster (replicas, disk, CPU, memory)
  - Delete cluster

Database as a Service

- Automatically handled:
  - Management of resources
  - Export data to monitoring
  - Service discovery
  - Disaster recovery

How to get DBaaS

- Pay someone (Google, AWS, Amazon)
  - Vendor-lock
  - Not always community PostgreSQL (i.e. Amazon RDS or Aurora)
  - You may not have all features (i.e. no superuser, logical replication, …)
- Build it yourself
  - Expensive and requires a lot of expertise outside of the database world
  - Duplication of efforts between different companies
  - Tied to your existing infrastructure
- Embrace the open-source

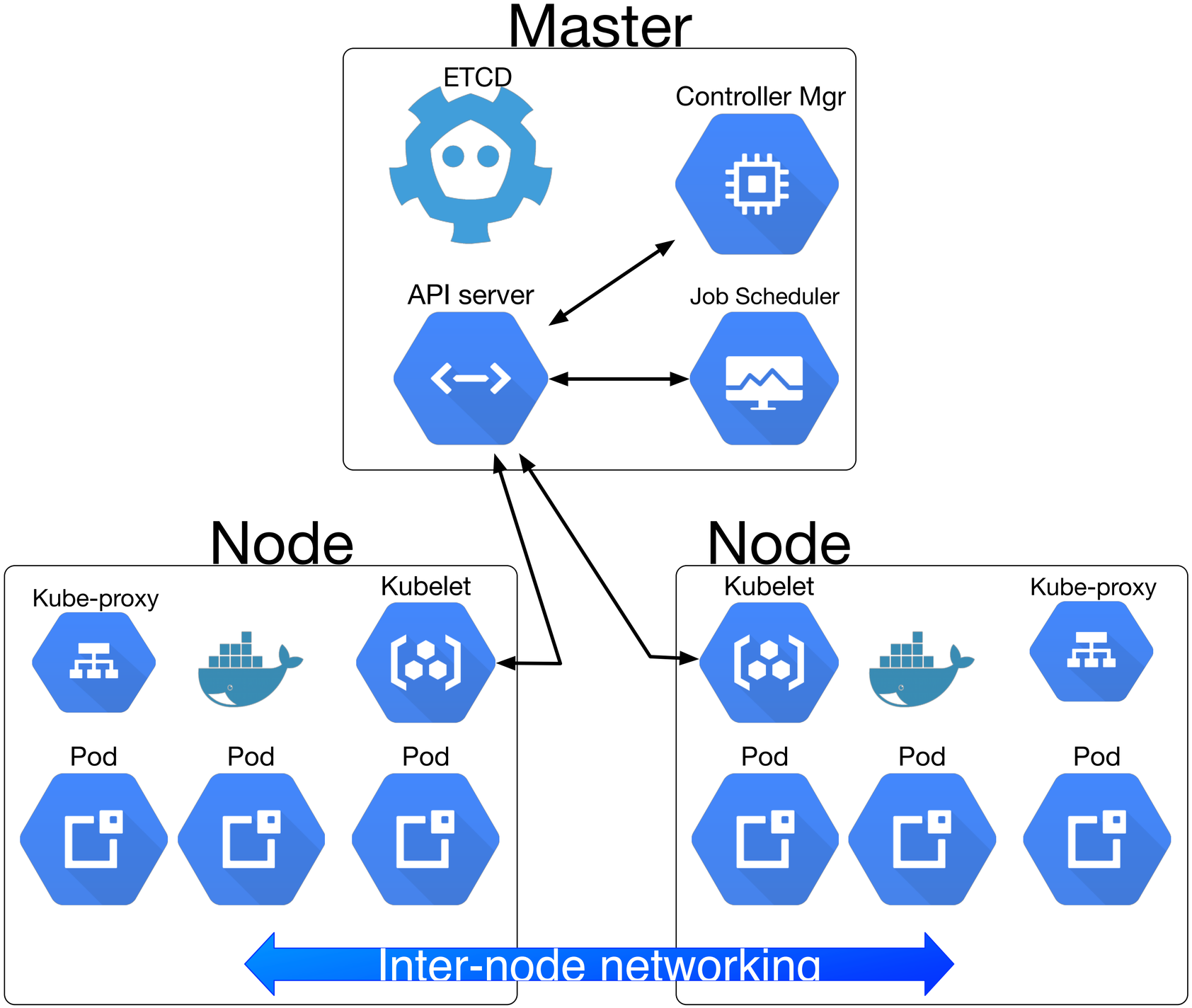
PostgreSQL DBaaS on Kubernetes

What is Kubernetes

- Set of open-source services
- Running on one or more servers
- Physical or cloud based (AWS, GCE, Azure, Digital Ocean etc)
- Automating deployment
- Scaling and management
- Container-based applications

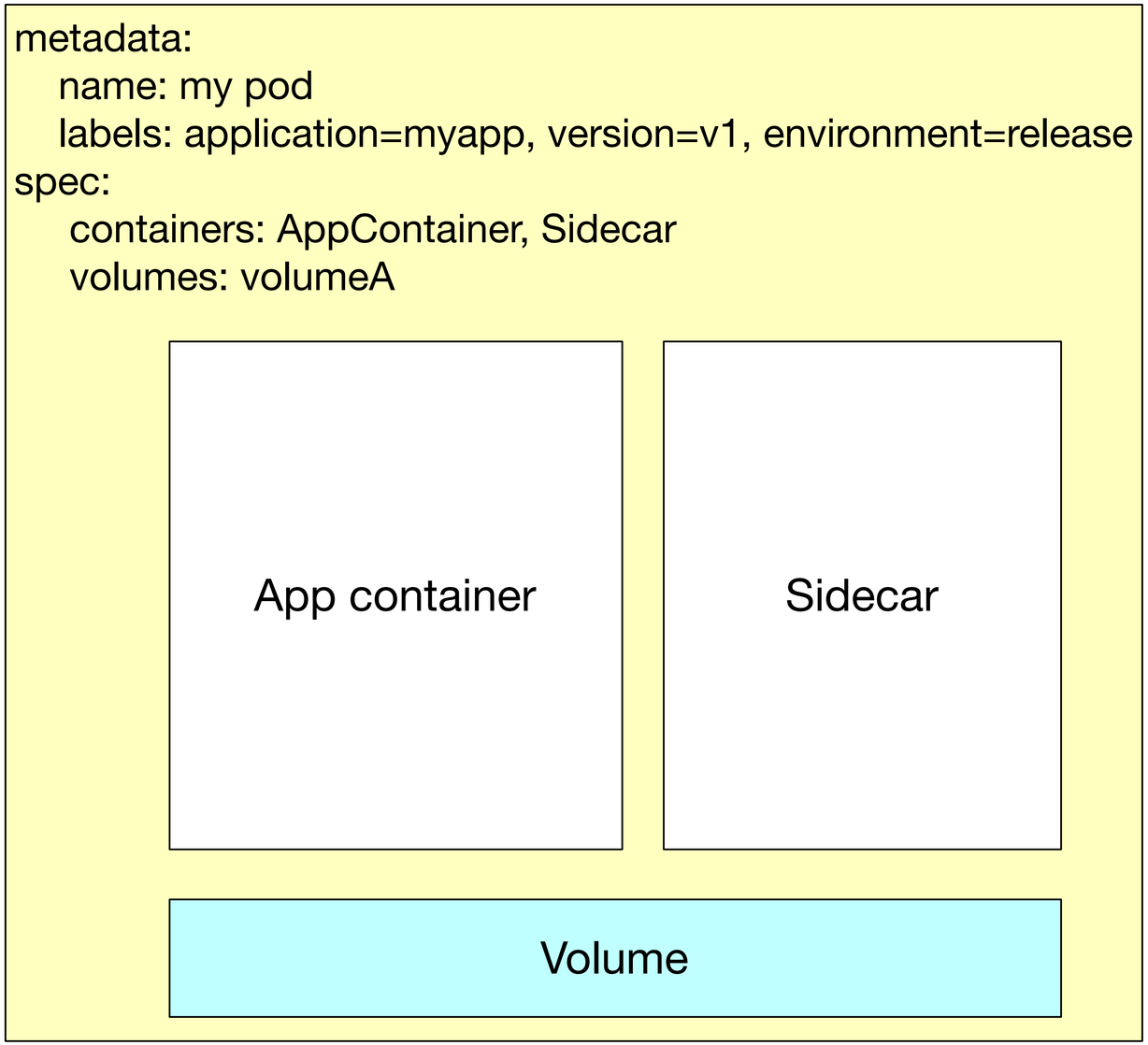
Kubernetes provides

- Unified API abstraction for multiple different infrastructure providers (i.e. AWS, GCP, Azure)
- Declarative based deployments of resources and applications
- Repeatable deployments with containers
- Extensible services to define and manage user-specified resources


