WIKIMEDIA FOUNDATION

From exceptional maintenance to automated routine operation

a story of the datacenter switchover at wikimedia

Giuseppe Lavagetto

(on behalf of the whole SRE team),

Wikimedia Foundation

Outline

- Who are you, and what is wikimedia?
- Our first datacenter switchover
- Streamlining the process
- Building the tools we needed
- Endgame: multi-dc
- How we keep the process efficient
- General lessons in SRE practice

WIKIMEDIA FOUNDATION

01 Introduction

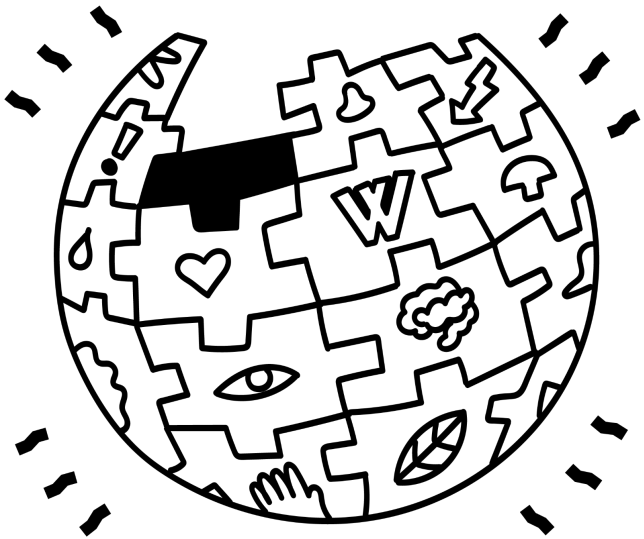
Wikimedia

The Wikimedia Foundation is the nonprofit organization supporting wikipedia and its sister projects (commons, wiktionary, wikisource, wikivoyage...)

