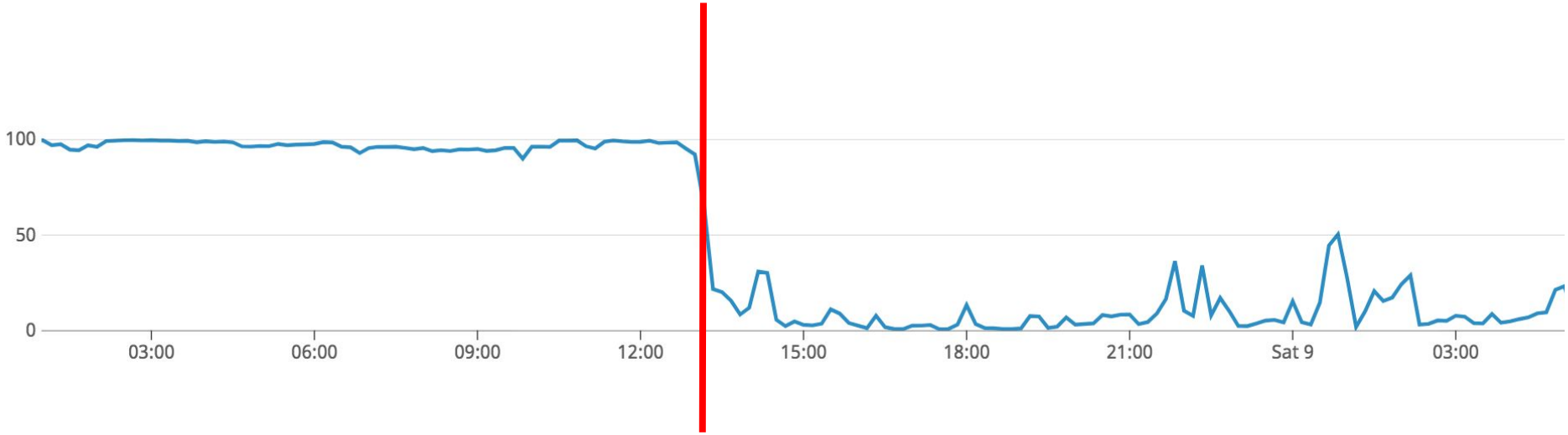
KENNA Security

Cache is King

KENNA
Security

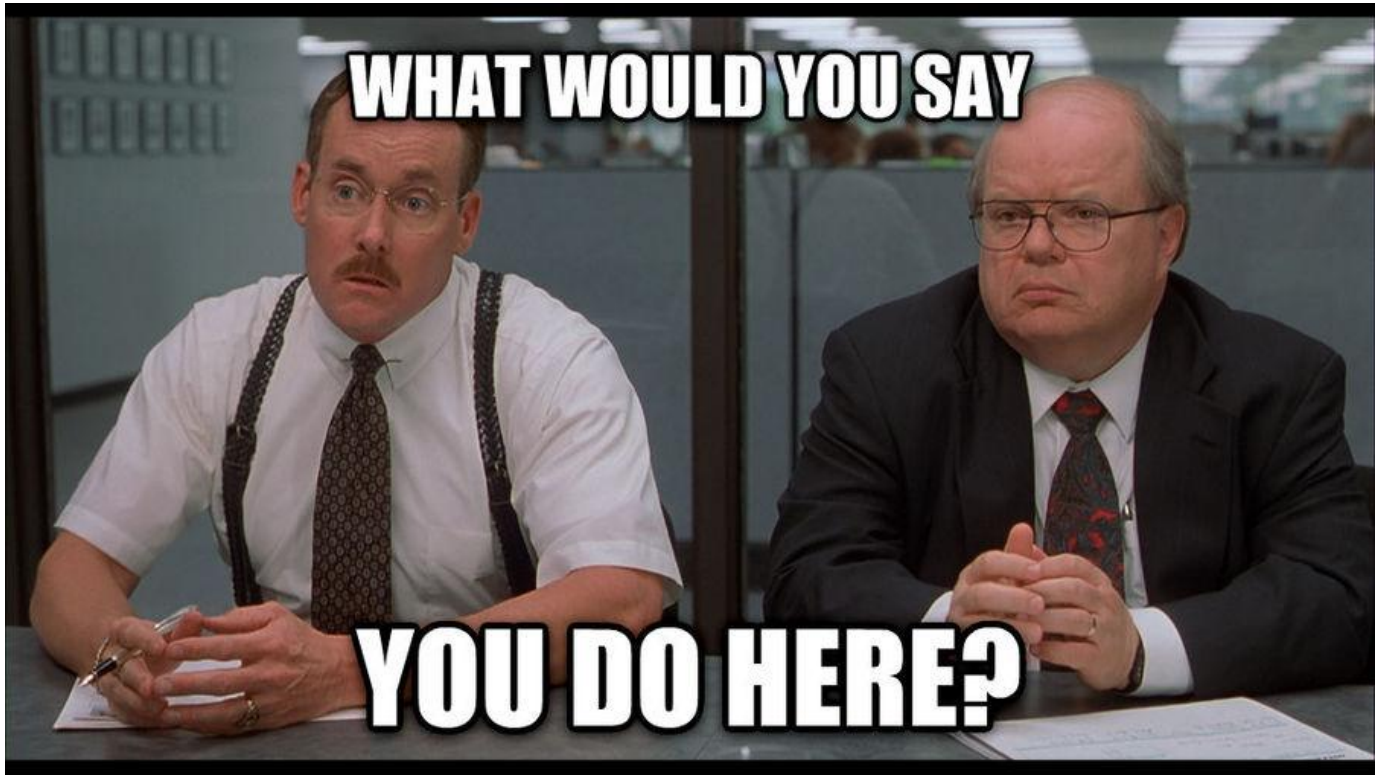


@molly_struve

KENNA
Security

@molly_struve

KENNA Security



KENNA Security

Cache is King

KENNA Security

Site Reliability Engineer

KENNA Security

elastic

PostgreSQL

redis

MySQL®

aws

@molly_struve

KENNA Security

Adding Indexes

Using SELECT statements

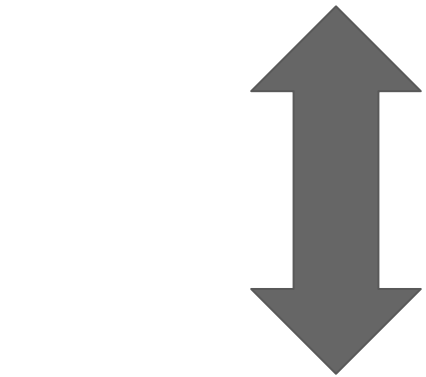
Batch Processing

KENNA Security

```
{
  "statusCode": 200,
  "took": "100ms"
}
```

KENNA Security

Resque



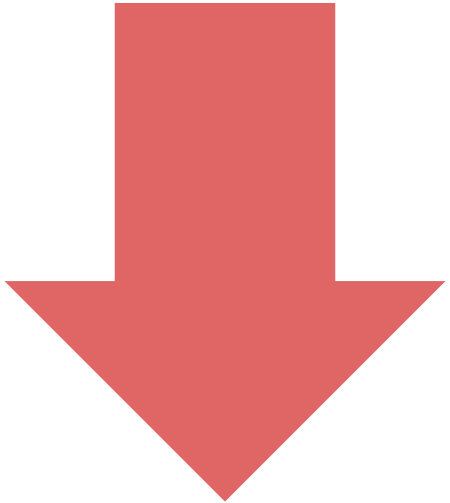

MySQL
elastic
redis

KENNA Security

Demo Time!

KENNA Security

Quantity of Datastore Hits



@molly_struve

The average company has...

KENNA Security

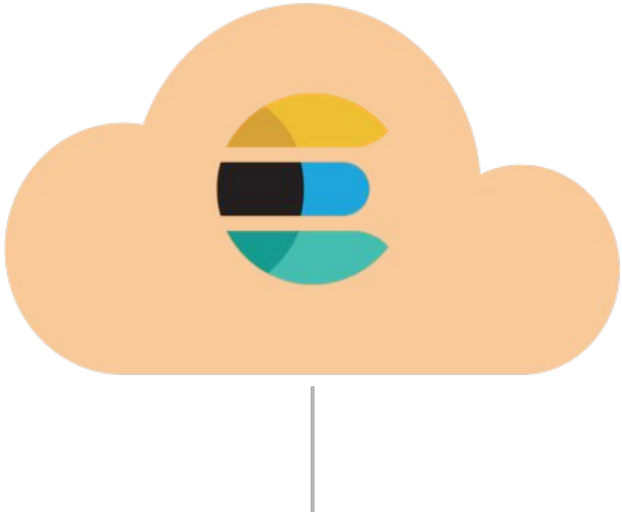
60 thousand assets

?

24 million vulnerabilities

KENNA Security

MySQL



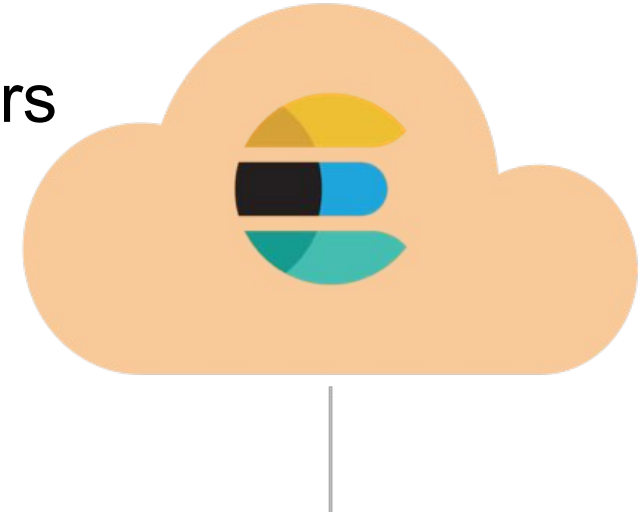
Elasticsearch Cluster

KENNA Security

Serialization

KENNA Security

ActiveModelSerializers



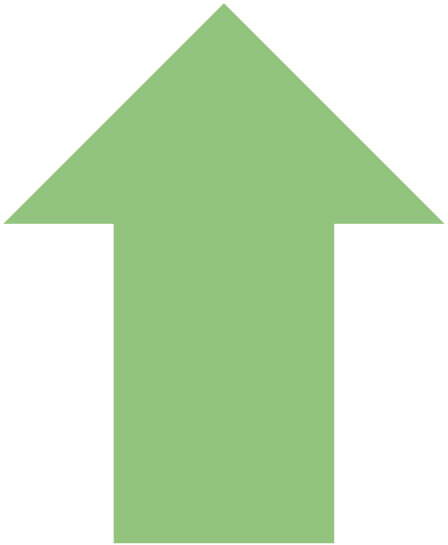
MySQL

Elasticsearch Cluster

KENNA Security

```
class Vulnerability < ActiveModel::Serializer
    attributes :id, :client_id, :created_at, :updated_at,
               :priority, :details, :notes, :asset_id,
               :solution_id, :owner_id, :ticket_id
end
```

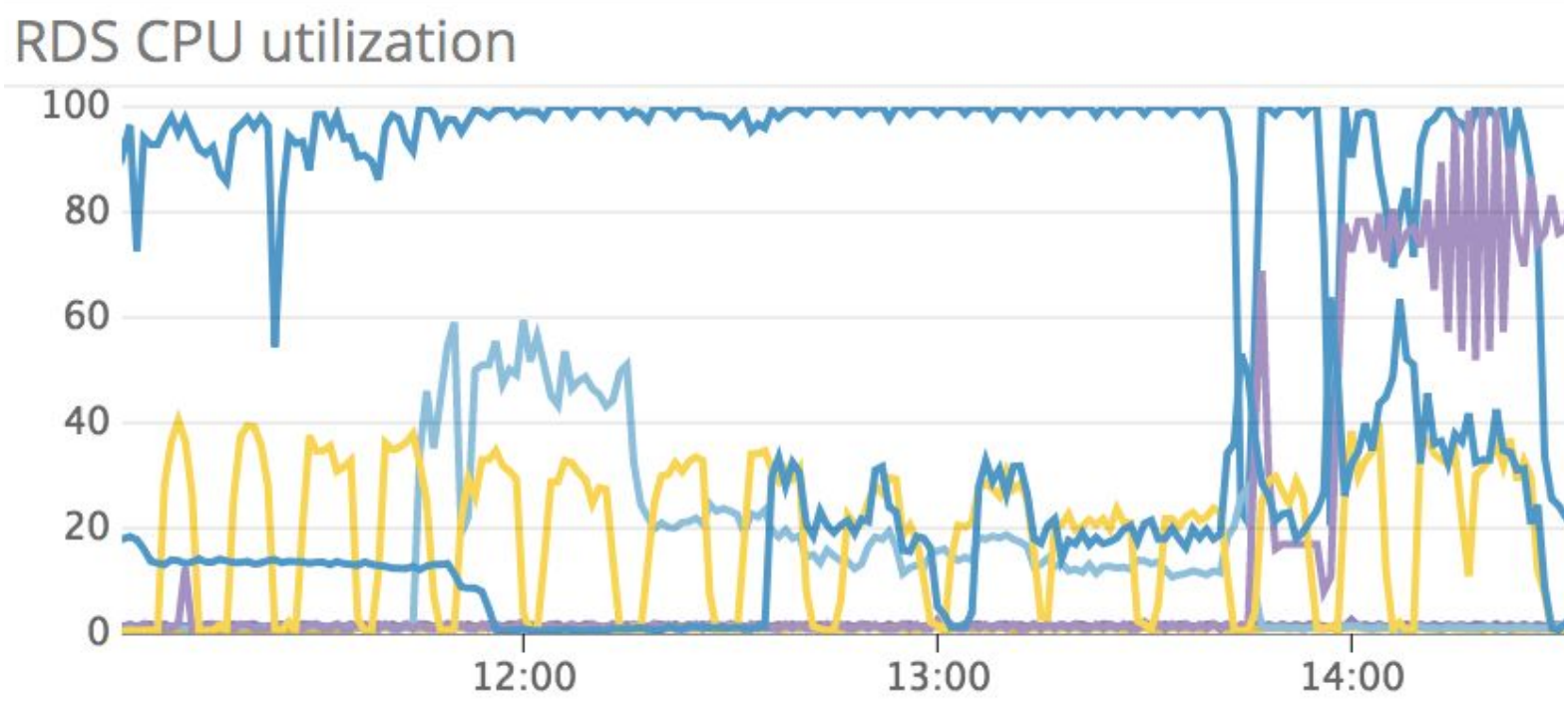
KENNA
Security



200 MILLION

11 hours and counting...

