Red Hat
Enterprise Linux

Developers Messing Up with Data in Cloud

Dos and Don'ts

Honza Horak
Senior Manager, Red Hat, Brno

March 31, 2023
Brno

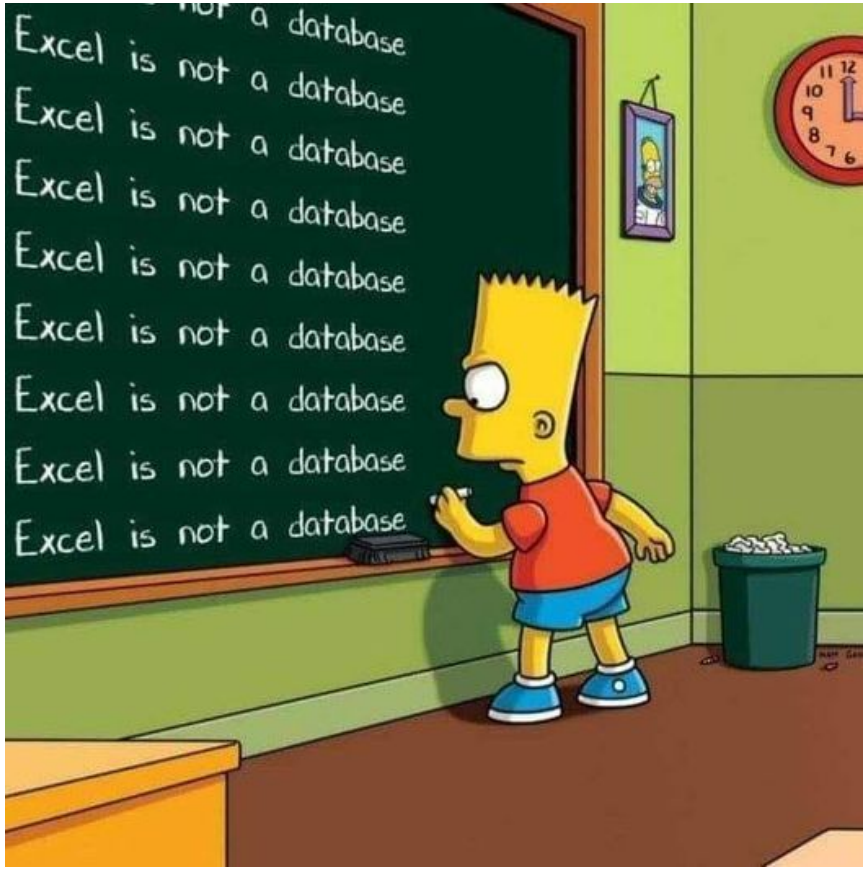
Red Hat

What actually is a database?

...a magic box?

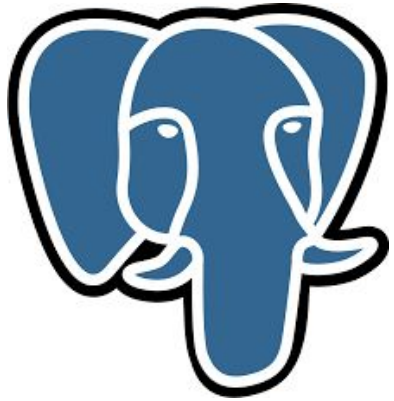
From a developer perspective, it's just a clever program that stores data permanently

- Somewhere
- Somehow



Red Hat

Despite my unhidden sympathy to PostgreSQL, this talk is purposefully DB-engine agnostic.



Big data and cloud

Big data primarily refers to data sets that are too large or complex to be dealt with by traditional data-processing application software.

...

Big data was originally associated with three key concepts: **volume, variety, and velocity**.

(Wikipedia)

Real examples of what is big

and what not actually

Small data

- Relevant portion of data still fits the RAM
- OLTP, not OLAP

Examples:

- Czech National Library
- Some no-named big e-shop

Big data

- Not structured/consistent
- Working with larger portions of data
- OLAP

Examples:

- An event in Large Hadron Collider
- Storing all clicks on <social_network>

Picking a correct tool

No database is the best. Not even PostgreSQL. Concrete use case matters.

