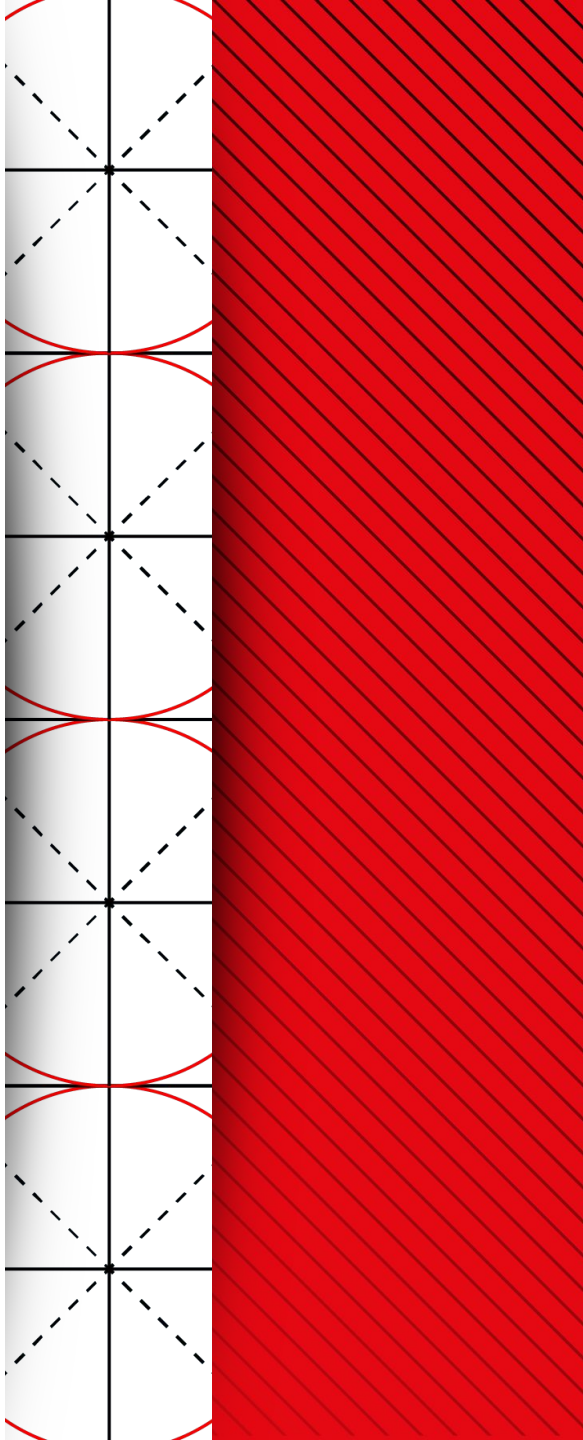
Building and Running Cloud Native Cassandra

Vinay Chella, Joey Lynch
Distributed Database Engineers
Netflix
NGCC 2019



Speakers

Vinay Chella

Distributed Systems Engineer

Focusing on Apache Cassandra and Data Abstractions

Cloud Data Engineering
Netflix

Joey Lynch

Distributed Systems Engineer

Distributed system addict and data wrangler

Cloud Data Engineering
Netflix

Outline

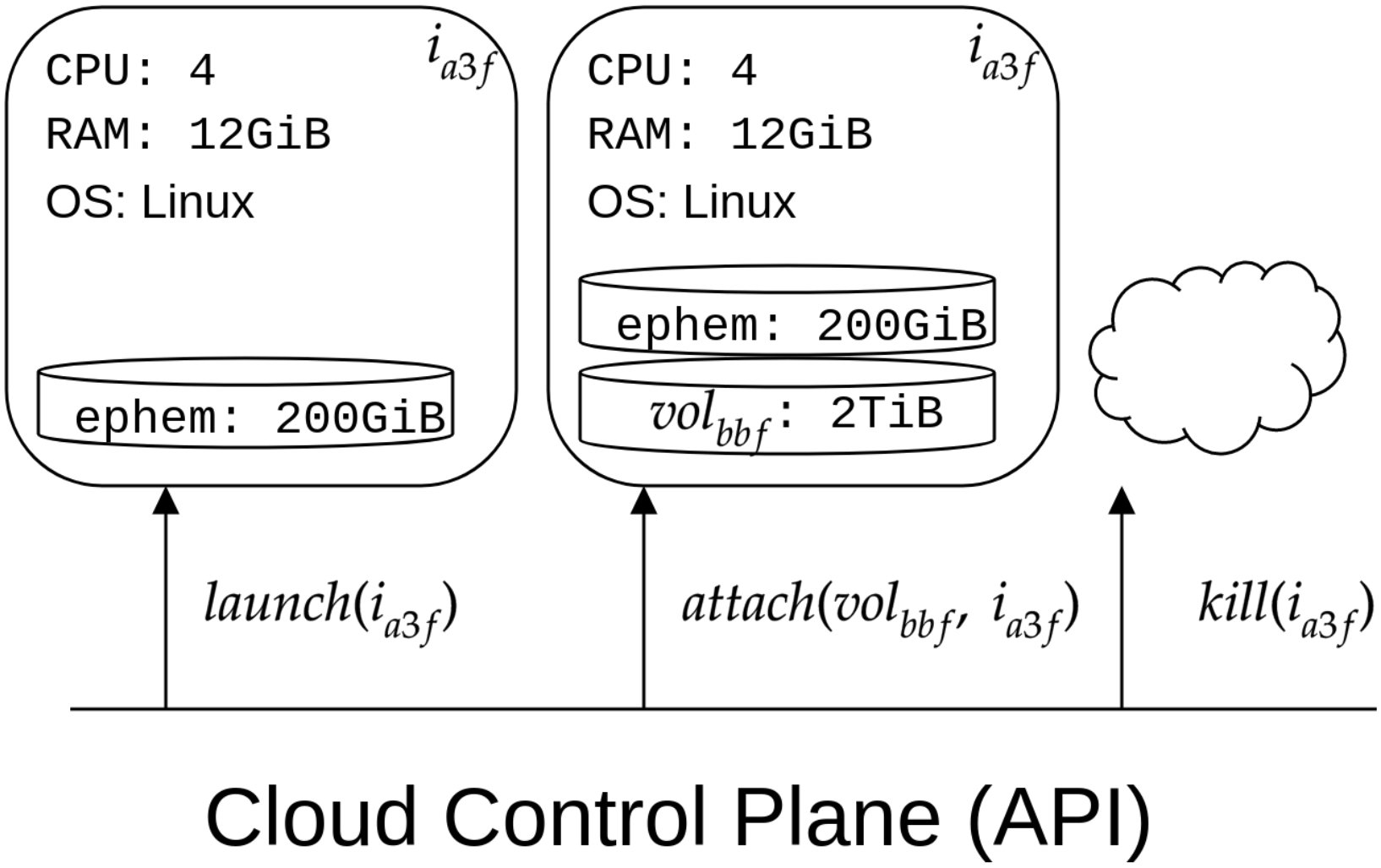
Goals to help align the project cloud principles.

- What is Cloud Native?
- Cassandra's Rough Edges
  - Development
  - Packaging
  - Starting a Cluster
  - Running a Cluster
- Our Proposed Solutions

What even is “Cloud Native”?


aws


Azure

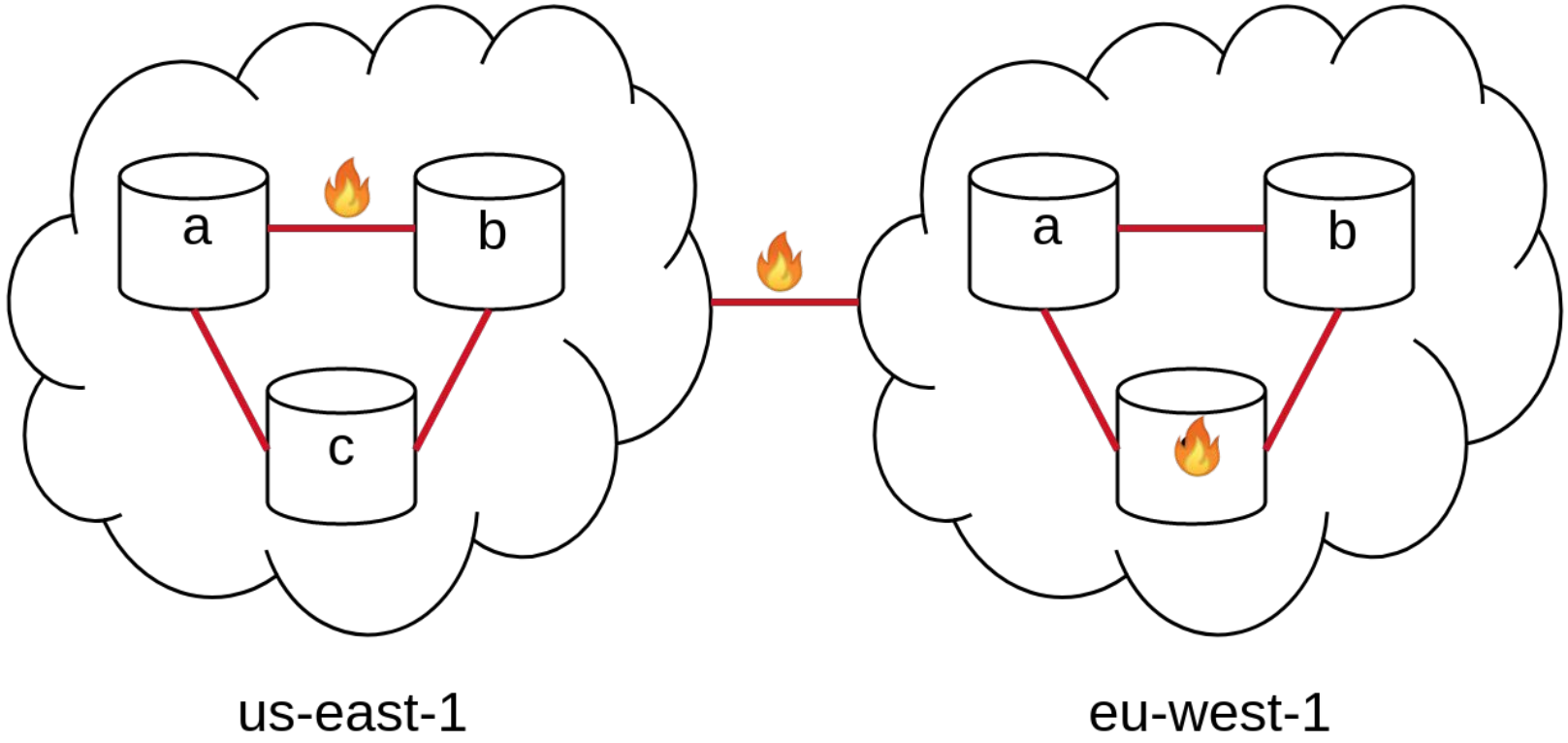


Any hardware configuration

Any operating system

Cattle not Pets

Will die constantly

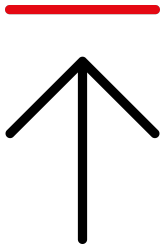


Clouds provide mappings on top of “datacenters”

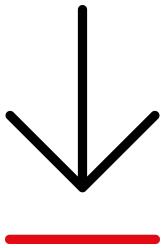
Zones will fail

Regions will fail

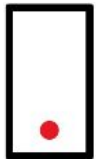
“Durable” storage is not so durable



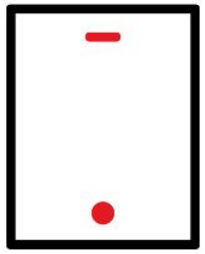
Develop



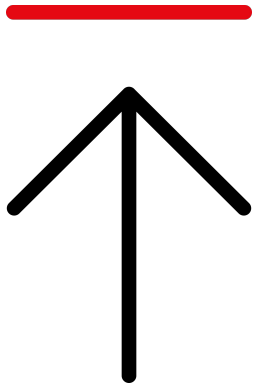
Package



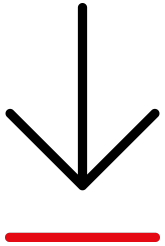
Starting



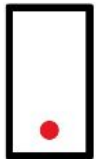
Scaling



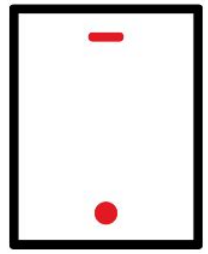
Develop



Package



Starting



Scaling

Friction for New Contributions

`docs`	Discovery, building mental model
`code`	Understand and modify
`build`	Create artifacts (jar, pkg, container)
`test`	Confidence they haven’t broken something

Apache Cassandra Documentation v4.0

Data Modeling

Todo
TODO

Overview

Todo
todo

First interactions?

