
SITE RELIABILITY ENGINEERING
KEYBOSS
START

TABLE OF CONTENTS

TABLE OF CONTENTS

M2 OBJECTIVES

01 **High Availability**

The system survives component, pod, and node failures without significant service degradation.

02 **Observability**

Move beyond `kubectl logs` to deep system visibility: metrics, alerts, dashboards, and long-term trends.

03 **Automation**

Eliminate operational toil with pipelines and scripts covering deploy, rollback, and failure recovery.

04 **Architectural Review**

Critically assess M1 decisions and identify what had to change to support resilience and scaling.

Transform the system from "it works" to "production-ready"

WHAT CHANGED FROM M1 TO M2?

M1 (Baseline)	M2 (Production)	Reason
PostgreSQL single-replica	**High Availability:** primary + standby + watchdog	Any DB crash = full outage; no way to recover without data loss
Redis single-node	**Sentinel:** master + replica + 3 sentinels	Pod restart lost all Celery jobs and cached sessions
Manual deploy (`kubectl apply`)	Single script (`**deploy.sh --deti**`) + auto rollback	No reproducibility; DETI env differences → silent failures
No monitoring	**Prometheus + Grafana + Alertmanager**	Zero visibility → issues only noticed after user impact
No autoscaling	**Horizontal Pod Autoscaler:** saleor-api 2 → 6, celery 2 → 4 (scale-up triggers: CPU >70%, Memory >80%)	Fixed replicas can't absorb traffic spikes
No backups	**Daily backup CronJob** (02:00 UTC): pg_dump + gzip to PVC, 7-day retention · restore verified ✓	PVC failure = permanent data loss with no recovery path

AUTOMATION: DEPLOY.SH

Auto Rollback

Trap ERR fires on any phase failure.
kubectl rollout undo deployment/X
StatefulSets excluded (persistent data).

Env Adaptation (sed)

keyboss → tenant-keyboss
local-path → longhorn
any PVC → 1Gi (DETI limit)
localhost → http://keyboss.deti/

```
# Two environments , one script:
bash deployment / M2 / deploy . sh -- local
bash deployment / M2 / deploy . sh -- deti
bash deployment / M2 / deploy . sh -- deti -- reset
```

#	Phase	Details
1	Namespace & Config	ConfigMap + Secrets
2	Stateful (PostgreSQL HA + Redis)	Wait for pod Ready
3	Django Migration	Idempotent Job
4	Deployments	API, Celery, Storefront, Dashboard
5	HPA + Sentinel + Watchdog	High Availability layer
6	Backup	PostgreSQL daily CronJob
7	Monitoring	Prometheus, Grafana, Alertmanager
8	ServiceMonitor	prometheus.deti integration

CONFIG AND SECRET MANAGEMENT

ConfigMap: keyboss-config

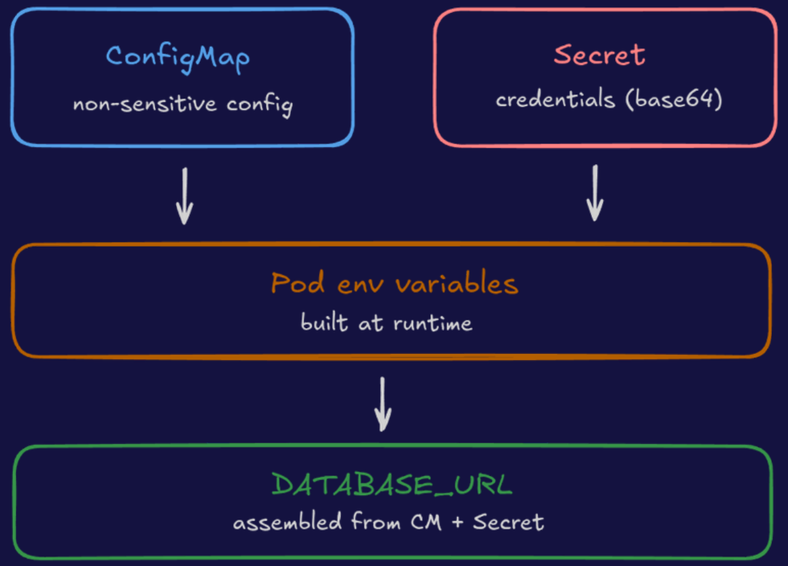
Non-sensitive parameters:

- **POSTGRES_HOST (patched on failover)**
- **REDIS_URL, CELERY_BROKER_URL**
- **API_URI, PUBLIC_URL**
- **NEXT_PUBLIC_API_URI**
- **ALLOWED_HOSTS**

Secret: keyboss-secrets

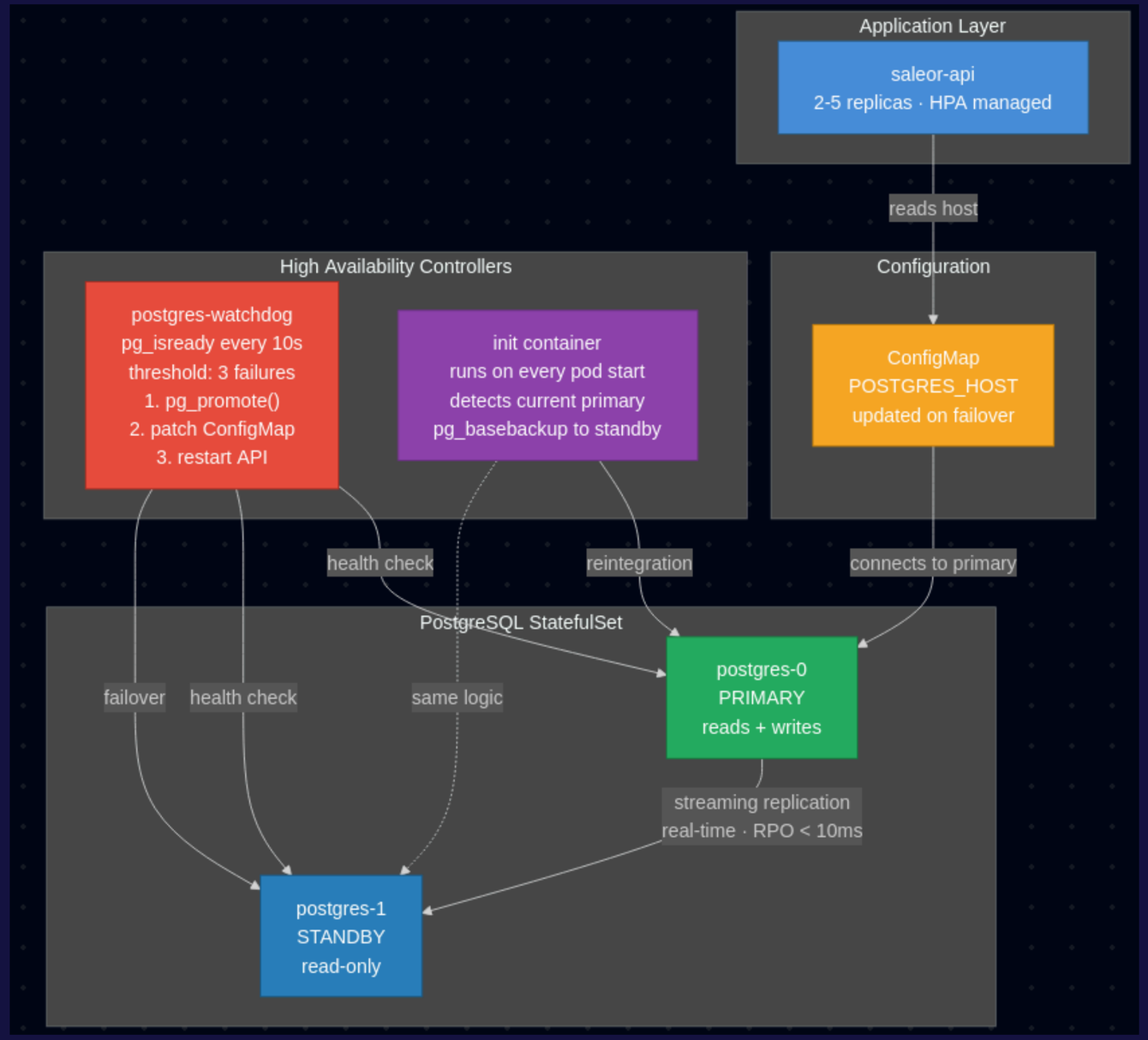
Sensitive credentials:

- **POSTGRES_PASSWORD**
- **Django SECRET_KEY**
- **RSA private key**
- **Replication credentials**



✔ **Rollback: kubectl rollout undo**
StatefulSets excluded → persistent data.