- What data do I need to store
- What applications will work with the data and how
- Budget
- Restrictions
- One size does not fit all

Scaling a database

Why one would like to scale a database?

Sure, here's a short list of reasons why someone may want to scale a database:

1. Handling increased workload
2. Improving performance
3. Ensuring availability
4. Meeting business needs
5. Cost efficiency

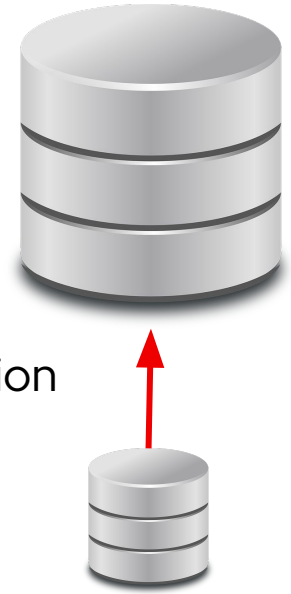
Red Hat

Scaling a database

Up or out? Maybe both.

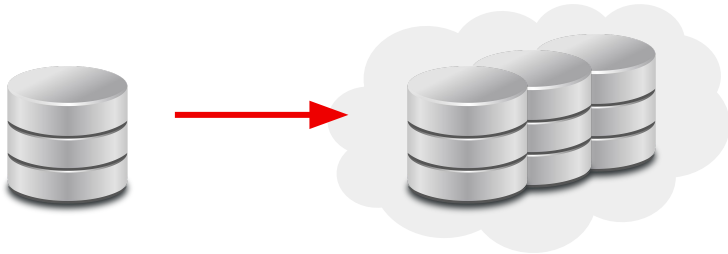
Vertical scaling (scale-up)

- How:
  - More CPU
  - More RAM
  - More storage
- (+) Transparent for the application
- (-) Limits of DB engine