Master
ETCD
Controller Mgr
API server
Job Scheduler
Node
Kube-proxy
Kubelet
Pod
Pod
Pod
Node
Kubelet
Kube-proxy
Pod
Pod
Pod
Inter-node networking

Building blocks: Pods

- Group one or more related containers
- On the same host
- Share host resources (i.e network)
- Usually one instance of the app
- Scheduled to run on nodes based on memory, cpu requirements

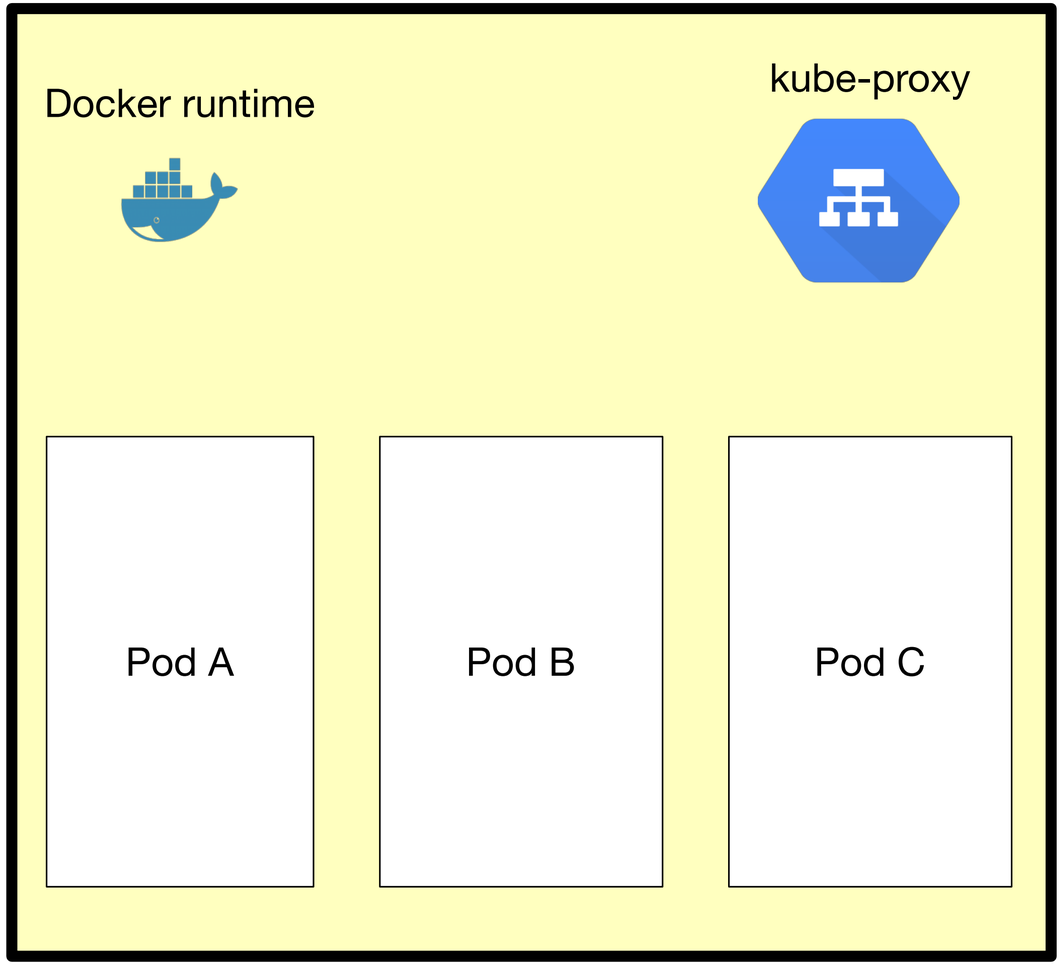


Building blocks: Metadata

- Labels (i.e. app=postgres, name = shop, role=master, environment=production)
- Selectors to choose objects based on labels
- Annotations to attach arbitrary key-value metadata (i.e image_version=p42)
- Attached to most objects (nodes, pods, persistent volumes, services, endpoints, etc)

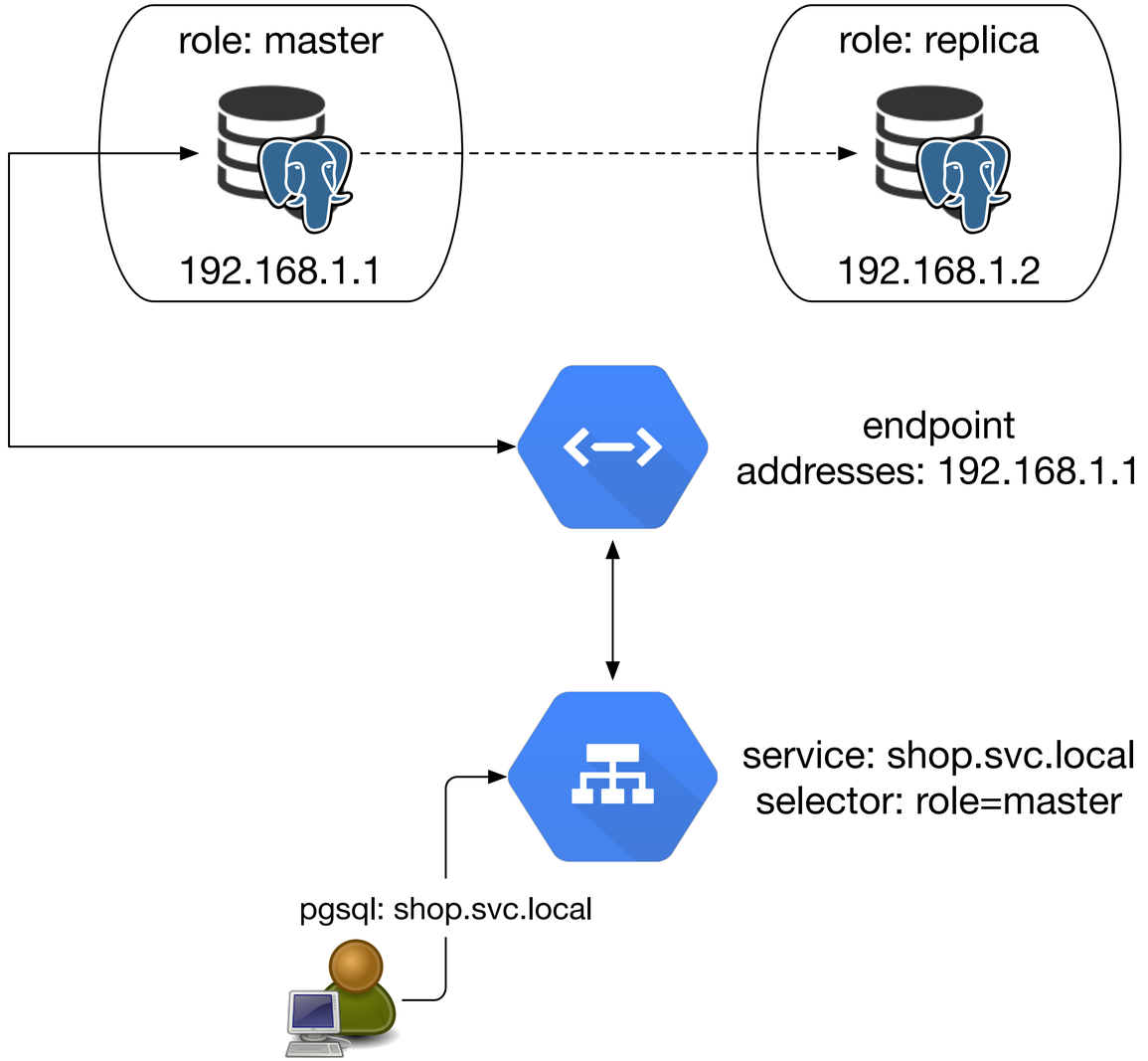
Building blocks: Nodes

- A physical or virtual server (i.e. EC2 or GCE instance)
- Running as many pods as it provides resources by Kubelet
- Container runtime (i.e. docker)
- kube-proxy to route requests to pods



