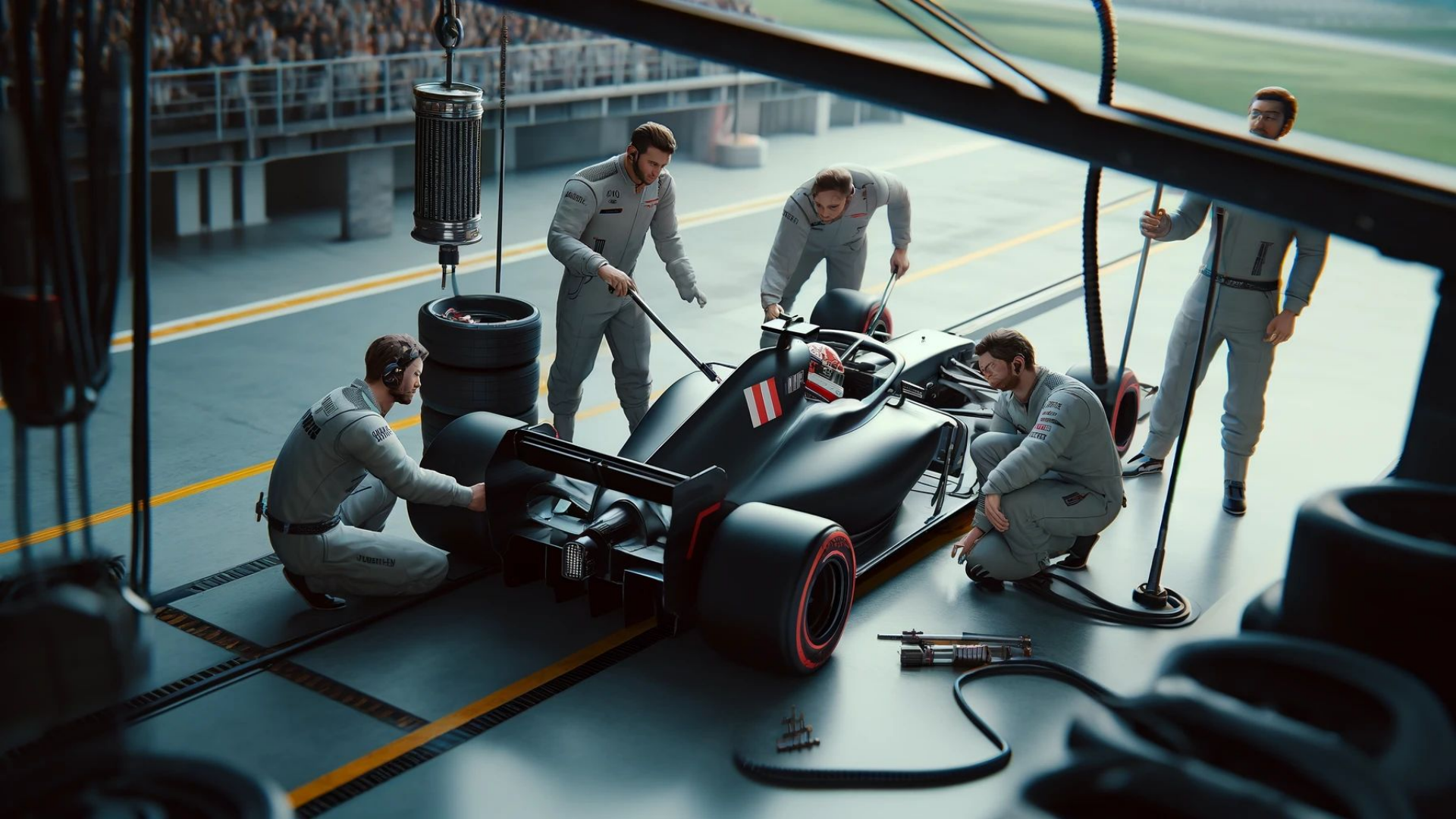
World's Best Racing Driver


44
PETRONAS
CROWDSTRIKE

Would You Let Josh Plan and Build the Engine ?



Bridging the DevOps to Data Divide with a Common Cloud Native Stack

Elad Hirsch & Mey Beisaron

Hello!

I am **Elad Hirsch**

- Software Architect for over ten years
- Tech Lead , CTO Office @ TERASKY
- Public Cloud & Cloud-Native
- Tech Advocate
-

Hello!

I am Mey Beisaron

- Backend Engineer for over five years
- Senior DevOps Engineer at FORTER
- Clojure, Nodejs, Groovy, Python
- International public speaker & mentor
- Sworn Star Wars fan

Provision, Set, Go!