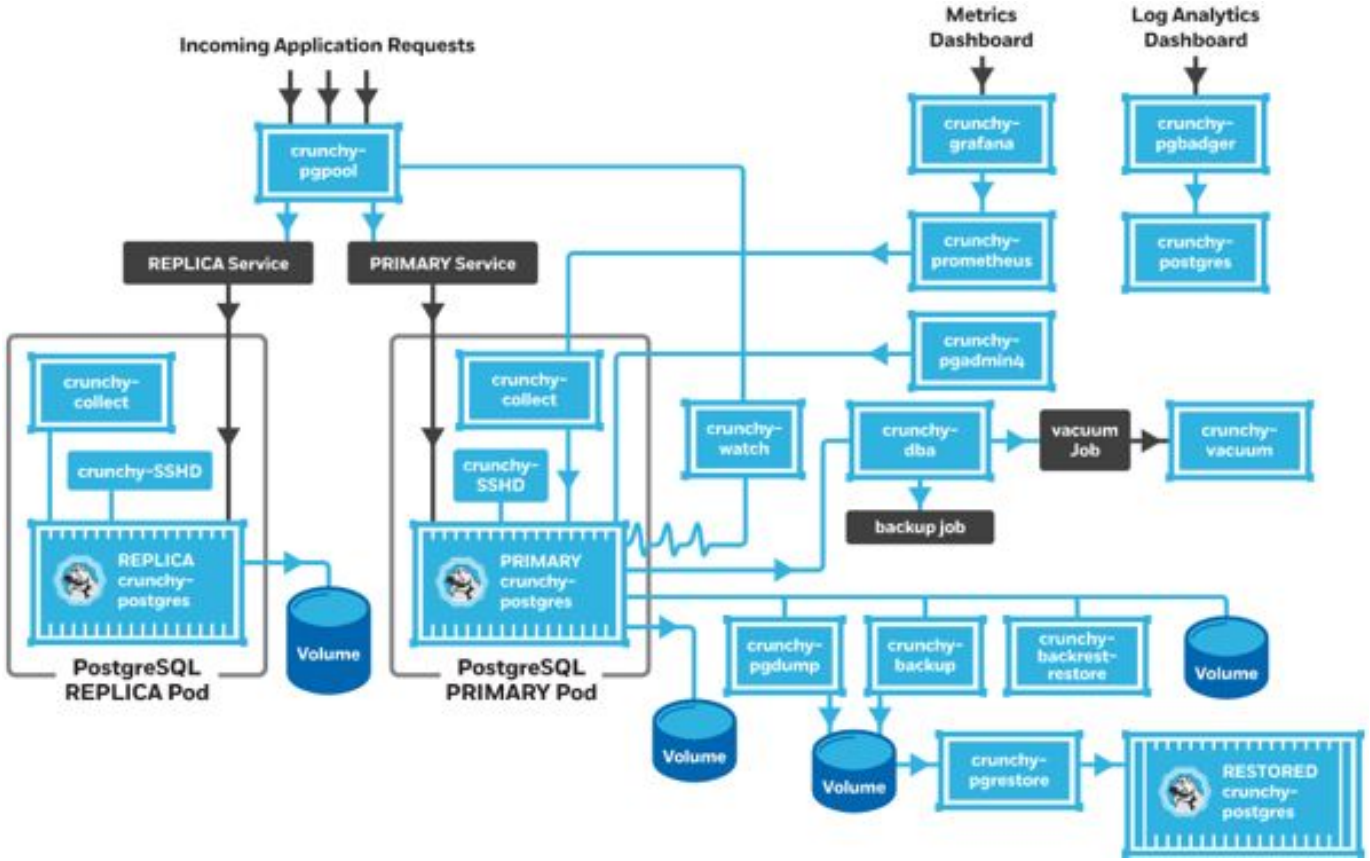


Horizontal scaling (scale-out)

- How:
  - Replication
  - Sharding
- (-) Increased complexity for the application
- (+) More cloud-friendly