— I don’t know how the database works

— Where can I run v4.0?

Dynamo

Gossip

Todo
todo

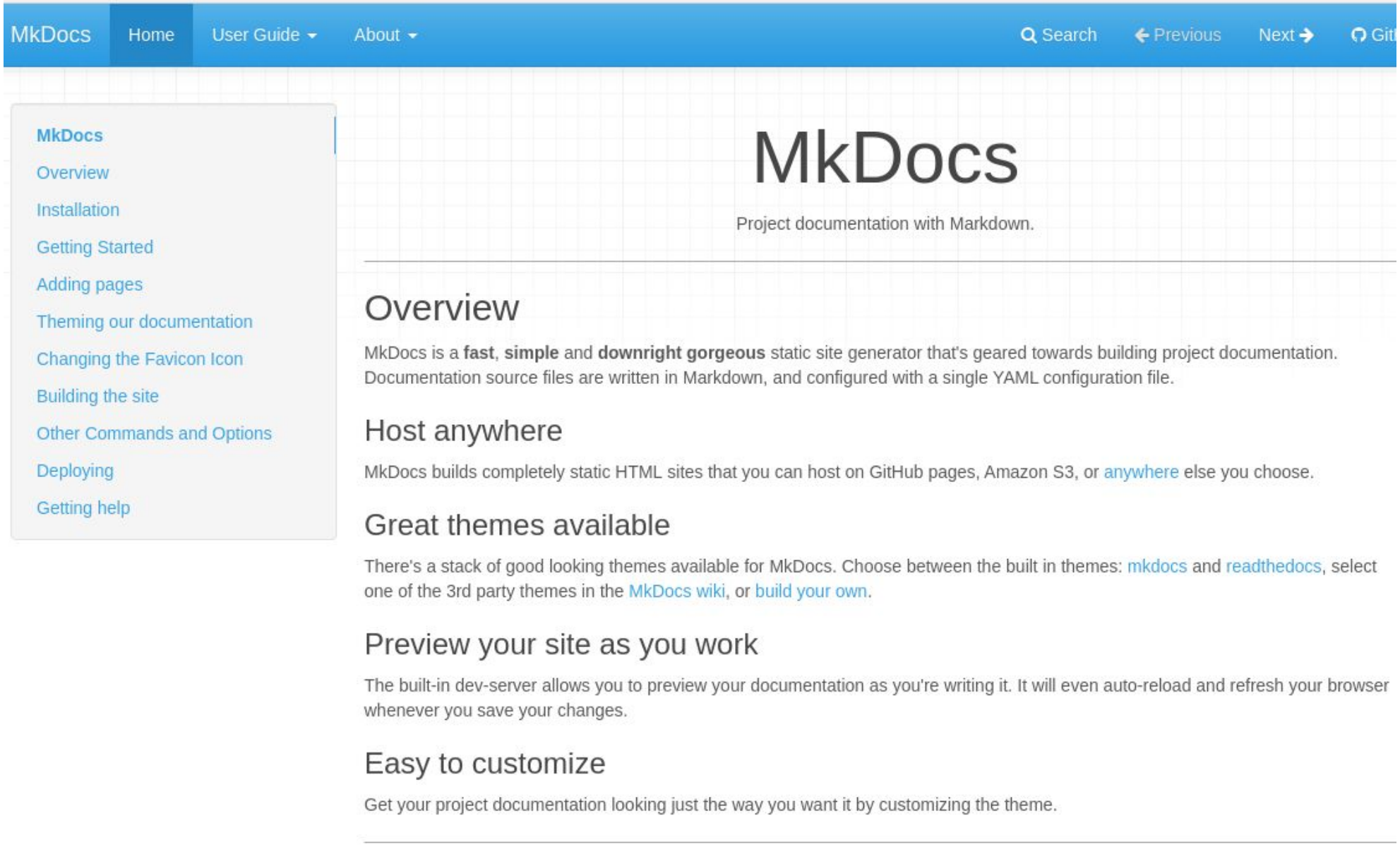
Failure Detection

Todo
todo

Token Ring/Ranges

Todo
todo

How Netflix Does This



How Netflix Does This

Branch: **gh-pages** ▾ | **New pull request**

Unknown Deployed `b2f5295` with MkDocs version: 1.0.4

assets	Deployed `b2f5295` with MkDocs version: 1.0.4
images	Deployed `b2f5295` with MkDocs version: 1.0.4
latest	Deployed `b2f5295` with MkDocs version: 1.0.4
search	Deployed `b2f5295` with MkDocs version: 1.0.4
.nojekyll	Deployed `b2f5295` with MkDocs version: 1.0.4
404.html	Deployed `b2f5295` with MkDocs version: 1.0.4
index.html	Deployed `b2f5295` with MkDocs version: 1.0.4
sitemap.xml	Deployed `b2f5295` with MkDocs version: 1.0.4
sitemap.xml.gz	Deployed `b2f5295` with MkDocs version: 1.0.4

How Netflix Does This

Source

master ··· cdemkdocs / docs / **cassandra** /

Source	Description
..	
images	
api.md	Adding Material design and site layout
data_explorer.md	Update docs for C* Data Explorer and Forklifter.
datamodel.md	Update datamodeling with column overhead learr
dual_writes.md	Update all "#cde slack channel" to a link.
faq.md	new Q: customization of wide row report

Markdown

Joseph Lynch `feature/josephl_tradeoffs → master`

Write the tradeoff docs

Overview Diff Commits

Pull Request

Data Model Search

CDE

Home

Getting Started

Cassandra

Overview

Spring Data Cassandra

Aeneas client

Astyanax client

CQL Data Modeling

Immediate Deploy

The most important choice you make when modeling data in Cassandra is what your **partition** key will be. Cassandra takes the data in the partition key, hashes it, and that determines which Cassandra host(s) will store your data.

If this partitioning is uneven, meaning that you end up with lots of data mapping to the certain partitions but not others, then you have created a **hotspot**, where query load (queries, and data size) is concentrated on a small number of nodes. Hotspots can knock over Cassandra clusters, harm your application availability and result in inconsistent performance. Avoid hotspots at all costs in your data models.

Proposal

`docs`	Move svn website to git branch in main repo
`code`	
`build`	Replace sphinx with markdown (pandoc)?
`test`	Automatically build and publish docs (jenkins + docker?)