Running DBs on K8s with CloudNativePG

Shift left, Stay Ahead

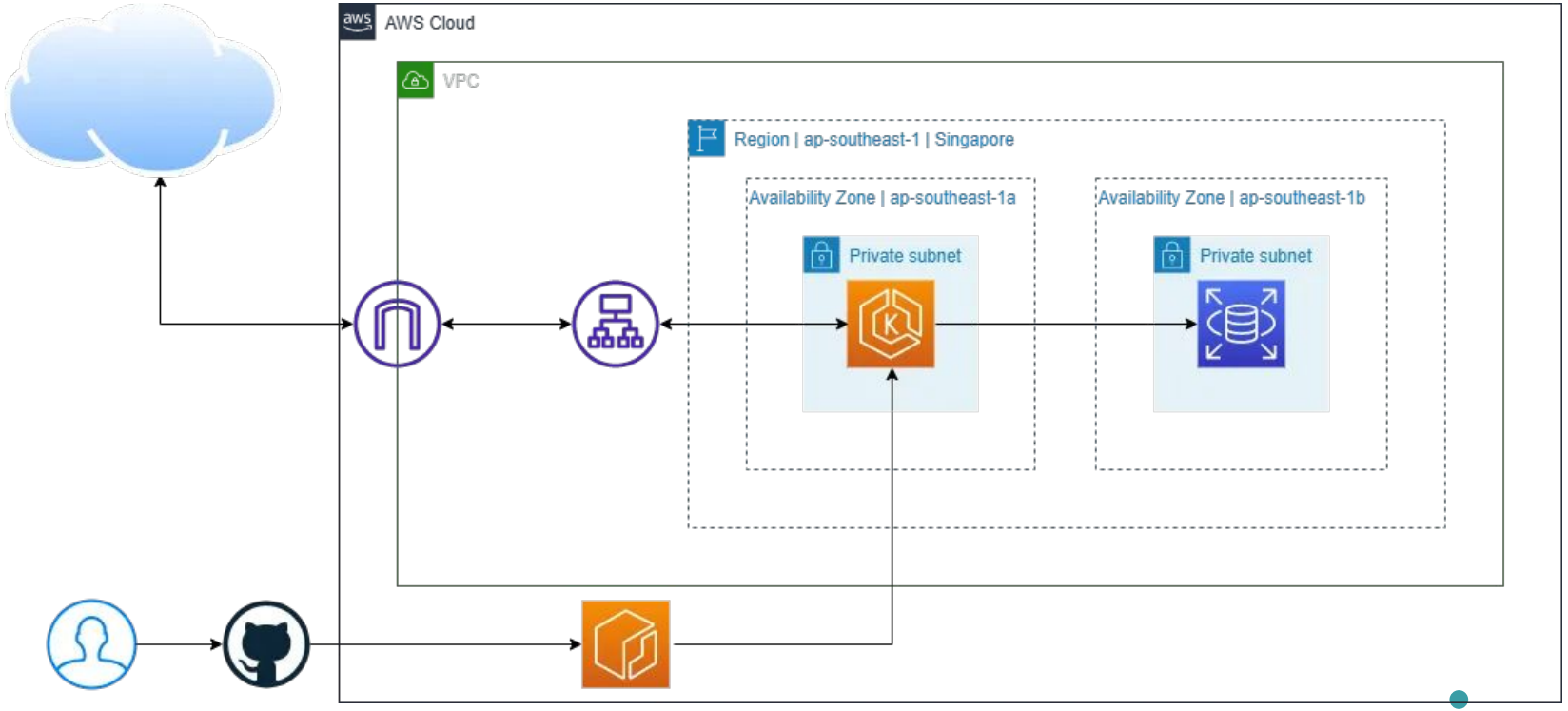
Manage DB Schema as an Infrastructure

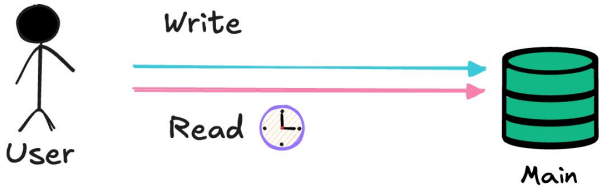
Fast Lane Monitoring

Adding visibility for DBs on K8s

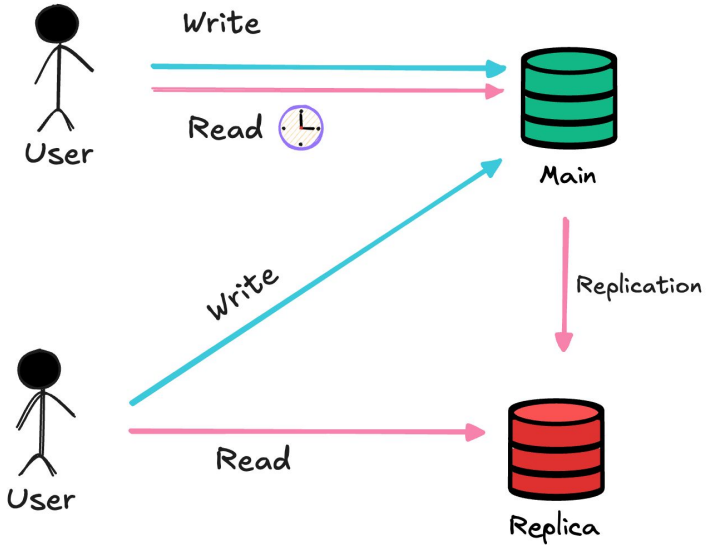
So You Need Good Database Reliability ...

You went with a managed DB, right?

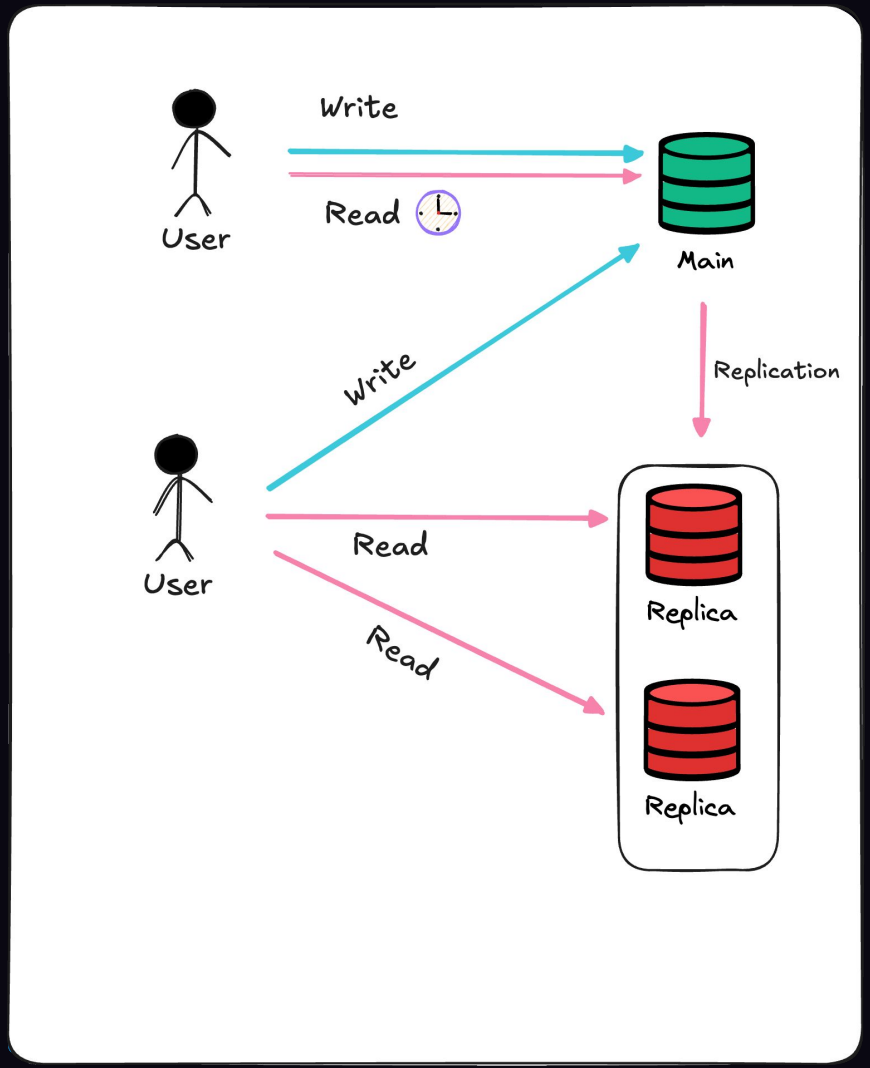




We'll need more than one database instance, right?