POSTGRESQL HA: ARCHITECTURE



POSTGRESQL HA: ARCHITECTURE

01

saleor-api (2–6 pods, HPA managed)

Connects via ConfigMap POSTGRES_HOST.

02

postgres-0 PRIMARY

reads + writes · streaming replication
RPO < 10 ms

03

postgres-1 STANDBY

read-only

04

postgres-watchdog

pg_isready/10 s · 3 fails ⇒ failover
pg_promote() → patch CM → restart API

RPO < 10 ms · RTO ≈ 30 s

POSTGRESQL HA: EVIDENCE

1. Trigger → simulate a node failure by deleting the primary pod:

```
^Cguilherme-silva@guilherme-silva:~/UNI/GIC/sre-project-grupo-keyboss$ kubectl delete pod postgres-1 -n tenant-keyboss
pod "postgres-1" deleted from tenant-keyboss namespace
```

The watchdog detects 3 consecutive failures (~30s) and acts automatically, no human intervention needed.

2. Result → watchdog promotes the standby and redirects traffic:

```
  }
}2026-05-26 05:14:02 [WATCHDOG] === FAILOVER COMPLETE — new primary: postgres-1.postgres ===
2026-05-26 05:14:02 [WATCHDOG] Watchdog now monitoring new primary: postgres-1.postgres
2026-05-26 05:16:52 [WATCHDOG] Primary UNREACHABLE (1/3)
2026-05-26 05:17:02 [WATCHDOG] Primary UNREACHABLE (2/3)
2026-05-26 05:17:12 [WATCHDOG] Primary UNREACHABLE (3/3)
2026-05-26 05:17:12 [WATCHDOG] === FAILOVER TRIGGERED ===
 }2026-05-26 05:17:16 [WATCHDOG] === FAILOVER COMPLETE — new primary: postgres-0.postgres ===
 2026-05-26 05:17:16 [WATCHDOG] Watchdog now monitoring new primary: postgres-0.postgres
```

POSTGRESQL HA: EVIDENCE

3. Auto-reintegration → when postgres-1 restarts:

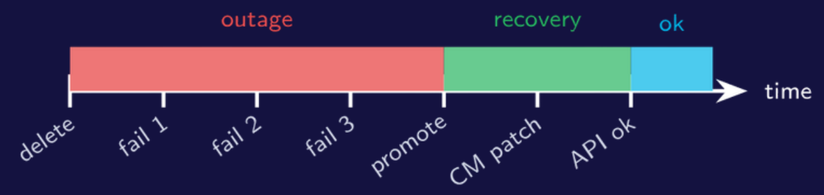
```
┌──(meowstermind㉿LEGION-LULU)-[~/Downloads/GIC/sre-project-grupo-keyboss]
└─$ export KUBECONFIG=/home/meowstermind/Downloads/GIC/sre-project-grupo-keyboss/tenant-keyboss-kubeconfig.yaml

kubectl logs postgres-1 -n tenant-keyboss -c init-replication 2>/dev/null || \
kubectl logs postgres-1 -n tenant-keyboss --previous 2>/dev/null | grep -iE "init|basebackup|Reintegration|standby|primary|tablespace|%"
Pod ordinal: 1  (hostname: postgres-1)
postgres-0.postgres is reachable. Checking if it became primary (up to 60s)...
REINTEGRATION: postgres-0.postgres is primary (attempt 1). Resyncing as standby...
waiting for checkpoint
 6468/45487 kB (14%), 0/1 tablespace
18268/45487 kB (40%), 0/1 tablespace
29031/45487 kB (63%), 0/1 tablespace
40353/45487 kB (88%), 0/1 tablespace
45497/45497 kB (100%), 0/1 tablespace
45497/45497 kB (100%), 1/1 tablespace
Reintegration complete. Starting as standby of postgres-0.postgres.
```

✔ **Real test on DETI cluster.**

Total downtime: ≈ 35 seconds.
Zero manual intervention.

Metric	Value	
RPO	< 10 ms	✔
RTO	≈ 30 s	✔
Automatic	Yes	✔

RPO (Recovery Point Objective): maximum data loss tolerated; replication keeps this under 10 ms.
RTO (Recovery Time Objective): maximum time to restore service; watchdog promotes standby in ~30 s automatically.