KENNA Security

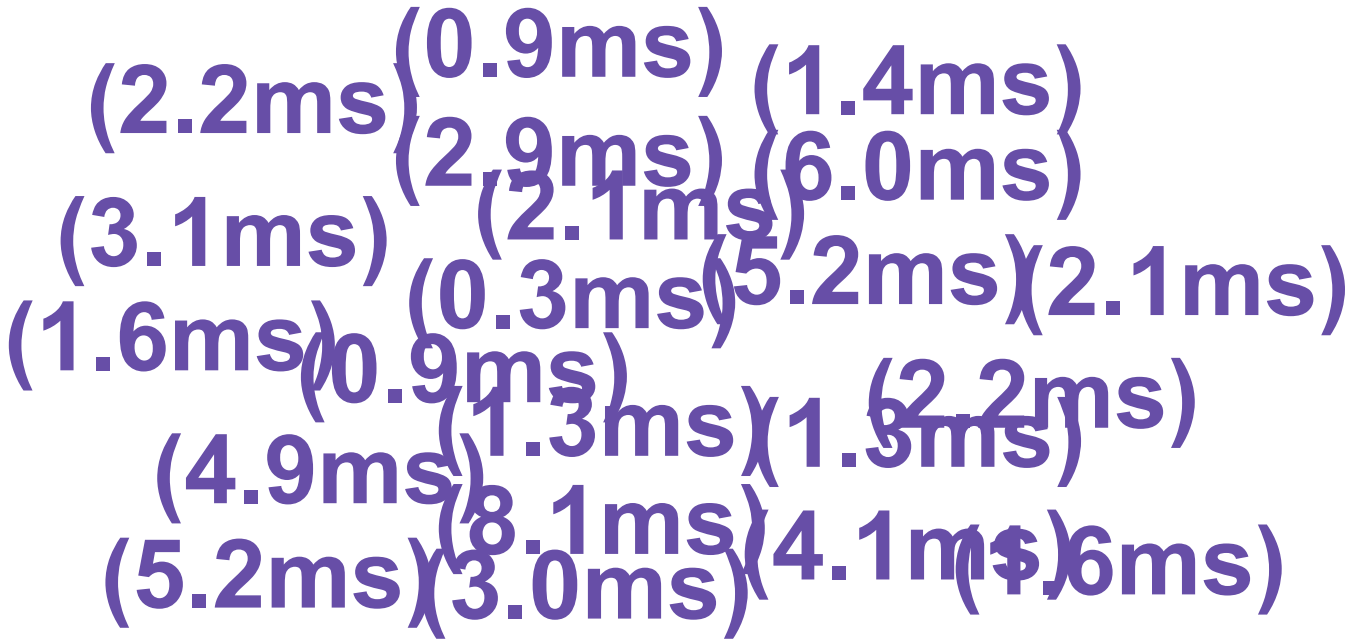


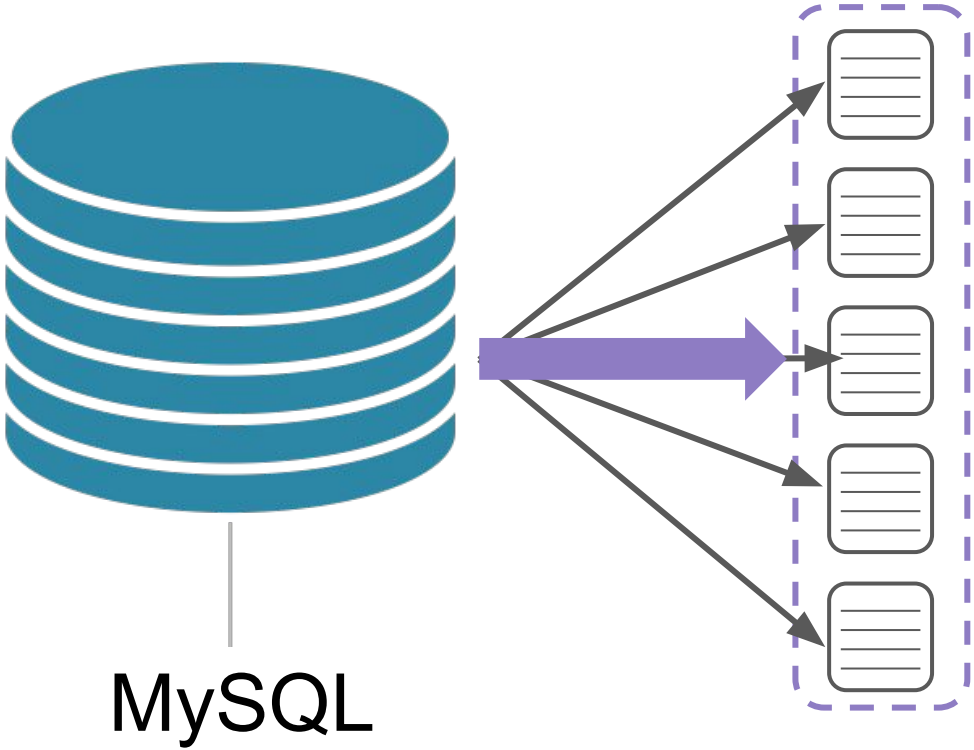
KENNA Security

KENNA Security

@molly_struve

KENNA Security




MySQL

KENNA Security

Bulk Serialization

KENNA Security

```
class BulkVulnerabilityCache
    attr_accessor :vulnerabilities, :client, :vulnerability_ids

    def initialize(vulns, client)
        self.vulnerabilities = vulns
        self.vulnerability_ids = vulns.map(&:id)
        self.client = client
    end

    # MySQL Lookups
end
```

KENNA Security

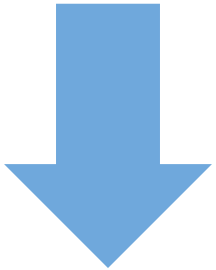
```
module Serializers
  class Vulnerability
    attr_accessor :vulnerability, :cache

    def initialize(vuln, bulk_cache)
      self.cache = bulk_cache
      self.vulnerability = vuln
    end
  end
end
```

KENNA Security

```
class Vulnerability
    has_many :custom_fields
end
```

KENNA Security

```
CustomField.where(:vulnerability_id => vuln.id)
```

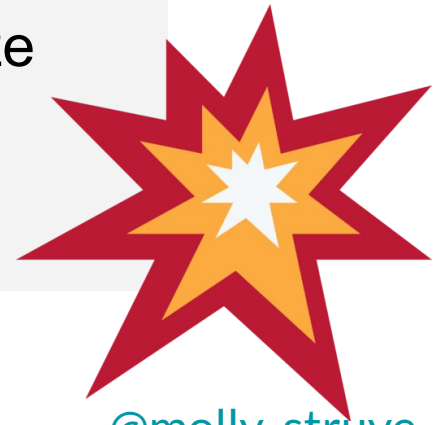


```
cache.fetch('custom_fields', vuln.id)
```

The Result...

KENNA Security

```
(pry)> vulns = Vulnerability.limit(300);
(pry)> Benchmark.realtime { vulns.each(&:serialize) }
=> 6.022452222998254

(pry)> Benchmark.realtime do
>    BulkVulnerability.new(vulns, [], client).serialize
> end
=> 0.7267019419959979
```