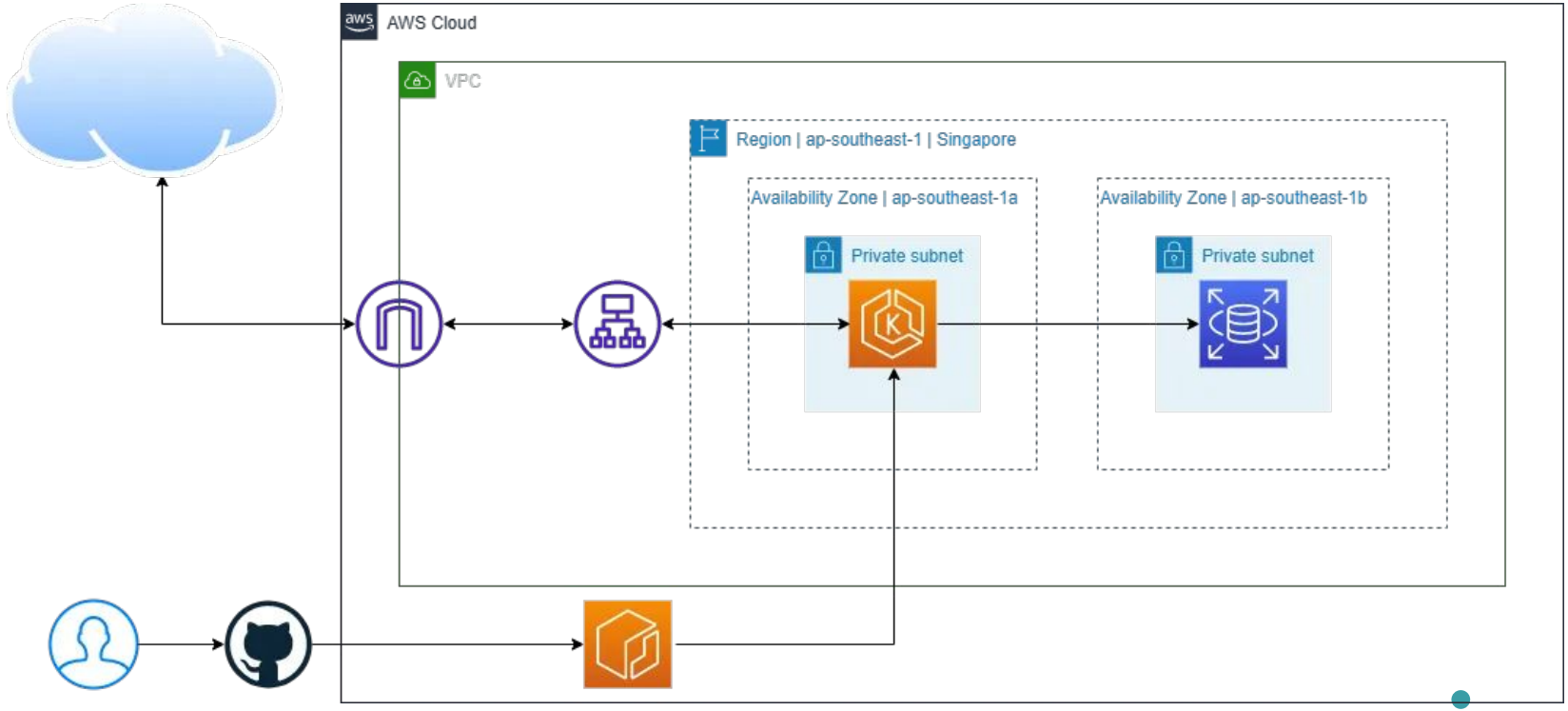


We'll need more than one database instance, right?

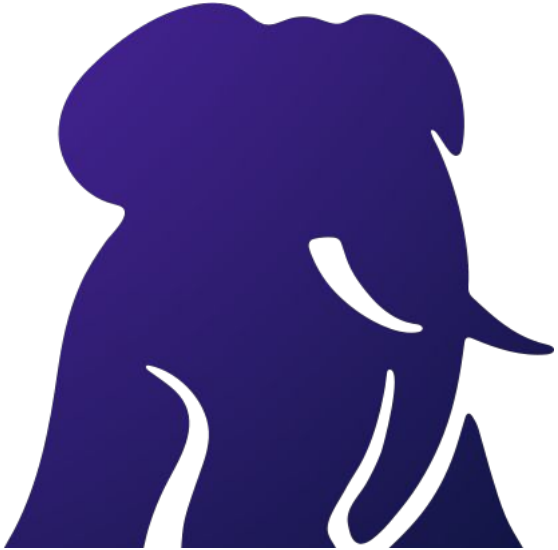


Do we need multiple AZ's for high availability?

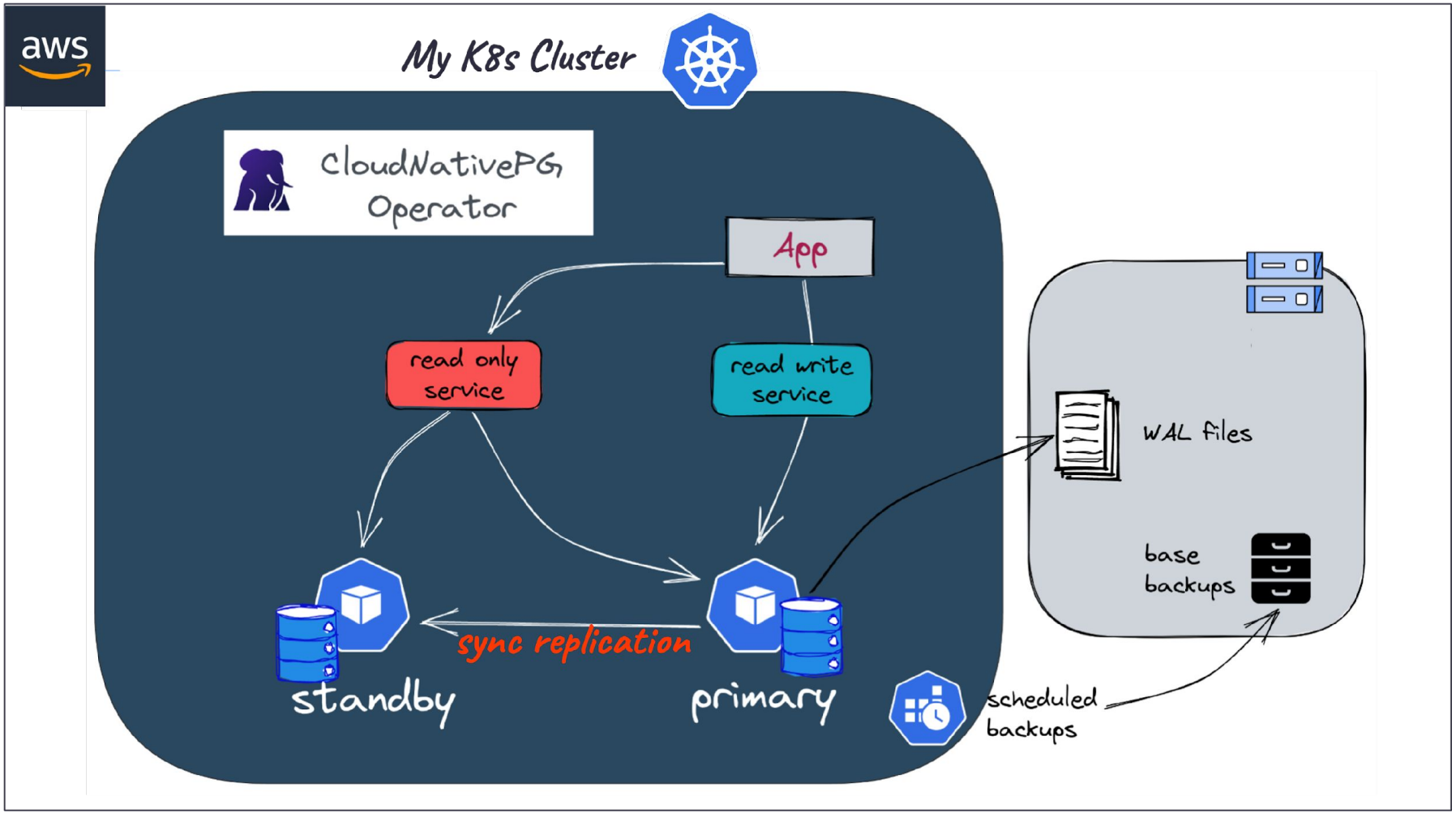
So it has to be managed DB, right?