Up to 400k rps

1800 bare metal servers

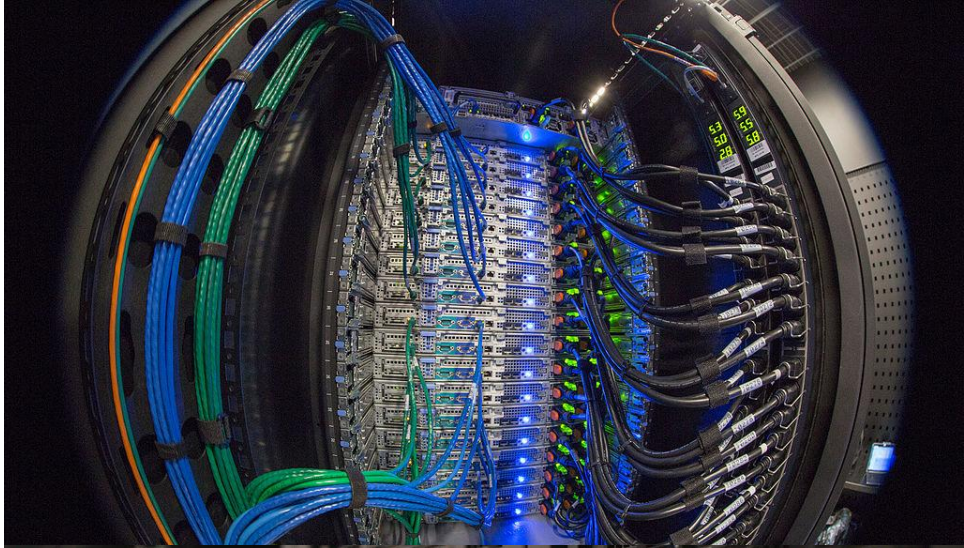
< 4M Yearly cost of infrastructure

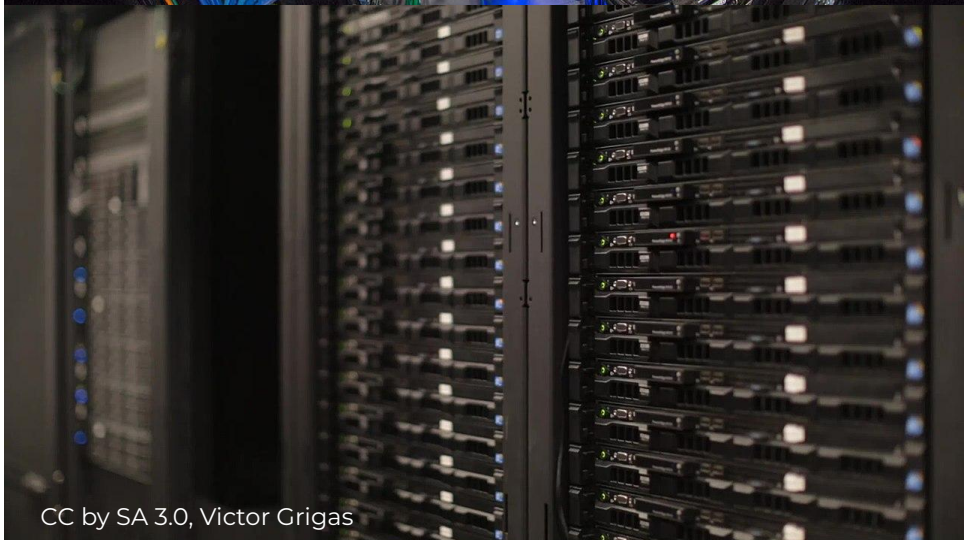


WIKIMEDIA FOUNDATION

Our infrastructure

The CDN/caching layer runs in 5 (soon 6!) datacenters distributed across the globe, while the application layer is replicated across two datacenters in the United States.





WIKIMEDIA FOUNDATION

02 Our first switchover

A second datacenter

- In 2015, we set up a second datacenter to replicate our 14-yrs old infrastructure
- Not all of our datastores work well with replication over non-negligible RTT and/or an insecure network
- Lots of duct tape and IPSec tunnels!

The first switchover

- After almost 6 months of preparation, we perform our first switchover in March 2016
- **15 code commits** across 5 repositories
- **45 minutes of read-only mode**
- 10 engineers involved
- About two hours of somewhat degraded performance

Lessons

- We need to do it again, every 6 months, or we won't be able to do it in an emergency
- We want it to be simpler, faster, less impacting
- **Gitops is not the right paradigm** to manage state and state transitions in a distributed system

WIKIMEDIA FOUNDATION

03 Streamlining the process

Managing state

We need a central repository for the state of the cluster that can be queried by every system in our distributed infrastructure.

Principles: flexibility, no pitfalls, loose coupling

Opting for flexibility
When you're retrofitting a state repository onto an existing infrastructure, flexibility is extremely important. We thus avoided the "batteries included" approach of Consul and chose etcd

Avoid obvious pitfalls
Forcing operators to write to the datastore directly would open us to mistakes of all sorts in a critical system. We thus created a cli tool (and python library) to make write operations as secure as possible.

Loose coupling
Systems should survive the unavailability of the datastore

conftool

- Store objects in a backend following a json schema, and associate tags with the object
- Usable as a cli tool or a python library
- Easy to extend for domain-specific cli tools (dbctl, requestctl)
- has been extended to manage application state, dns, traffic filtering

integration

- Modify applications to watch the etcd API (MediaWiki's EtcdConfig)
- Use confd for third-party applications we didn't write
- Always protect against bad data
- Reduced the number of code commits needed for a switchover to 0 (over several iterations)

MediaWiki

During a switchover we need to set Mediawiki to read-only, then change the main datacenter, then reset read-only.

Eventual consistency is ok
We cache the configuration for 15 seconds in APCu (php's global memory store).

Fault tolerance
MediaWiki is able to keep running if the whole etcd cluster is unavailable, unreachable or has invalid data.

Takeaway
Whenever possible, native clients to your state store should be implemented - as it will allow you maximum flexibility.