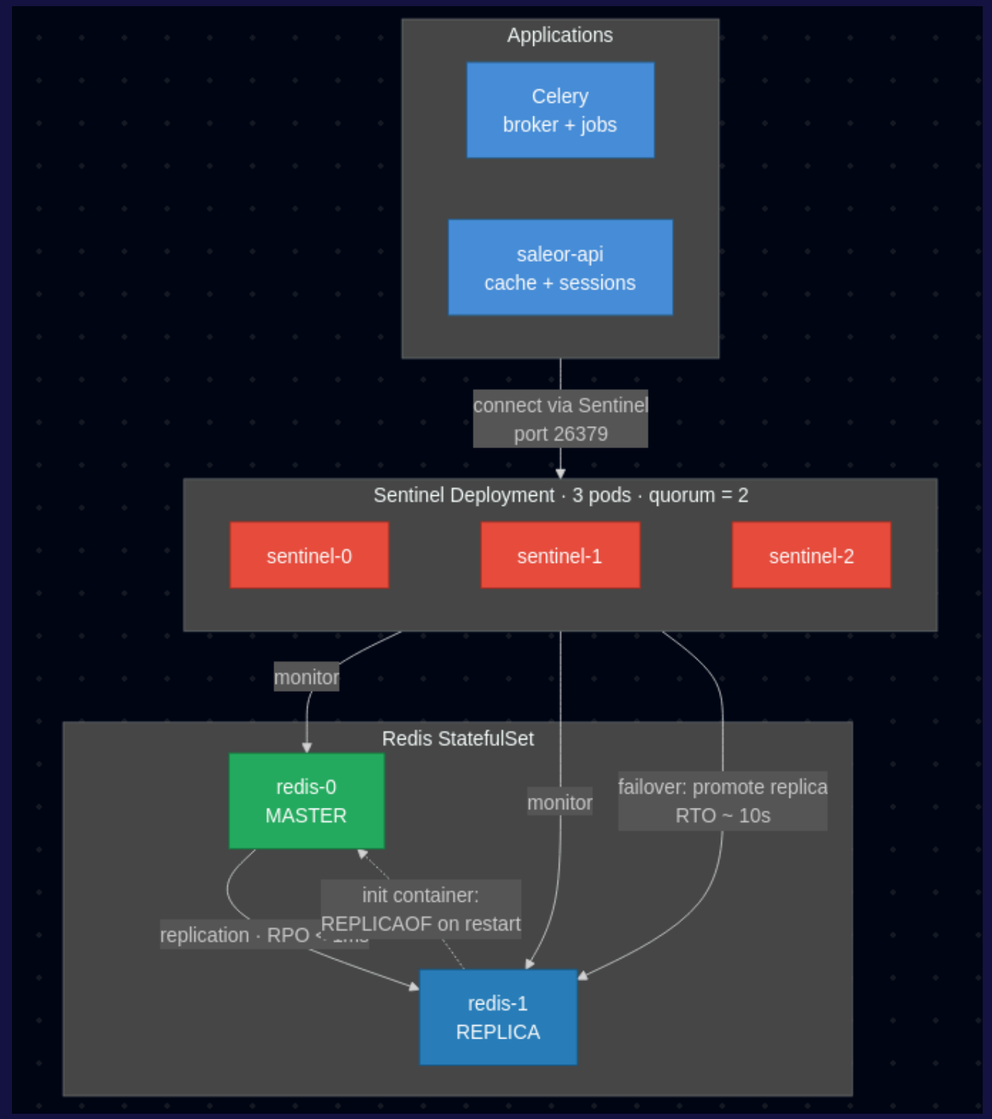


REDIS SENTINEL: ARCHITECTURE

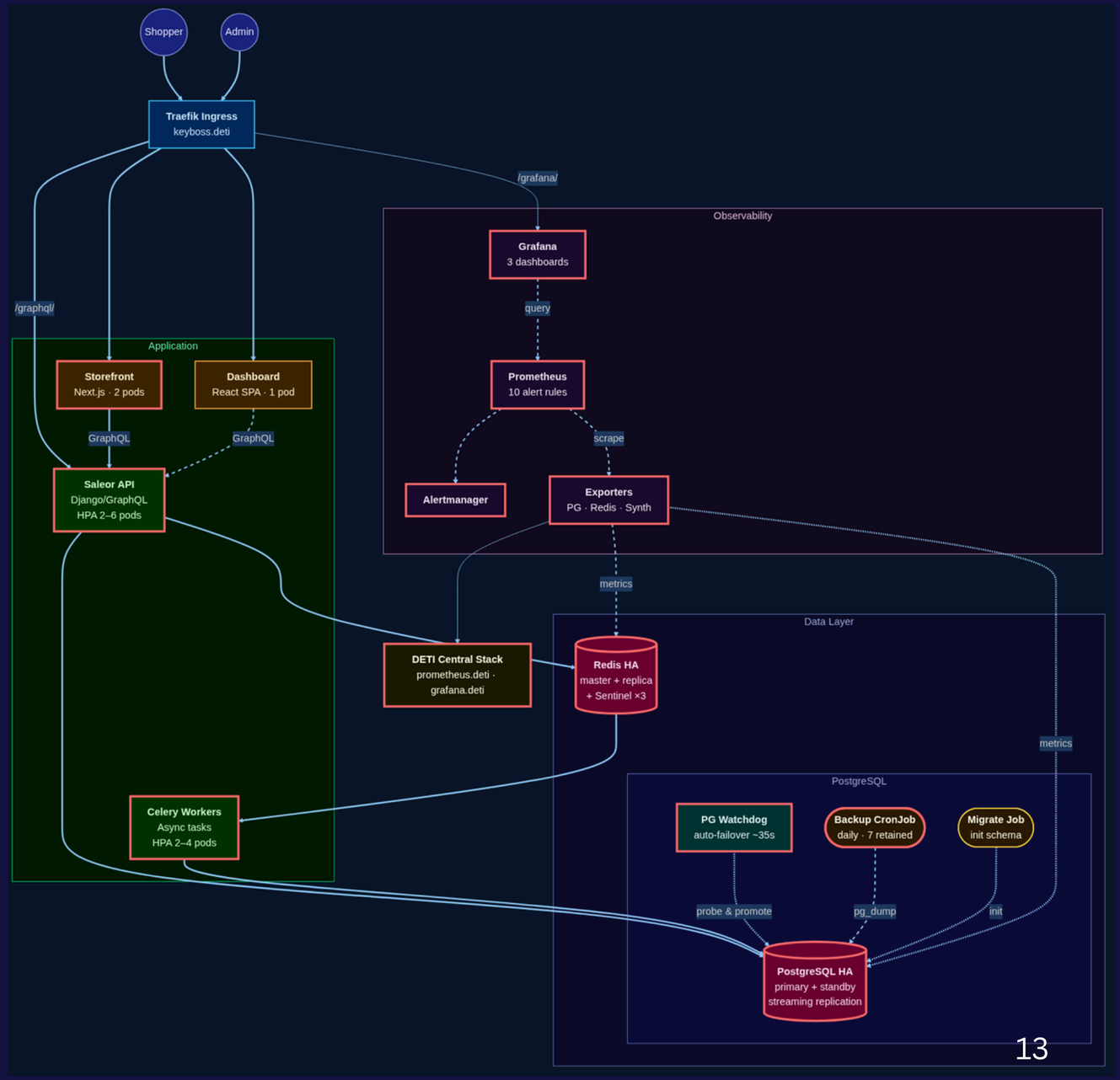
Redis is critical for KeyBoss. It is the **Celery broker** (game key delivery) and the **session cache**. Sentinel adds automatic failover with **no code changes** needed in the application.

- **3 Sentinels** → monitor the master independently and vote (quorum=2) to avoid false positives
- **Celery & API connect via Sentinel** → they always get redirected to the current master automatically
- **Init container handles reintegration** → when the old master comes back, it detects the new master and reconfigures itself as replica

Metric	Value	
RPO	< 1 s	✔
RTO	≈ 10 s	✔
Automatic	Yes	✔



ARCHITECTURE



HPA: AUTOSCALING

With fixed replicas, a traffic spike saturates the pods and degrades the service. HPA monitors CPU and memory continuously and adds replicas automatically when either threshold is exceeded (no manual intervention needed). Scale-down is delayed (5 min) to avoid oscillation under fluctuating load.

```
guilherme-silva@guilherme-silva:~/UNI/GIC/sre-project-grupo-keyboss$ kubectl get hpa -n tenant-keyboss
NAME                REFERENCE                    TARGETS                         MINPODS   MAXPODS   REPLICAS   AGE
celery-worker-hpa   Deployment/celery-worker   cpu: 0%/70%, memory: 50%/80%   2         4         2          12d
saleor-api-hpa      Deployment/saleor-api      cpu: 4%/70%, memory: 76%/80%   2         6         2          12d
```

On DETI →
saleor-api scaled naturally to 3 replicas due to memory pressure (84% > 80%):

```
saleor-api-hpa   cpu: 2%/70%  memory: 84%/80%   REPLICAS: 3
```

Pod anti-affinity: API pods spread across different nodes (no node holds more than 1 API pod).

Parameter	saleor-api	celery-worker
Min replicas	2	2
Max replicas	6	4
CPU Threshold	70%	70%
Scale-up cooldown	60 s	90 s
Scale-down cooldown	5 min	5 min

HPA: LOAD TEST EVIDENCE

Load test on DETI → ab against keyboss.deti/graphql/:

```
┌──(meowstermind㉿LEGION-LULU)-[~/Downloads/GIC/sre-project-grupo-keyboss]
└─$ echo '{"query":"{products(first:1,channel:\"default-channel\"){totalCount}}"}' > /tmp/gql.json

ab -n 5000 -c 50 -T 'application/json' -p /tmp/gql.json http://keyboss.deti/graphql/
Completed 2500 requests
Completed 3000 requests
Completed 3500 requests
Completed 4000 requests
Completed 4500 requests
apr_pollset_poll: The timeout specified has expired (70007)
Total of 4999 requests completed
```

