Spot VMs and Postgres

Kaarel Moppel

Freelance PostgreSQL Consultant

pgug.ee #8

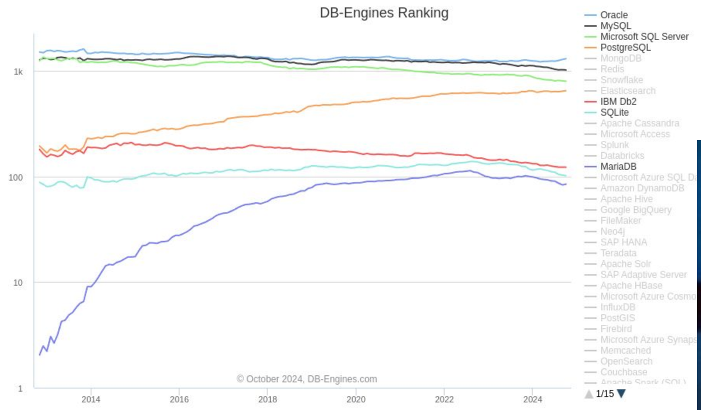
$ whoami

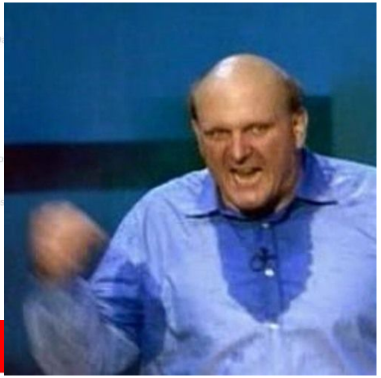
- Full-time “wrestling” with databases since 2007
- 20K+ hours in the Postgres ecosystem
- Have developed a pretty good gut feeling of what works reasonably well for some purpose and what not
- Up for:
    - Performance troubleshooting & tuning
    - HA setups / replication
    - Security / operational auditing
    - Backup / recovery procedures
    - Monitoring
    - Migrations
    - Trainings
    - Tooling development

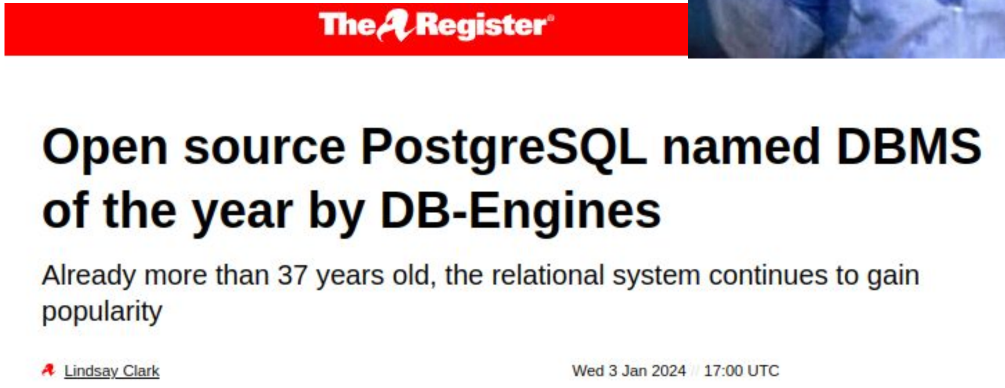
Talking points

- Deploying Postgres
- Spot VMs intro
- Managed vs K8s vs self-rolled
- Deployment options and considerations
- An example Postgres-specific Spot implementation

Postgres, postgres, postgres





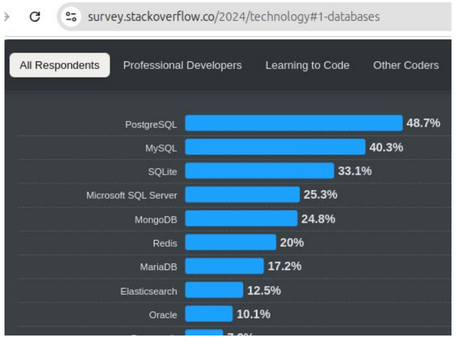


The Register

Open source PostgreSQL named DBMS of the year by DB-Engines

Already more than 37 years old, the relational system continues to gain popularity