CloudNativePG Operator



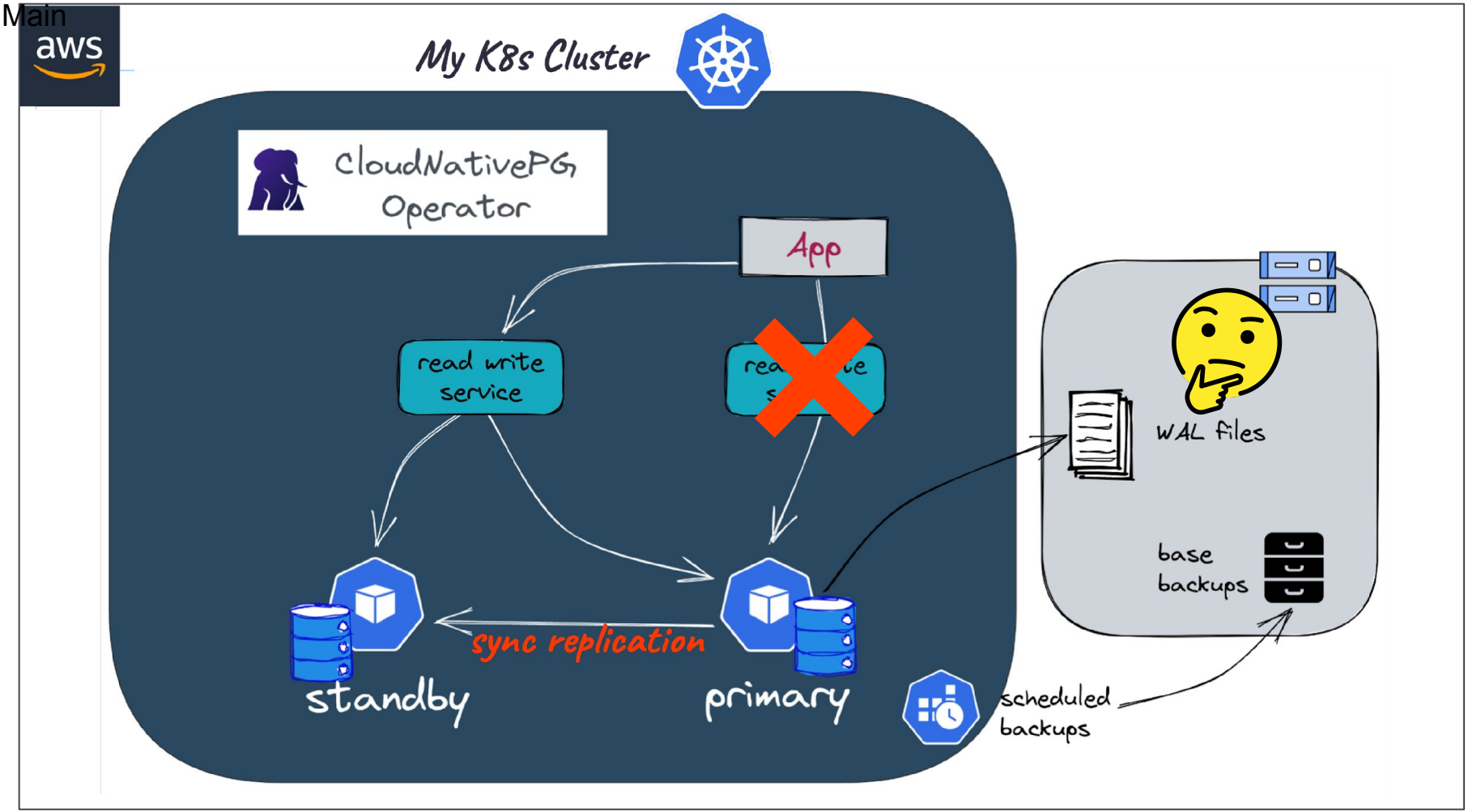
- Open Source and Vendor-Neutral
- Declarative, Immutable Design
- No Patroni and No StatefulSets
- HA with automated Failover


aws
My K8s Cluster
CloudNativePG
Operator
App
read only
service
read write
service
WAL files
base
backups
sync replication
standby
primary
scheduled
backups

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
 name: cluster-example
spec:
 instances: 3
 minSyncReplicas: 1
 maxSyncReplicas: 1
 affinity:
  nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
        - key: topology.kubernetes.io/zone
          operator: In
          values:
           - us-east-1a
           - us-east-1b
           - us-east-1c
```

```yaml
  tolerations:
   - key: node-role.kubernetes.io/postgres
     operator: Exists
     effect: NoSchedule
  topologySpreadConstraints:
   - maxSkew: 1
     topologyKey: topology.kubernetes.io/zone
     whenUnsatisfiable: DoNotSchedule
     labelSelector:
      matchLabels:
       cnpg.io/cluster: cluster-example
  storage:
    size: 40Gi
  walStorage:
    size: 10Gi
```

Main


My K8s Cluster
CloudNativePG Operator
App
read write service
WAL files
base backups
sync replication
standby
primary
scheduled backups