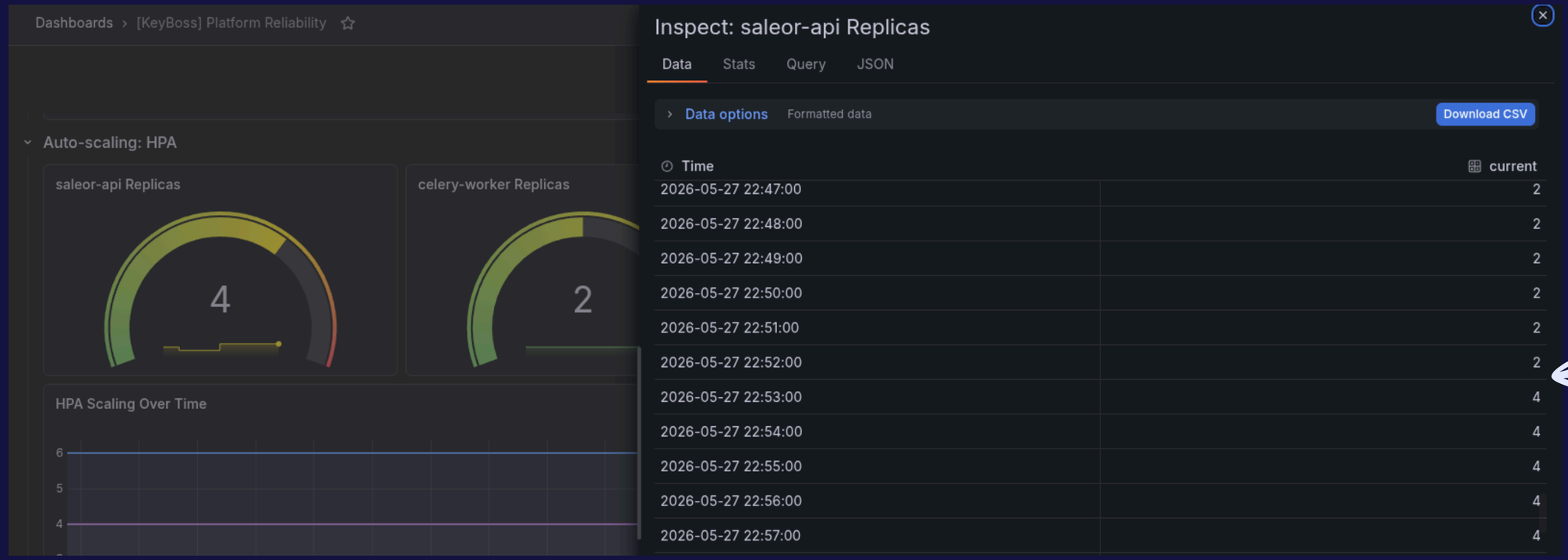
Result → HPA scaled 2 → 4 replicas in <60s:

```
export KUBECONFIG=~/Downloads/GIC/sre-project-grupo-keyboss/tenant-keyboss-kubeconfig.yaml
kubectl get hpa saleor-api-hpa -n tenant-keyboss --watchz
```

Moment	Replicas	CPU / Mem	
Before test	2	4% / 82%	
Peak load	4	73% / 83%	✔
Sustained load	4	77% / 83%	✔
After test	2	3% / 84%	

✔ 4999 requests completed
✔ Scale-up triggered by CPU 73% > 70%
✔ Zero application errors during test

HPA: LOAD TEST EVIDENCE



BACKUP AND DISASTER RECOVERY

01 **Daily CronJob at 02:00 UTC**

pg_dump → gzip compression → stored in PVC (1Gi) · 7-backup retention

02 **Restore tested on DETI**

gunzip dump.sql.gz | psql → 140 tables restored, 72KB

Component	RPO	RTO	Auto?
PostgreSQL (failover)	< 10 ms	~30s	✔
PostgreSQL (backup)	24 h	~5 min	Manual
Redis (failover)	< 1 ms	~10 s	✔
API / Workers	0	~15 s	✔ (HPA)

Why backup if HA replication?

HA copies everything, including bad migrations and accidental DELETEs. Backup is the only recovery for **logical errors**.

```
guilherme-silva@guilherme-silva:~/UNI/GIC/sre-project-grupo-keyboss$ KUBECONFIG=~/UNI/GIC/sre-project-grupo-keyboss/tenant-keyboss-kubeconfig.yaml   kubectl get pods -n tenant-keyboss
NAME                                  READY   STATUS      RESTARTS   AGE
alertmanager-7b58b9f487-rb54s         1/1     Running     0          14h
celery-worker-b58958c47-b7lpc         1/1     Running     0          7d2h
celery-worker-b58958c47-pxn5x         1/1     Running     0          7d2h
grafana-5586c85cc5-bbl52              1/1     Running     0          9m30s
kube-state-metrics-944c5b6b4-gxbvh    1/1     Running     0          15h
postgres-0                            1/1     Running     0          7d3h
postgres-backup-29646840-t4qmn        0/1     Completed   0          6d3h
postgres-backup-29648280-qgfsv        0/1     Completed   0          5d3h
postgres-backup-29649720-krsvb        0/1     Completed   0          4d3h
postgres-backup-29651160-qr4tb        0/1     Completed   0          3d3h
postgres-backup-29652600-9g9s7        0/1     Completed   0          2d3h
postgres-backup-29654040-fqzzf        0/1     Completed   0          27h
postgres-backup-29655480-bmscn        0/1     Completed   0          3h37m
prometheus-54cd5d59ff-ddpzh           1/1     Running     0          15h
redis-0                               1/1     Running     0          7d3h
saleor-api-796fc6ccd6-g224f           2/2     Running     0          14h
saleor-api-796fc6ccd6-g5hrj           2/2     Running     0          7h44m
saleor-api-796fc6ccd6-ggp5l           2/2     Running     0          14h
saleor-dashboard-76bd76898d-mmrn8     1/1     Running     0          14h
saleor-seed-jhqf7                     0/1     Completed   0          7d
saleor-storefront-8468f9768d-4lrkn    1/1     Running     0          7h44m
saleor-storefront-8468f9768d-xkdk4    1/1     Running     0          7h44m
```

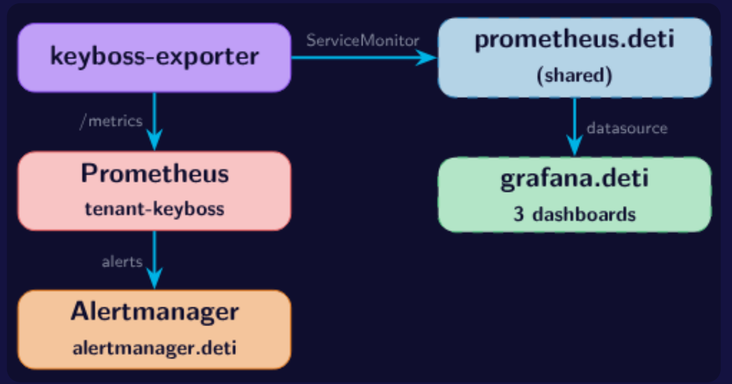
OBSERVABILITY:
MONITORING STACK

Components in tenant-keyboss

- kube-state-metrics: namespace-scoped via ksm-kubeconfig Secret
- KeyBoss Exporter: synthetic probes + app metrics
- ServiceMonitor: registers exporter with prometheus.deti
- Alertmanager deployed in tenant, exposed at alertmanager.deti via Ingress
- Prometheus in tenant scrapes internal metrics, fires alerts

Shared DETI infrastructure

- prometheus.deti → central, scrapes keyboss-exporter
- • grafana.deti → 3 KeyBoss dashboards
- • alertmanager.deti → shared routing



OBSERVABILITY:

BUSINESS & API HEALTH

http://grafana.deti/d/keyboss-business-health

grafana.deti → Business & API Health

- Endpoint availability: 5 endpoints UP/DOWN + product count
- SLO: availability % (5m); error budget remaining (1h)
- Latency: storefront + GraphQL p50/p95/p99
- Trends: availability over time; burn rate