We now established the pattern of modifying state via a cli tool and reading it via EtcdConfig: the system was extended to support other things like e.g. the mysql configuration.

Discovery

We need to have a global routing table for our applications: which datacenter should I call for this service?

State is stored in etcd

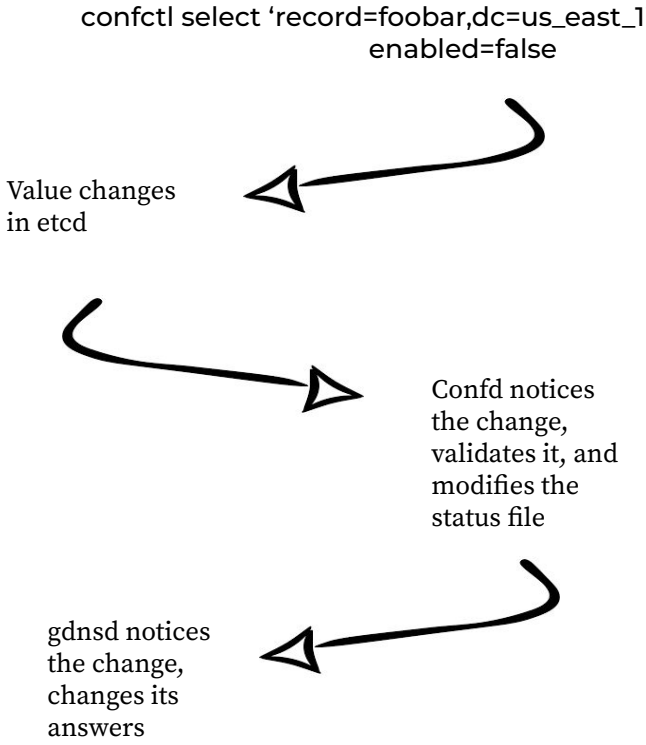
```
tags:
  datacenter: us_east_1
  record: foobar
data:
  enabled: true
  ttl: 300
```

DNS is the simplest globally available API

Based on the client IP address, provide the service IP for the record pointing to the nearest available datacenter.

Beyond DNS

More complex logic can be wired in the control plane of our service mesh, but we never truly needed it.



A bit more in detail

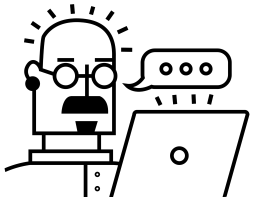
- Active/active and active/passive services have different logic
- “Fail fast” service for maintenance
- Clients need to revalidate DNS entries
- Using a service mesh ensures consistent behaviour

WIKIMEDIA FOUNDATION

04 Building the tools we need

What are the steps to perform?

- Warm up the new dc
- Kill all long running jobs, stop processing
- Set the application to read only



- Switch replication of all datastores
- Set the application to read-write
- Edge caches read from the new DC
- Start periodic jobs

WIKIMEDIA FOUNDATION

There’s a ghost tale we all tell young SREs, it starts like this:

(for some definition of small)

Introducing spicerack, our execution engine for complex maintenance tasks

(pip install spicerack)

WIKIMEDIA
FOUNDATION

spicerack

- Written in python, powered by the *cumin* distributed execution tool
- Define and run “cookbooks”, automated complex recipes operating on physical servers, datastores, remote APIs
- Each cookbook can have multiple stages
- Ability to query servers/services catalogs
- Builtin dry-run functionality
- Rollbacks

A simple cookbook

```
from spicerack.cookbook import CookbookBase, CookbookRunnerBase
from spicerack import dns

class WipePdnsCache(CookbookBase):
    """Wipes the dns recursor caches for a dns record."""
    def argument_parser(self):
        parser = super().argument_parser()
        parser.add_argument("record",  help="dns record to clean cache for")
        parser.add_argument("--expected", help="expected record after purging")

    def get_runner(self, args):
        return WipePdnsRunner(args, self.spicerack)
```