Lindsay Clark Wed 3 Jan 2024 // 17:00 UTC



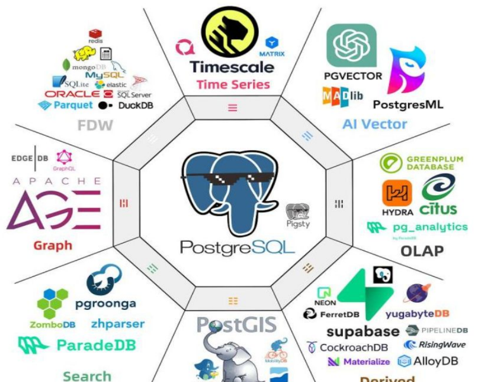
survey.stackoverflow.co/2024/technology#1-databases

All Respondents | Professional Developers | Learning to Code | Other Coders

PostgreSQL	48.7%
MySQL	40.3%
SQLite	33.1%
Microsoft SQL Server	25.3%
MongoDB	24.8%
Redis	20%
MariaDB	17.2%
Elasticsearch	12.5%
Oracle	10.1%

Peter Zaitsev • 1st
Entrepreneur | Driving Success with MySQL, MariaDB, MongoDB & Postgr…
3d •

"Postgres is eating the database world" - Do you agree ?
https://lnkd.in/ecUyy5Rj #postgres



Common Postgres deployment options

- **Local** Docker / testcontainers
    - The best way if persistence not needed
- Instance **sharing** - can work well up to a point (no security or perf considerations)
- A fully **managed** cloud service a la Amazon RDS or similar
    - Tooling support is generally pretty nice
- **K8s** - works pretty well nowadays
    - A lot of choice though. Might actually want* a support contract
    - Need to watch out perf-wise on fully managed K8s
- **Serverless**? Definitely an upcoming thing! (Neon, Xata, Aurora Serverless, …)
    - Allows coolness like data branching + HTTP access among other things
- Rolling **your own** 🥳

When to use what

Due to vast differences in exact objectives, data size, criticality, branch duration, etc…it’s pretty much impossible to derive a rule of thumb :/

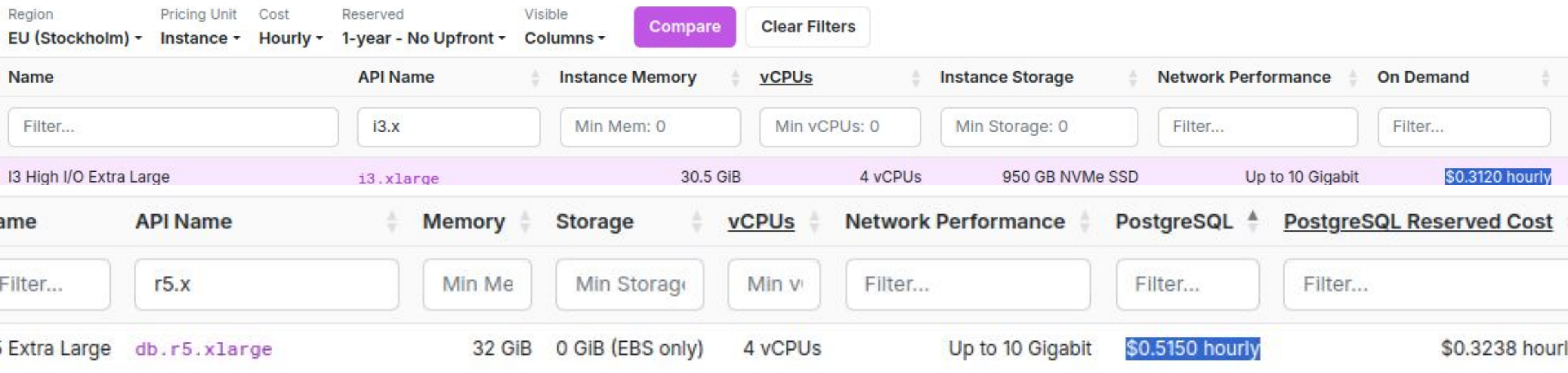
Some recommendations though can be made when optimizing for one key aspect only:

- **Resource utilization / cost** > K8s
- **Standardization** / **deployability** > K8s
- **Criticality** > Fully managed
- **Short lifetime** > Serverless
- **Customization** / **exotic extensions*** > Self-managed
- **Performance** > Self-managed or Fully managed on autoscale ($$$)

A cost / performance example

Managed RDS vs **self-managed** for this 4 CPU / 32 GB RAM configuration has a ~**4x “bang-for-buck” difference!** Thanks to much faster locally attached (yes, volatile) disks…

* Tested: a standard (pgbench) random read-only use case on a 200GB dataset, 10K TPS vs 20K TPS, price $456 vs $224.



Region: EU (Stockholm) · Pricing Unit: Instance · Cost: Hourly · Reserved: 1-year - No Upfront · Visible: Columns

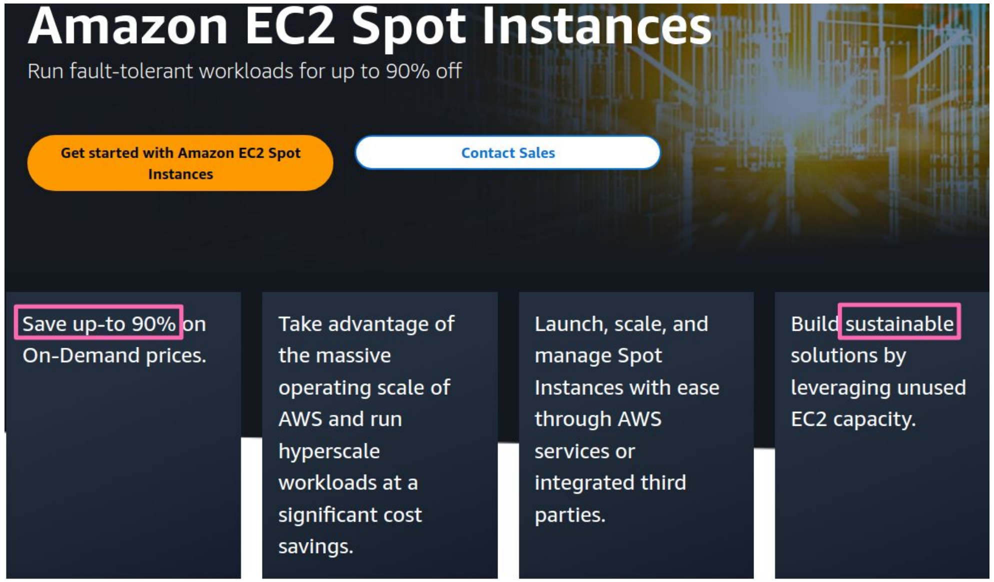
Name	API Name	Instance Memory	vCPUs	Instance Storage	Network Performance	On Demand
Filter...	i3.x	Min Mem: 0	Min vCPUs: 0	Min Storage: 0	Filter...	Filter...
I3 High I/O Extra Large	i3.xlarge	30.5 GiB	4 vCPUs	950 GB NVMe SSD	Up to 10 Gigabit	$0.3120 hourly

ame	API Name	Memory	Storage	vCPUs	Network Performance	PostgreSQL	PostgreSQL Reserved Cost
Filter...	r5.x	Min Me	Min Storag	Min v	Filter...	Filter...	Filter...
Extra Large	db.r5.xlarge	32 GiB	0 GiB (EBS only)	4 vCPUs	Up to 10 Gigabit	$0.5150 hourly	$0.3238 hourl

Spot instances!