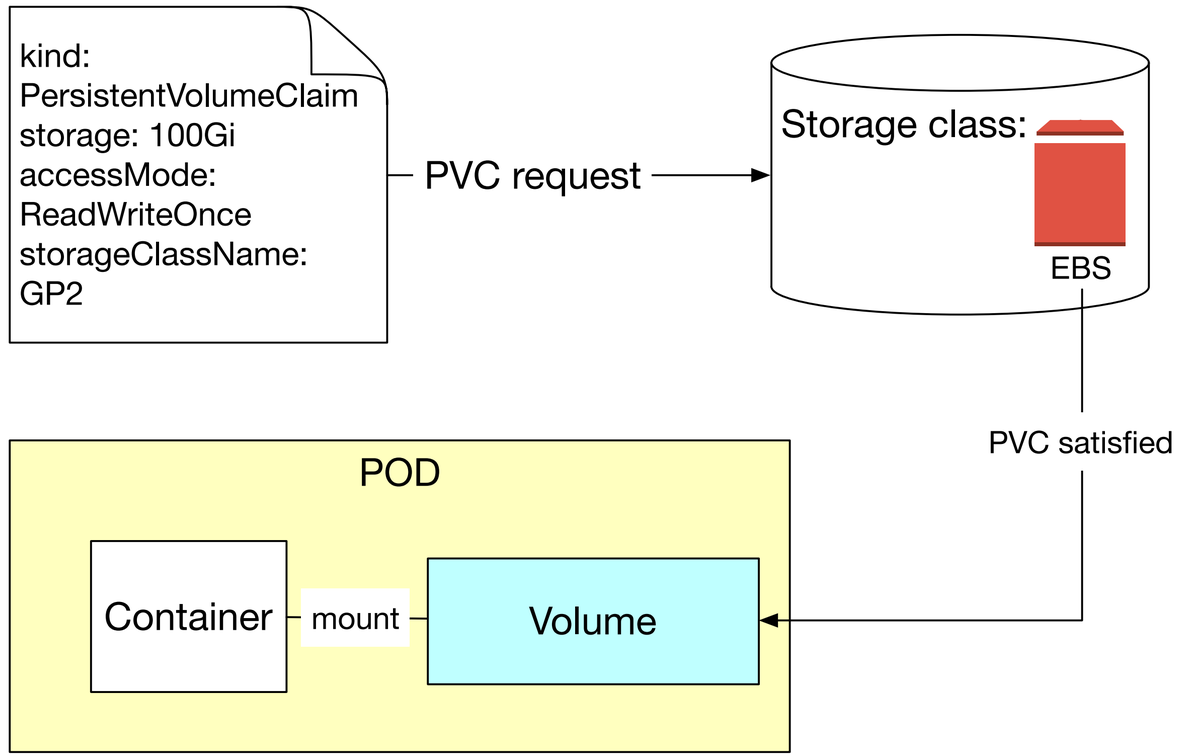
Building blocks: Services and Endpoints

- Define how do clients connect to pods
- Endpoints contain actual addresses
- Services can create endpoints
- Services may pick pods to connect using selectors



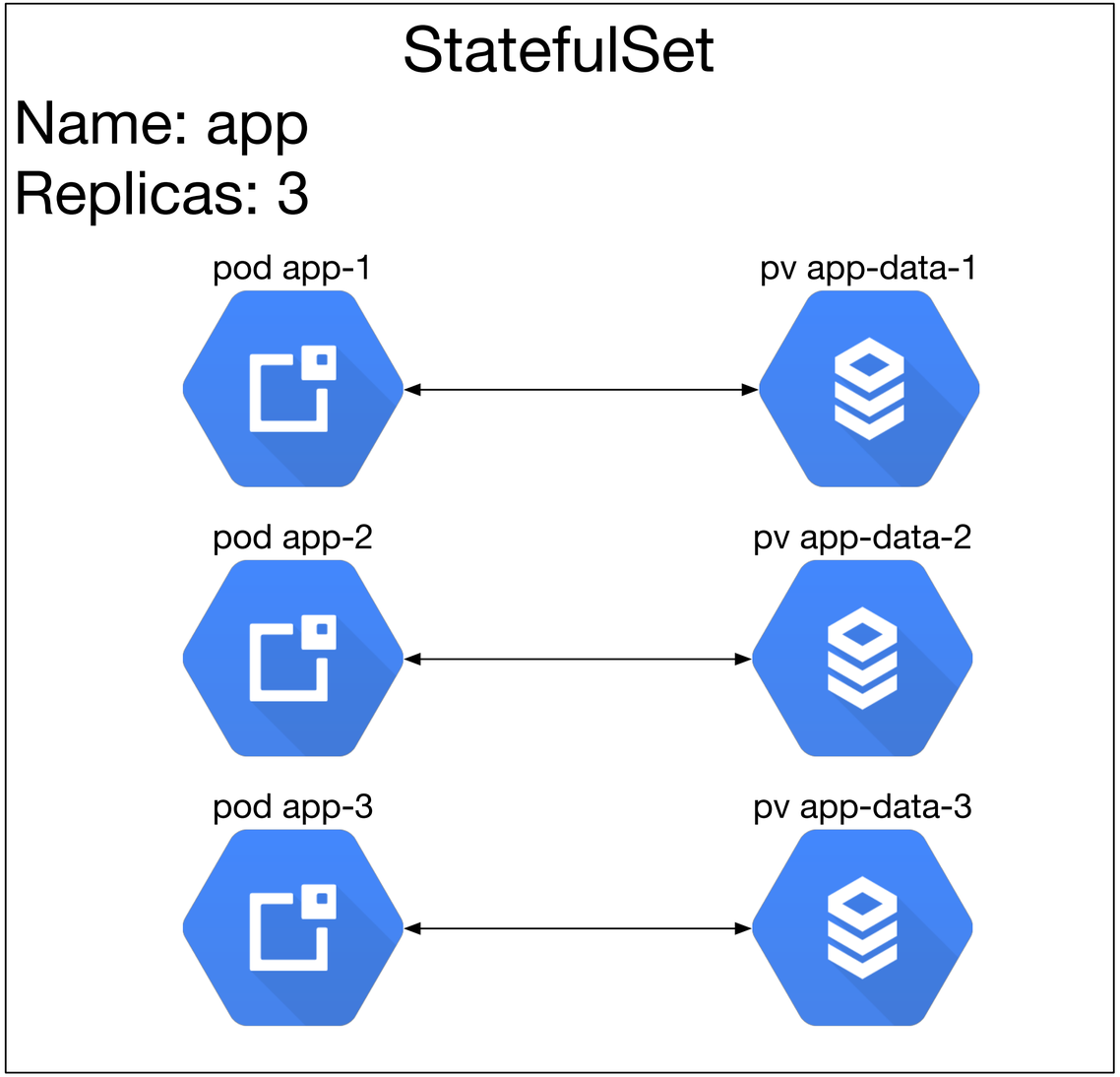
Building blocks: Persistent Volumes

- A storage volume that persists between pod terminations
- Examples: EBS, GCE PD, NFS
- Managed by Persistent Volume Claims (PVC)
- PVC may request storage, size and access mode
- Storage is controlled with StorageClasses



Building blocks: StatefulSets

- Controller that binds pods and persistent volumes together
- Each pod gets attached a persistent volume
- On restart, the same volume and IP address is attached to a pod
- Statefulset manages the defined amount of pods (killing excessive, starting missing)