A simple cookbook

```
class WipePdnsRunner(CookbookBase):
    """Runs the cache wipe"""
    def __init__(self, args, spicerack):
        self.hosts = spicerack.remote().query("A:pdns_resolvers")
        self.record = args.record
        self.expected_ip = args.expected

    @retry(tries=5, backoff_mode="constant")
    def run(self, args):
        self.hosts.run_sync(f"rec_control wipe-cache {self.record}")
        dns.check_record(self.record, self.expected_ip, resolvers=self.hosts)
```

switchover

- `sudo cookbook sre.switchdc.mediawiki dc-from dc-to`
- 9 stages, 16 sub-cookbooks
- We keep track of execution time and log every action taken by the cookbook
- “Live test” one week before our planned switchovers
- **< 2 minutes of read-only time**

WIKIMEDIA
FOUNDATION

WIKIMEDIA FOUNDATION

05 endgame

no Avengers references in the next slides, sorry.
My slides need to be CC-licensed.

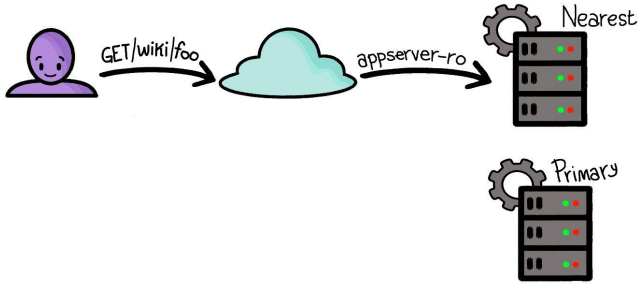
Applications should serve traffic from all datacenters

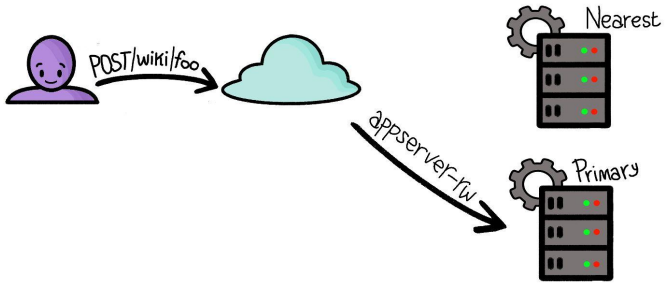
At worst, direct write traffic to the primary datacenter

benefits

- Caches are hot at all times everywhere
- No hardware or configuration rot
- Forces SREs and devs in a multi-dc mindset
- Increased peak capacity*

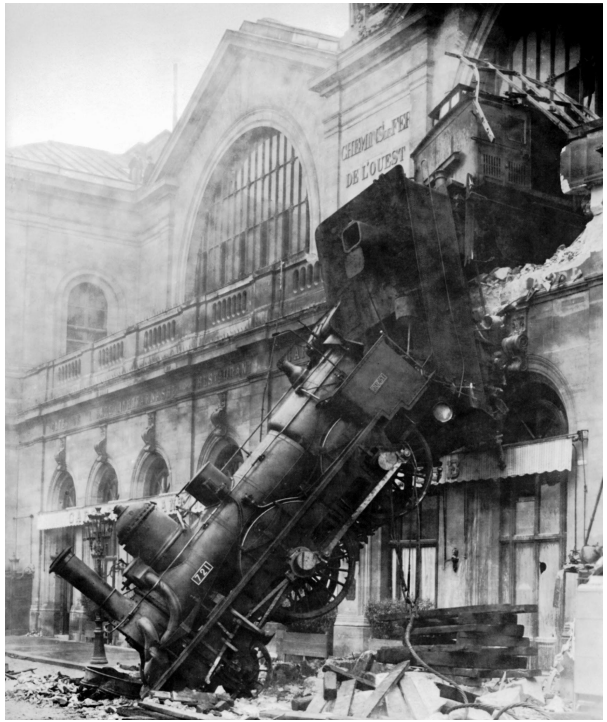
WIKIMEDIA
FOUNDATION





- When a user makes an edit, we send them to the primary DC
- A/A vs A/P discovery records
- “Stick to primary” cookie for consistency