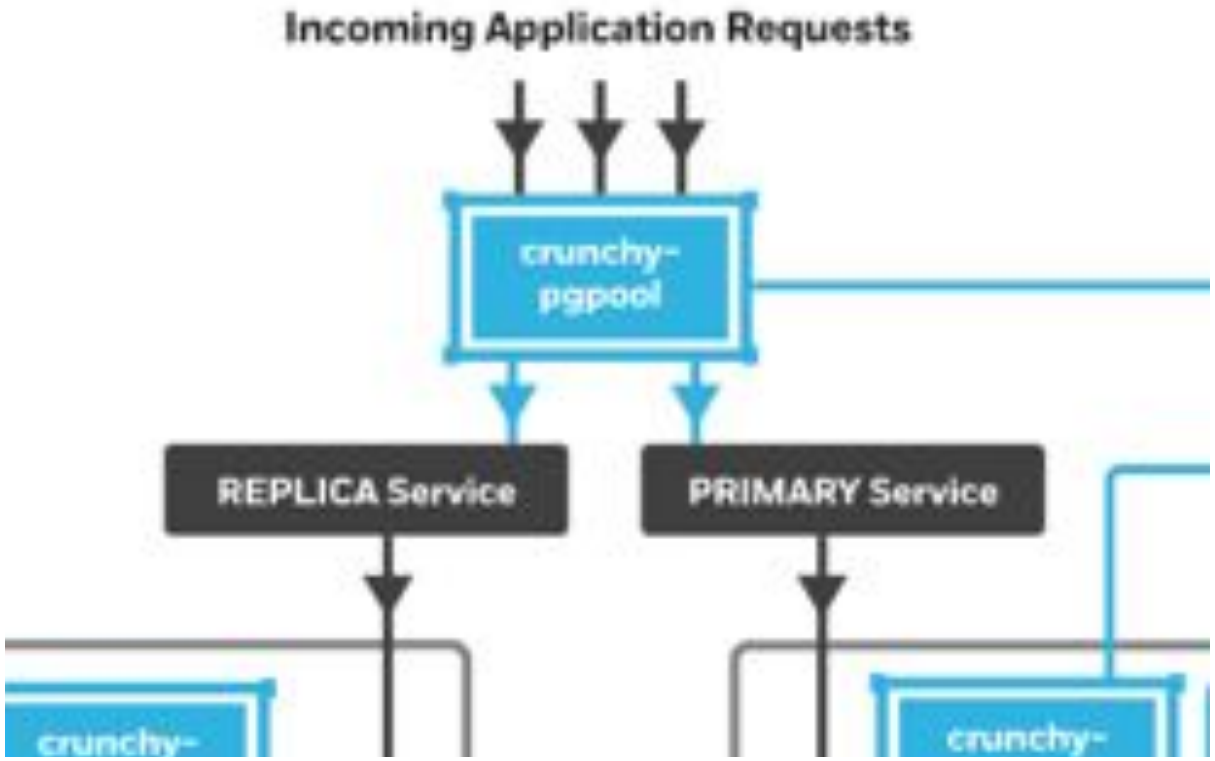
How scaling a DB looks like in reality



Source: https://nubenetes.com/crunchydata/#crunchydata-postgres-operator-401

Red Hat

How scaling a DB looks like in reality



Source: https://nubenetes.com/crunchydata/#crunchydata-postgres-operator-401

Scaling a database Replication

Red Hat

Replication

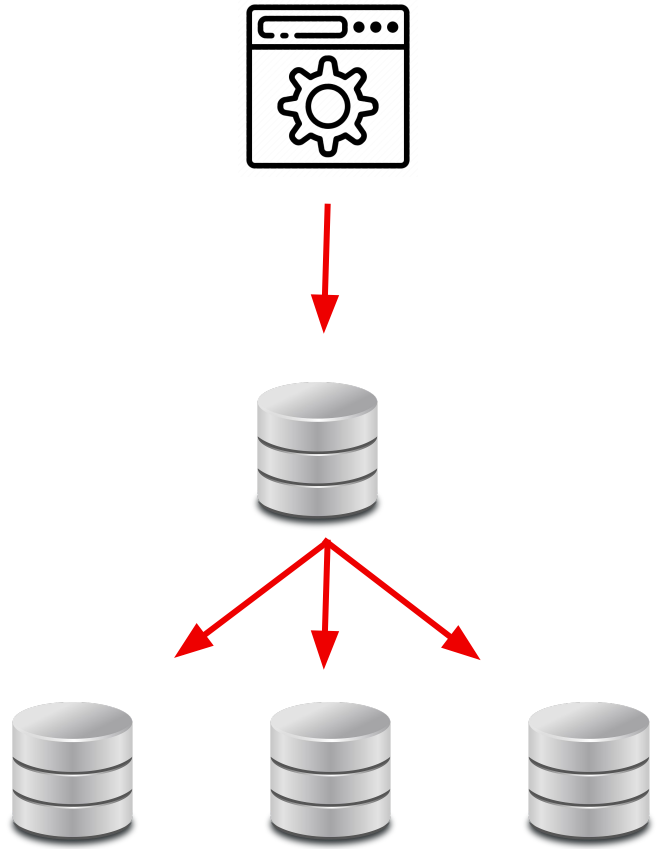
What a developer should take into consideration

Primary

- Reads and writes
- Crash is an availability problem -> failover

Replica

- Read-only
- Delay
- Crash means restarting the node
- Various roles: logging, back-up



Cloud specifics

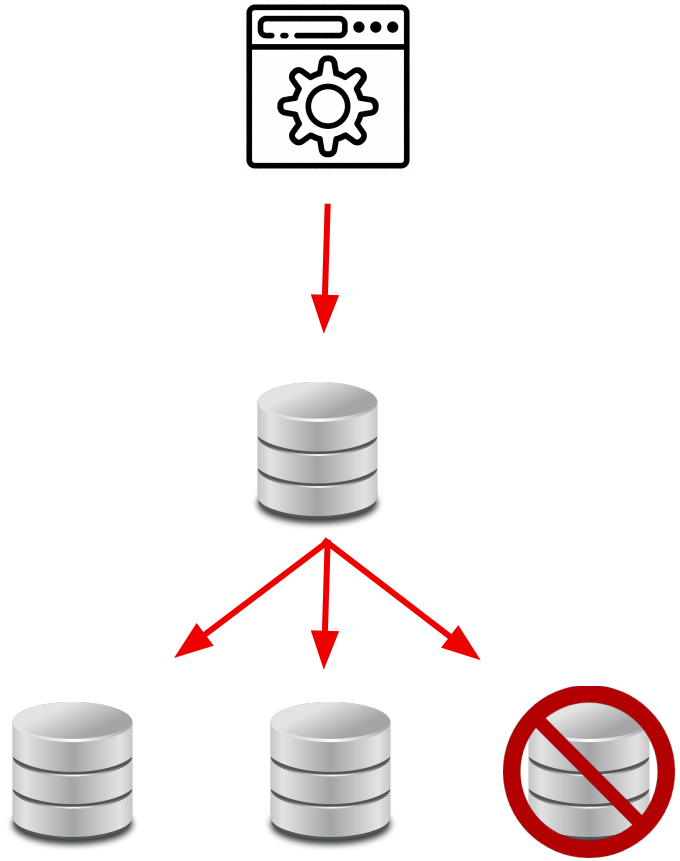
Application must be capable of operating in a dynamic environment