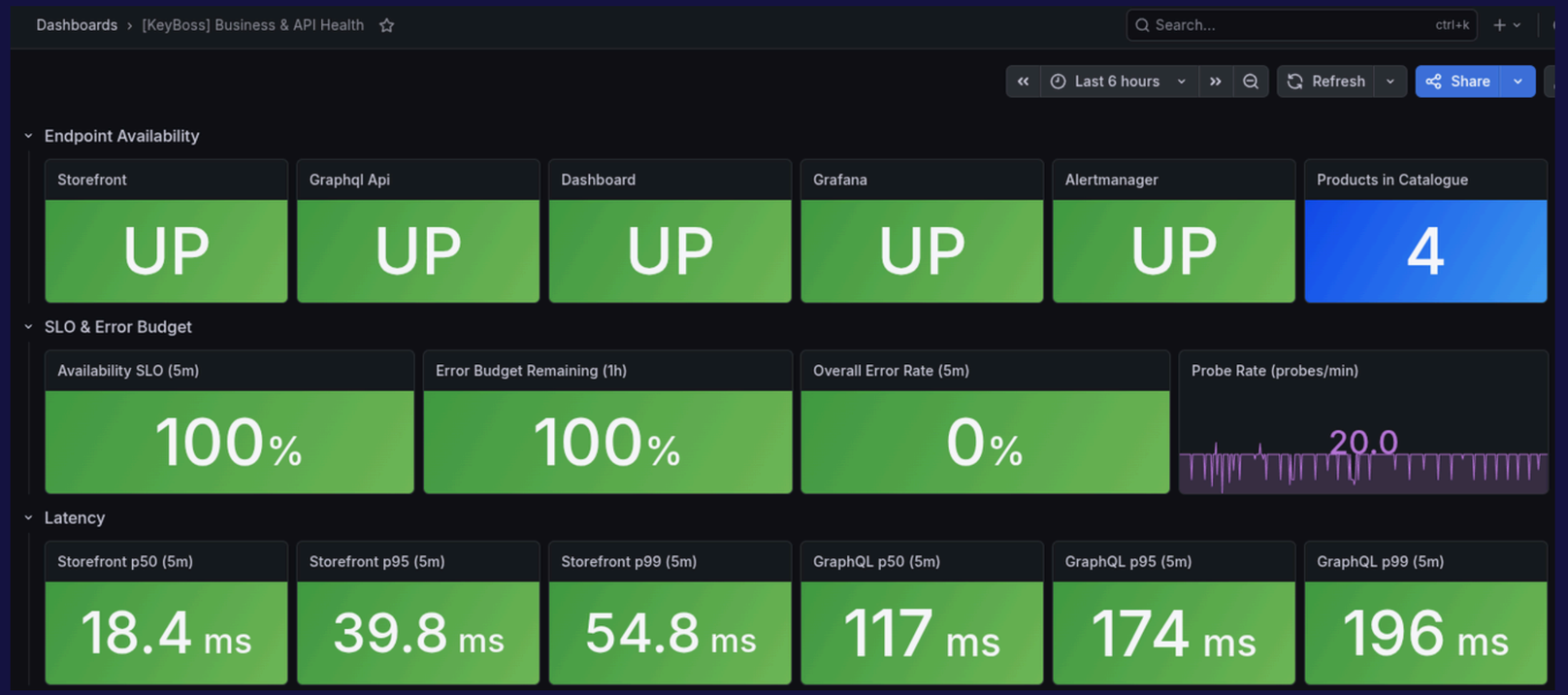
✔ Visible at http://prometheus.deti/targets
target: keyboss-exporter

✔ 100% availability · Error budget: 100% remaining
Storefront p50: 18.4ms · GraphQL p50: 117ms

OBSERVABILITY:

BUSINESS & API HEALTH

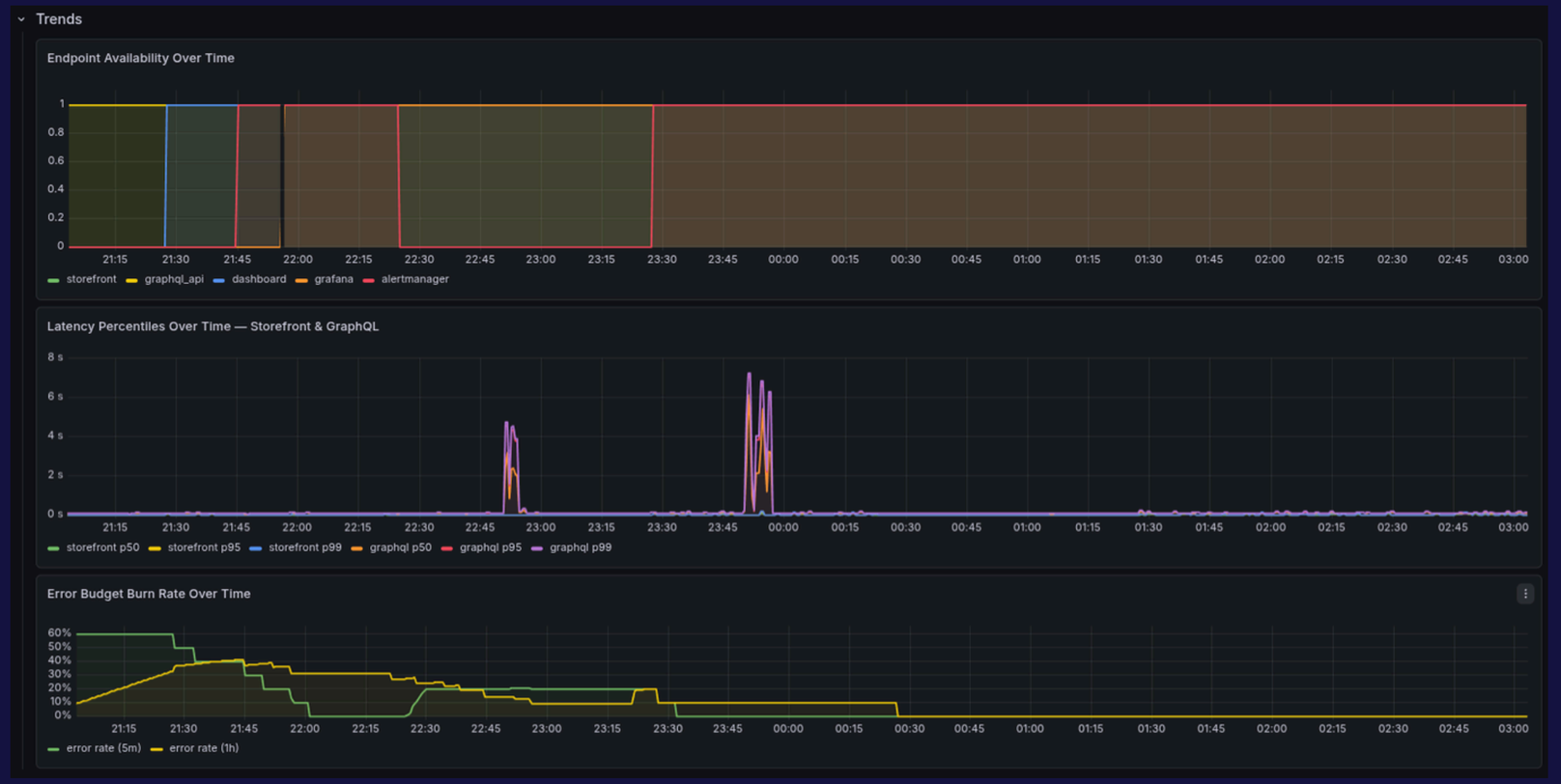
http://grafana.deti/d/keyboss-business-health



OBSERVABILITY:

BUSINESS & API HEALTH

http://grafana.deti/d/keyboss-business-health



OBSERVABILITY:
PLATFORM RELIABILITY

http://grafana.deti/d/keyboss-platform-reliability/

grafana.deti → Platform Reliability

- Data Layer: PostgreSQL 2/2 ready · Redis 2/2 ready
- Backup: last successful backup 4.72 min ago
- Watchdog: UP
- HPA: saleor-api 4 replicas · celery-worker 2
- Pod health: 24 pods Running · API 100% available