Datastores



- MariaDB (and most SQL databases) have a primary/replica model.
- Caches need to be consistent
- Distributed systems and logical consistency
- **primary/replica << distributed << local-only**

WIKIMEDIA
FOUNDATION

WIKIMEDIA FOUNDATION

06 Keeping the process efficient


We switch primary datacenter twice per year
We decided to do it in the weeks of the equinoxes so that the community expects it

Capacity test

For one week after the switchover, we run all traffic through the new primary datacenter. This also allows disruptive maintenance.

No haunted graveyards

The procedure needs to be **clear**, give **good feedback** to a first time user, and to be operated in an emergency situation.
We ensure this by having **the newest member of the team** perform the first switchover after they join

```
<joe> k: now ofc I want a quote for my presentation :P
<k> a quote?
<k> "
<k> ^ that good?
<joe> ...
<c> You're complaining when you got double what you asked
```

... I love my team

WIKIMEDIA FOUNDATION

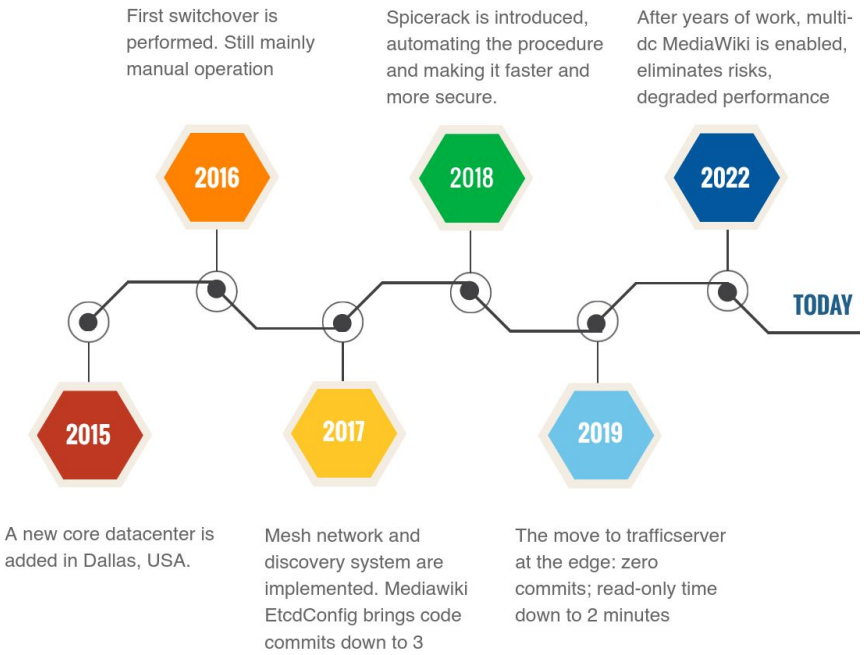
07 General lessons in SRE practice

“dripping water hollows out stone”

Titus Lucretius Carus

SWITCHOVER

MILESTONES



We kept improving bit by bit, small spike of work by small spike of work. Several SRE and dev teams involved - you need to get them onboard!