- **Idling** compute capacity
    - Supported on all top 3 clouds*
- **Low cost**
    - On average 3-5x savings compared to On-Demand
    - 3yr RIs only give ~2x
- **No guarantees obviously**
    - Can capture a 30s termination notice though before the rugpull
- **Dynamic pricing** - fluctuates based on demand
    - Huge differences between regions and even AZs!
- Best for **non-critical** or **fault-tolerant applications**
    - Web crawling, ETL jobs, queue processing, CI/CD, ...
- “Some” **eviction rate data available** to utilize Spot more succesfully

Massive cost-savings potential



Amazon EC2 Spot Instances

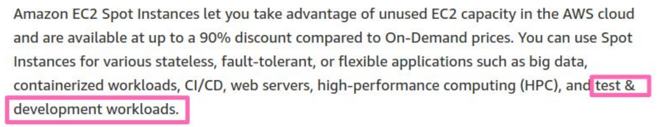
Run fault-tolerant workloads for up to 90% off

Save up-to 90% on On-Demand prices.

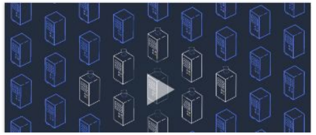
Take advantage of the massive operating scale of AWS and run hyperscale workloads at a significant cost savings.

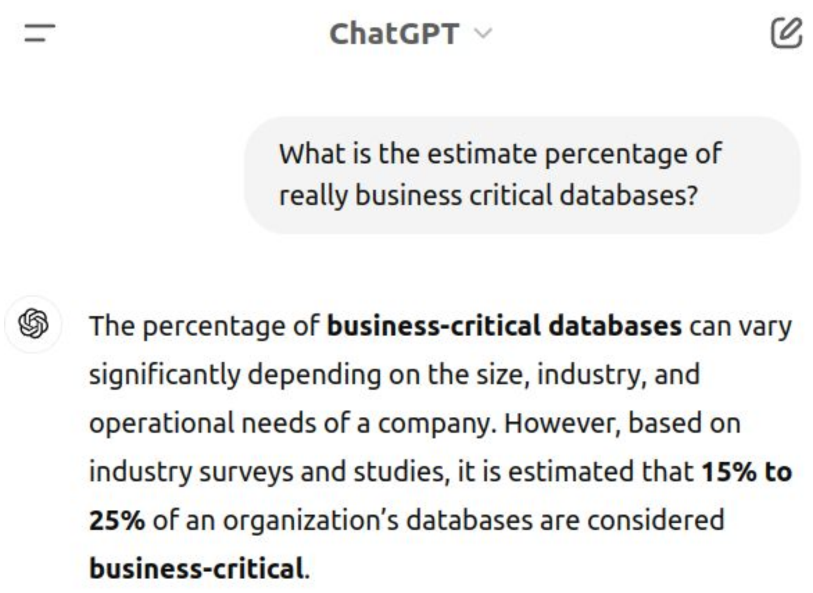
Launch, scale, and manage Spot Instances with ease through AWS services or integrated third parties.

Build sustainable solutions by leveraging unused EC2 capacity.



Amazon EC2 Spot Instances let you take advantage of unused EC2 capacity in the AWS cloud and are available at up to a 90% discount compared to On-Demand prices. You can use Spot Instances for various stateless, fault-tolerant, or flexible applications such as big data, containerized workloads, CI/CD, web servers, high-performance computing (HPC), and test & development workloads.





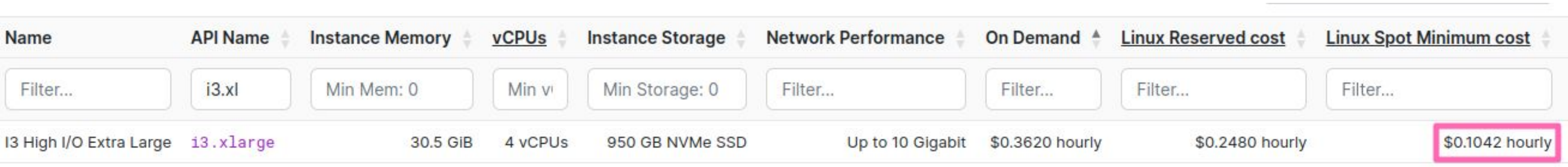
What is the estimate percentage of really business critical databases?

The percentage of **business-critical databases** can vary significantly depending on the size, industry, and operational needs of a company. However, based on industry surveys and studies, it is estimated that **15% to 25%** of an organization's databases are considered **business-critical**.