@molly_struve

Decrease in database hits

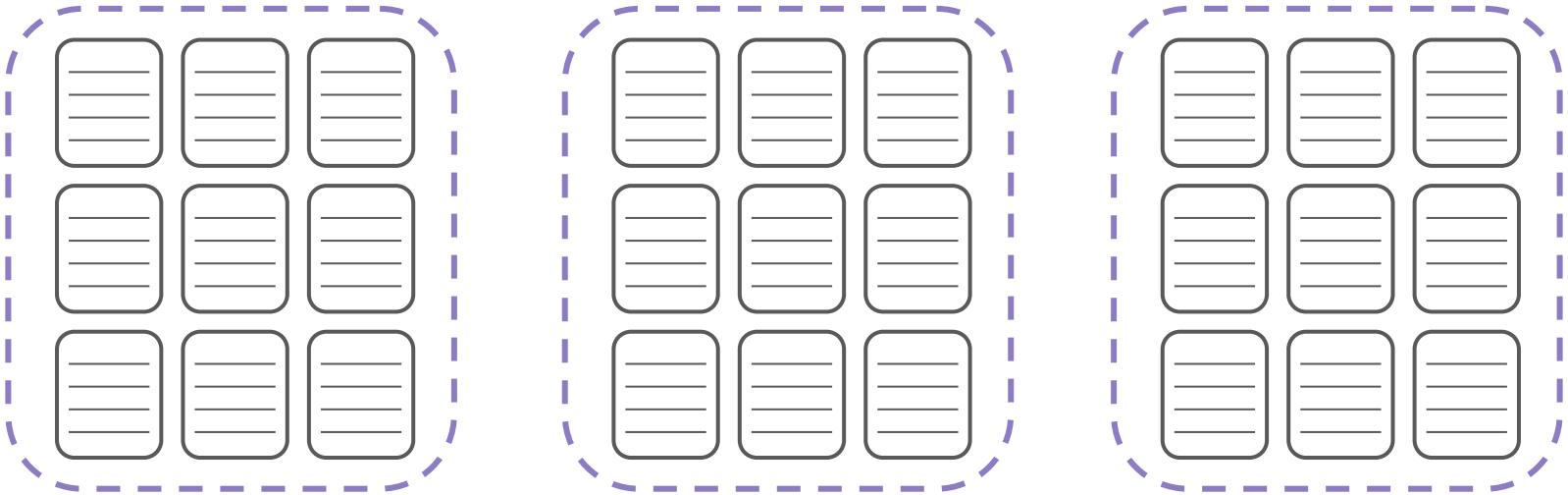
KENNA Security

Individual Serialization: **2,100**

Bulk Serialization: **7**

Vulnerability Batches

KENNA Security



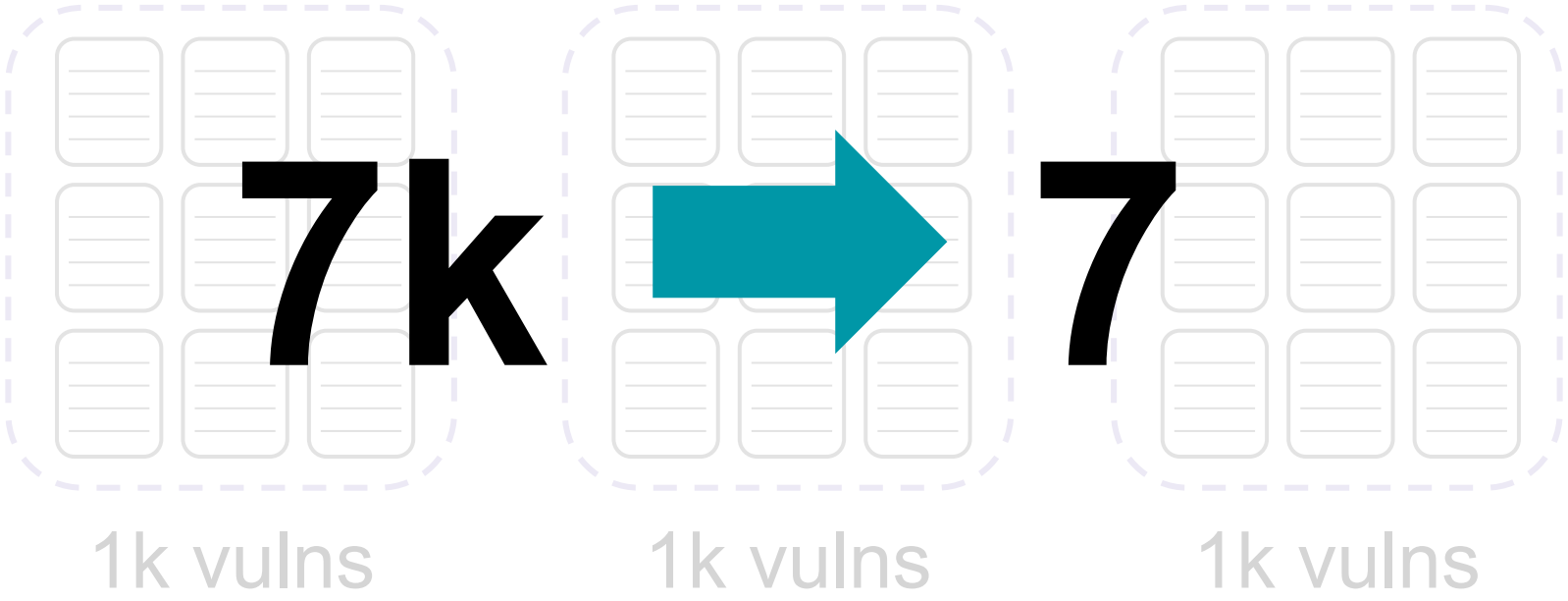
1k vulns

1k vulns

1k vulns

Vulnerability Batches

KENNA Security



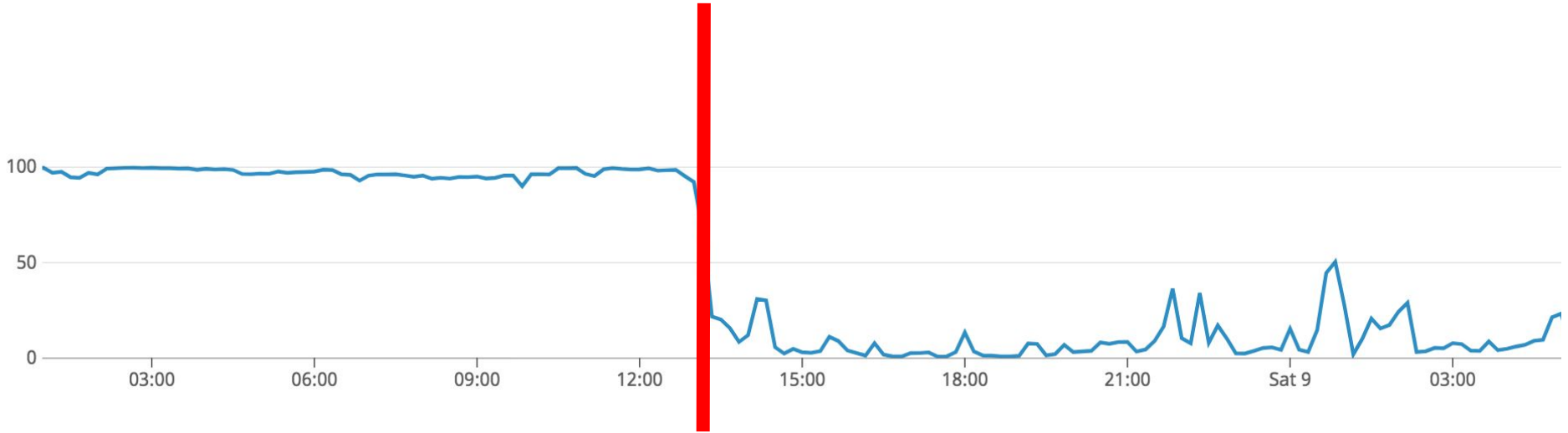
KENNA Security

MySQL Queries

Bulk Serialization Deployed

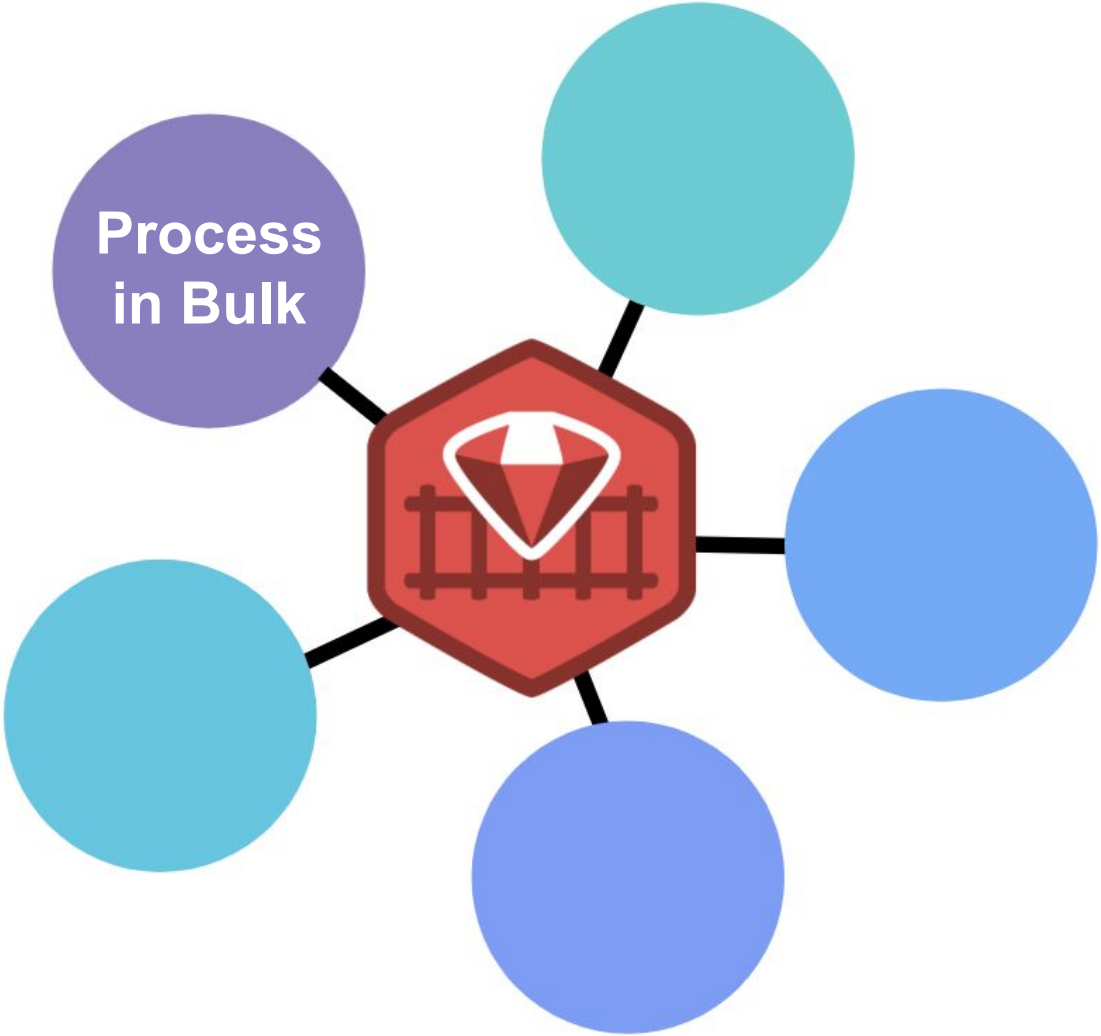
KENNA Security

RDS CPU Utilization



Bulk Serialization Deployed

KENNA Security



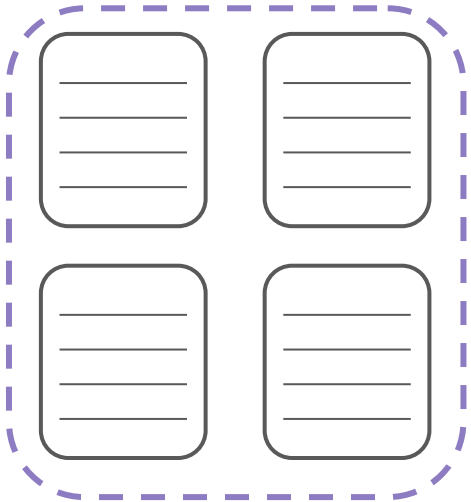

MySQL®


redis

KENNA
Security

@molly_struve

KENNA Security

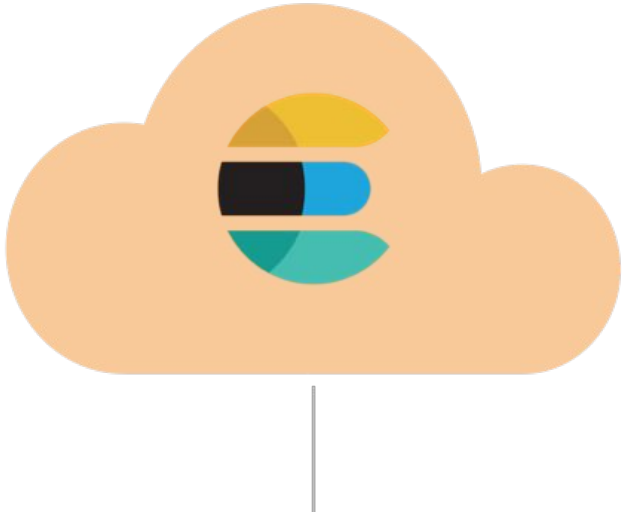


Vulnerabilities

MySQL

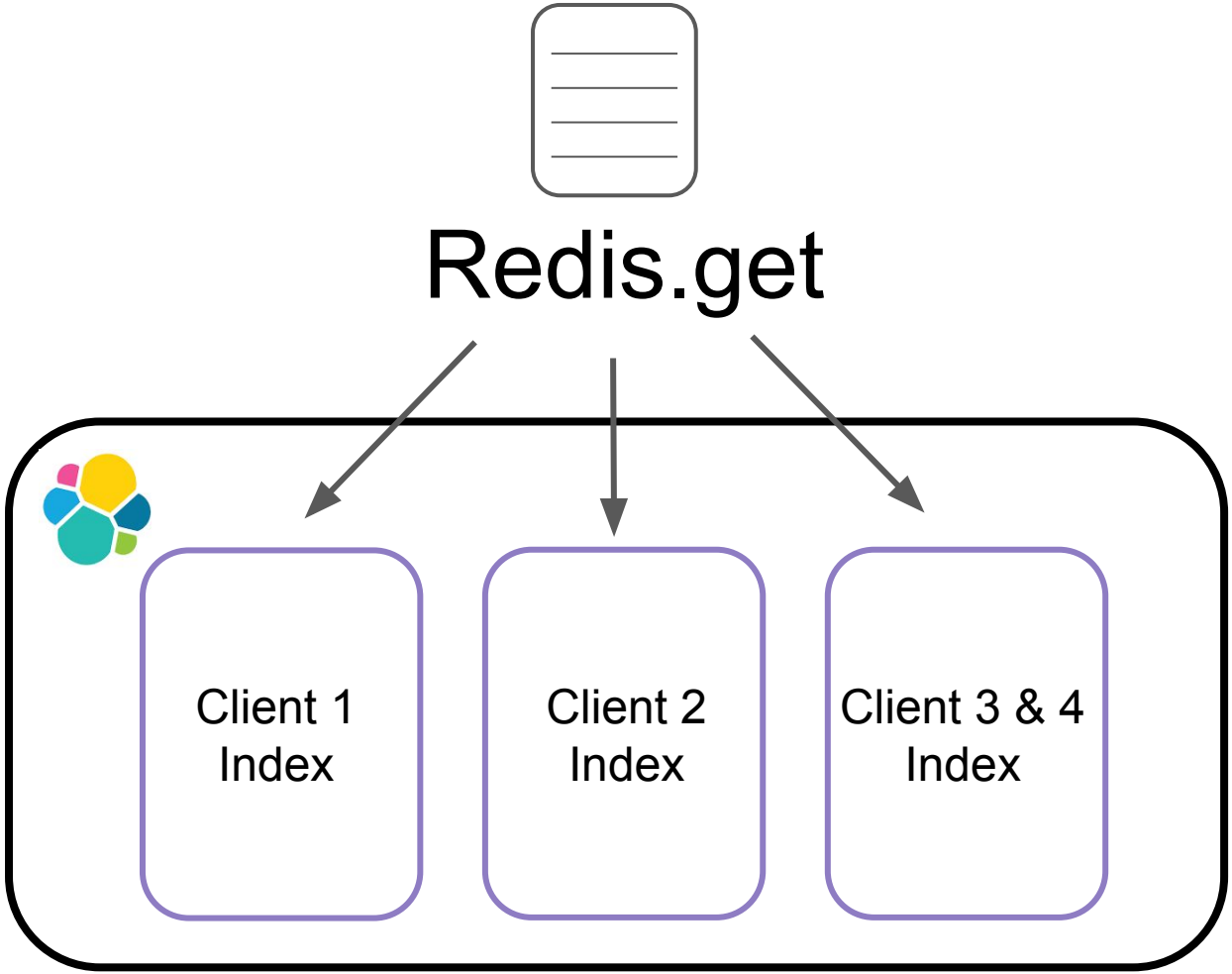
+

Redis



Elasticsearch Cluster

KENNA Security



KENNA Security

```
indexing_hashes = vulnerability_hashes.map do |hash|
  {
    :_index => Redis.get("elasticsearch_index_#{hash[:client_id]}")
    :_type => hash[:doc_type],
    :_id => hash[:id],
    :data => hash[:data]
  }
end
```

KENNA Security

```
indexing_hashes = vulnerability_hashes.map do |hash|
  {
    :_index => Redis.get("elasticsearch_index_#{hash[:client_id]}")
    :_type => hash[:doc_type],
    :_id => hash[:id],
    :data => hash[:data]
  }
end
```