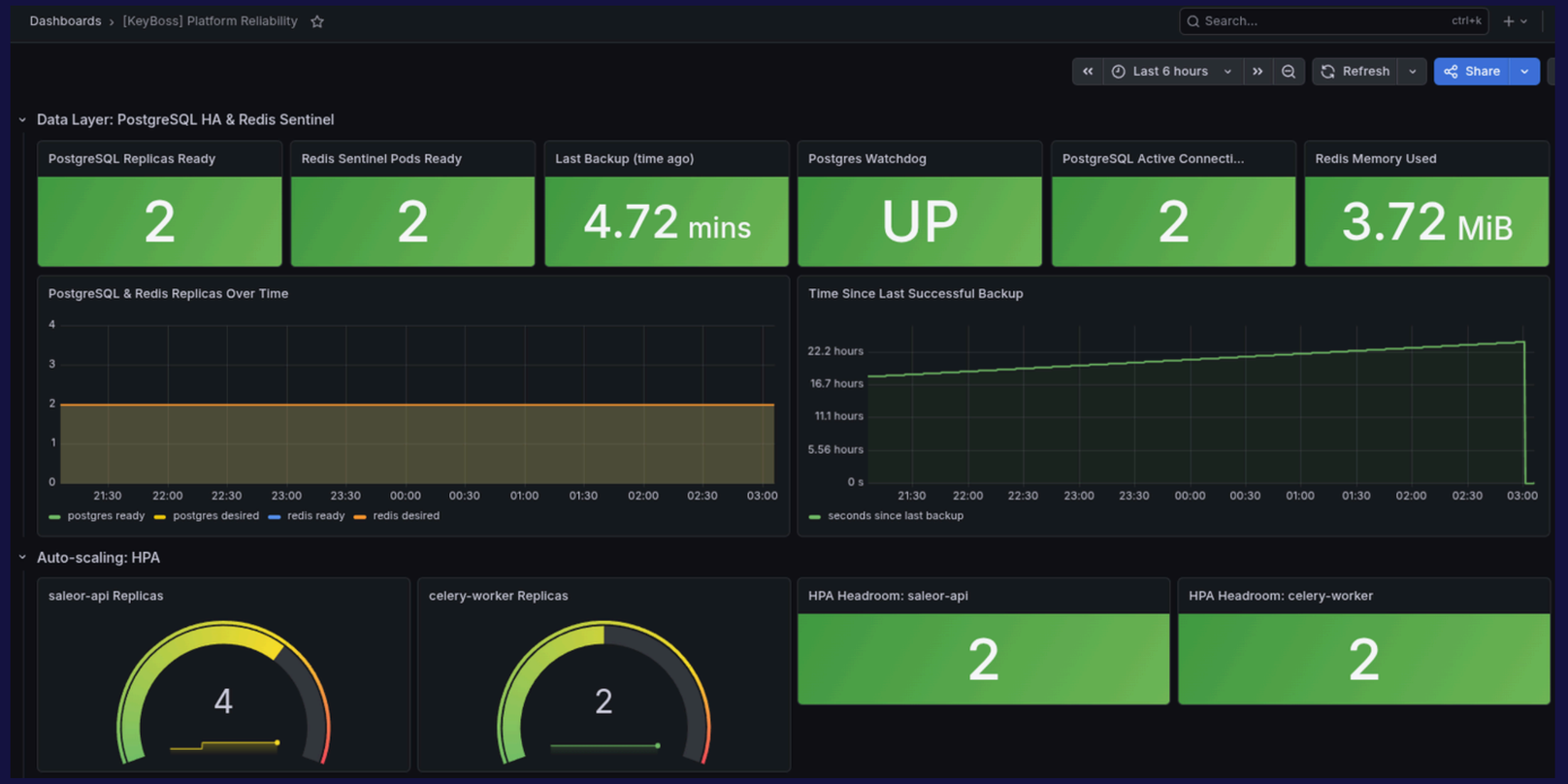
✔ Visible at http://prometheus.deti/targets
target: keyboss-exporter

✔ No single point of failure in data layer
HPA headroom: 2 slots free on each deployment

OBSERVABILITY:

PLATFORM RELIABILITY

http://grafana.deti/d/keyboss-platform-reliability/



OBSERVABILITY:

PLATFORM RELIABILITY

http://grafana.deti/d/keyboss-platform-reliability/

OBSERVABILITY:
POSTGRESQL & REDIS

http://grafana.deti/d/keyboss-infra/keyboss-postgresql-and-redis

grafana.deti → PostgreSQL & Redis

- postgres-exporter + redis-exporter deployed as ServiceMonitors
- PostgreSQL: role (PRIMARY/STANDBY), replication lag, connections, cache hit ratio, transaction rate
- Redis: connected clients, memory used, fragmentation ratio, keyspace hits/misses

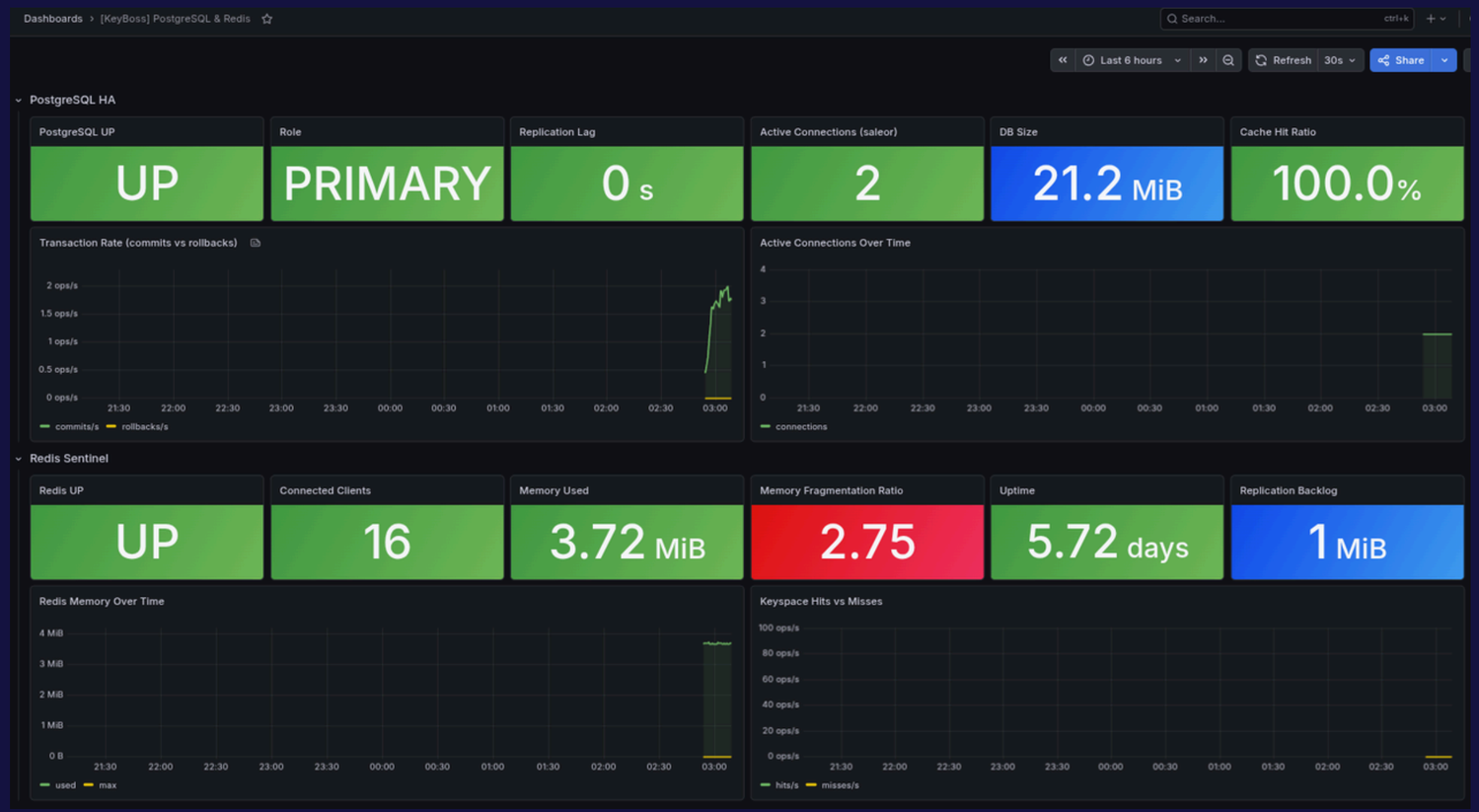
✔ Visible at http://prometheus.deti/targets
target: keyboss-exporter