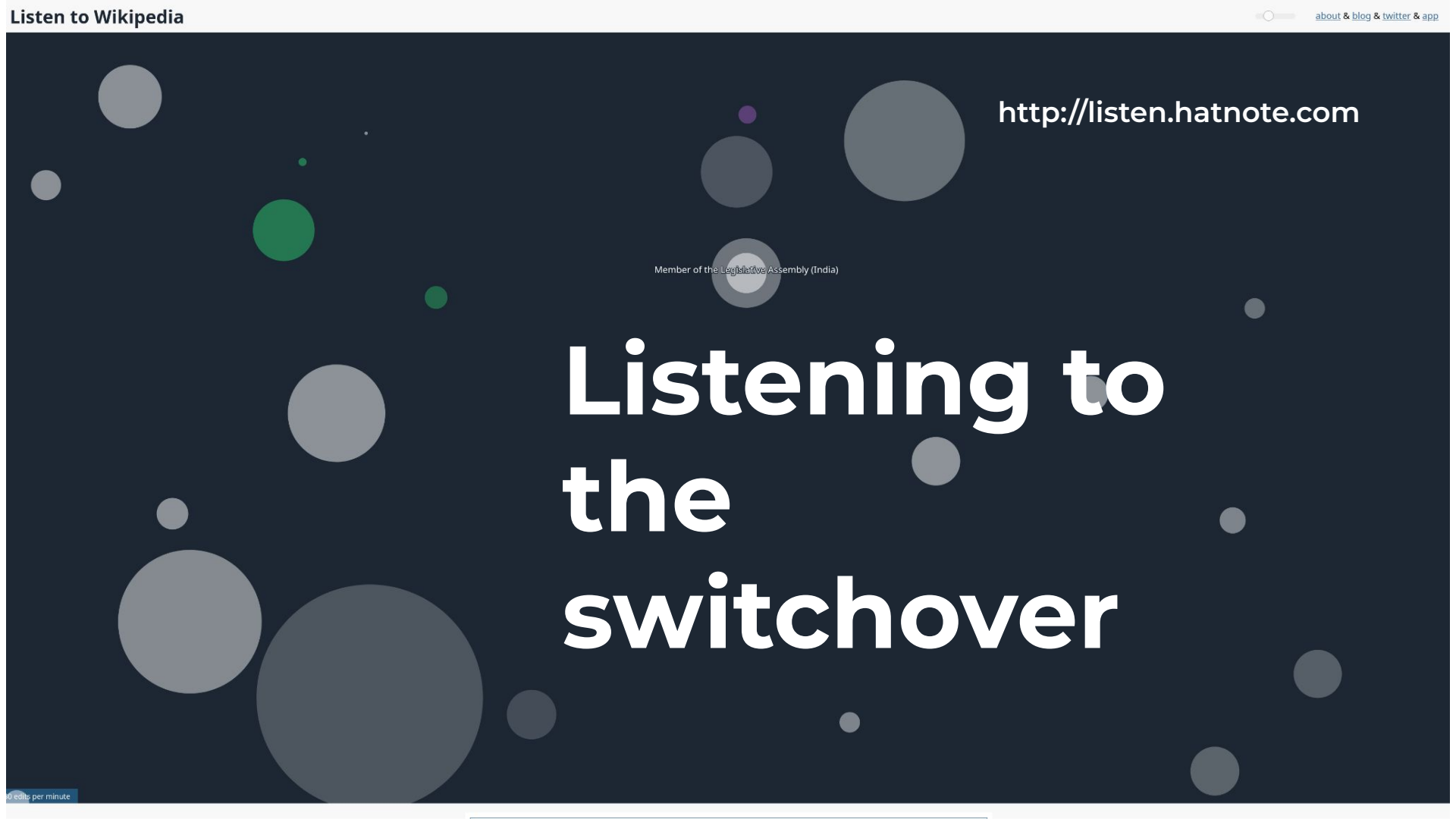
Config vs state

- Configuration changes behaviour of an application.
- State describes the dynamic behaviour of your distributed system
- Gitops is great for configuration, pretty terrible in managing state
- You need a fast reacting system to manage the state of your system

tooling

A tool shouldn't just reduce toil, but also anxiety:

- Remove any obvious (and non-obvious, eventually) pitfalls
- **UX >> feature completeness**
- A tool needs clear feedback. Also, be creative!


Listen to Wikipedia
about & blog & twitter & app
http://listen.hatnote.com
Member of the Legislative Assembly (India)
Listening to the switchover
0 edits per minute

Listening to the switchover

Questions?

https://wikimediafoundation.org/about/jobs/