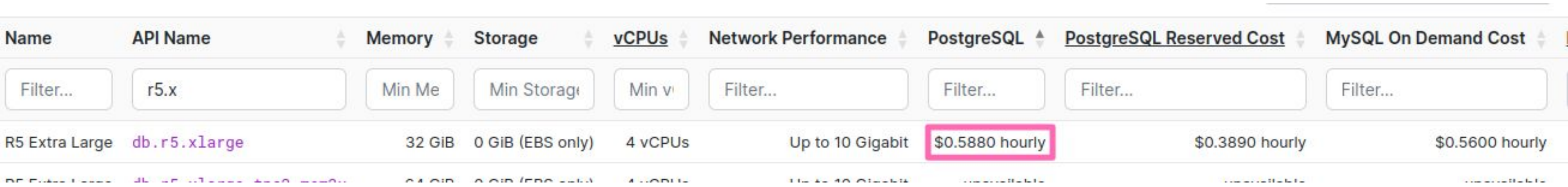
A cost / performance example v2 - with Spot

Managed RDS vs self-managed **Spot VM** for a 4 CPU / 32 GB RAM configuration has a ~**10x “bang-for-buck” difference!**

The savings potential for larger companies with dozens of DBs* is just huge!



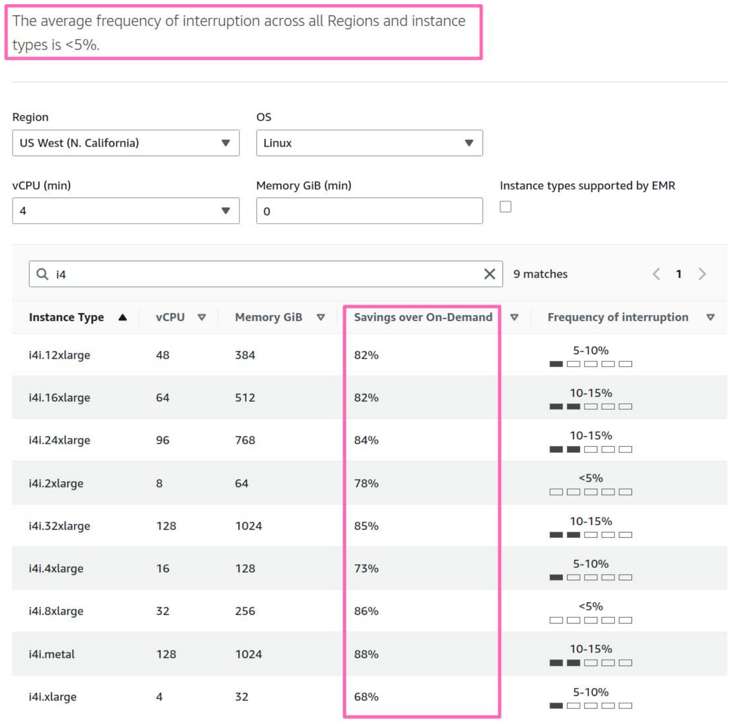
Name	API Name	Instance Memory	vCPUs	Instance Storage	Network Performance	On Demand	Linux Reserved cost	Linux Spot Minimum cost
Filter...	i3.xl	Min Mem: 0	Min v	Min Storage: 0	Filter...	Filter...	Filter...	Filter...
I3 High I/O Extra Large	i3.xlarge	30.5 GiB	4 vCPUs	950 GB NVMe SSD	Up to 10 Gigabit	$0.3620 hourly	$0.2480 hourly	$0.1042 hourly



Name	API Name	Memory	Storage	vCPUs	Network Performance	PostgreSQL	PostgreSQL Reserved Cost	MySQL On Demand Cost
Filter...	r5.x	Min Me	Min Storage	Min v	Filter...	Filter...	Filter...	Filter...
R5 Extra Large	db.r5.xlarge	32 GiB	0 GiB (EBS only)	4 vCPUs	Up to 10 Gigabit	$0.5880 hourly	$0.3890 hourly	$0.5600 hourly
R5 Extra Large	db.r5.xlarge.tpc2.mem2x	64 GiB	0 GiB (EBS only)	4 vCPUs	Up to 10 Gigabit	unavailable	unavailable	unavailable

Spot is actually not “that” scary

If to use the Spot Instance advisor tool by AWS



The average frequency of interruption across all Regions and instance types is <5%.