```
postgres@5d71369c69ff:~/data$ ls -l
total 132
drwx------ 2 postgres postgres  4096 Sep 30 17:24 pg_tblspc
drwx------ 2 postgres postgres  4096 Sep 30 17:24 pg_twophase
-rw------- 1 postgres postgres     3 Sep 30 17:24 PG_VERSION
drwx------ 4 postgres postgres  4096 Oct 14 17:00 pg_wal
drwx------ 2 postgres postgres  4096 Sep 30 17:24 pg_xact
-rw------- 1 postgres postgres    88 Sep 30 17:24 postgresql.auto.conf
-rw------- 1 postgres postgres 30827 Oct 16 12:27 postgresql.conf
-rw------- 1 postgres postgres    36 Oct 17 15:39 postmaster.opts
-rw------- 1 postgres postgres    94 Oct 17 15:39 postmaster.pid
postgres@5d71369c69ff:~/data$
```

Boosting Data Reliability and Transaction Efficiency

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
 name: cluster-example
spec:
 storage:
   size: 100Gi
 walStorage:
   storageClass: wal-fast-storage
   size: 10Gi
```

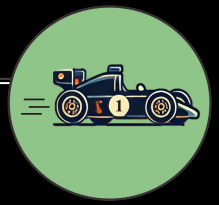
```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
 name: wal-fast-storage
provisioner: kubernetes.io/aws-ebs
parameters:
 type: io2          # High-performance SSD for WAL
 iopsPerGB: "100"   # Increase IOPS;
                    # Suitable for heavy-write workloads
 fsType: ext4
 encrypted: "true"
reclaimPolicy: Delete
allowVolumeExpansion: true
volumeBindingMode: WaitForFirstConsumer
```

Provision, Set, Go!

Running DBs on K8s with CloudNativePG

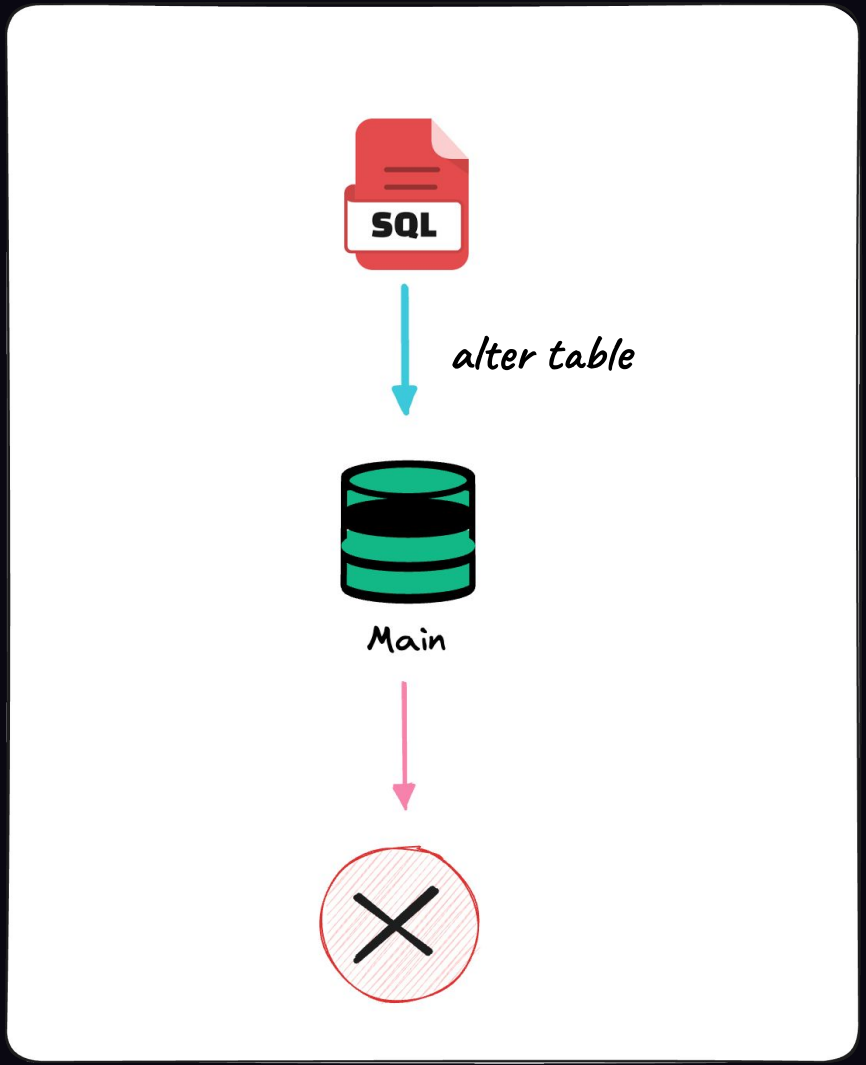
Shift left, Stay Ahead

Manage DB Schema as an Infrastructure



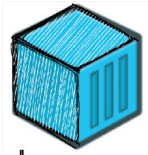
Fast Lane Monitoring

Adding visibility for DBs on K8s



So You Need a Reliable Way to Manage Schema Changes?

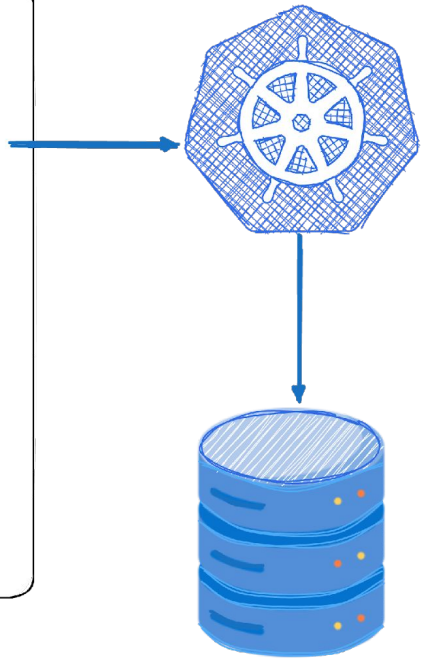
Schema update during startup



Java

Liquibase

```
<changeSet id="300" author="developer">
    <dropColumn tableName="book" columnName="name"/>
</changeSet>
```



Schema update in *InitContainers*

```
// schema.prisma

model User {
  id    Int     @id
@default(autoincrement())
  name  String
  email String  @unique
  bio   String?
}
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-app
spec:
 template:
   spec:
     containers:
       # Main application container

     initContainers:
       - name: prisma-migrate
         image: node:18
         command: ["npx", "prisma",
                   "migrate", "dev",
                   "--name", "change_set"]
         env:
           - name: DATABASE_URL
             valueFrom:
               secretKeyRef:
                 name: my-database-secret
                 key: DATABASE_URL
         volumeMounts:
           - name: app-source
             mountPath: /app

     volumes:
       - name: app-source
         emptyDir: {}
```