Building blocks: CRD

- Custom user-user-defined controllers
- Read YAML manifests submitted by users with custom-custom-defined schema (custom-resource definition instance)
- Create and maintain Kubernetes objects based on the CRD instance manifest

```yaml
apiVersion: "acid.zalan.do/v1"
kind: postgresql
metadata:
  name: acid-minimal-cluster
  namespace: test
spec:
  teamId: "ACID"
  volume:
    size: 1Gi
  numberOfInstances: 2
  users:
    zalando:
    - superuser
    - createdb
    foo_user:
 databases:
    foo: zalando
  postgresql:
    version: "10"
```

Building blocks: ConfigMaps

- Key-value storage of text string
- Useful for storing configuration

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: postgres-operator
data:
  watched_namespace: "*"
  cluster_labels: application:spilo
  cluster_name_label: version
  pod_role_label: spilo-role

  workers: "4"
  docker_image: spilo-cdp-10:1.5-p35
  super_username: postgres
  aws_region: eu-central-1
  db_hosted_zone: db.example.com
  pdb_name_format: "postgres-{cluster}-pdb"
  api_port: "8080"
  ...
```

Building blocks: Secrets

- Key-value storage of text string
- Values are base64 encoded
- Usually restrictive access
- Useful for storing logins-passwords