KENNA Security

GET

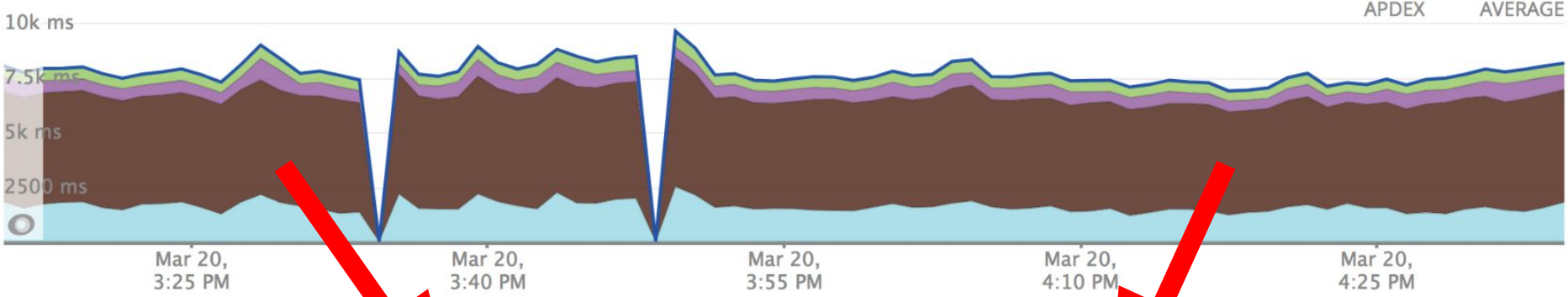
```
(pry)> index_name = Redis.get(“elasticsearch_index_#{client_id}”)
DEBUG -- : [Redis] command=GET args="elasticsearch_index_1234"
DEBUG -- : [Redis] call_time=1.07 ms
```

BulkIndexVulnFieldsWorker/perform

App performance | Map Beta

App server breakdown

N/A APDEX | 7.78 sec AVERAGE



Breakdown table

Category	Segment	% Time	Avg calls (per txn)	Avg time (ms)
Database	Redis get	64.9	4000	5,050

@molly_struve

KENNA
Security

KENNA Security

```
client_indexes = Hash.new do |h, client_id|
  h[client_id] = Redis.get("elasticsearch_index_#{client_id}")
end
```

KENNA Security

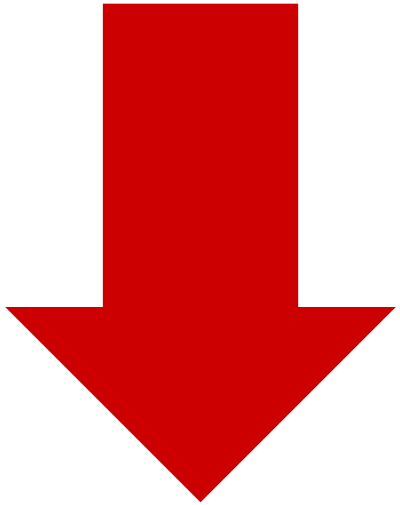
```
indexing_hashes = vuln_hashes.map do |hash|
  {
    :_index => client_indexes[hash[:client_id]],
    :_type => hash[:doc_type],
    :_id => hash[:id],
    :data => hash[:data]
  }
end
```

KENNA Security

Client 1 Client 2 Client 3

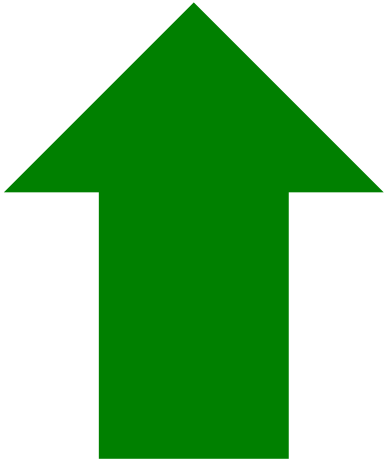
1 + 1 + 1

KENNA Security



1000x

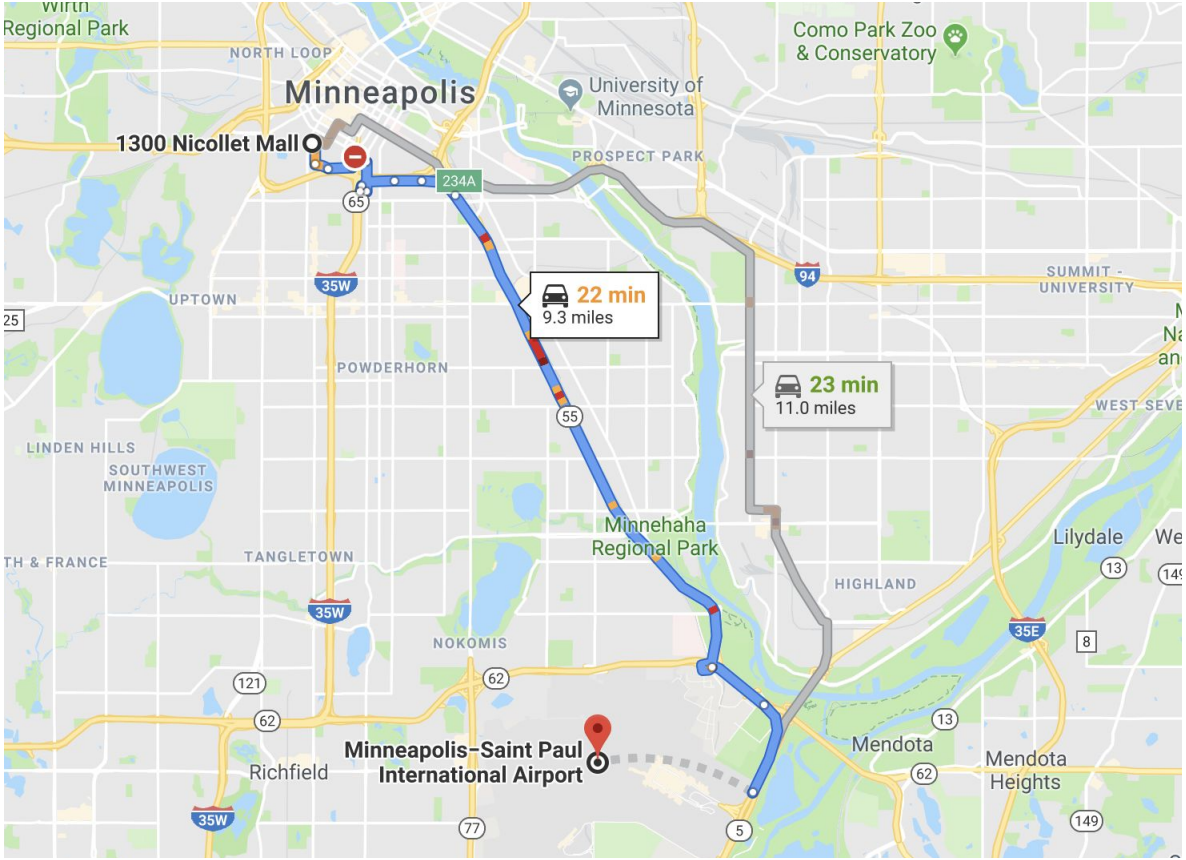
KENNA Security



65% job speed up

Local Cache

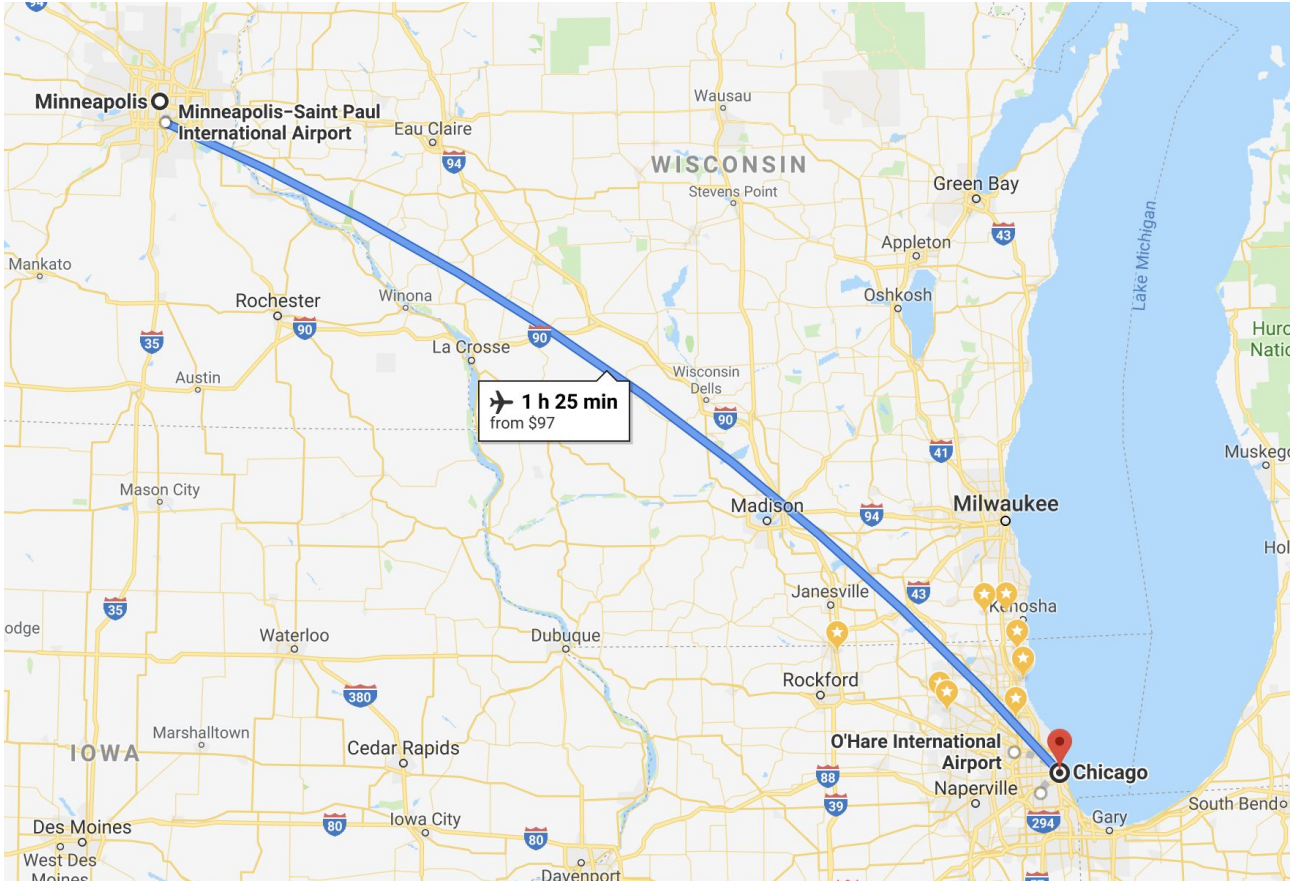
KENNA Security



@molly_struve

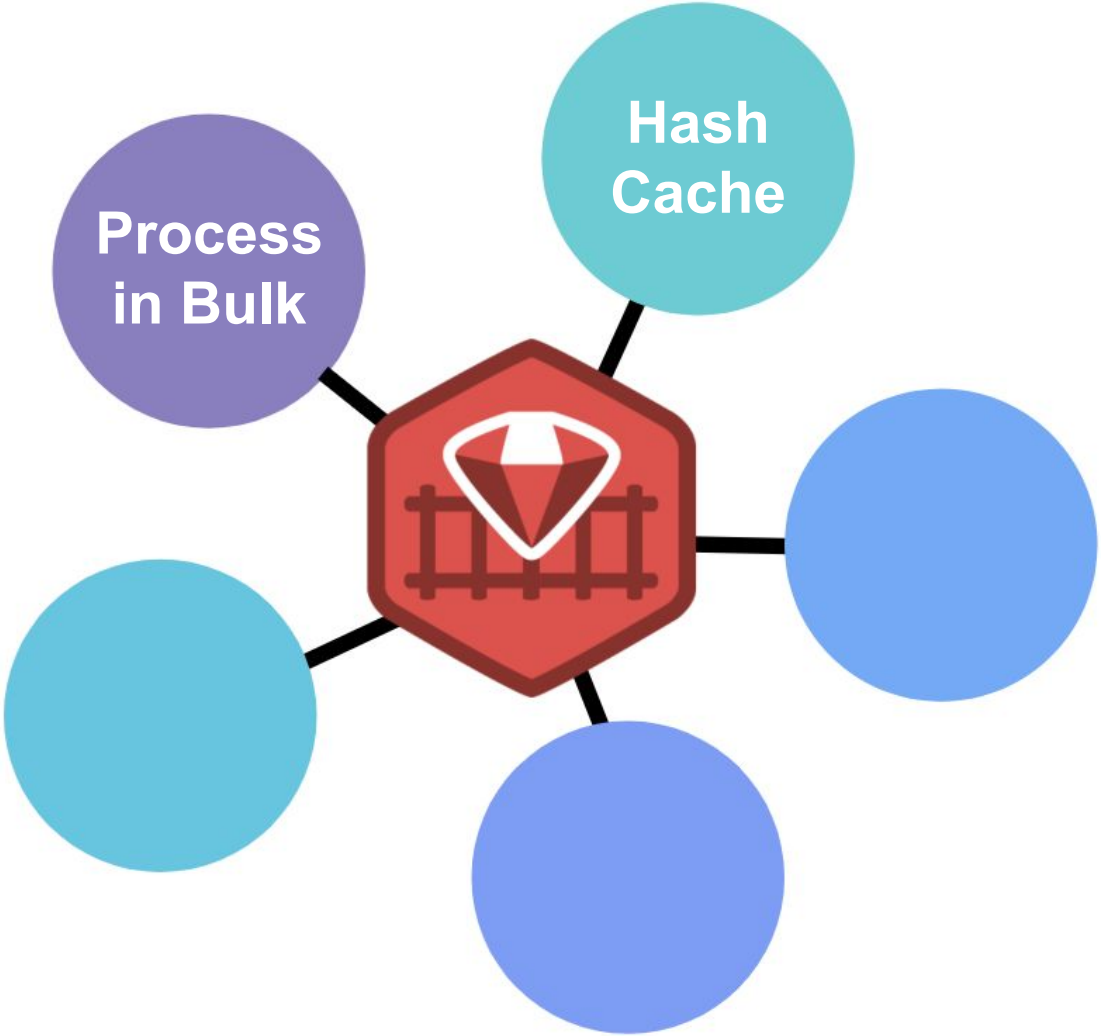
Redis

KENNA Security



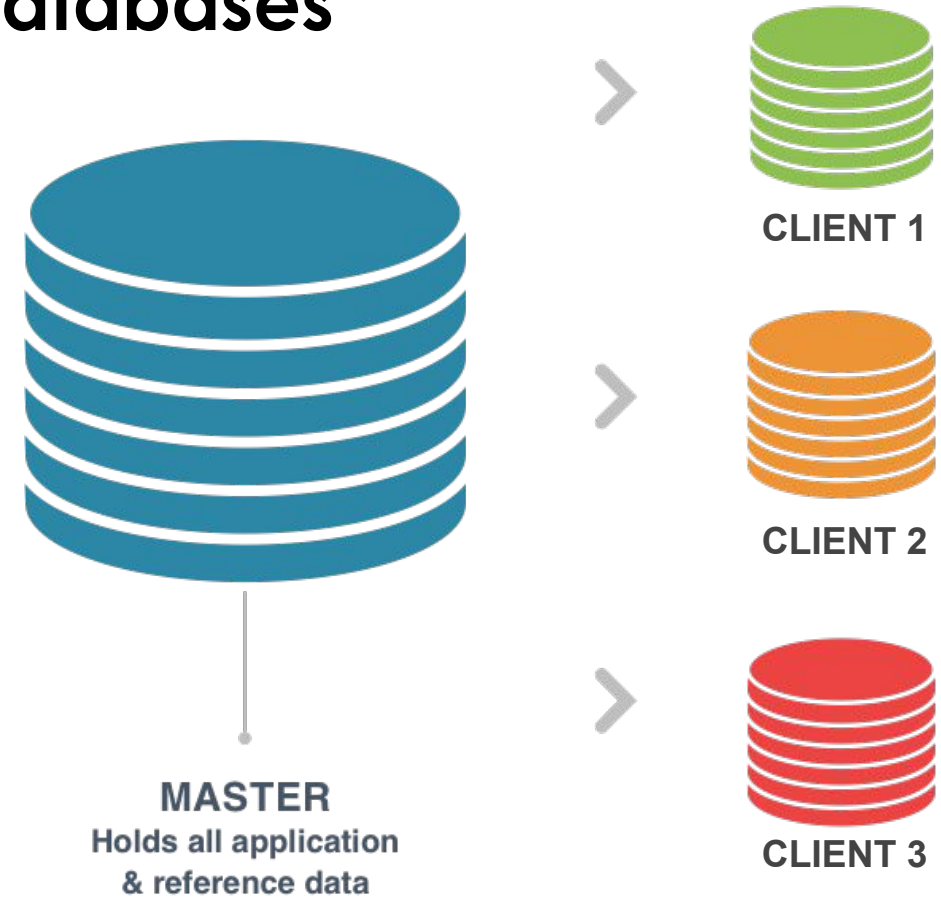
@molly_struve

KENNA Security



Sharded Databases

KENNA Security



KENNA Security

Octopus - Easy Database Sharding for ActiveRecord

build passing CODE CLIMATE

Octopus is a better way to do Database Sharding in ActiveRecord. Sharding allows multiple databases in the same rails application. While there are several projects that implement Sharding (e.g. DbCharmer, DataFabric, MultiDb), each project has its own limitations. The main goal of octopus project is to provide a better way of doing Database Sharding.

```
Asset.using(:shard_1).find(1)
```

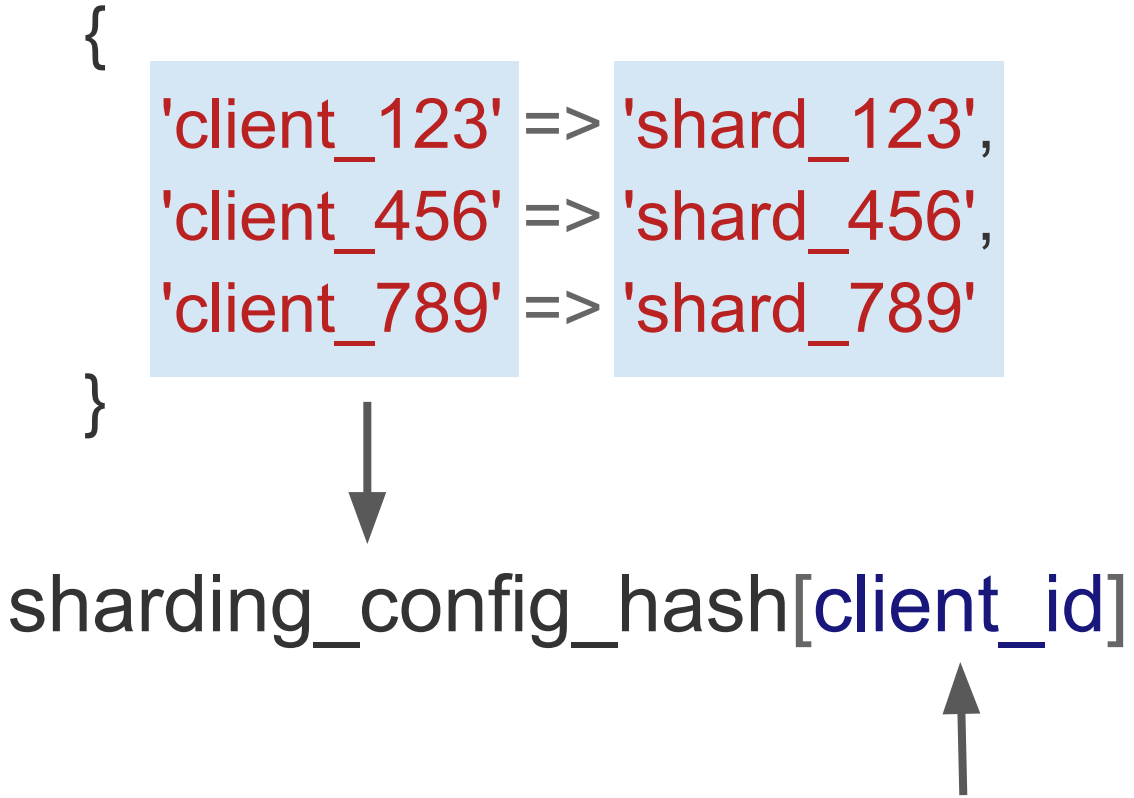
Sharding Configuration Hash

```
{
    'client_123' => 'shard_123',
    'client_456' => 'shard_456',
    'client_789' => 'shard_789'
}
```

KENNA Security

Sharding Configuration Hash

KENNA Security

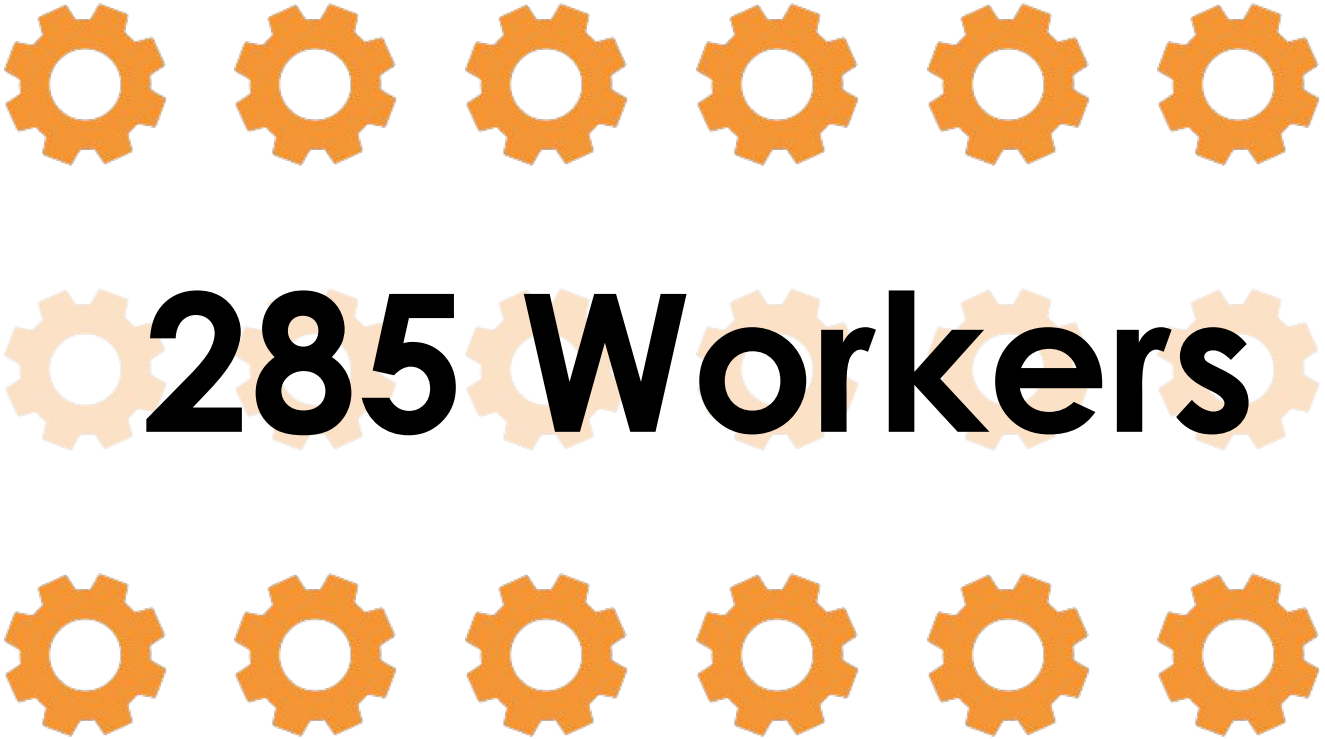



redis

Sharding Configuration Size

KENNA Security

KENNA Security



285 Workers

KENNA Security

7.8 MB/second

KENNA Security

ActiveRecord::Base.connection

```
(pry)> ActiveRecord::Base.connection
  => #<Octopus::Proxy:0x000055b38c697d10
       @proxy_config= #<Octopus::ProxyConfig:0x000055b38c694ae8
```

Octopus - Easy Database Sharding for ActiveRecord

Octopus is a better way to do Database Sharding in ActiveRecord. Sharding allows multiple databases in the same rails application. While there are several projects that implement Sharding (e.g. DbCharmer, DataFabric, MultiDb), each project has its own limitations. The main goal of octopus project is to provide a better way of doing Database Sharding.

KENNA Security

KENNA Security

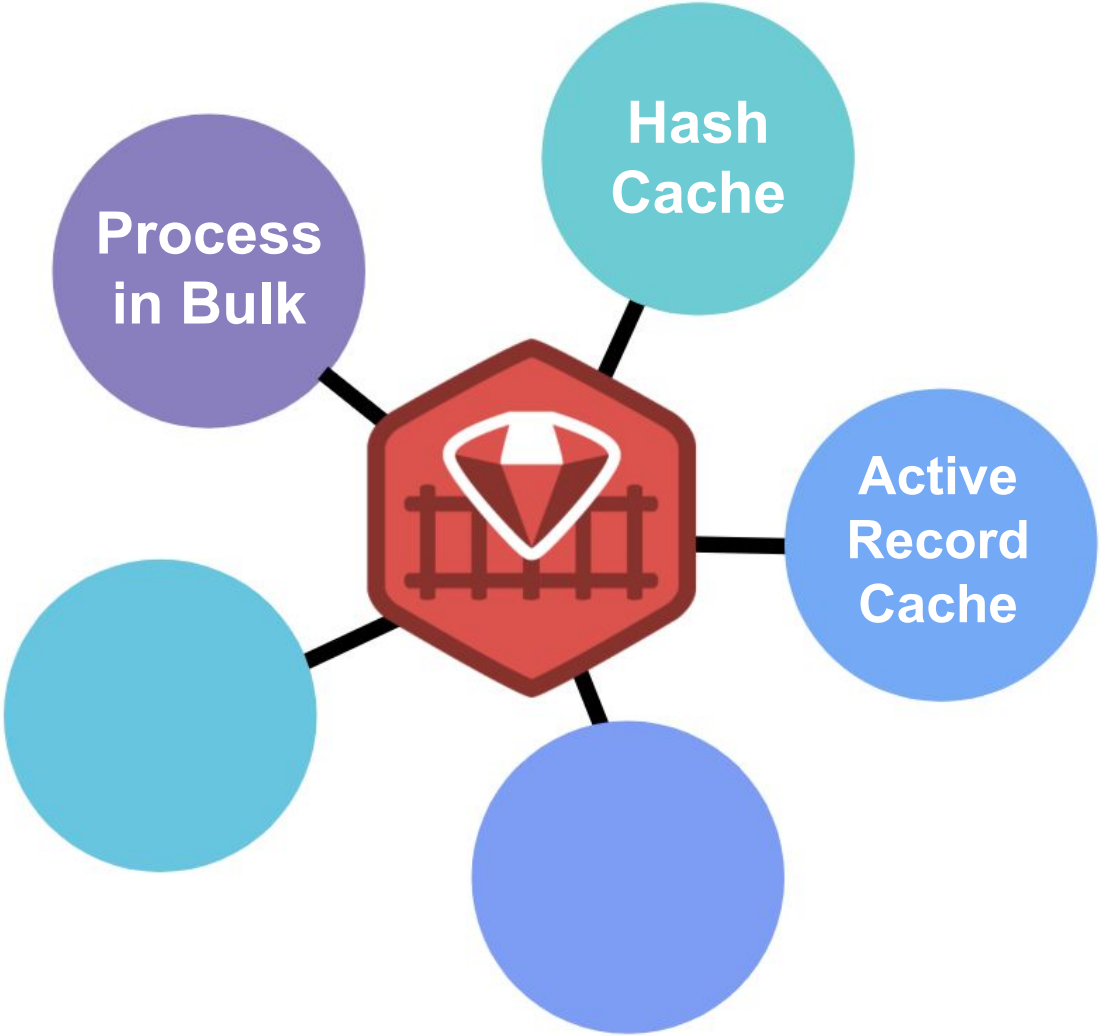
```
module Octopus
  class Proxy
    attr_accessor :proxy_config
    delegate :current_shard, :current_shard=,
             :current_slave_group, :current_slave_group=,
             :shard_names, :shards_for_group, :shards, :sharded,
             :config, :initialize_shards, :shard_name, to: :proxy_config, prefix: false
  end
end
```

@molly_struve

KENNA Security

Know your gems

KENNA Security



KENNA Security

Avoid making datastore hits you don't need

KENNA Security

```
User.where(:id => user_ids).each do |user|
    # Lots of user processing
end
```

KENNA Security

KENNA Security

```
(pry)> User.where(:id => [])
  User Load (1.0ms)  SELECT `users`.* FROM `users` WHERE 1=0
=> []
```

KENNA Security

@molly_struve

```ruby
return unless user_ids.any?
User.where(:id => user_ids).each do |user|
    # Lots of user processing
end
```

KENNA Security

```
(pry)> Benchmark.realtime do
>   10_000.times { User.where(:id => []) }
> end
=> 0.5508159045130014
(pry)> Benchmark.realtime do
>   10_000.times do
>     next unless ids.any?
>     User.where(:id => [])
>   end
> end
=> 0.0006368421018123627
```

KENNA Security

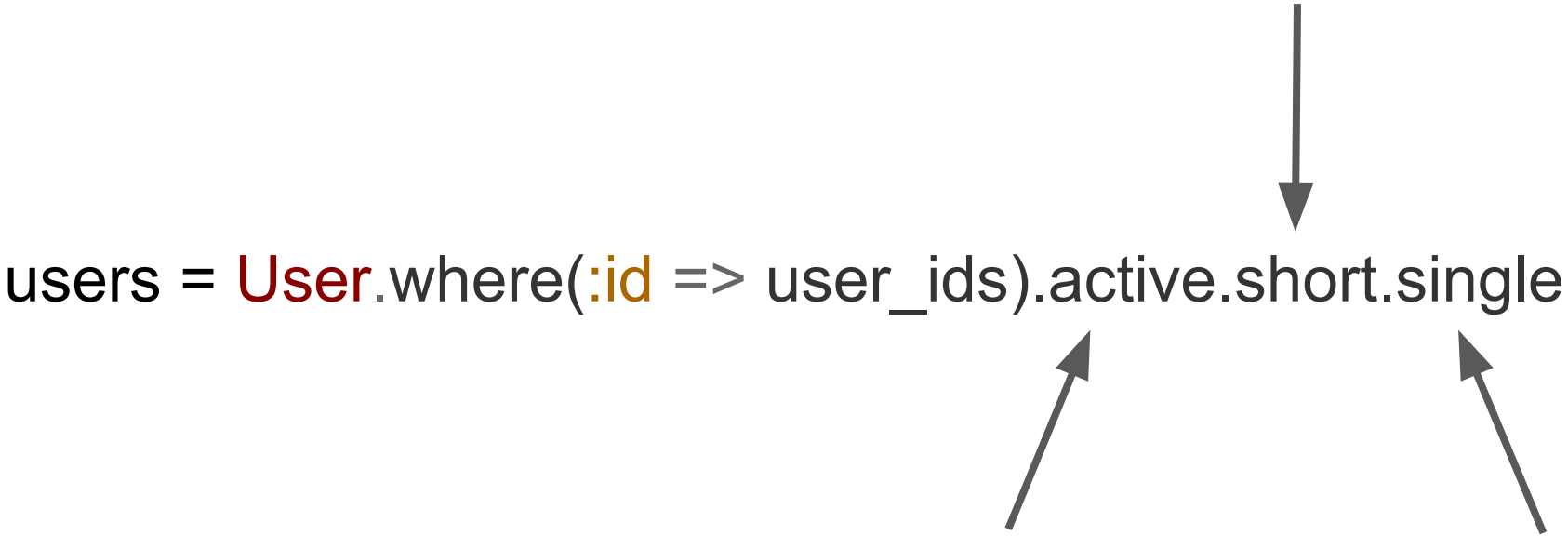
```
(pry)> Benchmark.realtime do
>   10_000.times { User.where(:id => []) }
> end
=> 0.5508159045130014
```

Hitting the database is slow!

KENNA Security

```
User.where(:id => user_ids).each do |user|
    # Lots of user processing
end
```

KENNA Security



KENNA Security

.none

method **none**

Ruby on Rails latest stable (v4.2.7) - 0 notes - Class: ActiveRecord::QueryMethods

1.0.0 1.1.1 1.1.6 1.2.0 1.2.6 2.0.0 2.0.3 2.1.0 2.2.1 2.3.2 2.3.8 3.0.0 3.0.5 3.0.9 3.1.0 3.2.1 3.2.3 3.2.8 3.2.13 4.0.2 4.1.8 4.2.1 **4.2.7**

none ( ) *public*

Returns a chainable relation with zero records.

The returned relation implements the Null Object pattern. It is an object with defined null behavior and always returns an empty array of records without querying the database.

.none in action...

```
(pry)> User.where(:id => []).active.tall.single
  User Load (0.7ms)  SELECT `users`.* FROM `users` WHERE 1=0 AND
`users`.`active` = 1 AND `users`.`short` = 0 AND `users`.`single` = 1
=> []

(pry)> User.none.active.tall.single
=> []
```

KENNA Security

@molly_struve

Gems

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

DISPLAYING GEMS **1** - **30** OF **9619** IN TOTAL

KENNA Security

Logging

```
pry(main)> Rails.logger.level = 0
```

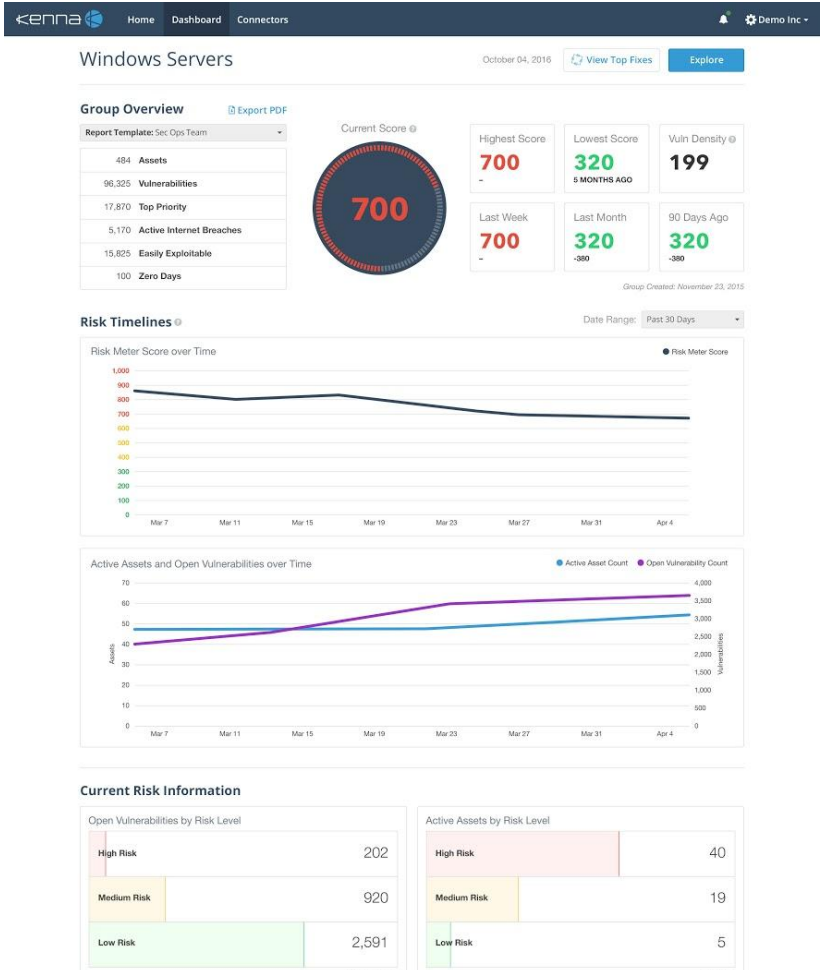
```
pry(main)> Search.connection.transport.logger = Logger.new(STDOUT)
```

```
$ redis-cli monitor > commands-redis-2018-10-01.txt
```

KENNA Security

Preventing useless datastore hits

@molly_struve



KENNA Security

@molly_struve

KENNA Security

Investigating Existing Reports

KENNA Security

```
(pry)> Report.active.count
=> 25842
(pry)> Report.average_asset_count
=> 1657
(pry)> Report.zero_assets.count
=> 10805
```

KENNA Security

KENNA
Security

@molly_struve

KENNA Security

return if report.asset_count.zero?

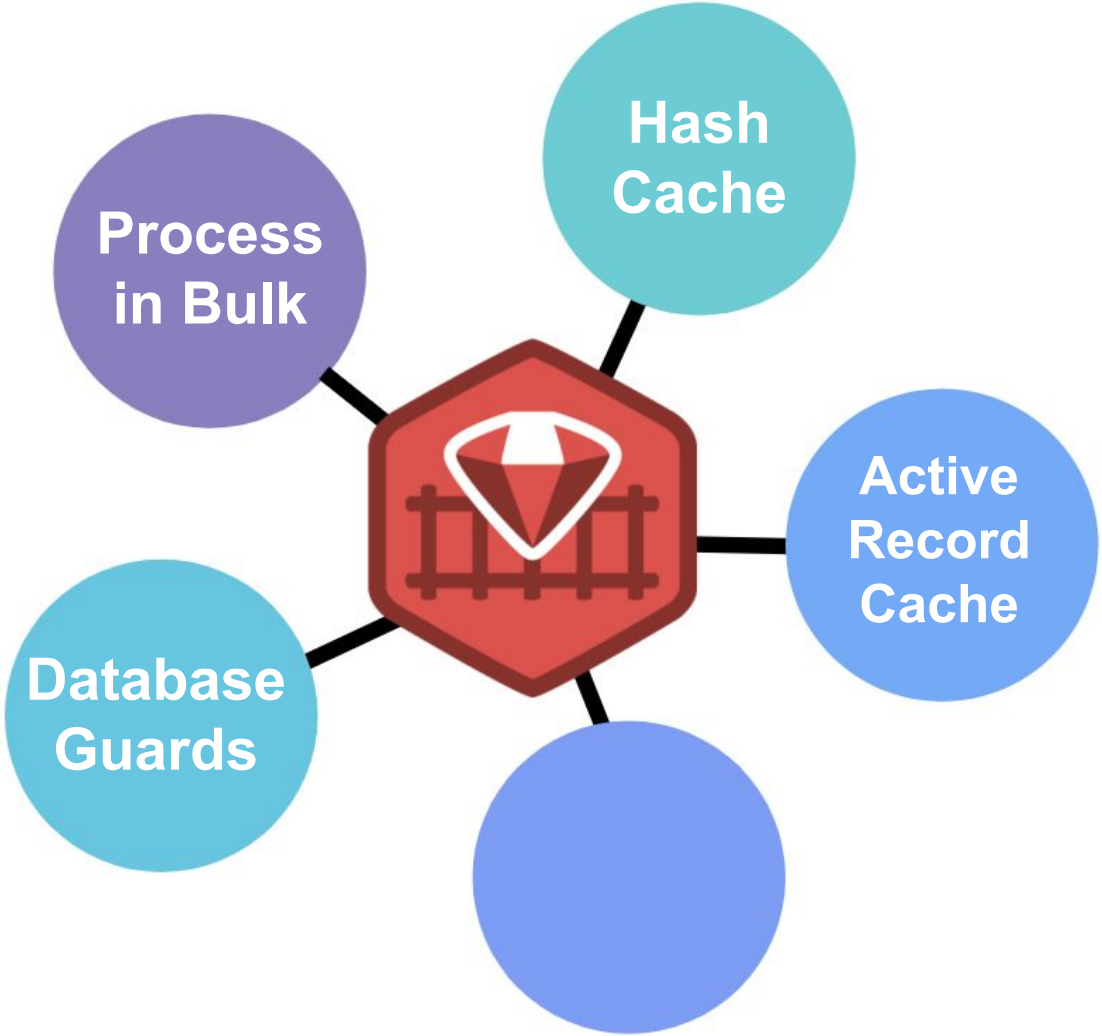
KENNA Security

10+ hrs

KENNA Security

3 hrs

KENNA Security

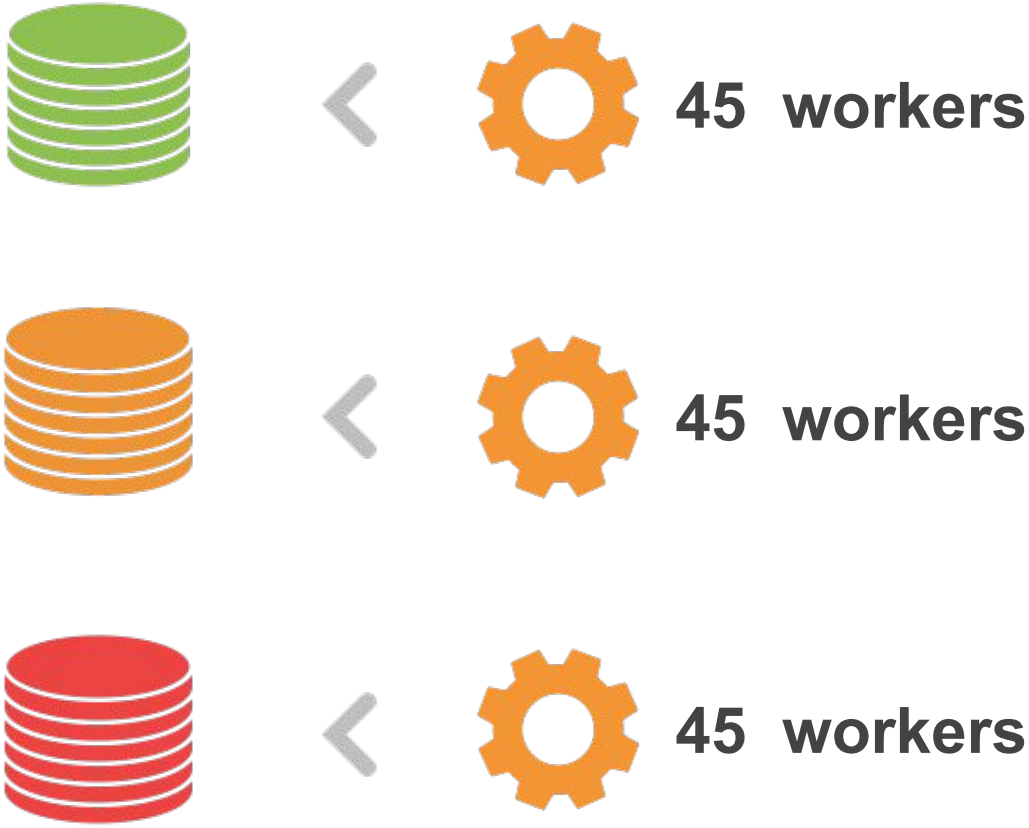


KENNA Security

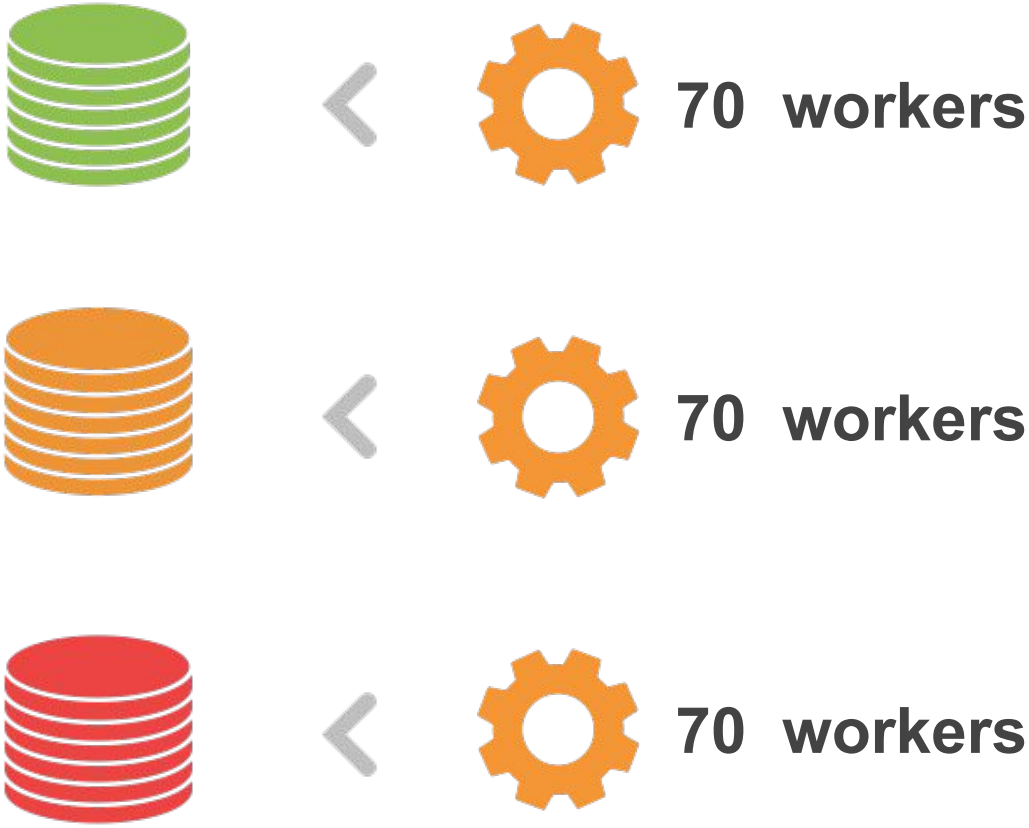
Resque Workers

Redis

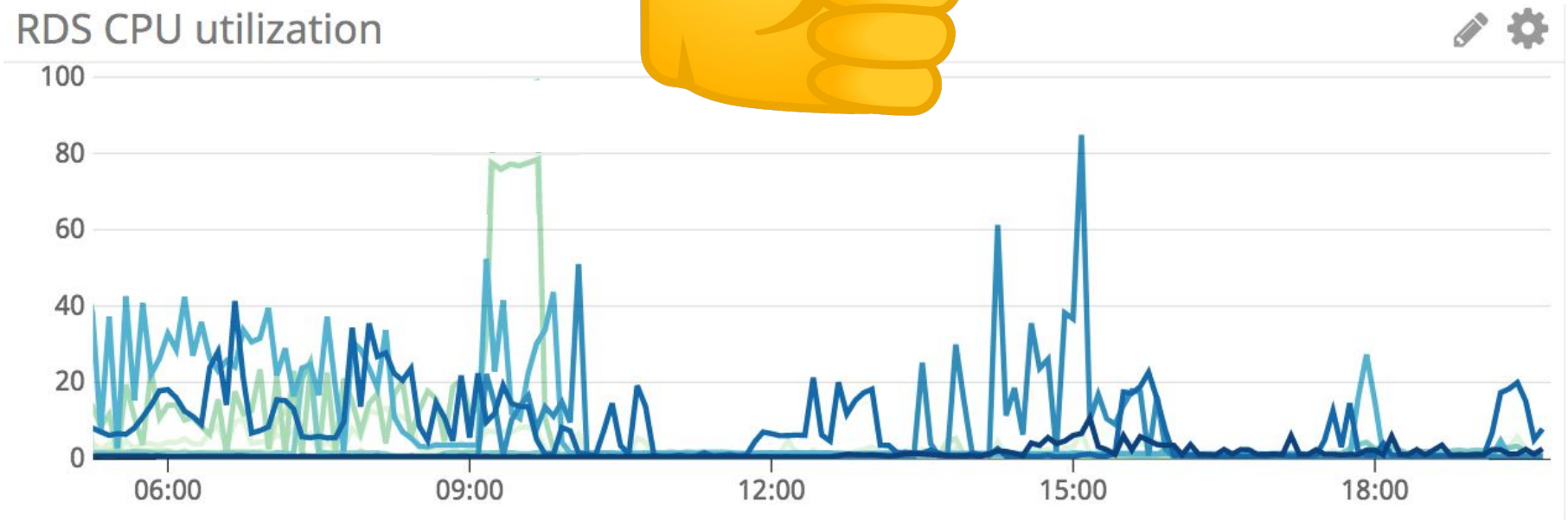
KENNA Security



KENNA Security



KENNA Security



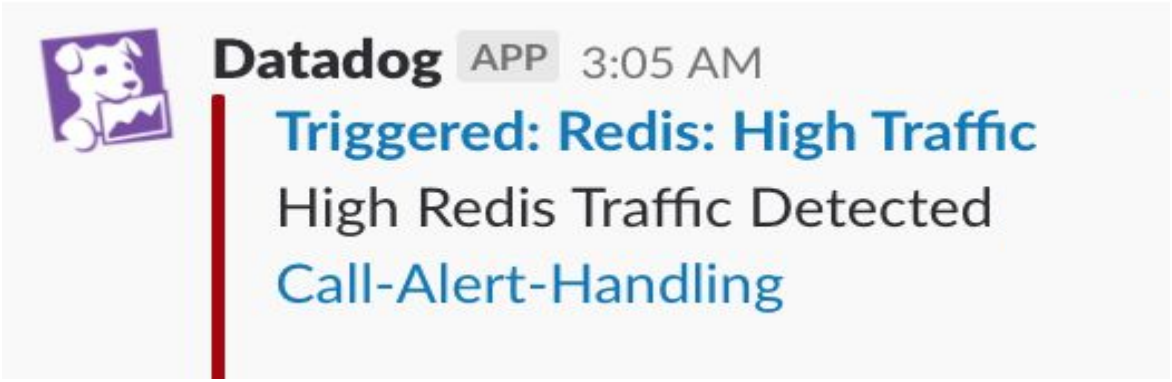
@molly_struve

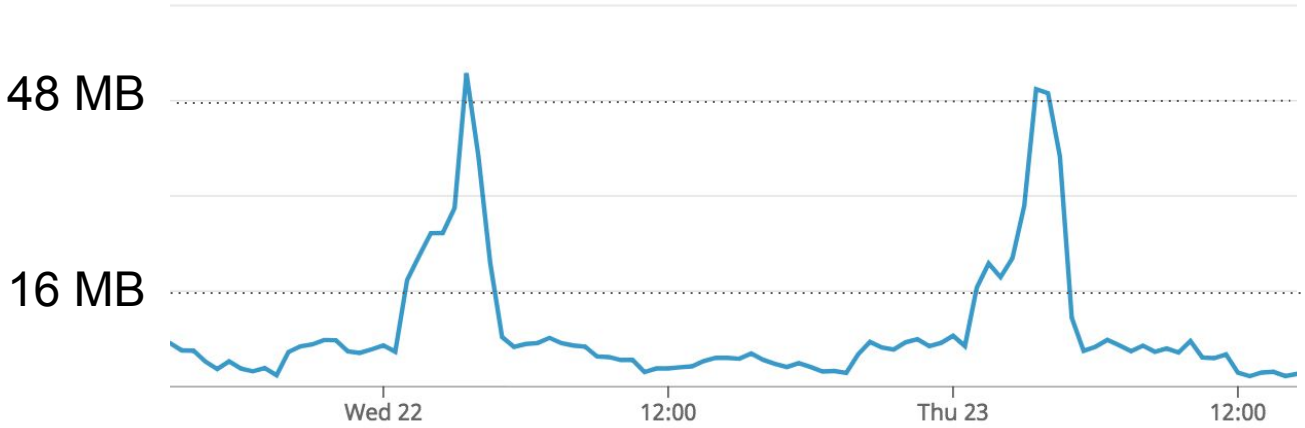
KENNA Security

KENNA
Security

@molly_struve

KENNA Security

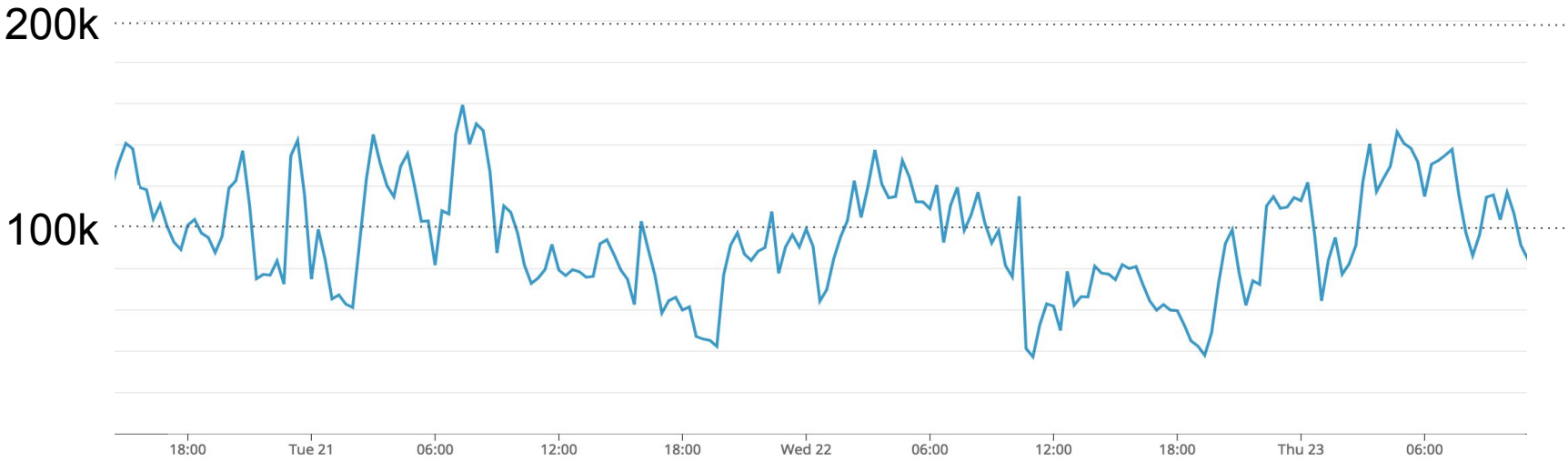




@molly_struve

Redis Requests

KENNA Security



<

70 workers

KENNA Security

Redis::ConnectionError

A month ago - 09/05/18 @ 4:34 pm -05:00

Summary | Comments | Backtrace | Context | Params | Environment | History

Status	Unresolved — We won't alert you again until the error is resolved.
Message	`Redis::ConnectionError: Connection lost (ECONNRESET)`