Schema Update Managed with Kubernetes Job

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: liquibase-migration-job
spec:
  template:
    spec:
      containers:
        - name: liquibase
          image: liquibase/liquibase:latest
          env:
          args:
            - "--changeLogFile=/changelog/change300.xml"
            - "update"
          volumeMounts:
            - name: changelog-volume
              mountPath: /liquibase/changelog
      restartPolicy: OnFailure
      volumes:
        - name: changelog-volume
          configMap:
            name: liquibase-changelog-config
```

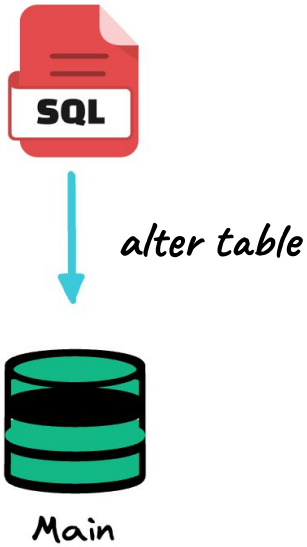
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: liquibase-changelog-config
data:
  changelog.xml: |
    <databaseChangeLog>
        <changeSet id="300" author="developer">
            <dropColumn tableName="book" columnName="name"/>
        </changeSet>
    </databaseChangeLog>
```

Back to Square One ?

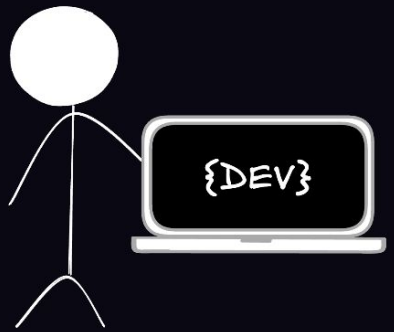
Am I Clear on My Schema Changes and how it’s going to R ?

Can I Simply Declare My Evolving Schema's as a Target State ?

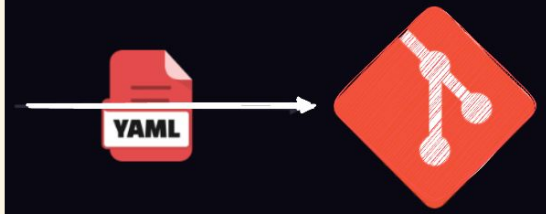
How Do I Ensure My Schema Stays Consistent with My Target State ?



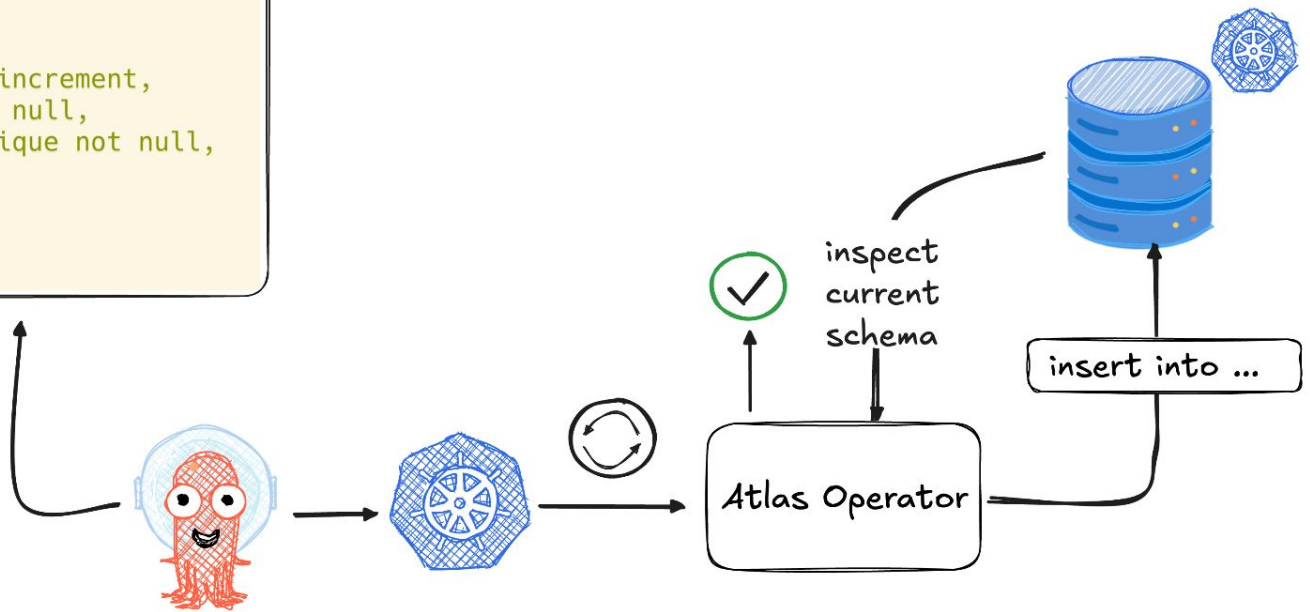
atlas Cloud Native Operator



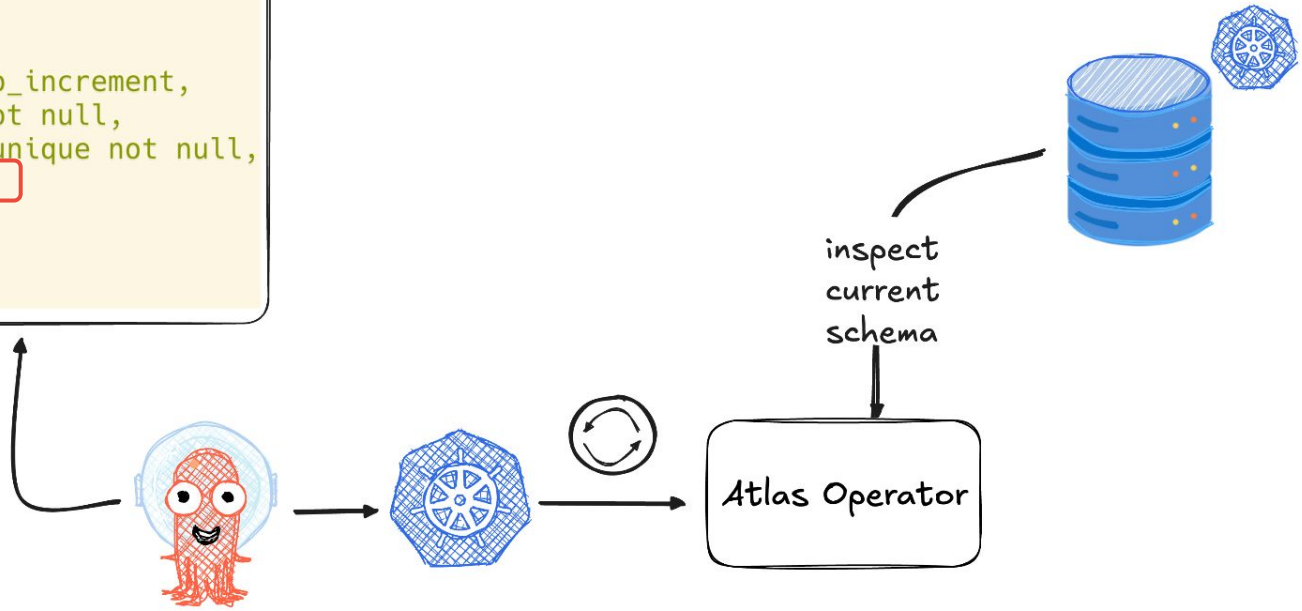

```yaml
apiVersion: db.atlasgo.io/v1alpha1
kind: AtlasSchema
metadata:
  name: myapp-schema
spec:
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        email varchar(255) unique not null,
        primary key (id)
      );
```