✔ PRIMARY · Replication lag: 0s · Cache hit: 100%
Redis UP · 16 clients · 3.72 MiB used · 5.72 days uptime

OBSERVABILITY:

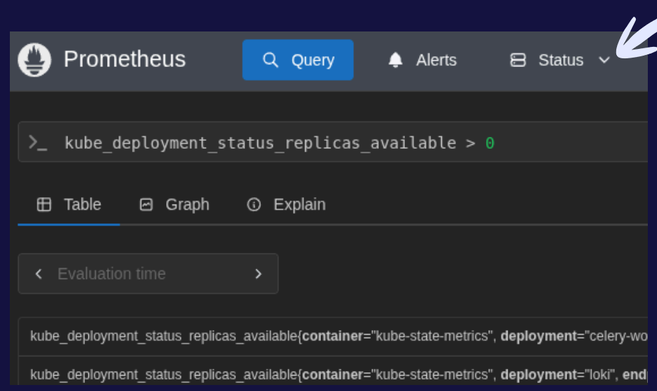
POSTGRESQL & REDIS

http://grafana.deti/d/keyboss-infra/keyboss-postgresql-and-redis



SLIS / SLOS

SLI	Prometheus Query	SLO
API Availability	kube_deployment_status_replicas_available > 0	99.5%
P99 Latency	histogram_quantile(0.99, traefik_duration...)	< 2s
Error Rate	5xx requests / total requests on /graphql/	< 1%
Workers Available	celery replicas > 0	99%



10 alert rules in 3 groups:

01 Availability (4)

SaleorAPIDown
CeleryWorkerDown
PostgreSQLDown
RedisDown

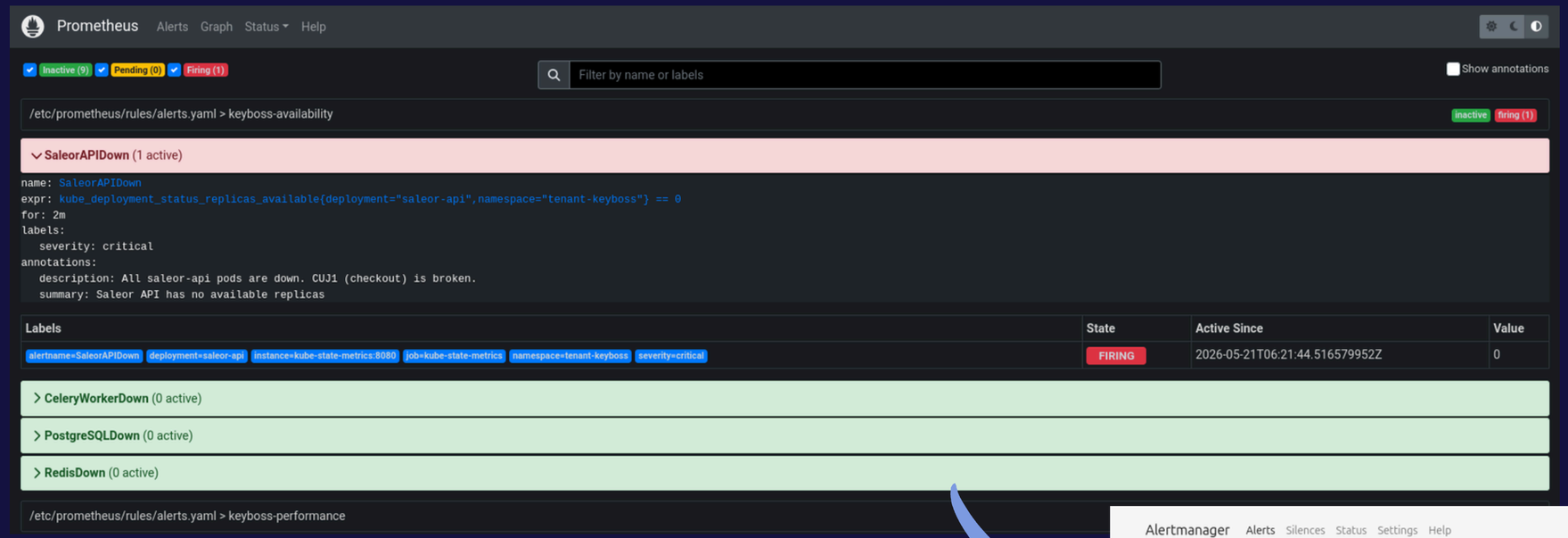
02 Performance (3)

HighAPIErrorRate
HighAPILatency
PodCrashLooping

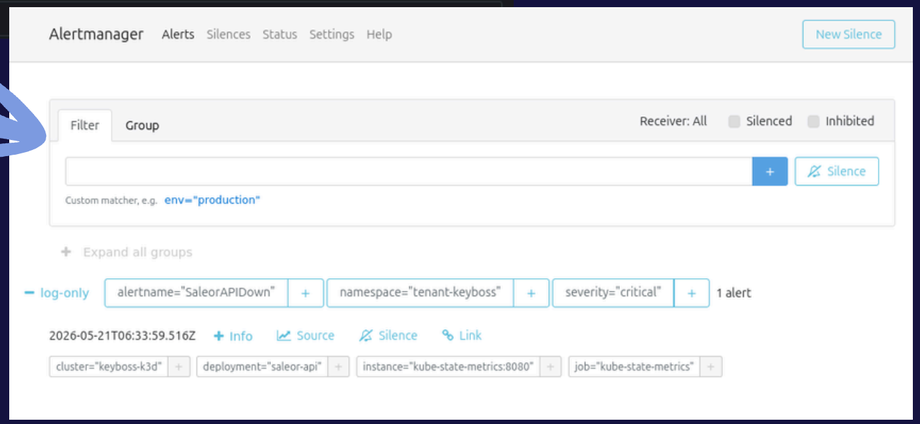
03 Saturation (3)

HighMemoryUsage
HighCPUUsage
HPAMaxedOut

ALERTS: EVIDENCE



Test: SaleorAPIDown alert pipeline on local k3d cluster



 # Scale to 0 to simulate outage

kubectl scale deploy / saleor - api -- replicas =0 -n tenant - keyboss

✔ **Full pipeline confirmed: Prometheus → Alertmanager**

for: 2m → only fires if the condition holds for 2 consecutive minutes, avoiding false positives on normal pod restarts

KEYBOSS METRICS

EXPORTER

Custom Python/Flask Exporter

Probes (every 15s)