KENNA
Security

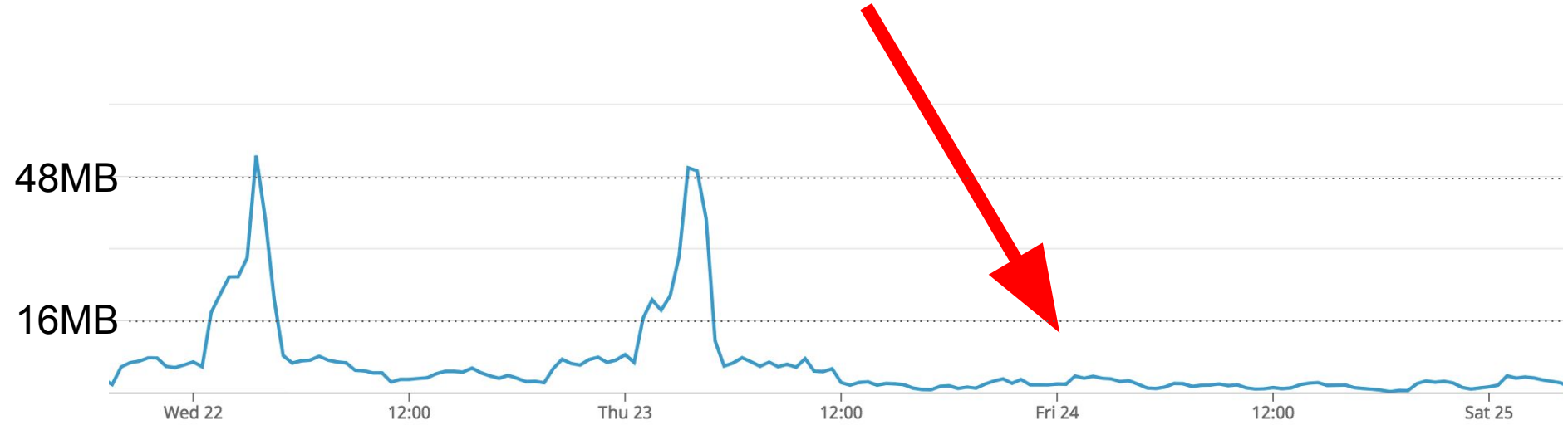
KENNA Security

Redis Requests

KENNA Security

KENNA Security

Redis Network Traffic



KENNA Security

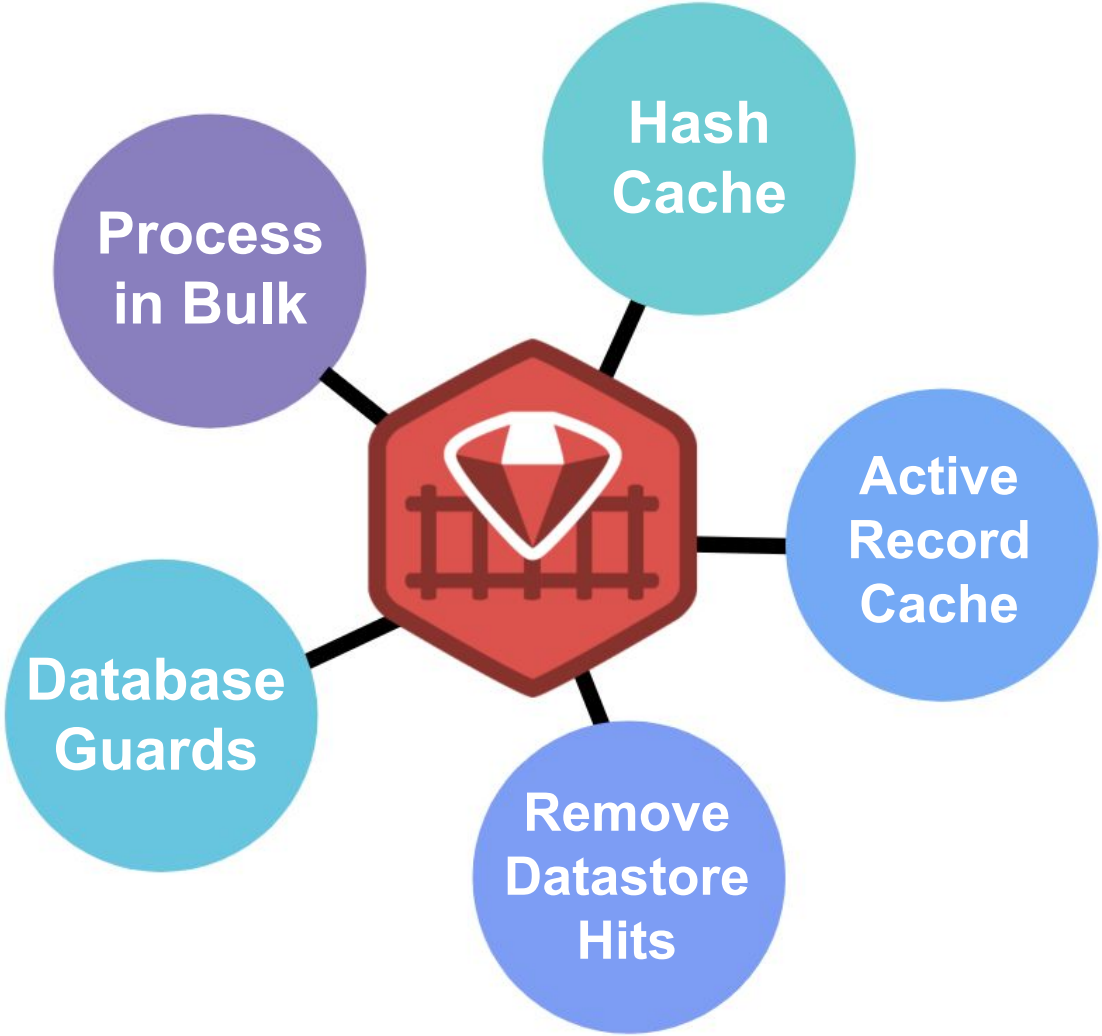
Redis::ConnectionError

A month ago - 09/05/18 @ 4:34 pm -05:00

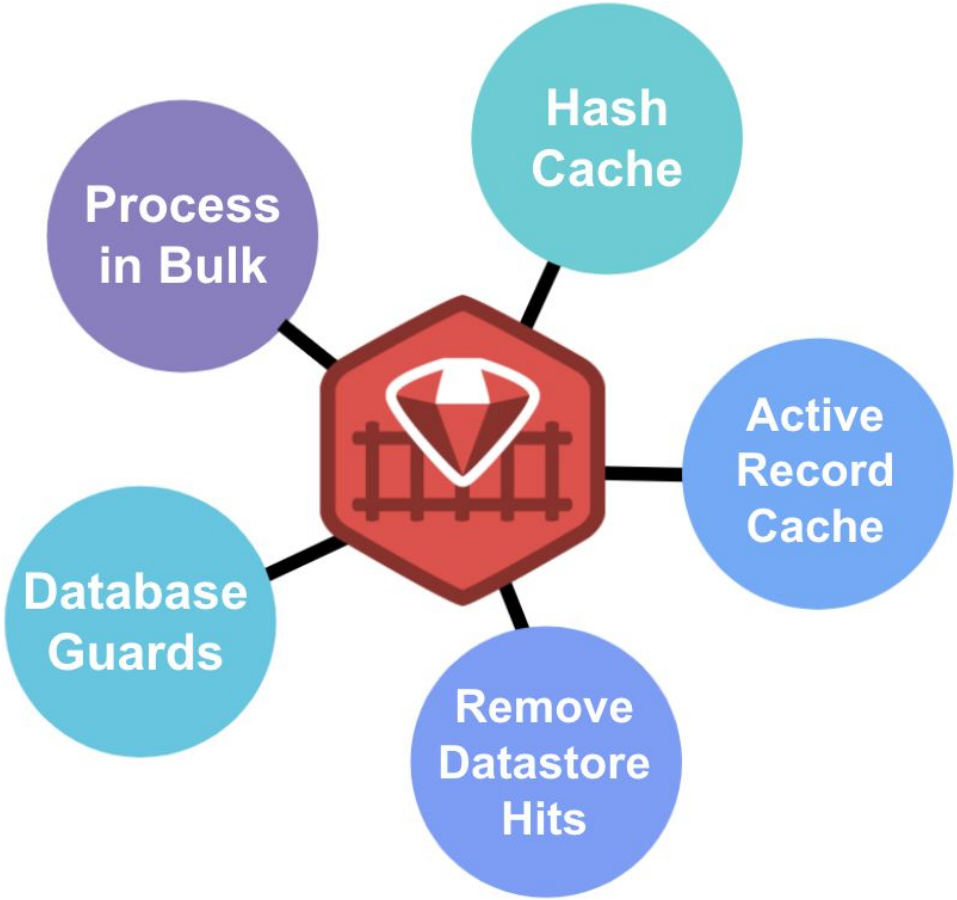
Summary | Comments | Backtrace | Context | Params | Environment | History

Status	Resolved — We won't alert you again until the error is resolved.
Message	`Redis::ConnectionError: Connection lost (ECONNRESET)`

KENNA Security



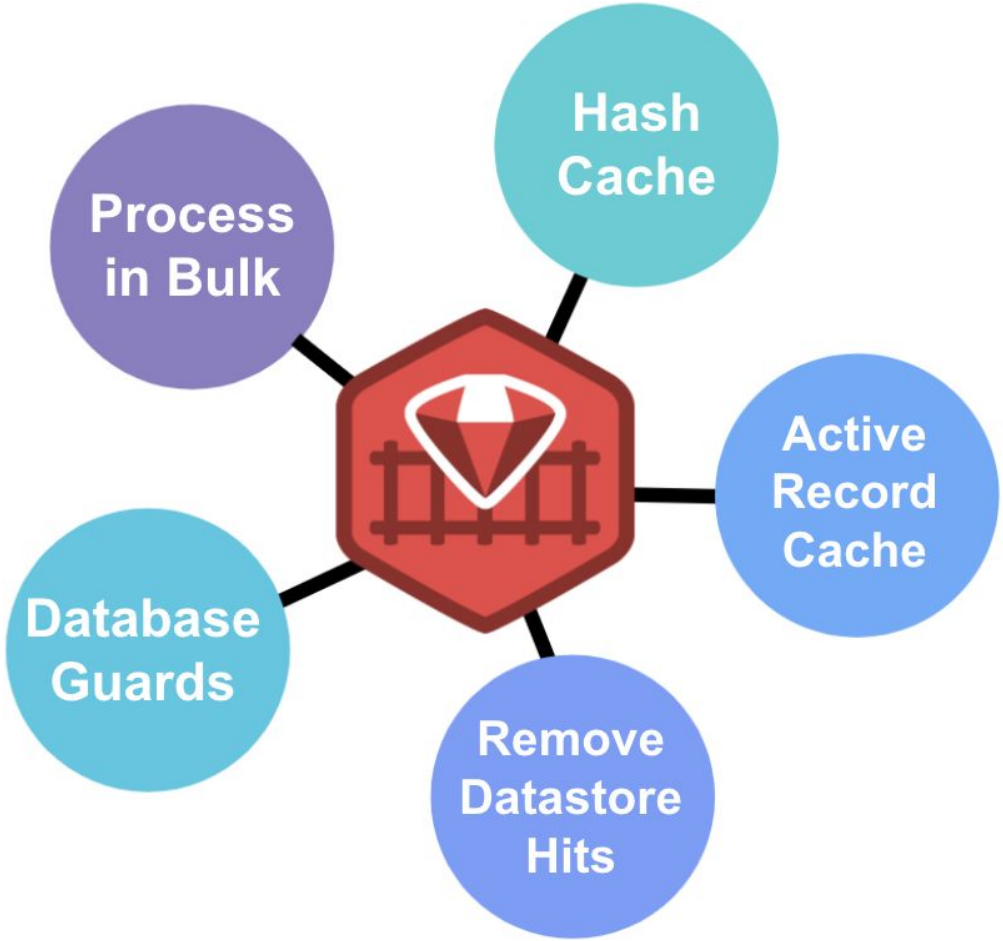
KENNA Security



Every datastore hit COUNTS

KENNA Security

KENNA Security



Questions

Contact

KENNA Security

https://github.com/mstruve

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

@molly_struve

molly.struve@gmail.com