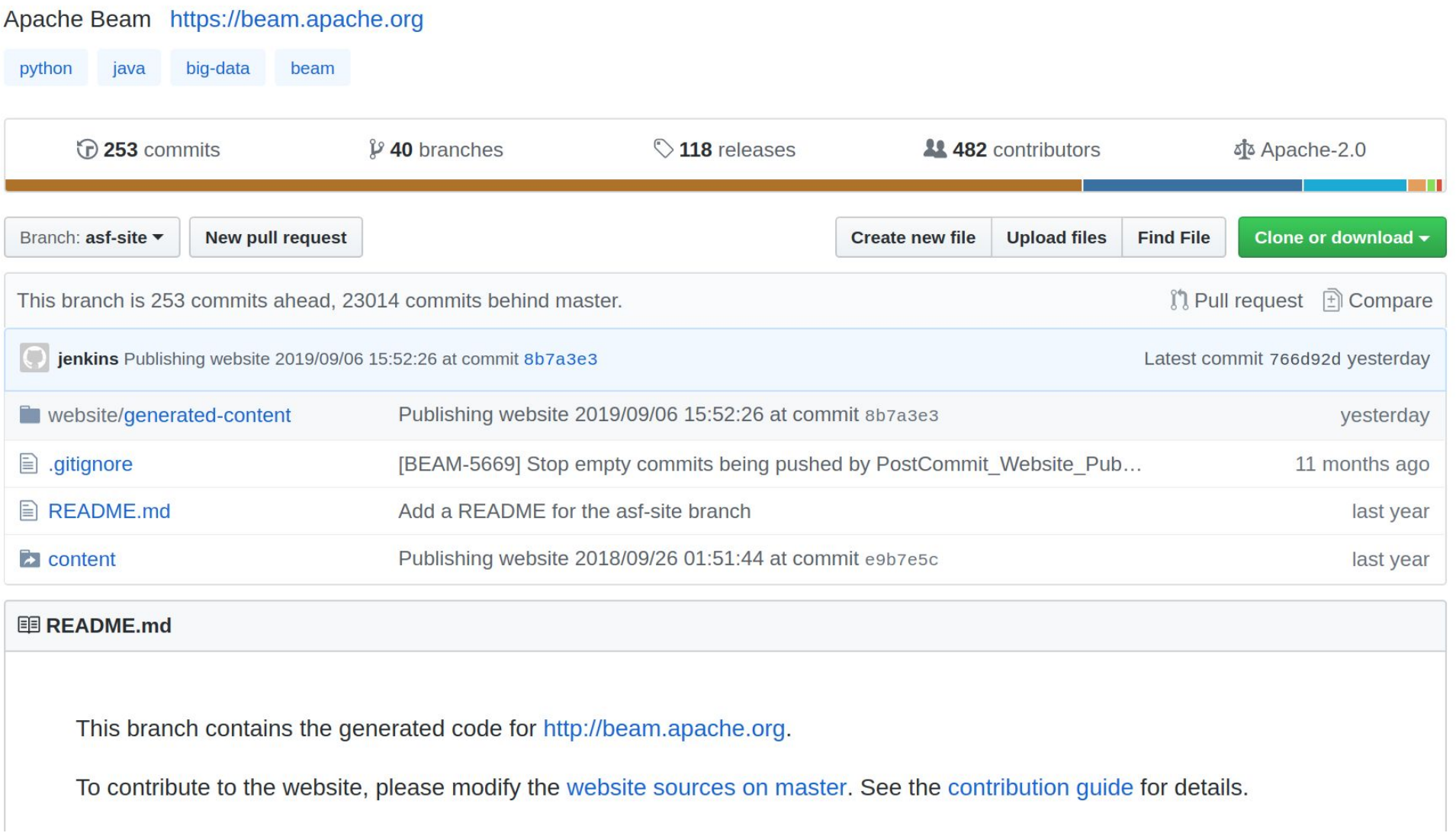
Proposal

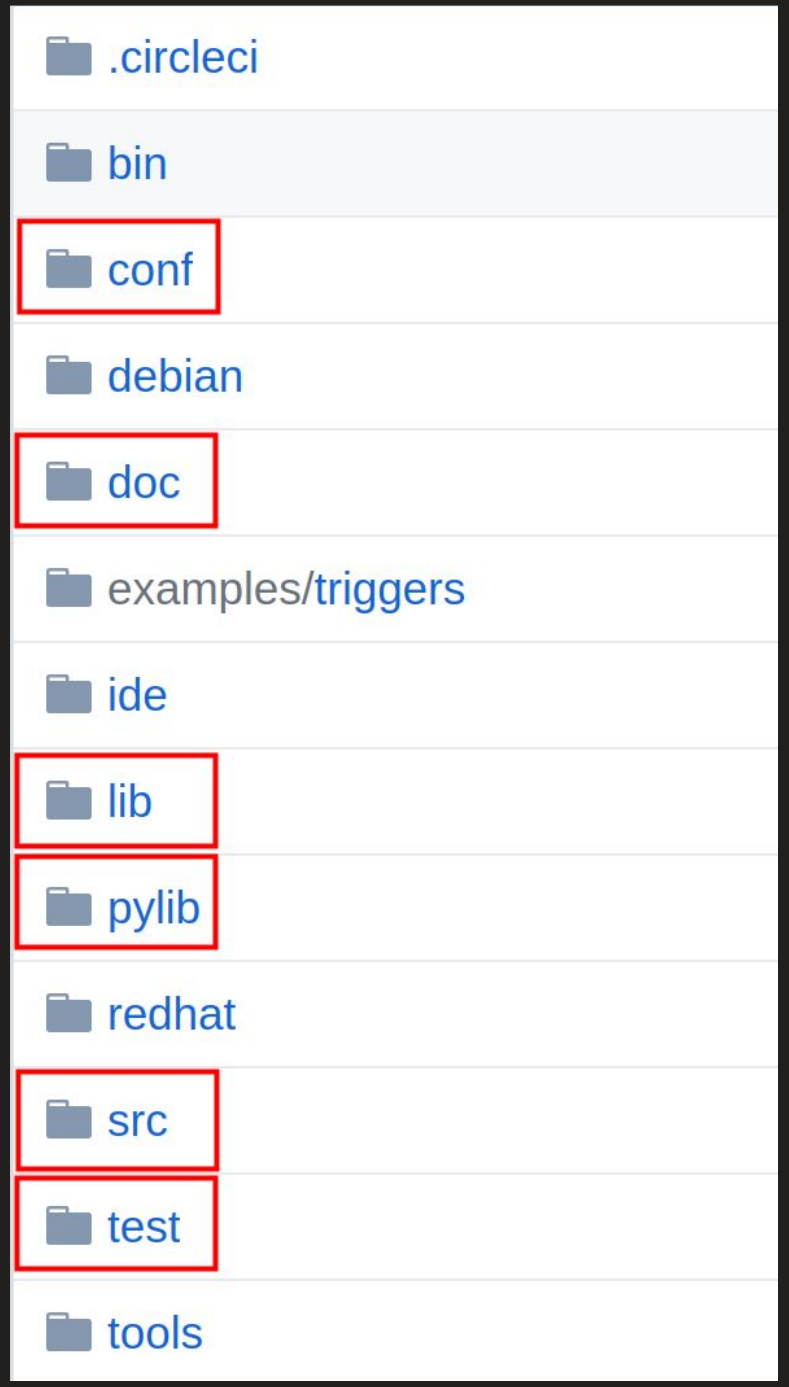
docs

code

build

test





Non standard layout

2260 line XML build.xml

Checked in library .jars

Checked in python library


681


3.6K


34.3K


355


69.2K


9.5K


2.8K


3h


1.8K


1.4K


1h

What is Gradle?

Gradle is a build tool with a focus on build automation and support for multi-language development. If you are building, testing, publishing, and deploying software on any platform, Gradle offers a flexible model that can support the entire development lifecycle from compiling and packaging code to publishing web sites.

What is Apache Maven?

Maven allows a project to build using its project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Once you familiarize yourself with how one Maven project builds you automatically know how all Maven projects build saving you immense amounts of time when trying to navigate many projects.

```
rootProject.name='ndbench'

include 'ndbench-api'
include 'ndbench-cass-plugins'
include 'ndbench-cli'
include 'ndbench-core'
include 'ndbench-cockroachdb-plugins'
include 'ndbench-dynamodb-plugins'
include 'ndbench-dax-plugins'
include 'ndbench-es-plugins'
include 'ndbench-geode-plugins'
include 'ndbench-janusgraph-plugins'
include 'ndbench-sample-plugins'
include 'ndbench-web'
include 'ndbench-aws'
```

Can use Gradle for Cassandra too

```
sourceSets.remove(sourceSets.main)
sourceSets.remove(sourceSets.test)

ant.importBuild('build.xml') { antTargetName ->
    'ant' + antTargetName
}
```

Import build.xml

```
dependencies {
    compile "com.netflix.priam:priam-cass-extensions:${priam_extensions_version}"
    compile 'com.jeffjirsa.cassandra.db.compaction:TimeWindowCompactionStrategy:2.1
}
```

Add any jars we want at build time

Can use Gradle for Cassandra too

```
ext {
    pname = "[redacted]nfcassandra-30x"
    cass_version = ant.properties['version']
    cass_home_dir = [redacted]
    cass_filename = "nf-cassandra-${project.ext.cass_version}-bin.tar.gz"
    priam_extensions_version = "3.11.52"
    priam_extensions_jar = "priam-cass-extensions-${priam_extensions_vers
}
```

```
buildDeb {
    dependsOn customizeFiles
}

ospackage {
    packageName project.ext.pname
    version = project.ext.cass_version

    println "Building deb for nf-cassandra ${version}"
    into("${project.ext.cass_home_dir}/lib") {
        from configurations.runtime.filter { it.name.startsWith('priam') }
        ...
```