```yaml
apiVersion: v1
data:
  # user batman with the password justice
  batman: anVzdGljZQ==
kind: Secret
metadata:
  name: postgresql-infrastructure-roles
  namespace: default
type: Opaque
```

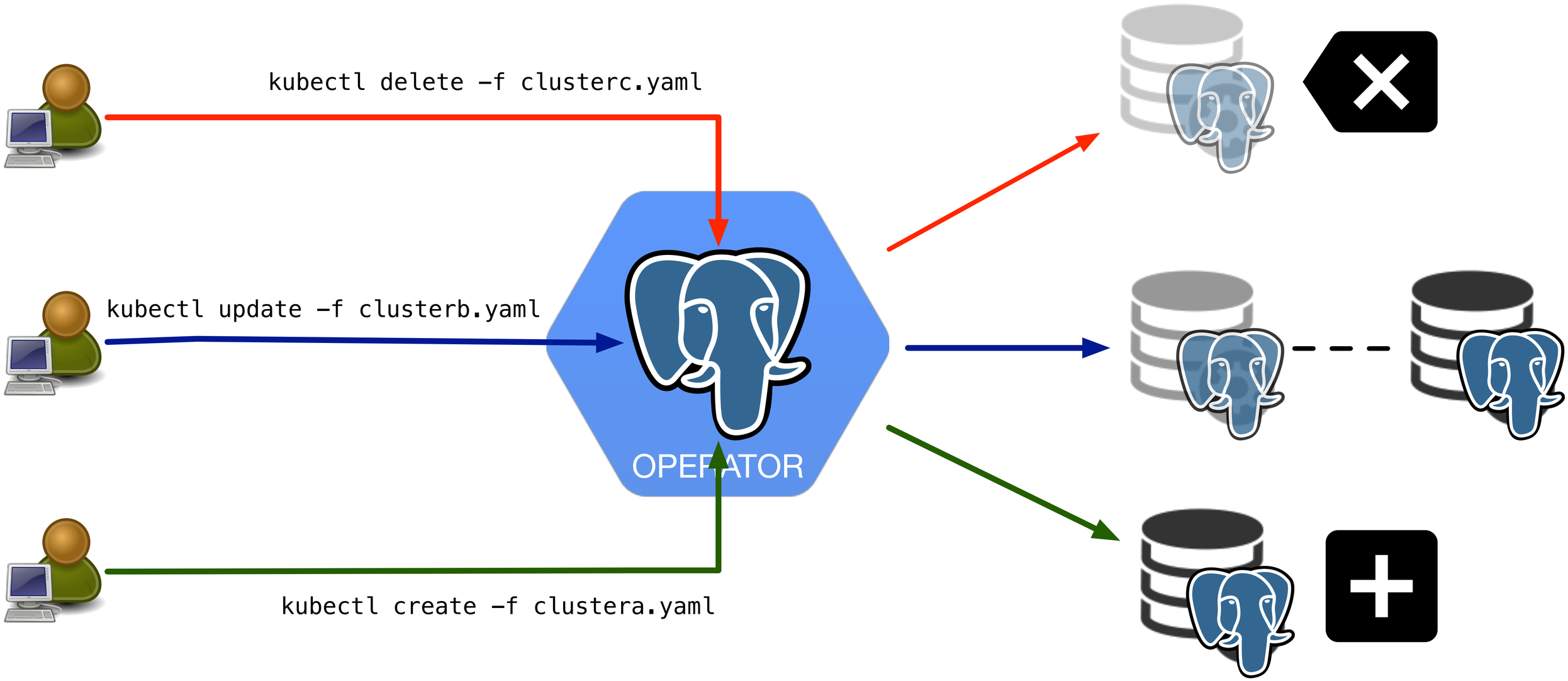
Operator pattern

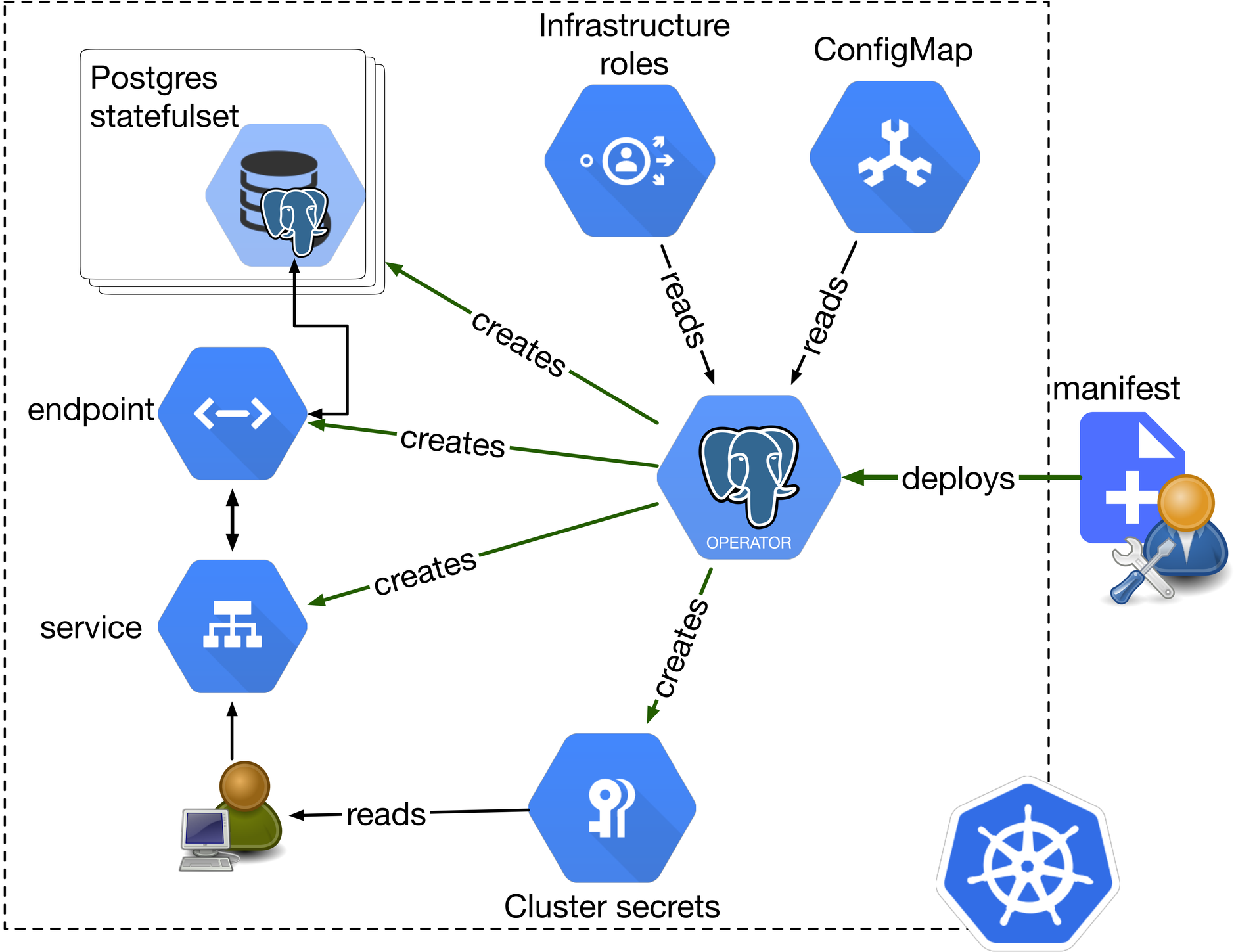
- Custom controller to process user-supplied resources
  - Register CRDs
  - Perform CRUD operations via the API
  - Encapsulate custom knowledge about the domain (i.e. databases)

Zalando Postgres Operator

- Implements the custom controller to manage Postgres HA clusters
- Watches CRD objects of type postgresql
- Creates and deletes clusters
- Updates Kubernetes resources and Postgres configuration
- Periodically validates running Kubernetes objects against manifest definitions

Zalando Postgres Operator actions




Infrastructure roles
ConfigMap
Postgres statefulset
reads
reads
creates
endpoint
creates
manifest
deploys
OPERATOR
creates
service
creates
reads
Cluster secrets

Postgres Dockerized

- Containerized binaries
- Data directory on an external volume mount
- Configuration controlled by environment variables
- Many extensions (contrib, pgbouncer, postgis, pg_repack) installed together with multiple versions of PostgreSQL.
- Zalando own open-source extension: pam_oauth2 and bgmon