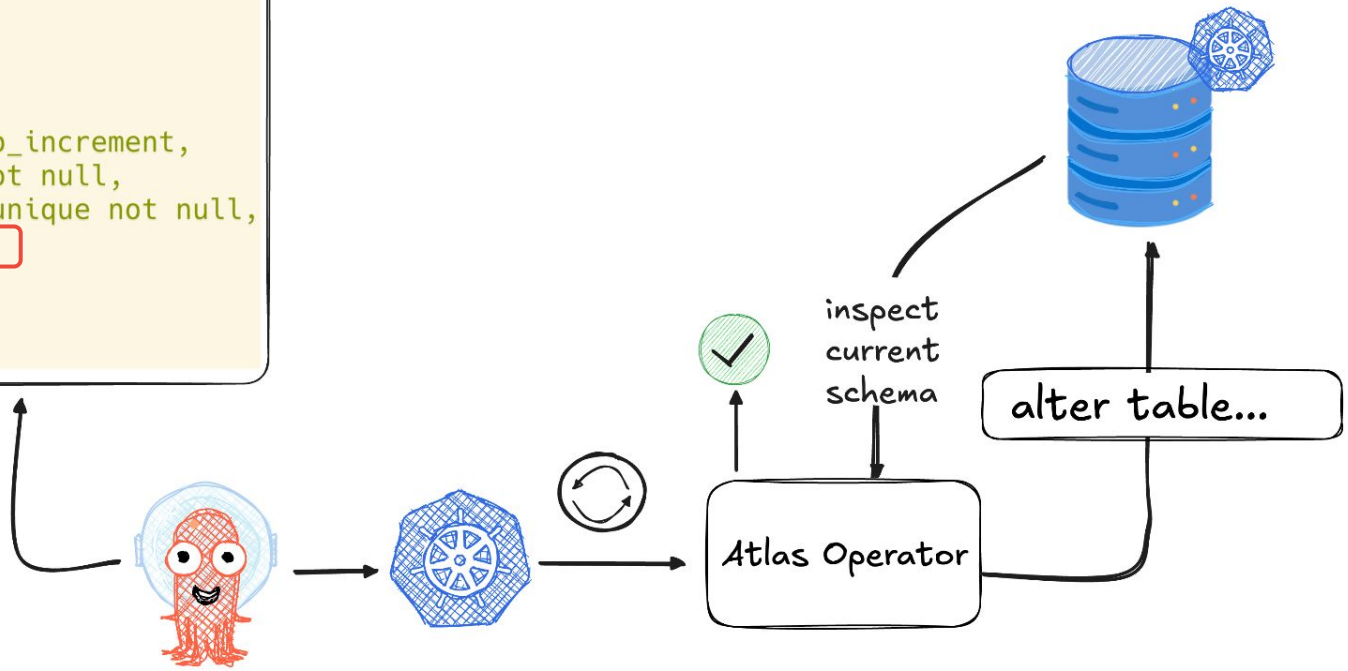
```yaml
kind: AtlasSchema
metadata:
  name: myapp-schema
spec:
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        email varchar(255) unique not null,
        primary key (id)
      );
```



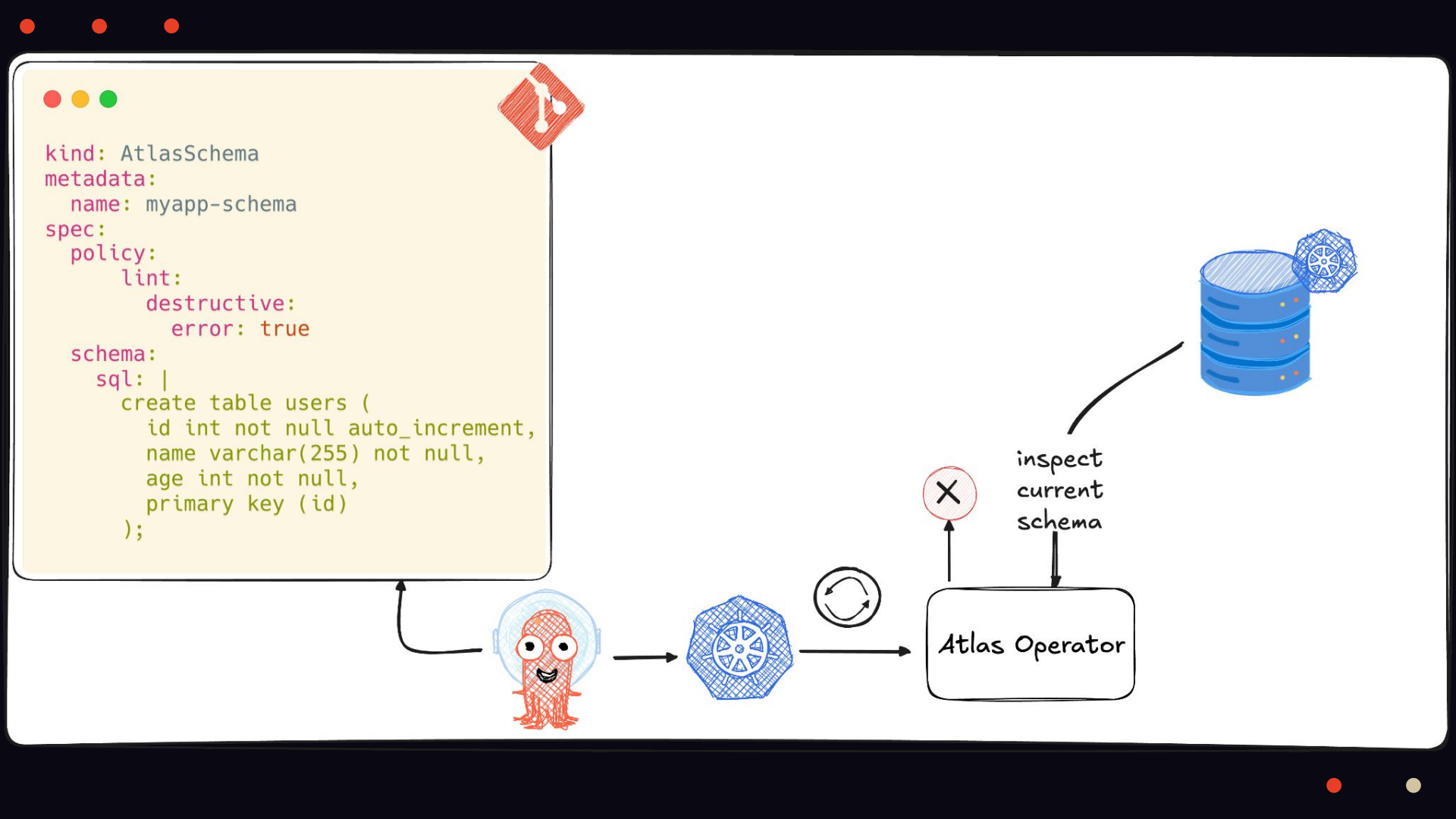
```yaml
kind: AtlasSchema
metadata:
  name: myapp-schema
spec:
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        email varchar(255) unique not null,
        age int not null,
        primary key (id)
      );
```



```yaml
kind: AtlasSchema
metadata:
  name: myapp-schema
spec:
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        email varchar(255) unique not null,
        age int not null,
        primary key (id)
      );
```



```yaml
kind: AtlasSchema
metadata:
  name: myapp-schema
spec:
  policy:
      lint:
        destructive:
          error: true
  schema:
    sql: |
      create table users (
        id int not null auto_increment,
        name varchar(255) not null,
        age int not null,
        primary key (id)
      );
```