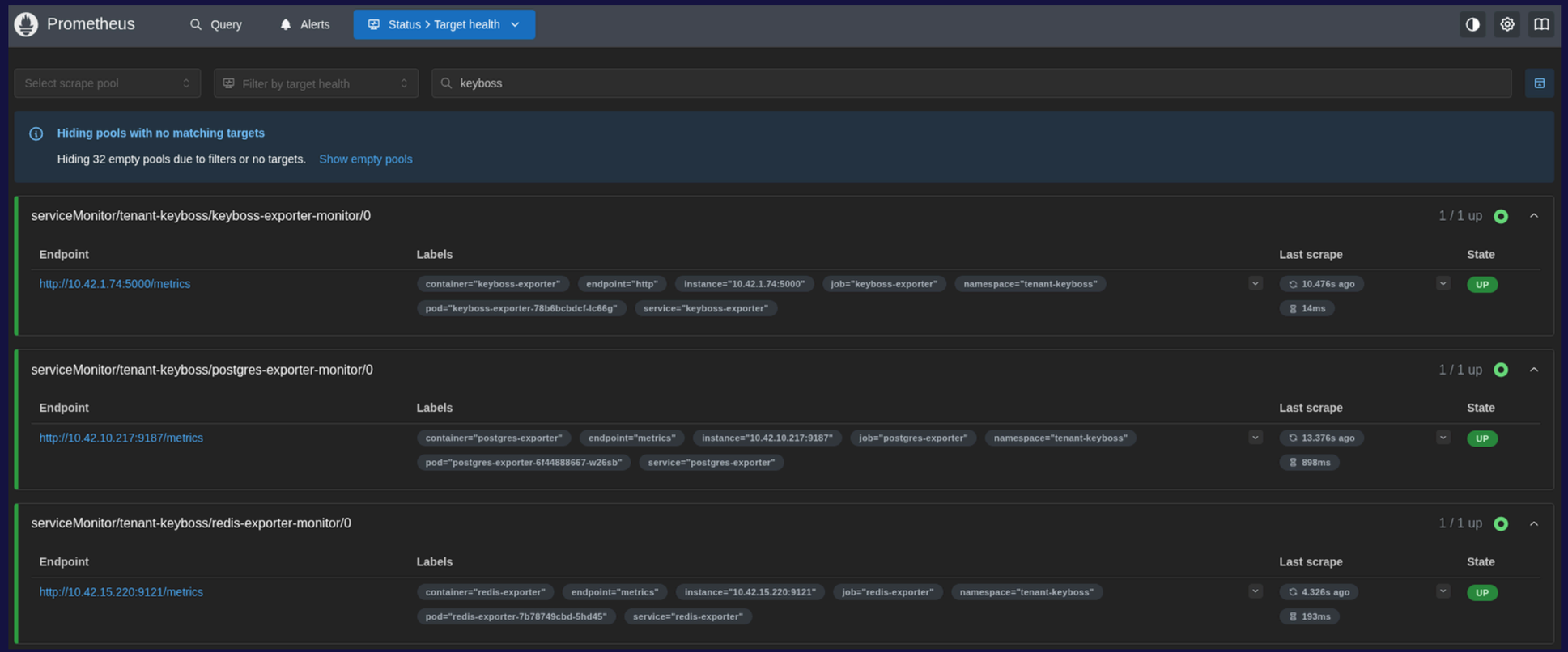
storefront	Saleor storefront (internal)
graphql_api	Saleor API /graphql/
dashboard	/dashboard/ (Ingress)
grafana	grafana.deti (shared)
alertmanager	alertmanager.deti (shared)

Exposed Metrics

Metric	Description
keyboss_probe_success{endpoint}	1 = endpoint up, 0 = down
keyboss_probe_duration_seconds{endpoint}	HTTP probe latency (s)
keyboss_probe_status_code{endpoint}	HTTP response code
keyboss_probe_runs_total{endpoint,result}	Probe counter (success/fail)
keyboss_graphql_up	GraphQL reachability (0/1)
keyboss_graphql_query_seconds	GraphQL query latency (s)
keyboss_catalogue_products_total	Product count in catalogue
keyboss_build_info	Exporter version and env

KEYBOSS METRICS

EXPORTER



KNOWN LIMITATIONS

PostgreSQL watchdog → single pod

f watchdog and primary fail simultaneously, failover is delayed until watchdog restarts.

Mitigation: liveness probe restarts quickly.
Future: 2 watchdog replicas + leader election.

Redis Cluster (sharding) not implemented

Current: master + replica, HA but no horizontal write scalability.

Current load: < 500 concurrent users, acceptable.
Future: Redis Cluster with 3+ shards.

Backup → no geo-redundancy

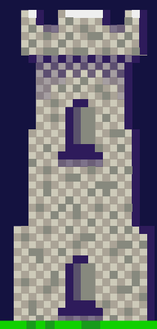
Backups stored in same DETI cluster PVC. If cluster is lost, backups are lost too.

Future: Export to external S3/MinIO.

HPA → no queue-depth autoscaling

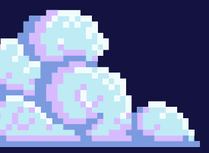
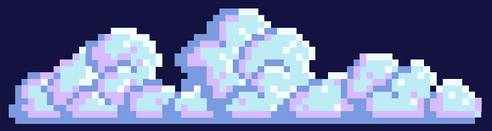
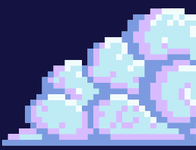
HPA scales on CPU/memory only. Celery queue depth is not a metric. A large backlog under low CPU will not trigger scale-up.

Future: KEDA with Redis queue length metric.



CONCLUSION

- **No single point of failure**
  PostgreSQL and Redis both recover automatically from failures. No human intervention, RTO < 35s.
- **No manual deployments**
  One script, two environments (--local / --deti), automatic rollback on any failure.
- **No blind spots**
  10 alerts, custom metrics exporter, 3 Grafana dashboards integrated with prometheus.deti.
- **No fixed capacity**
  HPA scales with real load, demonstrated in production: 2 → 4 replicas under load test.

deti
universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

THANK YOU!

Carolina Reis | nº 131193
Guilherme Silva | nº 131143

Computer Infrastructure Management

28 May 2026


REFERENCES

[1] Saleor Commerce, "Commerce Infrastructure. Made-to-Measure for You.", saleor.io, 2026. [Online]. Available: https://saleor.io/

[2] StackShare, "Overview", stackshare.io, 2026. [Online]. Available: https://stackshare.io/saleor

[3] O. Rodrigues, "Awesome-Open-Source-eCommerce-Platforms", GitHub, 2026. [Online]. Available: https://github.com/olivrg/Awesome-Open-Source-eCommerce-Platforms?tab=readme-ov-file#python

[4] Kubernetes Documentation, "Documentation", kubernetes.io, 2026. [Online]. Available: https://kubernetes.io/docs/

[5] k3d Documentation, "Overview", k3d.io, 2026. [Online]. Available: https://k3d.io/

[6] k3s Documentation, "Networking Services", docs.k3s.io, 2026. [Online]. Available: https://docs.k3s.io/networking/networking-services

[7] Saleor Commerce Documentation, "Saleor Documentation", docs.saleor.io, 2026. [Online]. Available: https://docs.saleor.io/

REFERENCES

[8] PostgreSQL Global Development Group, "Chapter 26. High Availability, Load Balancing, and Replication", postgresql.org, 2026. [Online]. Available: https://www.postgresql.org/docs/current/high-availability.html

[9] PostgreSQL Global Development Group, "9.28. System Administration Functions", postgresql.org, 2026. [Online]. Available: https://www.postgresql.org/docs/current/functions-admin.html#FUNCTIONS-RECOVERY-CONTROL

[10] Redis Ltd., "High availability with Redis Sentinel", redis.io, 2026. [Online]. Available: https://redis.io/docs/latest/operate/oss_and_stack/management/sentinel/

[11] Kubernetes Documentation, "Horizontal Pod Autoscaling", kubernetes.io, 2026. [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[12] Prometheus Authors, "Overview", prometheus.io, 2026. [Online]. Available: https://prometheus.io/docs/introduction/overview/

[13] Prometheus Authors, "Alerting Rules", prometheus.io, 2026. [Online]. Available: https://prometheus.io/docs/prometheus/latest/configuration/alerting_rules/

REFERENCES

[14] Prometheus Authors, "Alertmanager", prometheus.io, 2026. [Online]. Available: https://prometheus.io/docs/alerting/latest/alertmanager/

[15] Grafana Labs, "Grafana documentation", grafana.com, 2026. [Online]. Available: https://grafana.com/docs/grafana/latest/

[16] Prometheus Operator Authors, "API reference", prometheus-operator.dev, 2026. [Online]. Available: https://prometheus-operator.dev/docs/api-reference/api/#monitoring.coreos.com/v1.ServiceMonitor

[17] Google Site Reliability Engineering, "Service Level Objectives", sre.google, 2026. [Online]. Available: https://sre.google/sre-book/service-level-objectives/

[18] PostgreSQL Global Development Group, "pg_dump", postgresql.org, 2026. [Online]. Available: https://www.postgresql.org/docs/current/app-pgdump.html

[19] Kubernetes Documentation, "CronJob", kubernetes.io, 2026. [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/