- Compressed to save space and speedup pod startup
- Patroni-based automatic failover for HA clusters

Automatic Failover with Patroni

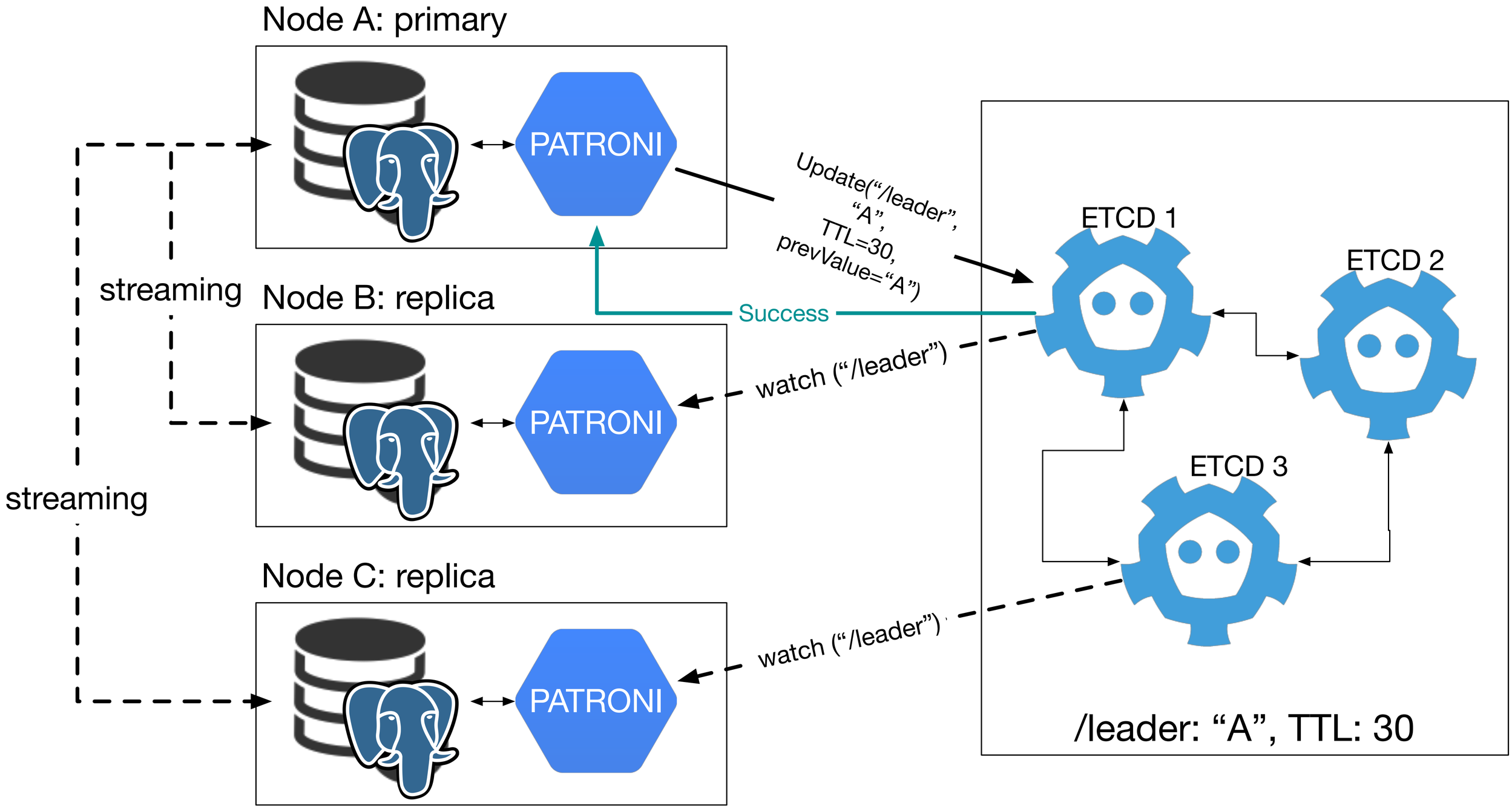
- Patroni is a Python daemon that manages one PostgreSQL instance.
- Patroni runs alongside PostgreSQL on the same system (needs access to the data directory)
- Instances are attributed to the HA cluster based on the cluster name in Patroni configuration.
- At most one instance in the HA cluster holds the master role, others replicate from it.

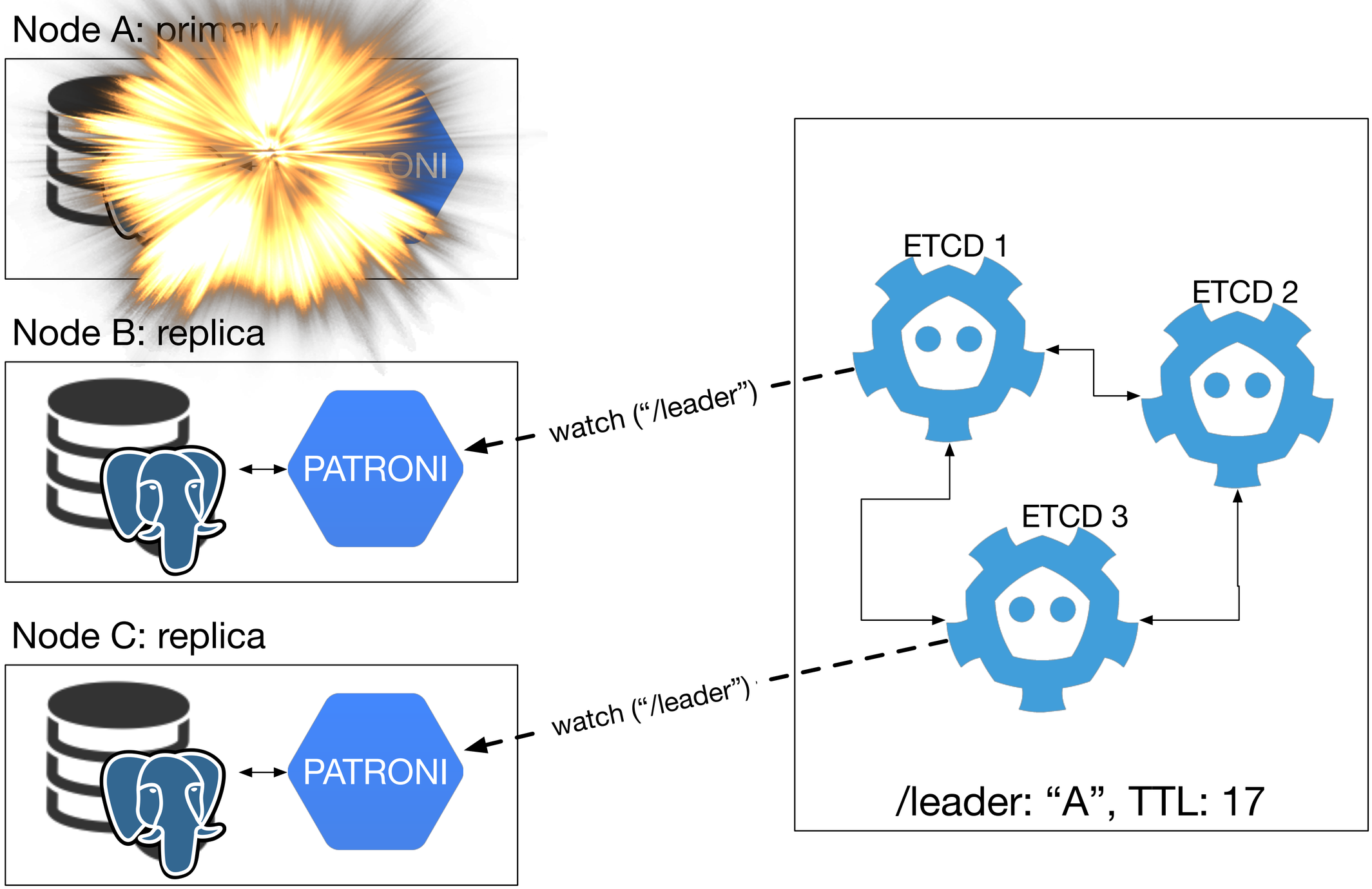
Managing cluster state

- Patroni keeps its cluster state in a distributed and strongly-consistent key-value system aka DCS (Etcd, Zookeeper, Consul or Kubernetes native API)
- A leader node name is set as a value of the leader key /$clustername/leader that expires after pre-defined TTL
- The leader node updates the leader key more often than expiration TTL, preventing its expiration
- A non-leader node is not allowed to update the leader key with its name (CAS operation).
- Each instance watches the leader key
- One the leader key expires, each remaining instance decides if it is “healthy enough” to become a leader
- The first “healthy” instance that creates the leader key with its name becomes the leader.