Region: US West (N. California)

OS: Linux

vCPU (min): 4

Memory GiB (min): 0

Instance types supported by EMR ☐

i4 — 9 matches — 1

Instance Type	vCPU	Memory GiB	Savings over On-Demand	Frequency of interruption
i4i.12xlarge	48	384	82%	5-10%
i4i.16xlarge	64	512	82%	10-15%
i4i.24xlarge	96	768	84%	10-15%
i4i.2xlarge	8	64	78%	<5%
i4i.32xlarge	128	1024	85%	10-15%
i4i.4xlarge	16	128	73%	5-10%
i4i.8xlarge	32	256	86%	<5%
i4i.metal	128	1024	88%	10-15%
i4i.xlarge	4	32	68%	5-10%

Meaning - **on average, one can expect to run a few months uninterrupted!**

Spot for DBs?

- Based on my experience from the past years - **works surprisingly well!**
  - For short-termish (some months) or non-critical use cases at least...
- My general recommendations learnt along the way:
  - Use the eviction rate data
  - Prefer instance storage for incredible perf per dollar
  - Be flexible with regions and AZs
  - Avoid the lowest SKUs and burstable instance types
  - Time-box everything, retry if something takes suspiciously long
  - Watch out for defaults - e.g. getting a public IP, GCP “Preemptible”
  - No need to specify a max price anymore

Spot for DBs - deployment options

- Managed K8s (EKS, GKE, AKS) with Spot node groups
  - If to hand-pick instance types and add some extra configuration to make poper use of instance storage
- Managed ECS with Spot
  - Very limited storage options
- Self-managed
  - Docker
  - Custom AMIs with batteries included
  - Standard VM-style via Ansible etc

Spot for DBs - a K8s example

- All 3 clouds allow running Spot node pools, à la:

```
$ eksctl create cluster --spot --instance-types=c3.large,c4.large,c5.large
```

- A solid option for small to medium DBs, given some Postgres is always running and if to hand-pick the instances types and also keep them updated!
  - As prices and eviction rates are always changing …

Self-managing Spot VMs?

PROS

- Zero up-front $$ investment
- Minimal up-front setup, VMs are low-level building blocks
- Unbeatable savings (managed K8s fixed to a region)
- Perfect hardware isolation and matching for the given task, every time
  - K8s + Karpenter can work to an extent also here
- No K8s knowledge needed

CONS

- Longer start / recovery times (similar to non-HA RDS though)
  - Resolve HW, check prices, wait for boot, install packages*
- No K8s
  - The custom glue can get messy of course…
  - Security side needs separate setup / review

Self-managed Spot VMs in practice

Wouldn’t it be nice though if someone else deals with the annoying details?

1. Looks for the cheapest VMs matching our requirements
2. Sets up tuned Postgres, users, databases, extensions and just gives us the connect string
3. Auto-discards the instance in x minutes/hours

From my own needs and experimenting grew out something like…

PG Spot Operator - available on PyPI + Docker

```
psql "$(pg_spot_operator  --region=eu-north-1  --ram-min=64  --storage-min=500 \
  --storage-type=local --tuning-profile=analytics   --instance-name=mypg1  \
  --admin-user=pgspotops  --admin-user-password=topsecret123   --connstr-output-only)"
...
INFO Current Spot discount rate in AZ eu-north-1a: -75.5% (spot $126.6 vs on-demand $516.2)
...
psql (16.4 (Ubuntu 16.4-1.pgdg24.04+2))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off)
Type "help" for help.
pgspotops@postgres=>
```

~6x savings compared to RDS!