What application should expect

- Connection crashes -> re-do?
- Query takes long -> UX
- Transactions

Other aspects

- Legal requirements
- Cost



CAP theorem

Any distributed data store can provide only two of the following three guarantees:

Consistency

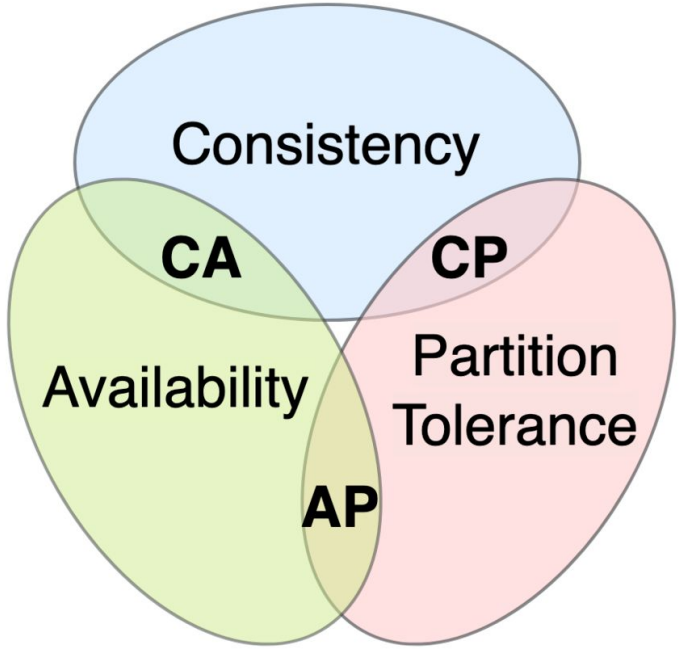
- Every read receives the most recent write or an error.

Availability

- Every request receives a (non-error) response, without the guarantee that it contains the most recent write.

Partition tolerance

- The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.



Source:
https://en.wikipedia.org/wiki/CAP_theorem

PACELC theorem

Extension to the CAP theorem, because response time is important in reality.

In case of network partitioning (P) in a distributed computer system, one has to choose between availability (A) and consistency (C) (as per the CAP theorem),

but else (E), even when the system is running normally in the absence of partitions, one has to choose between latency (L) and consistency (C).

Examples:

MySQL Cluster, and PostgreSQL are PC/EC: they refuse to give up consistency, and will pay the availability and latency costs to achieve it.

Source:
https://en.wikipedia.org/wiki/PACELC_theorem

Multi-master, solution that rules them all

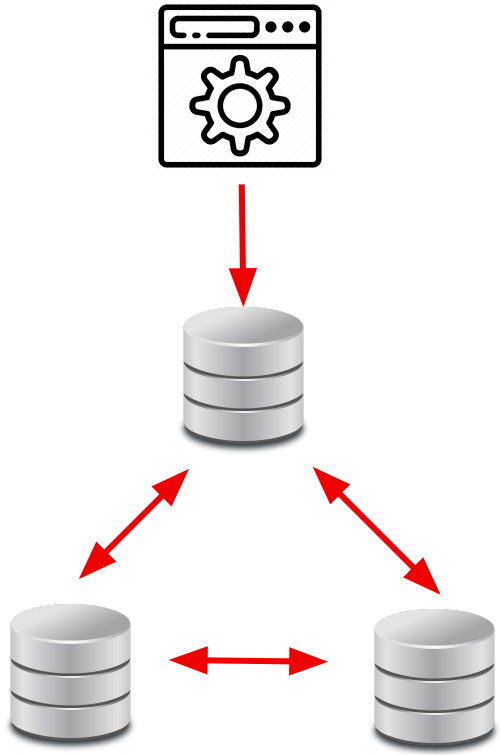
Or not...