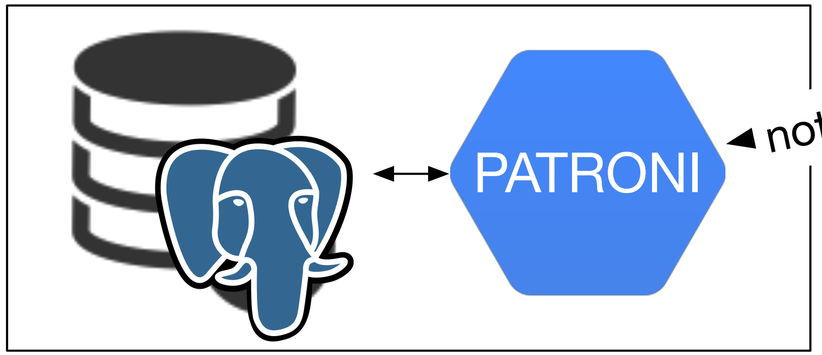
Avoiding split-brain

- Becoming a leader: first write the key in DCS, then promote.
- Demoting: first demote, then delete the leader key
- Member is never healthy if the old master is still running
- Member connects directly to other cluster members to get most up-to-date information
- Member is never healthy if its WAL position is behind some other member or too far behind the last known master position.

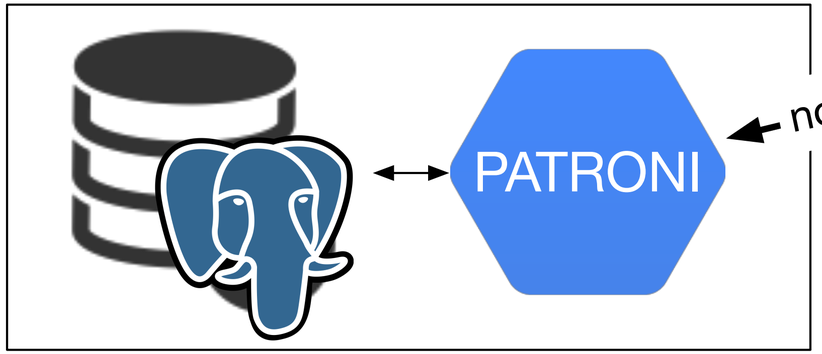

Node A: primary
PATRONI
Update("/leader",
"A",
TTL=30,
prevValue="A")
ETCD 1
ETCD 2
streaming
Node B: replica
Success
PATRONI
watch ("/leader")
streaming
ETCD 3
Node C: replica
PATRONI
watch ("/leader")
/leader: "A", TTL: 30


Node A: primary
Node B: replica
PATRONI
watch (“/leader”)
Node C: replica
PATRONI
watch (“/leader”)
ETCD 1
ETCD 2
ETCD 3
/leader: “A”, TTL: 17

Node B: readonly

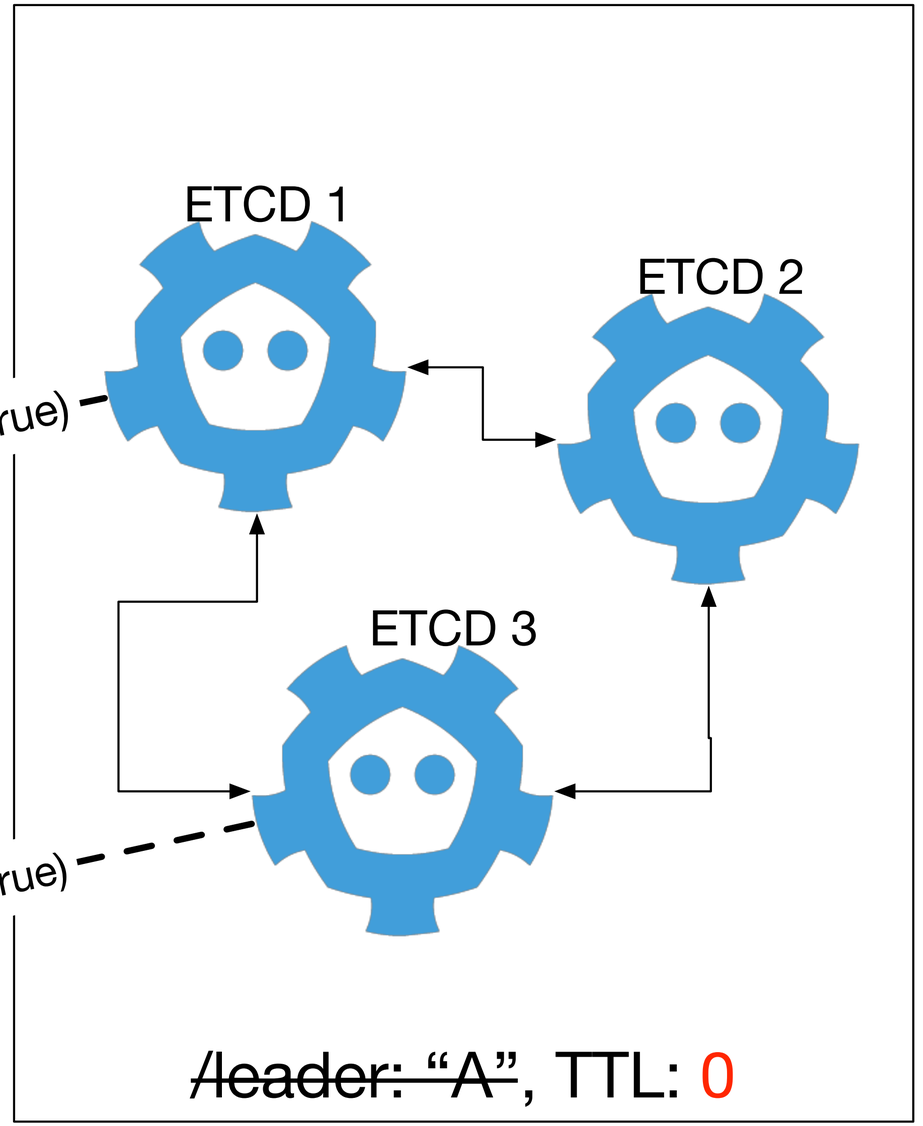


Node C: readonly



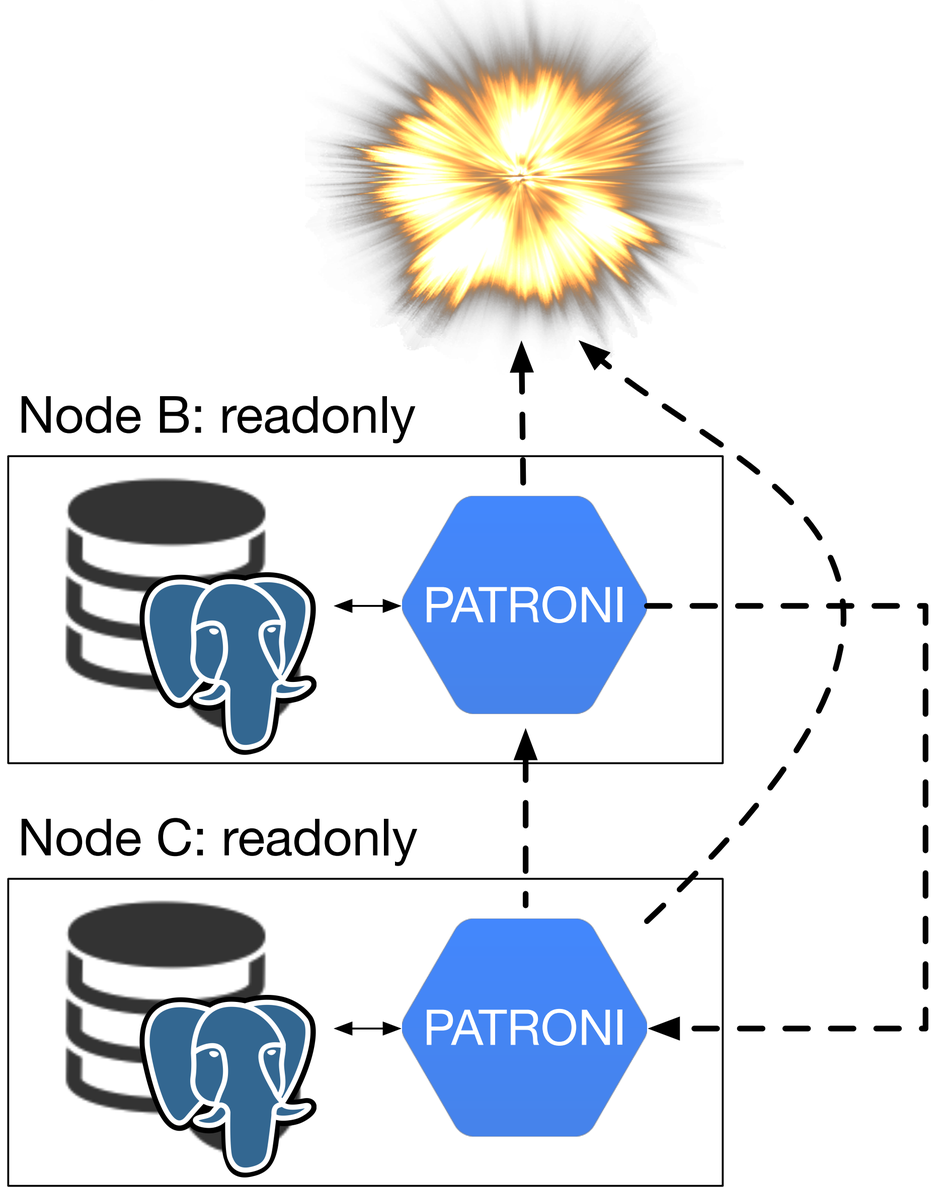
notify(/leader, expired=true)

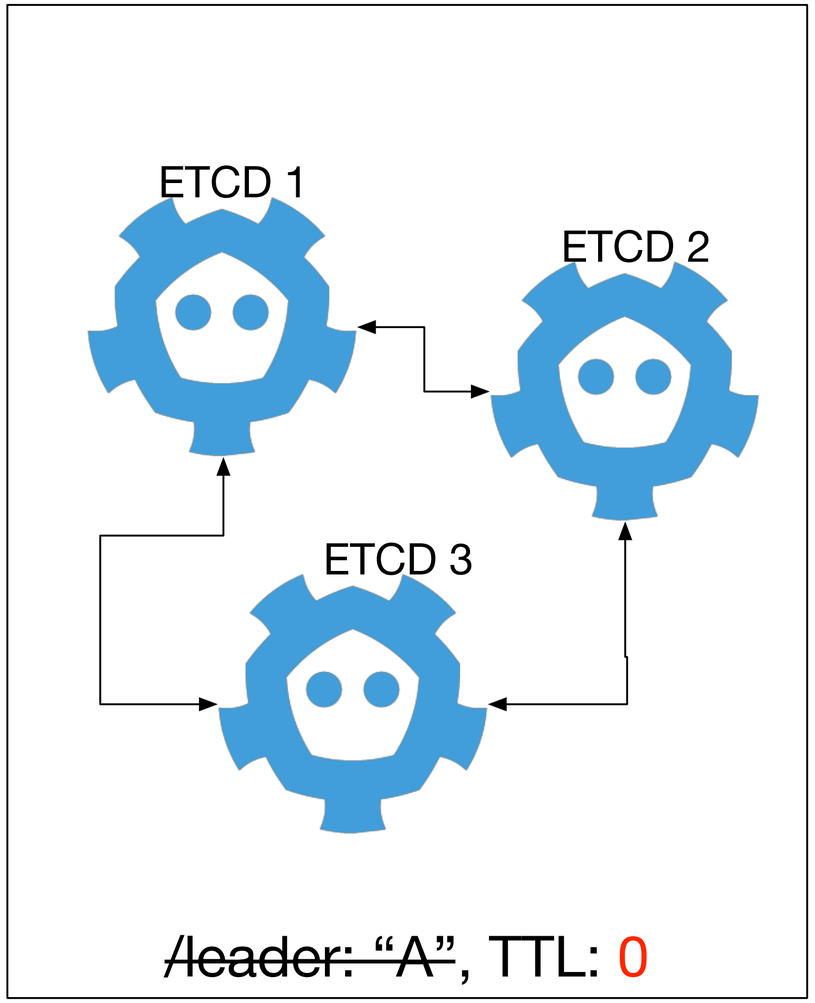
notify(/leader, expired=true)

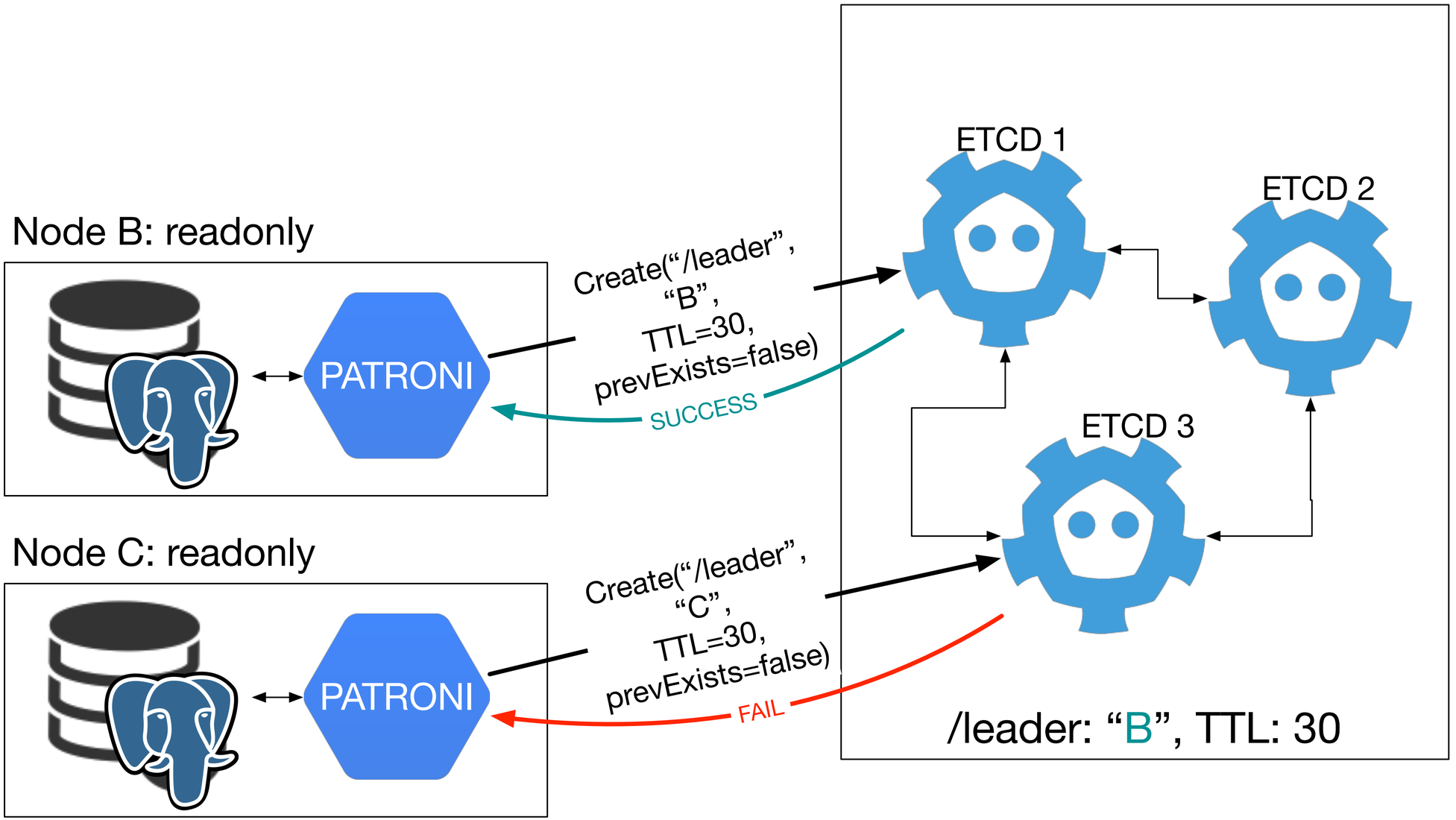


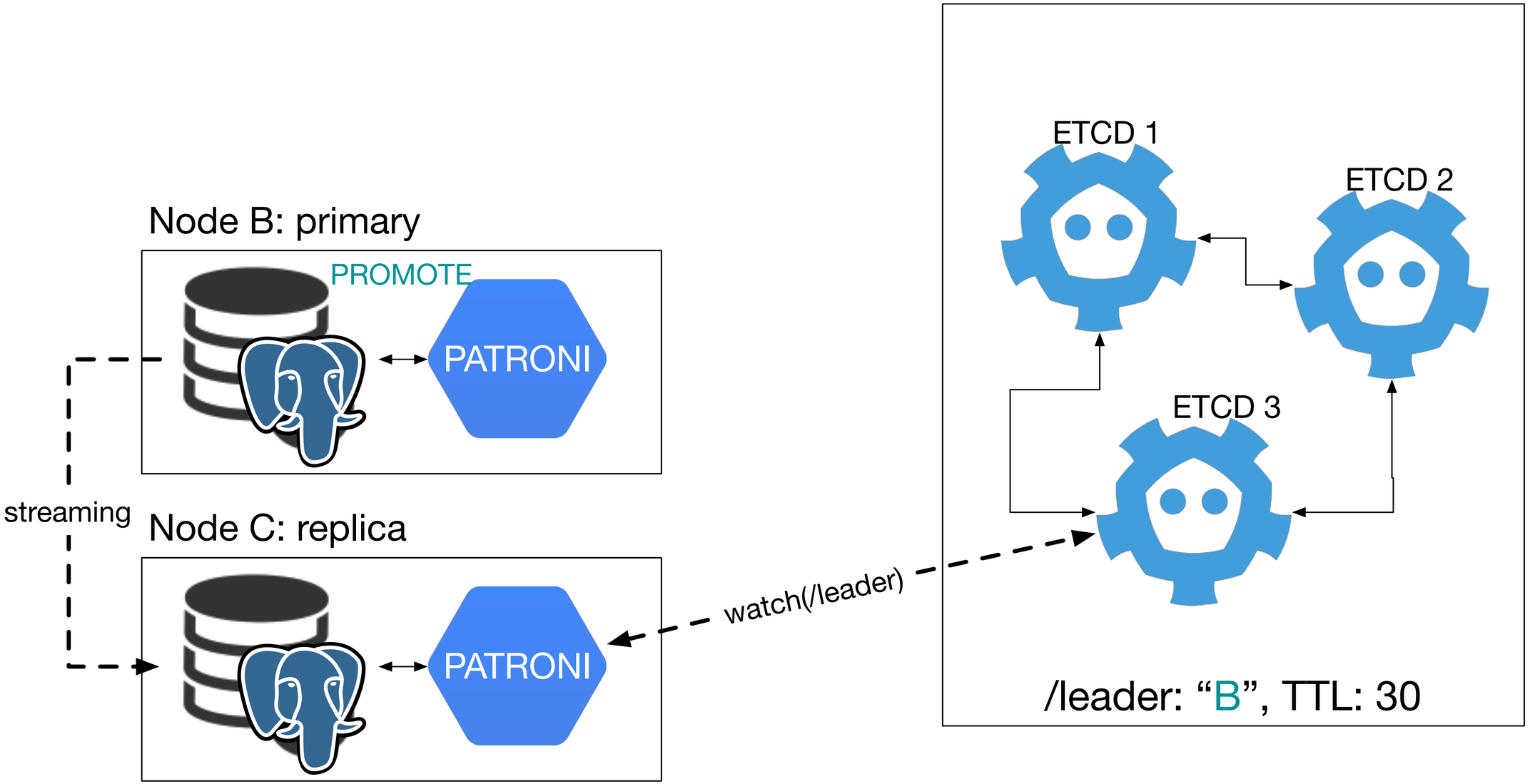
Node B:
GET A:8008/patroni -> **timeout**
GET C:8008/patroni -> wal_position: **100**

Node C:
GET A:8008/patroni -> **timeout**