Provision, Set, Go!

Running DBs on K8s with CloudNativePG

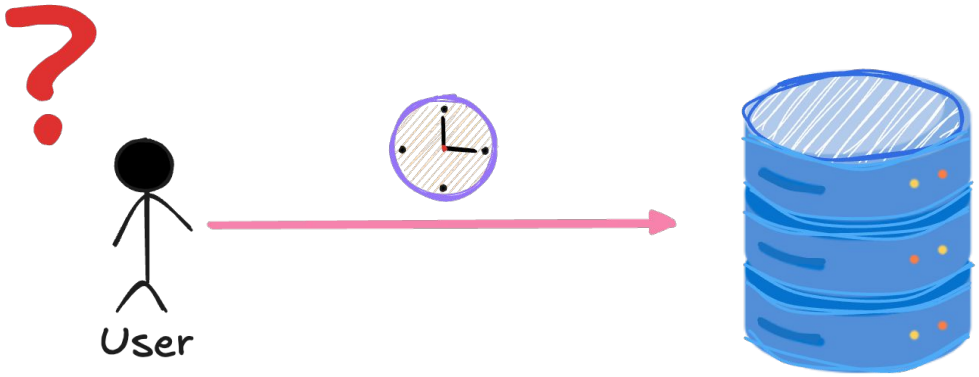
Shift left, Stay Ahead

Manage DB Schema as an Infrastructure

Fast Lane Monitoring

Adding visibility for DBs on K8s

So,
You've Added More Instances but the Query is Still Slow—Why?



Monitoring


default
Operator Namespace
cnpg-system
Database Namespace
test
Cluster
database-cluster
Instances
All
No alerts matching filters
Health
Replication
Healthy
Lag
Healthy
Storage
Healthy
Connections
Healthy
Backups
Healthy
WAL
Healthy
Operator Status
Online
Reconcile errors
None
Overview
Last failover
7 days ago
Version
15.4
TPS
14.1
CPU Utilisation
Missing request
Total 0.03
Memory
Health
Status
Clustering / replicas
Zone
Connections
database-cluster-1
Up
Yes
0
westeurope-2
25
0
03/15
03/17
03/19
03/21
Last: 1 Mean: 2.01
database-cluster-2
Up
Yes
2
westeurope-3
Last: 30 Mean: 24.5
database-cluster-3
Up
Yes
0
westeurope-3
2
Last: 1 Mean: 1
guration
(17 panels)
tional Stats
age
03/15 00:00
03/16 00:00
03/17 00:00
03/18 00:00
03/19 00:00
03/20 00:00
03/21 00:00
database-cluster-2
database-cluster-3
Memory Usage (w/o cach
1.50 GiB
1 GiB
512 MiB
0 B
03/15 00:00
database-cluster-1

Golden Path

Enable Comprehensive Monitoring

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
```

Enable Comprehensive Monitoring

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
```

Enable Comprehensive Monitoring

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
```

Golden Path and Support for Super Users

User Defined Metrics for Advanced Use Cases

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
    customQueriesConfigMap:
      - name: active-connections-metric
        key: custom-queries
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: active-connections-metric
  namespace: monitoring
data:
  custom-queries: |
    active_connections_query:
      query: |
        SELECT
          current_database() AS datname,
          state,
        COUNT(*) AS active_connections
        FROM pg_stat_activity
        WHERE state = 'active'
        GROUP BY state, datname;
      metrics:
        - datname:
            usage: "LABEL"
        - state:
            usage: "LABEL"
        - active_connections:
            usage: "GAUGE"
```

User Defined Metrics for Advanced Use Cases

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
    customQueriesConfigMap:
      - name: active-connections-metric
        key: custom-queries
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: active-connections-metric
  namespace: monitoring
data:
  custom-queries: |
    active_connections_query:
      query: |
        SELECT
          current_database() AS datname,
          state,
        COUNT(*) AS active_connections
        FROM pg_stat_activity
        WHERE state = 'active'
        GROUP BY state, datname;
      metrics:
        - datname:
            usage: "LABEL"
        - state:
            usage: "LABEL"
        - active_connections:
            usage: "GAUGE"
```

User Defined Metrics for Advanced Use Cases

```yaml
apiVersion: postgresql.cnpg.io/v1
kind: Cluster
metadata:
  name: example-cluster
  namespace: monitoring
spec:
  instances: 1
  storage:
    size: 10Gi
  monitoring:
    enablePodMonitor: true
    customQueriesConfigMap:
      - name: active-connections-metric
        key: custom-queries
```

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: active-connections-metric
  namespace: monitoring
data:
  custom-queries: |
    active_connections_query:
      query: |
        SELECT
          current_database() AS datname,
          state,
        COUNT(*) AS active_connections
        FROM pg_stat_activity
        WHERE state = 'active'
        GROUP BY state, datname;
      metrics:
        - datname:
            usage: "LABEL"
        - state:
            usage: "LABEL"
        - active_connections:
            usage: "GAUGE"
```

Key Takeaways

- Consider Cloud Native Databases
- Shift Left and Treat Schema as Infra
- No One-Size-Fits-All Solution
- Bridging the Data and DevOps Gap

THANKS!

Any questions?