The Walking Dead

A Survival Guide to Resilient Reactive Applications

Michael Nitschinger

lambda DAAS

@daschl

Couchbase

the right

Mindset

"The more you sweat in peace, the less you bleed in war."

– U.S. Marine Corps

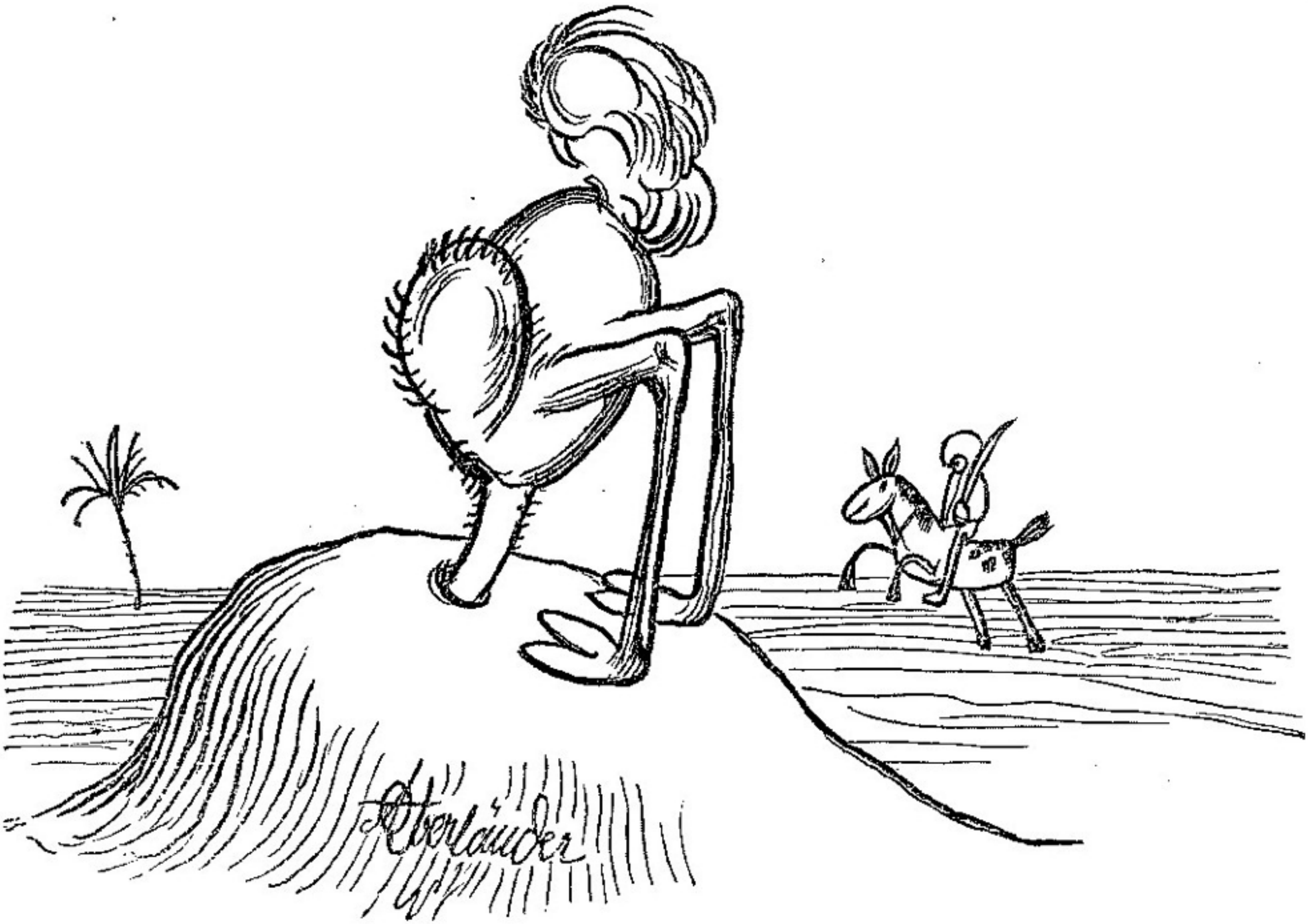
DevOps Borat
@DEVOPS_BORAT

Follow

90% of devops are fail simple interview question of what is most critical piece of infrastructure. Is outgoing mail on Nagios server!

RETWEETS 249 FAVORITES 81

7:02 PM - 15 Jan 2013



ALL 163 | CHANGED 0 | SOURCE CHANGED 0 | COVERAGE CHANGED 0

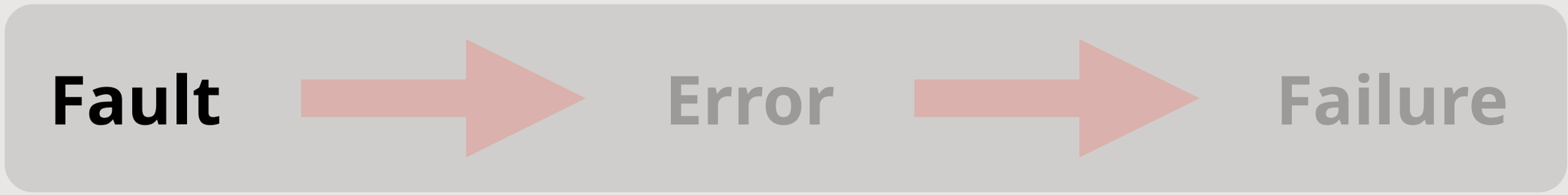
COVERAGE	FILE	LINES	RELEVANT	COVERED
100.0	app/helpers/enterprise_helper.rb	2	1	1
100.0	config/boot.rb	6	3	3
100.0	config/environment.rb	5	2	2
100.0	app/concerns/storyteller.rb	22	12	12
100.0	config/initializers/airbrake.rb	5	3	3
100.0	config/initializers/01_load_config.rb	3	3	3

Not so fast, mister fancy tests!

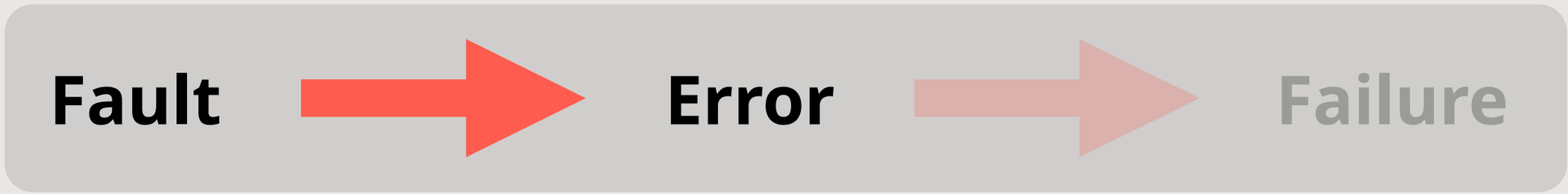
Always ask yourself

What can go wrong?

Fault Tolerance 101



A fault is a **latent defect** that can **cause an error** when activated.



Errors are the manifestations of faults.

Failure occurs when **the service no longer complies with its specifications**.

Errors are inevitable. We need to **detect, recover and mitigate** them before they become failures.

Reliability

is the probability that a system will perform failure free for a given amount of time.

MTTF Mean Time To Failure

MTTR Mean Time To Repair

Availability

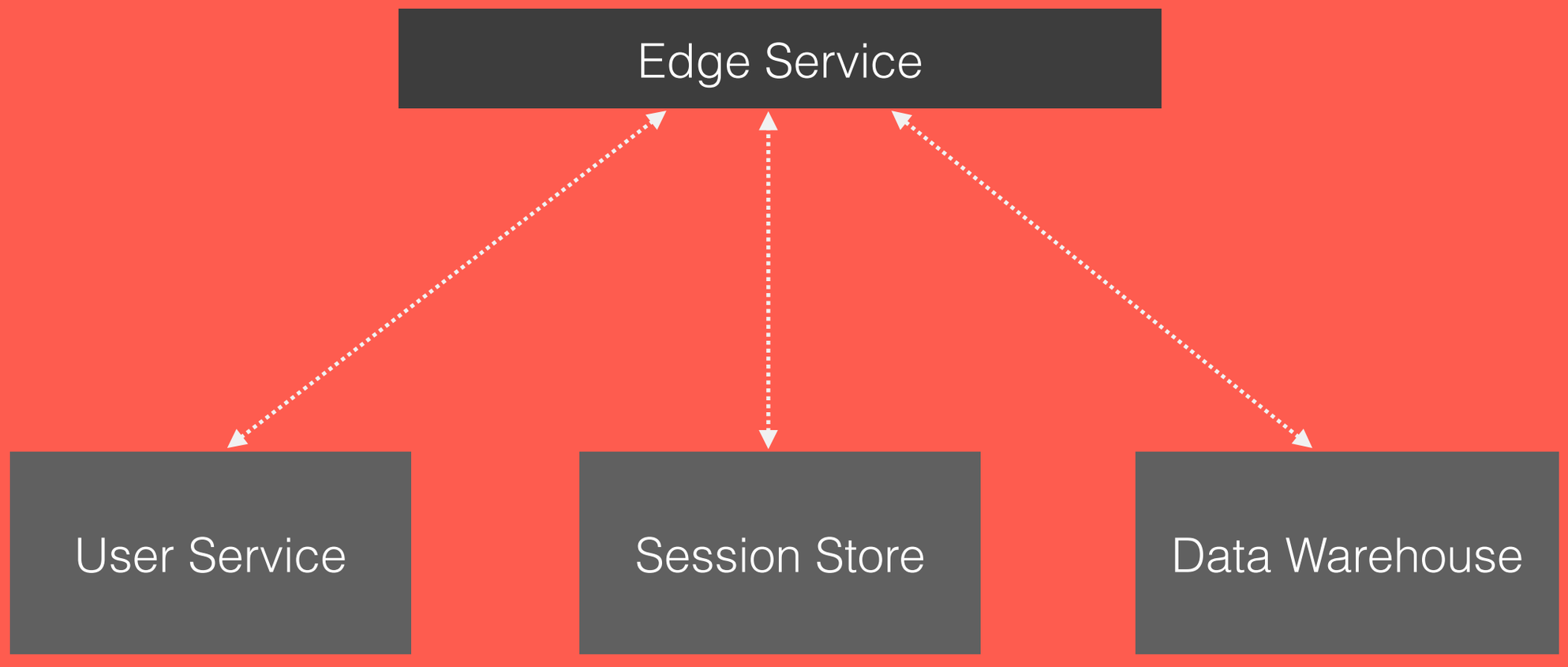
is the percentage of time the system is able to perform its function.

$$\text{availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

Expression		Downtime/Year
Three 9s	99.9%	525.6 min
Four 9s	99.99%	52.56 min
Four 9s and a 5	99.995%	26.28 min
Five 9s	99.999%	5.256 min
Six 9s	99.9999%	0.5256 min
	100%	0

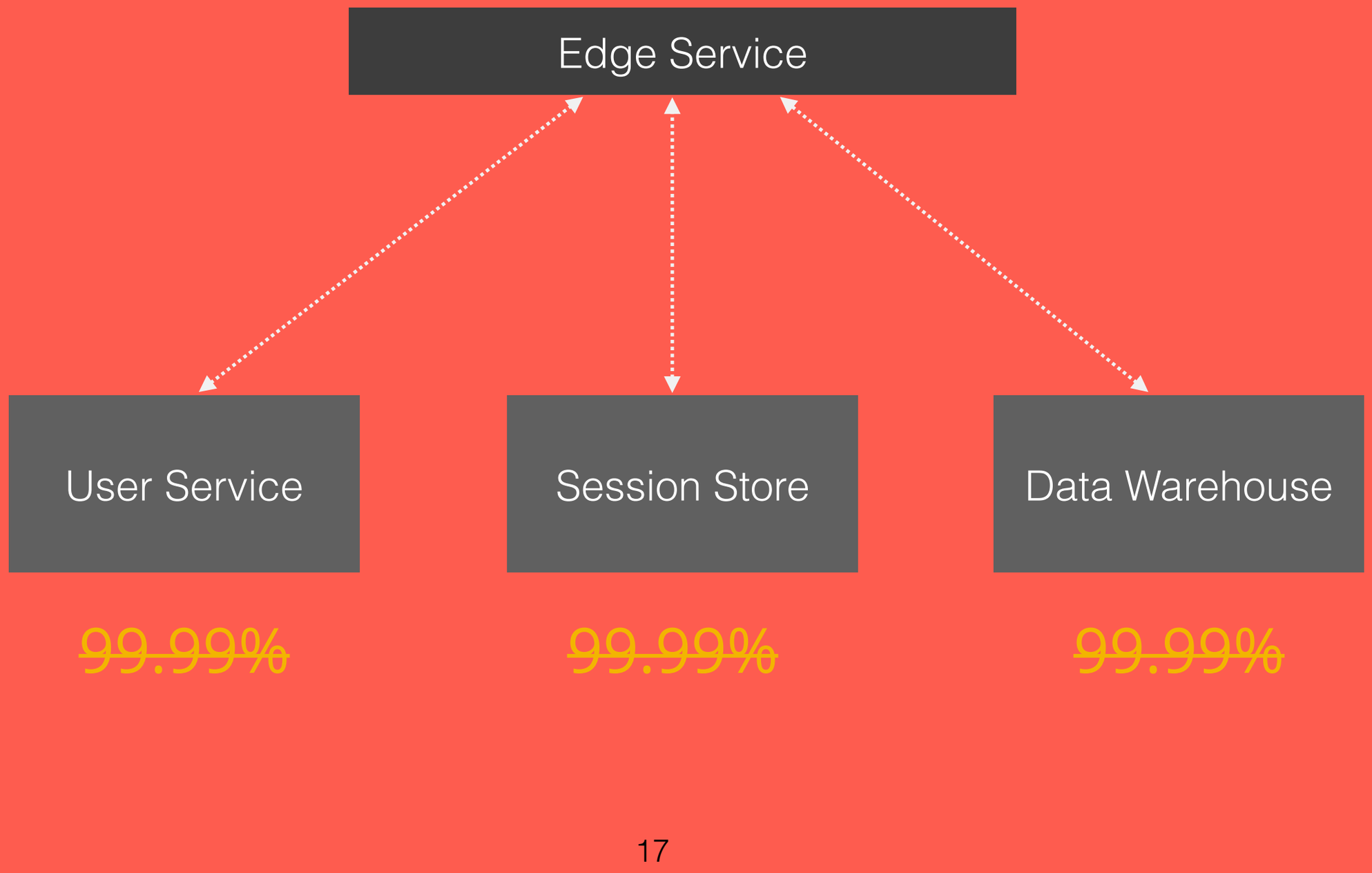
Pop Quiz!

Wanted: 99.99% Availability



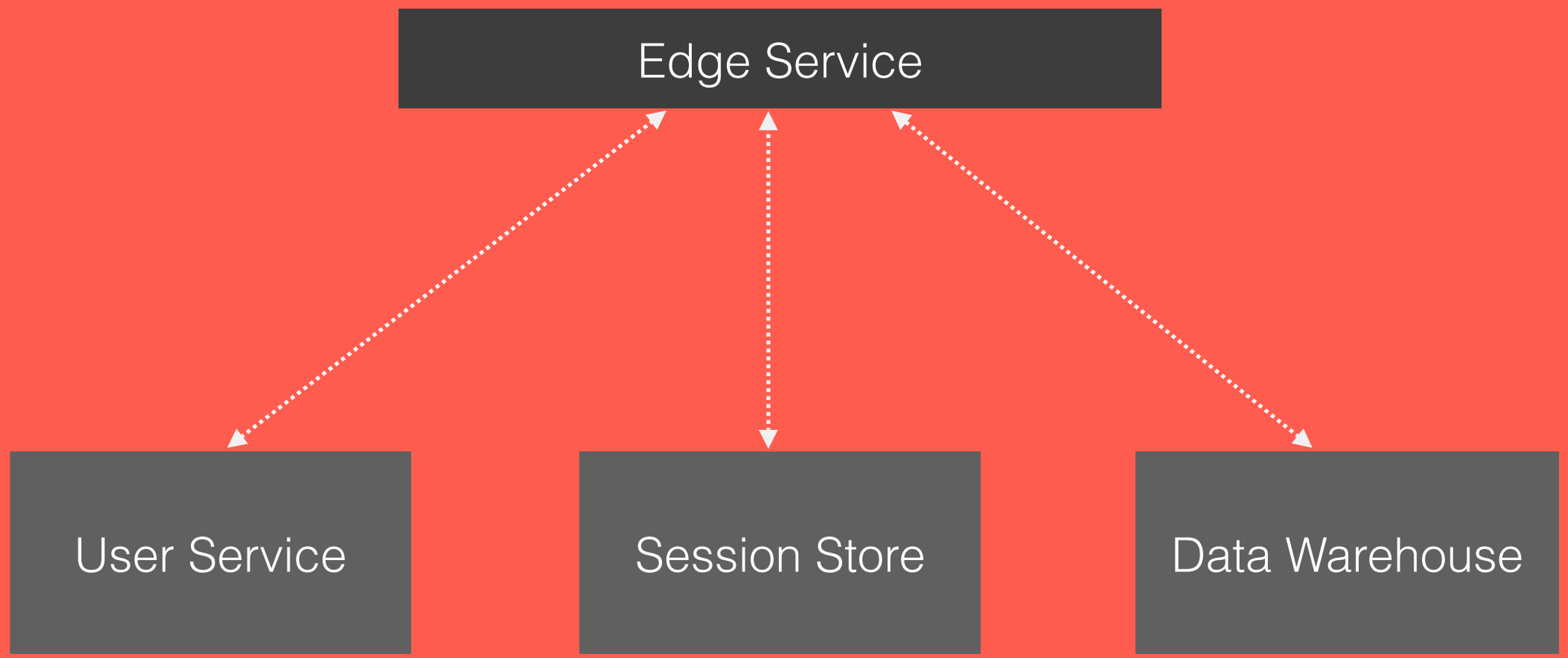
Pop Quiz!

Wanted: 99.99% Availability



Pop Quiz!

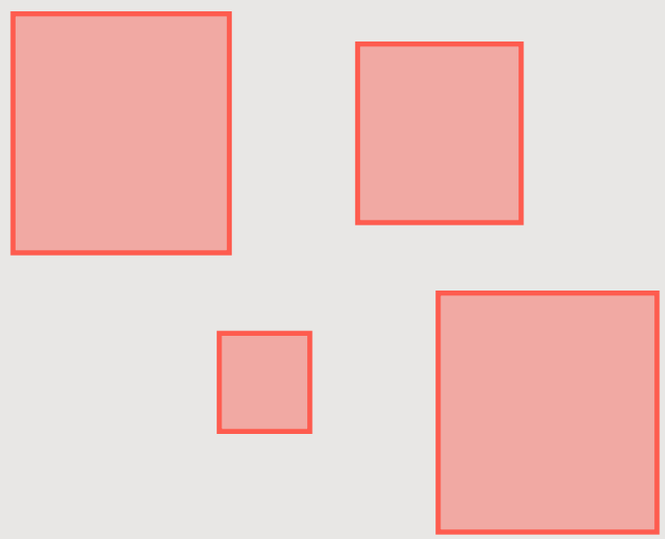
Wanted: 99.99% Availability



Fault Tolerant Architecture

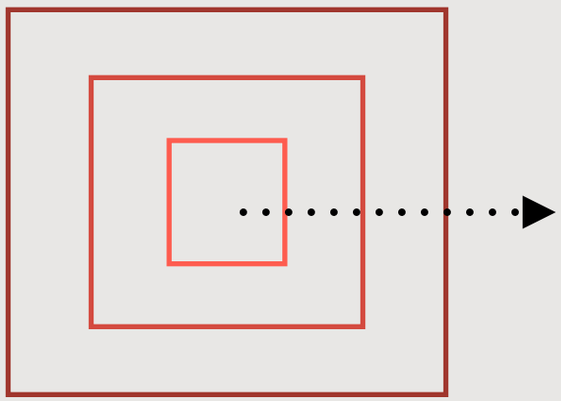
Units of Mitigation

are the basic units of
error containment and recovery.

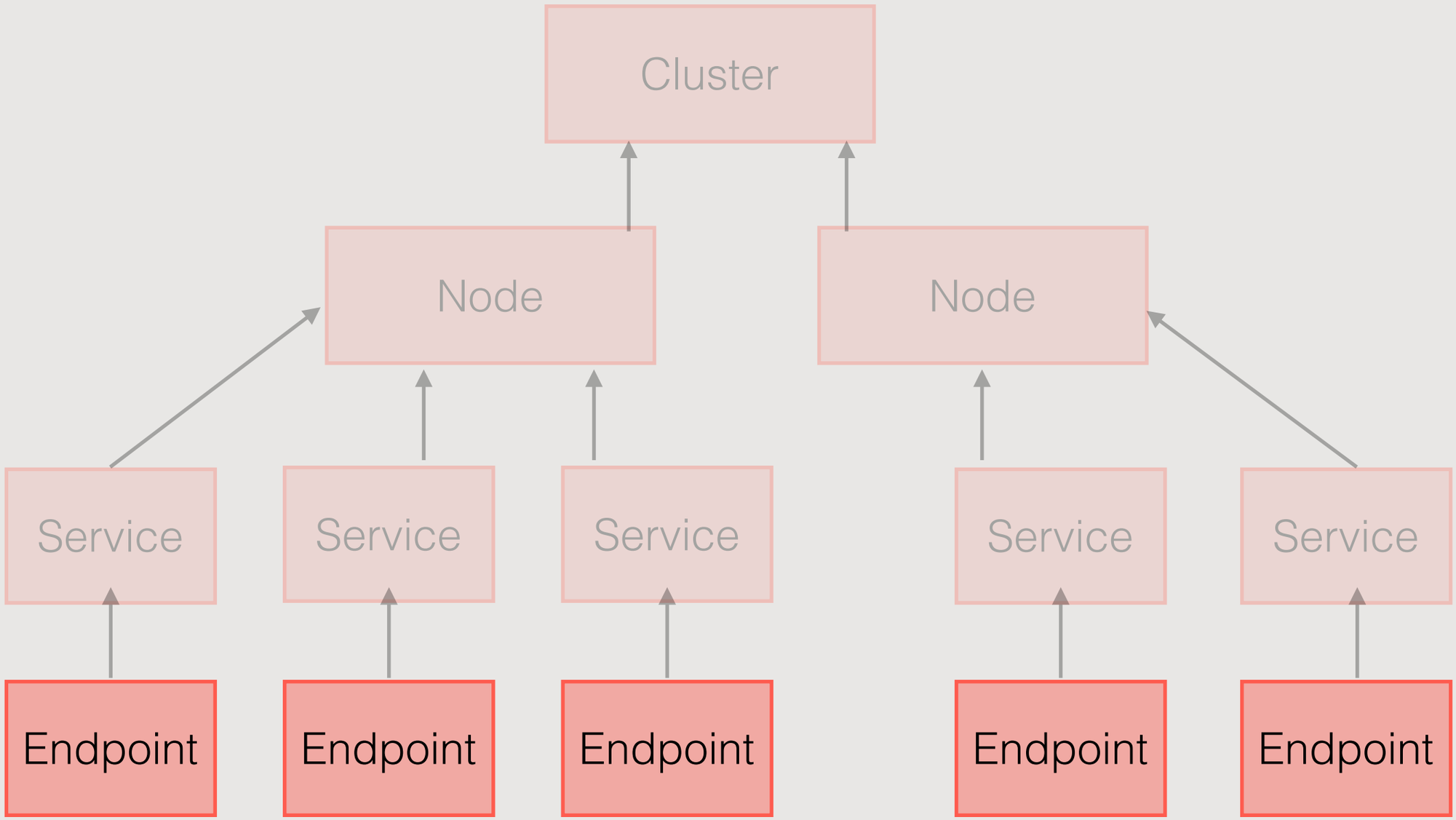


Escalation

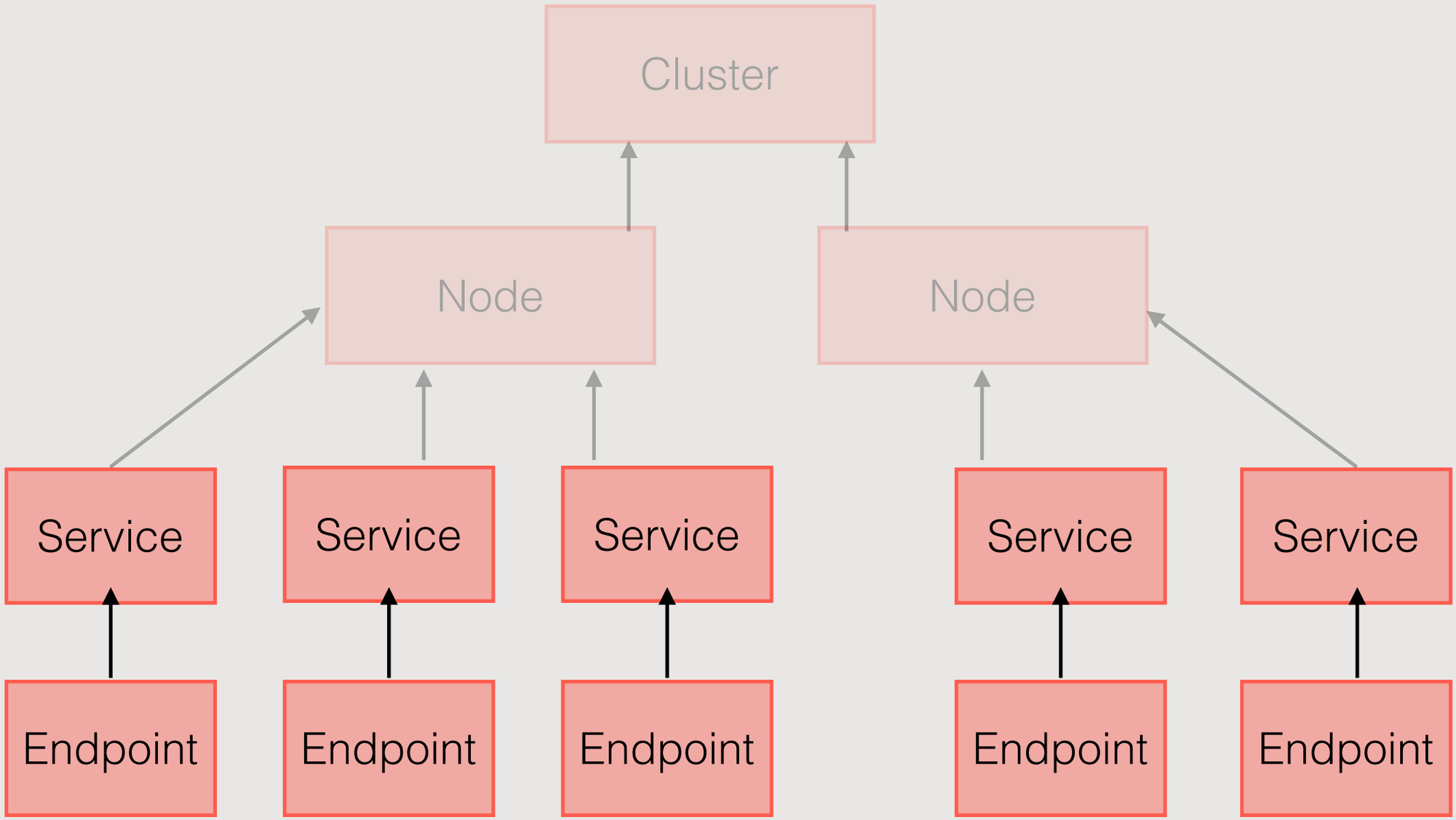
is used when recovery or mitigation is not possible inside the unit.



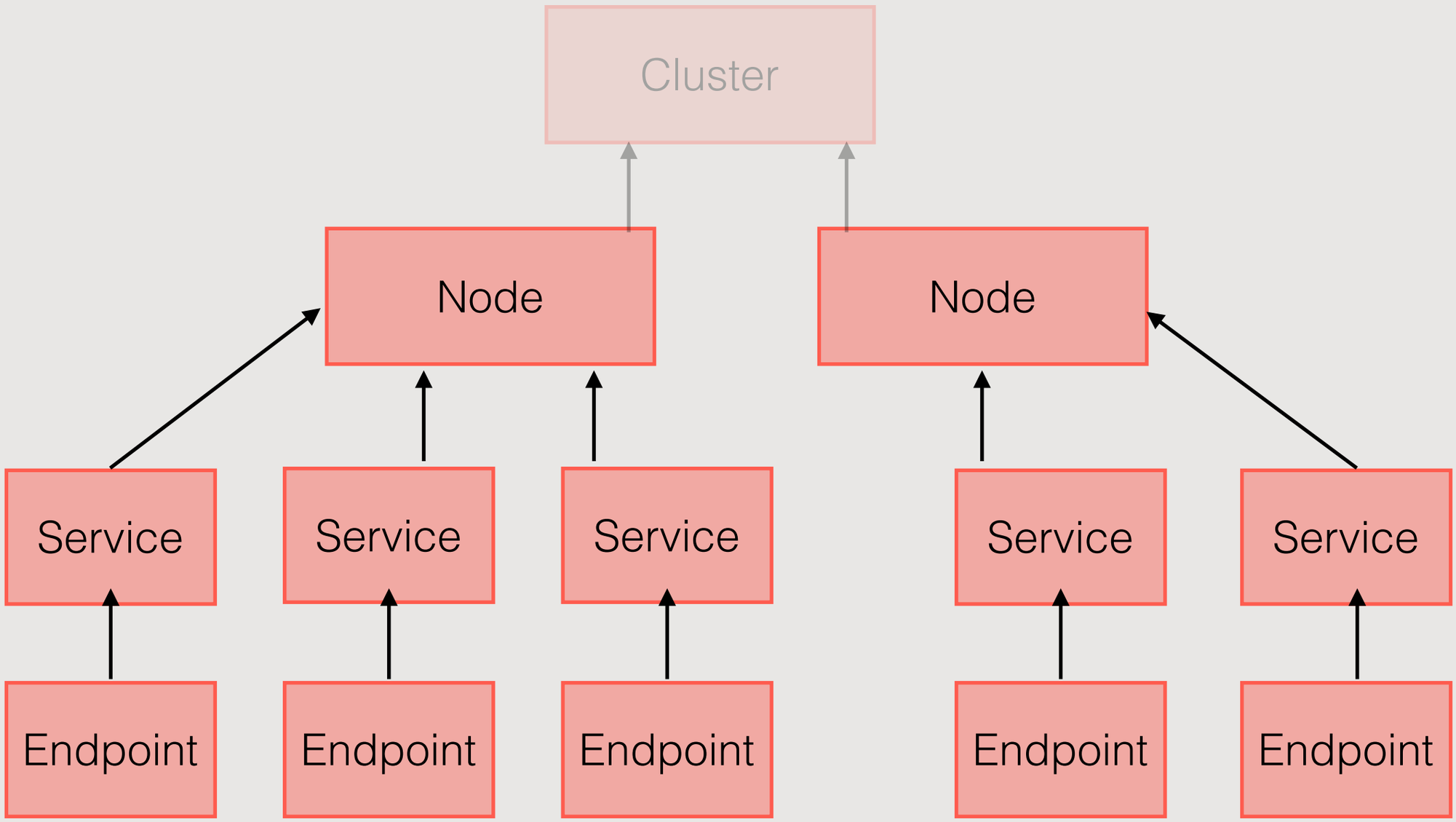
Escalation


Cluster
Node
Node
Service
Service
Service
Service
Service
Endpoint
Endpoint
Endpoint
Endpoint
Endpoint

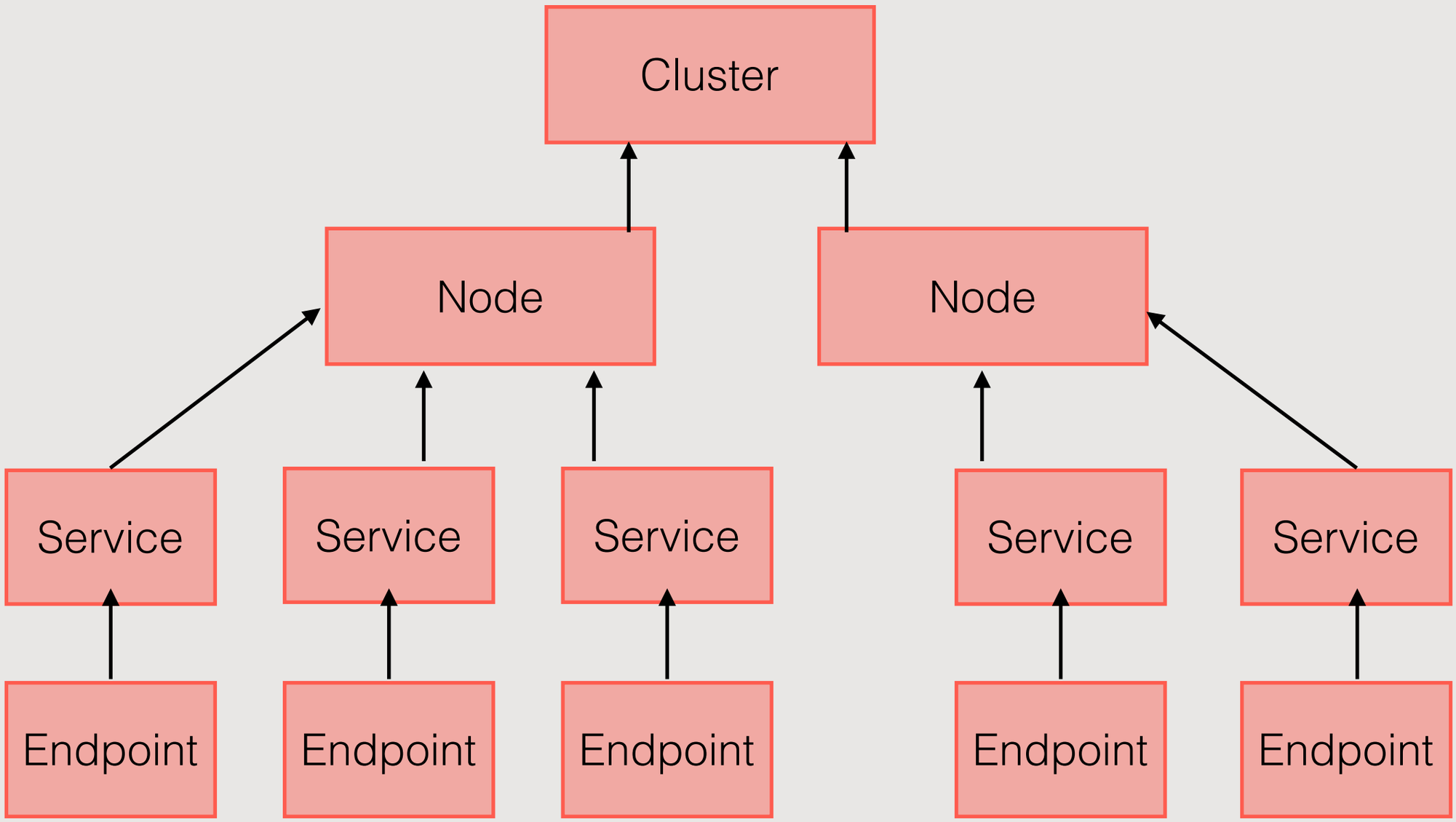
Escalation


Cluster
Node
Node
Service
Service
Service
Service
Service
Endpoint
Endpoint
Endpoint
Endpoint
Endpoint

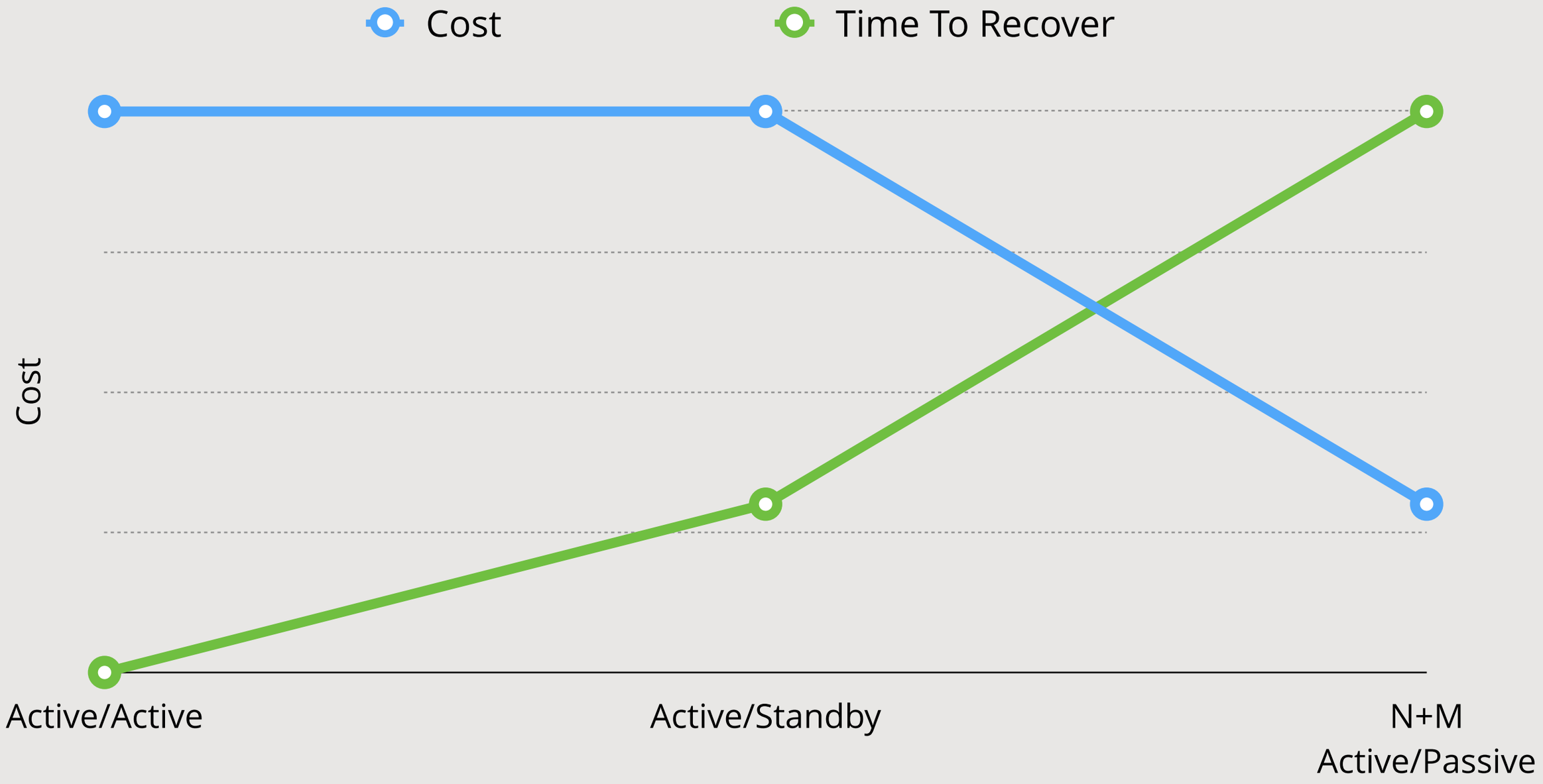
Escalation


Cluster
Node
Node
Service
Service
Service
Service
Service
Endpoint
Endpoint
Endpoint
Endpoint
Endpoint

Escalation

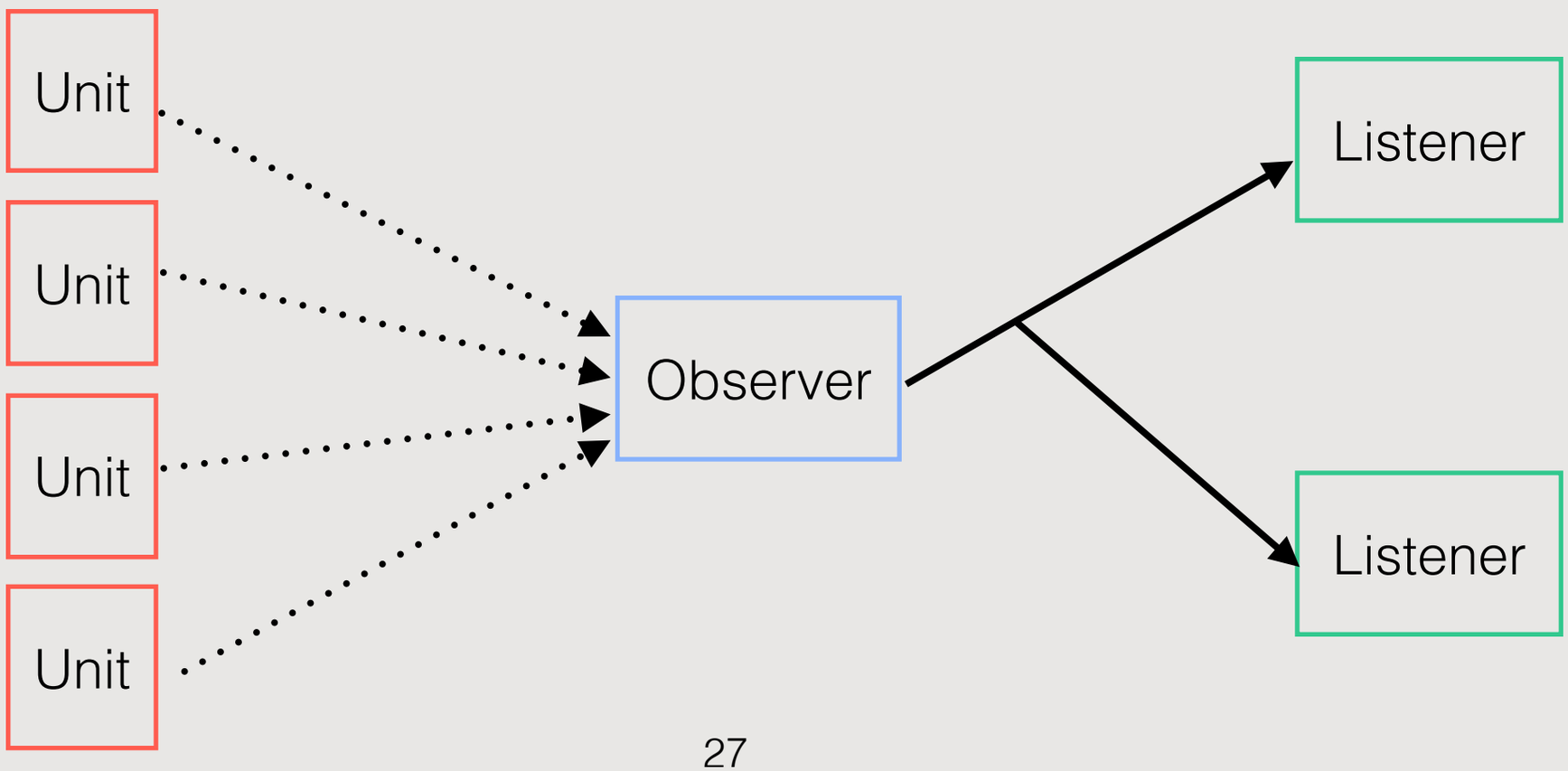

Cluster
Node
Node
Service
Service
Service
Service
Service
Endpoint
Endpoint
Endpoint
Endpoint
Endpoint

Redundancy


Cost
Time To Recover
Cost
Active/Active
Active/Standby
N+M
Active/Passive

The Fault Observer

receives system and error events *and can guide and orchestrate detection and recovery*



```
// Connect to the cluster
CouchbaseEnvironment environment = DefaultCouchbaseEnvironment.create();
CouchbaseCluster cluster = CouchbaseCluster.create(environment);

// Subscribe and just print out all events
environment
    .eventBus()
    .get()
    .subscribe(System.err::println);

Bucket bucket = cluster.openBucket();
cluster.disconnect();
```

```
2015-02-05 10:20:28 INFO  Node:212 - Connected to Node localhost
NodeConnectedEvent{node=localhost/127.0.0.1}
2015-02-05 10:20:29 INFO  ConfigurationProvider:264 - Opened bucket default
BucketOpenedEvent{name='default'}
BucketClosedEvent{name='default'}
NodeDisconnectedEvent{node=localhost/127.0.0.1}
NodeDisconnectedEvent{node=localhost/127.0.0.1}
2015-02-05 10:20:29 INFO  ConfigurationProvider:285 - Closed bucket default
2015-02-05 10:20:29 INFO  Node:222 - Disconnected from Node localhost
```

```
// Connect to the cluster
CouchbaseEnvironment environment = DefaultCouchbaseEnvironment.create();
CouchbaseCluster cluster = CouchbaseCluster.create(environment);

// Subscribe and just print out all events
environment
    .eventBus()
    .get()
    .subscribe(System.err::println);

Bucket bucket = cluster.openBucket();
cluster.disconnect();
```

```
2015-02-05 10:20:28 INFO  Node:212 - Connected to Node localhost
NodeConnectedEvent{node=localhost/127.0.0.1}
2015-02-05 10:20:29 INFO  ConfigurationProvider:264 - Opened bucket default
BucketOpenedEvent{name='default'}
BucketClosedEvent{name='default'}
NodeDisconnectedEvent{node=localhost/127.0.0.1}
NodeDisconnectedEvent{node=localhost/127.0.0.1}
2015-02-05 10:20:29 INFO  ConfigurationProvider:285 - Closed bucket default
2015-02-05 10:20:29 INFO  Node:222 - Disconnected from Node localhost
```

Detecting Errors

A silent system is a dead system.

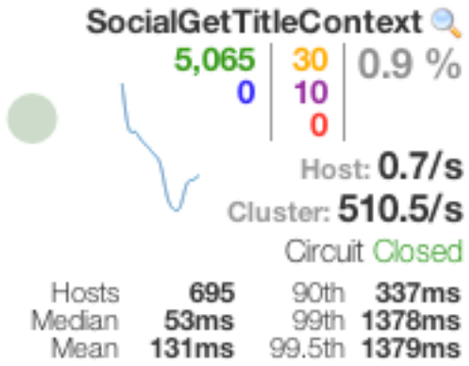
A System Monitor

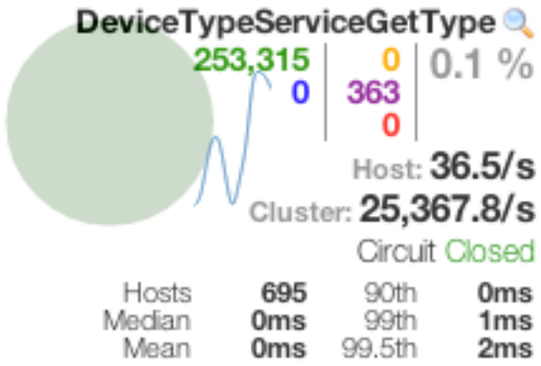
helps to study behaviour and to make sure it is operating as specified.

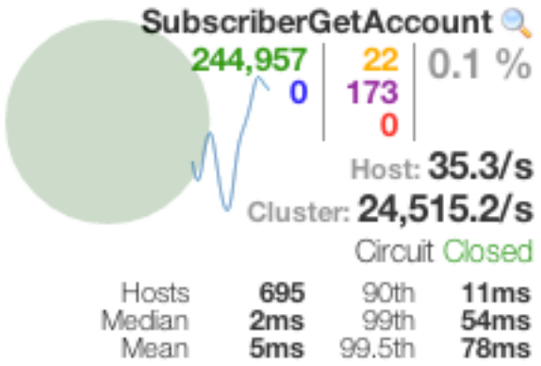
http://upload.wikimedia.org/wikipedia/commons/3/3b/Mission_control_center.jpg

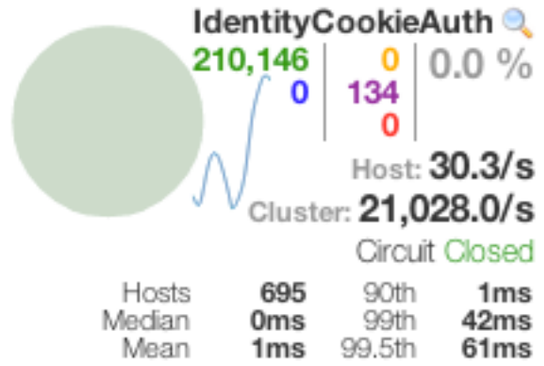
Circuit Breakers

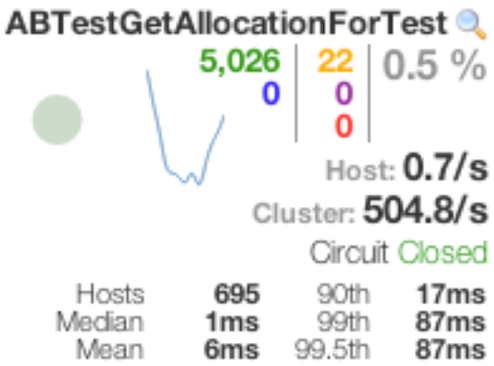
Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

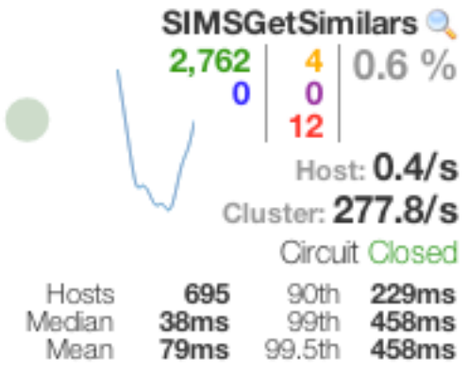


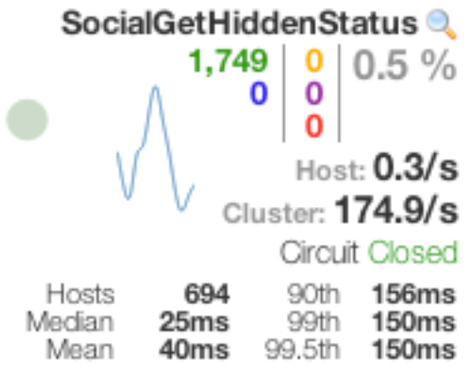


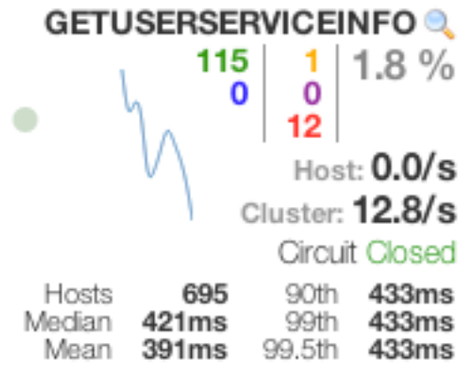


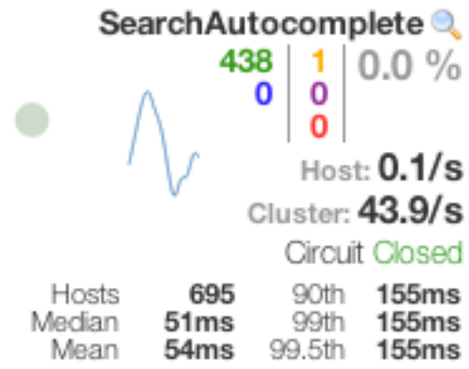


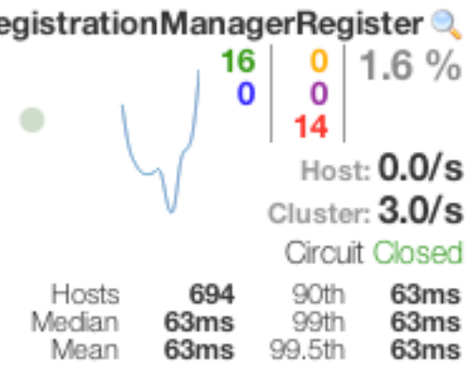


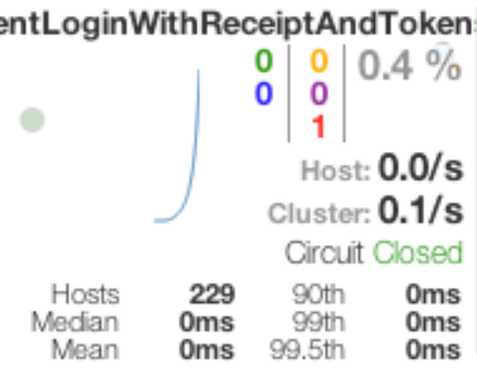


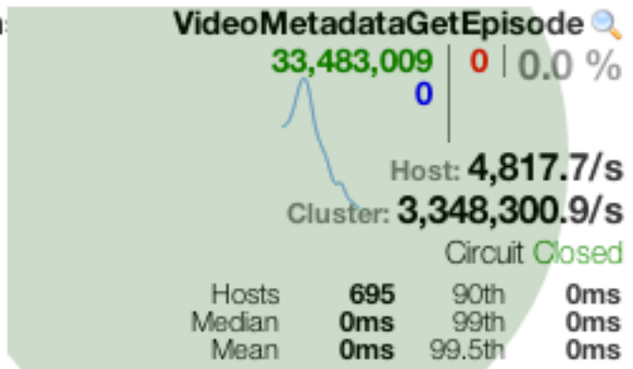


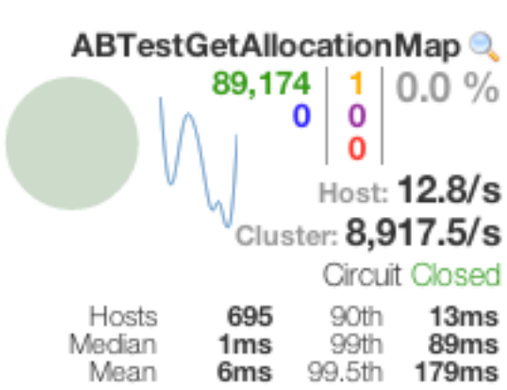


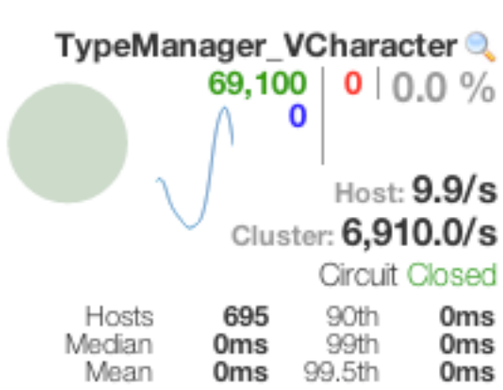


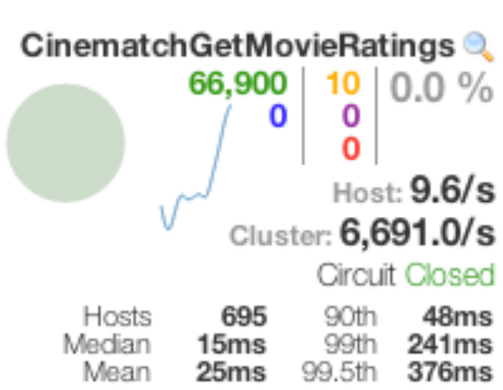


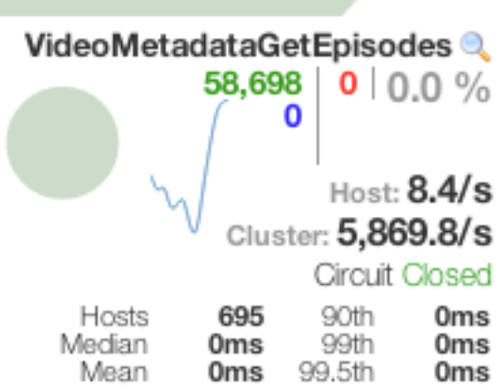










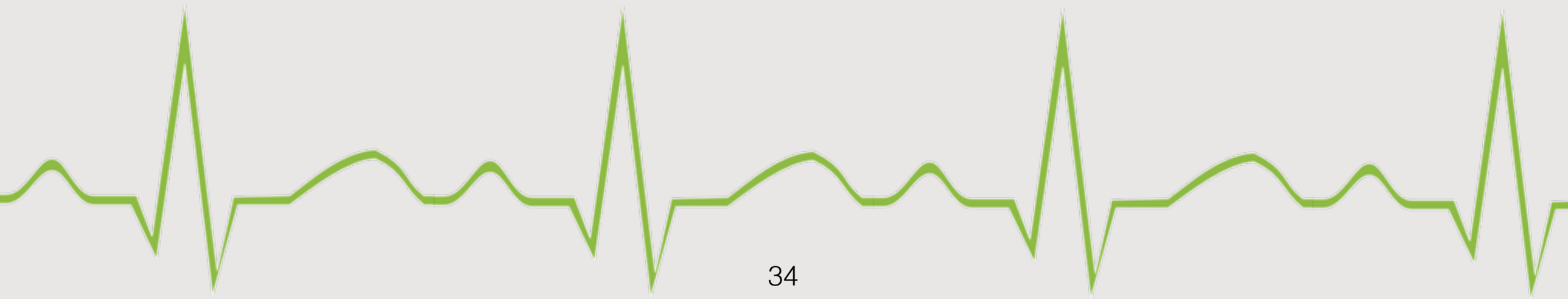


https://github.com/Netflix/Turbine

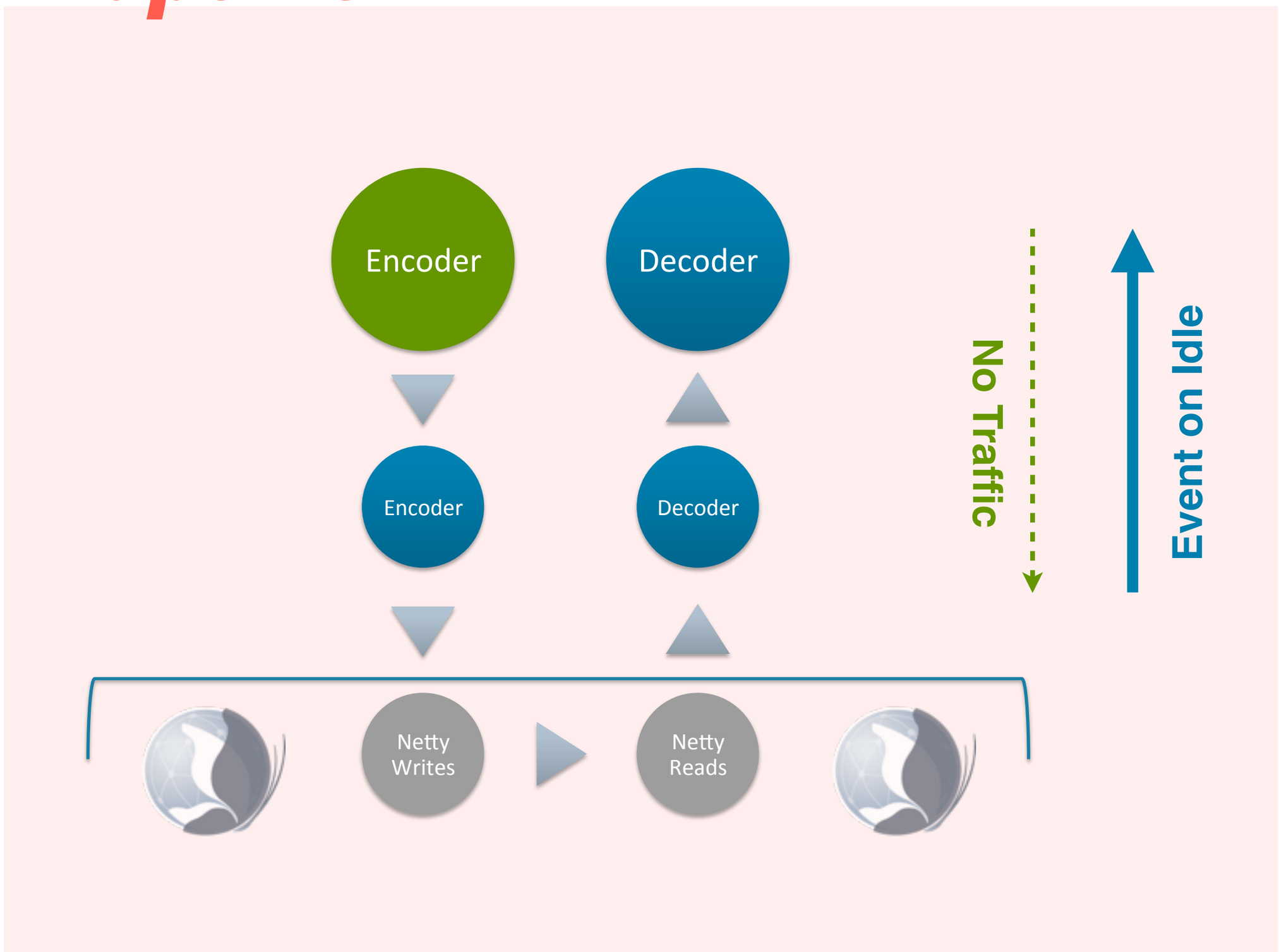
Periodic Checking

Heartbeats *monitor tasks or remote services and initiate recovery*

Routine Exercises *prevent idle unit starvation and surface malfunctions*



Endpoint


Encoder
Decoder
Encoder
Decoder
No Traffic
Event on Idle
Netty
Writes
Netty
Reads

Riding over Transients

is used to defer error recovery
if the error is temporary.

“‘Patience is a virtue’ to allow the true signature of an error to show itself.”

- Robert S. Hanmer

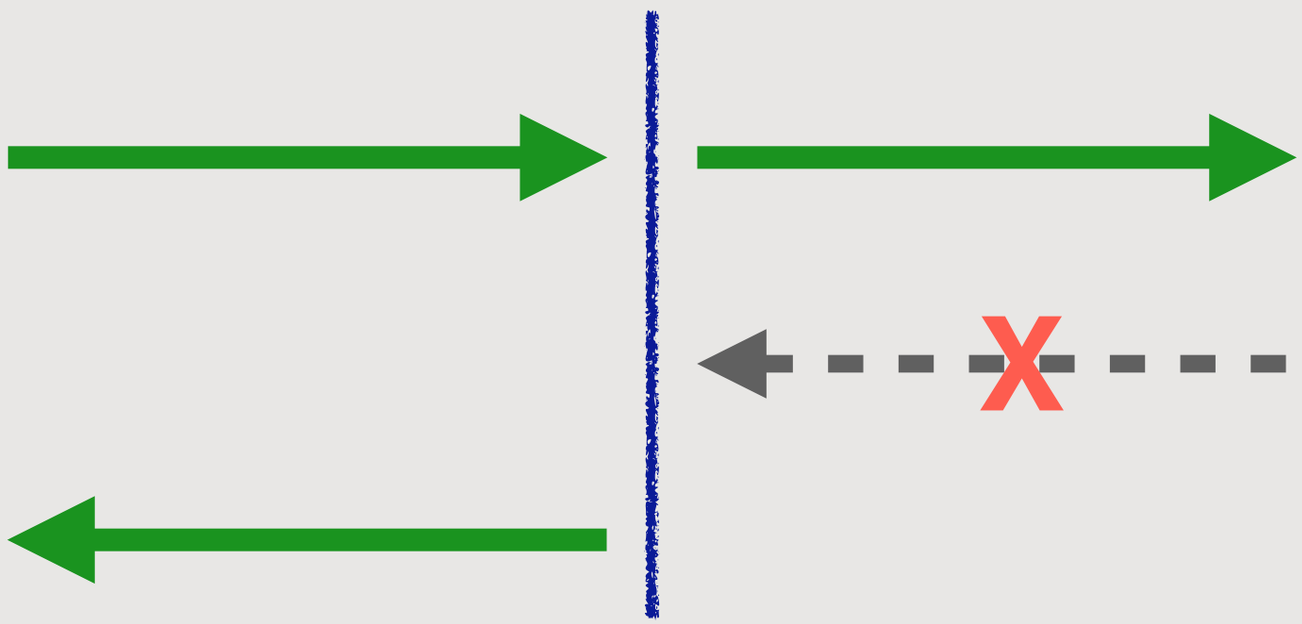
And more!

- Complete Parameter Checking
- Watchdogs
- Voting
- Checksums
- Routine Audits

Recovery and Mitigation of Errors

Timeout

to not wait forever and keep holding up the resource.

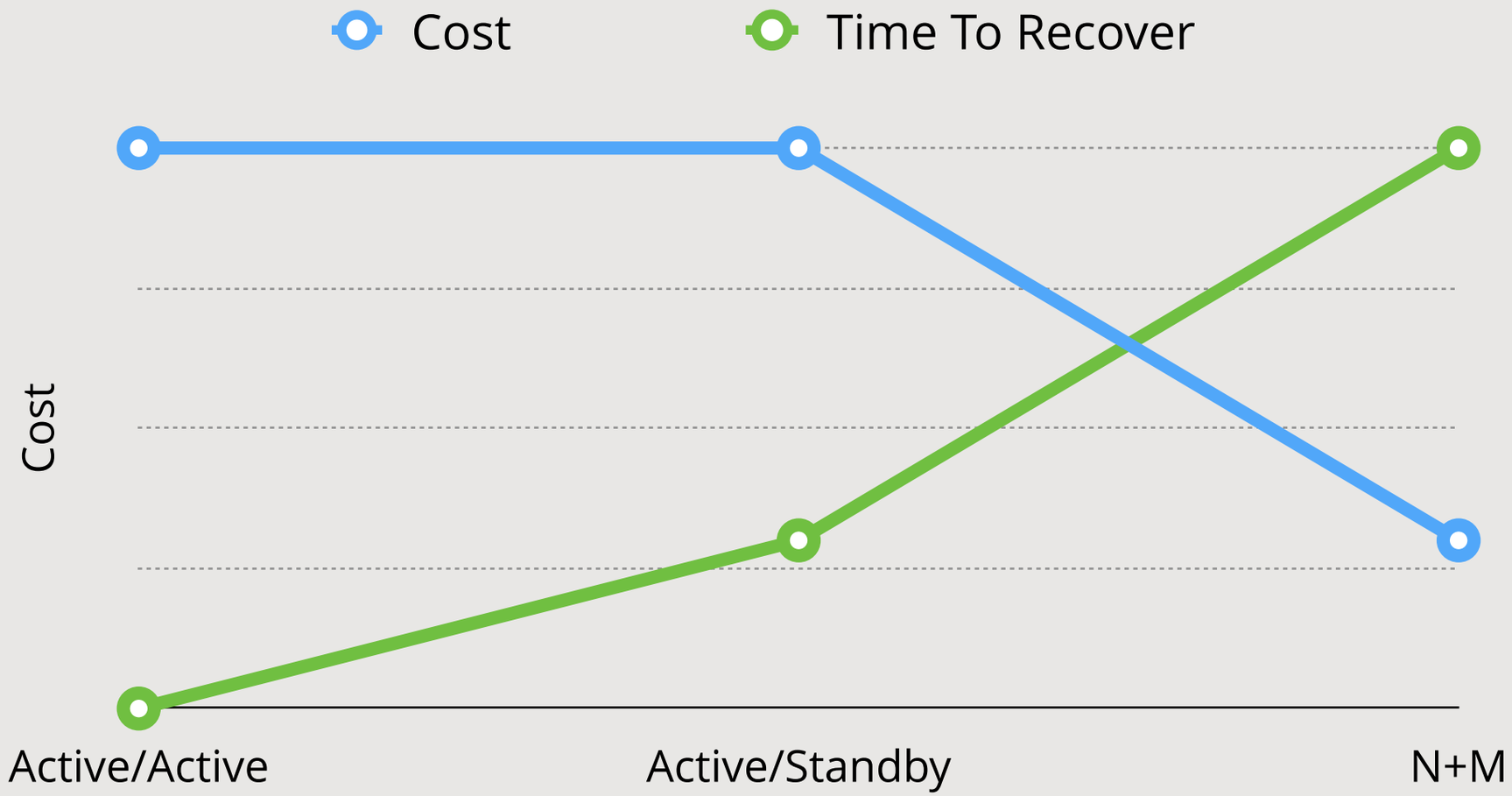


Failover

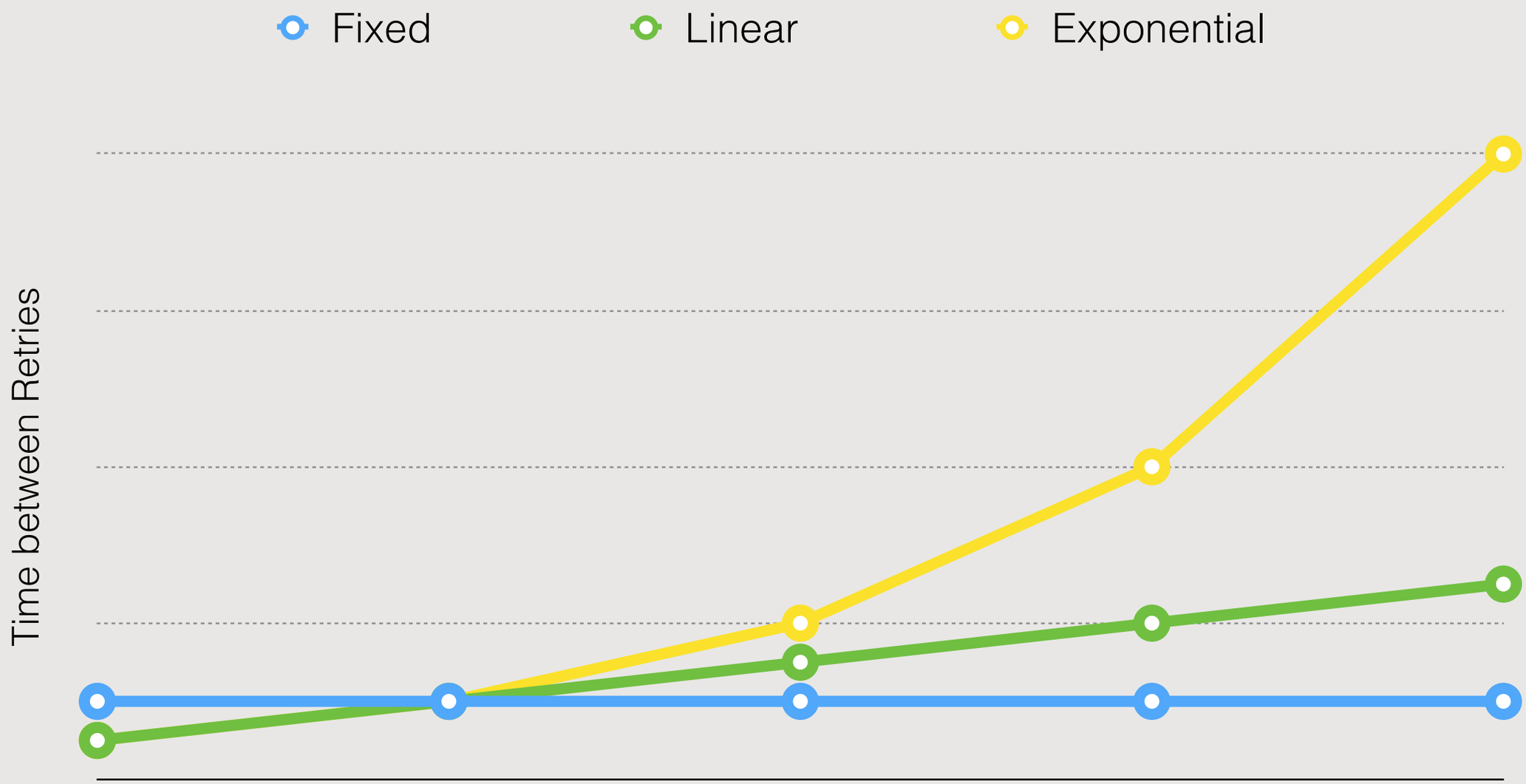
to a redundant unit when the error has been detected and isolated.

Redundancy

Reminder



Intelligent Retries

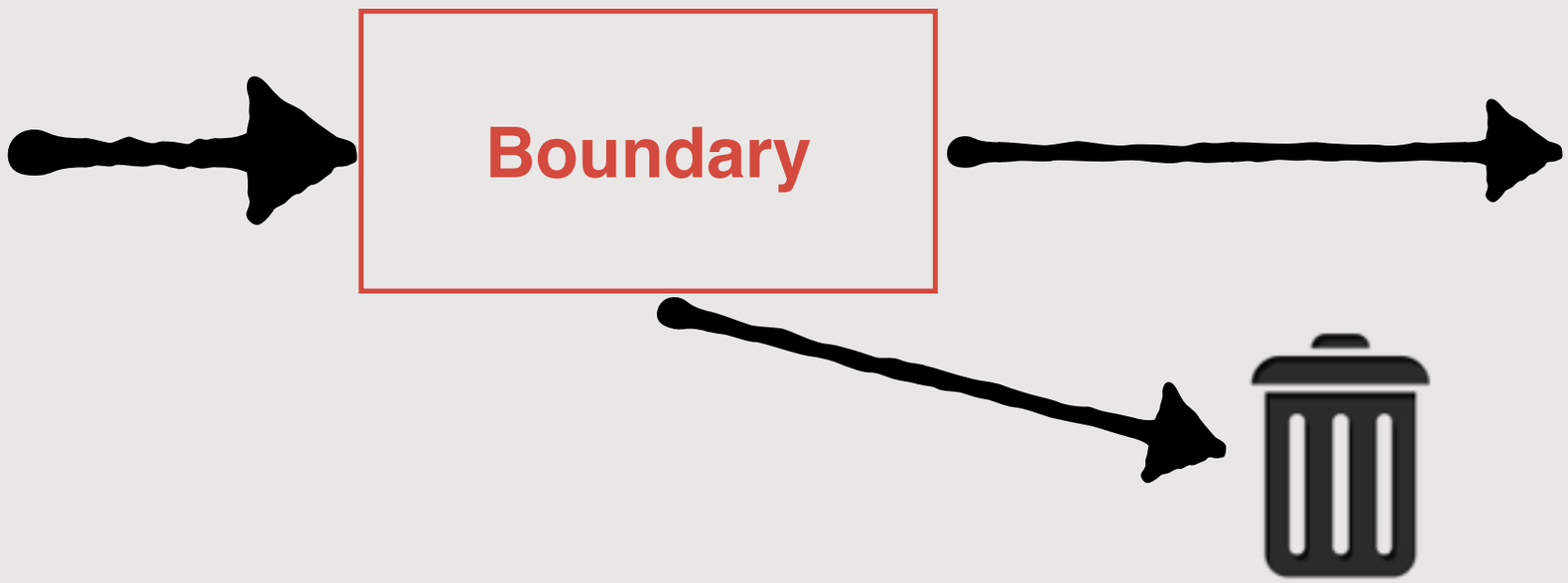


Restart

can be used as a last resort with the trade-off to lose state and time.

Fail Fast

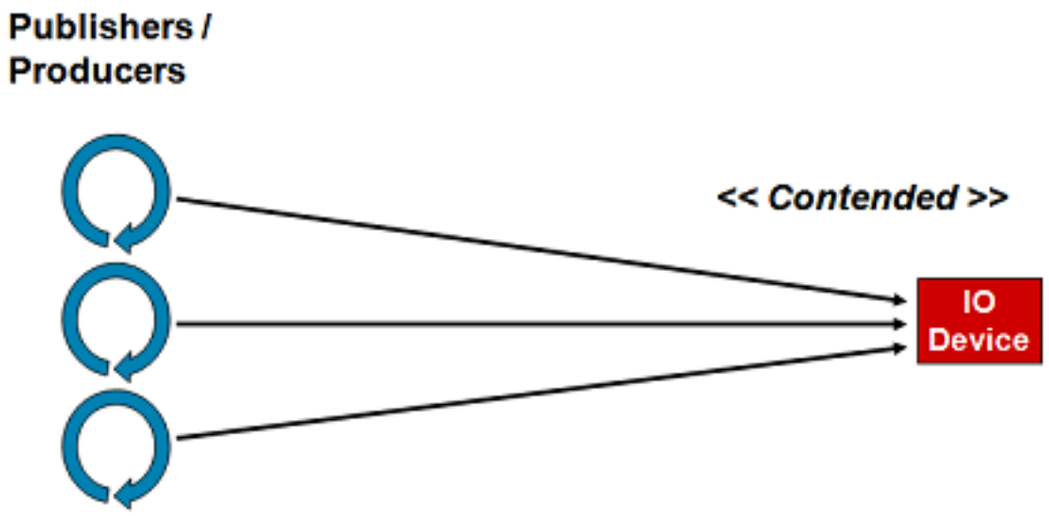
to shed load and give a partial great service than a complete bad one.

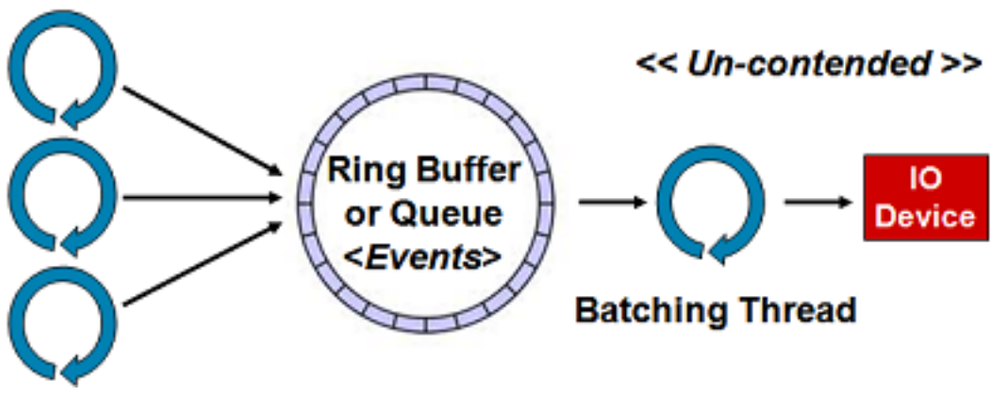


Backpressure

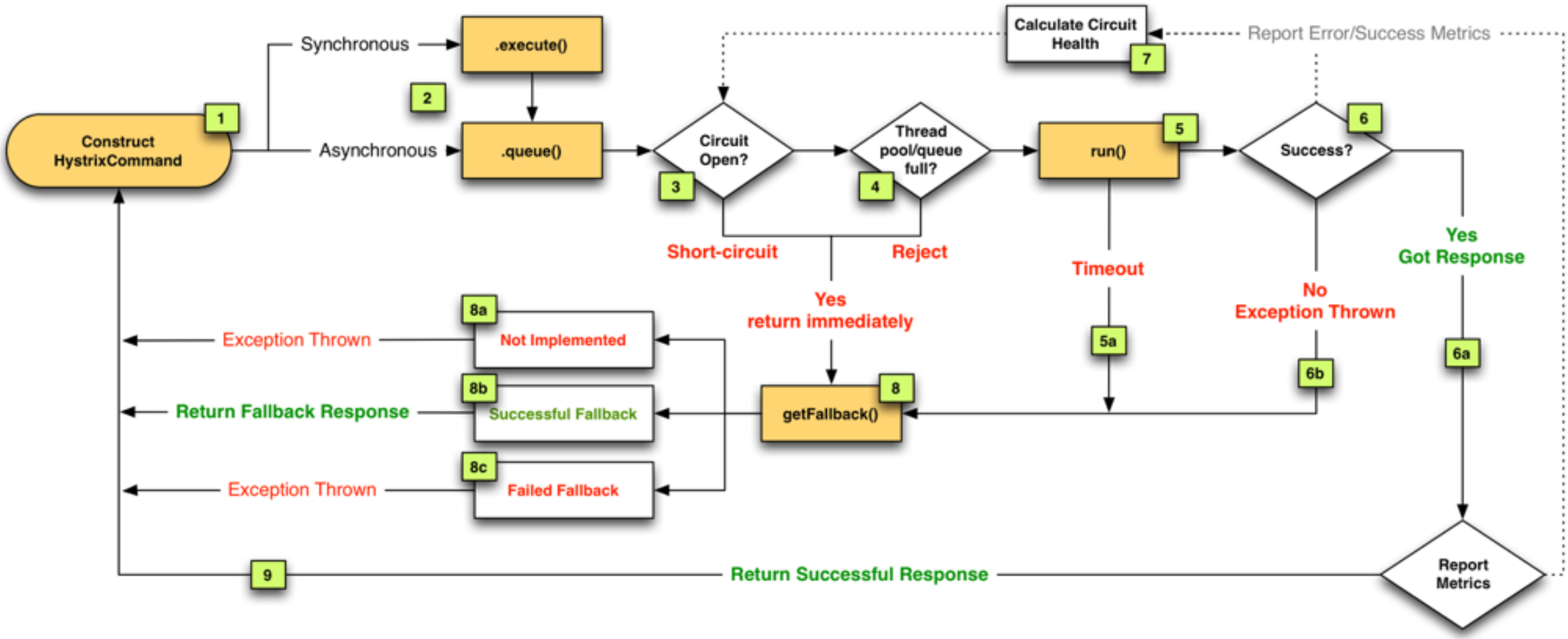
```
1  public <R extends CouchbaseResponse> Observable<R> send(CouchbaseRequest request) {
2        boolean published = requestRingBuffer.tryPublishEvent(REQUEST_TRANSLATOR, request);
3        if (!published) {
4            request.observable().onError(BACKPRESSURE_EXCEPTION);
5        }
6        return (Observable<R>) request.observable();
7  }
```

& Batching!





Case Study: Hystrix



https://raw.githubusercontent.com/wiki/Netflix/Hystrix/images/hystrix-flow-chart-original.png

And more!

Recovery

- Rollback
- Roll-Forward
- Checkpoints
- Data Reset

Mitigation

- Bounded Queuing
- Expansive Controls
- Marking Data
- Error Correcting Codes

And more!

Recovery

- Rollback
- Roll-Forward
- Checkpoints
- Data Reset

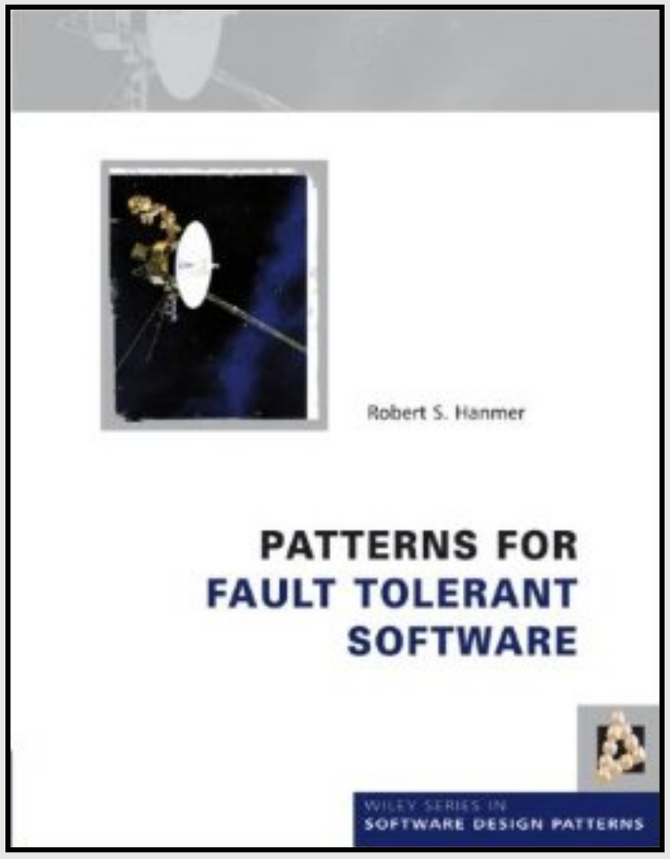
Mitigation

- Bounded Queuing
- Expansive Controls
- Marking Data
- Error Correcting Codes

Recommended Reading

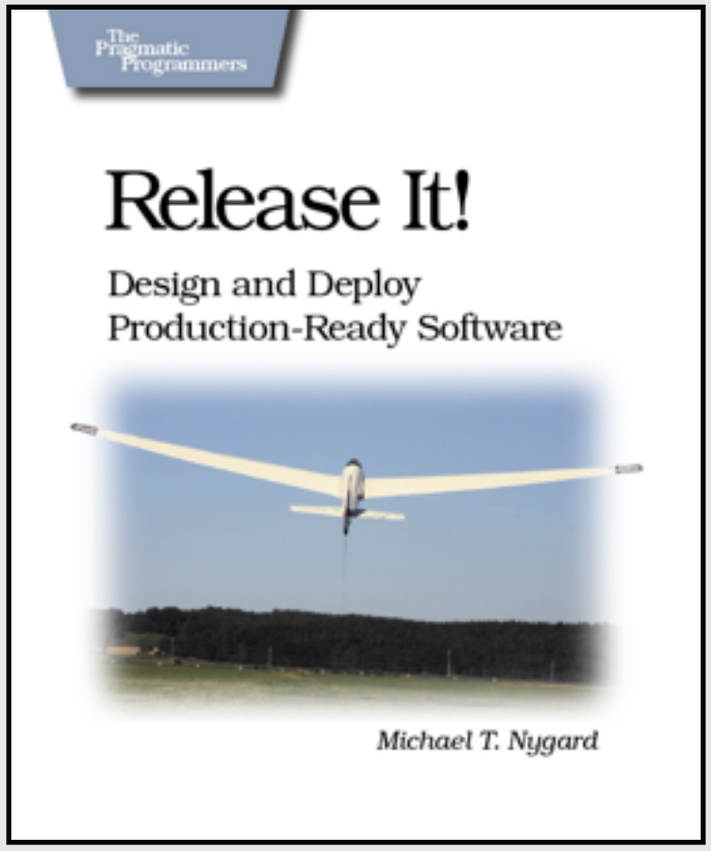
Patterns for Fault-Tolerant Software

by Robert S. Hanmer



Release It!

by Michael T. Nygard



Any

Questions?

Thank you!

twitter

@daschl

email

michael.nitschinger@couchbase.com