GET B:8008/patroni -> wal_position: **100**






Node B: readonly
PATRONI
Create("/leader",
"B",
TTL=30,
prevExists=false)
SUCCESS
Node C: readonly
PATRONI
Create("/leader",
"C",
TTL=30,
prevExists=false)
FAIL
ETCD 1
ETCD 2
ETCD 3
/leader: "B", TTL: 30


Node B: primary
PROMOTE
PATRONI
streaming
Node C: replica
PATRONI
watch(/leader)
ETCD 1
ETCD 2
ETCD 3
/leader: “B”, TTL: 30

From Kubernetes to Postgres HA

- Postgres Operator creates a StatefulSet
- A StatefulSet creates N identical pods
- Each pod runs Postgres docker image with Patroni
- Patroni initiates leader election, one pod is elected as primary
- Rest of the pods find the primary in the same cluster as they are and stream from it

Operator maintenance tasks

- Operator acts on manifest updates
- Configuration changes
- Resources changes (memory, disk, number of instances)
- Kubernetes cluster updates with minimum downtimes

Open-source

- Patroni: https://github.com/zalando/patroni
- Spilo (Postgres docker image): https://github.com/zalando/spilo
- PG Operator: https://github.com/zalando-incubator/postgres-operator
- Pam oauth: https://github.com/CyberDem0n/pam-oauth2
- bg_mon (background worker for top-like monitoring) https://github.com/CyberDem0n/bg_mon

Thank you!

Questions?

email: alexk@hintbits.com
twitter: @hintbits