Build debs directly

Proposal

docs

Start by importing build.xml

code

Gradually modernize build system via **gradle** multi-project builds

build

Use out of the box **dependency locking**

test

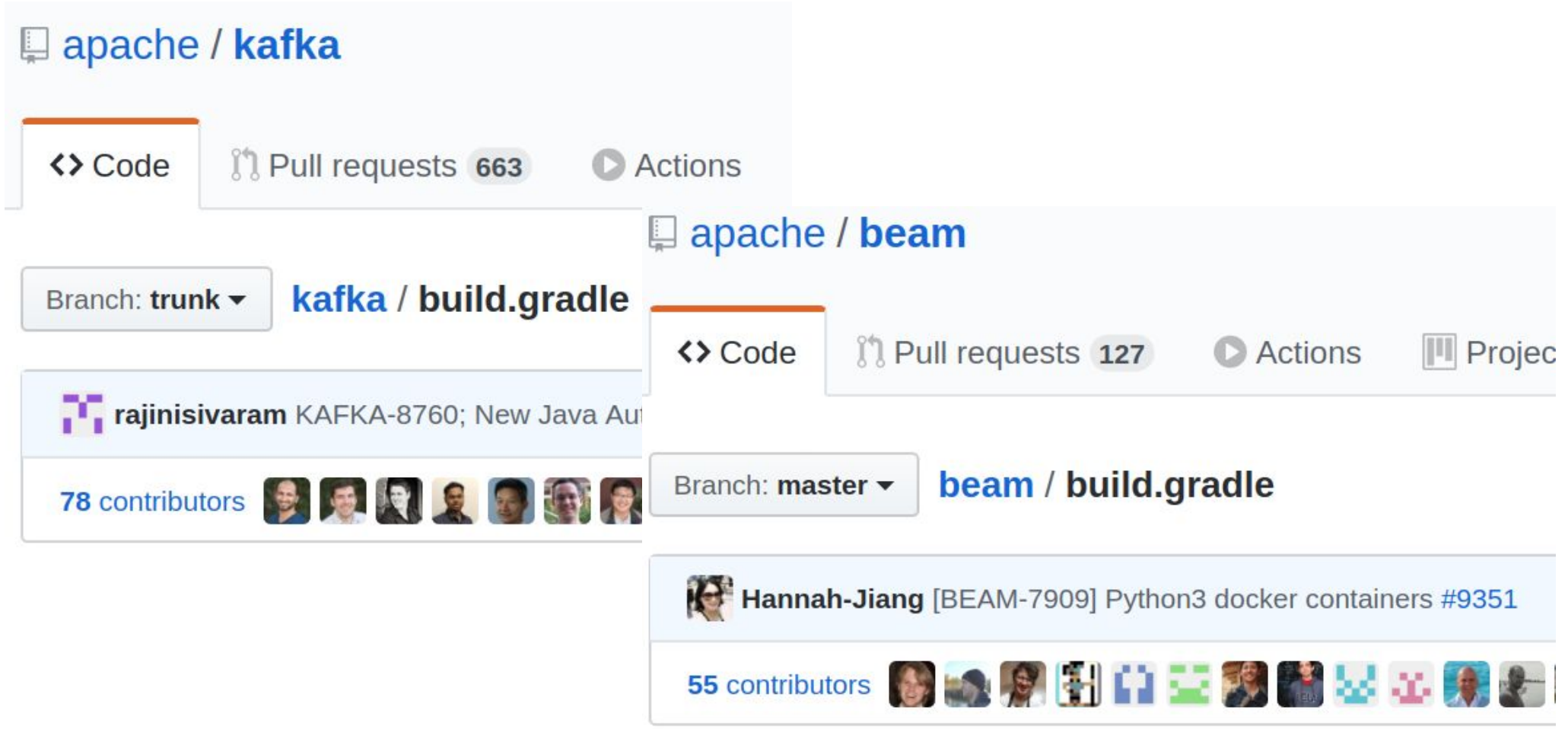
Proposal

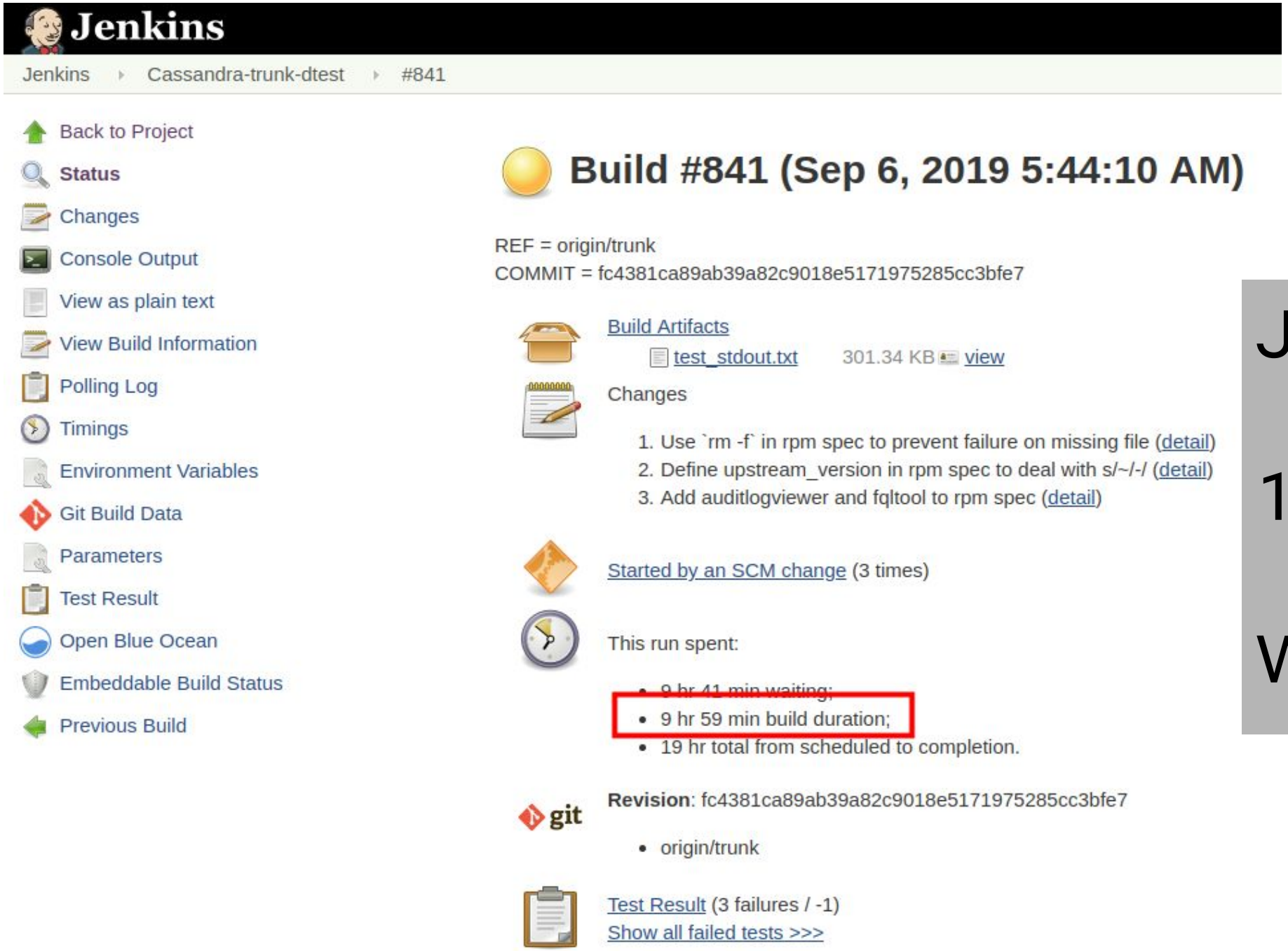
docs

code

build

test

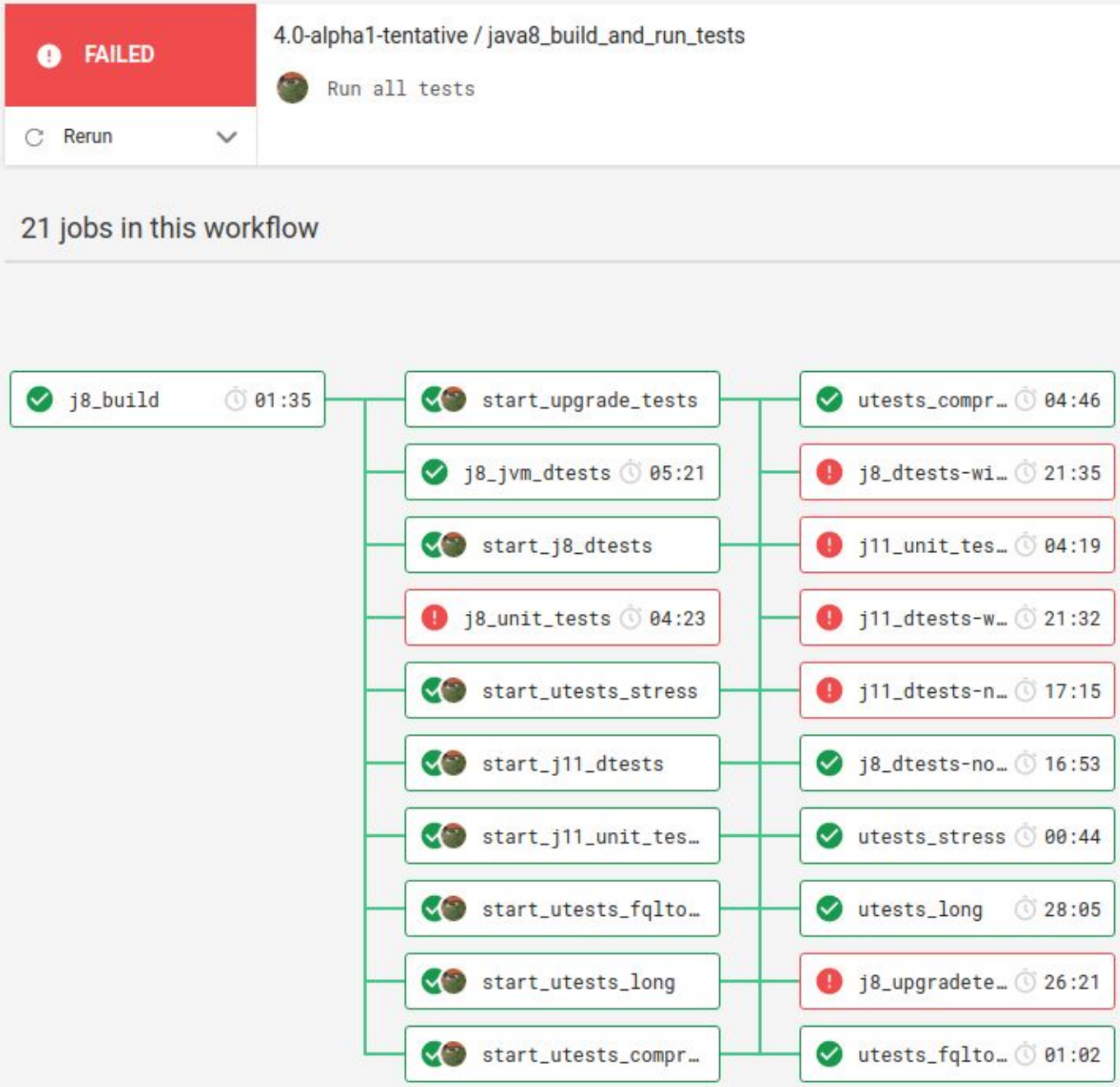




Jenkins

10 hour builds

Who can run?



CircleCI

20 minute builds

Anyone can run them

Proposal

`docs`

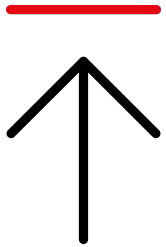
`code`

Keep moving with the CircleCI integration for testing (artifacts stay on jenkins)

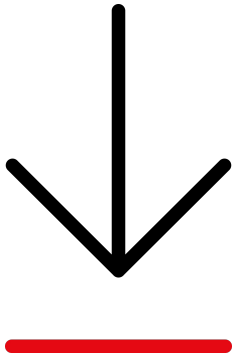
`build`

Run unit tests on every pull request (5 minutes, doable with free tier)

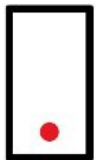
`test`



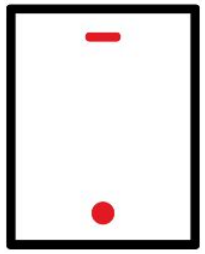
Develop



Distribute



Launch



Scale

Cloud Native Distribution

`source`	Building from source must be easy
`packages`	Integration with package managers `apt, yum, brew ...`
`containers`	Having Docker containers for testing, as well as production

Cloud Native Distribution

`source`

`packages`

`containers`

Already talked about this, we can improve in this area with previous proposals

Cloud Native Distribution

source	
packages	Already have pretty great deb and rpm integration.
containers	Custom packages slightly difficult

Cassandra / CASSANDRA-2356

make the debian package never start by default

Autostart is not Cloud Native

Details

Type:	Improvement	Status:	OPEN
Priority:	Low	Resolution:	Unresolved
Component/s:	Packaging	Fix Version/s:	4.x
Labels:	debian packaging remove-reopen		

Description

Currently the debian package that installs cassandra starts cassandra by default. It sounds like that is a standard debian packaging convention. However, if you want to bootstrap a new node and want to configure it before it creates any sort of state information, it's a pain. I would think that the common use case would be to have it install all of the init scripts and such but **not** have it start up by default. That way an admin can configure cassandra with seed, token, host, etc. information and then start it. That makes it easier to programmatically do this as well - have chef/puppet install cassandra, do some configuration, then do the service start.

With the current setup, it sounds like cassandra creates state on startup that has to be cleaned before a new configuration can take effect. So the process of installing turns into:

- install debian package
- shutdown cassandra
- clean out state (data/log dirs)
- configure cassandra
- start cassandra

That seems suboptimal for the default case, especially when trying to automate new nodes being bootstrapped.

Another case might be when a downed node comes back up and starts by default and tries to claim a token that has already been claimed by another newly bootstrapped node. Rob is more familiar with that case so I'll let him explain it in the comments.

Cloud Native Distribution

`source`

`packages`

`containers` Not doing well at containers.

https://github.com/docker-library/cassandra

Maintained by: the Docker Community

This is the Git repo of the Docker "Official Image" for `cassandra` (not to be confused with any official `cassandra` image provided by `cassandra` upstream). See the Docker Hub page for the full readme on how to use this Docker image and for information regarding contributing and issues.

The full image description on Docker Hub is generated/maintained over in the docker-library/docs repository, specifically in the `cassandra` directory.

How Netflix Does Testing Images

```
# Either tearing out nflx customizations we can't use in the containe
# or setting things so we startup faster
cass_config.update({
    "cluster_name": "docker",
    "listen_address": os.environ.get("LISTEN_ADDRESS", "127.0.0.1"),
    "rpc_address": os.environ.get("RPC_ADDRESS", "0.0.0.0"),
    "broadcast_rpc_address": os.environ.get("BROADCAST_RPC_ADDRESS",
    "partitioner": "org.apache.cassandra.dht.Murmur3Partitioner",
    "auto_bootstrap": False,
    "server_encryption_options": {"internode_encryption": "none"},
    "client_encryption_options": {"enabled": False},
    "endpoint_snitch": "SimpleSnitch",
    "seed_provider": [{
        "class_name": "org.apache.cassandra.locator.SimpleSeedProvid
        "parameters": [{"seeds": "127.0.0.1"}]
    }],
    "default_keyspace_rf": 1,
    "minimum_keyspace_rf": 1,
    "initial_token": "0",
    "file_cache_size_in_mb": 10,
})

# Allow the user to override any configuration options they want
extra_config = os.environ.get("CONFIG_OVERRIDE_JSON", "{}")
print("Found overrides: {}".format(extra_config))
cass_config.update(json.loads(extra_config))
```

~5s startup*

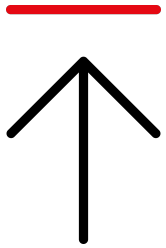
Easy to customize

Minimal memory footprint

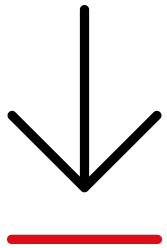
*skip_wait_for_gossip_to_settle=0, disable vnodes, etc ..

Proposal

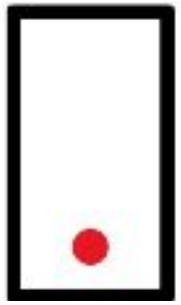
`source`	Do earlier suggestions
`packages`	Stop autostarting Cassandra packages
`containers`	Publish official testing container Publish official kubernetes/swarm containers



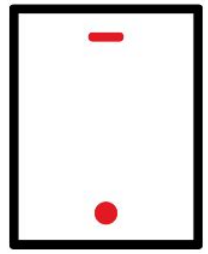
Develop



Distribute



Launch



Scale

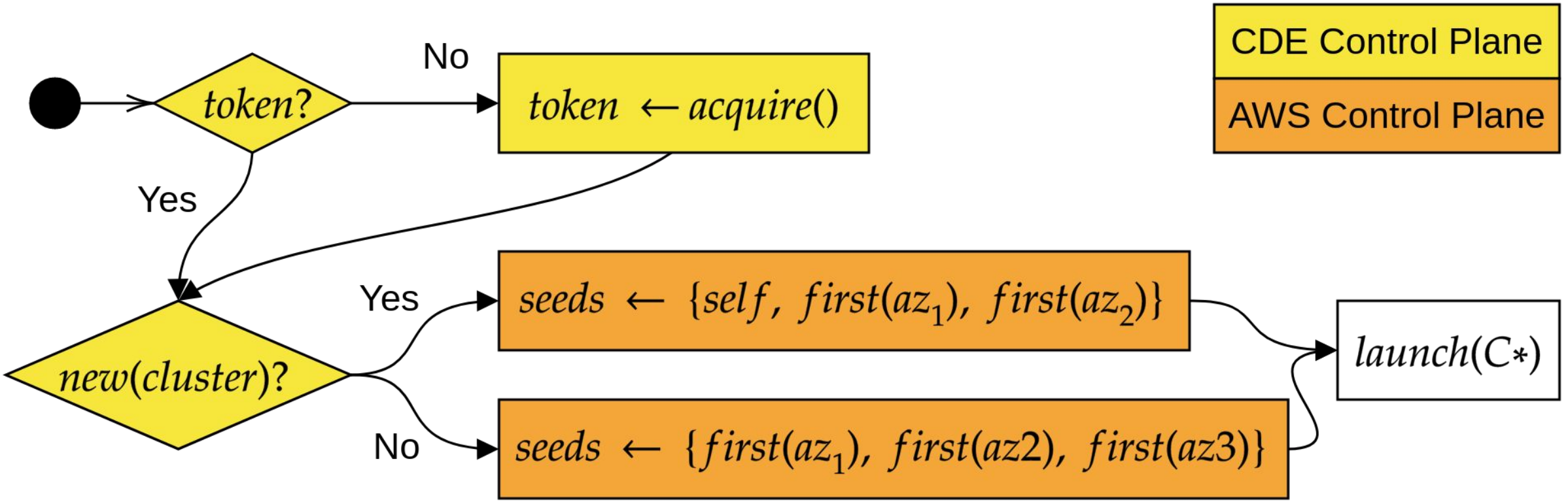
Launching Nodes

`seeds`	Which nodes will initiate the cluster
`tokens`	How will we assign tokens
`process`	How to manage Cassandra process

How does Netflix launch?

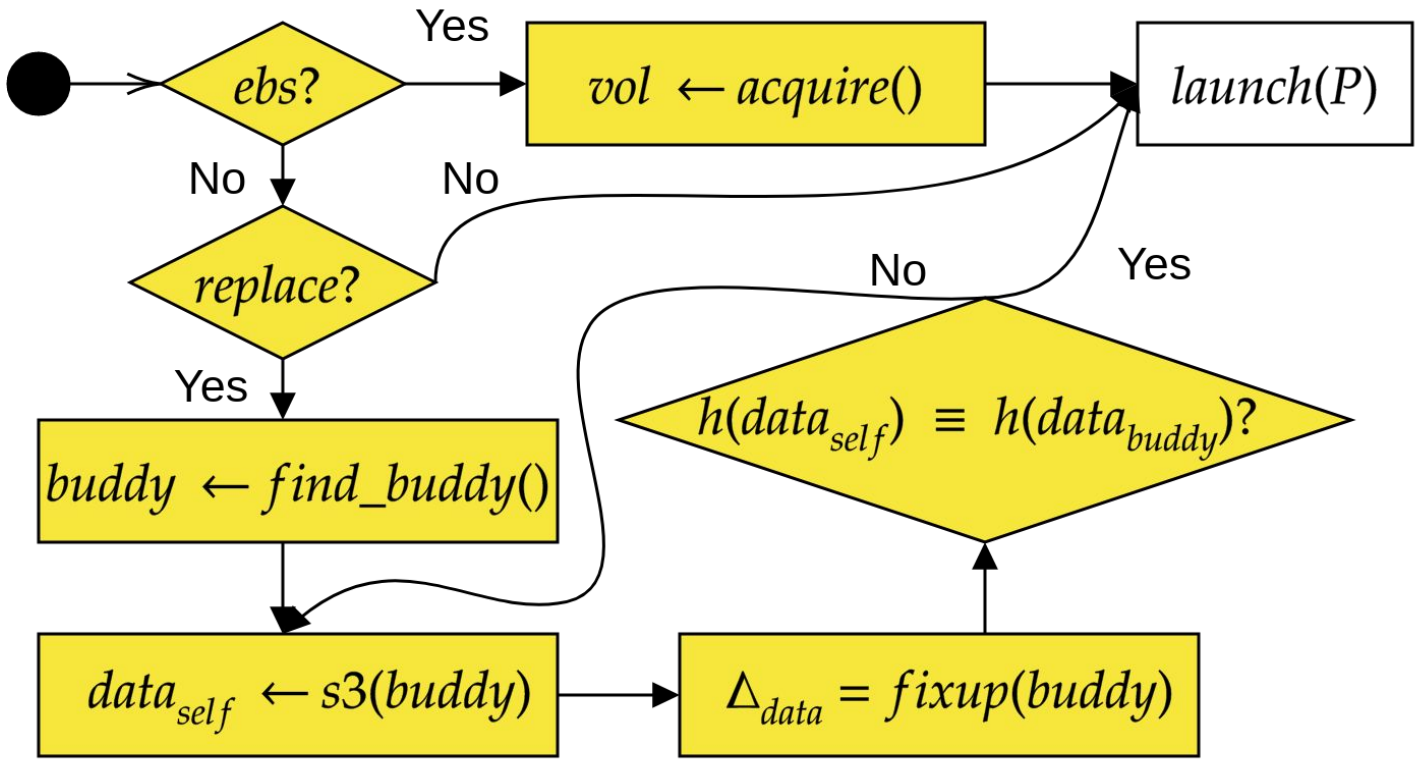
`seeds`	Ask AWS cloud control plane
`tokens`	Ask CDE cluster control plane
`process`	Use systemd

Declarative Control Plane



RIAM

Data Migration?



Avoid Cassandra Streaming

Use S3, Use EBS

Use direct file copy (sendfile)

Verify with xxHash

Data Migration: The Numbers

$$500 \frac{\text{GiB}}{node} \;/\; 200 \frac{\text{mb}}{second} \;=\; 5.96 \frac{hours}{node}$$

$$5.96 \frac{hours}{node} * 288\ node \;=\; \mathbf{2.35\ months}$$

Naive solution, sequential transfer

$$500\ \text{GiB} \;/\; 2000 \frac{\text{mb}}{second} \;=\; 0.6\ hours$$

$$0.6\ hours \;+\; 10\frac{minutes}{node} * 288\ node \;=\; \mathbf{2.02\ days}$$

Parallel S3 download sequential fixup

$$500\ \text{GiB} \;/\; 2000 \frac{\text{mb}}{second} \;=\; 0.6\ hours$$

$$0.6\ hours \;+\; 20\frac{minutes}{zone} * 3\ zone \;=\; \mathbf{1.596\ hours}$$

Parallel download and zone wide fixup

Process Control?

```
##/lib/systemd/system/cassandra.service

[Unit]
Description=Cassandra
# Don't ever give up on starting Cassandra
StartLimitInterval=0
StartLimitIntervalSec=1800

[Service]
RuntimeDirectory=cassandra
ExecStart=/usr/bin/cassandra_wrapper
LimitNOFILE=1048576
LimitNPROC=131072
LimitMEMLOCK=infinity
# always restart Cassandra
# If you want to stop Cassandra, either do:
# * nodetool drain
# * systemctl stop cassandra
Restart=always
RestartSec=30
StartLimitBurst=4
```

Just use systemd

Handles automatic restarts

Handles console logs

It’s great

Process Control?

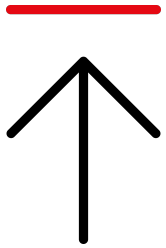
```
## /usr/bin/cassandra_wrapper
# OS Page Cache Loading
cd ${CASSANDRA_HOME/data}
set +e

if [ -e .happycache.gz ]; then
  happycache load
  # Delete the OS cache file even if the load fails
  rm -f .happycache.gz
fi
set -e

## Pid file for preventing multiple processes
## starting at once

PIDFILE=/run/cassandra/cassandra.pid
exec 3<> "${PIDFILE}"

if ! flock -n 3; then
  PID=`cat ${PIDFILE}`
  echo "Cassandra already running (best guess in PID:
${PID}), cannot start"
  exit 1
fi

echo $$ > ${PIDFILE}

# Start Cassandra
...
```

Reload OS page cache with happycache

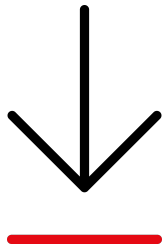
Guarantee single process running with flocked pidfiles

Proposal

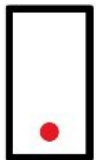
`seeds`	Sidecar + native cloud providers
`tokens`	Add TokenProvider interface, should happen for (#13701). More full SSTable streaming.
`process`	Just add a systemd unit



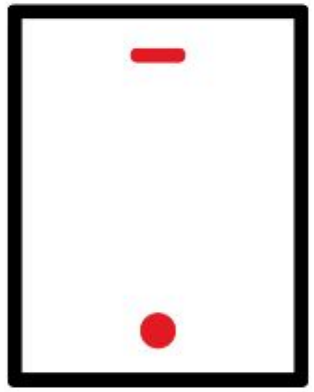
Develop



Distribute



Launch



Scale

Scaling Cassandra

`metrics`	Operational insights into performance
`monitor`	What is happening with the cluster
`integrity`	Data integrity (failure, corruption)
`administer`	Must be easy to configure and restart

Scaling Cassandra

`metrics`

`monitor`

`backup`

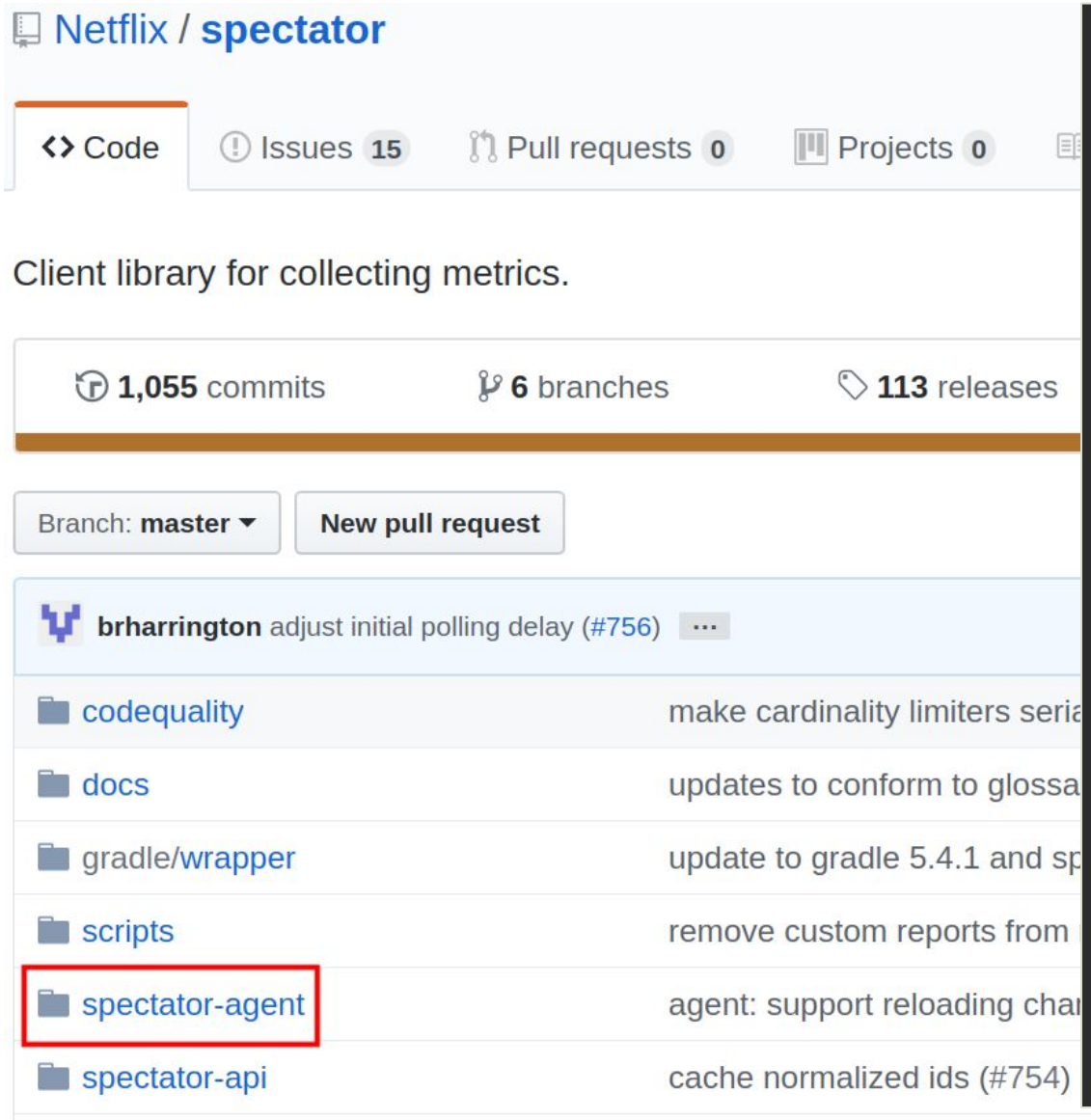
`administer`

Interoperable with … JMX, but not

- statsd
- prometheus
- spectator

Performant export is a problem

How does Netflix get metrics?




```
# cassandra-env.sh patch
# Pull in any agents present in CASSANDRA_HOME

for agent_file in ${CASSANDRA_HOME}/agents/*.jar; do
  if [ -e "${agent_file}" ]; then
    base_file="${agent_file%.jar}"
    if [ -s "${base_file}.options" ]; then
      options=`cat ${base_file}.options`
      agent_file="${agent_file}=${options}"
    fi
    JVM_OPTS="$JVM_OPTS -javaagent:${agent_file}"
  fi
done

for agent_file in ${CASSANDRA_HOME}/agents/*.so; do
  if [ -e "${agent_file}" ]; then
    base_file="${agent_file%.so}"
    if [ -s "${base_file}.options" ]; then
      options=`cat ${base_file}.options`
      agent_file="${agent_file}=${options}"
    fi
    JVM_OPTS="$JVM_OPTS -agentpath:${agent_file}"
  fi
done
```

Proposal

`metrics`	Allow pluggability of histogram
`monitor`	Official exporter agents + pluggable agents
`integrity`	
	HTTP metrics endpoint on sidecar
`administer`	

Scaling Cassandra

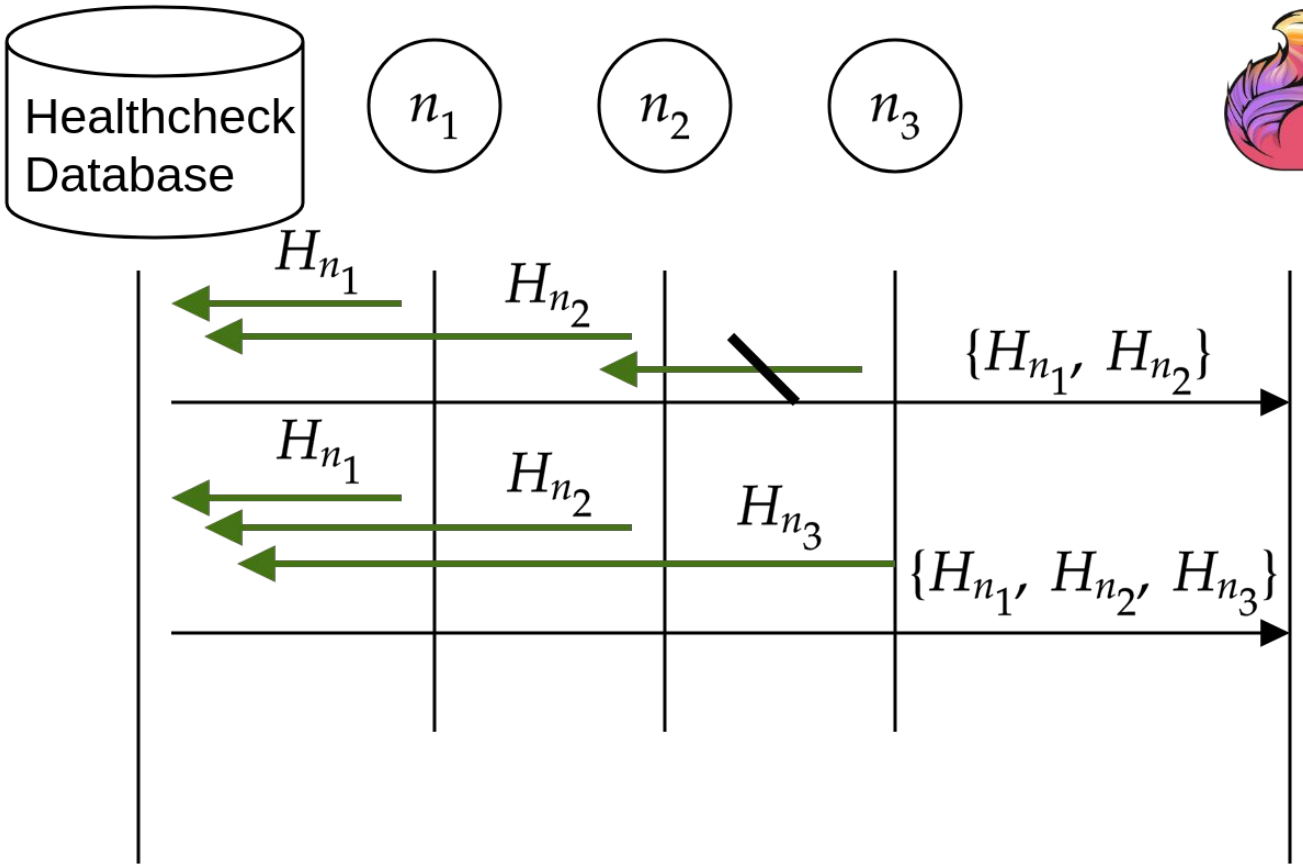
metrics

monitor — No real way to know cluster view

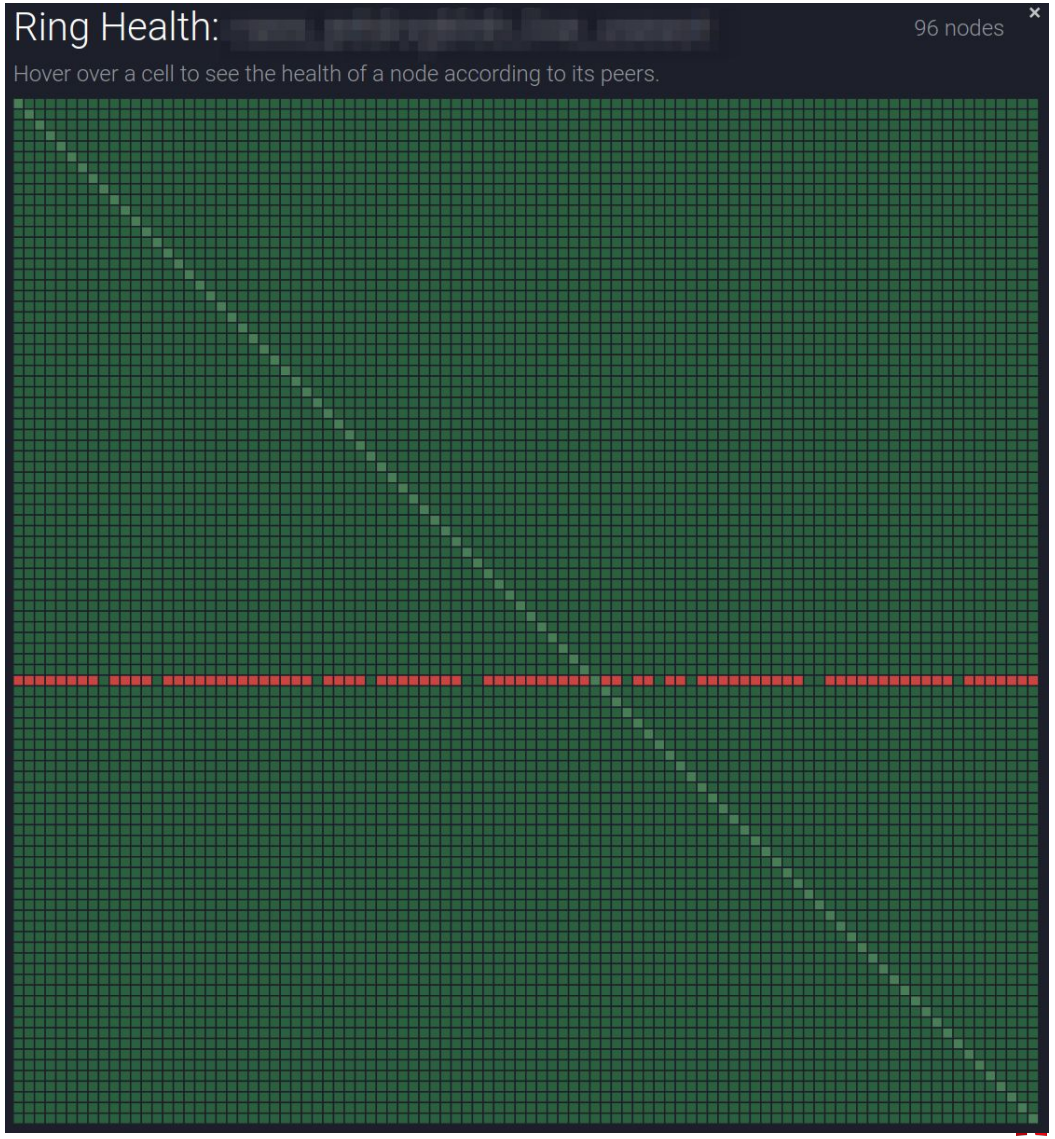
integrity — Is the ring healthy?
Where is my data located?

administer

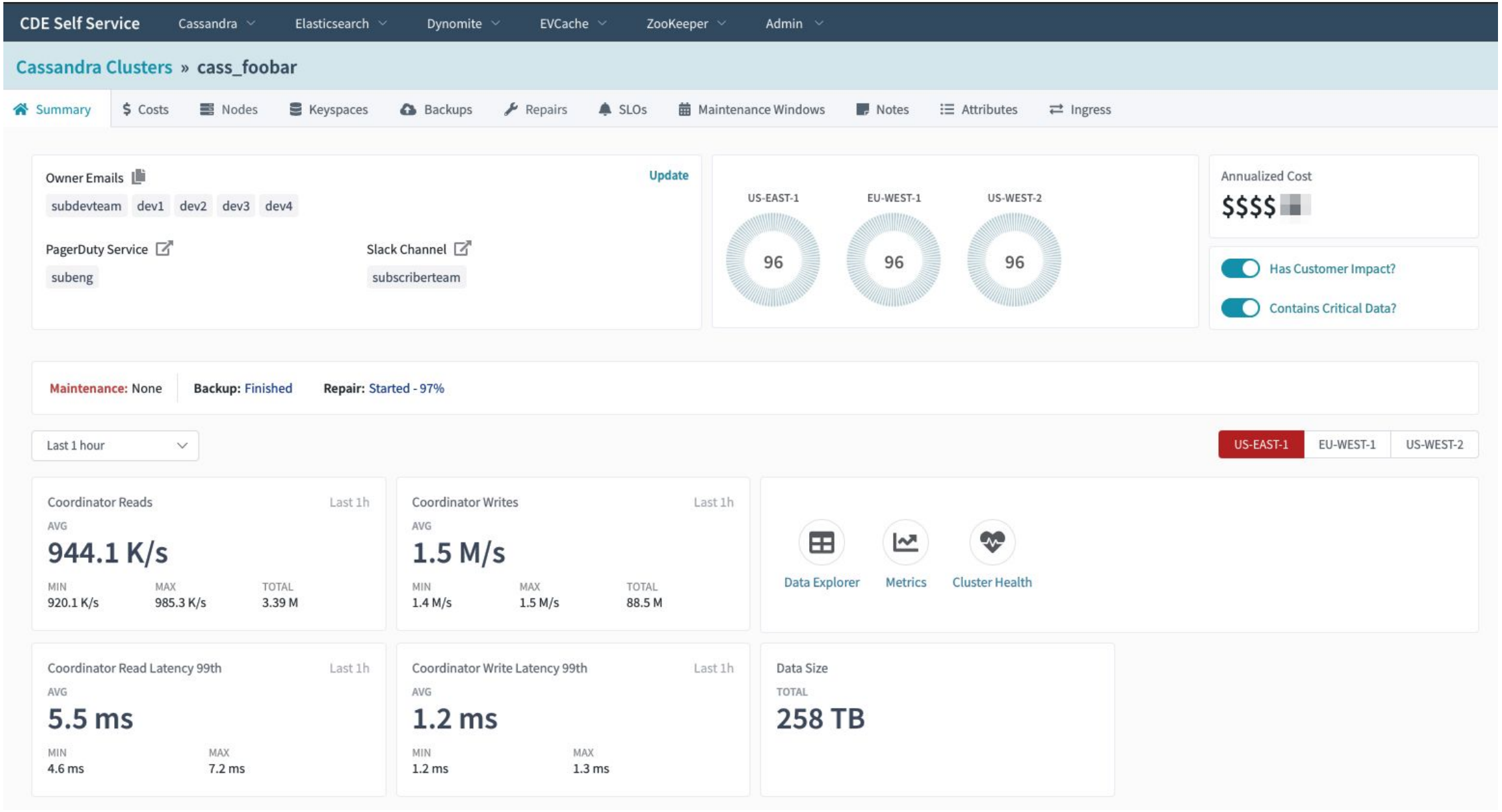
How does Netflix monitor?



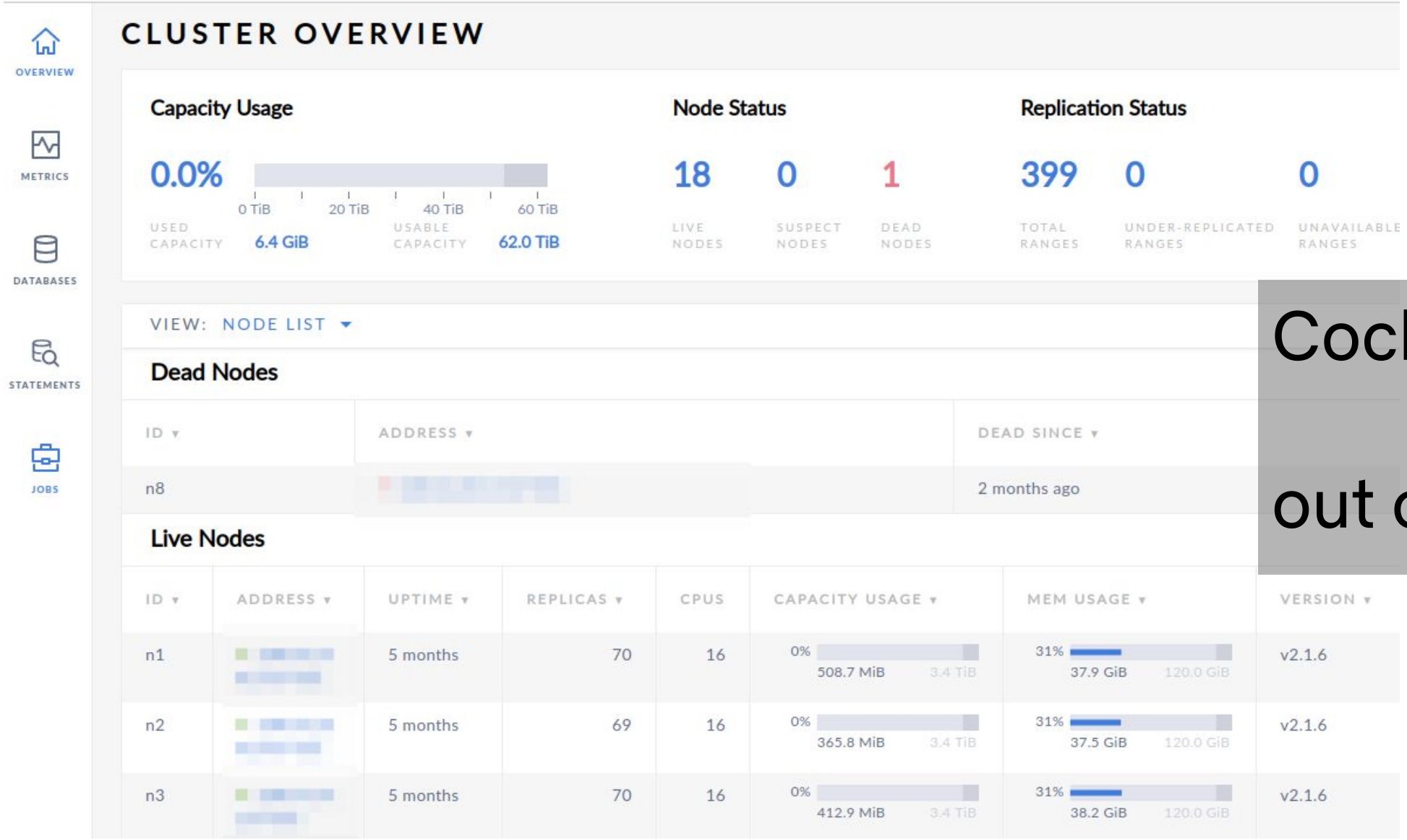
Cluster Health



How does Netflix monitor?



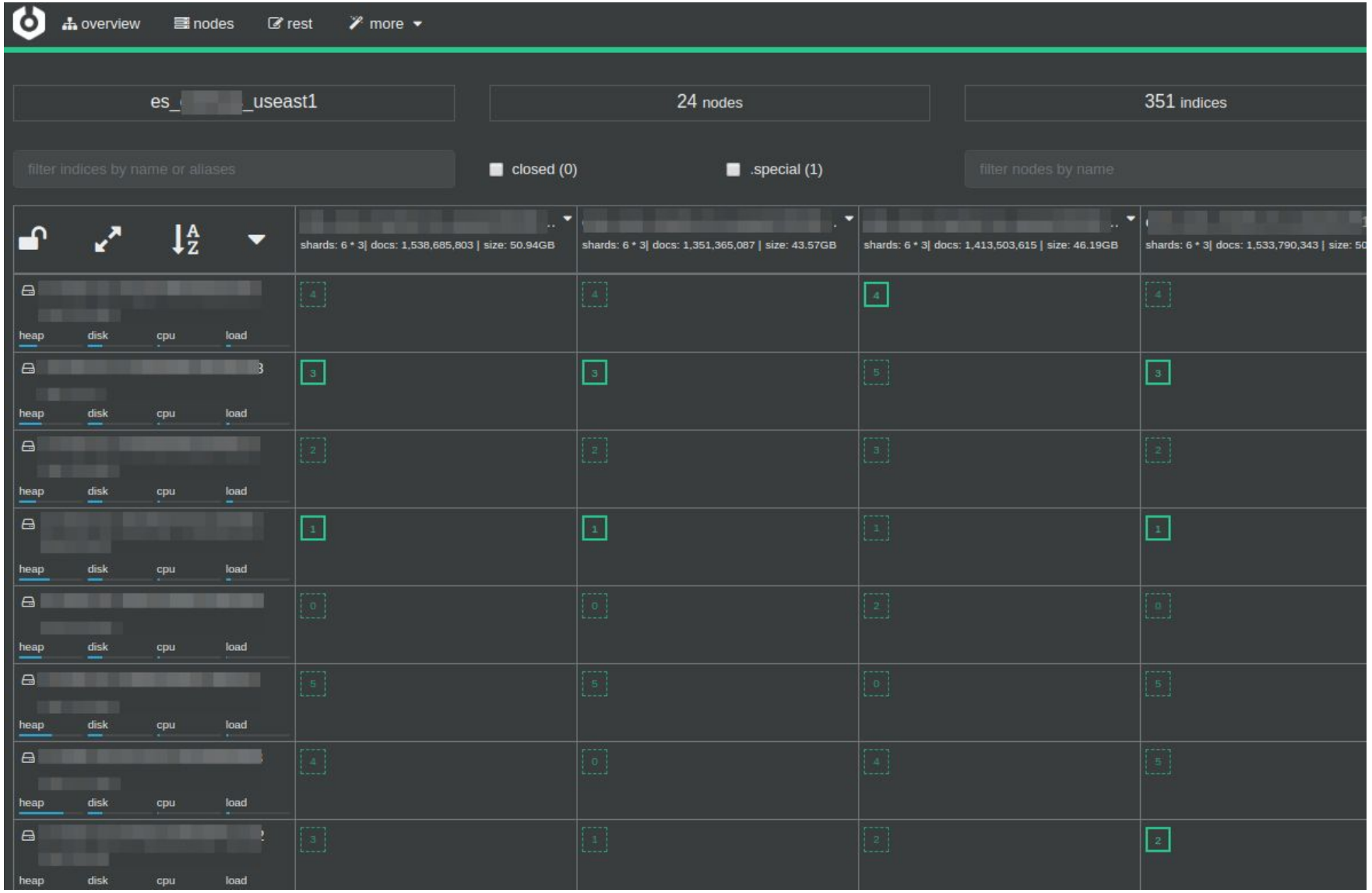
How do other DBs do this?



CockroachDB

out of the box

How do other DBs do this?



Elasticsearch plugs in Cerebro

Proposal

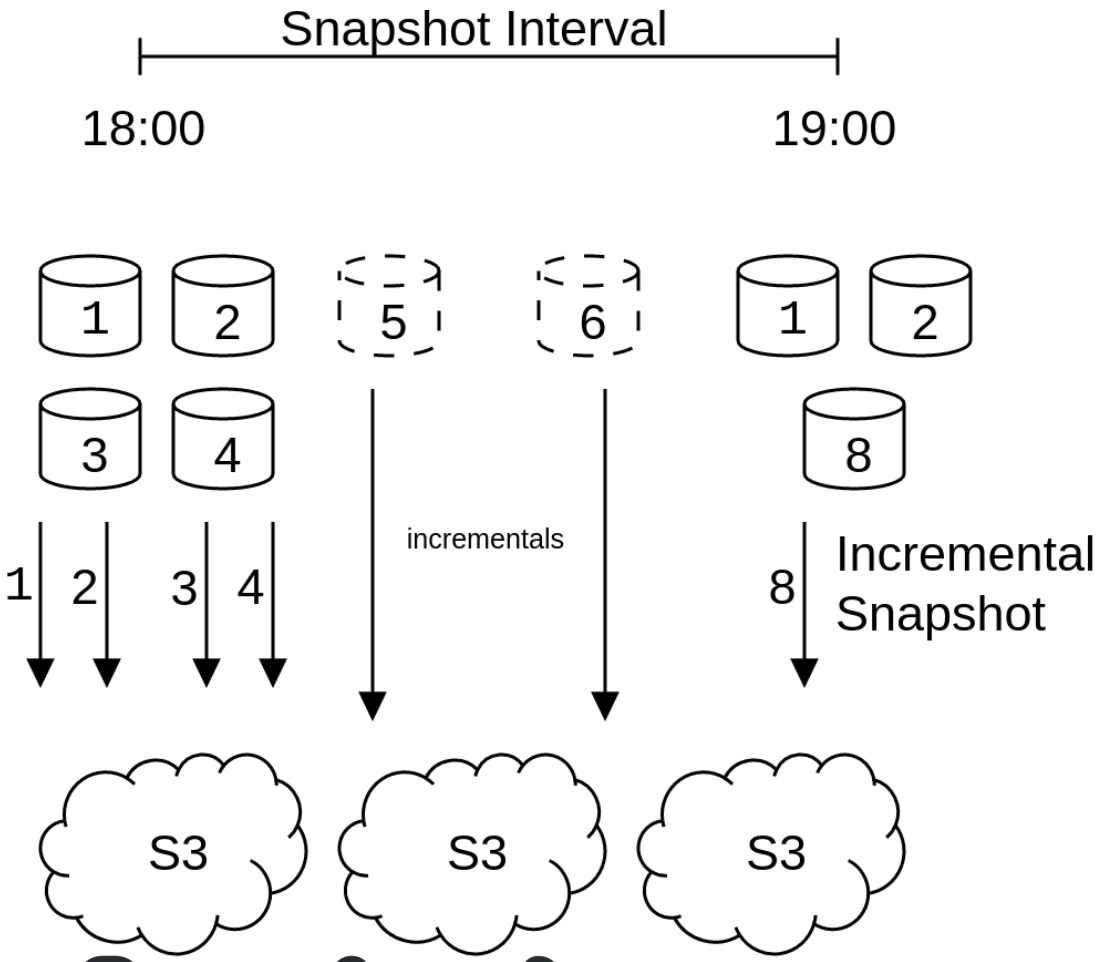
`metrics`	Ship out of the box health and status dashboards
`monitor`	JSON metrics/status may be sufficient
`integrity`	
`administer`	Sidecar may be a great place for this

Scaling Cassandra

`metrics`	Nodes and clusters constantly failing
`monitor`	
`integrity`	Lack critical backup, restore and repair scheduling functionality
`administer`	No standard way to configure and restart nodes in a cluster

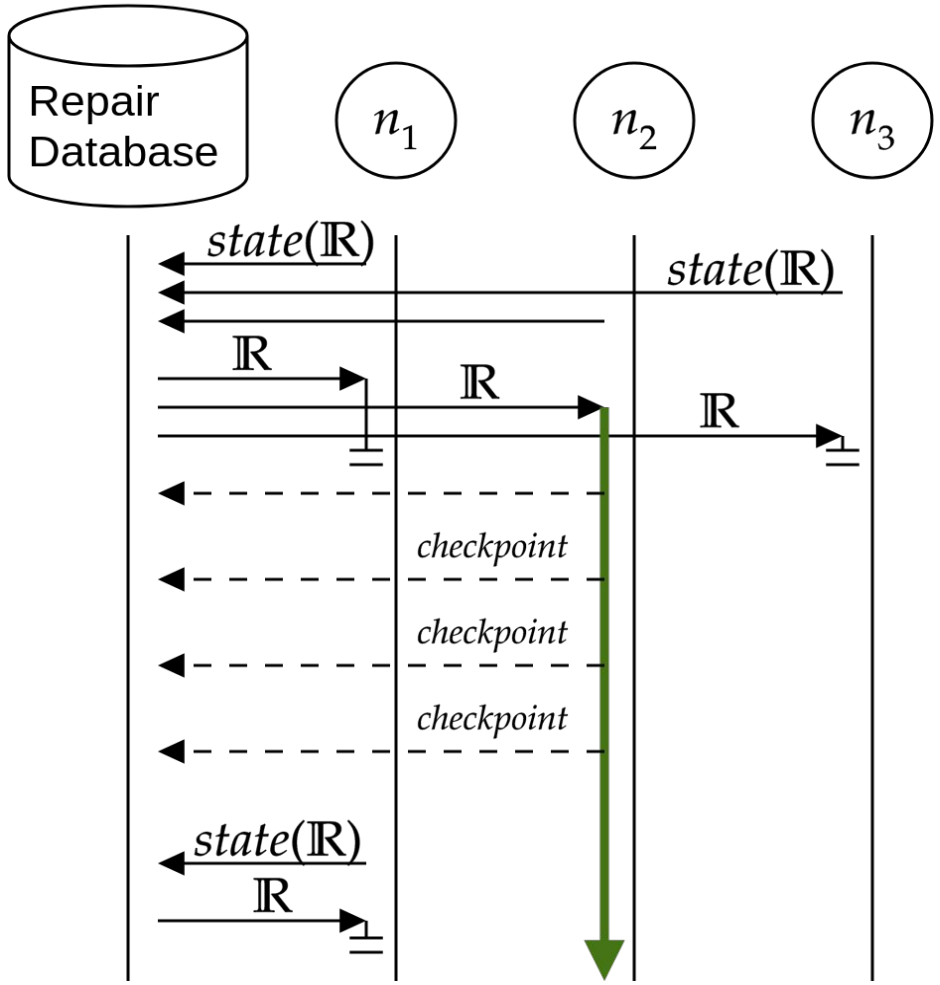
How does Netflix backup?

- Point in time
  - Incremental snapshot
  - Optimal performance
- Plugins for different clouds



RIAM

How does Netflix repair?

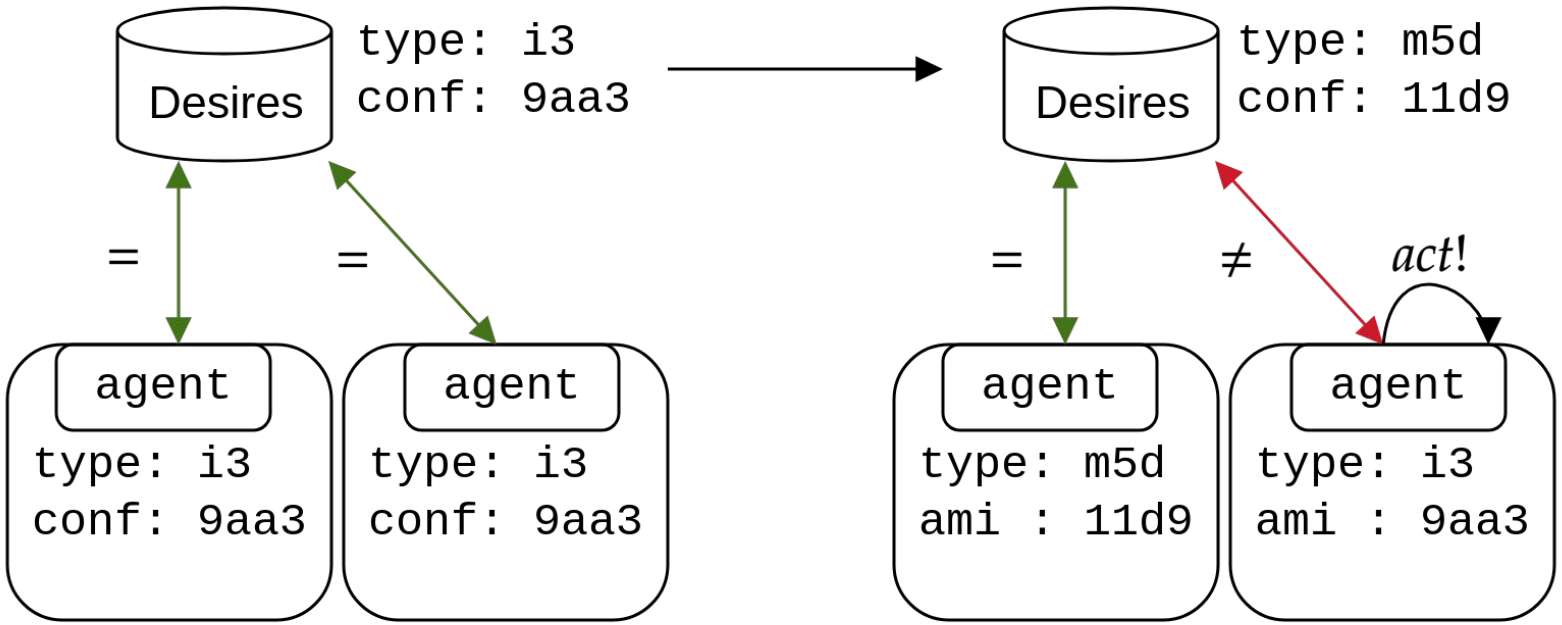


— Distributed Control Plane

— Every node follows **repair state machine (#14346)**

RIAM

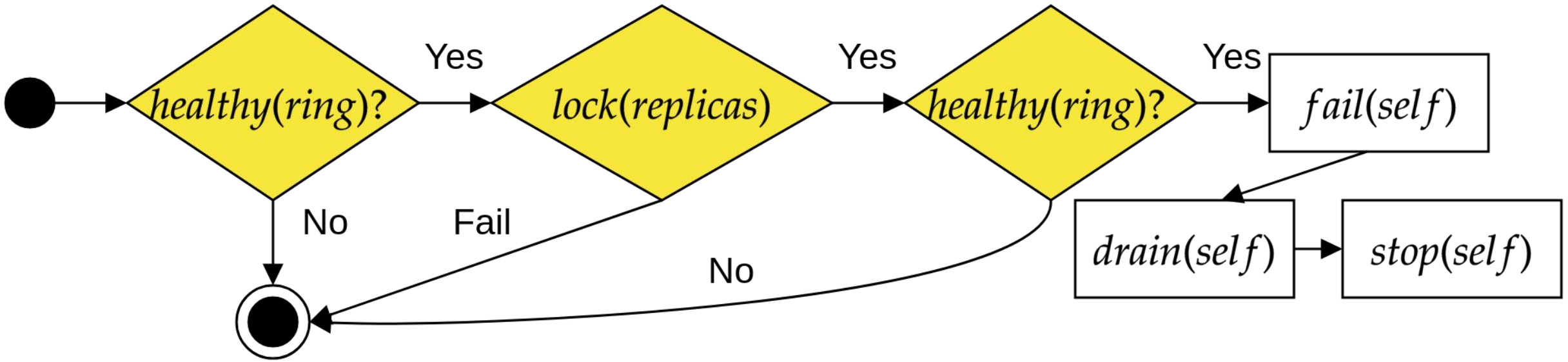
How does Netflix administer?



Declarative

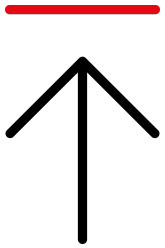
Desire based

Distributed State Machines

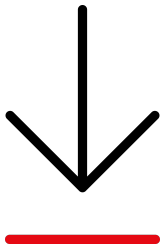


Proposal

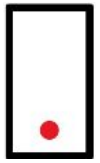
`metrics`	Add native cloud backup plugins
`monitor`	Continue investing in Apache Cassandra Sidecar project
`integrity`	Backups Repairs
`administer`	Restarts



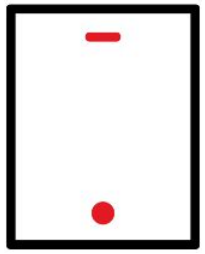
Develop



Package



Starting



Scaling

Our users need better solutions

We need the community's help!

With targeted investment we can solve this problem

Discussion?