** Assumes a local AWS CLI setup*

PG Spot Operator highlights

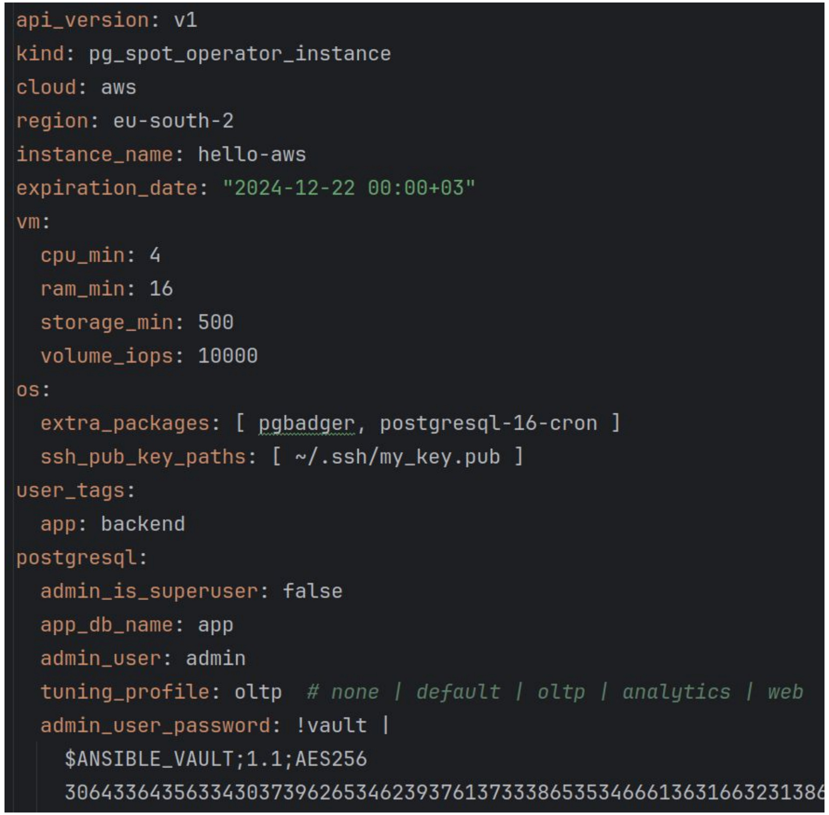
- One-liner Postgres at unbeatable price / performance
- Human-friendly HW specification
- Doesn’t rely on any other company infra by design
- Also for long-term workloads if 99.99% class uptime not needed
- Shines especially with data heavier workloads
- Can be used for anything really in --vm-only / --connstr-output-only mode
- A more polished commercial version in designing
    - Hybrid provisioning - use Spot only when DT budget allows
    - All top 3 clouds
    - Super-regions for even more savings
    - …

“UI” - CLI / Docker params or a “manifest”

```
# --check-price doesn’t need AWS creds!

pg_spot_operator \
  --check-price \
  --region eu-north-1 \
  --ram-min 128

docker run --rm \
  -e PGSO_CHECK_PRICE=y \
  -e PGSO_REGION=eu-west-1 \
  -e PGSO_RAM_MIN=128 \
  -e PGSO_STORAGE_TYPE=local \
  -e PGSO_STORAGE_MIN=200 \
  pgspotops/pg-spot-operator:latest
```



```yaml
api_version: v1
kind: pg_spot_operator_instance
cloud: aws
region: eu-south-2
instance_name: hello-aws
expiration_date: "2024-12-22 00:00+03"
vm:
  cpu_min: 4
  ram_min: 16
  storage_min: 500
  volume_iops: 10000
os:
  extra_packages: [ pgbadger, postgresql-16-cron ]
  ssh_pub_key_paths: [ ~/.ssh/my_key.pub ]
user_tags:
  app: backend
postgresql:
  admin_is_superuser: false
  app_db_name: app
  admin_user: admin
  tuning_profile: oltp  # none | default | oltp | analytics | web
  admin_user_password: !vault |
    $ANSIBLE_VAULT;1.1;AES256
    30643364356334303739626534623937613733386535346661363166323138
```

The codes

github.com/pg-spot-ops/pg-spot-operator

Licence: Functional Source License, Version 1.1, Apache 2.0 Future License

Status: Working Beta - any kind of feedback (or just a ⭐) very much appreciated! 🙏

Thank you!

SLIDES

kaarel.moppel@gmail.com

https://www.linkedin.com/in/kaarelmoppel/

https://kmoppel.github.io/