Use cases

- No single point of failure
- Does it help scaling writes?

Challenges

- Sync or async?
- Split brain



Multi-master, solution that rules them all

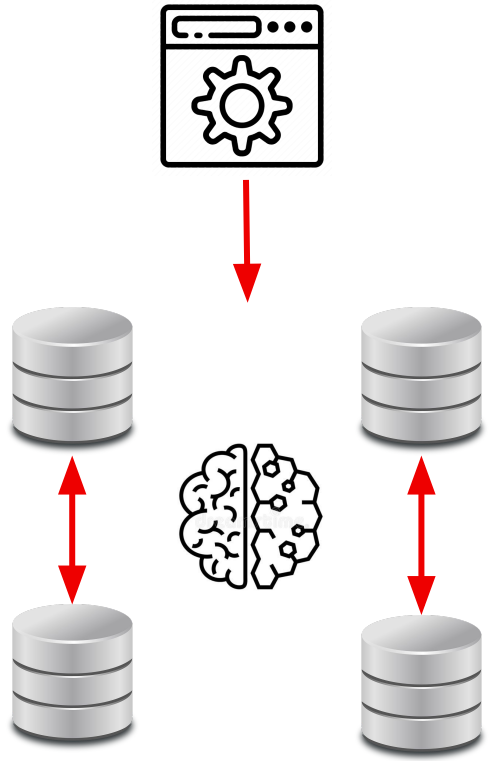
Or not...

Use cases

- No single point of failure
- Scale writes (sometimes)

Challenges

- Sync or async?
- **Split brain**

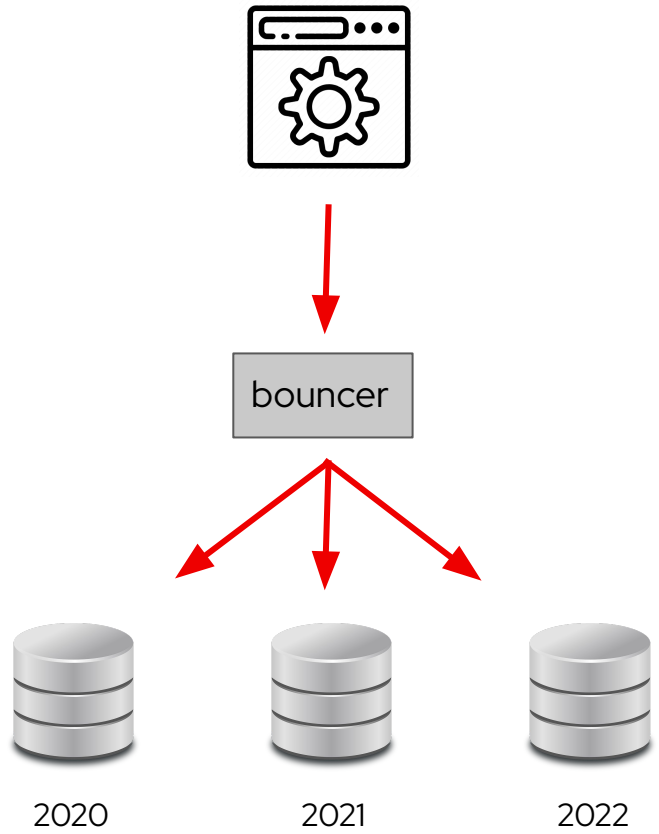


Scaling a database
Sharding

Red Hat

Sharding

When one node is not capable of keeping all the data



Use cases

- Time-series
- Regional-specific queries
- Other domains

Challenges

- Managing the shards
- Queries across shards, aggregations
- Figuring out the sharding key is ... the key

Database Performance

Red Hat

What affects the performance of a DB actually?

This is just beginning...

Deployment

- Network latency
- Resources of the machine
- Configuration (buffers, ...)

Application

- Data amount and structure
- What is done in DB and what in App
- Indexes (they are not for free)
- Be careful about joins, sorts
- ANALYZE+EXPLAIN are friends
- Transactions, locks

Need for DBA

In 2023...

- Operating DBs in cloud and/or for big data is hard
- We should not expect everybody knows everything
- Even if a DB is operated by a different team or even company, developers should be involved

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat