



elastic

Core Elasticsearch Operations

An Elastic Training Course

Puneeth Kumar Bodigere - 17-Sep-2017 Optum

training.elastic.co

Course: Core Elasticsearch Operations

Version 5.4.1

© 2015-2017 Elasticsearch BV. All rights reserved. Decompiling, copying, publishing and/or distribution without written consent of Elasticsearch BV is strictly prohibited.

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Lab Prep Checklist

- Download the PDF and zip file from links provided in email
- Minimum requirements:
 - 20% or more free disk space
 - Install Java
 - version 1.8.0_20 minimum/1.8.0_73 or newer recommended
 - 64-bit JDK preferred
 - set **JAVA_HOME** environment variable
- Unzip the downloaded lab zip file on your hard drive
 - this creates a directory called **CoreOperations**
 - do *not* unzip the lab file into a folder with spaces in the name

Agenda and Introductions

Puneeth Kumar Bodigutta - 17-Sep-2014 Optum

Course Agenda

1 Introduction to Elasticsearch

2 Installation and Configuration

3 Working with Nodes

4 Indexes

5 Cluster Health

6 Mappings and Analysis

7 Index Management

8 Capacity Planning

9 Cluster Management

10 Monitoring

11 Upgrading a Cluster

12 Production Checklist

Introductions

- Name
- Company
- What do you do?
- What are you using Elasticsearch for?
- What do you hope to get out of this training?

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Logistics

- Facilities
- Emergency Exits
- Restrooms
- Breaks/Lunch

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Chapter 1

Introduction to Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- The Story of Elasticsearch
- The Components of Elasticsearch
- Indexing Data
- CRUD Operations
- The Kibana Console
- Starting and Stopping Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2017

The Story of Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2011 Forum



Once upon a time...

- As any good story begins, “Once upon a time...”
 - More precisely: in 1999, Doug Cutting created an open-source project called *Lucene*
- Lucene is:
 - a *search engine library* entirely written in Java
 - a top-level Apache project, as of 2005
 - great for full-text search
- But, Lucene is also:
 - a library (you have to incorporate it into your application)
 - challenging to use
 - not originally designed for scaling

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

“

**Search is something
that any application
should have”**

Shay Banon
Creator of Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2017 Optum

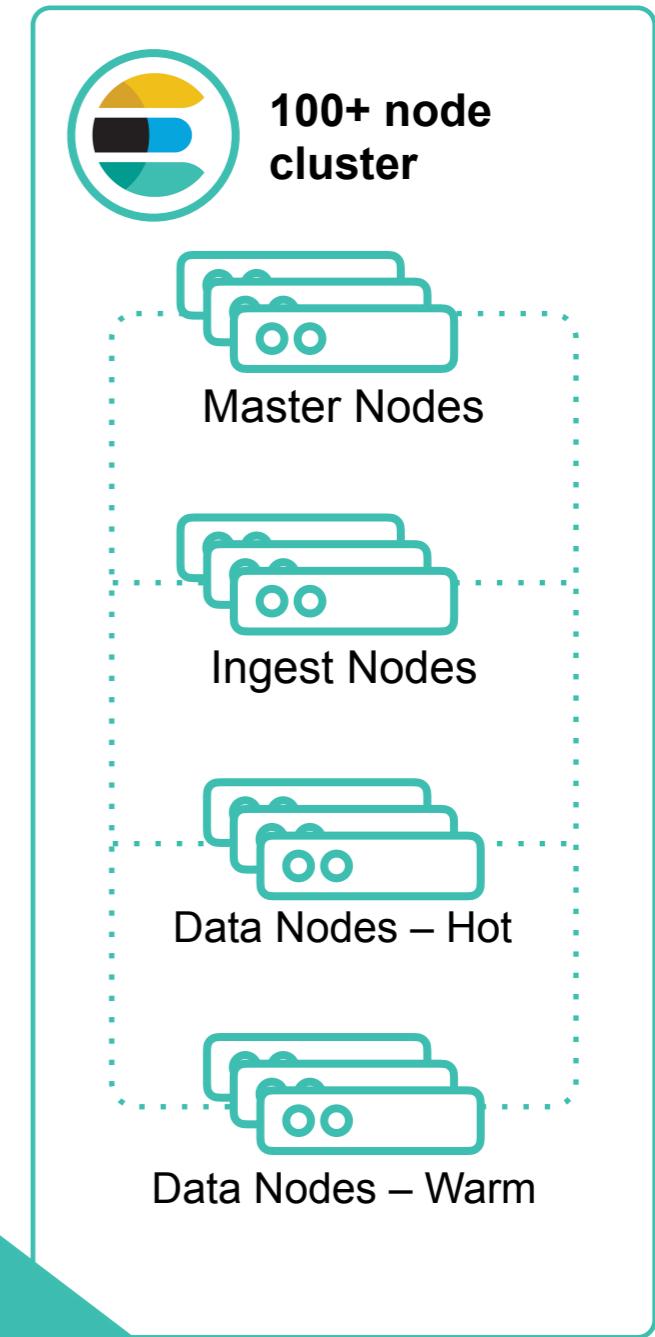
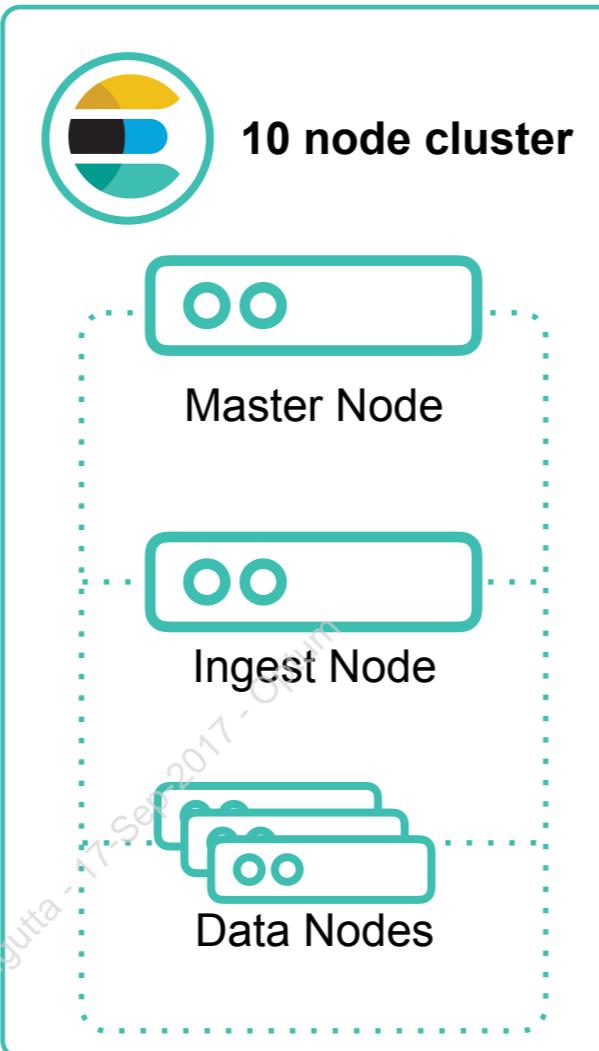
The Birth of Elasticsearch

- In 2004, Shay Banon developed a product called **Compass**
 - Built on top of Lucene, Shay's goal was to have search integrated into Java applications as simply as possible
- The need for **scalability** became a top priority
- In 2010, Shay completely rewrote Compass with two main objectives:
 1. *distributed from the ground up in its design*
 2. *easily used by any other programming language*
- He called it **Elasticsearch**
 - ...and we all lived happily ever after!
- Today Elasticsearch is the most popular enterprise search engine



1. Distributed search:

- Elasticsearch is distributed and scales horizontally:



A **node** is an instance of Elasticsearch

A **cluster** is a collection of Elasticsearch nodes

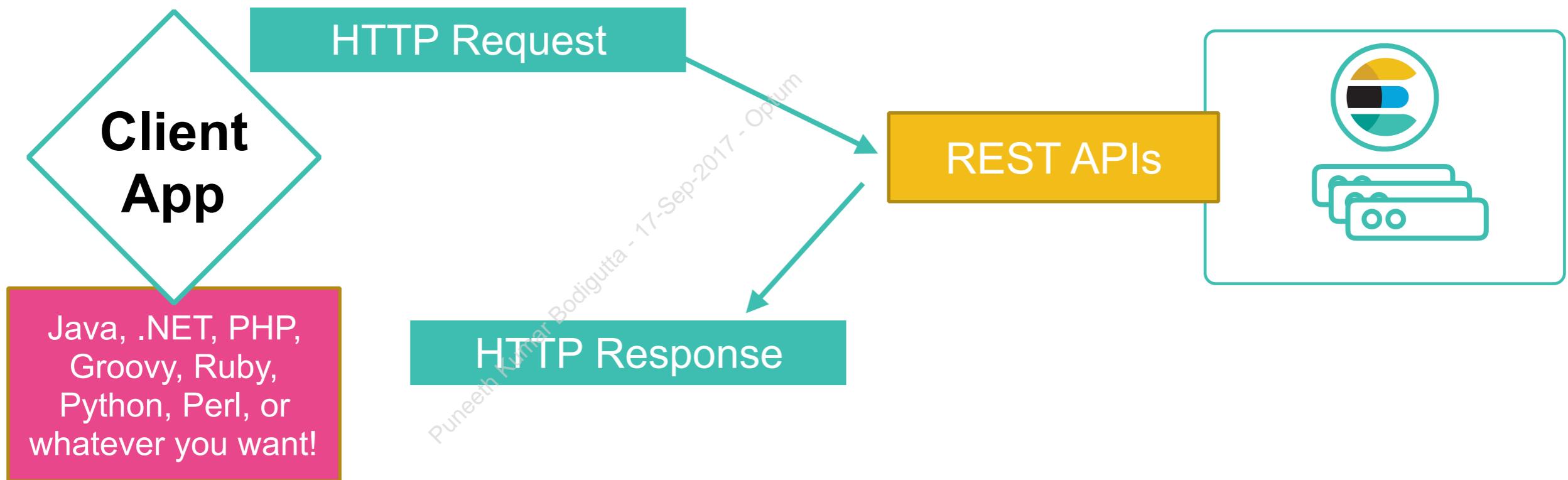
Your cluster can grow as your needs grow

Puneeth Kumar Bodigutta - IT Set 2017 - Cluster



2. Easily used by other languages:

- Elasticsearch provides REST APIs for communicating with a cluster over HTTP
 - Allows client applications to be written in any language



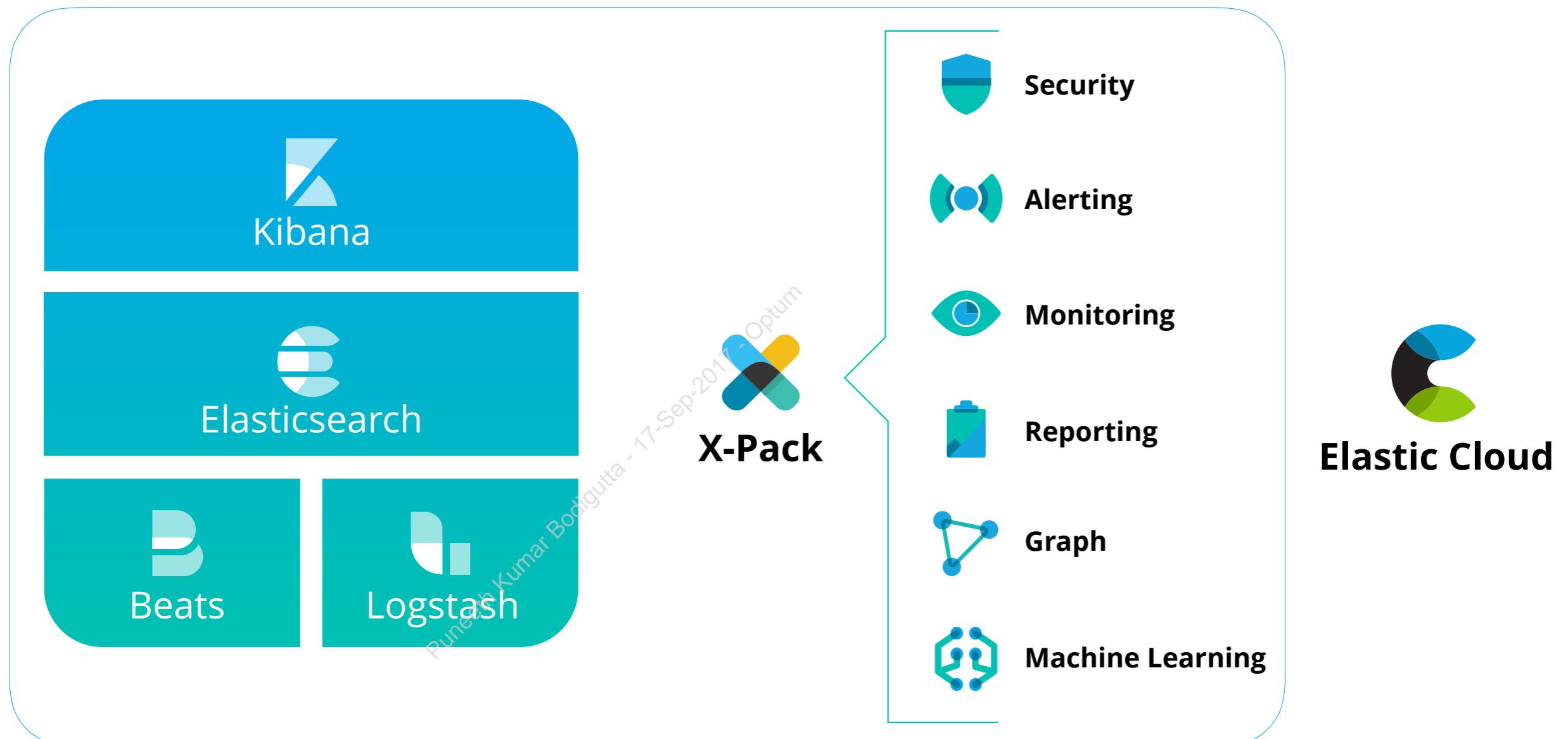
Elasticsearch History

- **0.4:** first version was released in February, 2010
- **1.0:** released in January, 2014
- **2.0:** released in October, 2015
- **5.0:** released in October, 2016
- Why the big jump in version number?
 - The Elastic Stack...

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

The Elastic Stack

- Elasticsearch is a component of the *Elastic Stack*



The Components of Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2017 | Octum



The Components of Elasticsearch

- The main components of Elasticsearch are:
 - ***node***: an instance of Elasticsearch
 - ***cluster***: one or more nodes that work together to serve the same data and API requests
 - ***document***: a piece of data that you want to search
 - ***index***: a collection of documents that have somewhat similar characteristics
 - ***shard***: a single piece of an Elasticsearch index

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



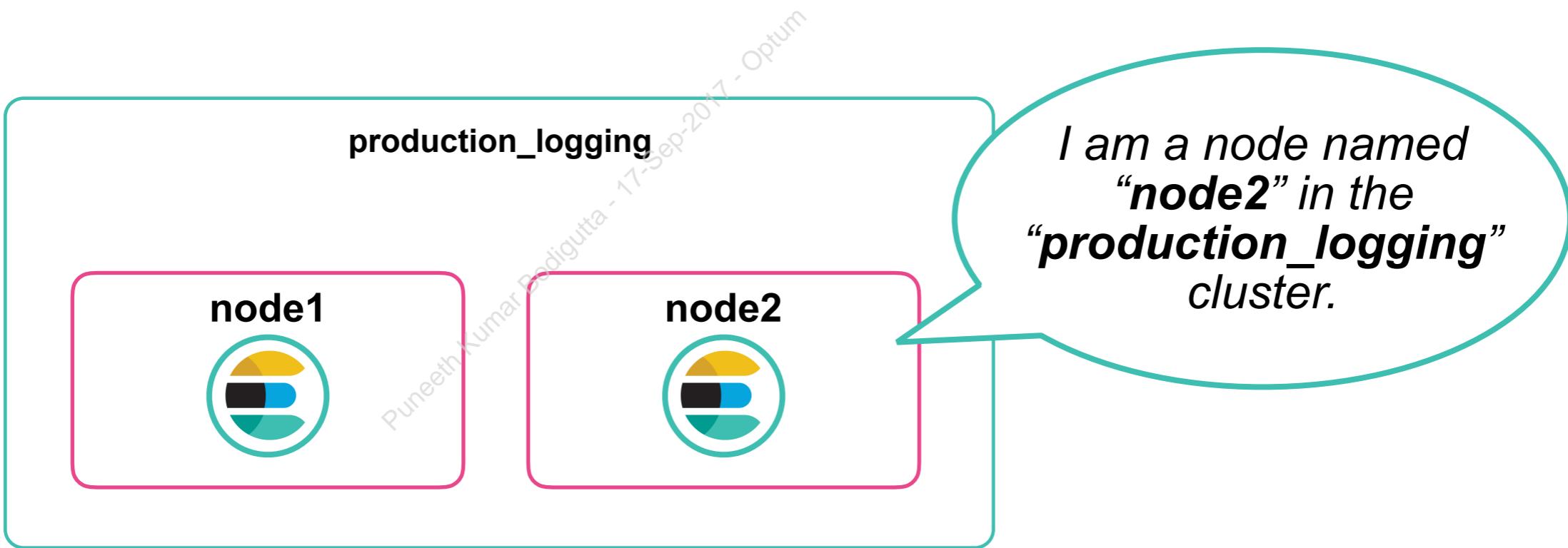
Cluster

- Everything in Elasticsearch happens in the context of a *cluster*
- A cluster is multiple instances of Elasticsearch working together in a distributed manner
 - Every cluster has a *name* (defaults to “elasticsearch”)
 - Every instance of Elasticsearch belongs to a cluster



What is a Node?

- A *node* is an instance of Elasticsearch
 - a Java process that runs in a JVM
- Every node has a name
 - and is a member of a single cluster
- A node is typically deployed 1-to-1 to a host



Document

- A **document** can be any text or numeric data you want to search and/or analyze
 - Specifically, a **document** is a top-level object that is serialized into JSON and stored in Elasticsearch
- Every document has a **unique ID**
 - which either you provide, or Elasticsearch generates for you

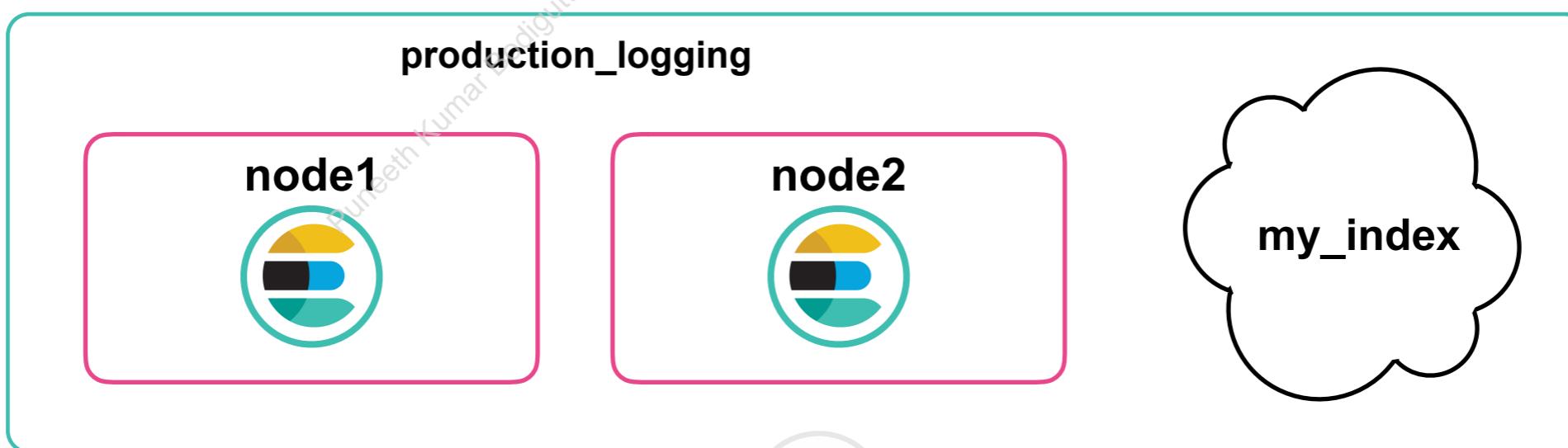
```
{  
  "username" : "kimchy",  
  "tweet" : "Search is something that any application should have!",  
  "tweet_time" : "2017-02-17T23:09:00Z"  
}
```

Documents must be in
the JSON format



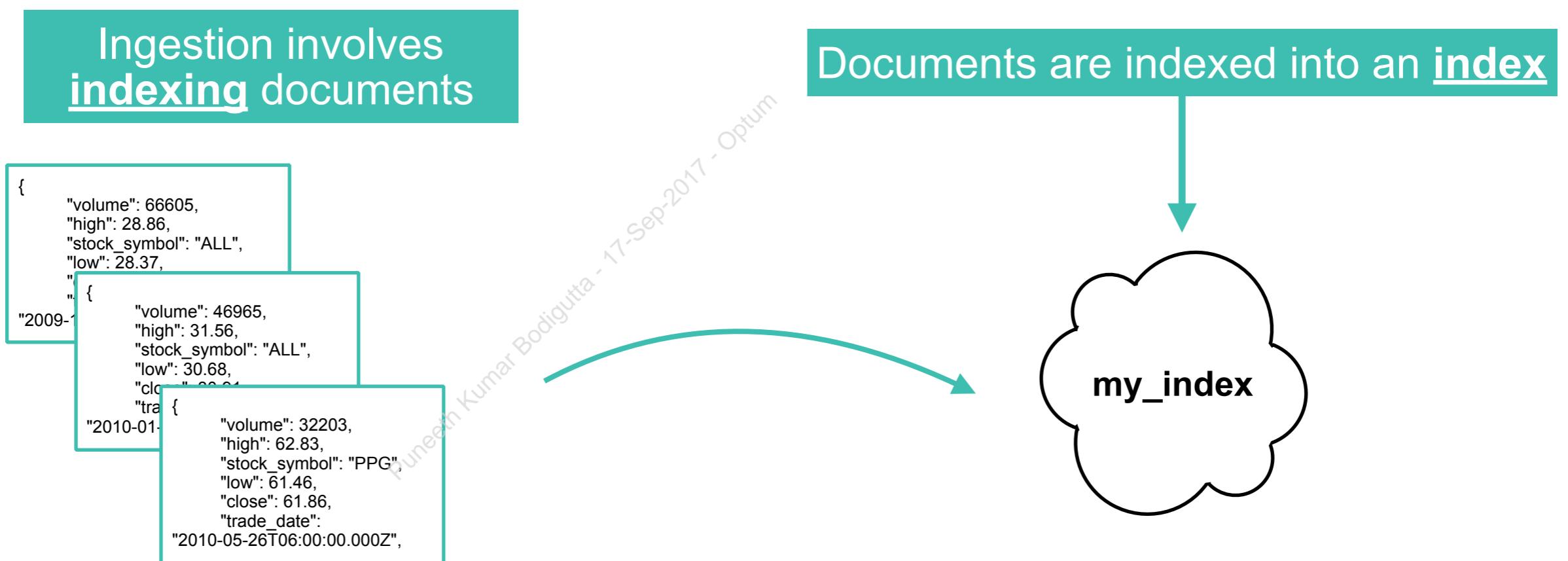
Index

- An **index** in Elasticsearch is a *logical* way of grouping data:
 - an index has a *mapping* that defines the fields in the index
 - an index is a *logical namespace* that maps to where its contents are stored in the cluster
- There are two different concepts in this definition:
 - an index has some type of data schema mechanism
 - an index has some type of mechanism to distribute data across a cluster



When we say “index”, we mean...

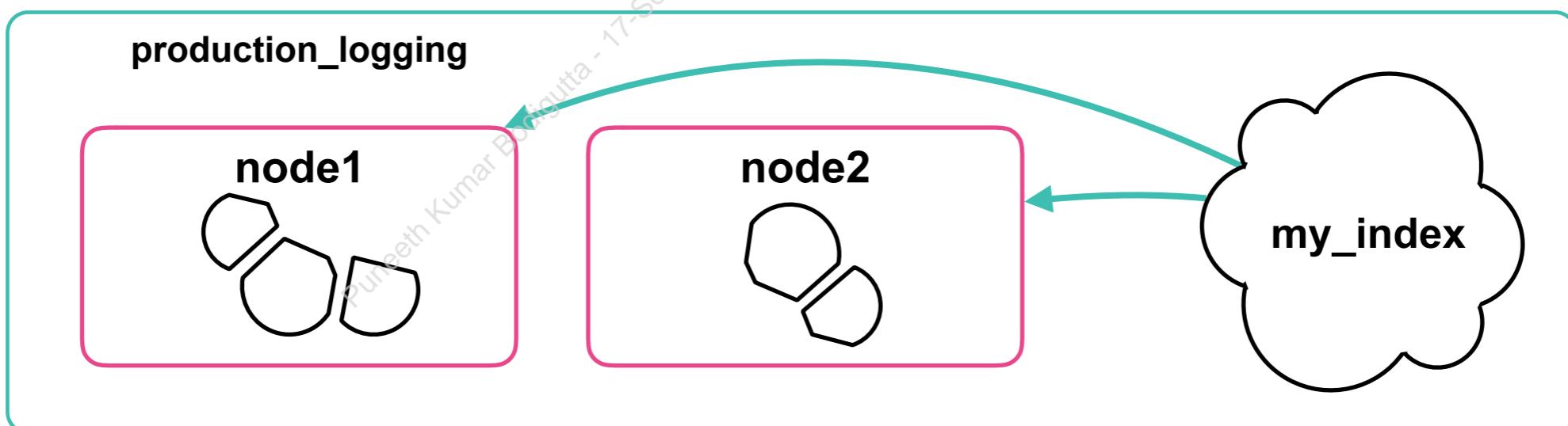
- We use the word “index” in multiple ways:
 - **Noun:** a document is put into an *index* in Elasticsearch
 - **Verb:** to *index* a document is to put the document into an index in Elasticsearch



Shard

- A **shard** is a single piece of an Elasticsearch index
 - Indexes are partitioned into shards so they can be distributed across multiple nodes
- Each shard is a standalone Lucene index
 - The default number of shards for an index is 5

shards are distributed across the nodes of a cluster

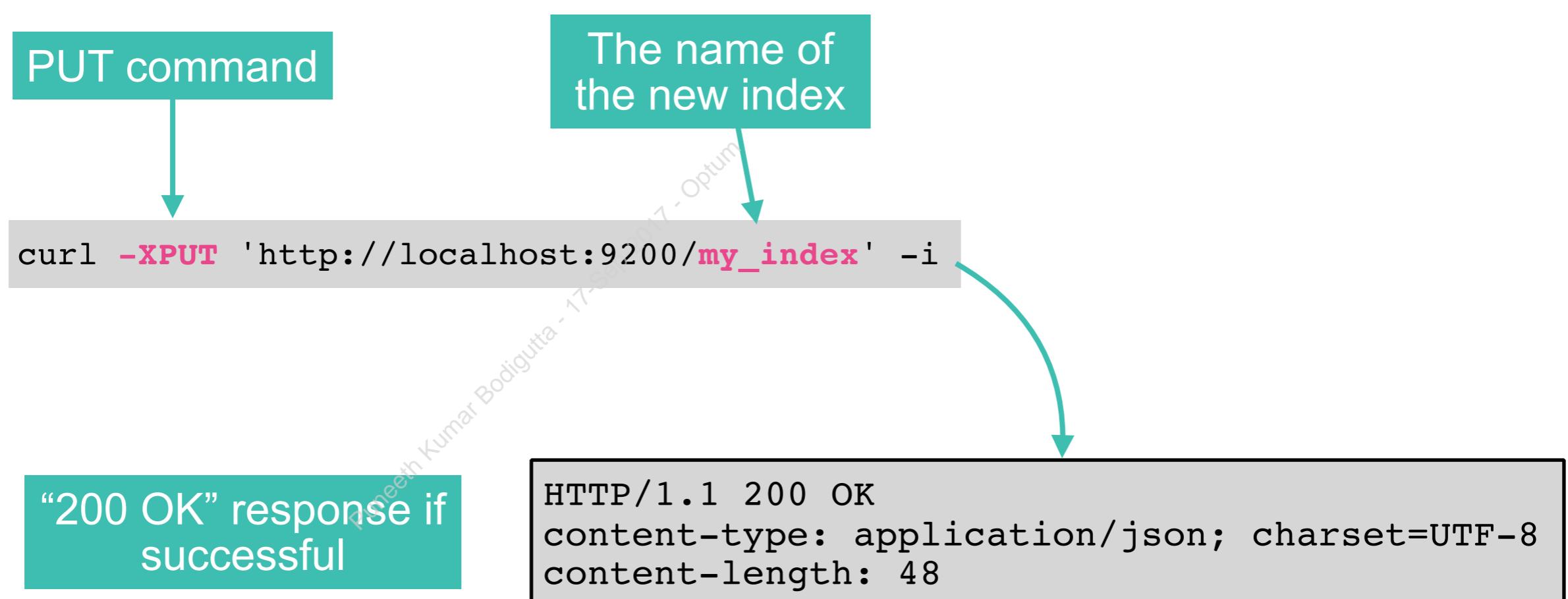


Indexing Data

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Define an Index

- Clients communicate with a cluster using Elasticsearch's REST APIs
- An index is defined using the **Create Index API**, which can be accomplished with a simple **PUT** command:



Define the Document

- The JSON format consists of “**fieldvalue**

```
{  
  "username" : "kimchy",  
  "tweet" : "Search is something that any application should have",  
  "tweet_time" : "2017-03-01T15:19:03+00:00"  
}
```

This JSON document
has 3 fields:
“username”, **“tweet”**
and **“tweet_time”**

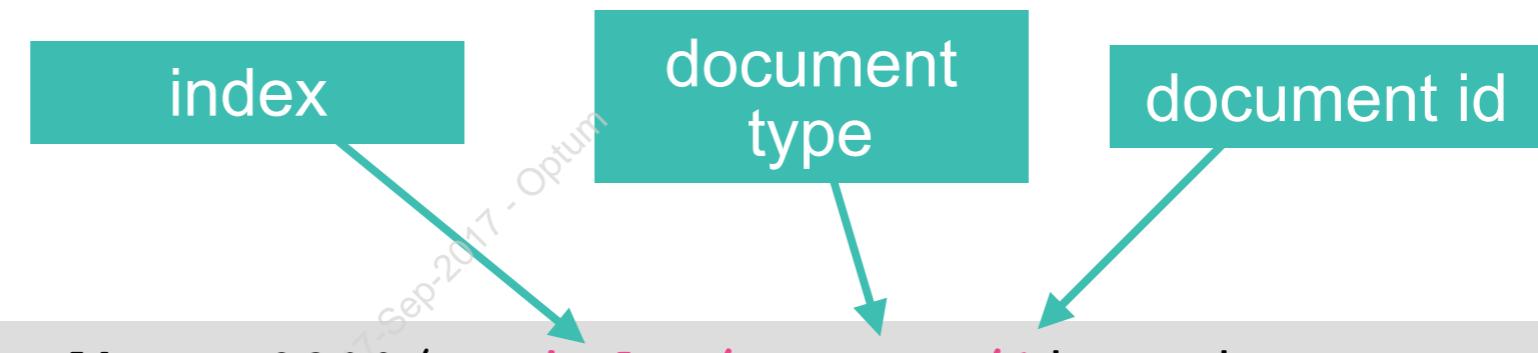
These are the values

Puneeth Kumar Bodigutta - 17-Sep-2017 - Open



Index a Document

- The **Index API** is used to index a document
- Use a **PUT** again, but this time send the document in the body of the **PUT** request:
 - notice you specify a document **type** and a **unique ID**
 - “201 Created” response if successful



```
curl -XPUT 'http://localhost:9200/my_index/my_type/1' -H 'Content-Type: application/json' -d '  
{  
  "username" : "kimchy",  
  "tweet" : "Search is something that any application should have",  
  "tweet_time" : "2017-03-01T15:19:03+00:00"  
}'
```



Dynamic Index Creation

- Elasticsearch will create the index for you during the indexing of a document, if the index is not already created:
 - set **action.auto_create_index** to **false** to disable this behavior
 - we could have simply executed the command on the previous slide (and skipped that first “**PUT my_index**” command)

my_index is created if it does not exist

my_type type is created if it does not exist

```
curl -XPUT 'http://localhost:9200/my_index/my_type/1' -H 'Content-Type: application/json' -d '  
{  
  "username" : "kimchy",  
  "tweet" : "Search is something that any application should have",  
  "tweet_time" : "2017-03-01T15:19:03+00:00"  
}'
```

Index Without Specifying an ID

- You can leave off the id and let Elasticsearch generate one for you:
 - But notice that only works with **POST**, not **PUT**
 - The generated id comes back in the response

POST instead of PUT

```
curl -XPOST 'http://localhost:9200/my_index/my_type/'  
-H 'Content-Type: application/json' -d '  
{  
    "username" : "elastic",  
    "tweet" : "#kibana is the window into #elasticstack",  
    "tweet_time" : "2017-03-01T15:33:07+00:00"  
}'
```

No id specified

CRUD Operations

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

CRUD Operations

Index	<pre>PUT my_index/doc/4 { "username" : "kimchy", "comment" : "I like search!" }</pre>
Create	<pre>PUT my_index/doc/4/_create { "username" : "kimchy", "comment" : "I like search!" }</pre>
Read	<pre>GET my_index/doc/4</pre>
Update	<pre>POST my_index/doc/4/_update { "doc" : { "comment" : "I love search!!" } }</pre>
Delete	<pre>DELETE my_index/doc/4</pre>

Puneeth Kumar Bodigere - 17-Nov-2017



Retrieving a Document

- Use **GET** to retrieve an indexed document
 - Returns 200 if the document is found
 - Returns a 404 error if the document is not found

```
curl -XGET 'http://localhost:9200/my_index/my_type/1'  
-H 'Content-Type: application/json'
```

```
{  
  "_index": "my_index",  
  "_type": "my_type",  
  "_id": "1",  
  "_version": 1,  
  "found": true,  
  "_source": {  
    "username": "kimchy",  
    "tweet": "Search is something that any application should have",  
    "tweet_time": "2017-03-01T15:19:03+00:00"  
  }  
}
```



Reindexing a Document...

- We already have a document with **id = 1**
 - What do you think is the result of the following **PUT** command?

```
curl -XPUT 'http://localhost:9200/my_index/my_type/1'  
-H 'Content-Type: application/json' -d '  
{  
    "new_field" : "new_value"  
}'
```

Puneeth Kumar Bodigutta - 17-Sep-2017



...Overwrites the Document

- Notice the old field/value pairs of the document are gone
 - the old document is deleted, and the new one gets indexed
- Notice every document has a **_version** that is incremented whenever the document is changed

```
curl -XGET 'http://localhost:9200/my_index/my_type/1'  
-H 'Content-Type: application/json'
```

The original document is overwritten, and the version incremented to 2

```
{  
  "_index": "my_index",  
  "_type": "my_type",  
  "_id": "1",  
  "_version": 2,  
  "found": true,  
  "_source": {  
    "new_field": "new_value"  
  }  
}
```

Updating a Document

- You update a document using the `_update` endpoint
 - and adding a “`doc`” section to the request

```
curl -XPOST 'http://localhost:9200/my_index/my_type/1/_update' -H  
'Content-Type: application/json' -d '  
{  
  "doc": {  
    "username" : "uri"  
  }  
}'  
  
curl -XGET 'http://localhost:9200/my_index/my_type/1' -H 'Content-Type:  
application/json'
```

_version is now 3

Puneeth Kumar Bodigutta - 11/2016

```
{  
  ...  
  "_version": 3,  
  "_source": {  
    "new_field": "new_value",  
    "username": "uri"  
  }  
}
```



The Bulk API

Puneeth Kumar Bodigutta - 17-Sep-2017 - Opnum

Cheaper in Bulk

- The **Bulk API** makes it possible to perform many write operations in a single API call, greatly increasing the indexing speed
 - useful if you need to index a data stream such as log events, which can be queued up and indexed in batches of hundreds or thousands
- Four actions: **create, index, update and delete**
- The response is a large JSON structure with the individual results of each action that was performed
 - The failure of a single action does not affect the remaining actions

Puneeth Kumar Borra - 17-Nov-2017



Syntax of _bulk

- Has a unique syntax based on lines of commands:
 - Each command appears on a single line

```
POST _bulk
{ "create" : { ... } }
{ DOCUMENT }
{ "index" : { ... } }
{ DOCUMENT }
{ "delete" : { ... } }
{ "update" : { ... } }
{ "doc" : { ... } }
```

The line following “**create**” or “**index**” is the document that gets indexed

“**delete**” does not have a following line

An “**update**” is followed by a “**doc**” line



Example of _bulk

```
POST _bulk
{"create" : {"_index" : "customers", "_type": "customer_type", "_id":102}}
{"firstname" : "Maureen", "lastname" : "O'Hara", "address" : "12 Elm St", "city": "Brooklyn"}
{"index" : {"_index" : "customers", "_type": "customer_type", "_id":103}}
{"firstname" : "Lucy", "lastname" : "Liu", "address" : "39 Pine Tree Ln", "city": "Albany"}
{"delete" : {"_index" : "customers", "_type": "customer_type", "_id":84}}
 {"update" : {"_index" : "customers", "_type": "customer_type", "_id":75}}
 {"doc" : {"firstname" : "Thomas"}}
```

The final \n is actually necessary

The Kibana Console

Puneeth Kumar Bodigutta - 17-Sep-2017 . Opum

The Kibana Console

- The commands we have seen so far have used **curl**
 - You can use any method you prefer to interact with the REST API of Elasticsearch
- The **Dev Tool** feature of Kibana provides a UI for sending requests and viewing responses:

The screenshot shows the Kibana Dev Tools Console interface. On the left is a sidebar with icons for Dev Tools, Settings, and Help. The main area has tabs for Dev Tools and Console, with Console selected. The console area contains two panes. The left pane shows a list of numbered requests:

```
1 #####  
2 # Chapter 1: Introduction to Elasticsearch  
3 PUT my_index  
4  
5 PUT my_index/my_type/1  
6 {  
7   "username" : "kimchy",  
8   "tweet" : "Search is something that any application  
should have",  
9   "tweet_time" : "2010-02-17T23:09:00Z"  
10 }  
11  
12 POST my_index/my_type/  
13 {  
14   "username" : "elastic",  
15   "tweet" : "#kibana is the window into #elasticsearch",  
16   "tweet_time" : "2017-01-20T08:31:12Z"  
17 }  
18  
19 GET /my_index/my_type/1  
20
```

The right pane shows the corresponding JSON responses for each request. The first response is:

```
1 {  
2   "_index": "my_index",  
3   "_type": "my_type",  
4   "_id": "1",  
5   "_version": 1,  
6   "found": true,  
7   "_source": {  
8     "username": "kimchy",  
9     "tweet": "Search is something that any application  
should have",  
10    "tweet_time": "2010-02-17T23:09:00Z"  
11  }  
12 }
```

A watermark "Suneeth Kumar Adigun - 17-Sep-2017" is diagonally across the page.

Starting and Stopping Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2021 / Optum



Starting

- When running Elasticsearch from the **.zip** installation, the **bin** folder has binaries for starting it:
 - **elasticsearch**: shell script for Mac and Linux
 - **elasticsearch.bat**: batch script for Windows

```
$ ./bin/elasticsearch
```

- Use **-d** to run as a daemon on Mac/Linux:

```
$ ./bin/elasticsearch -d -p pid
```

-p saves the process id in
the specified file



Configuring

- Several options for defining configuration properties:
 - the `elasticsearch.yml` config file (in the `/config` folder)
 - using `-E` on the command line:

```
./bin/elasticsearch -E cluster.name=production_logging -E node.name=lucy
```

- REST API calls (discussed in more detail later):

```
curl -XPUT 'http://localhost:9200/_cluster/settings'  
-H 'Content-Type: application/json' -d '  
{  
    "persistent" : {  
        "discovery.zen.minimum_master_nodes" : 2  
    }  
}'
```



Stopping Elasticsearch

1. If Elasticsearch is running in the foreground, enter **Ctrl+c** at the command prompt to kill the process
2. Or, you can kill it using the process id:

```
$ kill `cat pid`
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- Elasticsearch is a search and analytics engine built on top of Apache Lucene
- Clients communicate with a cluster using Elasticsearch's REST APIs
- A ***document*** can be any JSON-formatted data you want to search and/or analyze
- An ***index*** in Elasticsearch is a logical way of grouping data
- Adding documents to Elasticsearch is called ***indexing***
- A ***shard*** is a single piece of an Elasticsearch index
- Use **PUT** to define a new index
- Use **PUT** or **POST** to index a document into an index
- Use **GET** to retrieve a document from an index



Quiz

1. **True or False:** Elasticsearch uses Apache Lucene behind the scenes to index and search data.
2. What are two different uses of the term “*index*”?
3. **True or False:** Every document stored in Elasticsearch belongs to an index.
4. What happens if you attempt to index a new document into an index that is not defined?
5. Can you **PUT** a document into an index without specifying an ID?
6. What is the default number of shards of an index?

Puneeth Kumar Bodigutta - 17Sep2017 - Cptum

Lab 1

Introduction to Elasticsearch

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open

Chapter 2

Installation and Configuration

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration**
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Configuration Areas
- Installation
- Elasticsearch Directories
- Configuring Names
- Network Configuration
- Development vs. Production Mode
- JVM Settings

Puneeth Kumar Bodigutta - 17-Sep-2017 / Optum



Configuration Areas

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Configuration Areas

- Settings are configured in Elasticsearch at three levels
 - index
 - node
 - cluster
- Let's take a look at each of these configuration areas

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Index Settings

- ***Index settings*** include:
 - number of shards, replicas, refresh rates, read-only, etc.
 - typically configured using the REST APIs

```
PUT my_tweets
{
  "settings": {
    "number_of_shards": 3
  }
}
```

when the index is created

```
PUT my_tweets/_settings
{
  "index.blocks.write": false
}
```

change settings of an
existing index

Node Settings

- ***Node settings*** include:
 - file paths, labels, network interfaces (settings specific to the node)
 - typically configured in **elasticsearch.yml** or command line
- Settings in **elasticsearch.yml** (as with any YML file) can be flat:

```
cluster.name: my_cluster  
node.name: node1
```

- or indented:

```
cluster:  
  name: my_cluster
```



Cluster Settings

- ***Cluster settings*** include:
 - logging levels, index templates, scripts, shard allocation, etc
 - typically configured using the REST APIs

```
PUT /_cluster/settings
{
  "transient" : {
    "logger.org.elasticsearch.discovery" : "DEBUG"
  }
}
```

Can be **persistent** (survives restarts) or
transient (will not survive a full cluster restart)



Precedence of Settings

1. *Transient settings* take precedence over all
 2. *Persistent settings* take precedence over:
 - 3. *Command-line settings*, which take precedence over:
 - 4. *elasticsearch.yml* settings
-
- Note that not all settings can be updated via the API
 - *static* settings can only be set in **elasticsearch.yml** or from the command line
 - *dynamic* settings can be updated on a live cluster using the **Cluster Update API**

Purnendu Kumar Bodigutta - 11 Sep 2017 / optimum



Example of Dynamic Settings

- Suppose you are having issues with a node joining a cluster
 - You can dynamically change the logging level for the transport module:

```
PUT _cluster/settings
{
  "transient" : {
    "logger.org.elasticsearch.transport.TransportService.tracer" : "TRACE"
  }
}
```

Change it back to “**INFO**” once you are done
debugging the issue

Installation

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Preparing for Installation

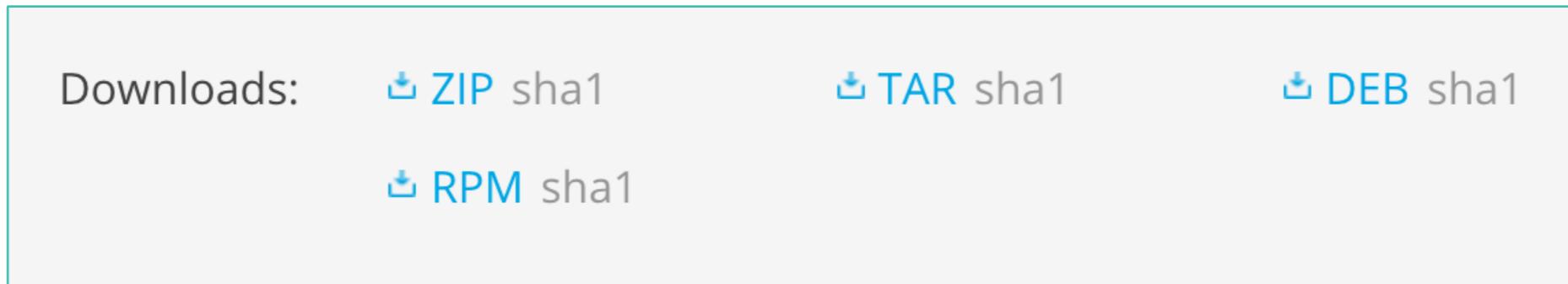
- Elasticsearch is a Java application
 - requires **JRE** (JavaSE runtime environment) or **JDK** (Java Development Kit)
 - JDK is preferred over JRE, due to tools such as **jstack** and **jps**, as well as better control on updating
- Elasticsearch 5.x requires at least Java 8
 - https://www.elastic.co/support/matrix#show_jvm
- Be sure to verify your environment is using the desired version
 - set **\$JAVA_HOME** appropriately

Puneeth Kumar Bodigutta - 17-Sep-2017 - ES6



Acquiring Elasticsearch

- Download the binaries at
<https://www.elastic.co/downloads/elasticsearch>



- Or build your own install from
<https://github.com/elastic/elasticsearch>
 - It is open source!

A screenshot of the Elasticsearch GitHub repository page. It shows the following details:

- Repository: elastic / elasticsearch
- Watchers: 1,888
- Stars: 20,494
- Forks: 7,066
- Code: 26,219 commits
- Issues: 1,072
- Pull requests: 112
- Projects: 1
- Pulse
- Graphs
- Settings
- Description: Open Source, Distributed, RESTful Search Engine <https://www.elastic.co/products/elast...>
- Apache-2.0 license

Installation Options

- Install options:
 - unzip the binaries (it really is that easy!)
 - MSI for Windows
 - Elastic Cloud Enterprise (ECE)
 - use Linux package manager ([.deb](#), [.rpm](#))
 - CM options like Ansible, Chef, Puppet
 - [Docker](#)
- For details on ECE:
 - <https://www.elastic.co/guide/en/cloud-enterprise/current/index.html>

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Elasticsearch Directories

Puneeth Kumar Bodigutta - 17-Sep-2021
Optum



Elasticsearch Directories

Folder	Description	Setting
bin	Binary scripts including elasticsearch to start a node and elasticsearch-plugin to install plugins	
config	Configuration files including elasticsearch.yml	path.conf
data	The location of the data files of each index and shard allocated on the node	path.data
lib	The Java JAR files of Elasticsearch	
logs	Elasticsearch log files location	path.logs
modules	Contains various Elasticsearch modules	
plugins	Plugin files location. Each plugin will be contained in a subdirectory	



The **path.data** Directory

- The **path.data** directory is where nodes store data
- Default is **\$ES_HOME/data**
 - except for RPM/DEB which defaults to **/var/lib/elasticsearch**
- You typically configure **path.data** to a different directory

```
path.data: /path/to/data
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Multiple Nodes on One Machine

- Only encouraged for dev/local development
 - two options for **path.data**: each node can have a separate **path.data**, or the nodes can share **path.data** directories
- Separate **path.data** directories:
 - preferred and creates the most isolation between nodes
 - allows easiest way to "pick up a node" and move it to its own machine
- Shared **path.data** directories
 - useful if you do not want to make separate directories on your local machine
 - requires setting **node.max_local_storage_nodes** on each node that is sharing a **path.data** directory (set it equal or greater than the number of nodes that are sharing the directory)

The path.logs Directory

- **path.logs** defaults to **\$ES_HOME/logs**
 - except for RPM/DEB: **/var/log/elasticsearch**

`path.logs: /path/to/logs/`

Log files are stored in **path.logs**, and log filenames are prefixed with **cluster.name**

Puneeth Kumar Bodigunta - 1-Sep-2017 - Optum



The path.conf Directory

- The location of the **config** directory can be changed with the **path.conf** setting
 - default is **\$ES_HOME/config**
 - for RPM/DEB : **/etc/elasticsearch**
- The config file is named **elasticsearch.yml**

```
# ----- Cluster -----  
#  
# Use a descriptive name for your cluster:  
#  
#cluster.name: my-application  
#  
# ----- Node -----  
#  
# Use a descriptive name for the node:  
#  
#node.name: node-1  
#  
# Add custom attributes to the node:  
#  
#node.attr.rack: r1  
...  
...
```

Notice by default all settings
are commented out



The `path.scripts` Directory

- Scripts in Elasticsearch are used by developers for searches, aggregations, modifying documents, etc.
 - **Painless** is the default scripting language (safest and does not require script files)
 - but you may want to execute scripts in other languages (Groovy, Python, etc) which are not safe in Elasticsearch
 - these script files need to be stored on every node in the cluster
- The `path.scripts` directory is where these file-based scripts must be saved
 - defaults to `$ES_HOME/config/scripts`
 - for RPM/DEB: `/etc/elasticsearch/scripts`

Purnendu Kumar Dasgupta - 17-Sep-2017



Configuring Names

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Cluster Name

- We have already seen that every cluster has a name
 - defaults to “elasticsearch”
 - change to something meaningful and unique (and something you like! It can not be changed)
 - helpful to include your company name (e.g. “Acme, Inc”)
- Set using **cluster.name**:

```
cluster.name : acme_production_logging
```

acme_production_logging

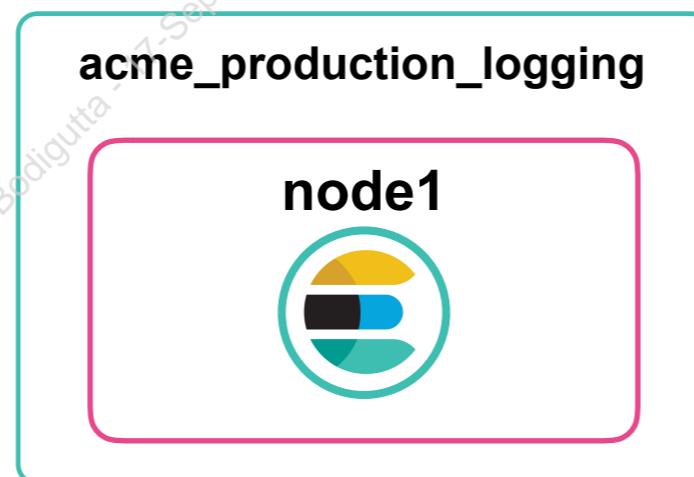


Node Name

- Every node should have a unique **node.name**
 - default is the first 7 characters of the node id (which is a UUID)
 - change to something meaningful
- Ideally use the hostname:

```
./bin/elasticsearch -E node.name=`hostname`
```

Puneeth Kumar Bodiguttula 21-Sep-2017

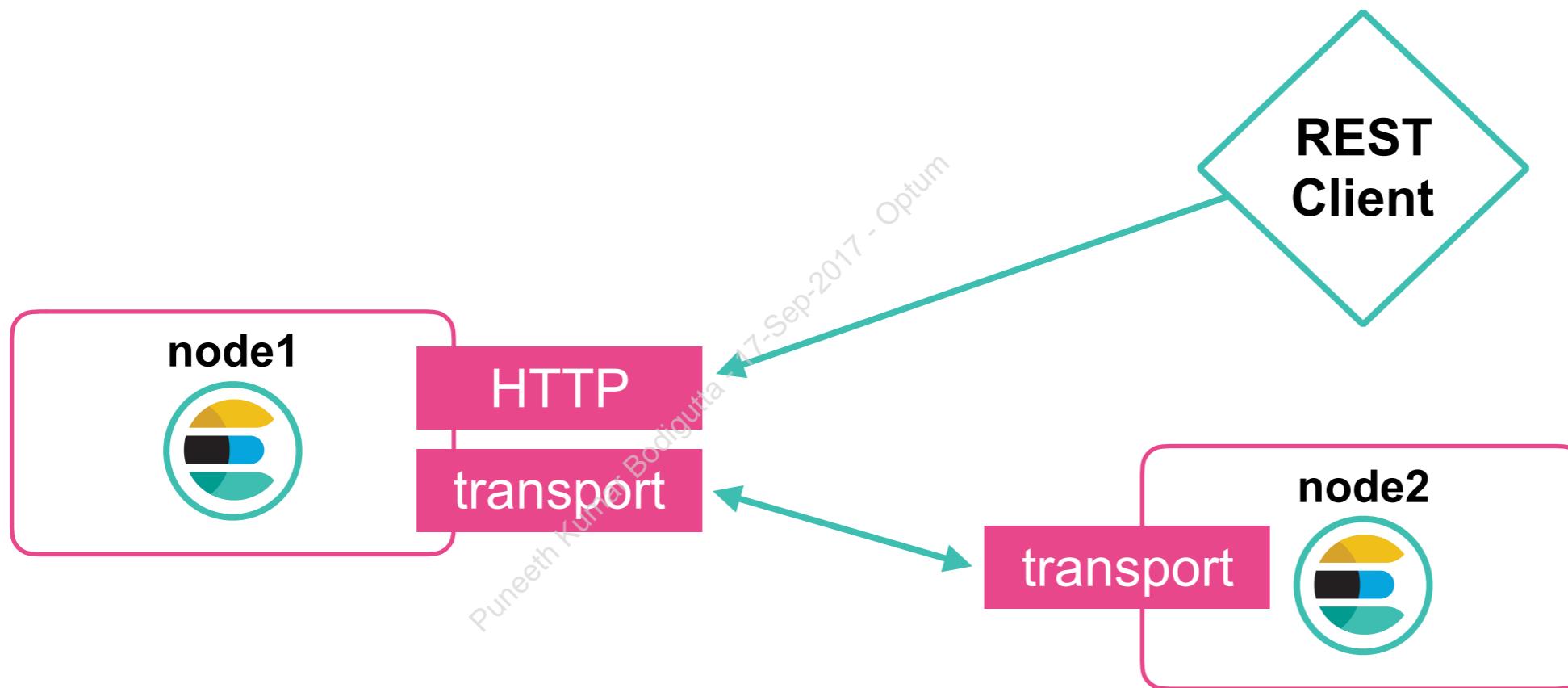


Network Configuration

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

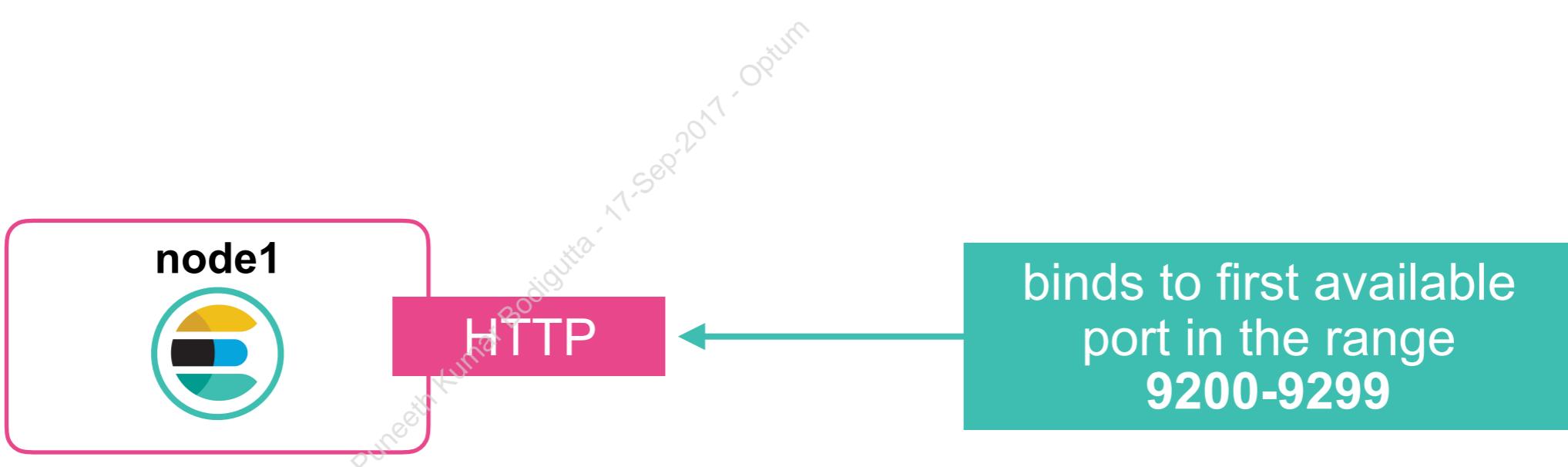
HTTP vs. Transport

- There are two important network communication mechanisms in Elasticsearch to understand:
 - **HTTP**: which is how the Elasticsearch REST APIs are exposed
 - **transport**: used for internal communication between nodes within the cluster



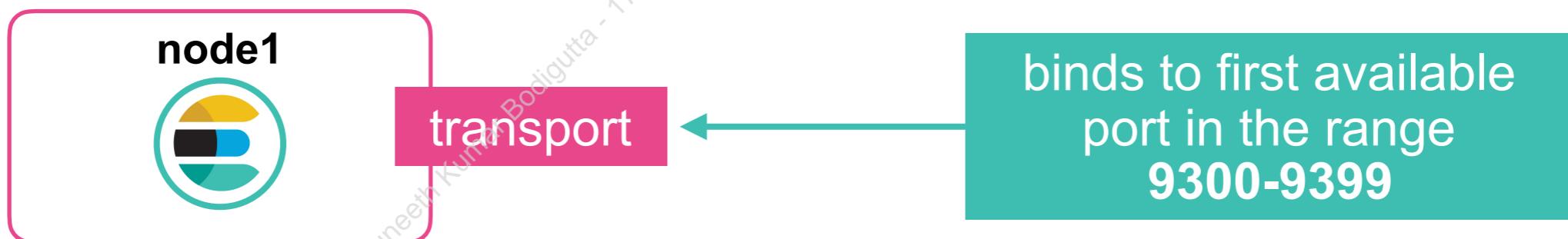
HTTP Communication

- The REST APIs of Elasticsearch are exposed over **HTTP**
- The HTTP module binds to **localhost** by default
 - configure with **http.host**
- Default port is the first available between **9200-9299**
 - configure with **http.port**



Transport Communication

- Each call that goes from one node to another uses the ***transport*** module
- Transport binds to **localhost** by default
 - configure with **transport.host**
- Default port is the first available between **9300-9399**
 - configure with **transport.tcp.port**



Configuring Network Settings

- Three ways to configure network settings:

network.*	specify settings for both protocols in one setting
transport.*	specify settings for the transport protocol
http.*	specify settings for the http protocol

- When configuring hosts, you can specify binding and publishing together, or separately:

*.host	specify both bind and publish in one setting
*.bind_host	interface to bind the specified protocol to
*.publish_host	interface used to advertise for other nodes to connect to



Special Values for network.host

- The following special values may be used for `network.host`:
 - and help to avoid hard-coding IP addresses in your config files

Value	Description
<code>_local_</code>	Any loopback addresses on the system (e.g. 127.0.0.1)
<code>_site_</code>	Any site-local addresses on the system (e.g. 192.168.0.1)
<code>_global_</code>	Any globally-scoped addresses on the system (e.g. 8.8.8.8)
<code>_[networkInterface]_</code>	Addresses of a network interface (e.g. _en0_)

Example Network Settings

- Example 1:

```
network.host: 192.168.1.2
```

Answer these questions for each example config:

- Example 2:

```
http.host: _local_
transport.host: 192.168.1.4
transport.publish_port: 5555
```

1. What are the **HTTP** and **transport** ports?
2. What are the **HTTP bind** and **publish hosts**?
3. What are the **transport bind** and **publish hosts**?

- Example 3:

```
http.enabled: false
transport.bind_host: ["10.42.0.17", "localhost"]
transport.tcp.port: 7777
transport.publish_host: _site_
transport.publish_port: 8888
```

Explanation from previous slide:

- Example 1:

```
network.host: 192.168.1.2
```

Both HTTP and transport bind and publish to 192.168.1.2 using default ports

- Example 2:

```
http.host: _local_
transport.host: 192.168.1.4
transport.publish_port: 5555
```

- HTTP bind and publish to loopback
- Transport bind and publish to 192.168.1.4
- Transport publishes on port 5555

- Example 3:

```
http.enabled: false
transport.bind_host: ["10.42.0.17", "localhost"]
transport.tcp.port: 7777
transport.publish_host: _site_
transport.publish_port: 8888
```

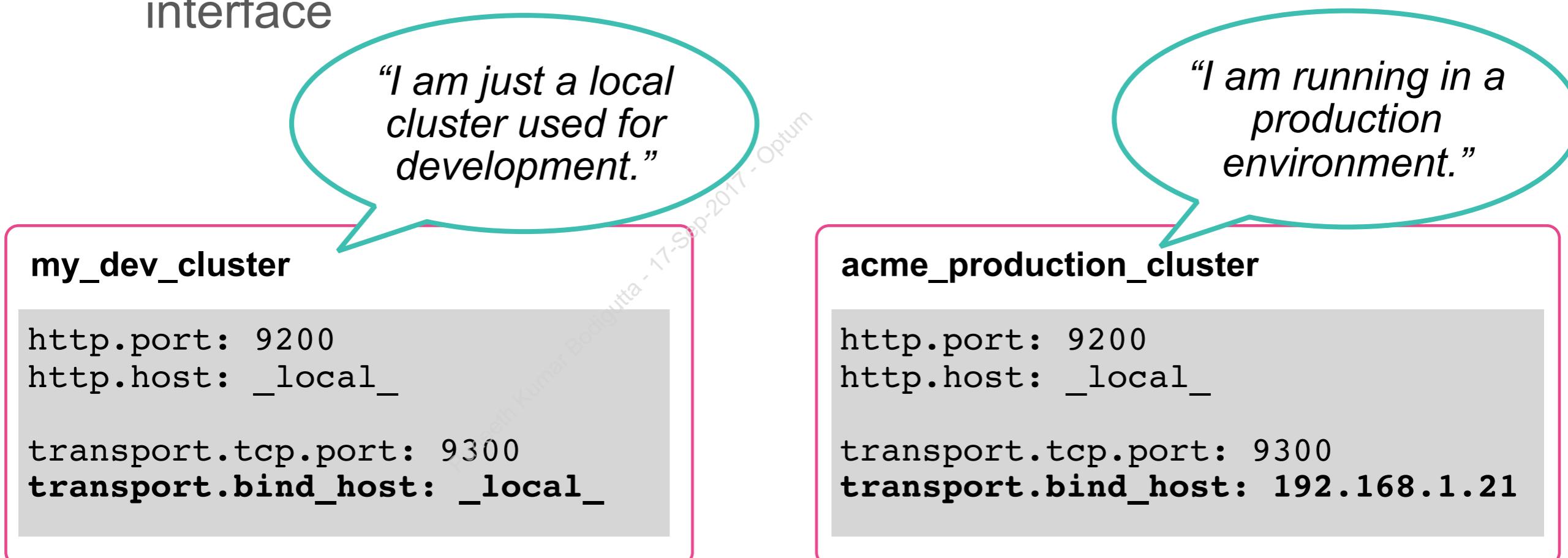
- No HTTP binding on any host or port
- Transport binds to two addresses on 7777
- Transport publishes to site addresses on port 8888

Development vs. Production Mode

Puneeth Kumar Bodigutta - 17-Sep-2021, Optum

Development vs. Production Mode

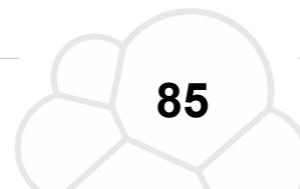
- As of Elasticsearch 5.0, a cluster is either in development mode or production mode:
 - development mode***: if it does *not* bind transport to an external interface (the default)
 - production mode***: if it does bind transport to an external interface



Explicitly Enforcing Bootstrap Checks

- If a node binds its transport to an external interface, then bootstrap checks are enforced
- You can enforce bootstrap checks *explicitly* by defining the following property
 - either in **config/jvm.options** file
 - or adding it to the **ES_JAVA_OPTS** environment variable

```
-Des.enforce.bootstrap.checks=true
```



Bootstrap Checks

- Elasticsearch has *bootstrap checks* upon startup:
 - inspect a variety of Elasticsearch and system settings
 - compare them to values that are safe for the operation of Elasticsearch

A node in production mode
must pass all checks in
order to startup

<input checked="" type="checkbox"/>	heap size check
<input checked="" type="checkbox"/>	memory lock check
<input checked="" type="checkbox"/>	maximum number of threads check
<input checked="" type="checkbox"/>	maximum map count check
<input checked="" type="checkbox"/>	serial collector check
<input type="checkbox"/>	file descriptor check
<input type="checkbox"/>	...

- View the current list of checks at
<https://www.elastic.co/guide/en/elasticsearch/reference/current/bootstrap-checks.html>



JVM Settings

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



JVM Configuration

- You can configure the Java Virtual Machine (JVM) two ways:

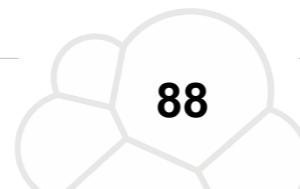
- the **config/jvm.options** file (preferred)

```
-Xms30g  
-Xmx30g
```

- setting the **ES_JAVA_OPTS** environment variable

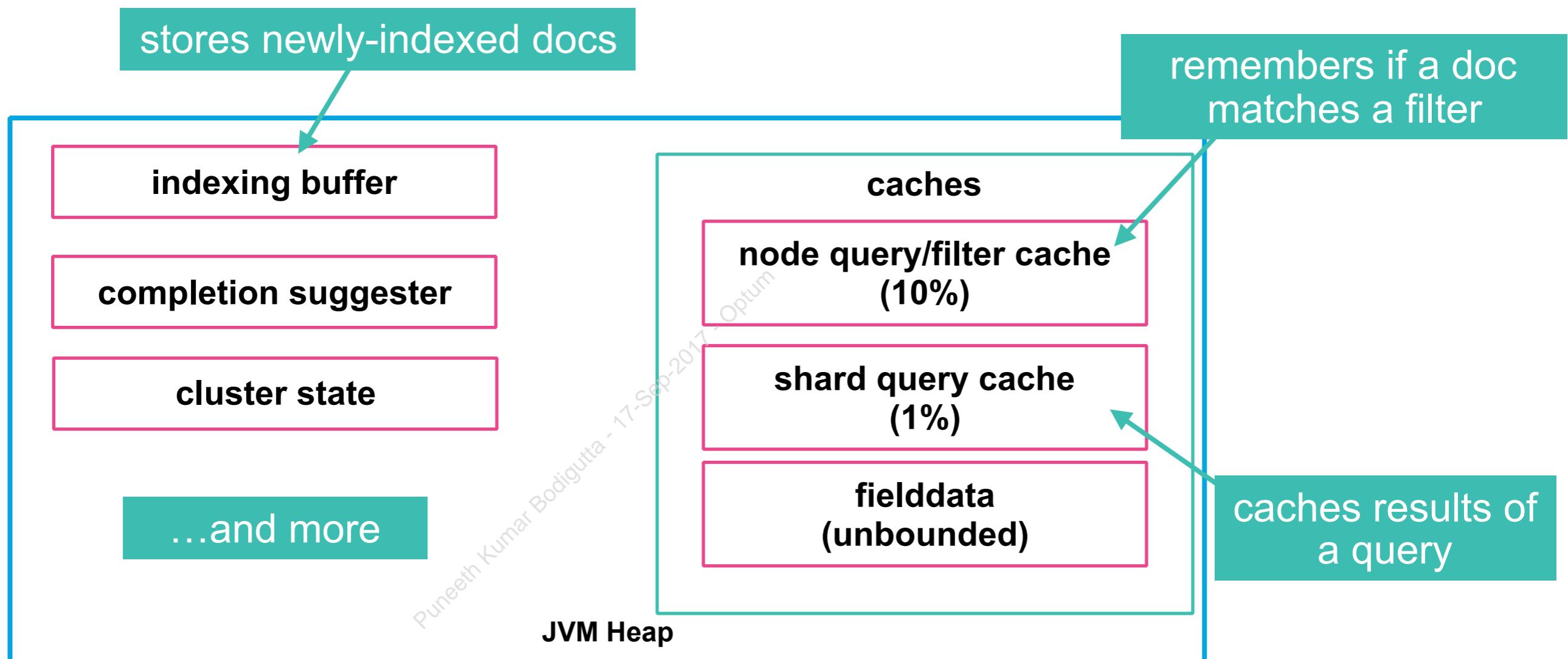
```
ES_JAVA_OPTS="-Xms30g -Xmx30g" bin/elasticsearch
```

- Elasticsearch has very good JVM defaults
 - avoid and be careful when changing them



What Goes on the Heap?

- Some of the major usage of the heap by Elasticsearch includes:



JVM Heap Size

- By default, the JVM heap size is 2 GB
 - likely not high enough for production
 - you can change it using **Xms** (min heap) and **Xmx** (max heap)

```
ES_JAVA_OPTS="-Xms8g -Xmx8g" ./bin/elasticsearch
```

- Some guidelines for configuring the heap size:
 - set **Xms** and **Xmx** to the same size (bootstrap check)
 - set **Xmx** to no more than 50% of your physical RAM
- Rule of thumb for setting the JVM heap is:
 - do not exceed more than 30GB of memory (to not exceed the compressed ordinary object pointers limit)
 - <https://www.elastic.co/blog/a-heap-of-trouble>

Production JVM Settings

1. We do not support G1GC garbage collection
 - and strongly discourage its use
2. JDKs have two modes of a JVM: *client* and *server*
 - *server* JVM is required in production mode
3. Configure the JVM to **disable swapping**
 - by requesting the JVM to lock the heap in memory through **mlockall** (Unix) or **virtual lock** (Windows)

Puneeth Kumar Bodigutta - 17-Sep-2017 Open

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- Installing Elasticsearch can be as easy as unzipping a file!
- Production deployments will typically define custom folders for **path.data**, **path.logs**, **path.scripts** and **path.conf**.
- Elasticsearch uses two network communication mechanisms: **HTTP** for REST clients; and **transport** for inter-node communication
- Every Elasticsearch 5.x instance is either in ***development mode*** or ***production mode***
- Some cluster settings in Elasticsearch are dynamic and can be modified using the Update Cluster API
- Set **Xms** and **Xmx** to the same size, and typically to no more than 50% of your physical RAM

Quiz

1. Which setting would you use to modify the directory that a node uses to store its data?
2. **True or False:** Every node in a cluster needs to enable the HTTP protocol in order to communicate with other nodes.
3. What is the effect of the following configuration:

```
network.host : _global_
```

4. **True or False:** *Transient* settings take precedence over *persistent* settings.
5. **True or False:** In production mode, a node will not start unless **Xms** and **Xmx** are set to the same value.

Puneeth Kanchanadigutta - 17-Sep-2017, Optum



Lab 2

Installation and Configuration

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open

Chapter 3

Working with Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- The Discovery Module
- Cluster State
- Node Types
- Master Eligible Nodes
- Avoiding Split Brain
- Data Nodes
- Coordinating Only Nodes
- Ingest Nodes
- Tribe Nodes
- Sample Architectures

Puneeth Kumar Bodigutta - 17 Sep-2017 - Optum

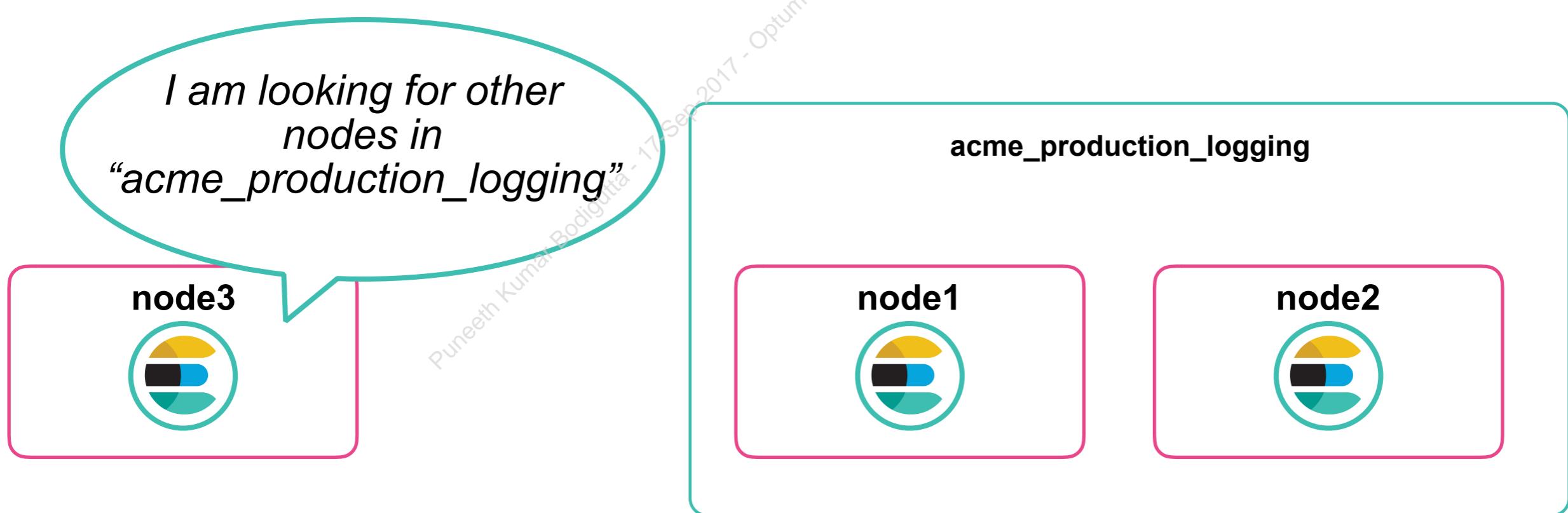


The Discovery Module

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

The Discovery Module

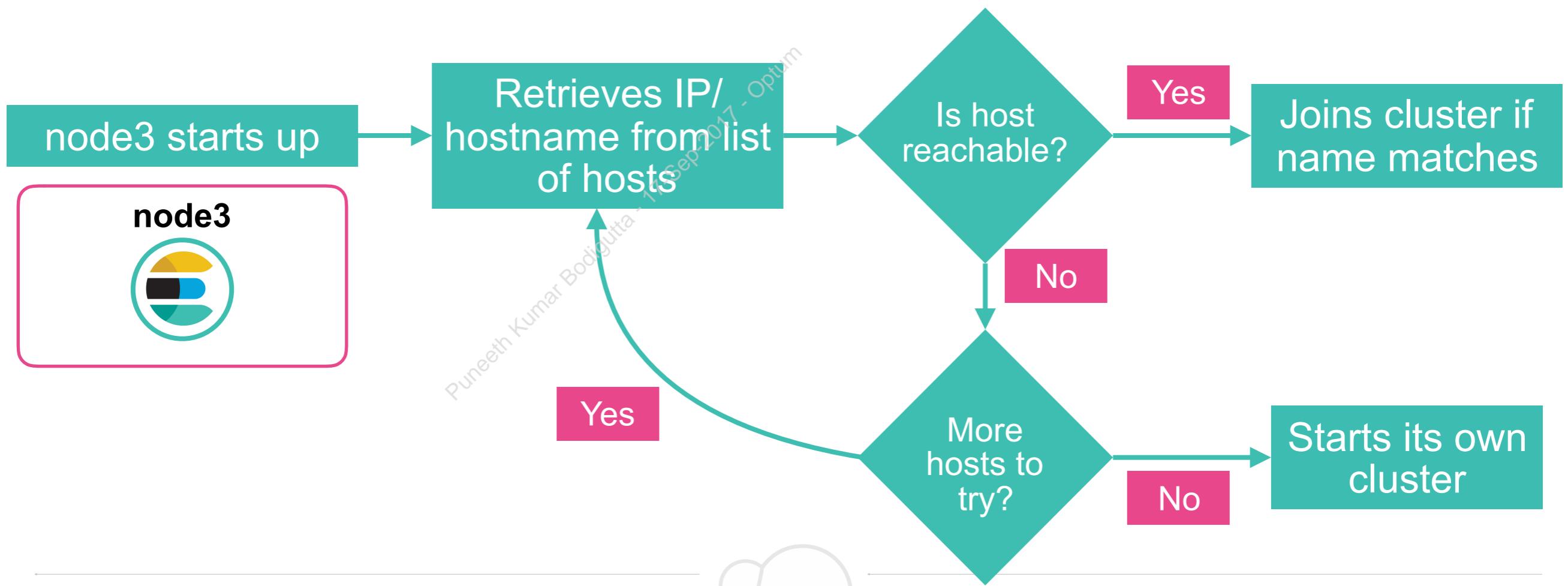
- The *discovery module* is responsible for discovering nodes within a cluster
 - there are different techniques to do this
- The default is *zen discovery*
 - but there are also discovery modules for EC2, GCE and Azure



Zen Discovery

- With ***zen discovery***, a node issues ping requests to find other nodes on the same network
 - start with a list of hosts to act as *gossip routers*, configured with **discovery.zen.ping.unicast.hosts**

```
discovery.zen.ping.unicast.hosts : [ "node1", "node2" ]
```



Zen Configuration

- Unicast discovery requires a list of hosts to use that will act as gossip routers
 - the **discovery.zen.ping.unicast.hosts** property

```
discovery.zen.ping.unicast.hosts : [ "host", "host:port" ]
```

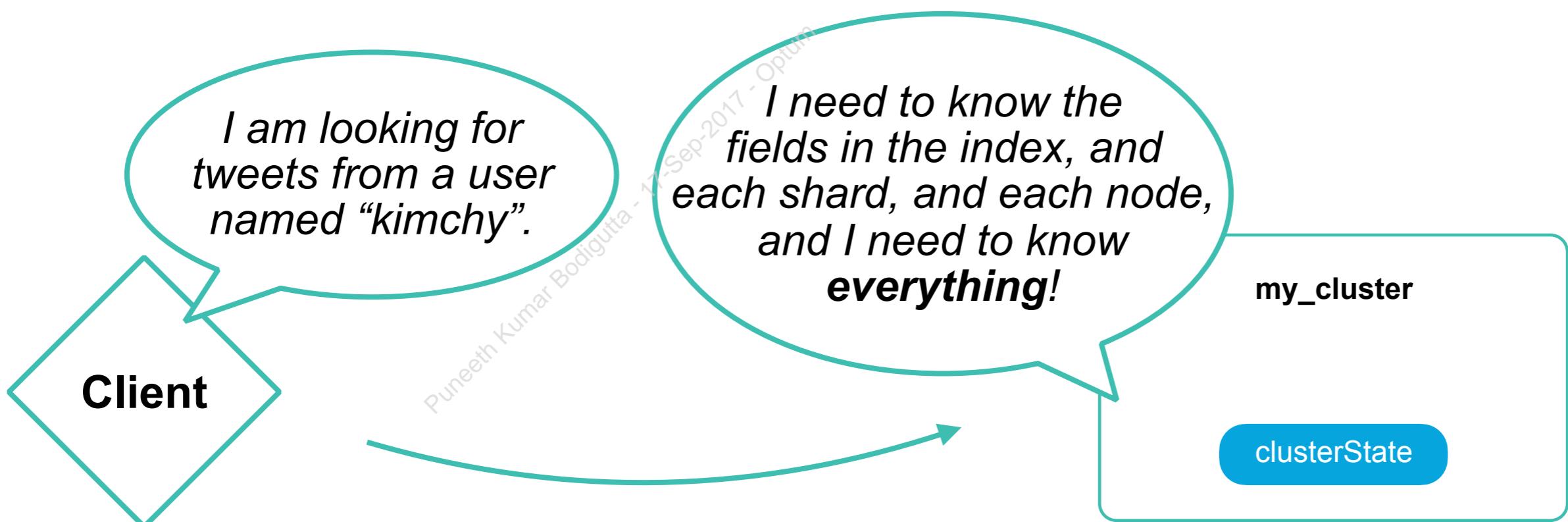
- Which nodes should be included in this list?
 - You do ***not*** need to list all the nodes in the cluster
 - A new node only needs to find one of the nodes on the list
- Dedicated master nodes are a good choice to include
 - (we will discuss those shortly)

Cluster State

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

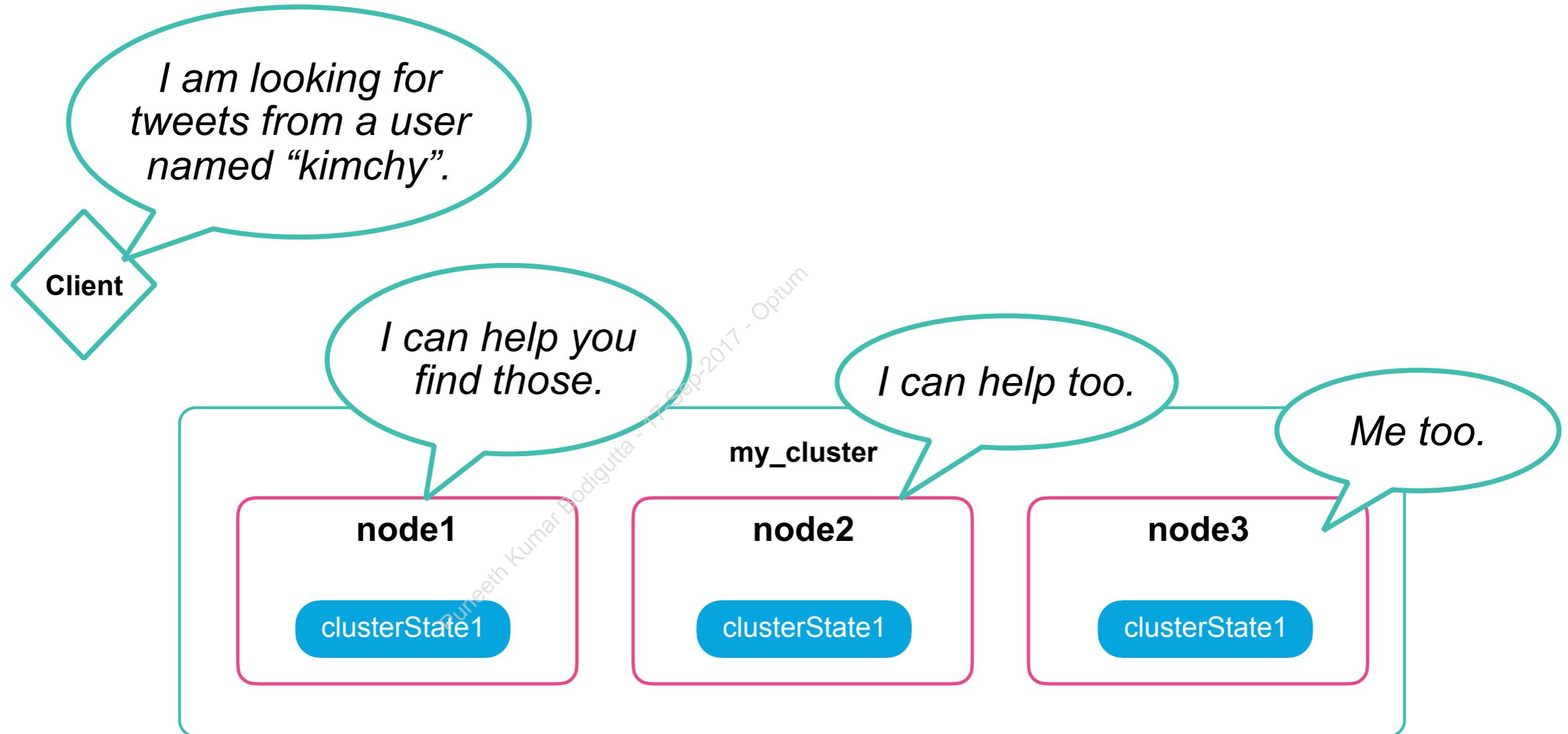
Cluster State

- A cluster has a lot of **state** that needs to be maintained:
 - the details of all the indices (mappings, settings, etc.)
 - the nodes in the cluster
 - which shards exist on which nodes
 - the state of each shard



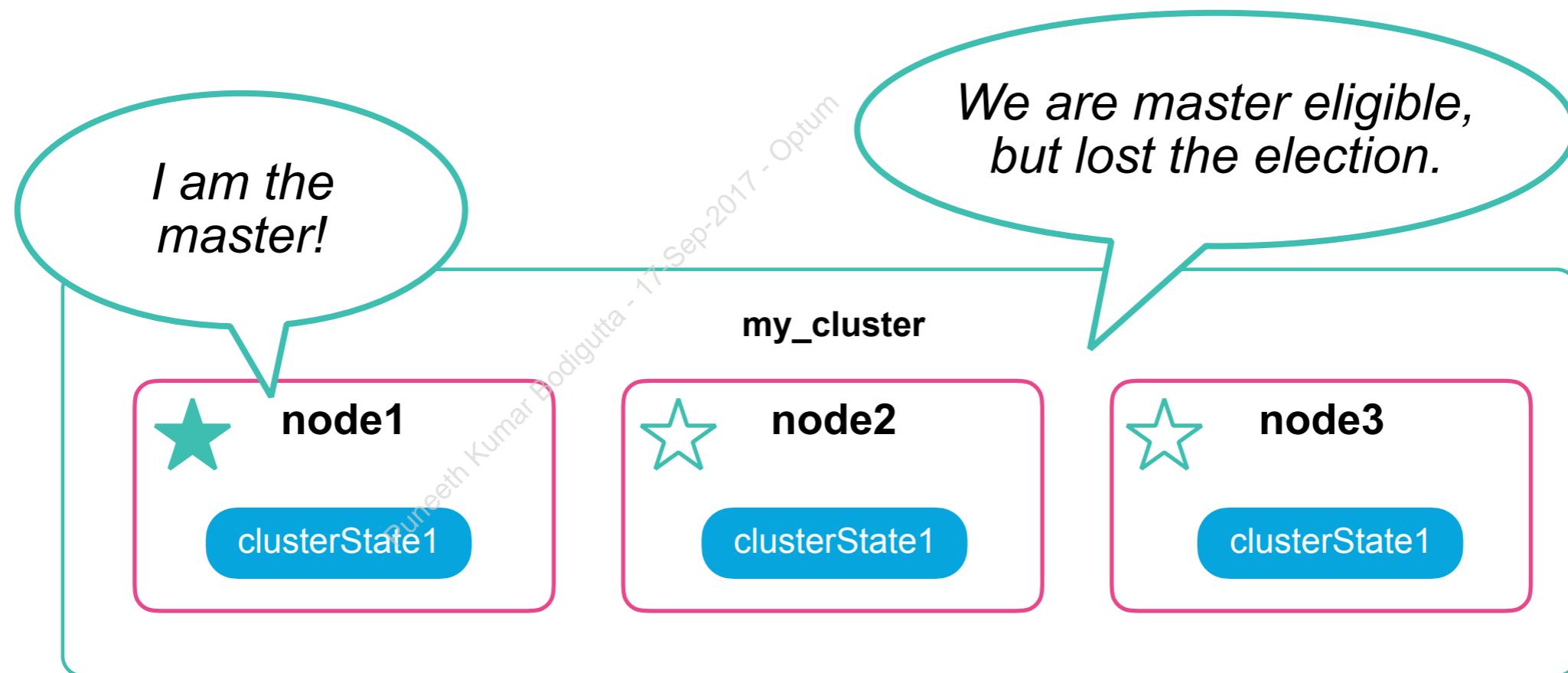
Every node needs the cluster state...

- Clients can send requests to any node in the cluster
 - therefore, every node must have the cluster state



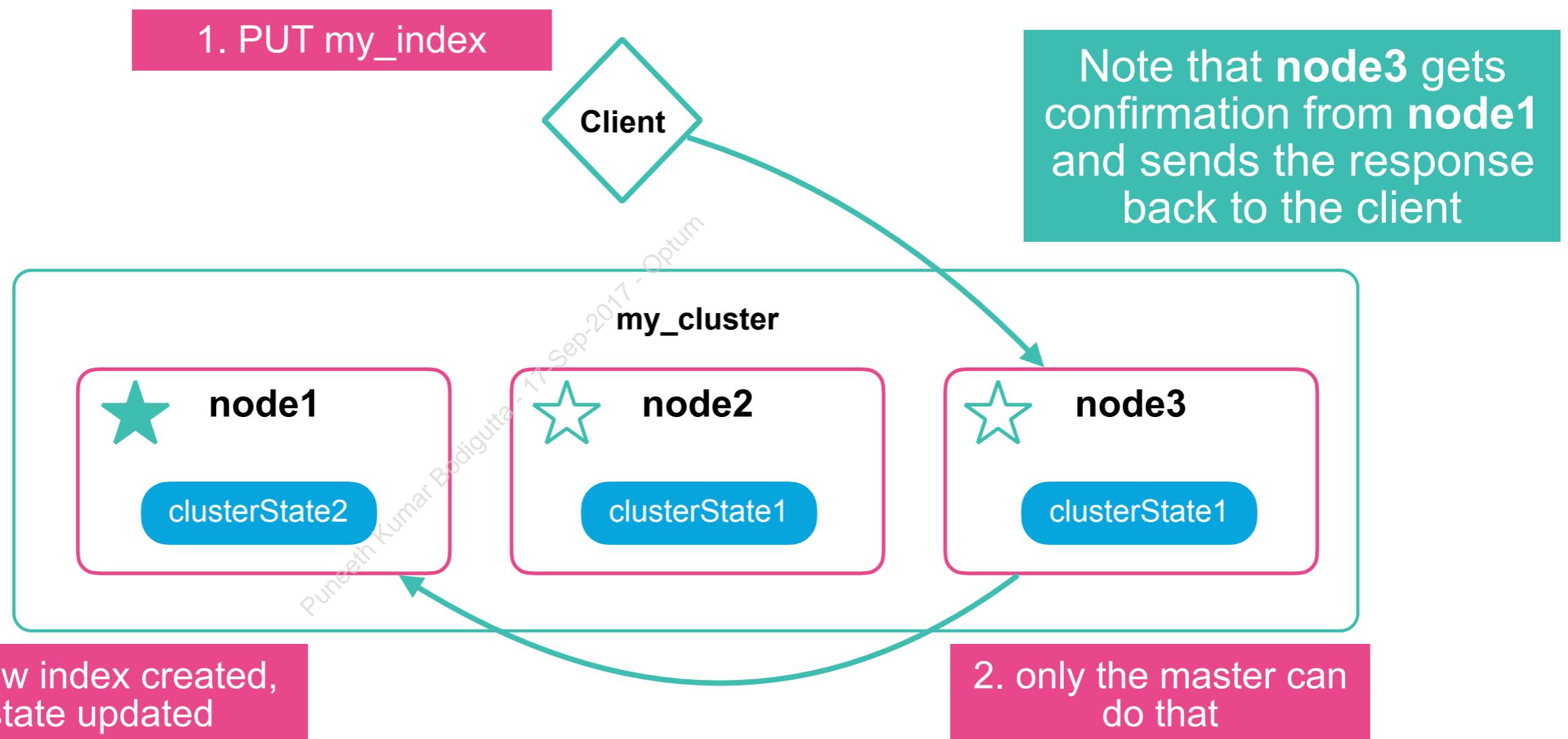
...but only one node maintains it

- Every cluster has a single node called the ***master node*** that controls the cluster
 - chosen by election from a group of ***master eligible nodes*** ★
 - the only node that can change the cluster state
 - sends state changes/diffs to the rest of the nodes in the cluster



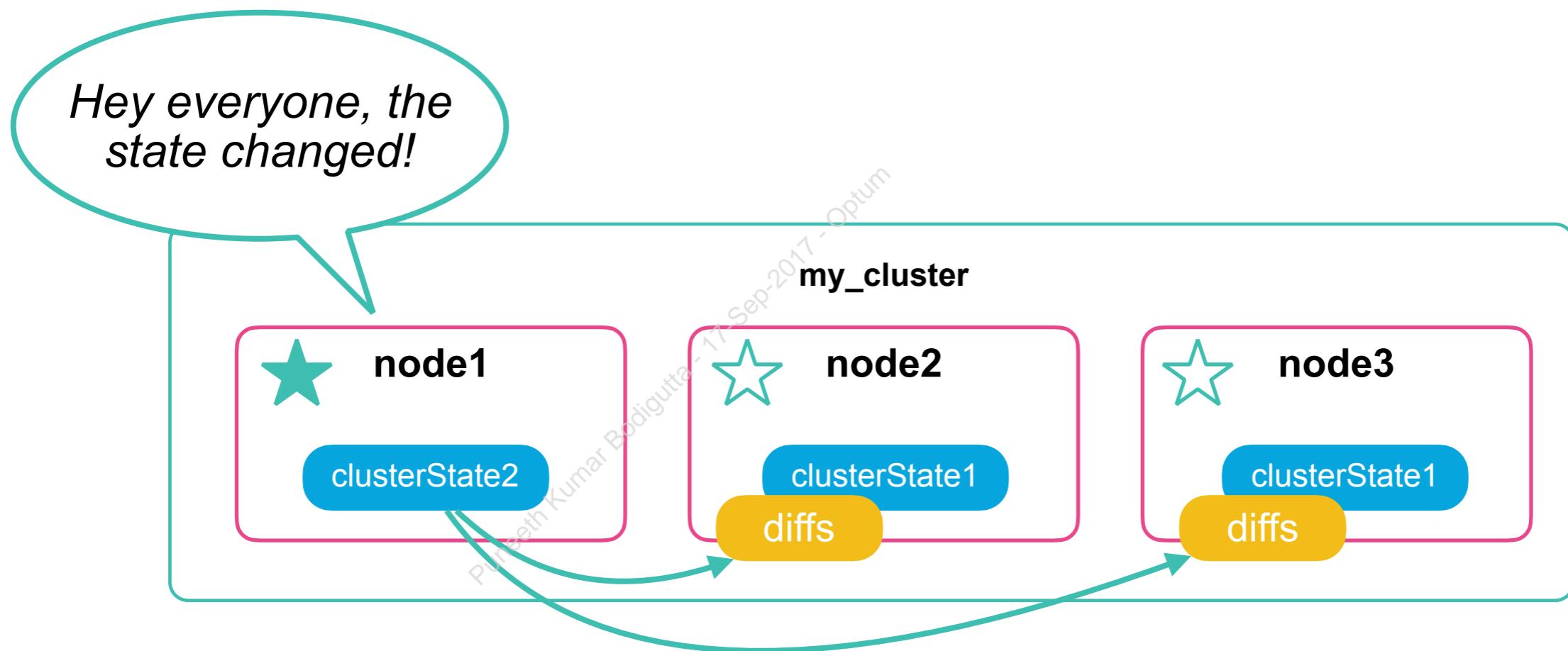
Changes to the Cluster State

- Only the master can change the state
 - any API requests that require state change are internally forwarded to the master



Updates are Published to All Nodes

- Once a change is made to the cluster state, the master pushes the diffs (on a per index basis) to all other nodes



The Cluster State API

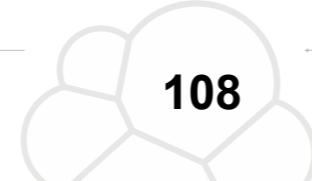
- You can **GET** the cluster state using the `_cluster` endpoint

`GET _cluster/state`



```
{  
  "cluster_name": "my_cluster",  
  "version": 14,  
  "state_uuid": "F_fhCTZWTheaFsQuIn_dng",  
  "master_node": "wYk1_Mh8Sy63NUazeg6yiw",  
  "blocks": {},  
  "nodes": {  
    ...  
  },  
  "metadata": {  
    "cluster_uuid": "abBfuaS7Rq05eKw07eFqxw",  
    "templates": {},  
    "indices": {  
      "my_index": {  
        "state": "open",  
        "settings": {  
          ...  
        }  
      },  
      "mappings": {},  
      "aliases": [],  
      ...  
    },  
    "routing_table": {  
      ...  
    },  
    "routing_nodes": {  
      ...  
    }  
  }  
}
```

Puneeth Kumar Bodigutta - 17-Sep-2017



Node Types

Puneeth Kumar Bodigutta - 17-Sep-2017 , Optum

Node Roles

- There are several roles a node can have:
 - Master eligible
 - Data
 - Ingest
 - Coordinating Only
 - Machine Learning
 - Tribe (deprecated)
- Nodes can take on multiple roles at the same time
 - or they can be *dedicated* nodes that only take on a single role

Puneeth Marur Bodige 2017-Sep-2017 - Optum

Configuring Node Types

- By default, a node is a master eligible, data, and ingest node:

Node type	Configuration parameter	Default value
master eligible	node.master	true
data	node.data	true
ingest	node.ingest	true
coordinating only	n/a	set node.master , node.data and node.ingest to false
machine learning	node.ml and xpack.ml.enabled	true (if X-Pack is enabled)

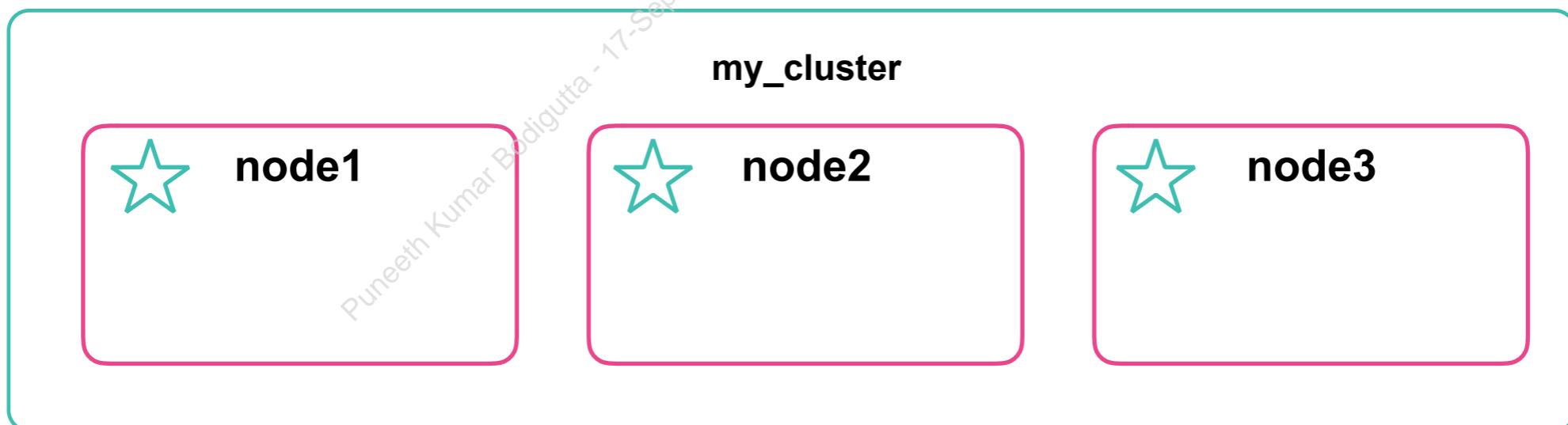
- *Tribe nodes* are configured differently (see this later)

Master Eligible Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

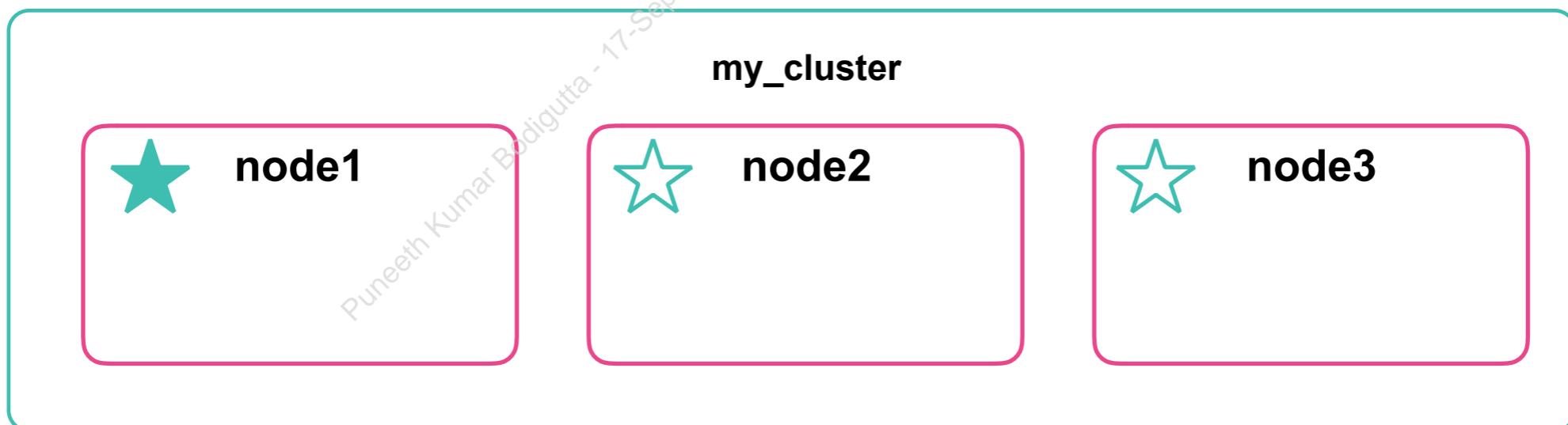
The Master Node

- The *master node* is responsible for:
 - cluster-wide actions such as creating or deleting an index
 - tracking which nodes are part of the cluster
 - deciding which shards to allocate to which nodes
- Critical to have a stable master node for cluster health
 - preferred to have multiple, dedicated master eligible nodes



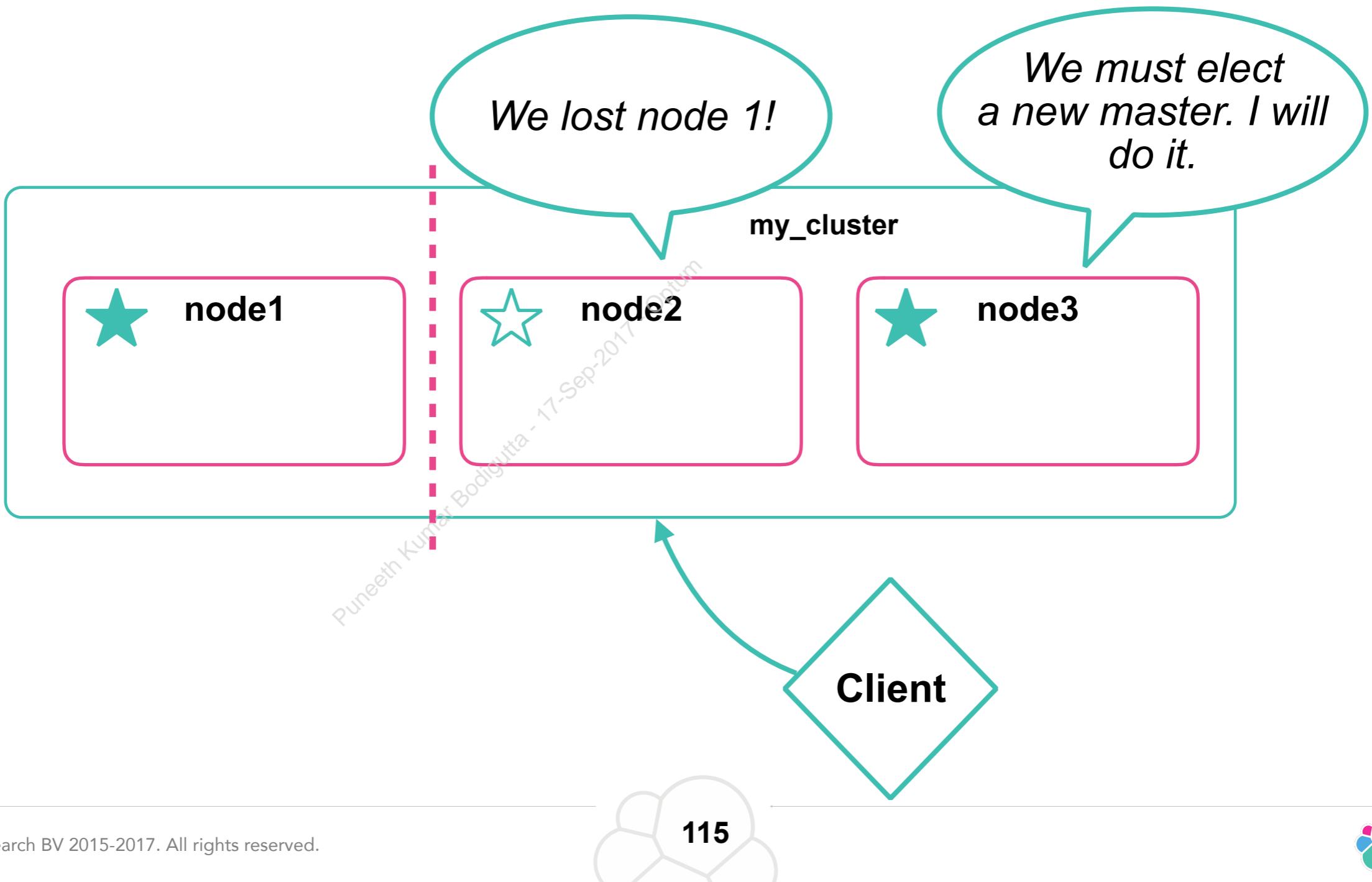
The Election Process

- During discovery, an election for the master occurs
 - as part of the ping process
 - if a decision needs to be made, the node with the “lower” node id will “win”
- Once a master is elected
 - subsequent master eligible nodes will still join the cluster, but not as the master



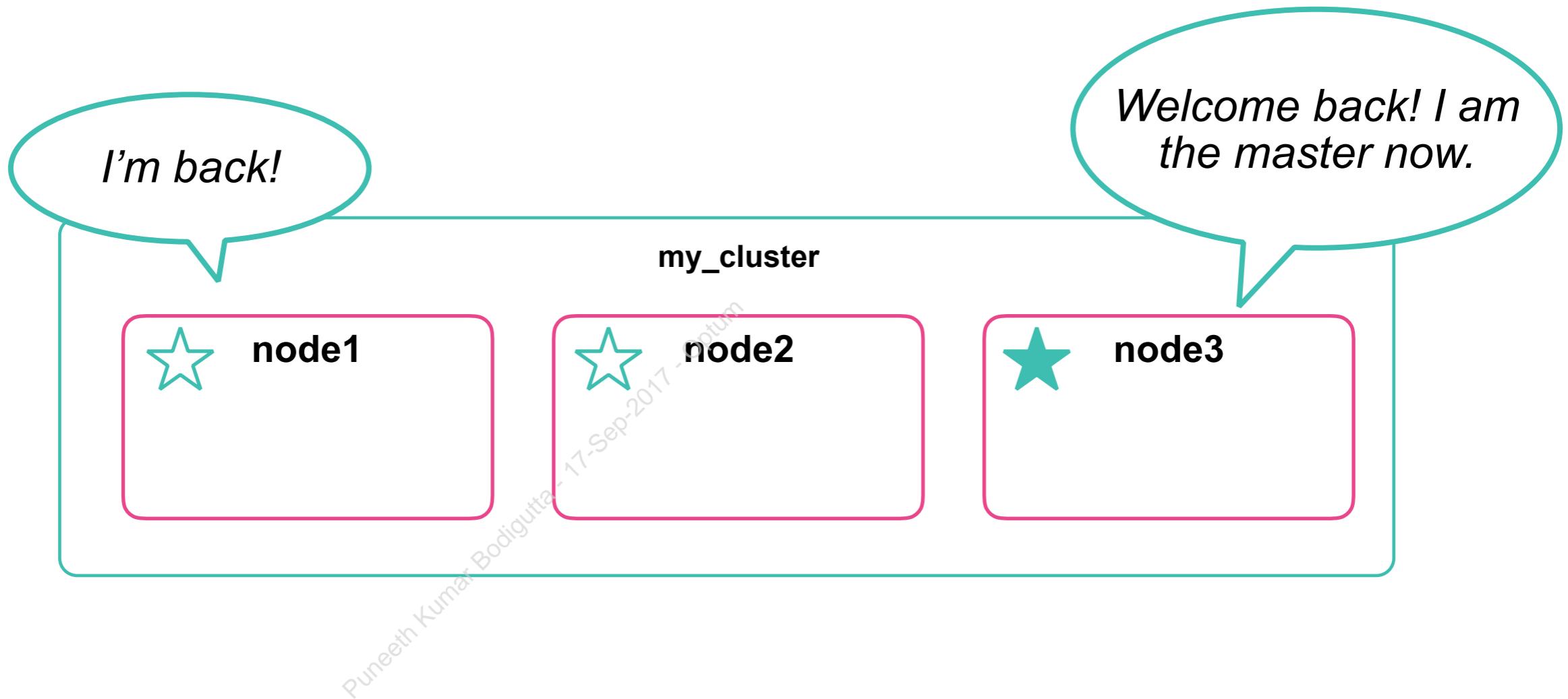
Failover of a Master

- Multiple master eligible nodes provide cluster redundancy
 - if a master fails, a new master gets elected



Recovery of a Failed Node

- When **node1** comes back online, **node3** will remain the master

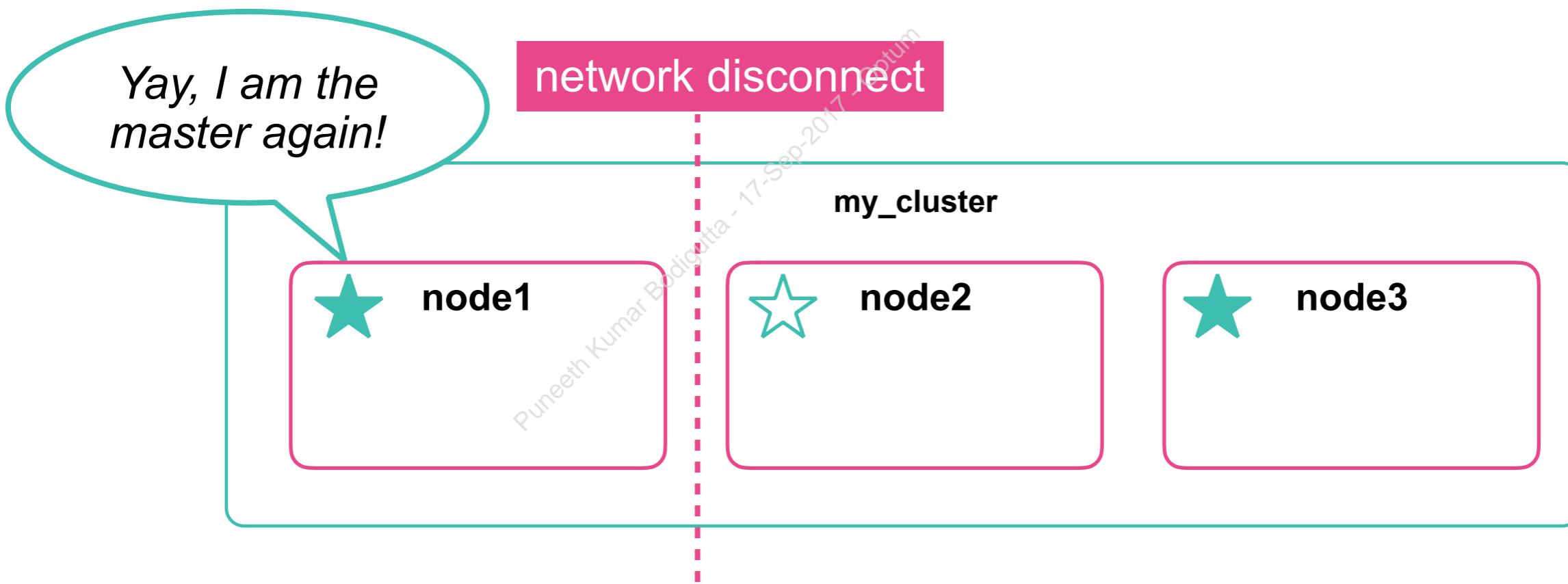


Avoiding Split Brain

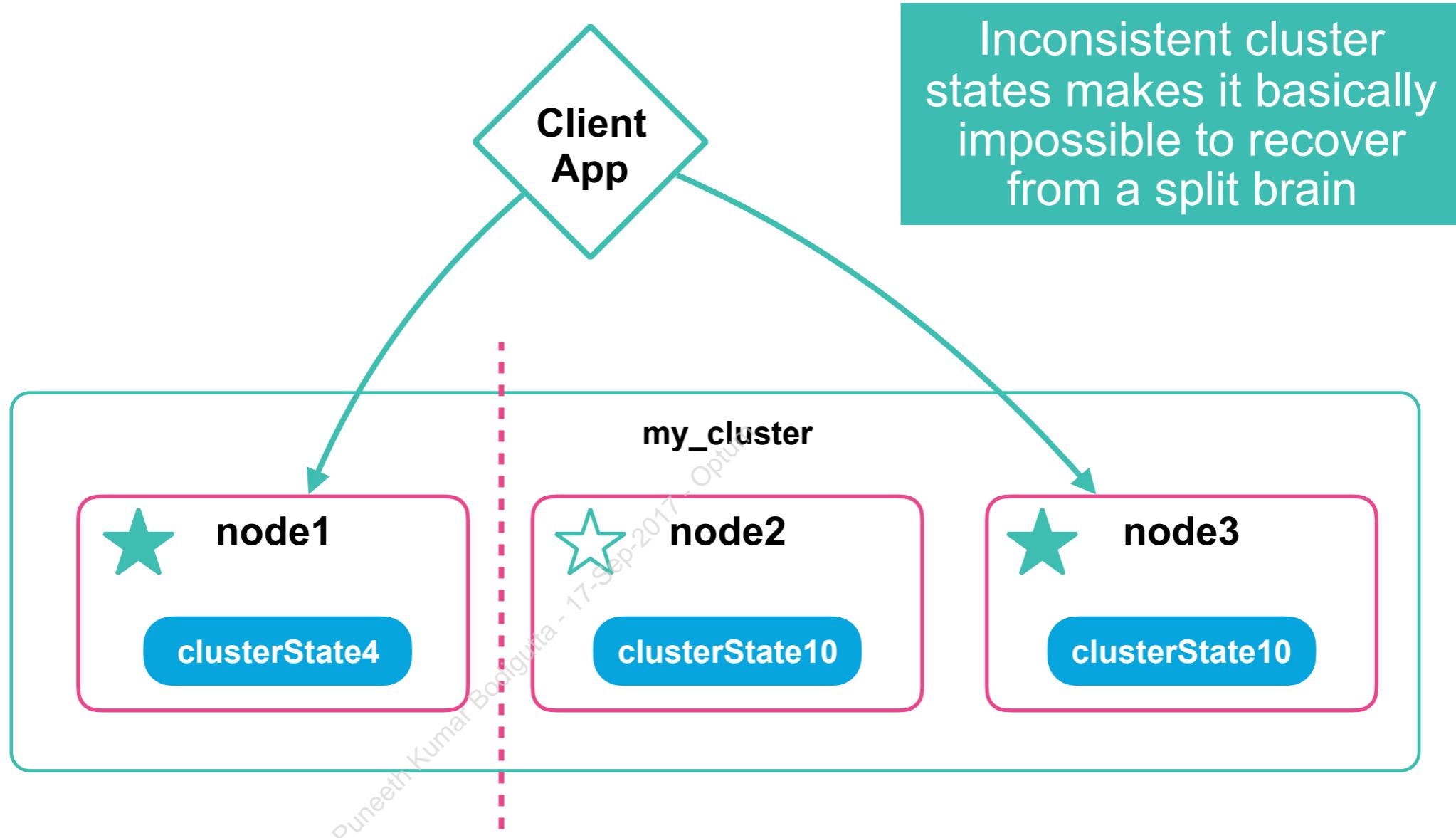
Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Split Brain

- Even though the master node is the single source of truth when it comes to the cluster state, problems can still happen
 - In particular, a big concern is if the cluster gets **partitioned**
 - The cluster could inadvertently elect two masters, which is referred to as a “**split brain**”

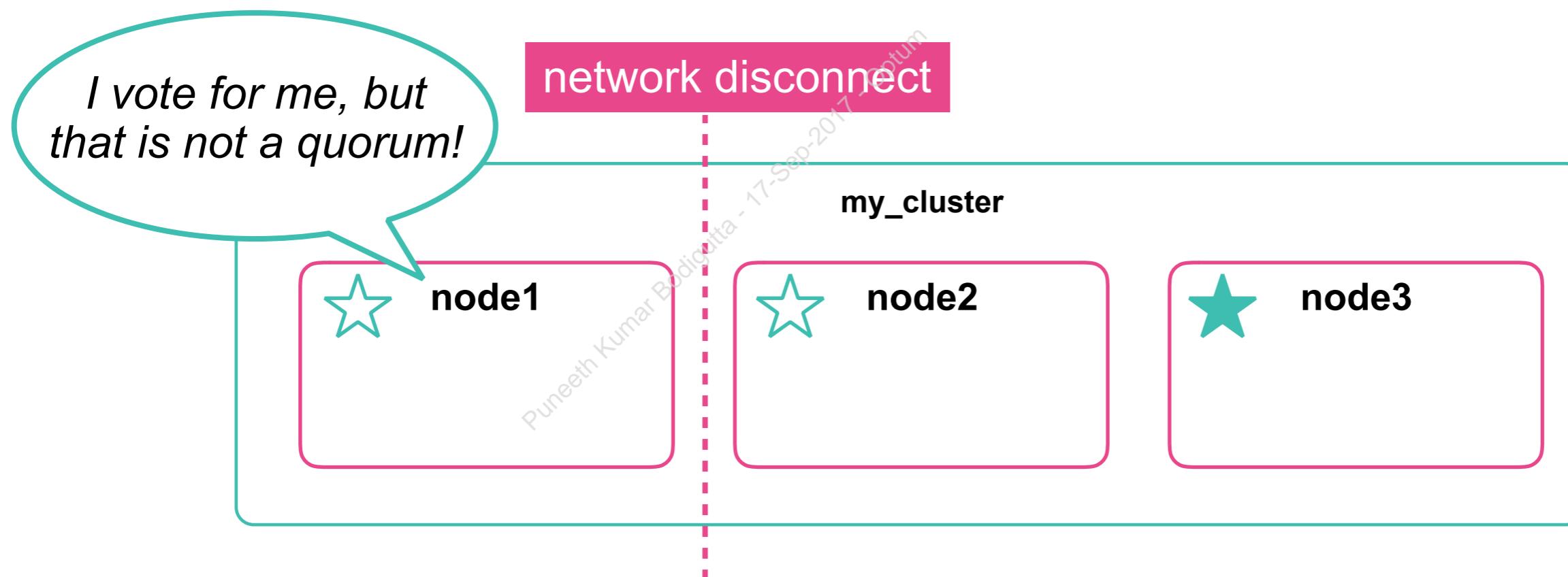


Two masters can lead to inconsistency:



Avoiding Split Brain

- A master-eligible node needs at least **minimum_master_nodes** votes to win an election
 - Setting it to a quorum helps avoid the split brain scenario
- Our simple recommendation is for production systems to always have 3 dedicated master eligible nodes
 - and set **minimum_master_nodes** to 2

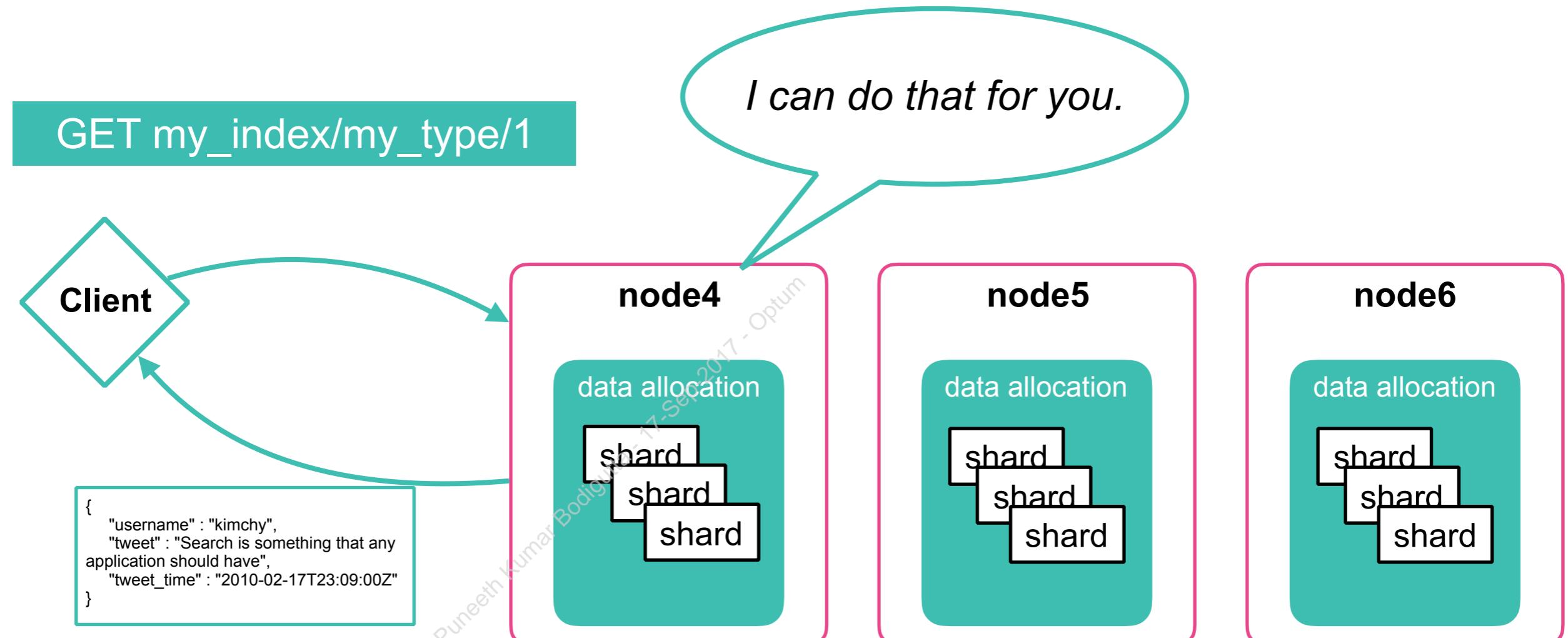


Data Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

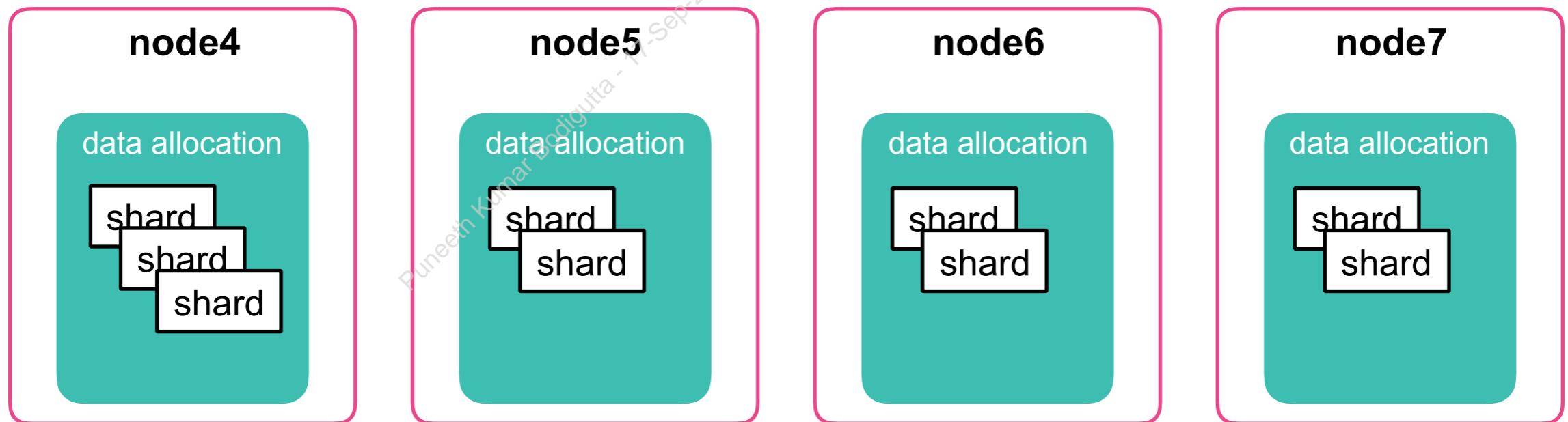
Data Nodes

- **Data nodes** are allowed to hold shards and also host the Elasticsearch REST API



Data Nodes Scale

- When we say Elasticsearch is scalable, data nodes play the key role:
 - if you need more space, add more data nodes
 - if you need more computing power, add more data nodes
- Master node will determine how to redistribute the shards

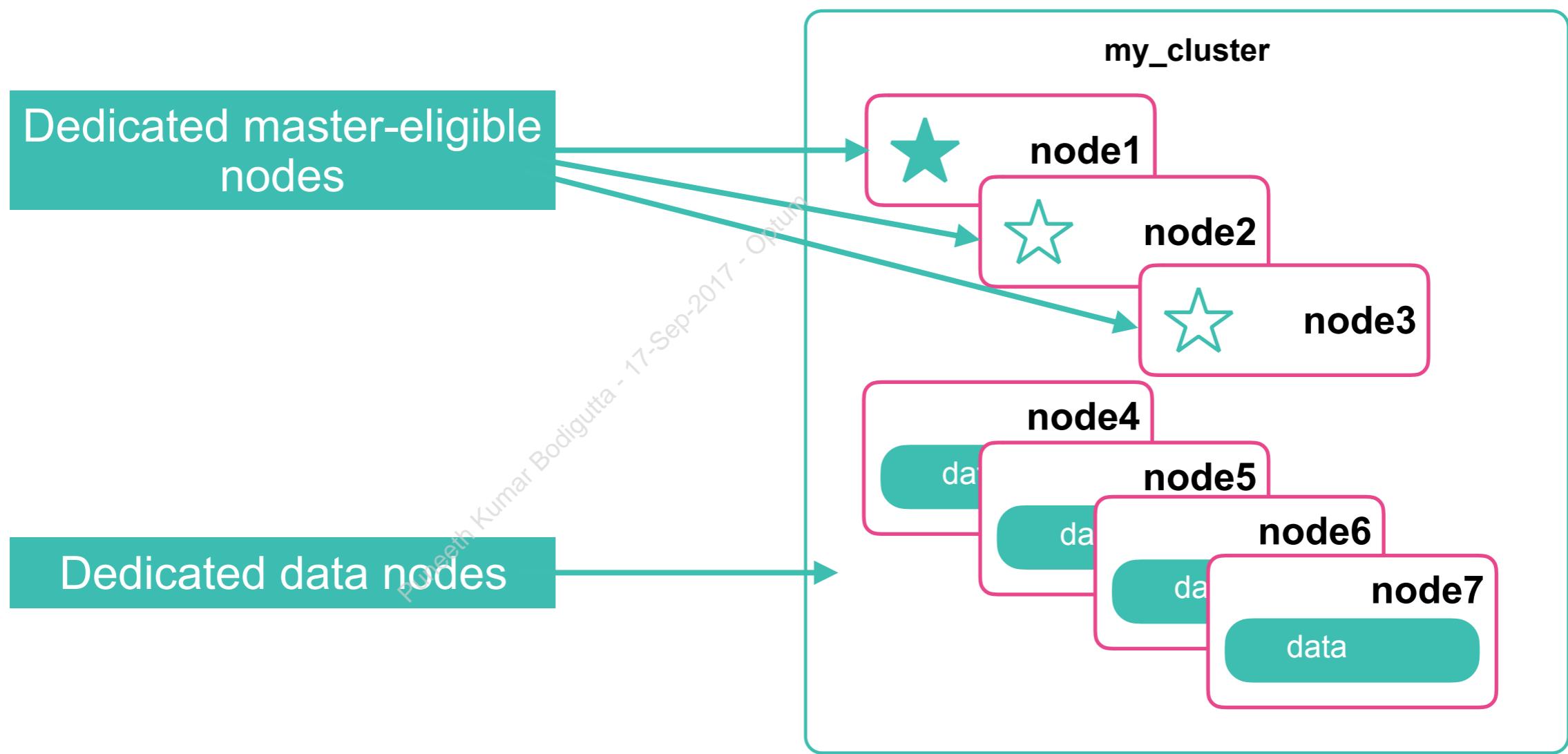


Dedicated Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 - Open

Dedicated Master and Data Nodes

- You can configure a node to be just a data or master eligible node
 - provides a nice separation of the master and data roles
 - do *not* send client requests to dedicated master nodes



Why Use Dedicated Nodes?

- *Dedicated master nodes* are less likely to have JVM heap and memory issues:
 - they are not holding any shards (data)
 - they are not handling all the API connections/requests
 - they are not involved with high-frequency index and search operations, so they are not a scaling bottleneck
- *Dedicated data nodes* can be configured differently:
 - to focus on data storage and processing client requests

Puneeth Kumar Bodigutta - 17.Sep.2017 - Open
Source

Configuring Dedicated Nodes

- For a dedicated master eligible node:

```
node.master: true  
node.data: false  
node.ingest: false
```

- For a dedicated data node:

```
node.master: false  
node.data: true  
node.ingest: false
```

Puneeth Kumar Bodigutta - 17-Nov-2017 - Optum

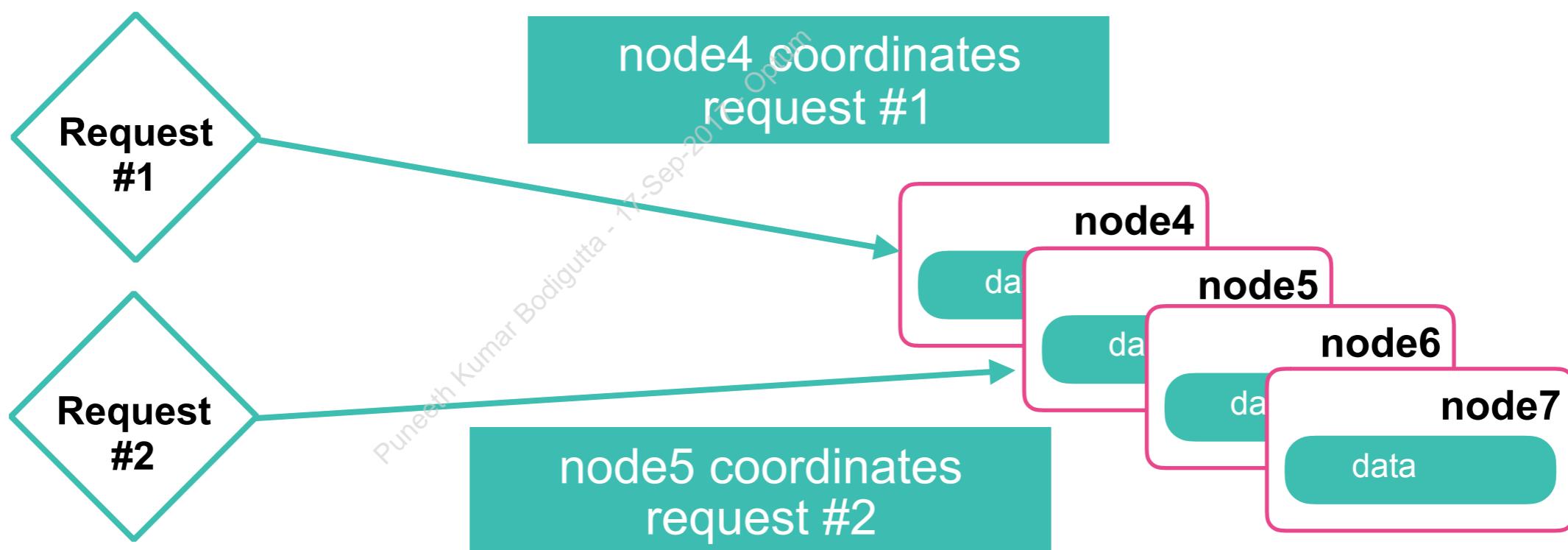


Coordinating Only Nodes

Puneeth Kumar Bodigutta - 17-Sep-2011, Optum

Coordinating Node

- A *coordinating node* is the node that receives a specific client request
 - every node is implicitly a coordinating node
 - forwards the request to other relevant nodes, then combines those results into a single result



Coordinating Only Node

- A *coordinating only node* is specifically configured to *not* be a master, data or ingest node

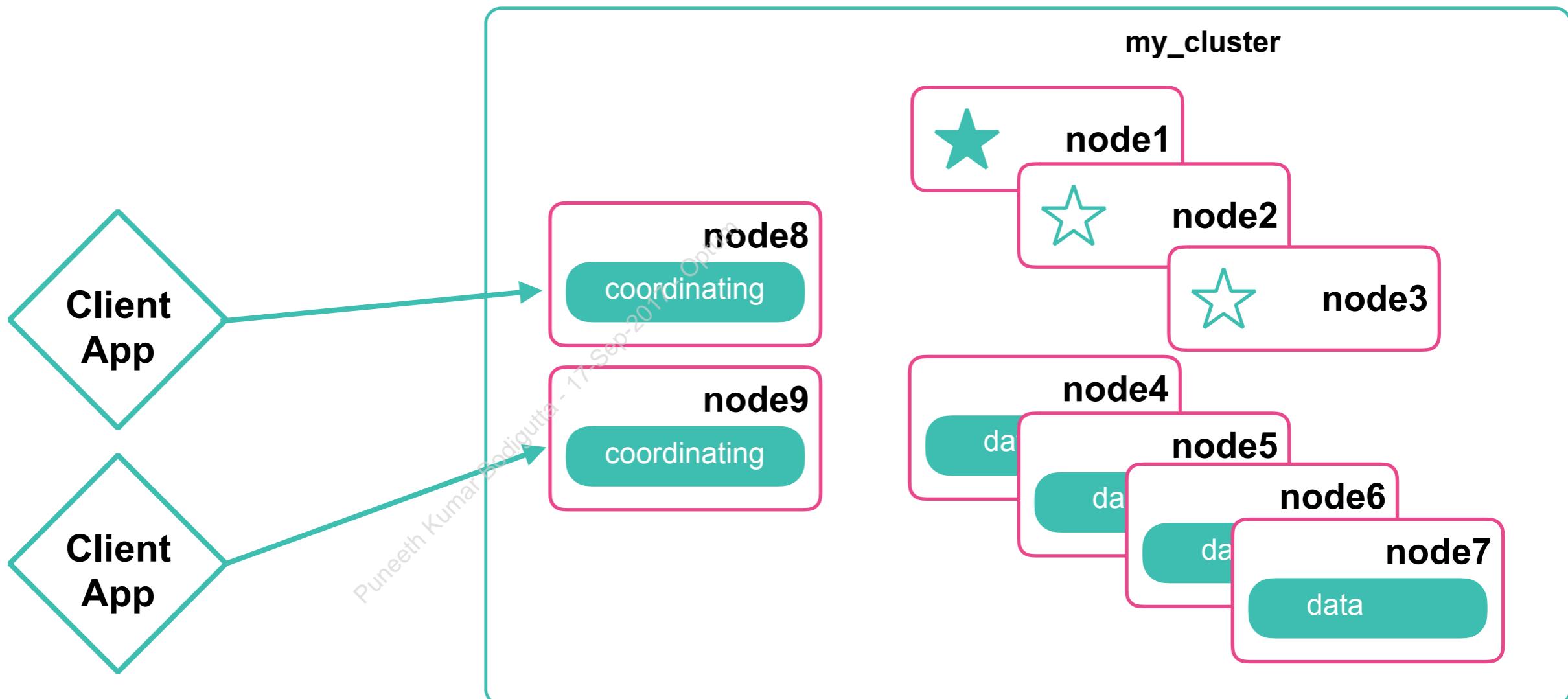
```
node.master: false  
node.data: false  
node.ingest: false
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Benefits of Coordinating Only Nodes

- behave like smart load balancers
- perform the gather/reduce phase of search queries
- lightens the load on data nodes

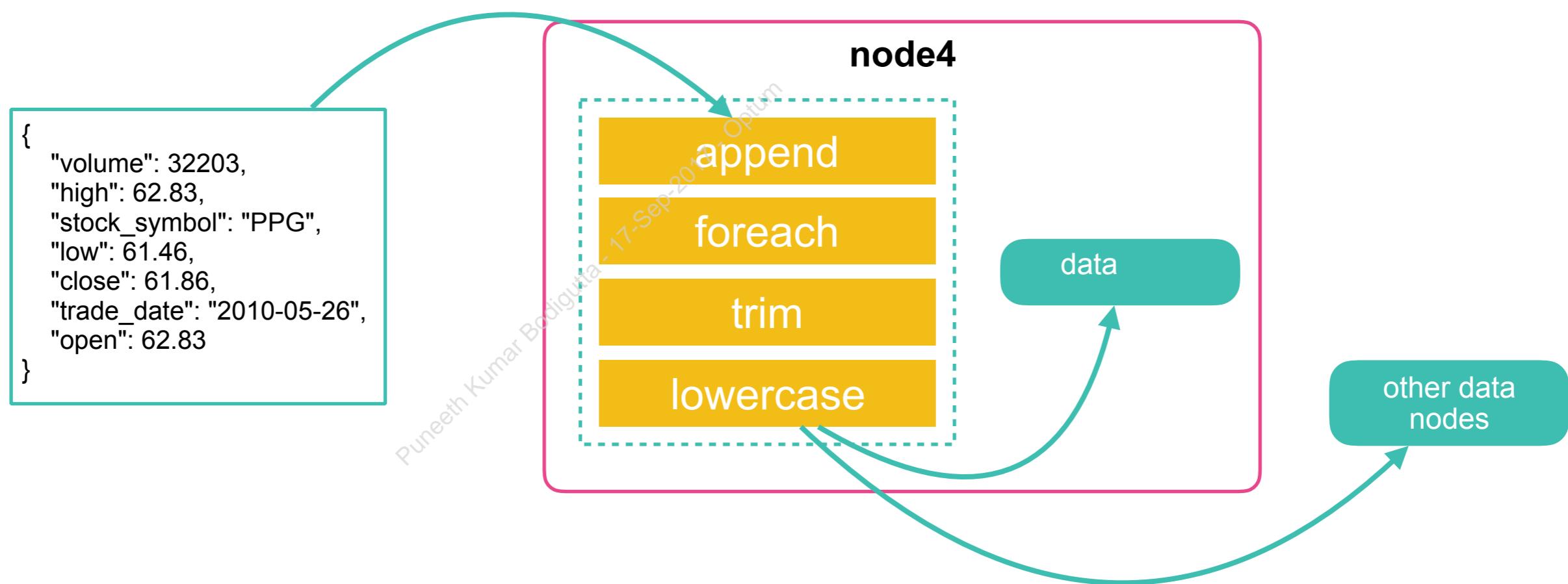


Ingest Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Ingest Nodes

- An *ingest node* is used for pre-processing documents right before they get indexed
 - Documents are passed through a *pipeline of processors* that you configure



Configuring Ingest Nodes

- Ingest is enabled by default on all nodes
 - disable ingest on a node with **node.ingest: false**
- You can have dedicated ingest nodes as well:

```
node.master: false  
node.data: false  
node.ingest: true
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Machine Learning Nodes

Puneeth Kumar Bodigutta - 17-Sep-2011, Optum

Machine Learning Nodes

- Run X-Pack Machine Learning jobs:

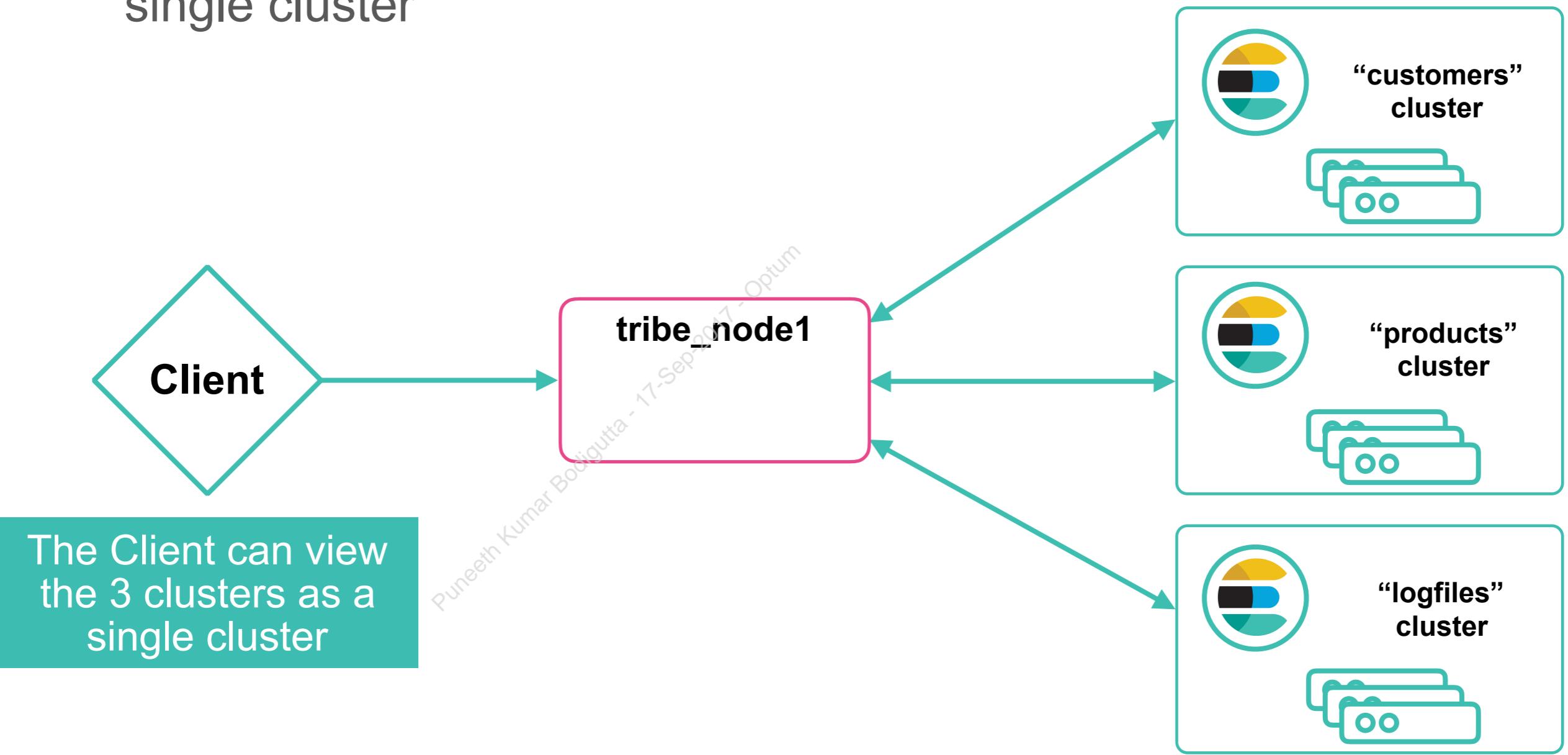


Tribe Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 , Optum

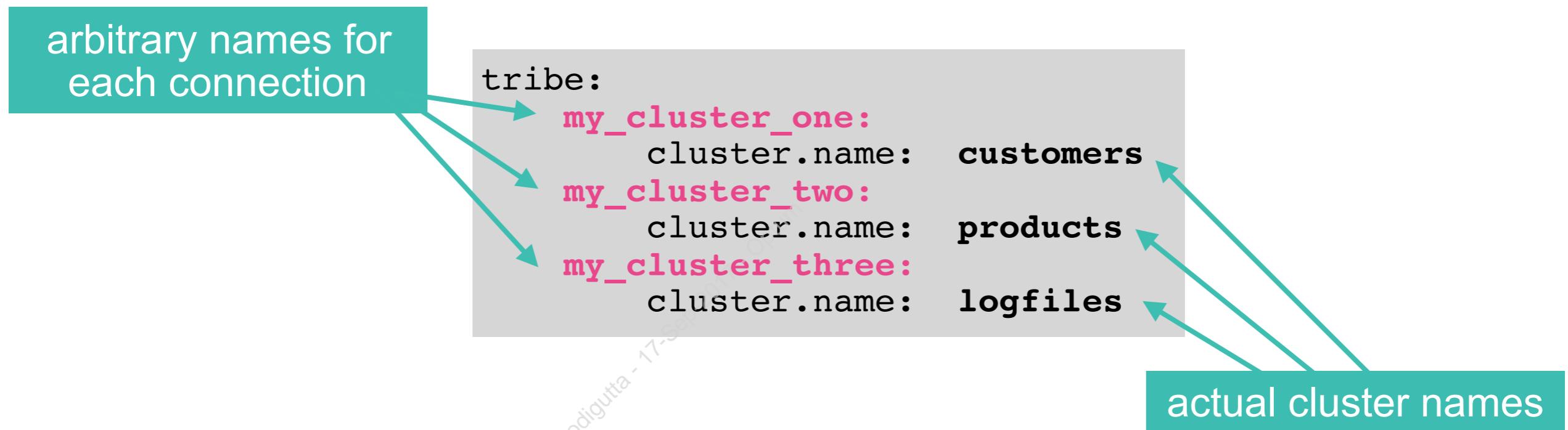
Tribe Node

- A *tribe node* connects to multiple Elasticsearch clusters and allows you to view them as if they were one big cluster
 - This allows you to search across multiple clusters as if it were a single cluster



Configuring a Tribe Node

- Configured using `tribe.*` settings:



Cross Cluster Search

- Elasticsearch 5.3 introduced a new, built-in ***cross cluster search*** feature that has made the tribe node obsolete
 - Cross cluster search is an improvement over tribe nodes and the preferred technique for this behavior
 - Tribe nodes were deprecated in ES 5.4

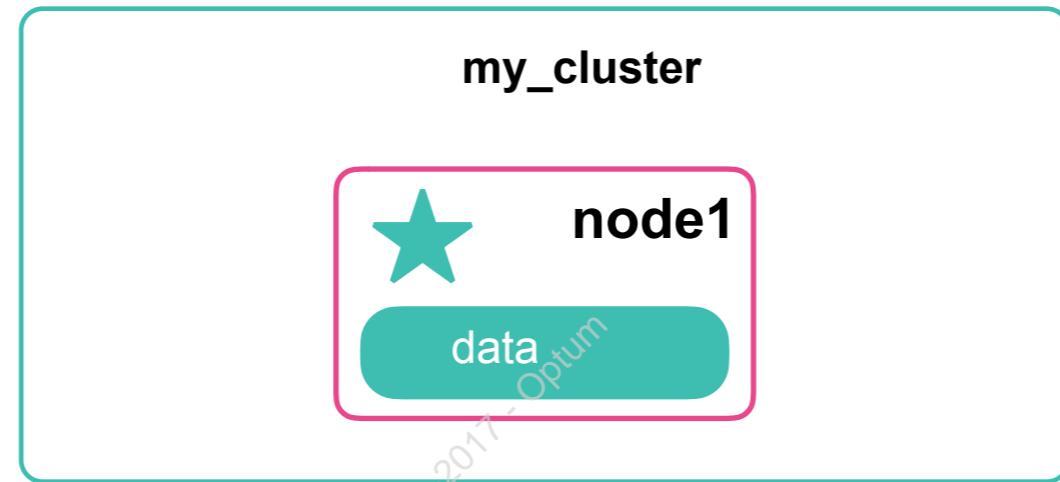
Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Sample Architectures

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optim

1-Node Cluster

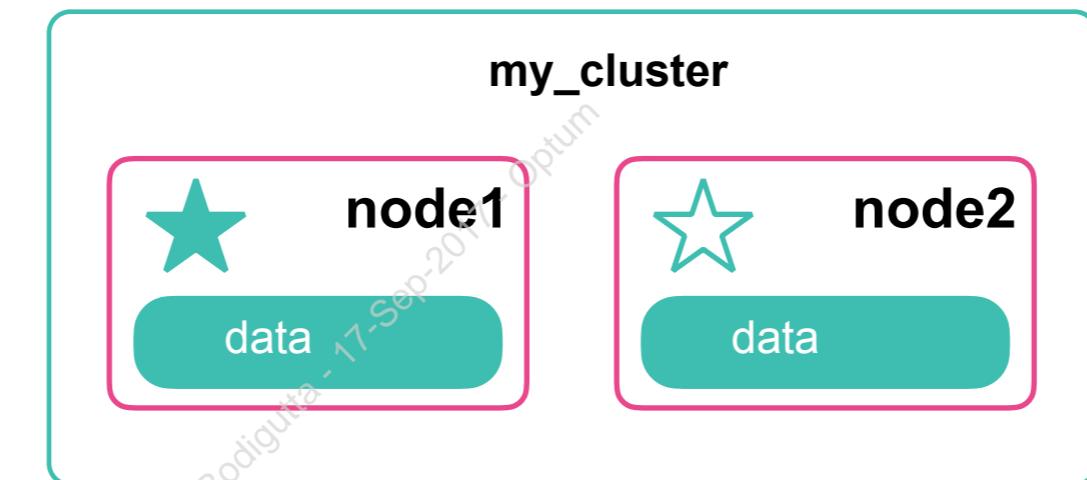
- A single node is still a cluster
 - just without any high-availability



Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

2-Node Cluster

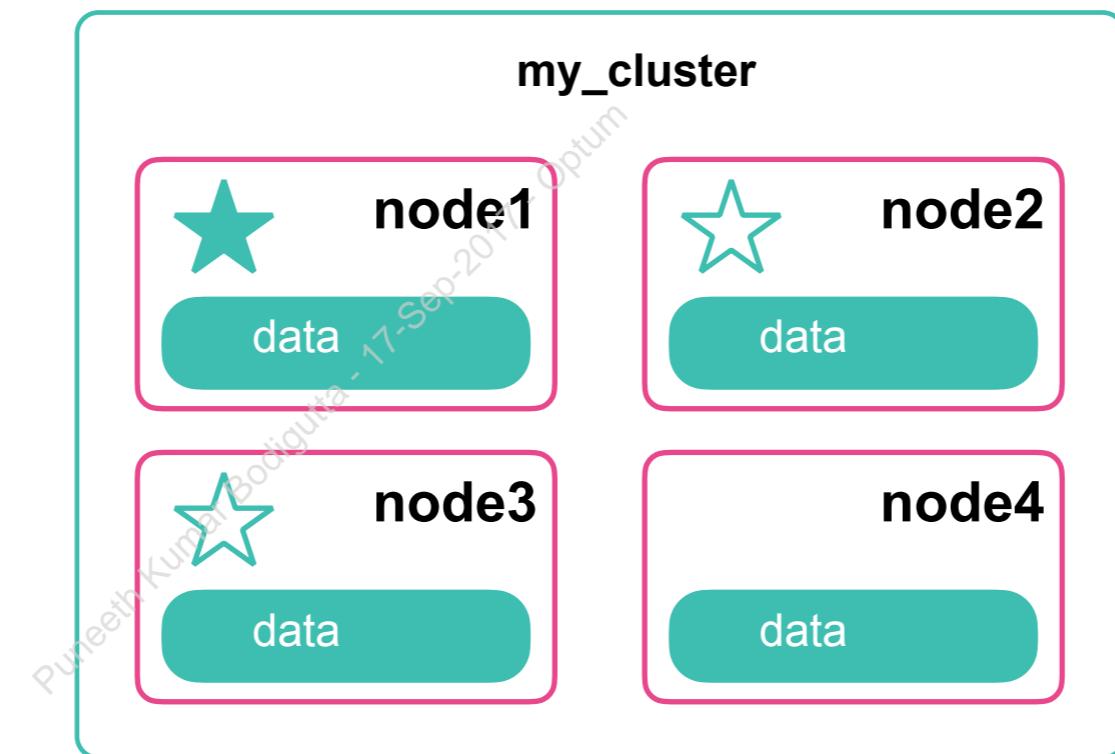
- 2-node clusters have the potential for split brain
 - either configure one dedicated master,
 - or set **minimum_master_nodes = 2**



Be careful with this cluster - it has the potential for split brain

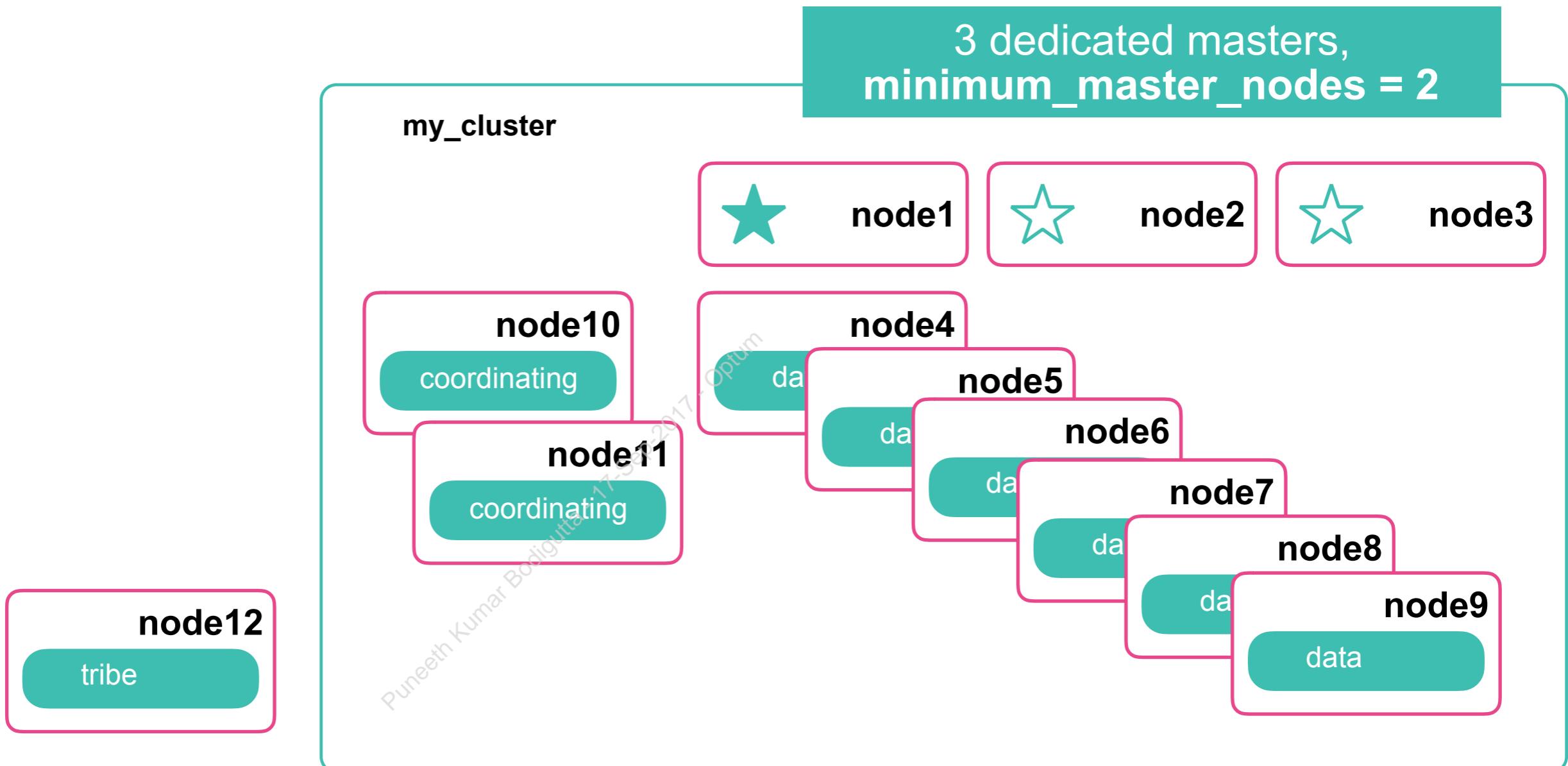
3-5 Node Cluster

- Suppose you have a small production cluster
 - it is always preferred to have 3 master eligible nodes,
 - and set **minimum_master_nodes = 2**



Larger Clusters

- Larger clusters should have dedicated nodes



Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- With ***zen discovery***, a node issues ping requests to find other nodes on the same network
- ***Cluster state*** is managed by the master node and maintained on each node in the cluster
- The ***master node*** is chosen by an election process that occurs between all master eligible nodes
- ***Data nodes*** hold the shards that contain the documents you have indexed
- A ***coordinating only node*** is specifically configured to *not* be a master, data or ingest node
- An ***ingest node*** is used for pre-processing documents right before they get indexed



Quiz

1. **True or False:** In zen discovery, nodes find each other using multicast.
2. What are the three roles that a node takes on by default?
3. **True or False:** To create a new index, a client application must send the request to the master node.
4. **True or False:** You can configure a node so that it is not a coordinating node.
5. How do you configure a dedicated data node?
6. How do you configure a coordinating only node?
7. What is the preferred deployment of master eligible nodes in a production cluster?
8. **True or False:** If a master node fails, the cluster becomes unavailable to clients until the failed node rejoins.



Lab 3

Working with Nodes

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optimal

Chapter 4

Indexes

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Understanding Shards
- Configuring an Index
- Document Routing
- Deleting an Index
- Aliases
- Index Templates

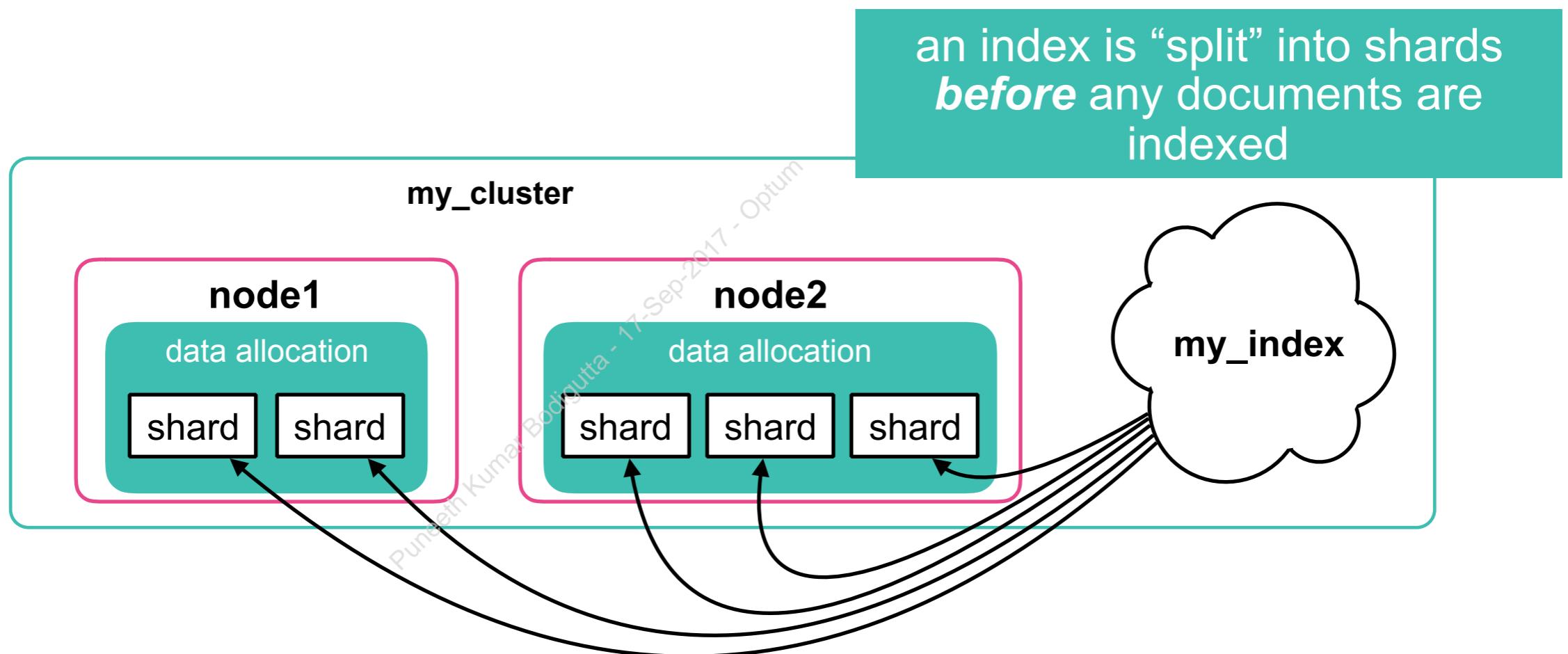
Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Understanding Shards

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

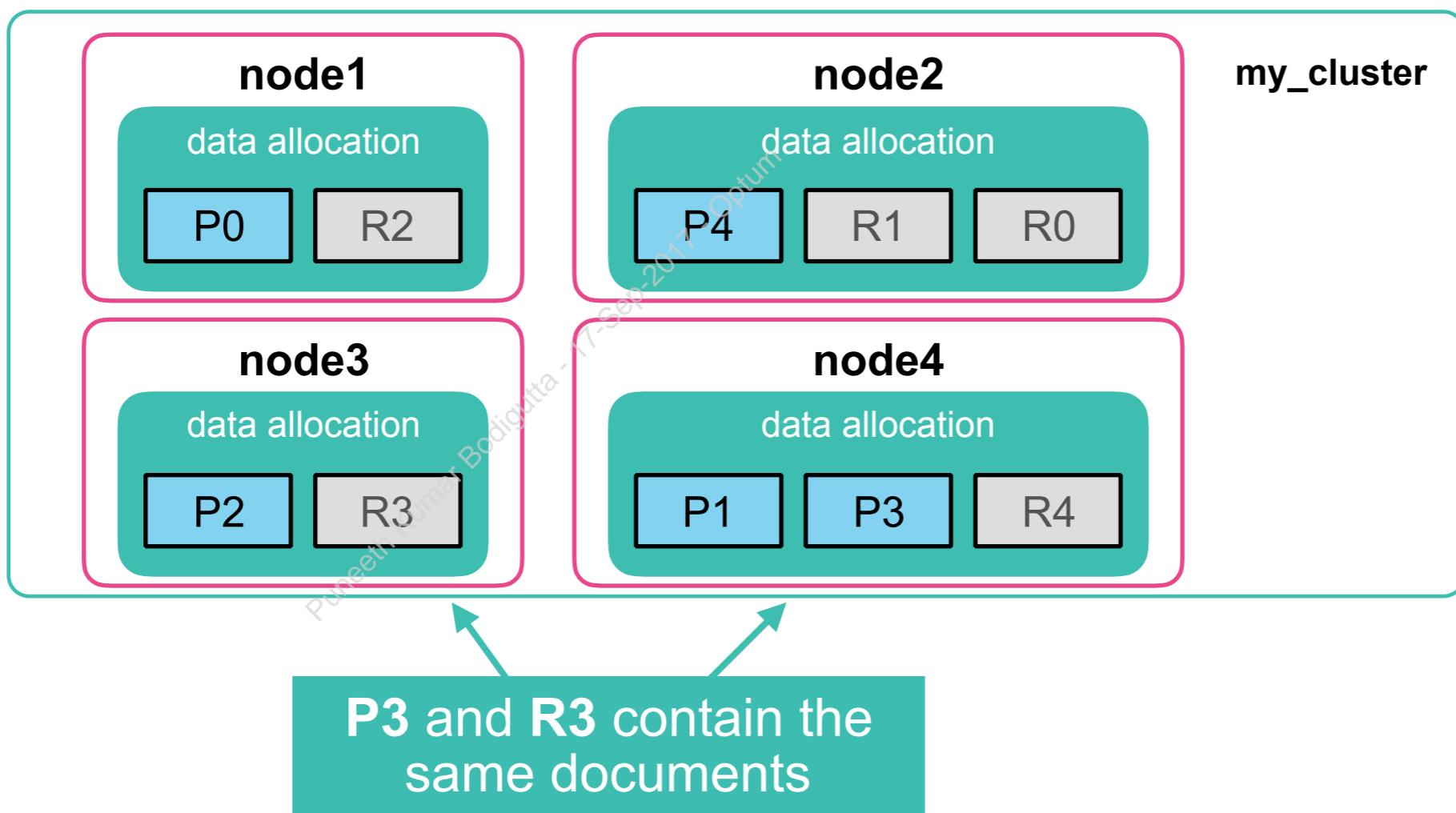
What are Shards?

- A **shard** is a worker unit that holds data and can be assigned to nodes
 - An index is a virtual namespace which points to a number of shards



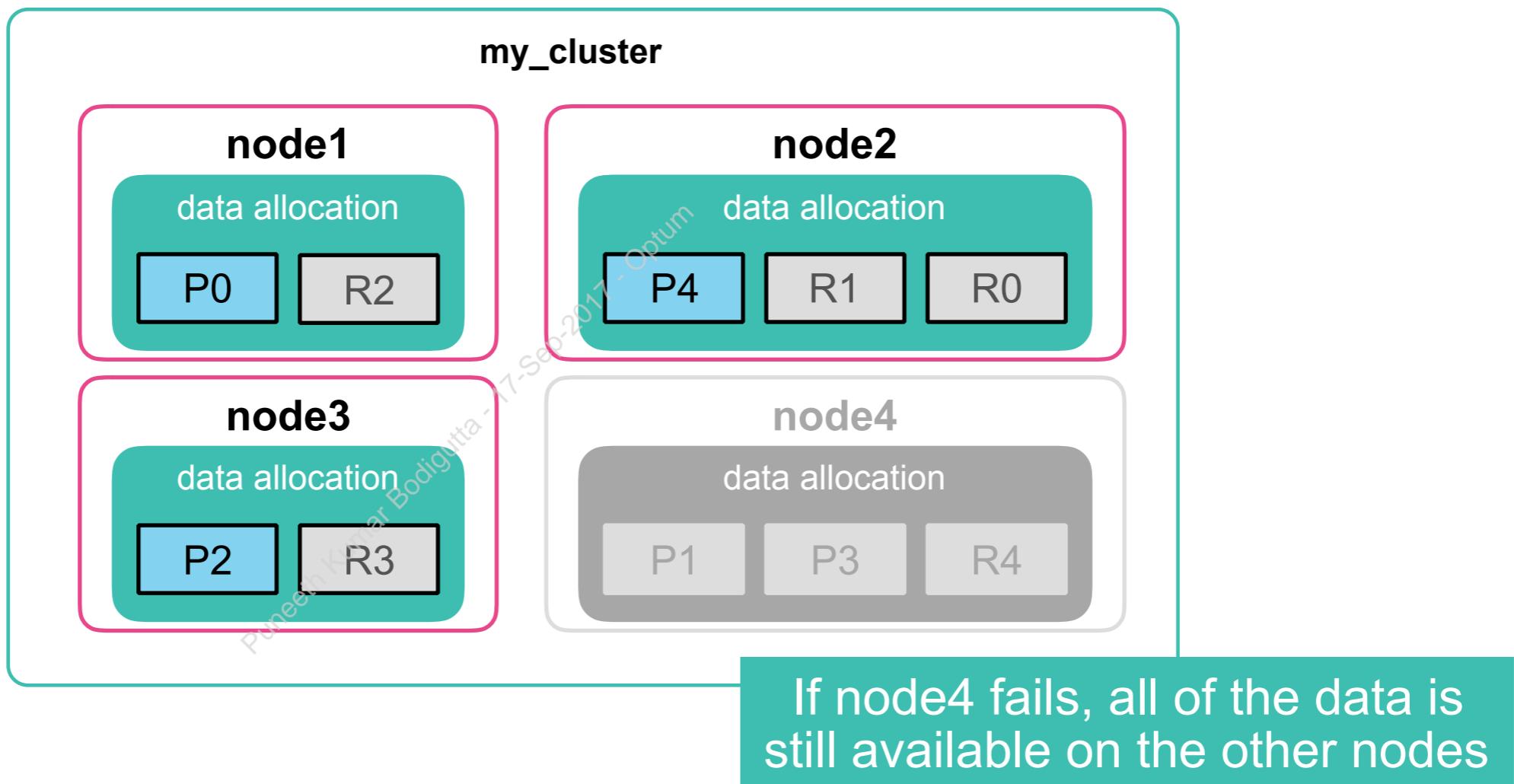
Primary vs. Replica

- There are two types of shards
 - **primary**: the original shards of an index
 - **replicas**: copies of the primary
- Documents are replicated between a primary and its replicas



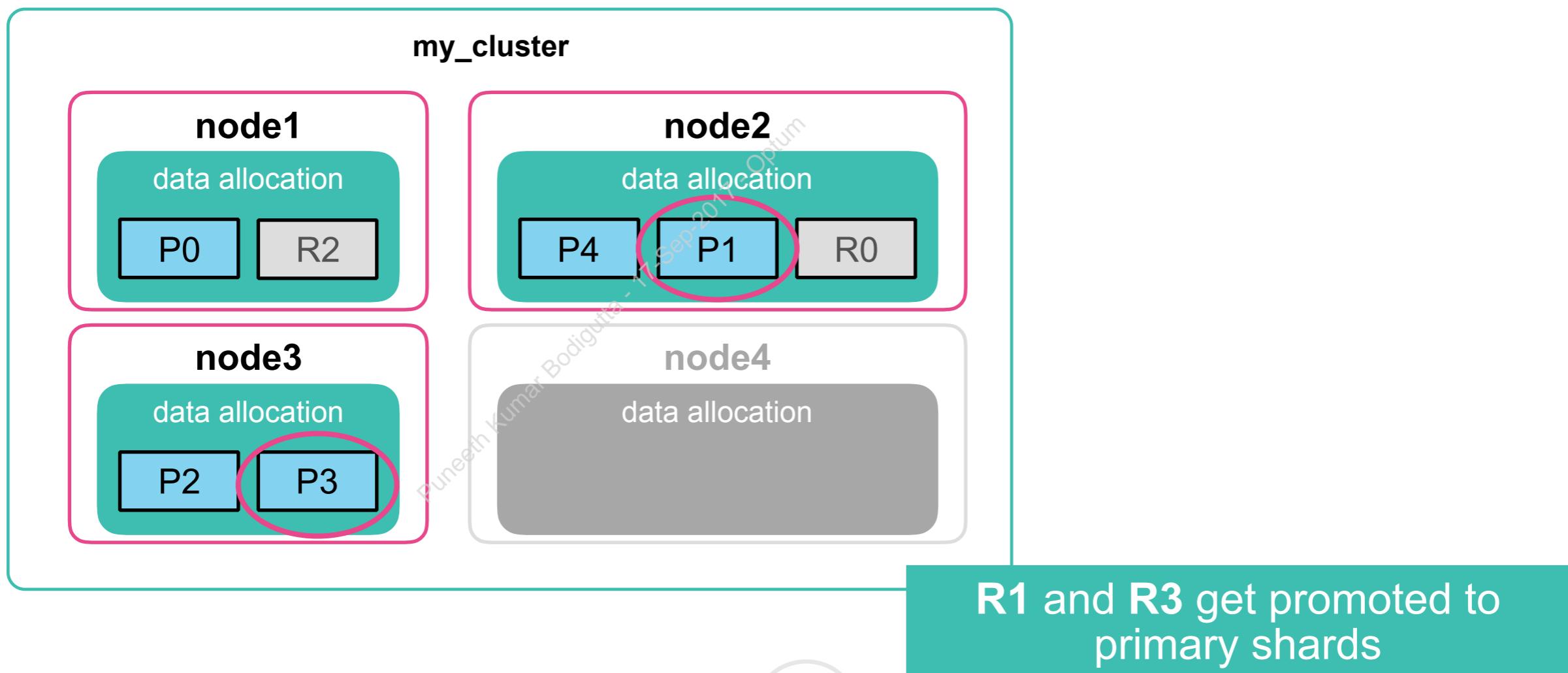
Why Create Replicas?

- High availability
 - we can lose a node and still have all the data available



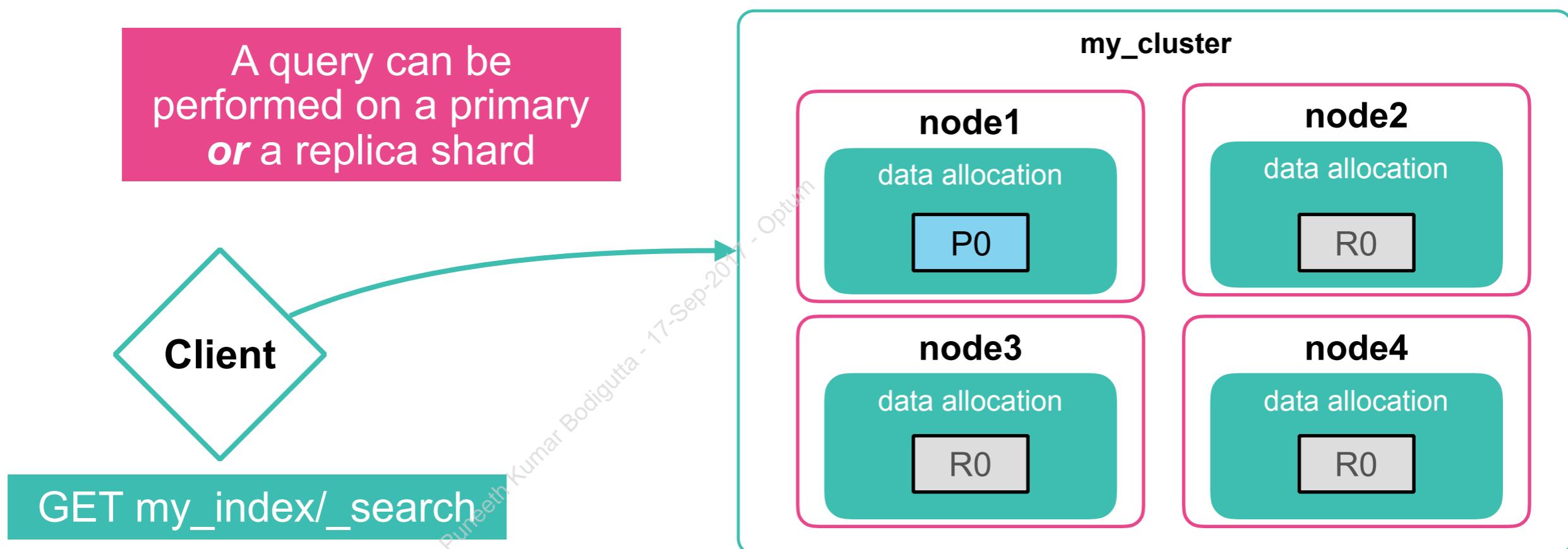
Why Create Replicas?

- High availability
 - After losing a primary, Elasticsearch will automatically promote a replica to a primary
 - and start replicating unassigned replicas



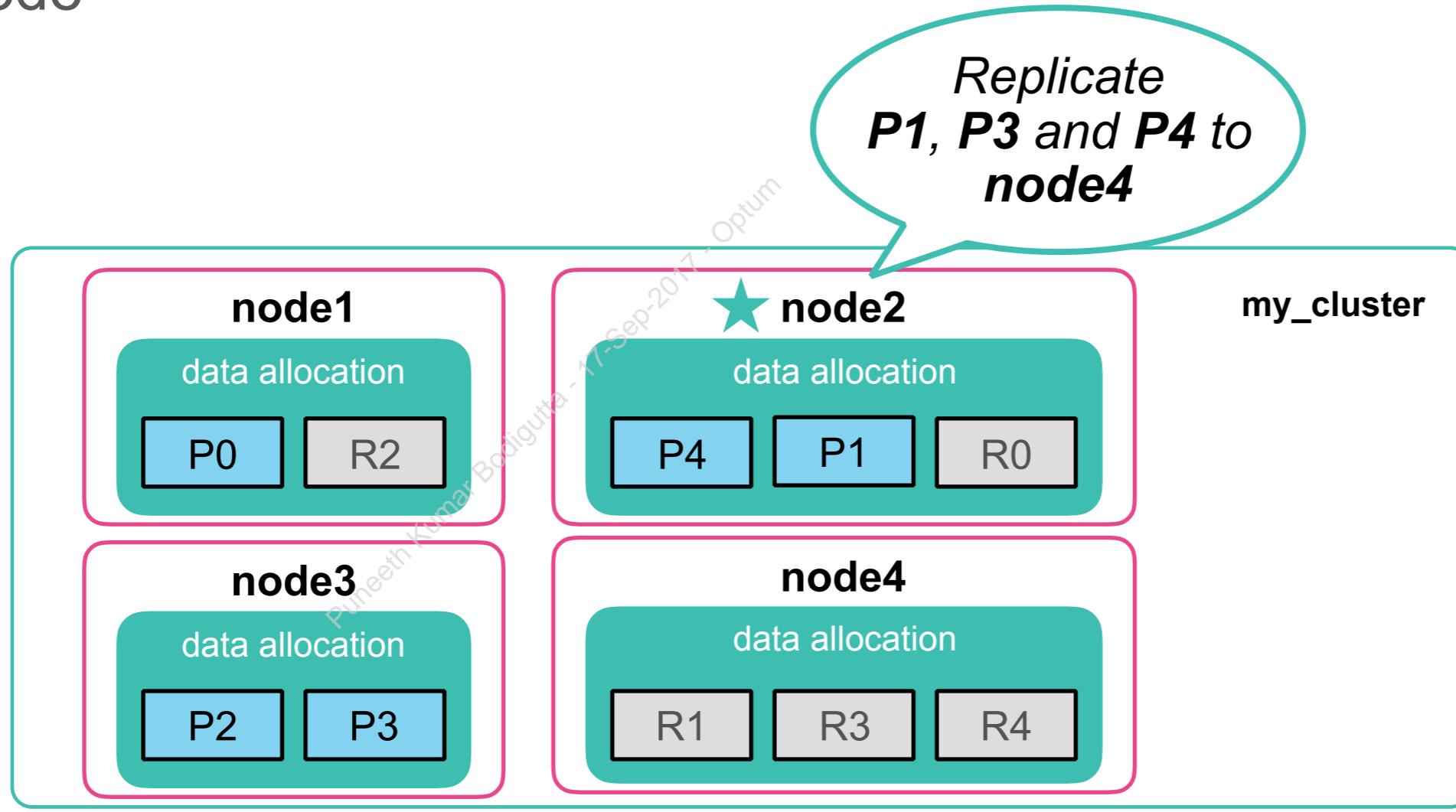
Why Create Replicas?

- Read throughput
 - Replicas can handle query/read requests from client applications
 - allows you to scale your data and better utilize cluster resources



Shard Allocation

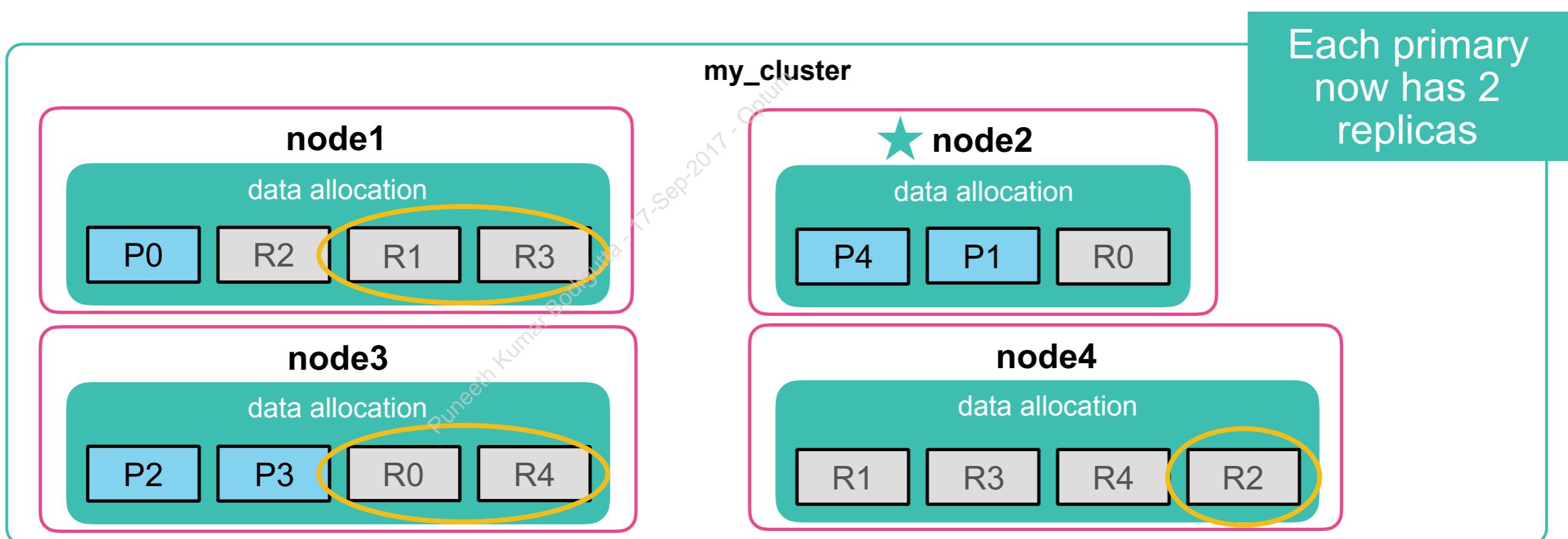
- The master node decides which shards to allocate to which nodes
 - published as part of the cluster state
 - two shards with the same ID will *never* be allocated to the same node



The Number of Replicas is Dynamic

- You can change the number of replicas for an index at any time using the `_settings` endpoint:

```
PUT my_index/_settings
{
  "number_of_replicas": 2
}
```



The Number of Primary Shards is Fixed

- The default number of primary shards for an index is 5
 - You specify the number of shards when you create the index
 - You can **NOT** increase the number of primary shards after an index is created, so choose wisely! (or *reindex your data*)

```
PUT my_new_index
{
  "settings": {
    "number_of_shards": 3
  }
}
```

number of *primary* shards

Configuring an Index

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Configuring Shards

- We have seen how to create a new index:

```
PUT my_index
```

5 primary with 1 replica = 10 total shards

- You can include **settings** for your new index as part of the **PUT** command:

```
PUT my_index
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}
```

3 primary with 2 replica = 9 total shards

Dynamic Indexes

- An index will ***dynamically*** be created during a document index request if the index does not already exist:

a new **logs** index will be created if it is not defined yet

```
PUT logs/log/1
{
  "level" : "ERROR",
  "message" : "Unable to reach host"
}
```

Puneeth Kumar Boddu
Optum

Disabling Dynamic Indexes

- You can disable this dynamic behavior completely using the dynamic **action.auto_create_index** setting:

```
PUT _cluster/settings
{
  "persistent": {
    "action.auto_create_index" : false
  }
}
```

- Or, you can whitelist certain patterns:

```
PUT _cluster/settings
{
  "persistent": {
    "action.auto_create_index" : "+.monitoring-es*,+logstash-*"
  }
}
```

Index Compression

- Elasticsearch compresses your documents during indexing
 - documents are grouped into blocks of 16KB, and then compressed together using **LZ4** by default
 - if your documents are larger than 16KB, you will have larger chunks that contain only one document
- You can change the compression to **DEFLATE** using the **index.codec** setting:
 - reduced storage size at slightly higher CPU usage

```
PUT my_index
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2,
    "index.codec" : "best_compression"
  }
}
```



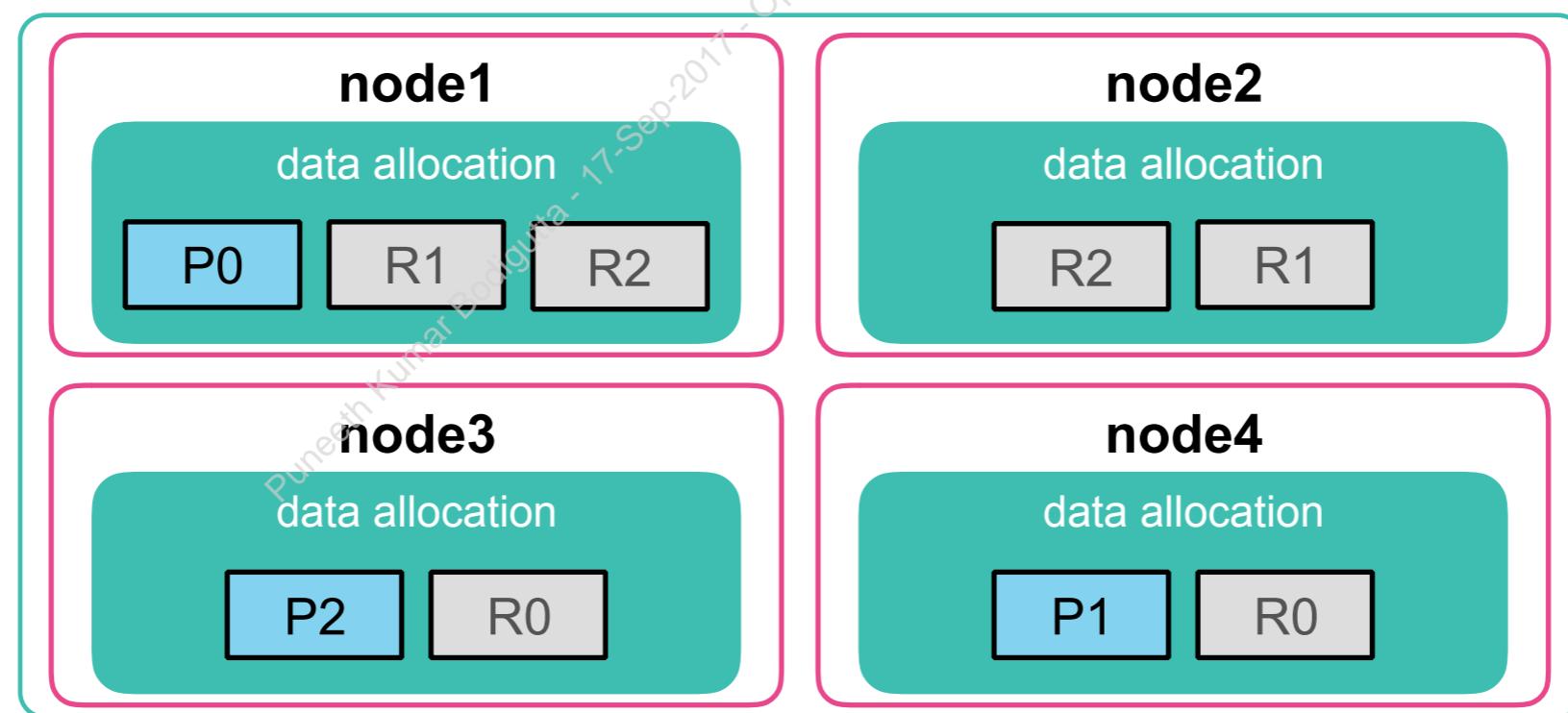
Document Routing

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Shards are Numbered

- When you create an index, the shards are numbered 0,1,2,3... and so on:

```
PUT my_index
{
  "settings": {
    "number_of_shards": 3,
    "number_of_replicas": 2
  }
}
```



Document Routing

- When you index a document, it gets routed to a specific primary shard
- By default, the shard number a document gets routed to is determined by the following formula:

```
shard = hash(routing) % number_of_primary_shards
```

“34” gets routed to P2

```
PUT my_index/doc/34  
{  
  "username": "elastic",  
  "tweet": "NEST 5.5.0 our .NET  
client released!"  
}
```

node3

data allocation

P2

Document Routing

- The routing value defaults to the document's id
 - but you can specify a different value using the **routing** parameter

This document will use “**elastic**” as its routing value

```
PUT my_index/doc/121?routing=elastic
{
  "username": "elastic",
  "tweet": "The world of #Elasticon is coming to Toronto"
}
```

Puneeth Kumar Bodugula
© Elasticsearch BV 2015-2017. All rights reserved.

Deleting an Index

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Deleting an Index

- Use a **DELETE** request to delete an index:

```
DELETE my_index
```

- You can delete multiple indices at once:

```
DELETE my_index1,my_index2
```

- You can use wildcards:

```
DELETE my_*
```

- You can delete all indices two ways:

```
DELETE _all
```

— or

```
DELETE *
```

Disabling Wildcards

- You can disable the ability to delete indices via wildcards or `_all`:
 - set `action.destructive_requires_name` to `true`

```
PUT _cluster/settings
{
  "persistent": {
    "action.destructive_requires_name" : true
  }
}
```

Puneeth Kumar Boddu (ta - 17-Sep-2017) Optum



Aliases

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

The `_aliases` Endpoint

- You can define an **alias** for an index
 - allows a “virtual” index name to be applied to one or more indices
- Defined using the `_aliases` endpoint
 - there are two actions: “**add**” and “**remove**” (similar syntax)

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "INDEX_NAME",
        "alias": "ALIAS_NAME"
      }
    }
  ]
}
```



Define an Alias

- Use the **add** action to define an alias:

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "my_index",
        "alias": "my_alias"
      }
    }
  ]
}
```

“my_alias” is an alias
for “my_index”

GET my_alias/_settings



```
{
  "my_index": {
    "settings": {
      ...
    }
  }
}
```

Define an Alias with the Endpoint

- You can also define an alias using a **PUT** sent to `_aliases`:

```
PUT my_index/_aliases/my_alias
```

index

new alias

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

An Alias with Multiple Indices

- A single alias can be associated with more than one index:

```
POST _aliases
{
  "actions": [
    {"add": {"index": "employees", "alias": "contacts"}},
    {"add": {"index": "clients", "alias": "contacts"}}
  ]
}
```

“**contacts**” is an alias for both
“**employees**” and “**clients**”

GET **contacts**/_settings

Puneeth Kondigutta - 17-Sep-2017 - Optum

```
{
  "clients": {
    "settings": {
      ...
    }
  },
  "employees": {
    "settings": {
      ...
    }
  }
}
```



Viewing Current Aliases

- Use a **GET** to view all the currently defined aliases:

GET _aliases

```
{  
  "clients": {  
    "aliases": {  
      "contacts": {}  
    }  
  },  
  "my_index": {  
    "aliases": {  
      "my_alias": {}  
    }  
  },  
  "employees": {  
    "aliases": {  
      "contacts": {}  
    }  
  }  
}
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Remove an Alias

- Use the “remove” action to delete an alias:

```
POST _aliases
{
  "actions": [
    {
      "remove": {
        "index": "my_index",
        "alias": "my_alias"
      }
    }
  ]
}
```

- You can also **DELETE** an alias, but beware that deletes all indices that it points to:

```
DELETE employees/_aliases/contacts
```

index

alias to delete

Final Comments on Aliases

- Use them!
 - you future-proof yourself if you need to point to multiple indices
 - gives Operations power to change where requests and responses come from without changing application code or configuration
- An alias pointing to multiple indices will be read-only
- It is not possible to make an alias point to another alias

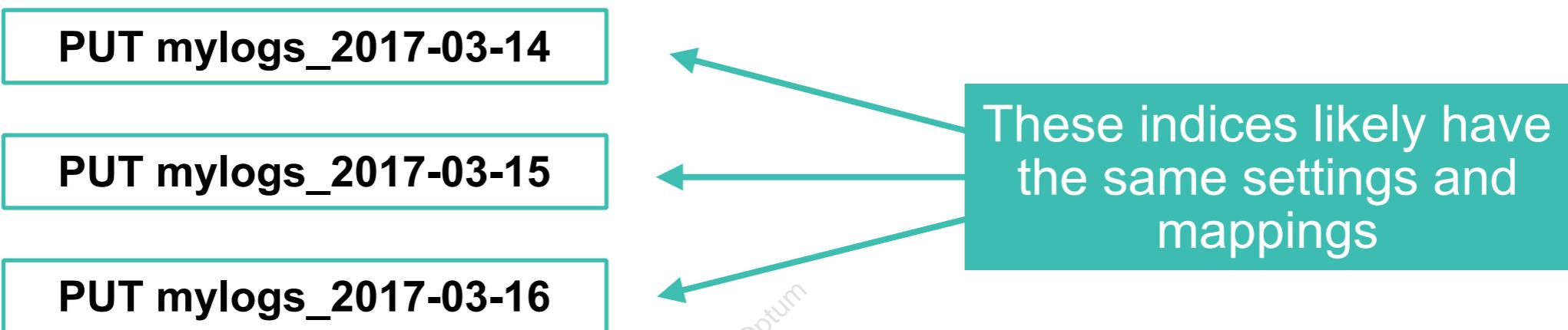
Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Index Templates

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Index Templates

- Suppose you are continuously indexing log data
 - and defining a new index for each day



- Index templates*** allow you to define mappings and settings that will automatically be applied to newly-created indices
 - e.g., we could create a template for indices named “mylogs_”

Multiple Templates

- You can have multiple index templates applied to one index
 - They get “merged” and applied to the created index
 - You provide an “**order**” value to control the merging process
- When a new index is created:
 1. Settings and mappings from the lowest “**order**” template are applied
 2. Settings and mappings from the higher “**order**” templates are applied, overriding previous settings along the way
 3. The setting and mappings from the PUT command of the index are applied last and override all previous templates

Puneeth Kumar
Optum
17-May-2017

Defining a Template

- Use the `_template` endpoint to add, view and delete templates

```
PUT _template/my_template_1
{
  "template" : "*",
  "order" : 1,
  "settings": {
    "number_of_shards": 3
  }
}
```

name of the template

Apply this template to any new index

Any new index will have 3 shards, unless otherwise overridden

Puneeth Kumar Bodigunta / Sept 2017 Optum

Let's define a second template:

```
PUT _template/my_template_2
{
  "template" : "mylogs_*",
  "order" : 5,
  "settings": {
    "number_of_shards": 5,
    "number_of_replicas": 2
  }
}
```

Apply this template to
any new index that
starts with “**mylogs_**”

any new index that starts with
“**mylogs_**” will have 5 primary
shards and 2 replicas

Now define a new index:

- Which template(s) are applied to each of the following indexes?

```
PUT test_index
```

```
PUT mylogs_2017-03-14
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Let's verify the templates:

```
GET test_index/_settings
```

```
"settings": {  
  "index": {  
    "number_of_shards": "3",  
    "number_of_replicas": "1",  
    ...  
  }  
}
```

```
GET mylogs_2017-03-14/_settings
```

```
"settings": {  
  "index": {  
    "number_of_shards": "5",  
    "number_of_replicas": "2",  
    ...  
  }  
}
```

Viewing and Deleting Templates

- Use a **GET** to view your defined templates:

```
GET _template
```

- You can **DELETE** a template also:

```
DELETE _template/my_template_2
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Aliases in a Template

- You can include **aliases** in an index template:

```
PUT _template/my_template_2
{
  "template" : "mylogs_*",
  "order" : 5,
  "settings": {
    "number_of_shards": 5,
    "number_of_replicas": 2
  },
  "aliases" : {
    "mylogs_today" : {},
    "{index}-alias" : {}
  }
}
```

This example adds two aliases to the new index

{index} is the name of the index being created

Aliases in a Template

- What would the aliases be for the following new index?

```
PUT mylogs_2017-03-15
```

- It will have two aliases:

```
mylogs_2017-03-15-alias  
mylogs_today
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- A **shard** is a worker unit that holds data and can be assigned to nodes
- There are two types of shards: **primary** and **replicas**
- The number of primary shards is fixed at index creation time, but the number of replicas is dynamic and can be changed at any time
- Documents are compressed using **LZ4** by default
- You can define an **alias** for an index using the **_aliases** endpoint
- **Index templates** allow you to define mappings and settings that will automatically be applied to newly-created indices
- You can define **aliases** in an index template



Quiz

1. An **index** is a virtual _____ which points to a number of _____.
2. What is the default value of **number_of_replicas** for an index?
3. **True or False:** Changing the compression of an index to **DEFLATE** provides better performance.
4. What is the result of the command “**DELETE ***”?
5. **True or False:** An index template with **order=10** would override the settings of an index template with **order=5**?
6. **True or False:** An alias pointing to multiple indices will be read-only.

Puneeth Kumar Bodigere
17-Sep-2017

Lab 4

Indexes

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open

Chapter 5

Cluster Health

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Cluster Health
- Diagnosing Health Issues
- The `wait_for_status` Parameter
- Shard Allocation
- The Lifecycle of a Shard
- Diagnosing Shard Issues

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Cluster Health

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Cluster Health

- A cluster has a *health* that contains various statistics and the status of the cluster
 - use the **Cluster Health API** to view a cluster's health:

GET _cluster/health

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized

```
{  
  "cluster_name": "my_cluster",  
  "status": "yellow",  
  "timed_out": false,  
  "number_of_nodes": 2,  
  "number_of_data_nodes": 1,  
  "active_primary_shards": 62,  
  "active_shards": 62,  
  "relocating_shards": 0,  
  "initializing_shards": 0,  
  "unassigned_shards": 69,  
  "delayed_unassigned_shards": 0,  
  "number_of_pending_tasks": 0,  
  "number_of_in_flight_fetch": 0,  
  "task_max_waiting_in_queue_millis": 0,  
  "active_shards_percent_as_number": 47.32  
}
```



Health of a Specific Index

- You can request the health of a specific index using the following syntax:

```
GET _cluster/health/my_index
```

shard stats are for
“my_index” only

```
{  
  "cluster_name": "my_cluster",  
  "status": "yellow",  
  "timed_out": false,  
  "number_of_nodes": 2,  
  "number_of_data_nodes": 1,  
  "active_primary_shards": 5,  
  "active_shards": 5,  
  "relocating_shards": 0,  
  "initializing_shards": 0,  
  "unassigned_shards": 5,  
  "delayed_unassigned_shards": 0,  
  "number_of_pending_tasks": 0,  
  "number_of_in_flight_fetch": 0,  
  "task_max_waiting_in_queue_millis": 0,  
  "active_shards_percent_as_number": 47.014  
}
```

Health Stats

- Here are a few of the stats that are a part of the health:

active_primary_shards	aggregate total of <i>all primary</i> shards across all indices
active_shards	aggregate total of <i>all</i> shards (primary + replicas)
number_of_pending_tasks	size of the queue of cluster-level tasks to be performed
task_max_waiting_in_queue_millis	maximum time a task spent waiting in the pending-tasks queue

- We will discuss the other shard stats later in this chapter

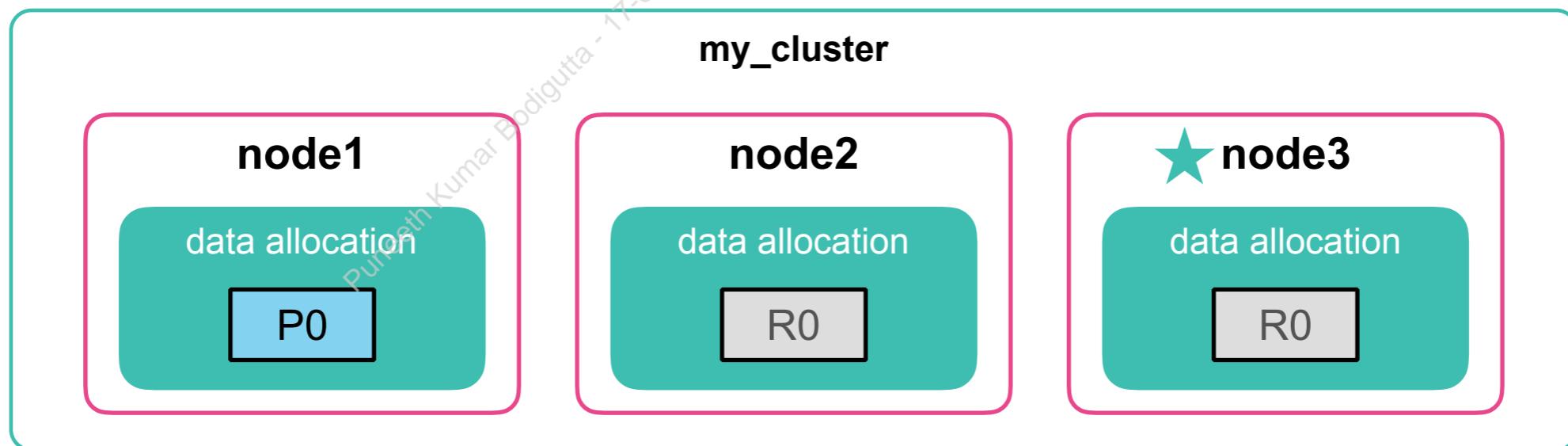
Health Status

- The ***health status*** is either green, yellow or red and exists at three levels: shard, index, and cluster
- ***shard health***
 - **red**: at least one primary shard is not allocated in the cluster
 - **yellow**: all primaries are allocated but at least one replica is not
 - **green**: all shards are allocated
- ***index health***
 - status of the **worst shard** in that index
- ***cluster health***
 - status of the **worst index** in the cluster

Green Status

- **Green** is good! All of your primary and replica shards are allocated and active:

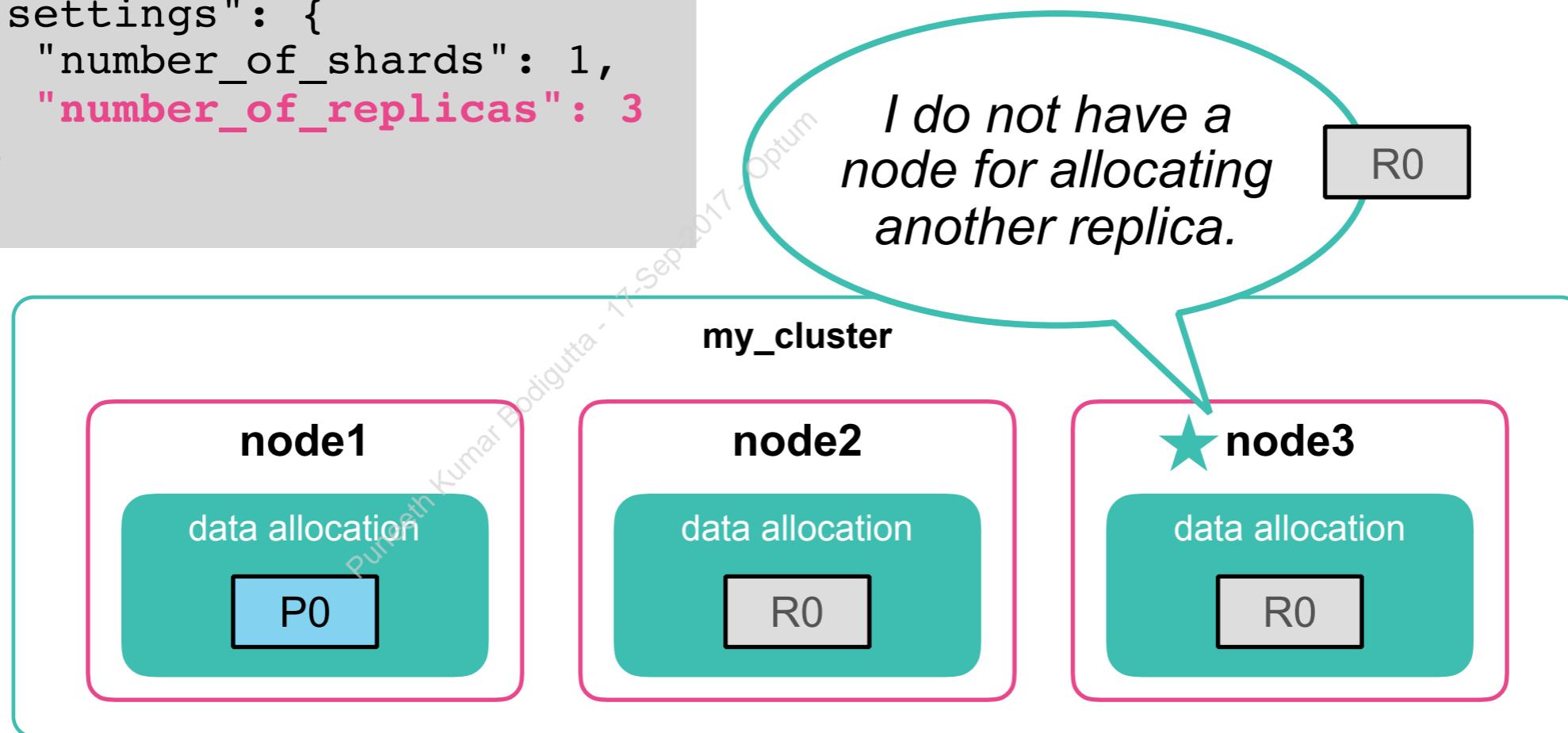
```
PUT my_index_1
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 2
  }
}
```



Yellow Status

- **Yellow** means all your primary shards are allocated, but **one or more replicas are not**
 - you may not have enough nodes in the cluster, or a node may have failed

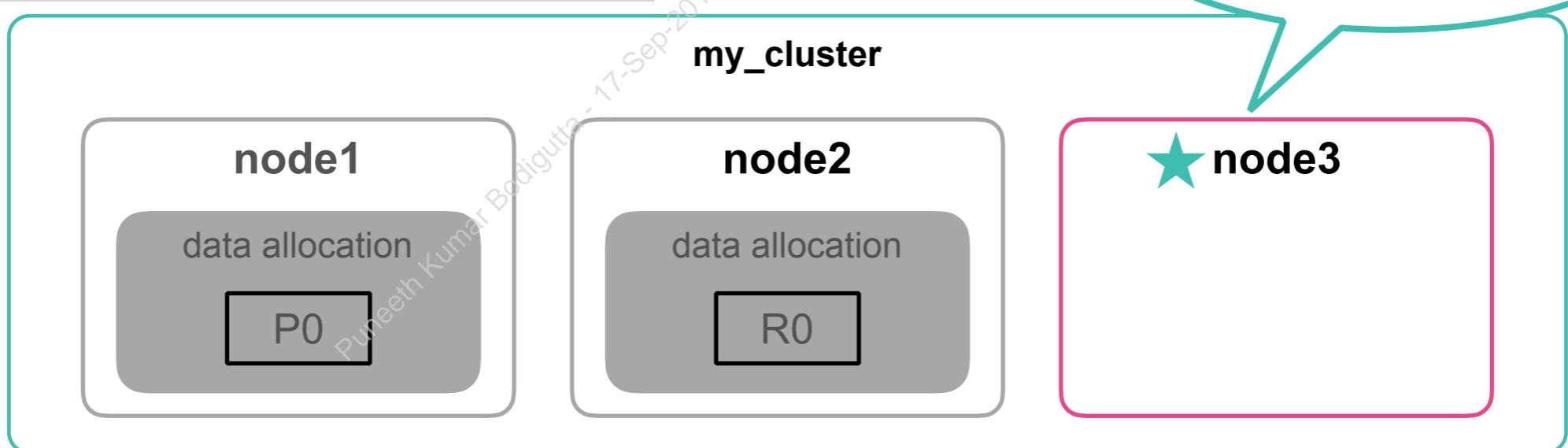
```
PUT my_index_2
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 3
  }
}
```



Red Status

- **Red** means you have *at least one primary* missing
 - searches will return partial results and indexing might fail

```
PUT my_index_3
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1
  }
}
```



Diagnosing Health Issues

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Finding Health Issues

- Suppose your cluster is having issues
 - getting the health at the *cluster level* can reveal if nodes are down

GET `_cluster/health`

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

```
{  
  "cluster_name": "my_cluster",  
  "status": "red",  
  "timed_out": false,  
  "number_of_nodes": 2,  
  "number_of_data_nodes": 2,  
  "active_primary_shards": 60,  
  "active_shards": 120,  
  ...  
}
```

Suppose you have a 4 node cluster, but only 2 appear in the health



Drilling Down to Indices

- A red cluster means you have at least one red index
 - getting the health at the *index level* reveals the problem index (or indices)

```
GET _cluster/health?level=indices
```

set “**level**” equal to “**indices**” to view health of all indices

“**my_index**” should have 5 active primaries, but it only has 2

```
{  
  "cluster_name": "my_cluster",  
  "status": "red",  
  ...  
  "indices": {  
    "my_index": {  
      "status": "red",  
      "number_of_shards": 5,  
      "number_of_replicas": 1,  
      "active_primary_shards": 2,  
      "active_shards": 3,  
      "relocating_shards": 0,  
      "initializing_shards": 0,  
      "unassigned_shards": 5  
    },  
    ...  
  }  
}
```

Shard Level

- You can drill down even further and determine which shards within the index are red:

```
GET _cluster/health?level=shards
```

helpful, but it is often
too many details

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

```
{  
  "cluster_name": "my_cluster",  
  "status": "red",  
  ...  
  "indices": {  
    "my_index": {  
      "status": "red",  
      "number_of_shards": 5,  
      "number_of_replicas": 1,  
      "active_primary_shards": 2,  
      "active_shards": 3,  
      "relocating_shards": 0,  
      "initializing_shards": 0,  
      "unassigned_shards": 5,  
      "shards": {  
        "0": {  
          "status": "red",  
          "primary_active": false,  
          "active_shards": 0,  
          "relocating_shards": 0,  
          "initializing_shards": 0,  
          "unassigned_shards": 0  
        },  
        ...  
      },  
      ...  
    },  
    ...  
  },  
  ...  
}
```



The wait_for_status Parameter

Puneeth Kumar Bodigutta - 17-Sep-2014 Optum

The `wait_for_status` Parameter

- You can specify a `wait_for_status` parameter when requesting health
 - the call will block until the status matches the value you provide
 - useful in automated scripts or in testing
 - add the `timeout` parameter to specify how long you want to wait

blocks until the cluster
status is **green**

```
GET _cluster/health?wait_for_status=green
```

```
GET _cluster/health/my_index?wait_for_status=yellow
```

blocks until the status
of “**my_index**” is
yellow or **green**



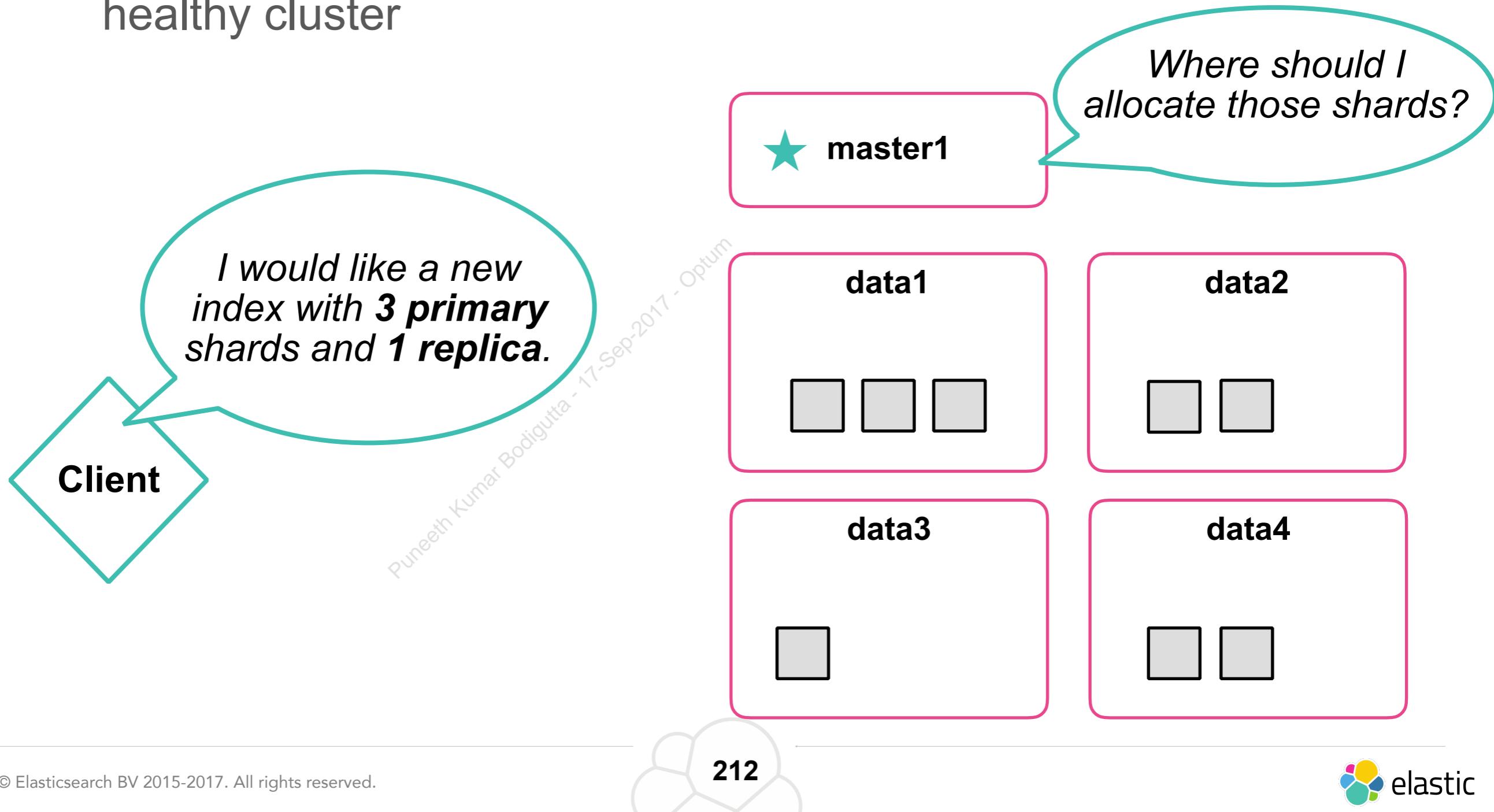
Shard Allocation

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open



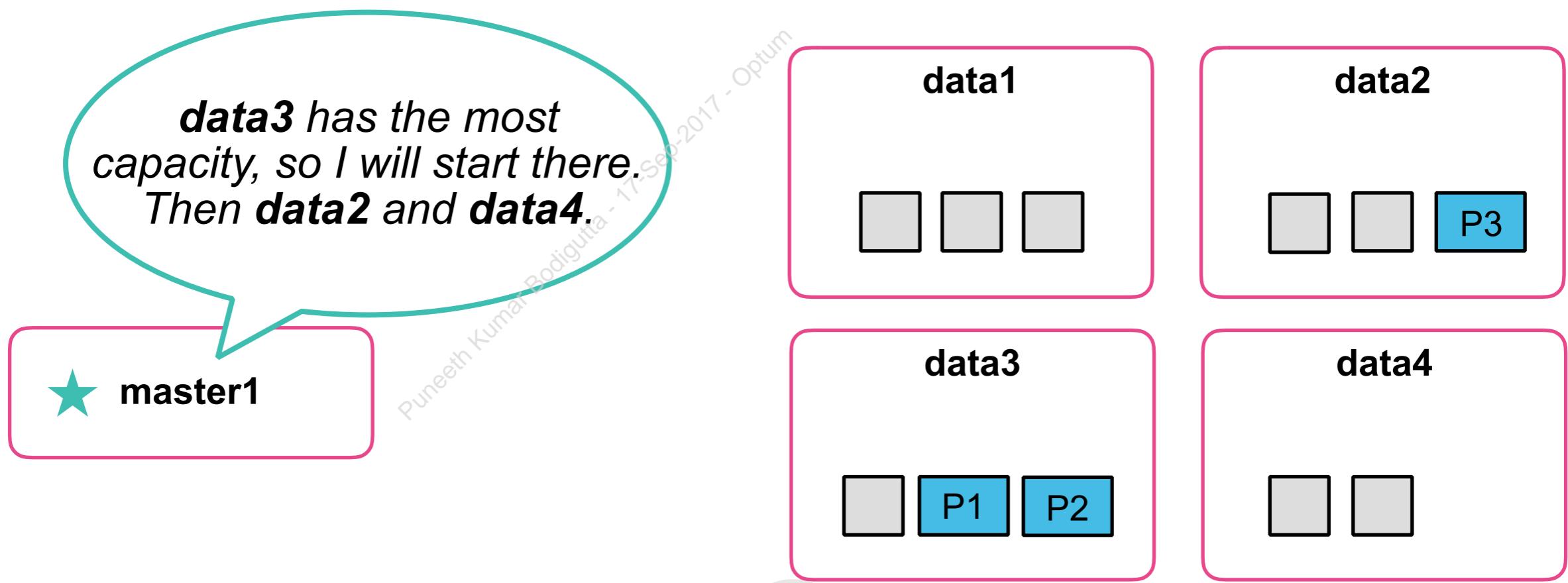
Shard Allocation

- **Shard allocation** is the process of assigning a shard to a node in the cluster
 - assigning shards to the best node possible is essential for a healthy cluster



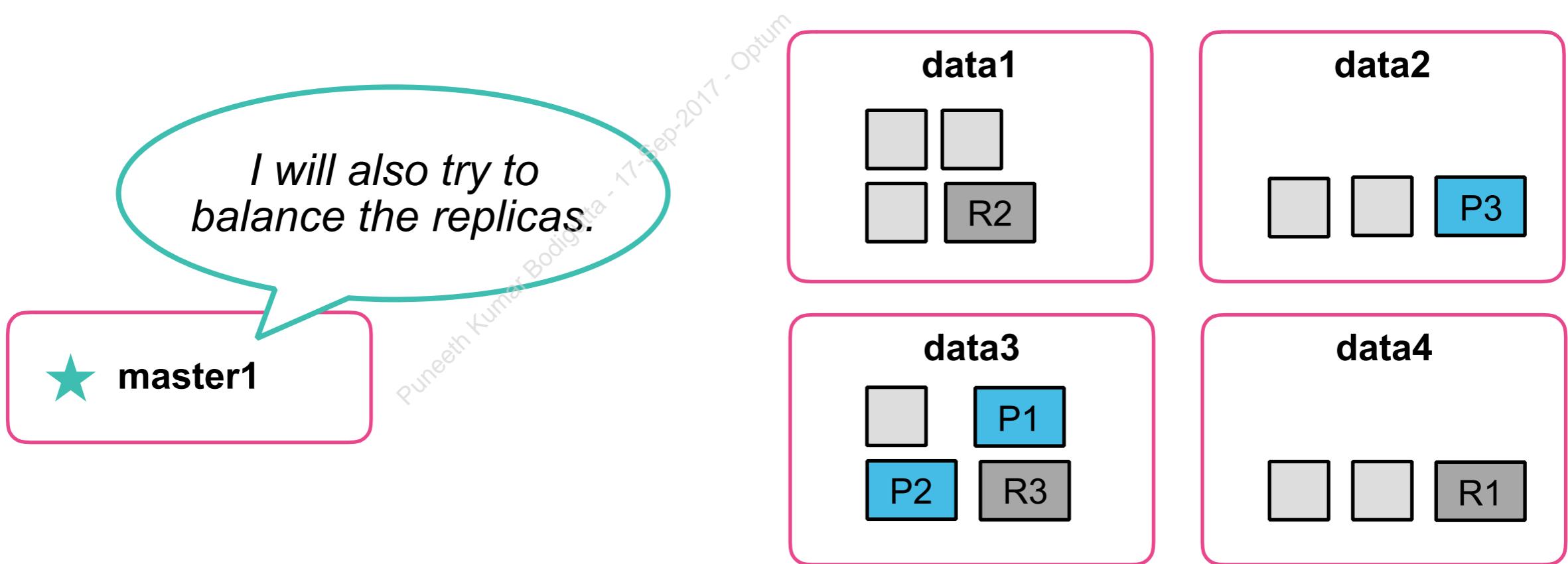
New Index Allocation

- For newly created indices, nodes with the least amount of shards are assigned first
 - assuming other requirements like permission and storage limitations are met
 - the goal is to keep the cluster balanced



Allocating the Replicas

- When allocating replicas, a replica shard can not appear on the same node as its primary shard
 - otherwise redundancy would not be achieved



The Routing Table

- The *routing table* contains which nodes are hosting which indices and shards
 - part of the cluster state, so every node has this information

```
GET _cluster/state/routing_table
```

```
{  
  "cluster_name": "my_cluster",  
  "routing_table": {  
    "indices": {  
      "my_index": {  
        "shards": {  
          "0": [  
            {  
              "state": "UNASSIGNED",  
              "primary": true,  
              "node": null,  
              "relocating_node": null,  
              "shard": 0,  
              "index": "my_index",  
              "...  
            ]  
          ]  
        }  
      }  
    }  
  }  
}
```

Puneeth Kumar Bodigutta - 17-Sep-2017

Routing Table of an Index

- You can get just the routing table of an index:

```
GET _cluster/state/routing_table/employees
```

Returns the routing table
for just the **employees**
index

```
{  
  "cluster_name": "my_cluster",  
  "routing_table": {  
    "indices": {  
      "employees": {  
        "shards": {  
          "0": [  
            {  
              "state": "UNASSIGNED",  
              "primary": true,  
              "node": null,  
              ...  
            ]  
          ]  
        }  
      }  
    }  
  }  
}
```

The Lifecycle of a Shard

Puneeth Kumar Bodigutta - 17-Sep-2011 · Optum

The Lifecycle of a Shard

- Shards go through several states:
 - UNASSIGNED
 - INITIALIZING
 - STARTED
 - RELOCATING
- Let's follow the lifecycle of a shard...

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized



UNASSIGNED Shards

- **UNASSIGNED** describes shards that exist in the cluster state, but cannot be found in the cluster itself
 - e.g., more replicas assigned than nodes in the cluster; a node failed; brand new index
 - What is the state of this cluster right now?

```
PUT my_index
{
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 1
  }
}
```

Puneeth Kumar Bodugula - 17 Sep 2017 - Optum

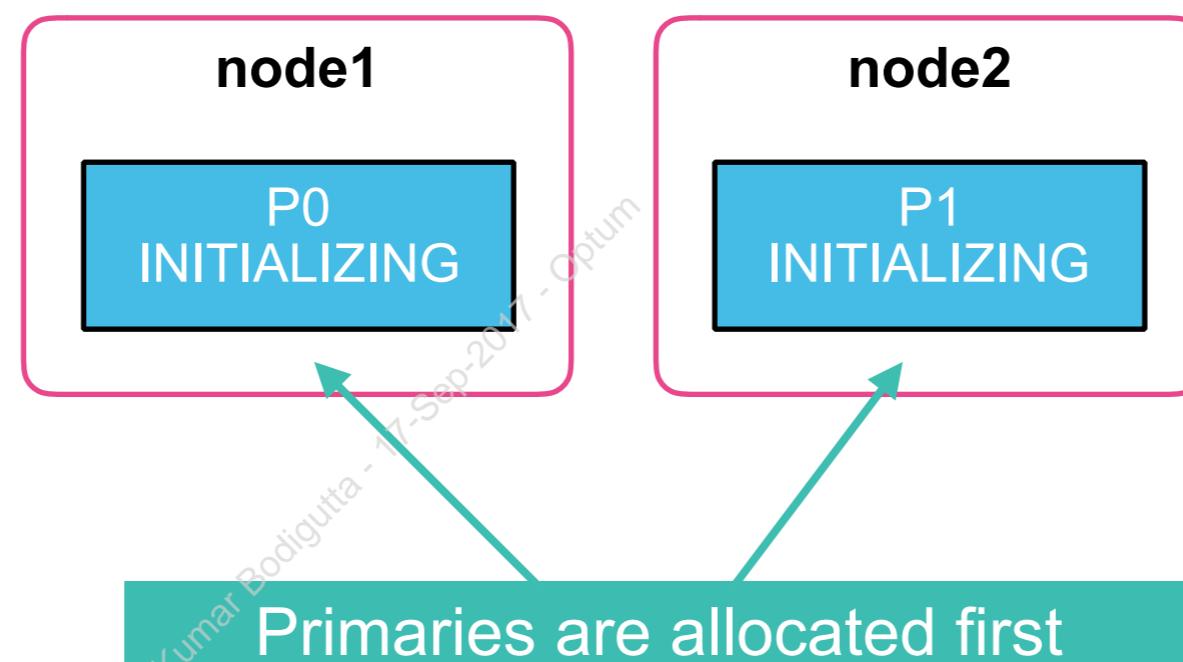
Four shards total, all in
the UNASSIGNED state

node1

node2

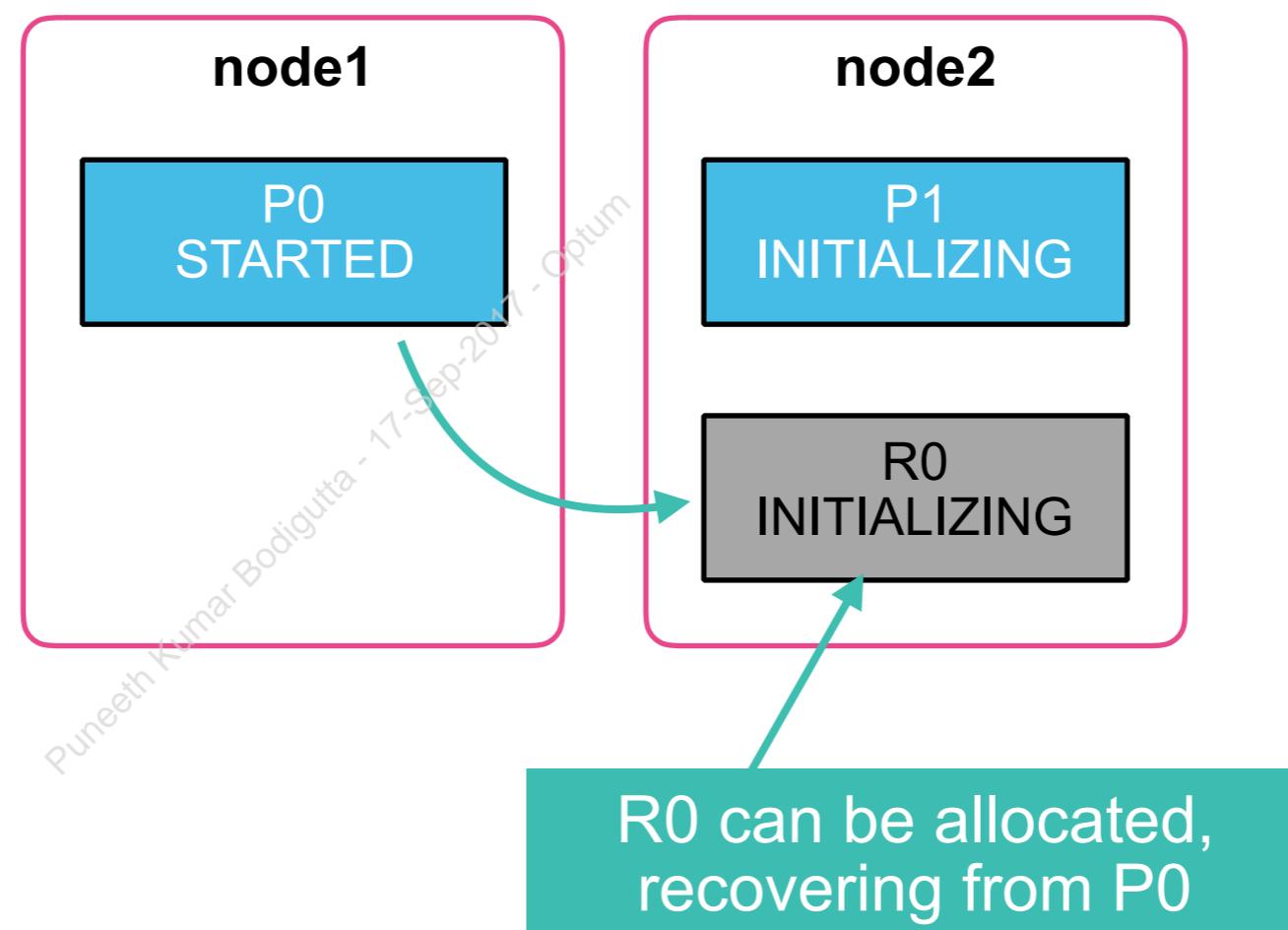
INITIALIZING

- Shards are briefly in the **INITIALIZING** state as they are being created:



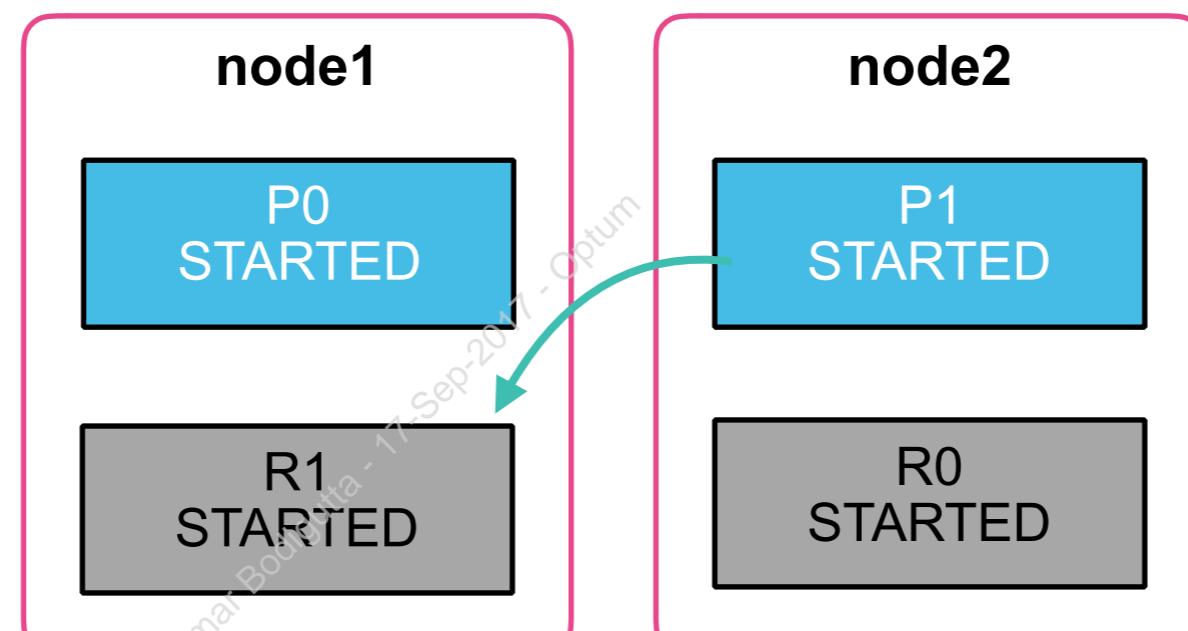
STARTED

- When a shard is done initializing, it moves to the **STARTED** state
 - and its replicas can now be allocated



Green Status

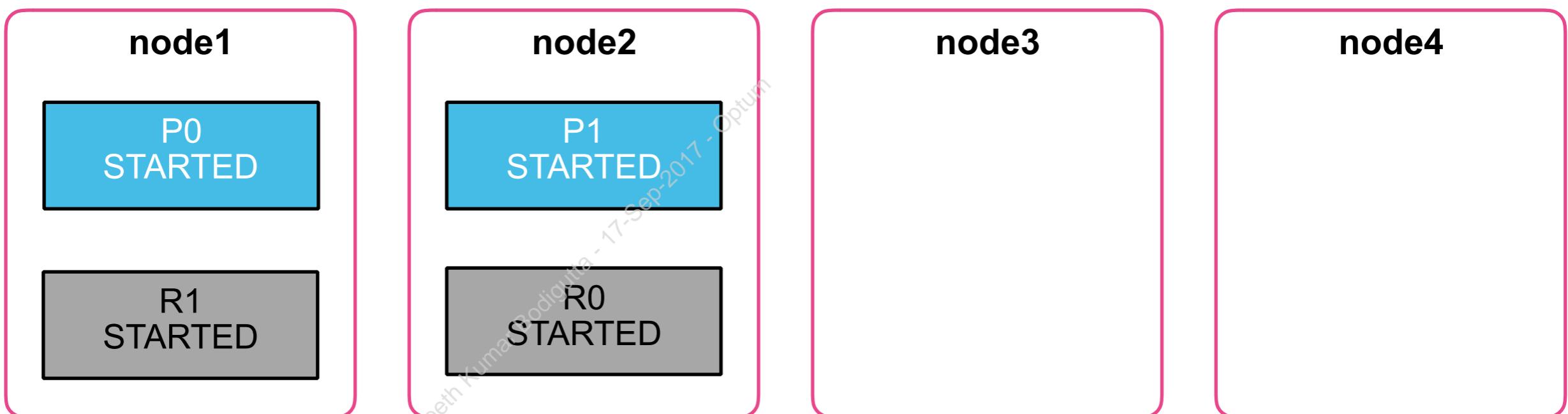
- Eventually, each primary and each replica will move from **INITIALIZING** to **STARTED**
 - the new index is fully allocated and distributed on the cluster



Puneeth Kumar Boddu - 11 Sep 2017 - Optum

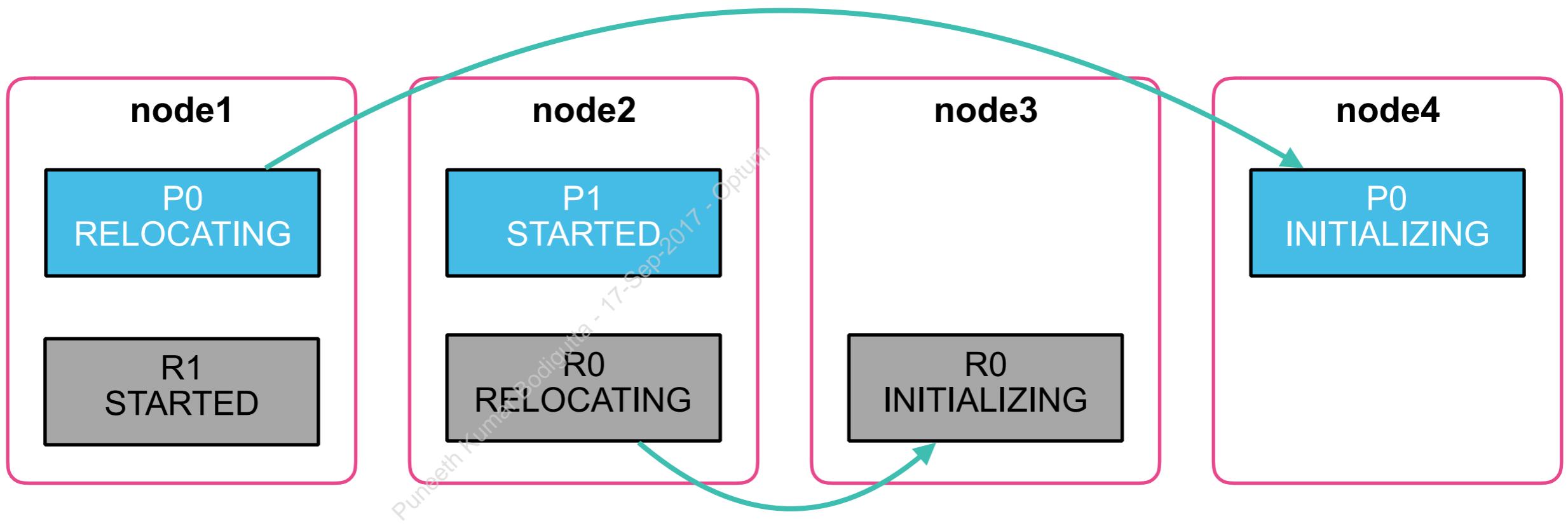
Adding Nodes

- Suppose we add a couple of nodes to our cluster:
 - the master node will attempt to balance the cluster by distributing shards to the new nodes



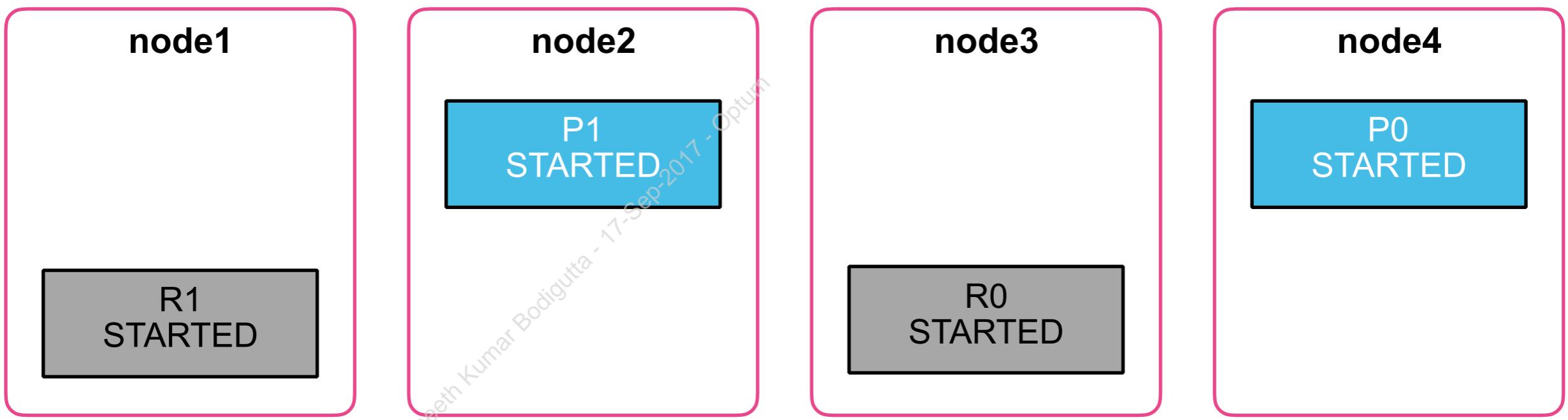
RELOCATING

- A shard that is currently moving from one node to another is in the **RELOCATING** state
 - operations (index, delete, search, etc.) can still be done on the shard during relocation



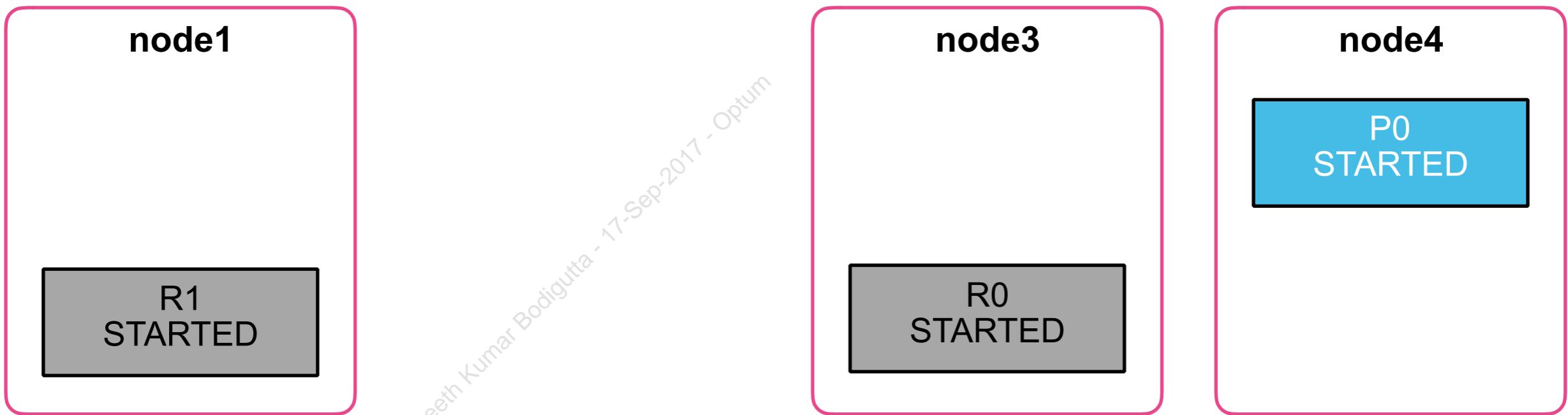
RELOCATING

- After the relocated shard is started, the original shard gets removed:



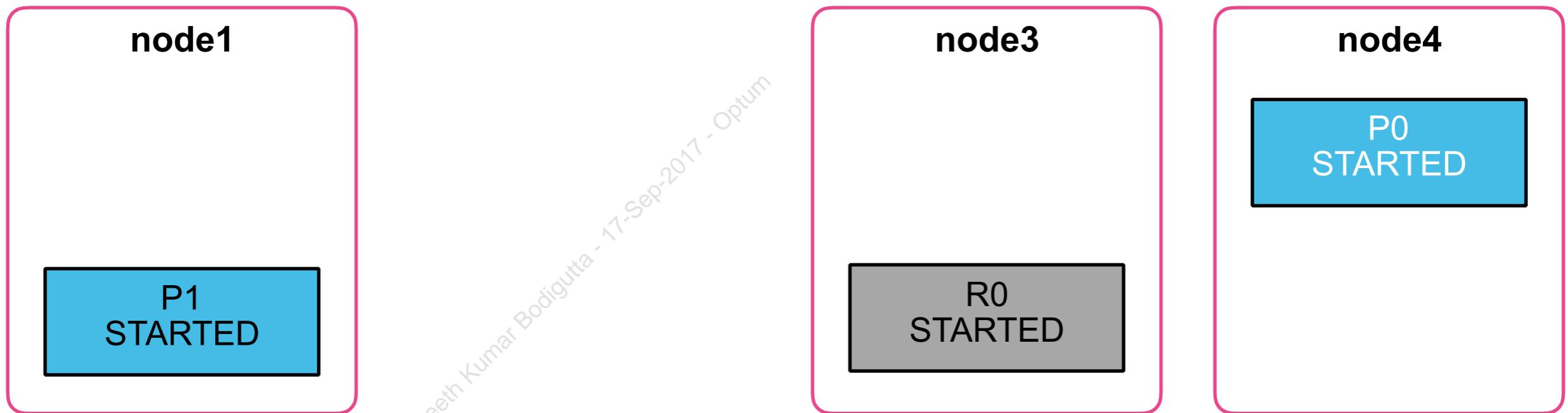
Removing Nodes

- Suppose we remove **node2** from the cluster (or it fails somehow):
 - no data is lost, because **R1** is a copy of **P1**



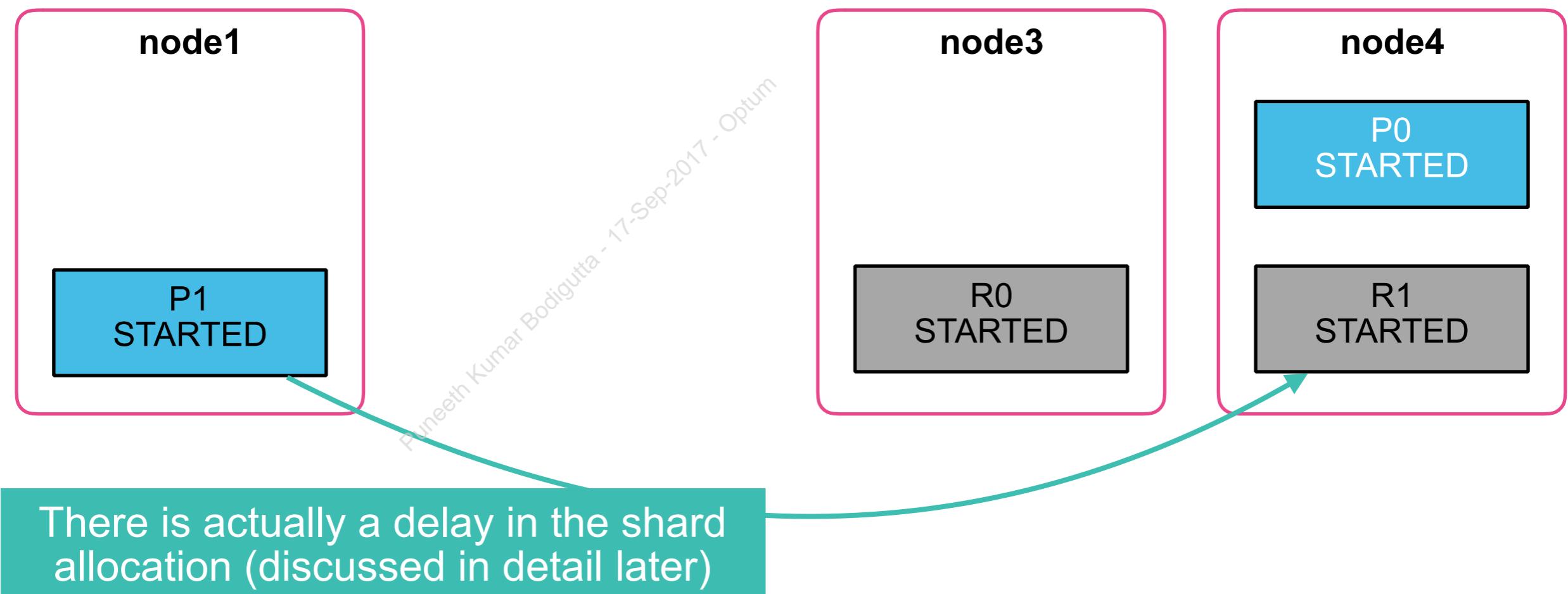
Shard Promotion

- R1 gets promoted to a primary shard, becomes P1:
 - What is the state of the cluster now?



More Replicas

- P1 needs a replica, so the master node allocates a replica shard on either **node3** or **node4**:
 - which will go from **INITIALIZING** to **STARTED**



Diagnosing Shard Issues

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Cluster Allocation Explain API

- Having an **UNASSIGNED** primary shard means your cluster is not in a good state!
 - lots of failures when attempting to index documents, retrieve documents, run queries, etc.
- Elasticsearch provides the **Cluster Allocation API** to help you locate any **UNASSIGNED** shards:
 - also known as the **explain API** for short

GET `_cluster/allocation/explain`

Returns details of the first unassigned shard that it finds

Locating Unassigned Shards

- The response from the ***explain API*** lists unassigned shards
 - and an explanation of why they are unassigned

```
GET _cluster/allocation/explain
```

we either have a node down or improper permission settings

```
{  
  "index": "my_index",  
  "shard": 1,  
  "primary": true,  
  "current_state": "unassigned",  
  "unassigned_info": {  
    "reason": "INDEX_CREATED",  
    "at": "2017-02-02T22:59:36.686Z",  
    "last_allocation_status": "no_attempt"  
  },  
  "can_allocate": "no",  
  "allocate_explanation": "cannot allocate because  
allocation is not permitted to any of the nodes",  
  ...  
}
```

shard 1 is a primary shard that is unassigned

Specific Shard Allocation Details

- You can also request the allocation details of a specific shard in a specific index:

```
GET _cluster/allocation/explain
{
  "index" : "my_index",
  "shard" : 0,
  "primary" : true
}
```

Returns details for just
this particular shard

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- A cluster has a ***health*** that contains various statistics and the status of the cluster
- A cluster's ***health status*** is either green, yellow, or red, depending on the current shard allocation
- ***Shard allocation*** is the process of assigning a shard to a node in the cluster
- A cluster's ***routing table*** contains which nodes are hosting which indices and shards
- The state of a shard is either **UNASSIGNED**, **INITIALIZING**, **STARTED**, or **RELOCATING**
- Elasticsearch provides the ***Cluster Allocation API*** to help you locate any **UNASSIGNED** shards



Quiz

1. Suppose you have a 3-node cluster and a new index gets created with **number_of_replicas** set to 3. What is the “best” health status your cluster will ever be in?
2. **True or False:** A cluster with a yellow status is missing indexed documents.
3. As a shard is allocated to a node, its state goes from **UNASSIGNED** to _____.
4. If your cluster has an unassigned primary shard, what is the cluster’s health status?
5. When does the following request get a response?

```
GET _cluster/health?wait_for_status=yellow&timeout=10s
```

Lab 5

Cluster Health

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Chapter 6

Mappings and Analysis

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Text Analysis
- Mappings
- Defining a Mapping
- Searching Data
- Distributed Search
- Segments
- Merging Segments

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

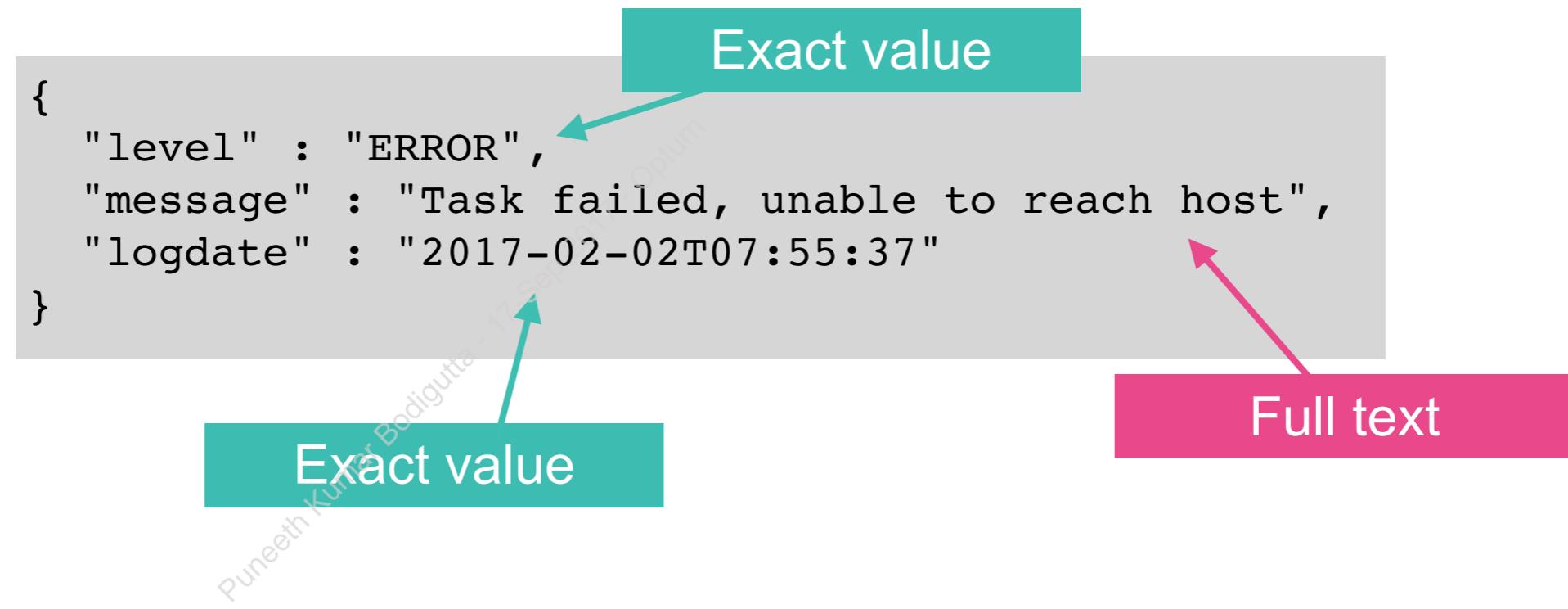


Text Analysis

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Exact Values vs. Full Text

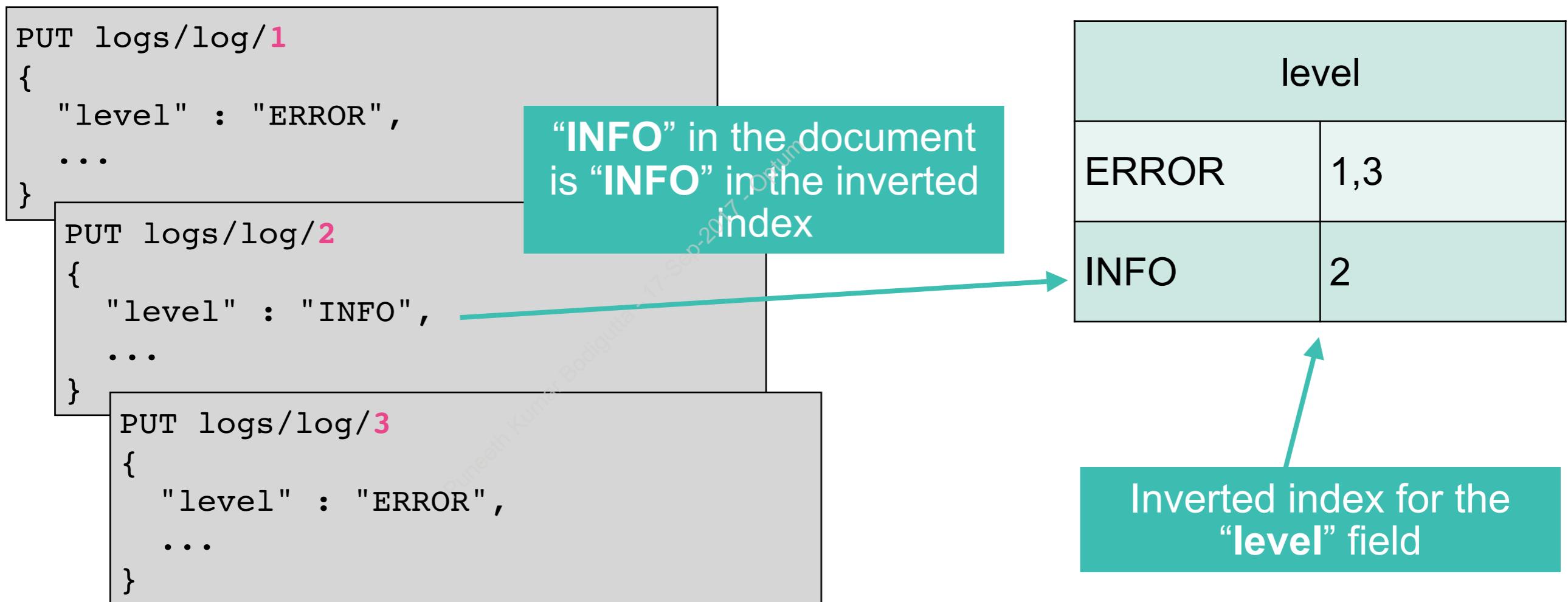
- In general, data in Elasticsearch can be categorized into two groups:
 - Exact values:** includes numeric values, dates, and certain strings
 - Full text:** unstructured text data that we want to search



- Why does it matter if a value is **exact** or **full text**?

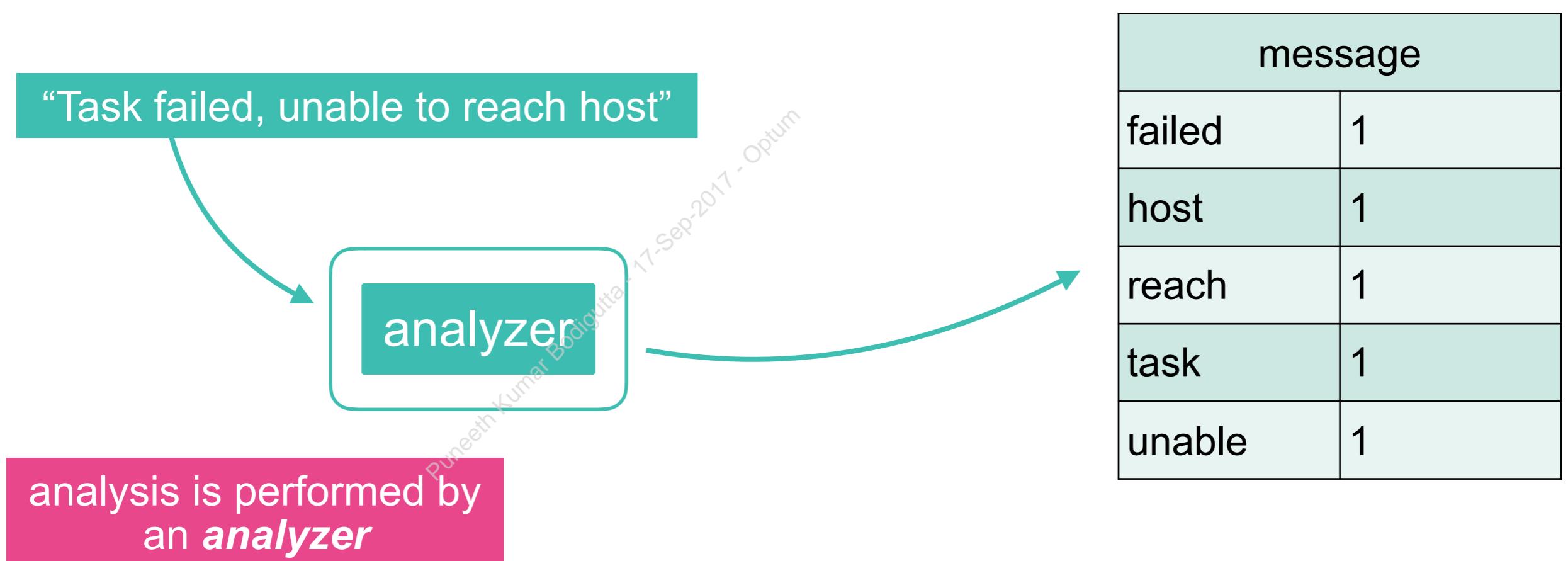
Exact Values are Not Analyzed

- An *inverted index* is built for each *field* of a document
 - the **exact values** of a document become terms in the index (without any changes made to the values of the field)



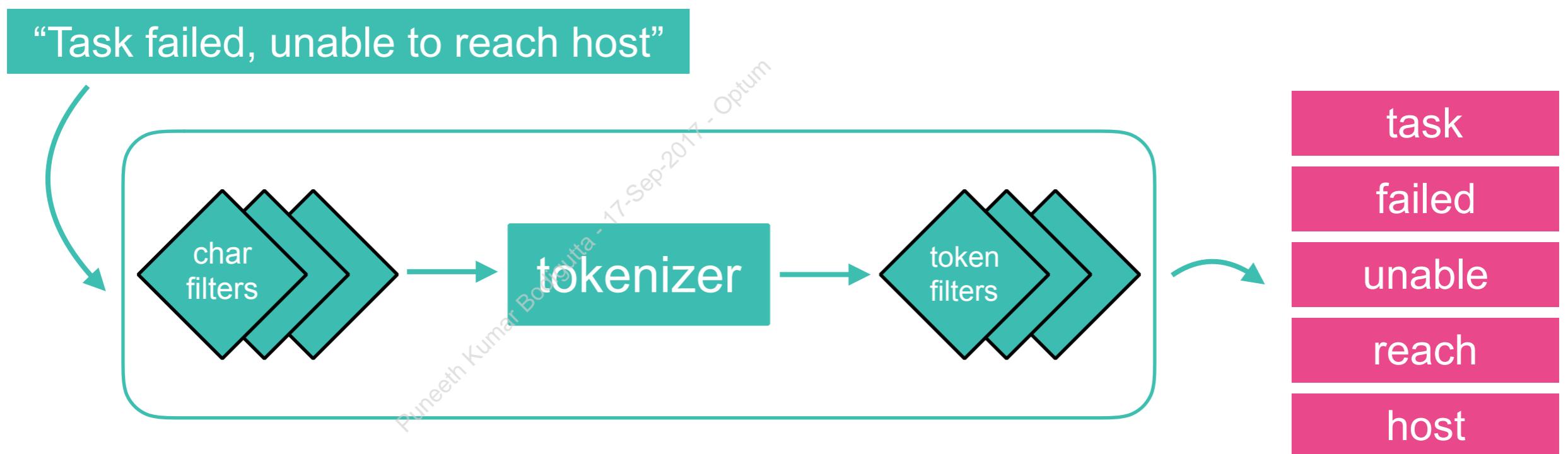
Full Text is Analyzed

- *Analysis* is the process of converting full text into *terms* for the inverted index
- The strings in a “text” field of a mapping get analyzed before being added to that field’s inverted index



Anatomy of an Analyzer

- An analyzer consists of three parts:
 1. Character Filters
 2. Tokenizer
 3. Token Filters



Built-in Analyzers

- **standard**
 - the default analyzer, it splits text on word boundaries, removes punctuation, and lowercases all terms
- **simple**
 - splits text on anything that is not a letter and lowercases
- **whitespace**
 - splits text on whitespace (no lowercasing or punctuation removed)
- View the other built-in analyzers at
<https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-analyzers.html>



Mappings

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

What is a Mapping?

- Elasticsearch will happily index any document without knowing its details (number of fields, their data types, etc.)
 - However, behind-the-scenes Elasticsearch assigns data types to your fields in a *mapping*
- A *mapping* is a **schema definition** that contains:
 - names of fields
 - data types of fields
 - how the field should be indexed and stored by Lucene

Puneeth Kumar Bodigutta - 17562017 - Optum



Field Data Types

- **Simple Types, including:**
 - **text**: for full text (analyzed) strings
 - **keyword**: for exact value strings
 - **date**: string formatted as dates, or numeric dates
 - integer types: like **byte**, **short**, **integer**, **long**
 - floating-point numbers: **float**, **double**, **half_float**, **scaled_float**
 - **boolean**
 - **ip**: for IPv4 or IPv6 addresses
- **Hierarchical Types**: like **object** and **nested**
- **Specialized Types**: **geo_point**, **geo_shape** and **percolator**

This list is not every data type available

“string” in older versions
of Elasticsearch

A Simple Example

- Let's index a document into **logs**
 - Because it is the first document indexed, Elasticsearch will define a new mapping based on the field names and data types of each field/value pair

The document type becomes
the **name** of the mapping

```
PUT logs/log/1
{
  "level" : "ERROR",
  "message" : "Unable to reach host",
  "logdate" : "2017-02-02T07:55:37"
}
```

The fields become “**properties**”
in the mapping

Elasticsearch will “guess” the
data types of your values

A Simple Example

GET logs/_mappings

“level” and “message”
both got mapped as
“text” and “keyword”

“logdate” got mapped
as a **“date”**

```
{  
  "logs": {  
    "mappings": {  
      "log": {  
        "properties": {  
          "level": {  
            "type": "text",  
            "fields": {  
              "keyword": {  
                "type": "keyword",  
                "ignore_above": 256  
              }  
            }  
          },  
          "logdate": {  
            "type": "date"  
          },  
          "message": {  
            "type": "text",  
            "fields": {  
              "keyword": {  
                "type": "keyword",  
                "ignore_above": 256  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

Puneeth Kumar Bodugutta - 17-Sep-2017 - Optimized

Defining a Mapping

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Defining a Mapping

- Often you need to define your own mapping
 - Elasticsearch may not “guess” your data types properly
 - you may want to customize settings
 - you can optimize disk space (especially with text fields)
 - you may want to customize text analysis (discussed later in the chapter)
- Add a “**mappings**” section when creating a new index:

```
PUT logs2
{
  "mappings": {
    ...
  }
}
```

Example of a Mapping

```
PUT logs2
{
  "mappings": {
    "log": {
      "properties": {
        "level": {
          "type": "keyword"
        },
        "logdate": {
          "type": "date"
        },
        "response": {
          "type": "long"
        },
        "message": {
          "type": "text"
        }
      }
    }
  }
}
```

“**level**” is a finite list of values, so “**keyword**” works well in that scenario

“**message**” is a lot of random text, so “**text**” works well in that scenario

Can you change a mapping?

- **No** - not without reindexing your documents
 - You *can* add fields and modify a few properties, but...
 - ...all other schema changes require a reindexing
- **Why not?**
 - If you switch a field's data type, all existing documents with that field already indexed would become unsearchable on that field
- **Why can I add fields without reindexing?**
 - Adding a field has no effect on the existing indexed fields or documents
 - The index can freely grow, but indexed fields can not change data types

Puneet Kumar Bodiguttapalli Sep-2017 Optum

Searching Data

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Searching Data

- Once your data is indexed, you typically want to search it!
 - We have seen how to send requests from the Kibana Console
 - You can run search queries from the Console as well

The screenshot shows the Kibana Dev Tools Console interface. On the left, there's a sidebar with various navigation links like Discover, Visualize, Dashboard, etc. The main area has tabs for 'Console' and 'Search Profiler', with 'Console' selected. In the console area, there's a code editor containing a bulk indexing request:

```
1 POST logs2/_bulk
2 { "index" : { "_id" : "1"}}
3 { "level" : "WARN", "message" : "Something is not right here", "logdate" : "2017-02-02T07:57:49"}
4 { "index" : { "_id" : "2"}}
5 { "level" : "ERROR", "message" : "Disconnected from host", "logdate" : "2017-02-02T07:59:12"}
6 { "index" : { "_id" : "3"}}
7 { "level" : "INFO", "message" : "Reconnected to host", "logdate" : "2017-02-02T08:01:37"}
```

A pink callout box with the text "Let's start by indexing a few documents into logs2" is overlaid on the console area, with a pink arrow pointing towards the code. To the right of the code editor is the response pane, which displays the results of the bulk operation:

```
1 {
2   "took": 3,
3   "errors": false,
4   "items": [
5     {
6       "index": {
7         "_index": "logs2",
8         "_type": "log",
9         "_id": "1",
10        "_version": 4,
11        "result": "updated",
12        "_shards": {
13          "total": 2,
14          "successful": 1,
15          "failed": 0
16        },
17        "created": false,
18        "status": 200
19      },
20      {
21        "index": {
22          "_index": "logs2",
23          "_type": "log",
24          "_id": "2",
25          "_version": 4,
26          "result": "updated",
27          "_shards": {
28            "total": 2,
29            "successful": 1,
30            "failed": 0
31          }
32        }
33      }
34    ]
35  }
```

The match_all Query

- **match_all** is like a “SELECT * FROM table”

Console Search Profiler

```
1 GET logs2/_search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
```

Puneeth Kumar Bodigutta - 17.09.2017, Optimized

```
1 {
2   "took": 0,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "failed": 0
8   },
9   "hits": {
10    "total": 3,
11    "max_score": 1,
12    "hits": [
13      {
14        "_index": "logs2",
15        "_type": "log",
16        "_id": "2",
17        "_score": 1,
18        "_source": {
19          "level": "ERROR",
20          "message": "Disconnected from host",
21          "logdate": "2017-02-02T07:59:12"
22        }
23      },
24      {
25        "_index": "logs2",
26        "_type": "log",
27        "_id": "1",
28        "_score": 1,
29        "_source": {
30          "level": "WARN",
31          "message": "Something is not right here"
32        }
33      }
34    ]
35  }
36}
```

The match Query

- match is like a “SELECT * FROM table where col=value”

Console Search Profiler

```
1 GET logs2/_search
2 {
3   "query": {
4     "match_all": {}
5   }
6 }
7
8 GET logs2/_search
9 {
10  "query": {
11    "match": {
12      "message": "host"
13    }
14  }
15 }
```

Puneeth Kumar Bodigutta - 11-Sep-2017 - Optum

```
9 ,
10   "hits": {
11     "total": 2,
12     "max_score": 0.25316024,
13     "hits": [
14       {
15       "_index": "logs2",
16       "_type": "log",
17       "_id": "2",
18       "_score": 0.25316024,
19       "_source": {
20         "level": "ERROR",
21         "message": "Disconnected from host",
22         "logdate": "2017-02-02T07:59:12"
23       }
24     },
25     {
26       "_index": "logs2",
27       "_type": "log",
28       "_id": "3",
29       "_score": 0.25316024,
30       "_source": {
31         "level": "INFO",
32         "message": "Reconnected to host",
33         "logdate": "2017-02-02T08:01:37"
34     }
35   ]
}
```



Lots of Search Queries

- The Elasticsearch query language has lots of queries for many types of searches:
 - <https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html>

Console Search Profiler

```
16  
17 GET logs2/_search  
18 {  
19   "query": {  
20     "bool": {  
21       "must": [  
22         {  
23           "match": {  
24             "message": "host"  
25           }  
26         }  
27       ],  
28       "filter": {  
29         "term" : {  
30           "level" : "ERROR"  
31         }  
32       }  
33     }  
34   }  
35 }  
36  
37  
38  
39  
40  
41  
42
```

Find ERROR events that contain “host” in the message.

```
1 {  
2   "took": 0,  
3   "timed_out": false,  
4   "_shards": {  
5     "total": 5,  
6     "successful": 5,  
7     "failed": 0  
8   },  
9   "hits": {  
10    "total": 1,  
11    "max_score": 0.25316024,  
12    "hits": [  
13      {  
14        "_index": "logs2",  
15        "_type": "log",  
16        "_id": "2",  
17        "_score": 0.25316024,  
18        "_source": {  
19          "level": "ERROR",  
20          "message": "Disconnected from host",  
21          "logdate": "2017-02-02T07:59:12"  
22        }  
23      }  
24    ]  
25  }  
26 }
```

Aggregations

Search criteria find hits that match our search parameters

Search

Destination/Hotel Name:
San Francisco

Work Leisure

Check-in
Check-in Date

Check-out
Check-out Date

Rooms 1
Adults 2
Children 0

Search

Filter by:

▼ Star Rating

<input type="checkbox"/> No preference	454
<input type="checkbox"/> 1 star	16
<input type="checkbox"/> 2 stars	80
<input type="checkbox"/> 3 stars	72
<input type="checkbox"/> 4 stars	55
<input type="checkbox"/> 5 stars	9
<input type="checkbox"/> Unrated	222

▼ Accommodation Type

<input type="checkbox"/> Apartments	186
<input type="checkbox"/> Hotels	175
<input type="checkbox"/> Motels	39
<input type="checkbox"/> Vacation Homes	21
<input type="checkbox"/> Hostels	18
<input type="checkbox"/> Inns	9
<input type="checkbox"/> Bed and Breakfasts	6

San Francisco: 454 properties found
3 Reasons to Visit: Golden Gate Bridge, Fisherman's Wharf & cable cars

Check out the best of San Francisco before you find a place to stay...

Lock in a great price for your stay on these dates:

Jul 27 — Jul 28	Jul 28 — Jul 29	Jul 29 — Jul 30	Jul 30 — Jul 31
-----------------	-----------------	-----------------	-----------------

Our Top Picks First Stars Distance From Downtown Review Score

Hotel Nikko San Francisco ★★★★ Excellent 8.6
Union Square, San Francisco – Subway Access
Just a 5-minute walk from Union Square, Hotel Nikko San Francisco features an on-site restaurant. Pillow-top bedding and a flat-screen TV are provided in all rooms.
Booked 62 times today



Show prices 

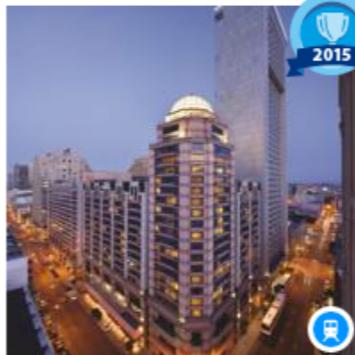
Hotel Zephyr San Francisco ★★★★ Very good 8.3 WiFi 8.9
Fisherman's Wharf, San Francisco
Offering a fitness center, Hotel Zephyr San Francisco is located a short 330 yards from Pier 39 Fisherman's Wharf. Free WiFi access is available in this waterfront hotel.
Booked 65 times today

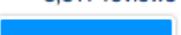


Luxury Deal 

Show prices 

Hilton San Francisco Union Square ★★★★ Good 7.6
Union Square, San Francisco – Subway Access
Located in the heart of downtown San Francisco and just a 5-minute walk from the Powell Street subway station, this Hilton features an on-site Herb n' Kitchen and a beautiful courtyard pool.
Booked 25 times today



Show prices 

Example of an Aggregation

Console Search Profiler

```
36
37 GET logs2/_search
38 {
39   "size": 0,
40   "aggs": {
41     "logs_by_level": {
42       "terms": {
43         "field": "level",
44         "size": 10
45       }
46     }
47   }
48 }
```

*How many log events
of each type have
occurred?*

Puneeth Kumar Bodiguttla - 17-Sep-2017 - Optum

```
1 {
2   "took": 0,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "failed": 0
8   },
9   "hits": {
10    "total": 3,
11    "max_score": 0,
12    "hits": []
13 },
14   "aggregations": {
15     "logs_by_level": {
16       "doc_count_error_upper_bound": 0,
17       "sum_other_doc_count": 0,
18       "buckets": [
19         {
20           "key": "ERROR",
21           "doc_count": 1
22         },
23         {
24           "key": "INFO",
25           "doc_count": 1
26         },
27         {
28           "key": "WARN",
29           "doc_count": 1
30         }
31       ]
32     }
33 }
```

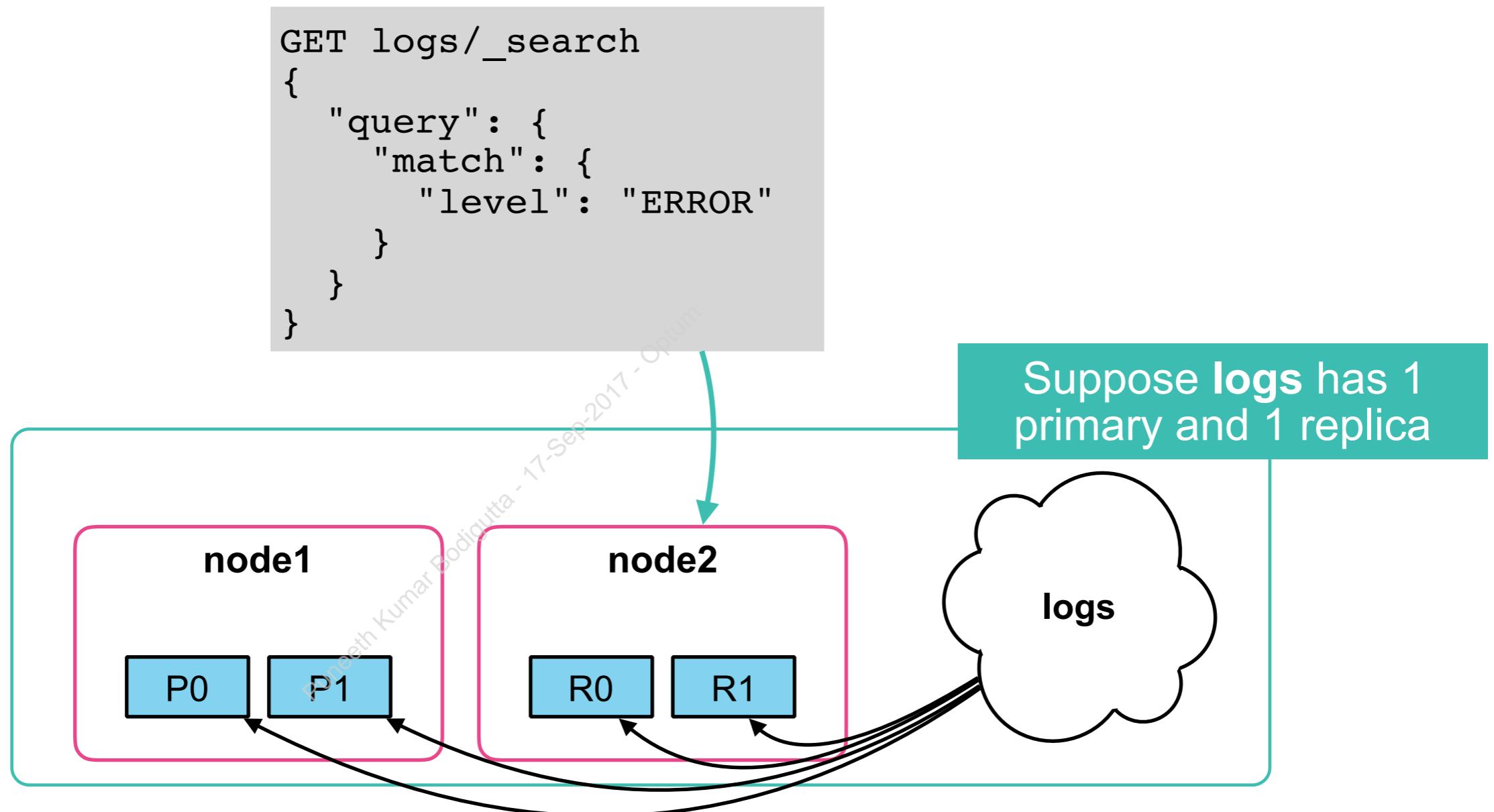


Distributed Search

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

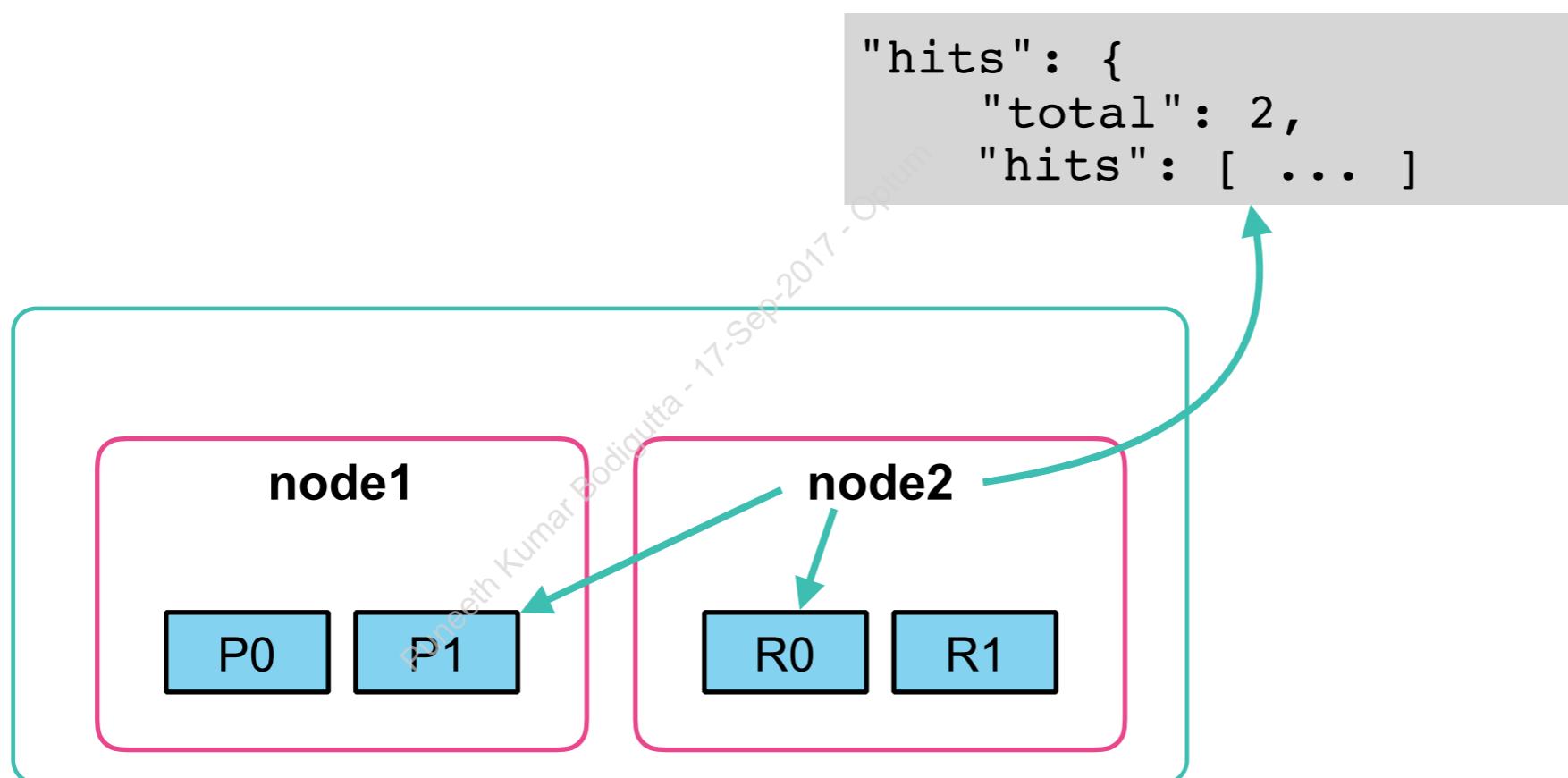
How Distributed Search Works

- A query is submitted to a node in the cluster
 - referred to as the *coordinating node* for that request



How Distributed Search Works

- The coordinating node sends the request to each shard of the index (either a primary or replica)
 - it also determines the top hits, retrieves the documents, and sends back the response to the client application



Segments

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Inverted Index in Practice

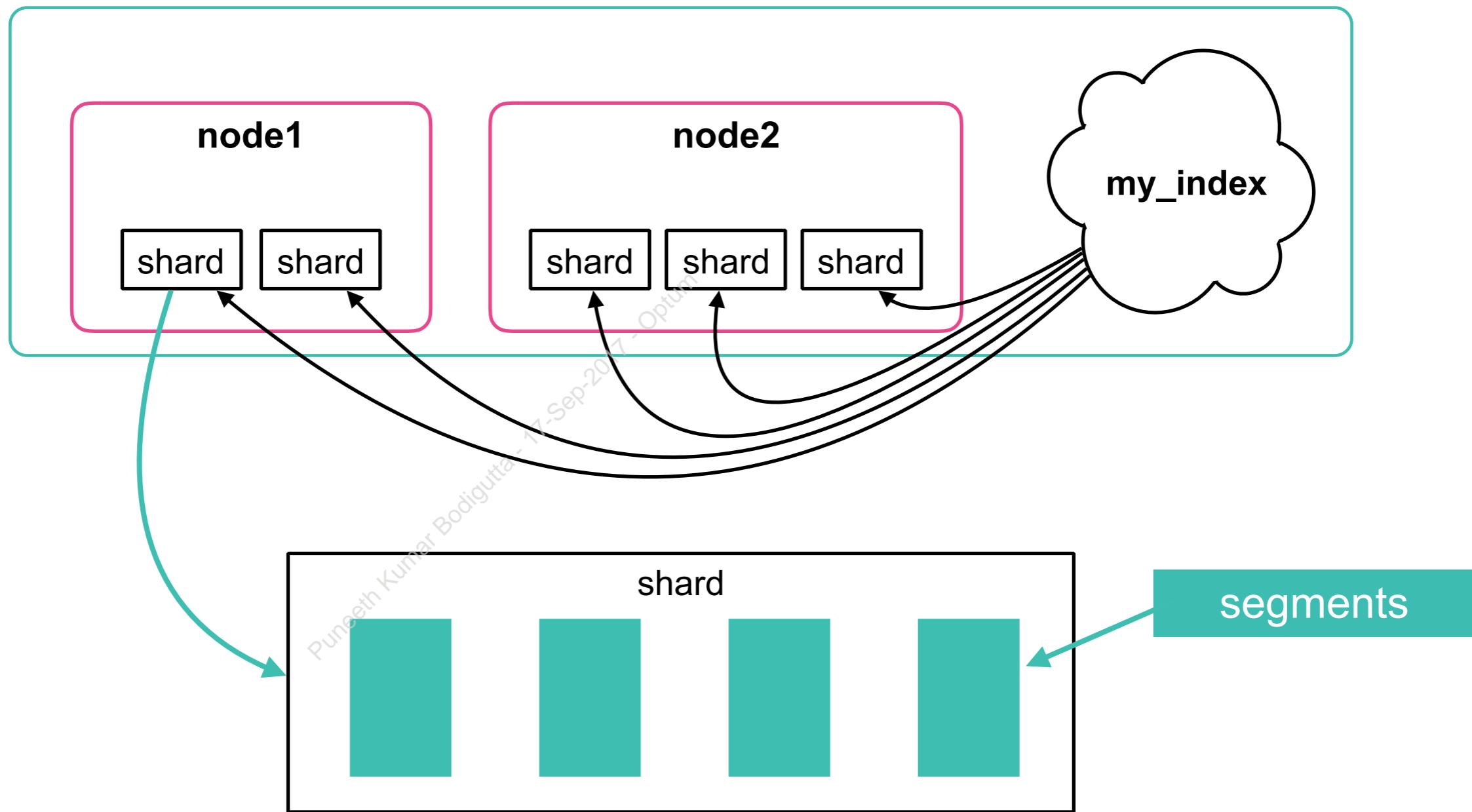
- The inverted index concept discussed in this section is practically represented inside a package called a Lucene **segment**
 - a segment is a self-contained immutable inverted index

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



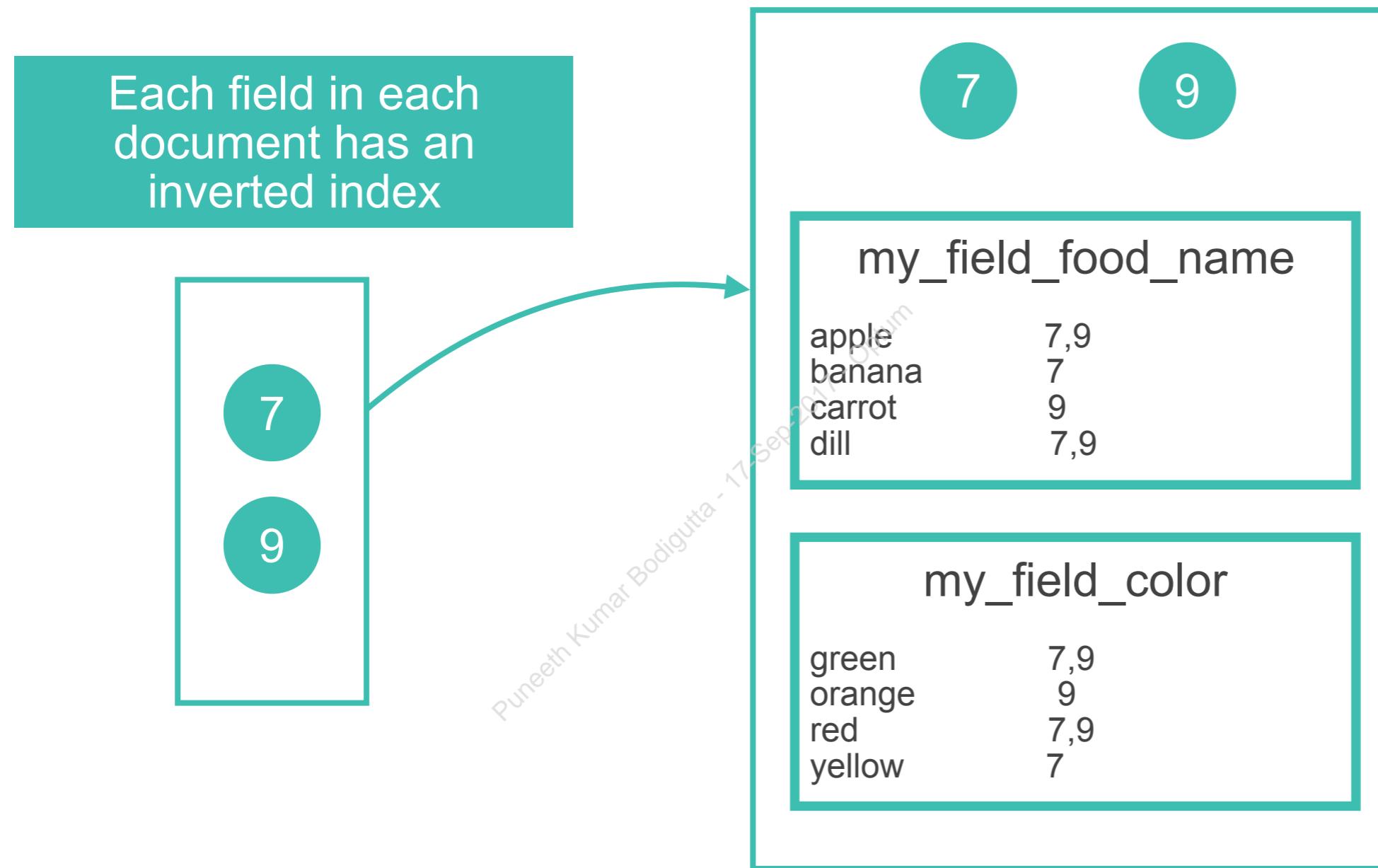
Index -> Shard -> Segments

- An index consists of one or more shards
- A shard consists of segments



Segment is a Package

- A **segment** is a package of many data structures
 - including an inverted index for each field that is searchable



Indexing Documents

time

index a doc

buffer

index a doc

buffer

index a doc

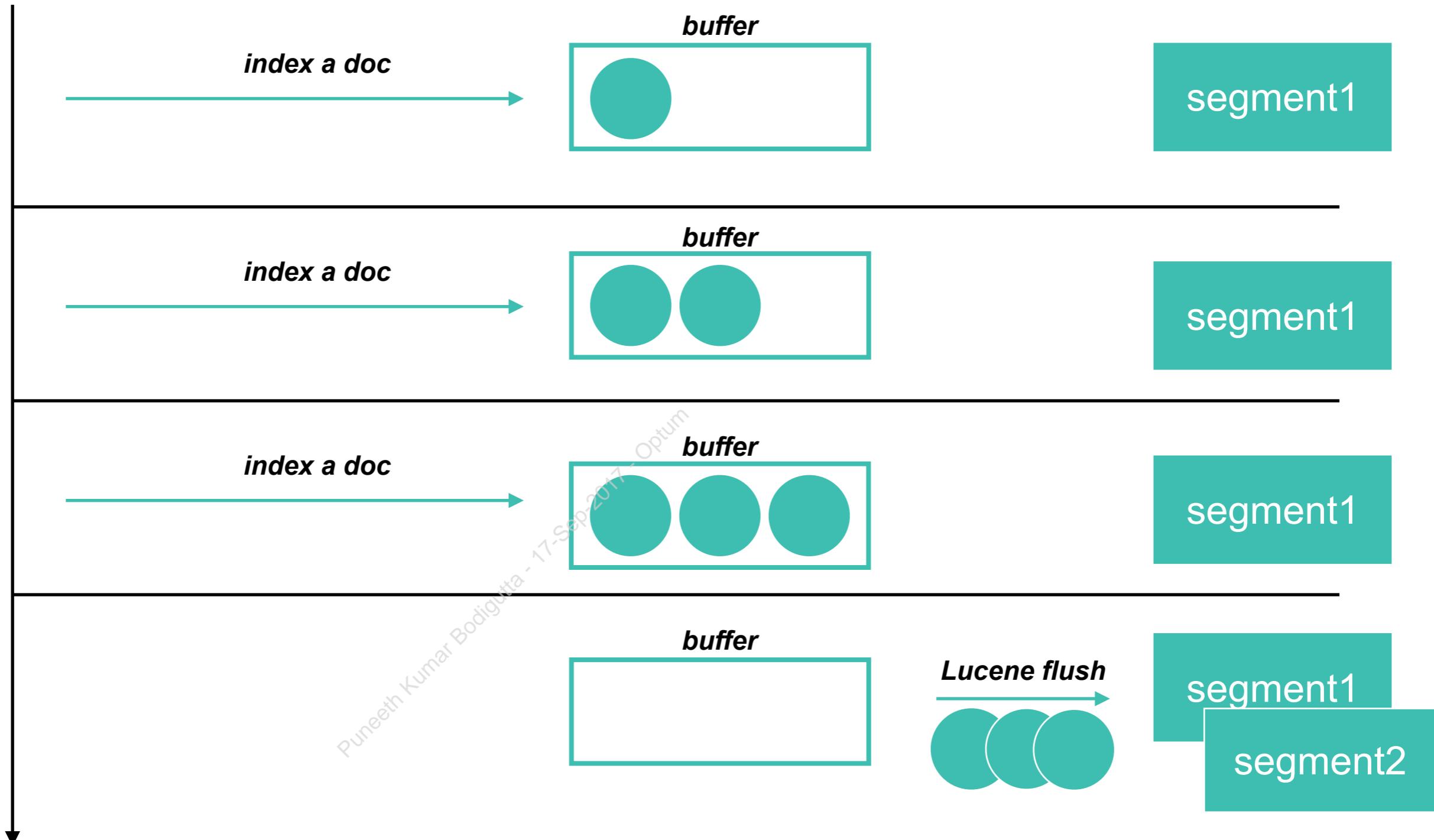
buffer

Lucene flush

segment1

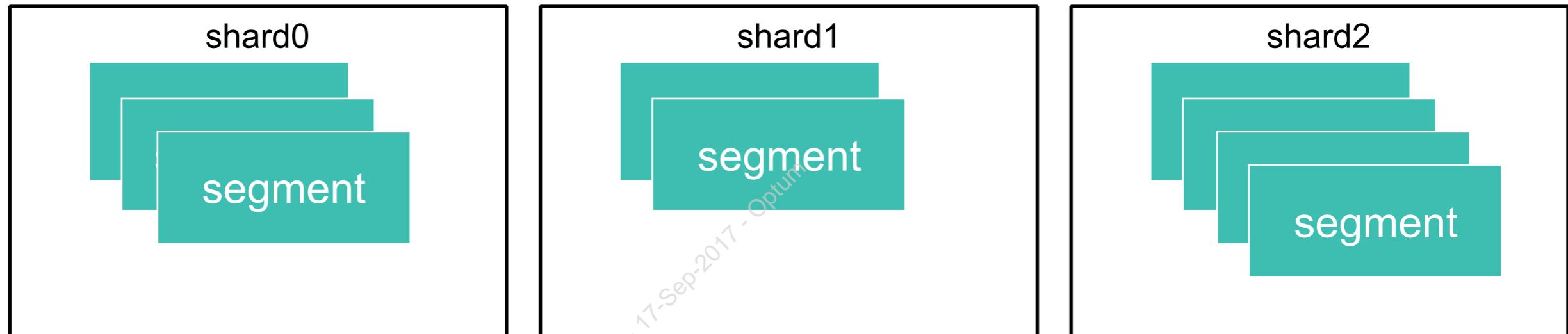
Indexing Documents

time

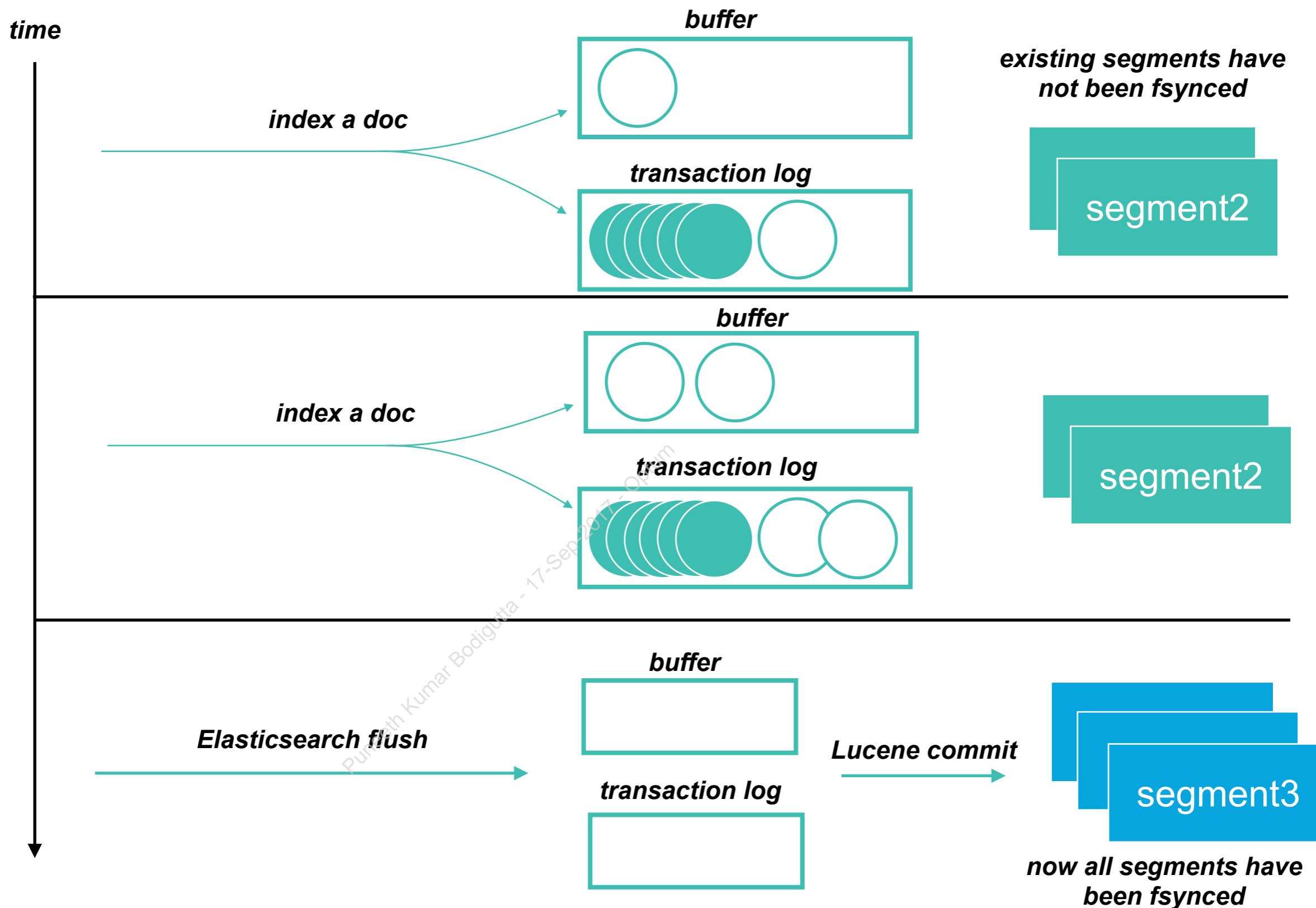


How often are new segments created?

- Very often! The default value for `index.refresh_interval` is 1 second
 - During heavy indexing, you will get lots of segments



The Transaction Log



The Transaction Log

- Segments are fsynced to disk during a *Lucene commit*:
 - a relatively heavy operation
 - should **not** be performed after every index or delete operation
- Until a commit, your data is susceptible to loss
- To prevent this data loss, each shard has a *transaction log*
 - Any index or delete operation is written to the transaction log after being processed by the internal Lucene index
- In the event of a crash or hardware failure, recent transactions can be replayed from the transaction log when the shard recovers

Puneeth Kharbanda - 17-Sep-2017



Configuring the Refresh Interval

- You can configure how often a refresh operation occurs on an index:
 - Suppose you are going to be indexing a lot of documents over a short period of time:

```
PUT my_index/_settings
{
  "refresh_interval": "60s"
}
```

Change refresh interval to
60 seconds for **my_index**

- When you are done indexing, set **refresh_interval** back to its default value:

```
PUT my_index/_settings
{
  "refresh_interval": "1s"
}
```

Change refresh interval
back to 1 second



The Flush API

- You can force an Elasticsearch flush:

```
POST my_index/_flush
```

- Do you need to call _flush?
 - Typically, no!
 - Flush happens automatically depending on how many operations get added to the transaction log, how big they are, and when the last flush happened

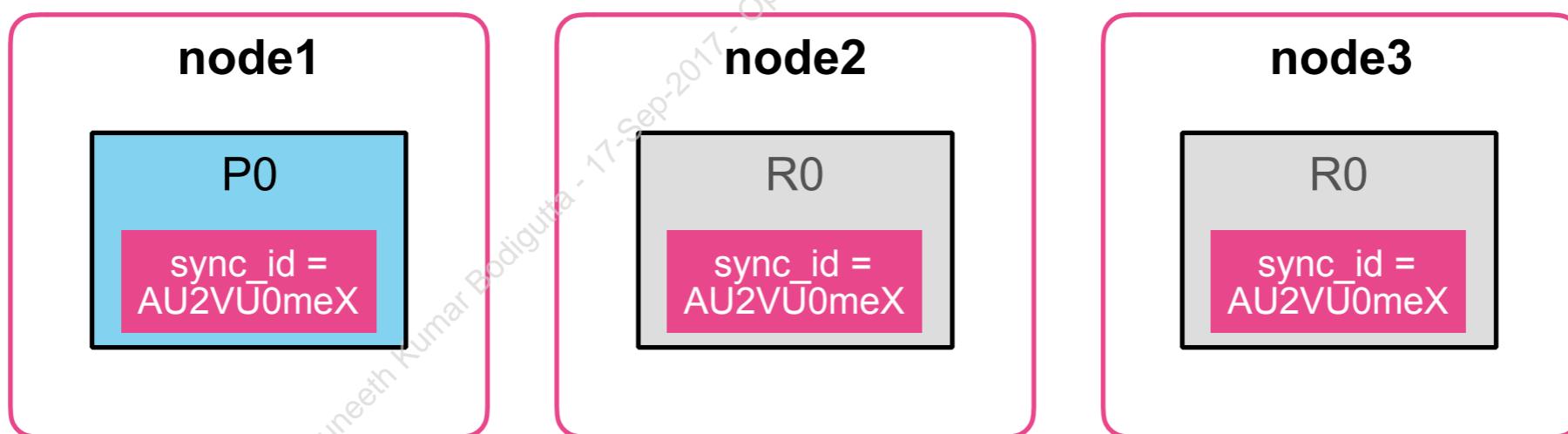
Puneeth Kumar Bodigutta - 17-Sep-2017



Synced Flush

- What is a *synced flush*?
 - A synced flush performs a normal flush, then adds a generated unique marker (`sync_id`) to all shards
 - `sync_id` provides a quick way to check if two shards are identical

```
POST my_index/_flush/synced
```



This primary shard and its two replicas are in sync

Merging Segments

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Merging Segments

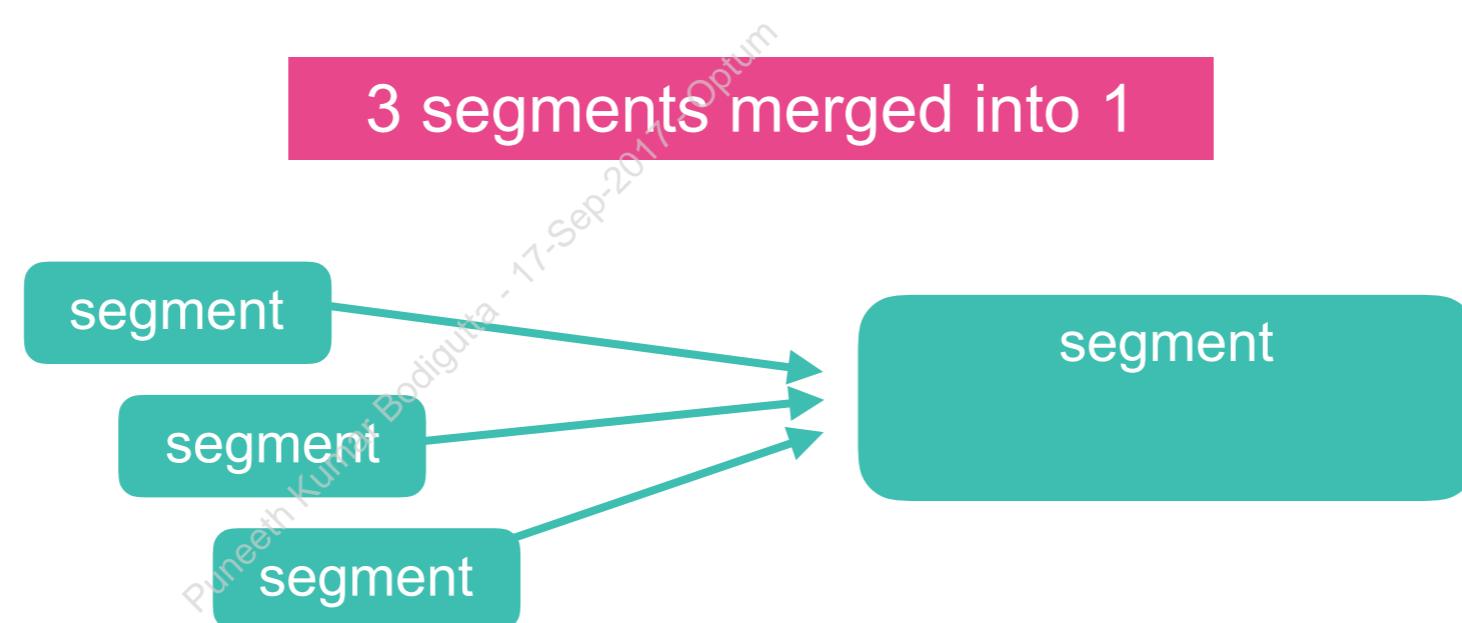
- You can have too many segments
 - A full query of an index has to hit every segment, so too many segments can decrease performance
- Also, segments are immutable, but you can **DELETE** a document in Elasticsearch
 - The document is not immediately deleted from the underlying segment though

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Merging Segments

- Every once in a while the segments need to be *merged*
 - to lower the number of segments
 - and to clean up deleted documents
- Smaller segments are periodically *merged* into larger segments:



Merging with Indexing Only



- <http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html>

Merging with Updates



- <http://blog.mikemccandless.com/2011/02/visualizing-lucenes-segment-merges.html>

The force merge API

- In general, Elasticsearch and Lucene do a good job of deciding when to merge segments
- However, you can force an index to merge its segments:

```
POST my_index/_forcemerge
```

- Only use `_forcemerge` on indices that will never have write operations executed in the future

Puneeth Kumar Bodigutta - 17-Sep-2017



Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- **Analysis** is the process of converting full text into **terms** for the inverted index
- A **mapping** is a schema definition that contains names and data types of fields
- Elasticsearch dynamically creates and updates mappings based on your documents
- Define your own mapping by adding a “**mappings**” section when creating a new index
- Use the **_search** endpoint to query documents in indices
- A Lucene **segment** is a package of many data structures
- Segments in Elasticsearch are **flushed**, while in Lucene they are **merged**.



Quiz

1. Assuming the mapping is created by Elasticsearch dynamically, what data type would be assigned to a field whose value is
 1. 582
 2. “I love this restaurant.”
 3. “2017-02-28”
2. The process of converting full text into terms for the inverted index is called _____
3. Why does it matter if a field is an exact value vs. full text?
4. **True or False:** Use the `_search` endpoint for both “query” and “aggs” requests.
5. The node that handles a client request is referred to as the _____ node.

6. What is something that could cause a long startup time for a node after it is restarted, and how could we prevent it?

Puneet Kumar Bodiguttula 1-Sep-2017 Optum

Lab 6

Mappings and Analysis

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optimized

Chapter 7

Index Management

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- The Reindex API
- Closing an Index
- Delayed Shard Allocation
- Total Shards Per Node
- Hot/Warm Architecture
- Shard Filtering

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

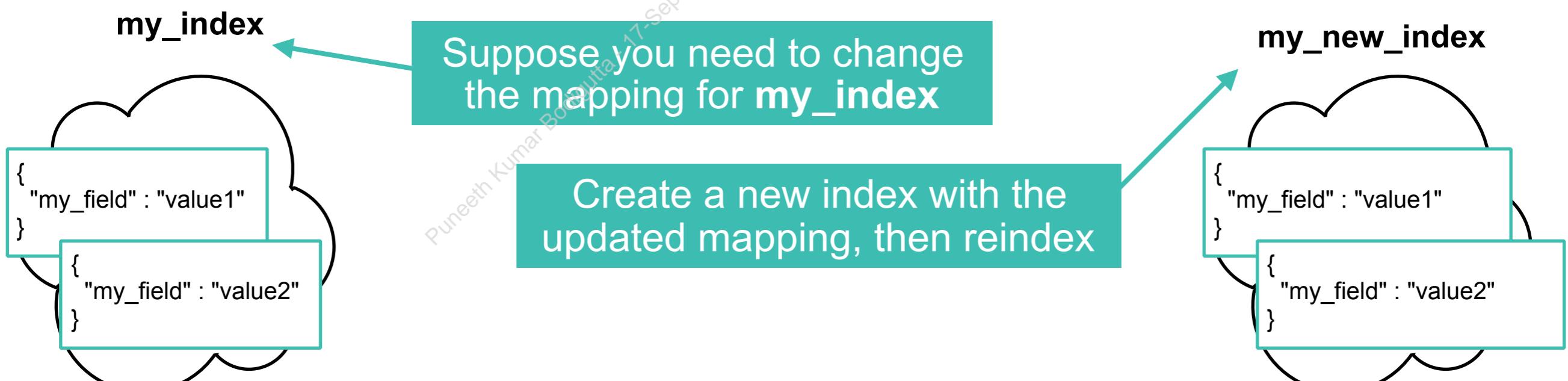


The Reindex API

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

The Reindex API

- You can copy documents from one index to another using the *Reindex API*
- Why reindex?
 - change the number of shards
 - change mappings for fields
 - upgrade index file format
 - use a script to change documents during reindexing



Using _reindex

- The `_reindex` command indexes the documents from “source” into the “dest” index

```
POST _reindex
{
  "source" : {
    "index" : "my_index"
  },
  "dest" : {
    "index" : "my_other_index"
  }
}
```

Puneeth Kumar Bedam
@puneethkb

All the documents from “my_index” are indexed into “my_other_index”



Dealing with Versions

- Different options for dealing with document versions
- By default, `_reindex` blindly dumps documents into the target, ***overwriting docs*** that have the same type and id
 - equivalent to setting “`version_type`” to “`internal`”

```
POST _reindex
{
  "source" : {
    "index" : "my_index"
  },
  "dest" : {
    "index" : "my_other_index",
    "version_type" : "internal"
  }
}
```

overwrites any existing
target docs with the same
type and id

“external” Version Type

- Setting “`version_type`” to “`external`” will:
 - preserve the **version** from the source,
 - create any documents that are missing,
 - update any documents that have an older version in the destination index than they do in the source index

```
POST _reindex
{
  "source" : {
    "index" : "my_index"
  },
  "dest" : {
    "index" : "my_other_index",
    "version_type" : "external"
  }
}
```

Version Conflicts

- A version conflict can occur when using “**external**”
- Set “**conflicts**” equal to “**proceed**” to ignore conflicts
 - keeps a running count of them instead

```
POST _reindex
{
  "conflicts" : "proceed",
  "source" : {
    "index" : "my_index"
  },
  "dest" : {
    "index" : "my_other_index",
    "version_type" : "external"
  }
}
```



```
{
  "took": 5,
  "timed_out": false,
  "total": 2,
  "updated": 0,
  "created": 0,
  "deleted": 0,
  "batches": 1,
  "version_conflicts": 1,
  "noops": 0,
  ...
}
```

“create” op_type

- Setting “op_type” to “create” is similar to “external”, except:
 - `_reindex` will only create missing documents in the target index
 - all existing documents will cause a version conflict

```
POST _reindex
{
  "source" : {
    "index" : "my_index"
  },
  "dest" : {
    "index" : "my_other_index",
    "op_type" : "create"
  }
}
```



Multiple Sources

- The “**index**” in “**source**” can be a list of index names:
 - allows you to reindex documents from multiple indices into a single destination index

```
POST _reindex
{
  "source" : {
    "index" : ["logs-2017-01","logs-2017-02"]
  },
  "dest" : {
    "index" : "logs-2017"
  }
}
```

Puneeth Kumar Boddu
Elasticsearch 2017



Reindex from a Remote Cluster

- `_reindex` supports reindexing from a remote Elasticsearch cluster:

```
POST _reindex
{
  "source": {
    "index": "my_index",
    "remote": {
      "host": "http://otherhost:9200",
      "username": "USERNAME",
      "password": "PASSWORD"
    }
  },
  "dest": {
    "index": "my_other_index"
  }
}
```

Configure the remote cluster
in a “remote” section

Closing an Index

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Closing an Index

- Suppose you want to keep the data of an index, but you are not going to query it anymore
 - You can **close** an index using the **Open/Close Index API**
 - The data stays on disk and a part of the cluster state
- What are the benefits?
 - a closed index has ***almost no overhead*** on the cluster
 - no CPU resources
 - no RAM utilization
 - faster than restoring from backup external to cluster
- If closed, an index is not replicated (backup before closing!)

Nithineth Kumar Kodigutta - 17-Sep-2017 - Optum



The `_close` Endpoint

- Use the `_close` endpoint to close an index:
 - perform a synced flush

```
POST tweets-yesterday/_flush/synced  
POST tweets-yesterday/_close
```

- A closed index is not available for search:

```
GET tweets-yesterday/_search
```

Puneeth Kumar Boddu / Elasticsearch 2017 Optim

```
{  
  "error": {  
    "root_cause": [  
      {  
        "type": "index_closed_exception",  
        "reason": "closed",  
        "index_uuid": "x1EpB_3YSri9yB20gr5uHw",  
        "index": "tweets-yesterday"  
      }  
    ],  
    ...  
  }  
}
```

Open an Index

- You can *open* a closed index using the `_open` endpoint
- Be aware that opening an index can take a few seconds or even a couple of minutes
 - The primary shards may need some updates applied
 - A Lucene index will be opened (can take time), and the transaction log will be replayed

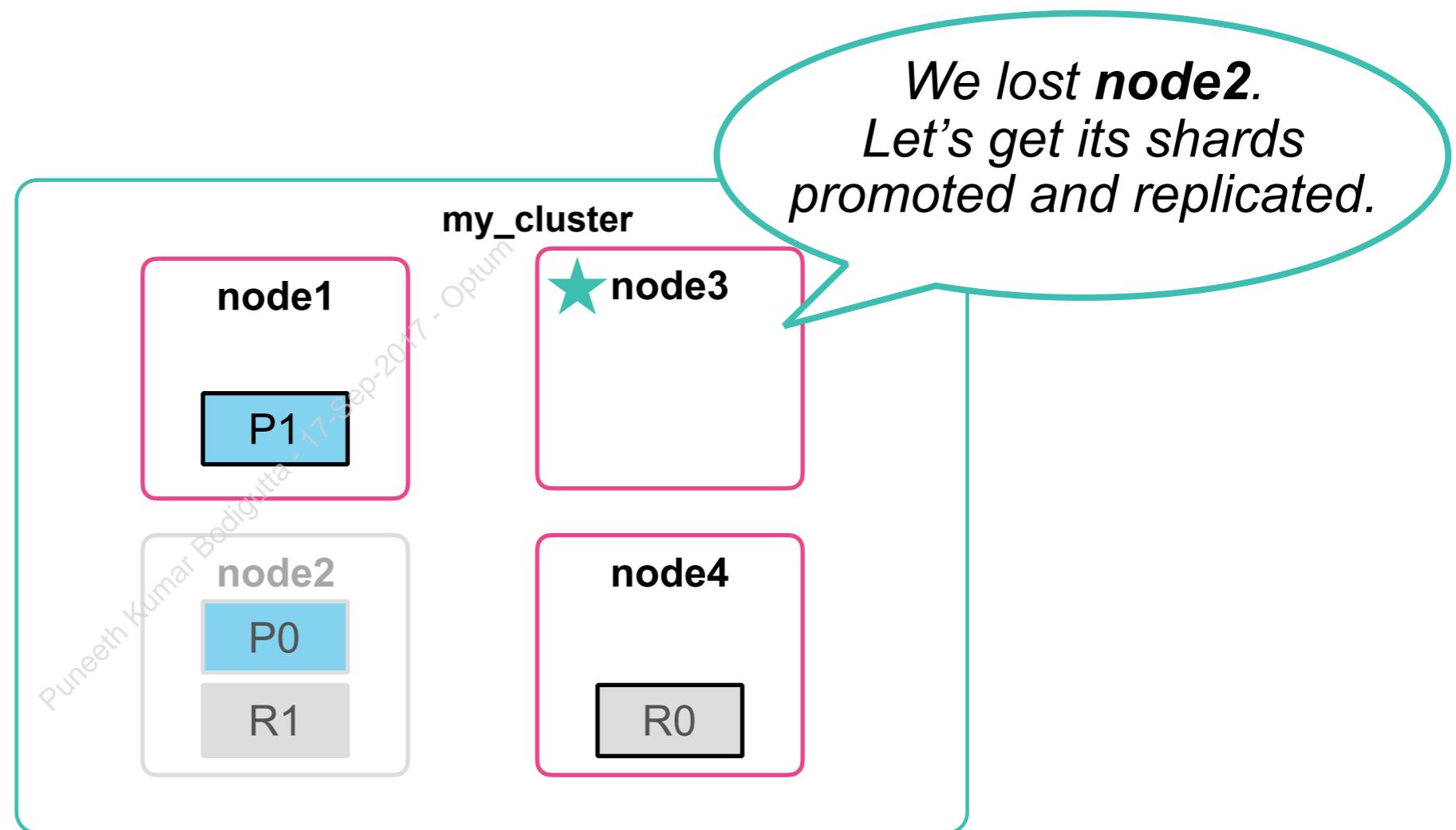
```
POST tweets-yesterday/_open
```

Delayed Shard Allocation

Puneeth Kumar Bodigutta - 17-Sep-2017
Dotum

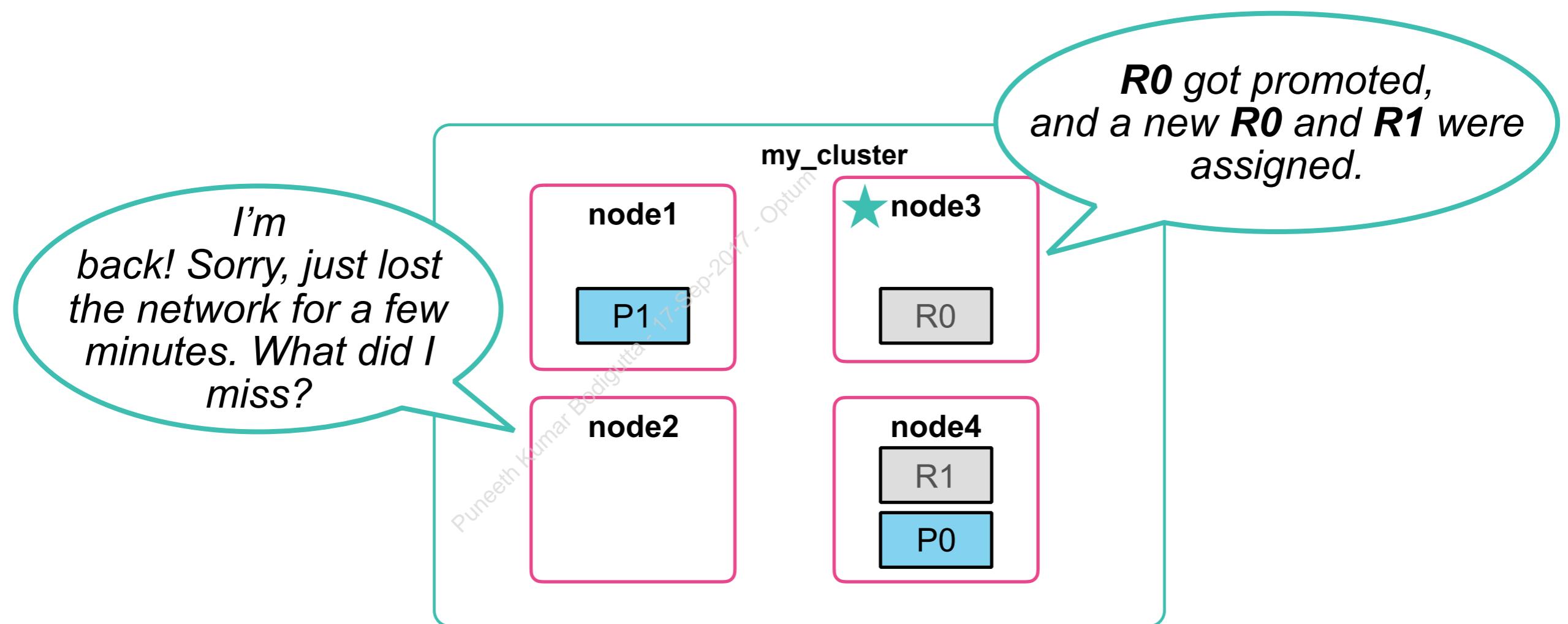
Delaying Shard Allocation

- Suppose a node loses network connectivity or fails in some way
 - the shards lost from that node will need to be promoted and/or replicated on the other nodes



Delaying Shard Allocation

- If node2 is only down for a short time, there may have been a lot of replicas created for no reason
 - the *promotion* of replicas needs to occur immediately,
 - but you can delay the *allocation* of unassigned replicas



Configuring Delayed Shard Allocation

- By default, replicas from a lost node are not immediately assigned
 - the `index.unassigned.node_left.delayed_timeout` setting controls the delay
 - default is 1 minute
- It is a dynamic index setting:

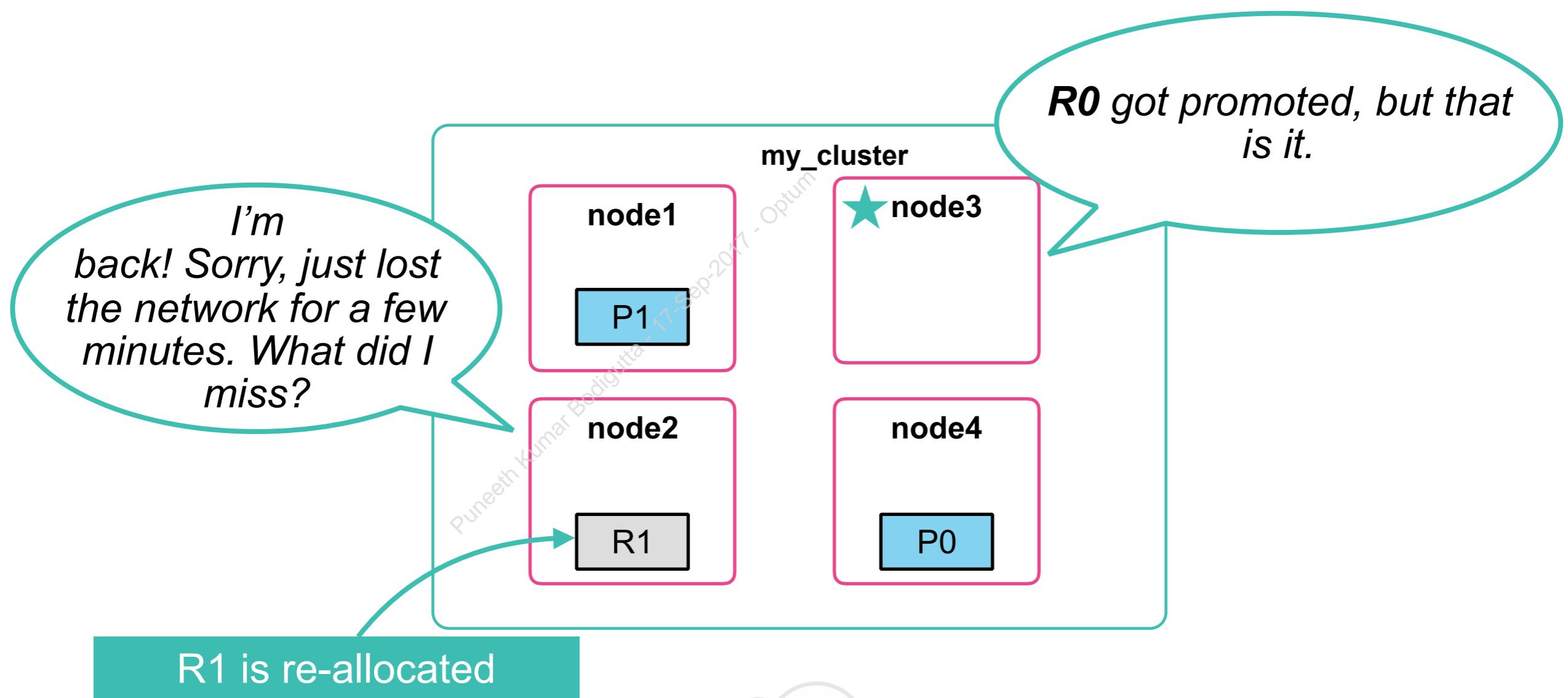
`_all` applies the setting to every index,
but you can list a specific index as well

```
PUT _all/_settings
{
  "settings" : {
    "index.unassigned.node_left.delayed_timeout" : "5m"
  }
}
```

wait 5 minutes before allocating
unallocated replicas

Delaying Shard Allocation

- Now if node2 goes down:
 - the *promotion of replicas still occurs immediately*,
 - but unassigned replicas will exist for at least 5 minutes (and the cluster will be yellow)



Index Recovery Prioritization

- There is an order of priority to how unallocated replicas are recovered:
 - **index.priority** setting (higher before lower)
 - index creation date (newer indices first)
 - index name (alphabetically)
- **index.priority** is a dynamic index setting:

```
PUT my_index_1
{
  "settings": {
    "index.priority" : 5
  }
}
PUT my_index_2
{
  "settings": {
    "index.priority" : 10
  }
}
```

my_index_2 will be
recovered before
my_index_1

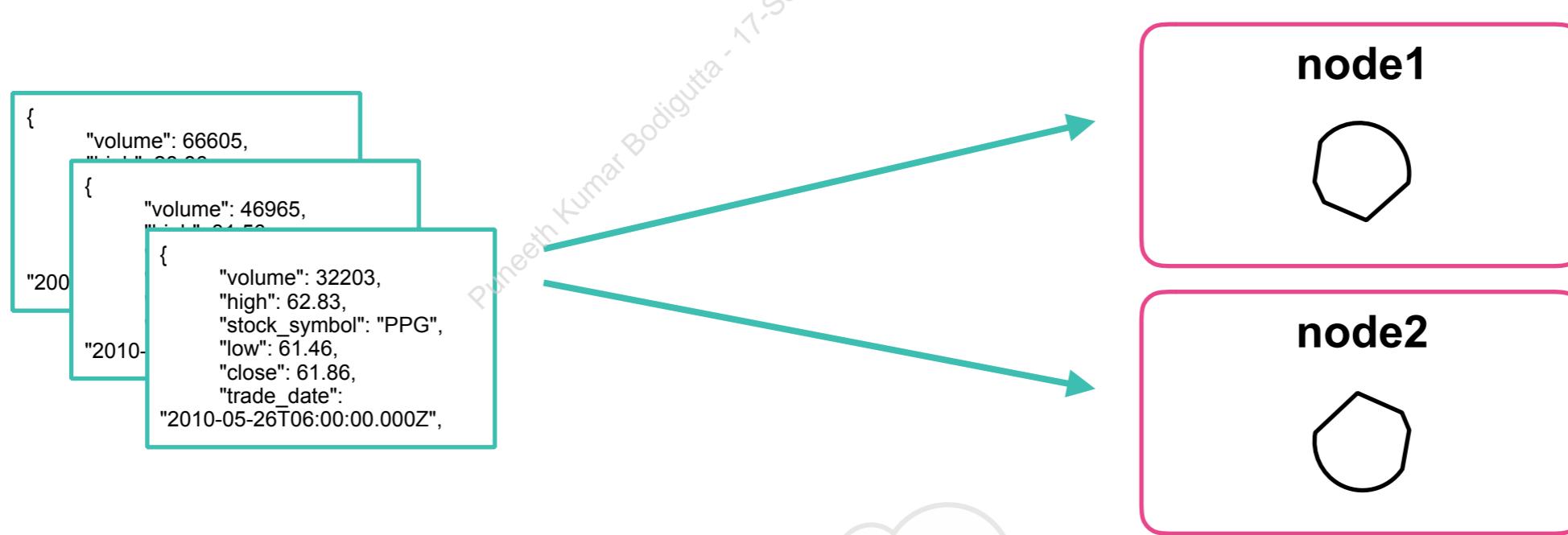
Total Shards Per Node

Puneeth Kumar Bodigutta - 17-Sep-2017 - Opum

Use Case for Controlling Shard Totals

- You can configure a limit on the number of shards that can be assigned to a node
 - during heavy indexing, you can **avoid bottlenecks** by ensuring the shards are evenly distributed
 - you can also use this feature along with shard filtering (discussed later) to ensure shards for a particular index are not only on specific machines, but also **evenly distributed**

You could configure no more than one shard of “**my_stocks**” on each node during heavy indexing



Total Shards Per Node (Index)

- At the index level, you can configure a *hard limit on the number of shards allowed per node*

```
PUT my_index_3
{
  "settings": {
    "routing": {
      "allocation": {
        "total_shards_per_node": 3
      }
    }
  }
}
```

No more than 3 shards (primary or replicas) will be allocated to a single node for **my_index_3**



Hot/Warm Architecture

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

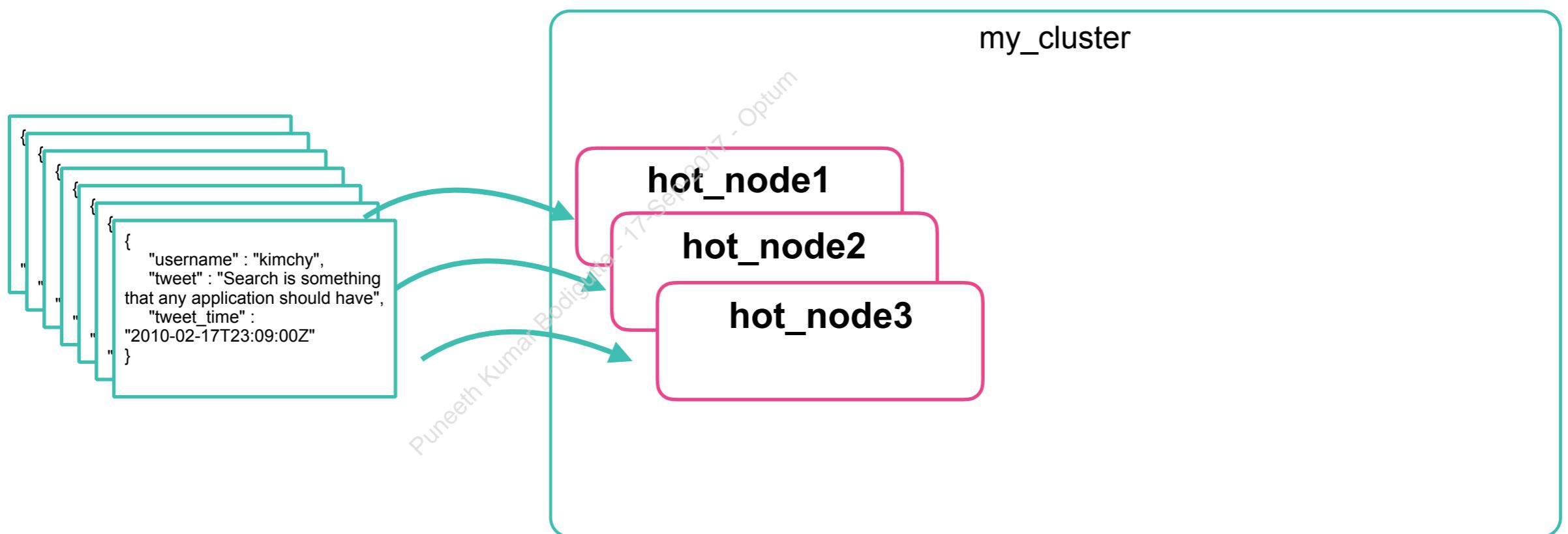
Hot/Warm Architecture

- You can configure the nodes in your cluster to use a *hot/warm architecture*
 - useful for scenarios where you want to control which nodes perform indexing vs. query handling
- **Hot nodes**
 - for supporting the indices with new documents being written to
- **Warm nodes**
 - for handling read-only indices that are not as likely to be queried frequently

Puneeth Kumar Bodigutta - Elasticsearch 2017 - Optum

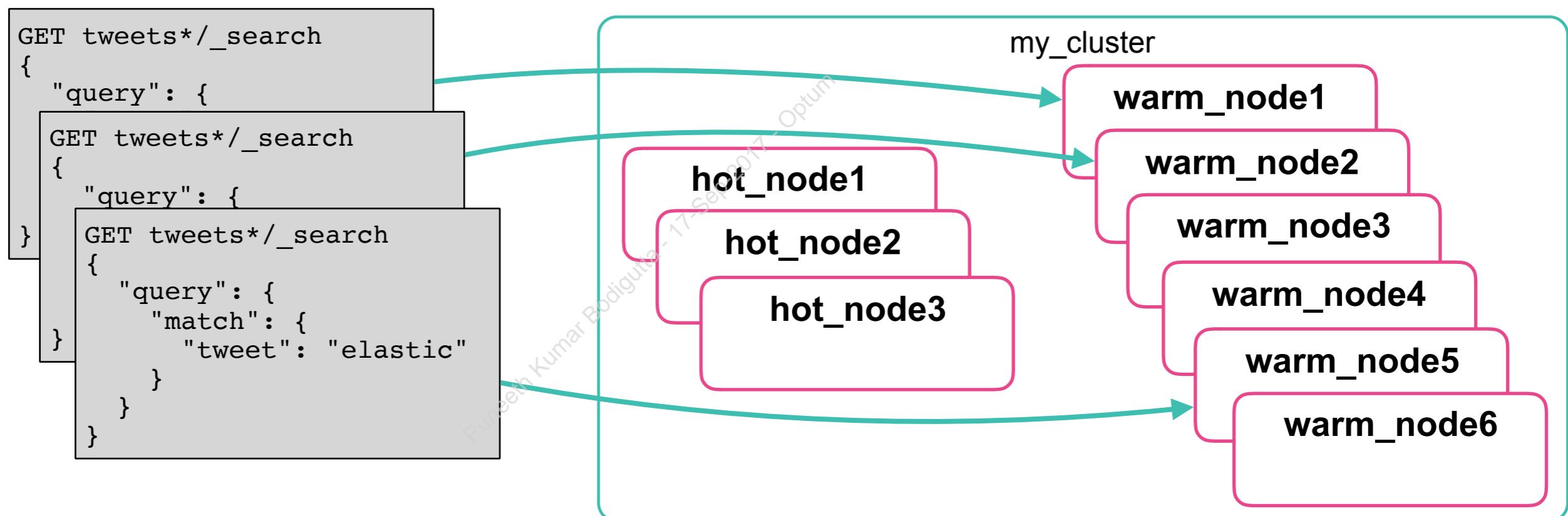
Hot Nodes

- Use *hot nodes* for the indexing
 - indexing is a CPU and IO intensive operation, so hot nodes should be powerful servers
 - faster storage than the warm nodes



Warm Nodes

- Use *warm nodes* for older, read-only indices
 - tend to utilize large attached disks (usually spinning disks)
 - larger amounts of data may require additional nodes to meet performance requirements



Shard Filtering

- How is a hot/warm architecture deployed?
 - you configure ***shard filtering***, which we discuss next...

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Shard Filtering

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

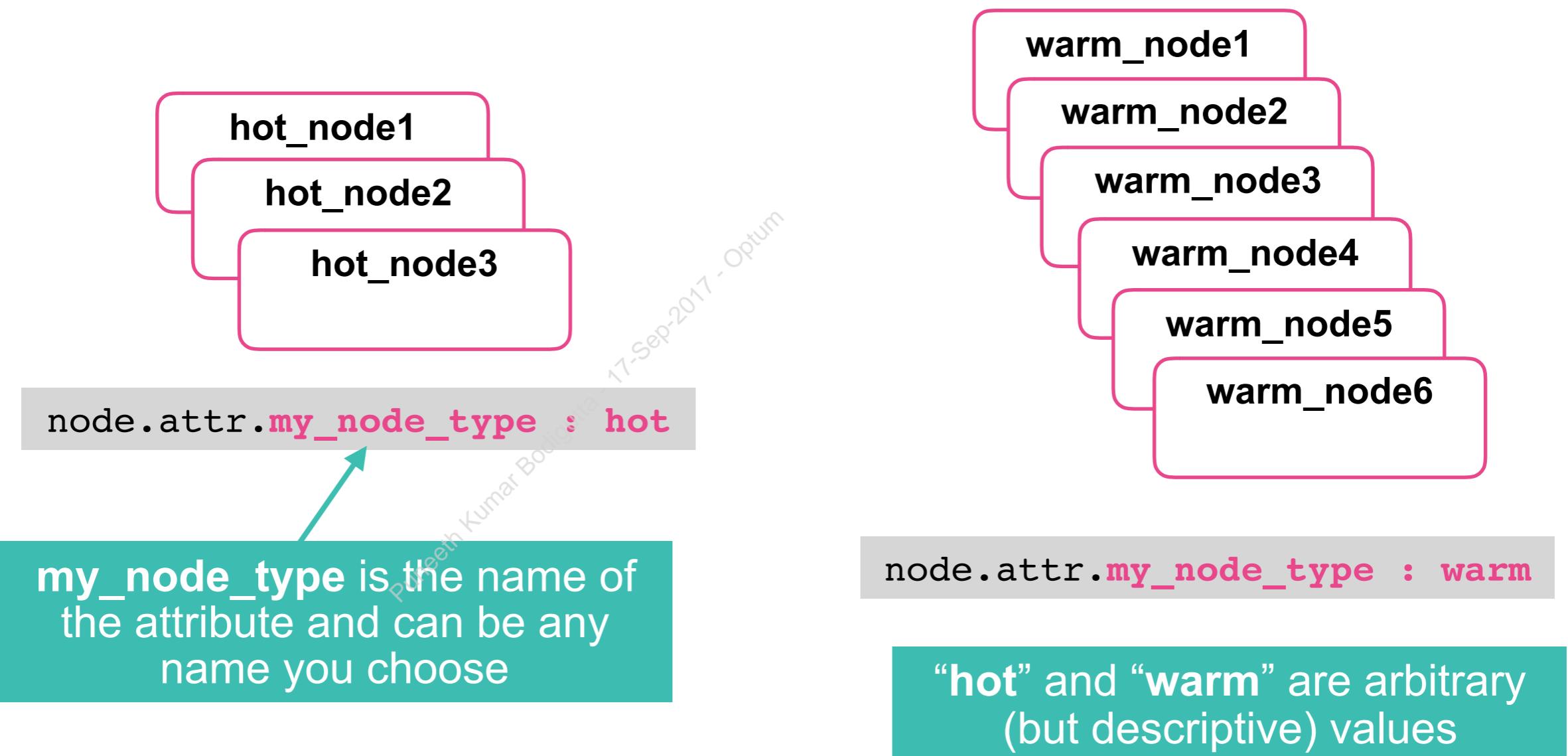
Shard Filtering

- **Shard filtering** refers to the ability to control which nodes the shards for an index are allocated:
 - use **node.attr** to tag your nodes
 - use **index.routing.allocation** to assign indexes to nodes
- Three types of rules for assigning indexes to nodes:

dynamic setting	assign the index to a node whose {attr} has:
<code>index.routing.allocation.include.{attr}</code>	at least one of the values
<code>index.routing.allocation.exclude.{attr}</code>	none of the values
<code>index.routing.allocation.require.{attr}</code>	all of the values

1. Tag the Nodes

- Step 1: *tag* your nodes using the `node.attr` property:
 - a node attribute can be any name and any value
 - either in `elasticsearch.yml` or the `-E` command line option



2. Configure the Hot Data

- **Step 2:** configure your indexes to be allocated to the tagged nodes
 - suppose we want the logs from March, 2017, to be allocated to a “hot” node:

```
PUT logs-2017-03
{
  "settings": {
    "index.routing.allocation.require.my_node_type" : "hot"
  }
}
```

The shards of
logs-2017-03 will only be
on “hot” nodes

3. Move Older Shards to Warm

- Let's move the log index from the previous month to "warm" nodes
 - index.routing.allocation** is a dynamic setting, so we can change it using the API:

```
PUT logs-2017-02/_settings
{
  "index.routing.allocation.require.my_node_type" : "warm"
}
```

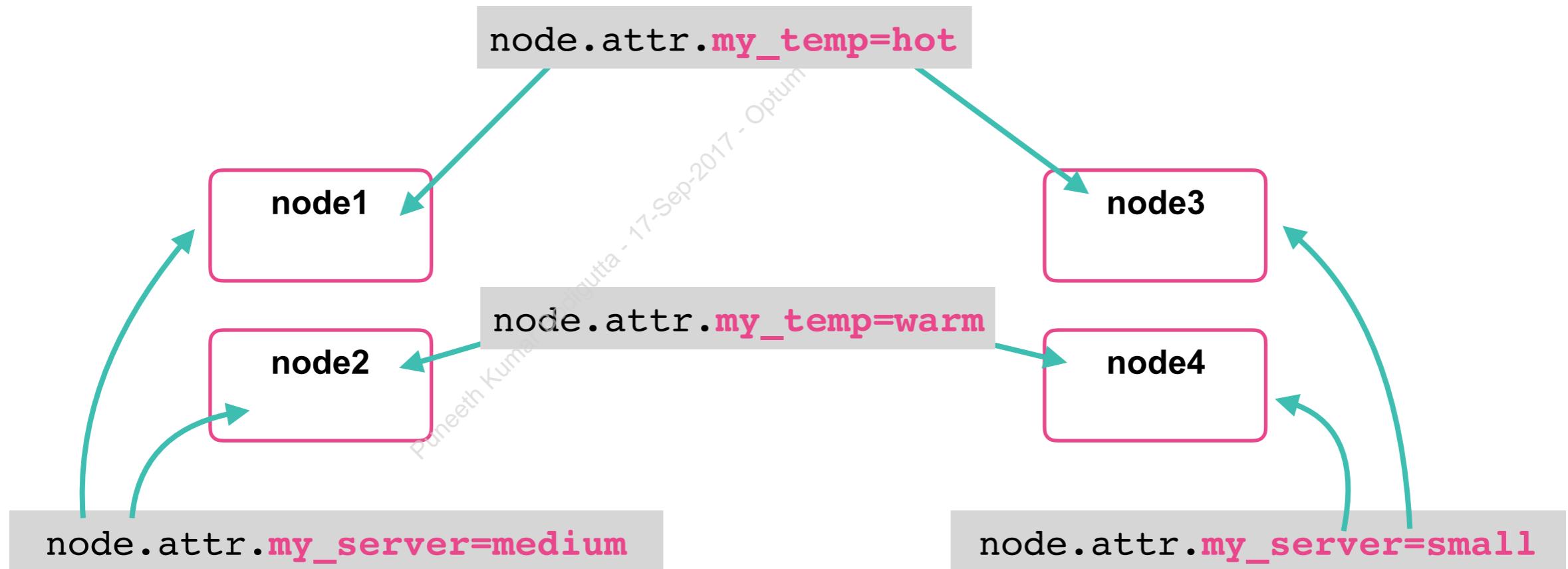
Move the shards from February, 2017
to the warm nodes

Shard Filtering for Hardware

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Shard Filtering for Hardware

- Suppose you are implementing a hot/warm architecture
 - and your nodes are tagged accordingly with “`my_temp`”
- But you also have different sizes of hardware:
 - so you tag those using “`my_server`” as “`small`”, “`medium`”, “`large`”



Configure Your Indices

```
PUT my_index1
{
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 1,
    "index.routing.allocation.include.my_server" : "medium",
    "index.routing.allocation.require.my_temp" : "hot"
  }
}
```

*Put **my_index1**
on a **medium** server that
is also “**hot**”*

```
PUT my_index2
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 1,
    "index.routing.allocation.include.my_server" : "medium,small",
    "index.routing.allocation.exclude.my_temp" : "hot"
  }
}
```

*Put **my_index2** on any
server that is not “**hot**”*



Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- You can configure a ***delay*** for shard allocation when a node fails
- At the index and cluster level you can configure a ***hard limit on the number of shards*** allowed per node
- You can copy documents from one index to another using the ***Reindex API***
- You can ***close*** and ***open*** indices using the ***Open/Close Index API***
- You can use shard filtering to configure a ***hot/warm architecture*** for your cluster
- ***Shard filtering*** refers to the ability to control to which nodes an index is allocated

Puneeth Kumar Bodiguttu - 11 Sep 2017 - Optum



Quiz

1. **True or False:** When using the Reindex API, the target index must exist already.
2. When reindexing, what does "**version_type**" : "external" do?
3. **True or False:** You can **GET** a document from an index that has been closed.
4. Suppose you created a new index every day for that day's log files. How could this scenario benefit from a hot/warm architecture?
5. What happens if you configure an index's shard filtering with a scenario that is impossible for the cluster to implement?

Puneeth Kumar Bodapati - 17-Sep-2017 Optum



Lab 7

Index Management

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Chapter 8

Capacity Planning

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Designing for Scale
- Capacity Planning
- Scaling with Replicas
- Scaling with Indices
- Capacity Planning Use Cases
- Time-based Data

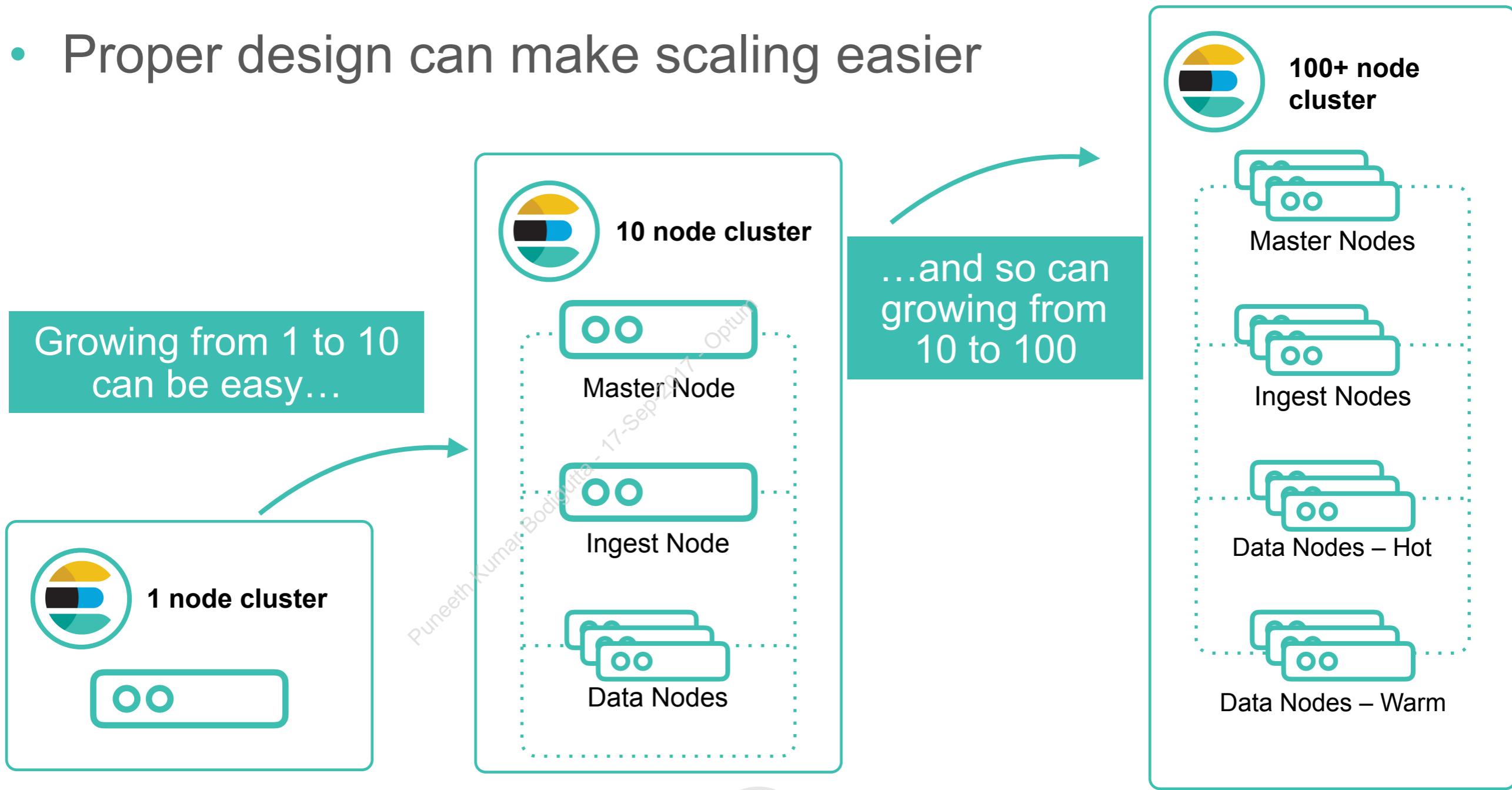
Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Designing for Scale

Puneeth Kumar Bodigutta - 17-Sep-2017 · Opum

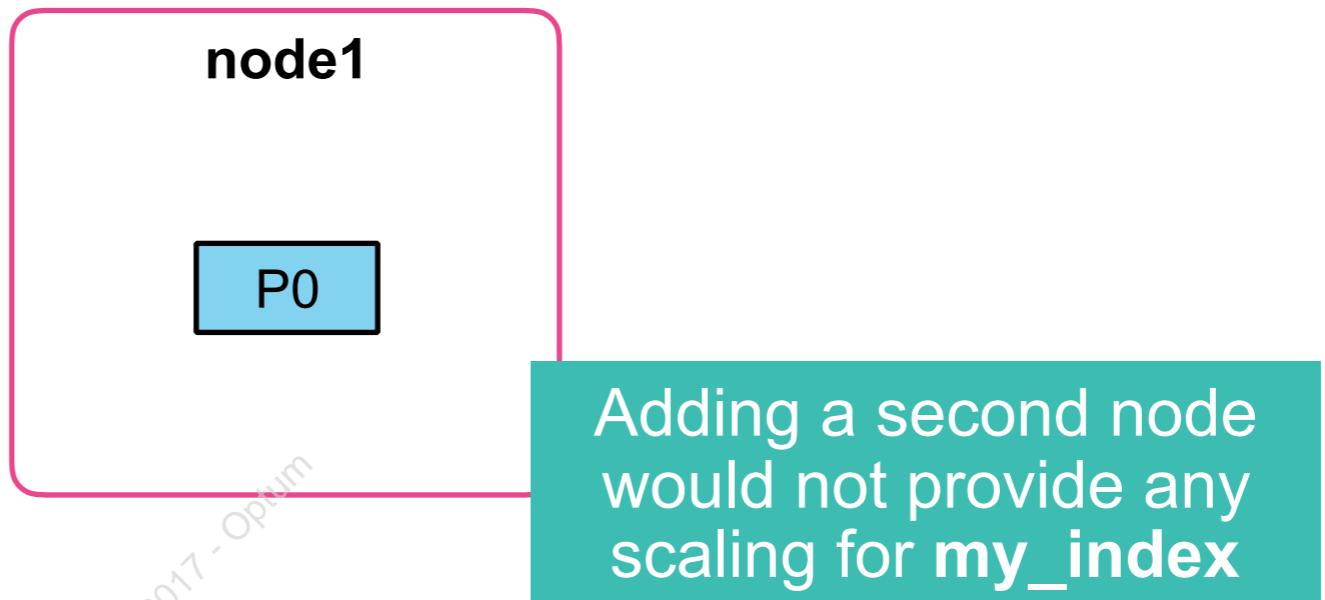
Designing for Scale

- Elasticsearch is built to scale
 - and the default settings can take you a long way
- Proper design can make scaling easier



One Shard...

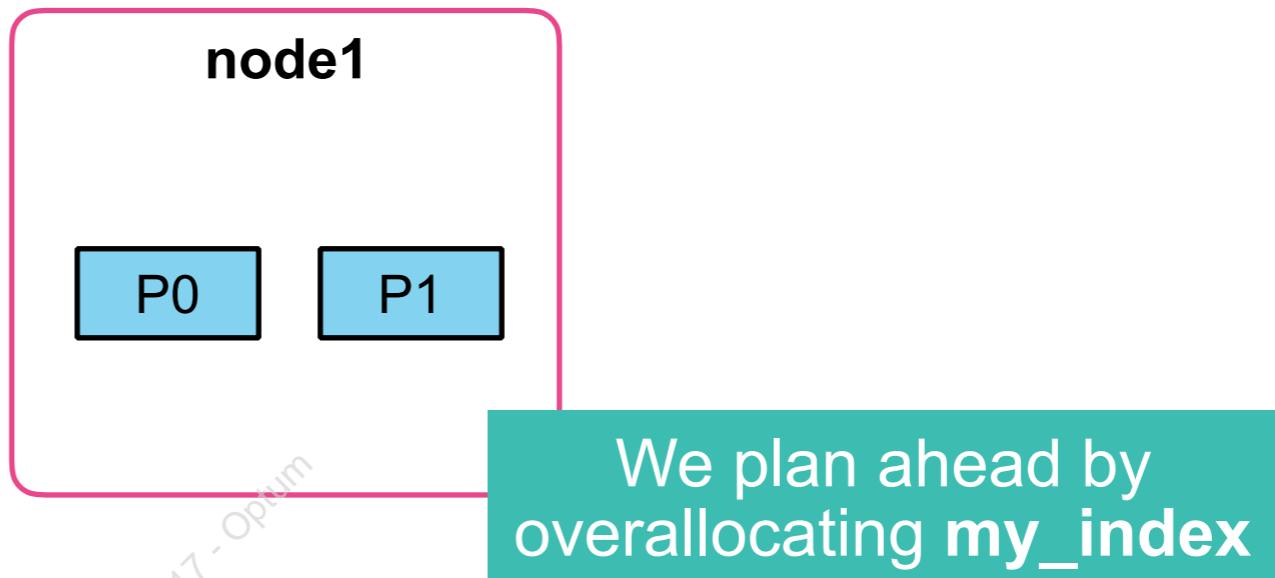
- ...does not scale very well:



```
PUT my_index
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 0
  }
}
```

Two Shards...

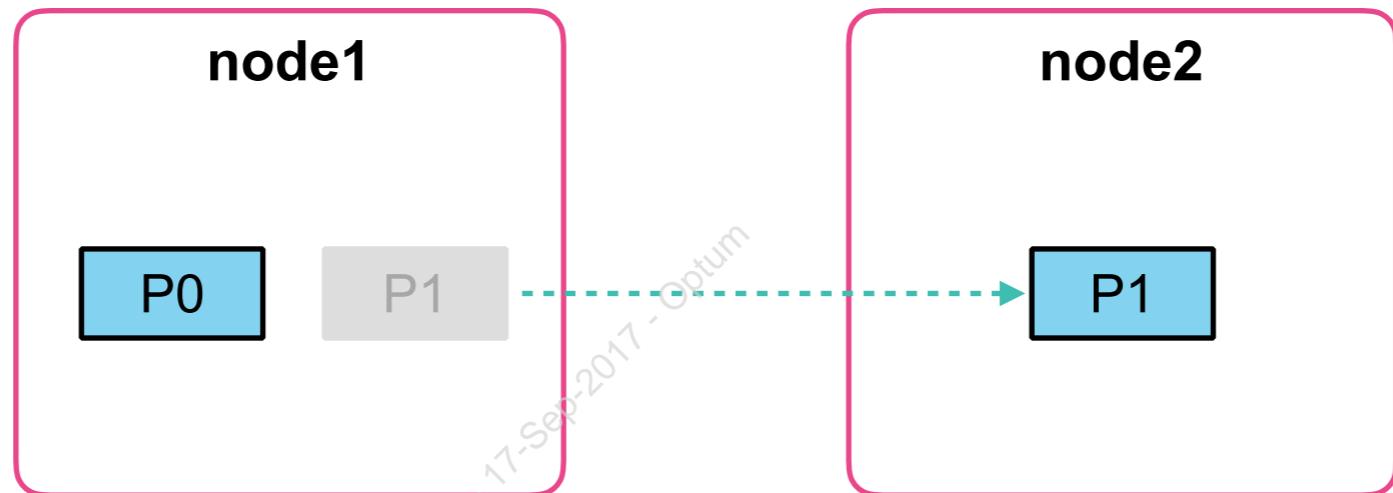
- ...can scale if we add a node:



```
PUT my_index
{
  "settings": {
    "number_of_shards": 2,
    "number_of_replicas": 0
  }
}
```

Balancing of Shards

- Elasticsearch automatically balances the shards:



Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

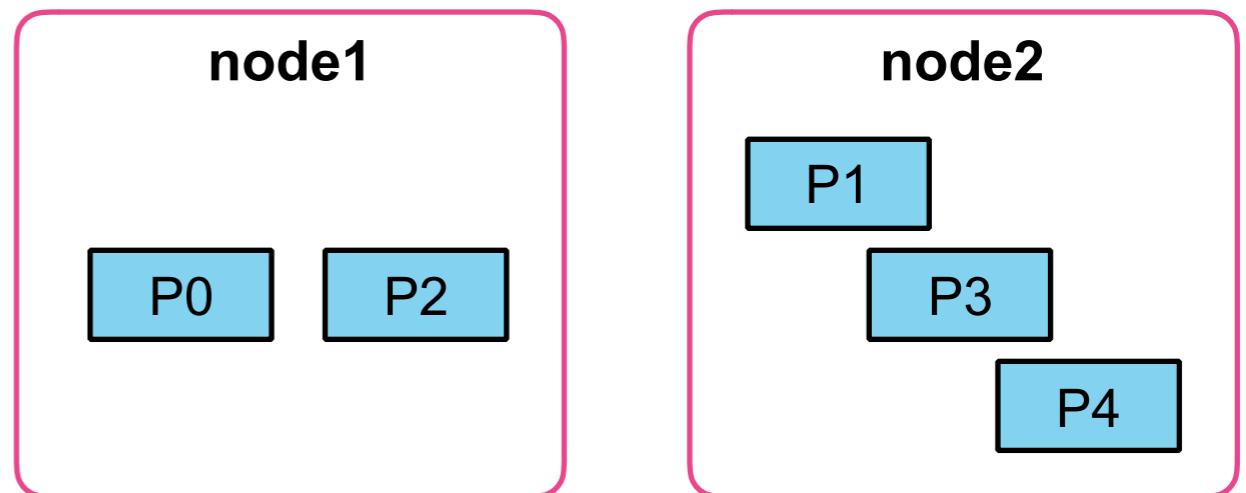
Adding a node causes
the cluster to rebalance

Shard Overallocation

- If you are expecting your cluster to grow, then it is good to plan for that by overallocating shards:
 - $\#shards > \#nodes$
- Shards can move between nodes quickly as the cluster grows
 - and there is no downtime during the autosharding

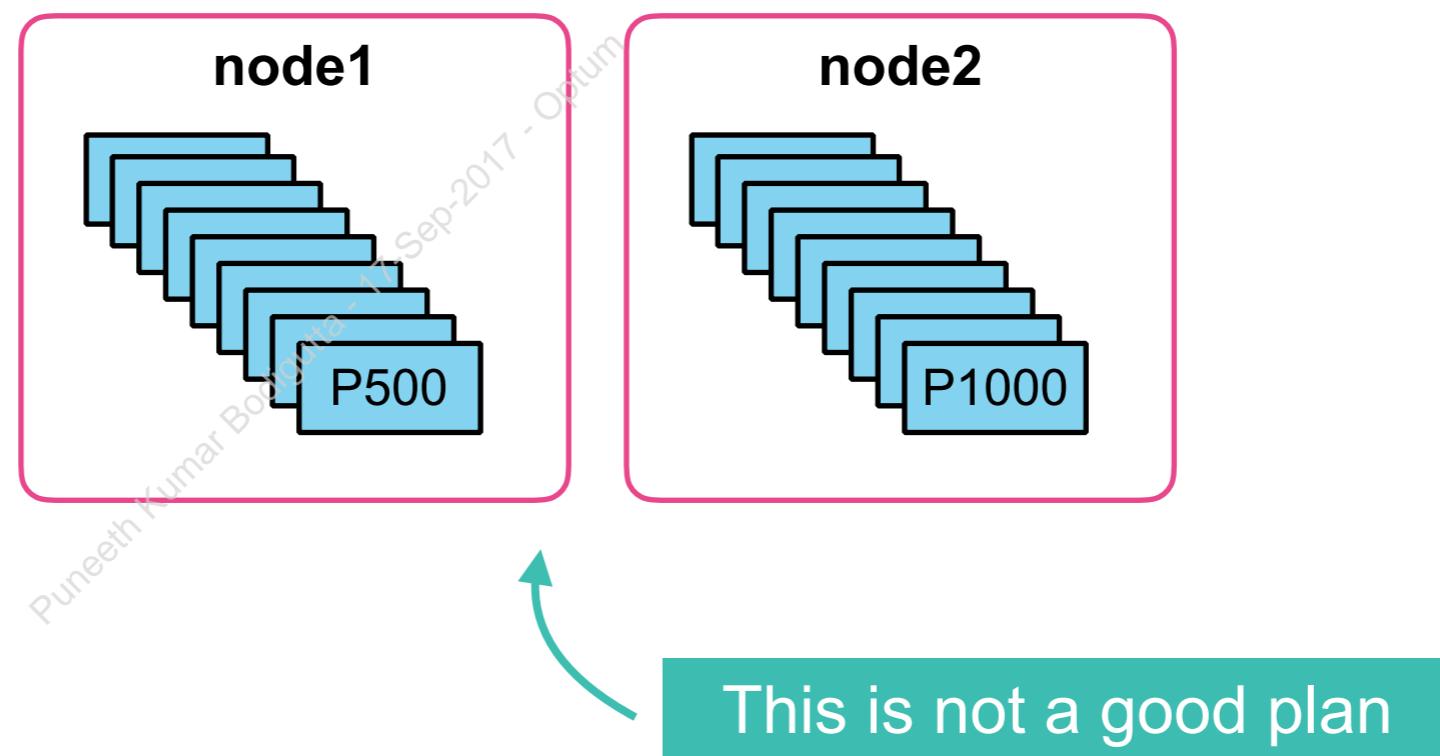
```
PUT my_index
{
  "settings": {
    "number_of_shards": 5,
    "number_of_replicas": 0
  }
}
```

Puneet Agarwal - 17-Sep-2017 - Optum



“Too Much” Overallocation

- A little overallocation is good
- A “kagillion” shards is not:
 - each shard comes at a cost (Lucene indices, file descriptors, memory, CPU)
 - also, a search request needs to hit every shard in the index



Capacity Planning

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Capacity Planning

- So how many shards should I configure for my index?
 - no simple formula to it
 - “It depends!”
- Too many factors involved:
 - hardware
 - # of documents
 - size and complexity of your documents
 - how you index the data
 - how you search and aggregate the data
 - how many indices the data will be spread across

Capacity Planning

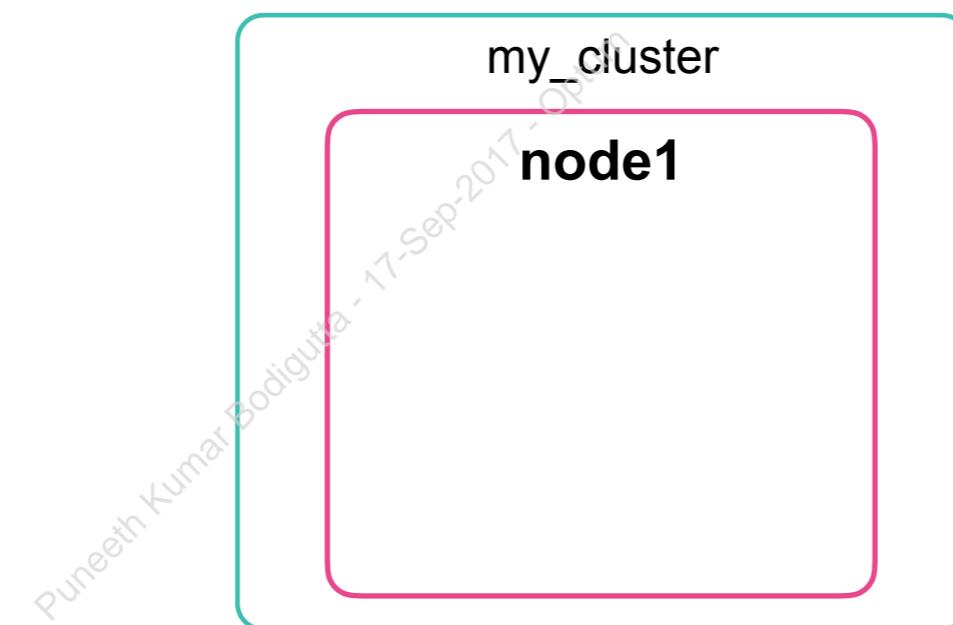
- Before trying to determine your capacity, you need to determine your SLA:
 - How many ***docs/second*** do you need to index?
 - How many ***queries/second*** do you need to process?
 - What is the ***maximum response time*** for queries?
- Get some production data
 - ***actual documents*** you are going to index
 - ***actual queries*** you are going to run in production
 - ***actual mappings*** you are going to use

Puneeth Kumar Podigutta 17-Sep-2017/Optum

Maximum Shard Capacity

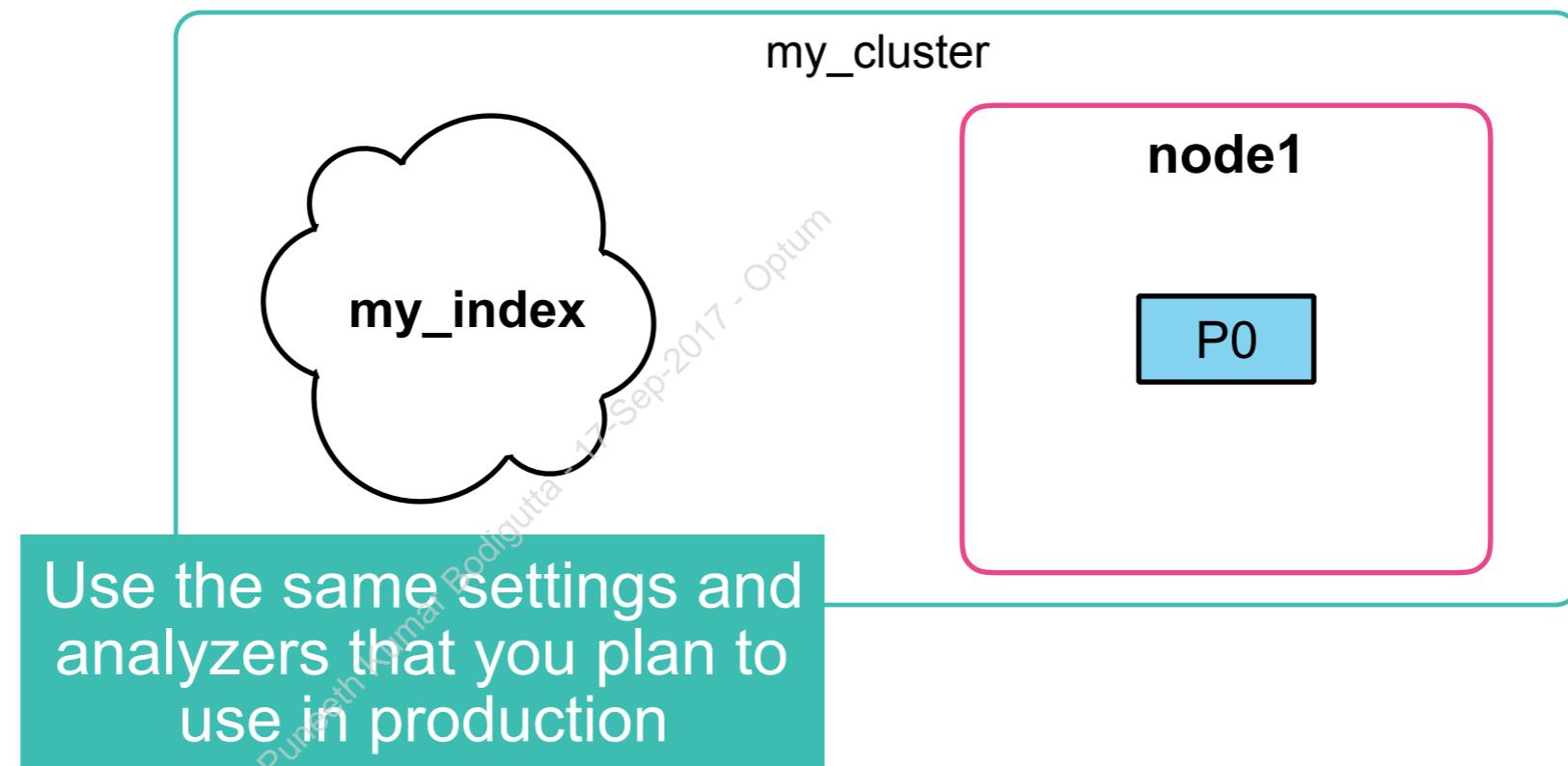
- You can evaluate the maximum shard size for your particular use case

1. Create a 1-node cluster using your production hardware



Maximum Shard Capacity

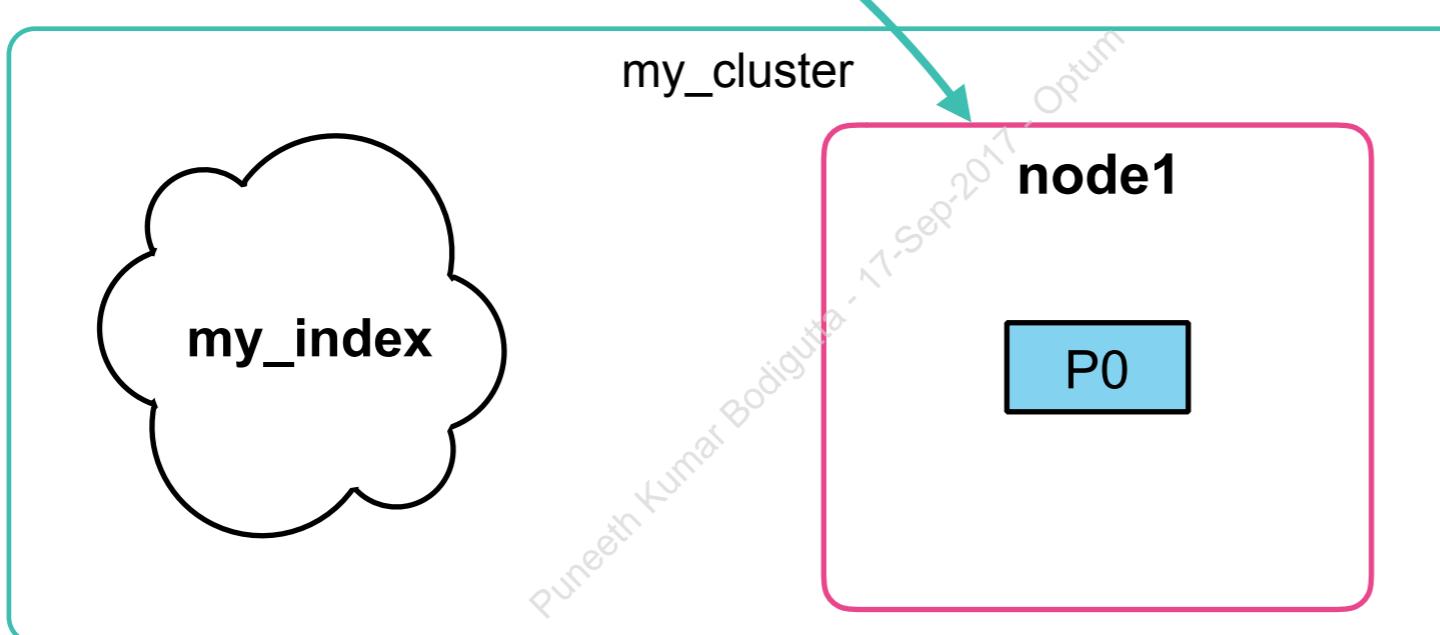
2. Create a single index with 1 shard and no replicas



Maximum Shard Capacity

```
GET my_index/_search
{
  ...
}
```

3. Incrementally index documents and run your searches and aggregations. Push this shard and index docs until it “breaks” and you will find its max capacity



“**breaks**” depends on your own definition - ingestion rates, search rates, latency, etc.

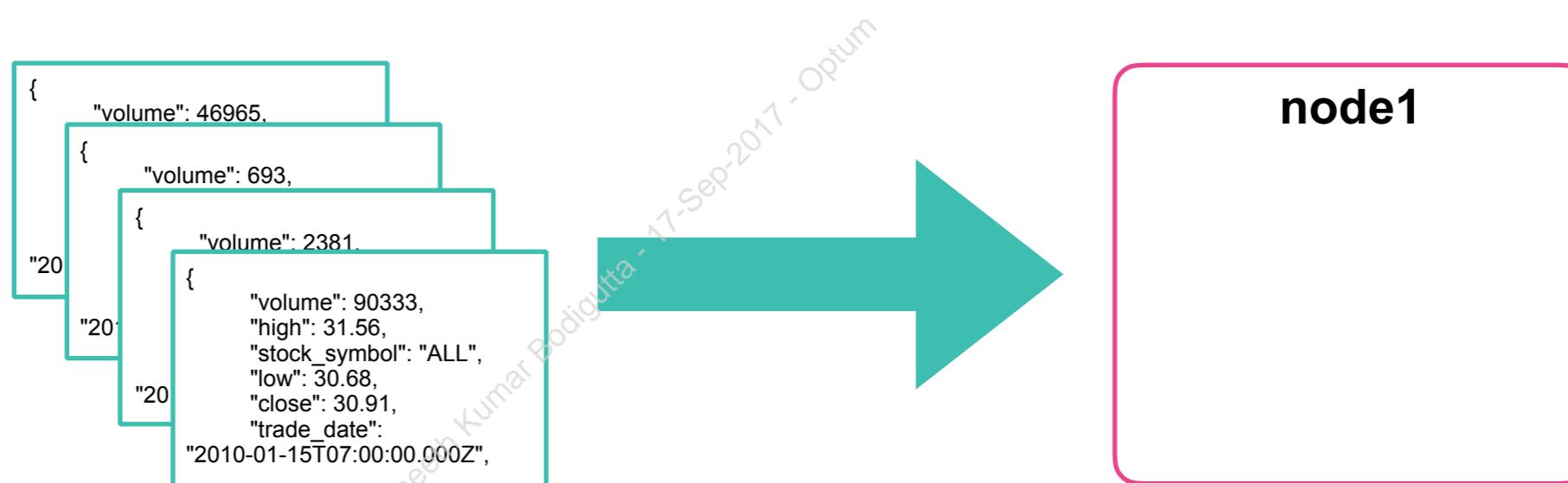
Number of Primary Shards

- It is not an exact science, but you can:
 - estimate the total amount of data for your index
 - leave room for growth (if applicable)
 - divide by the maximum capacity of a single shard
- The number of primary shards is usually determined by indexing speed
- For searching, the total number of shards is the measurement (agnostic of index)
 - note we are not talking about replicas here, just number of primary shards
 - we will talk about replicas next...



Measure Indexing Capacity of a Node

- You can calculate the indexing capacity of a single node
 - index documents in a similar way as your application will
 - index in parallel until 429 responses begin to come back
 - use this to calculate the number of **docs/second**
 - you can now calculate how to scale your nodes for ingestion based on your SLA



Index docs until a 429 response, which mean ES can not keep up with indexing

Scaling with Replicas

Puneeth Kumar Bodigutta - 17-Sep-2017, Opnum

Scaling with Replicas

- Adding *replicas* provides an additional level of scaling
 - We know replicas provide high availability
 - They also provide scaling for reads and searches (if you add additional hardware)

Suppose our index has
one shard on one node

```
PUT my_index
{
  "settings": {
    "number_of_shards": 1,
    "number_of_replicas": 0
  }
}
```

node1

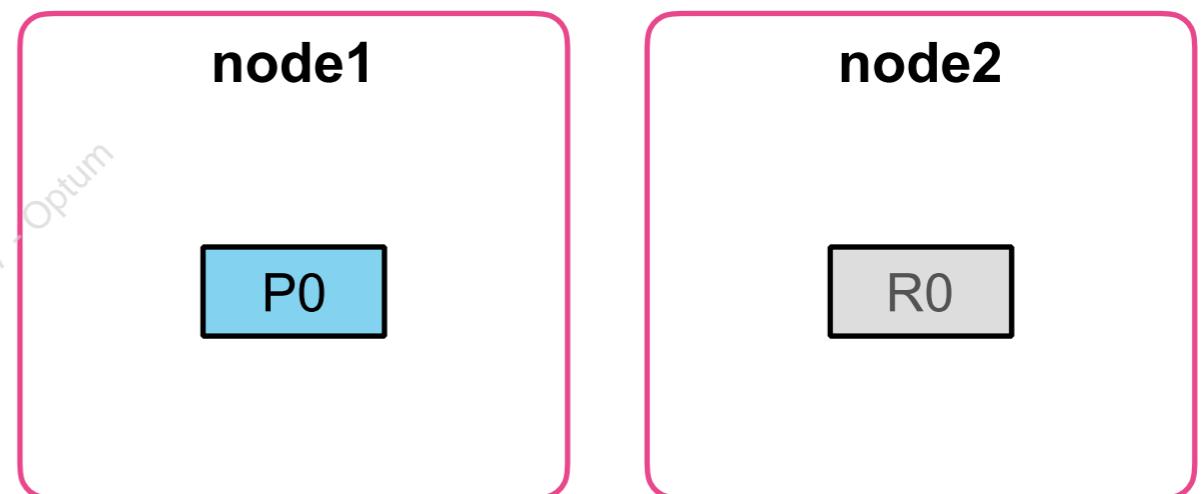
P0

Scaling with Replicas

- If you add another node, you can increase the number of replicas:
 - “**number_of_replicas**” is a dynamic setting

```
PUT my_index/_settings
{
  "number_of_replicas": 1
}
```

Puneeth Kumar Bodigutta
17 Sep 2017, Optum



Adding a node **and** replicas adds compute power to the cluster

The Cost of Replicas

- Replicas have a cost associated with them as well, including:
 - slower indexing speed (although replicas are indexed in parallel, so it is not a cumulative cost)
 - more storage on disk
 - larger associated heap memory footprint of the extra shard(s)

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Scaling with Indices

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

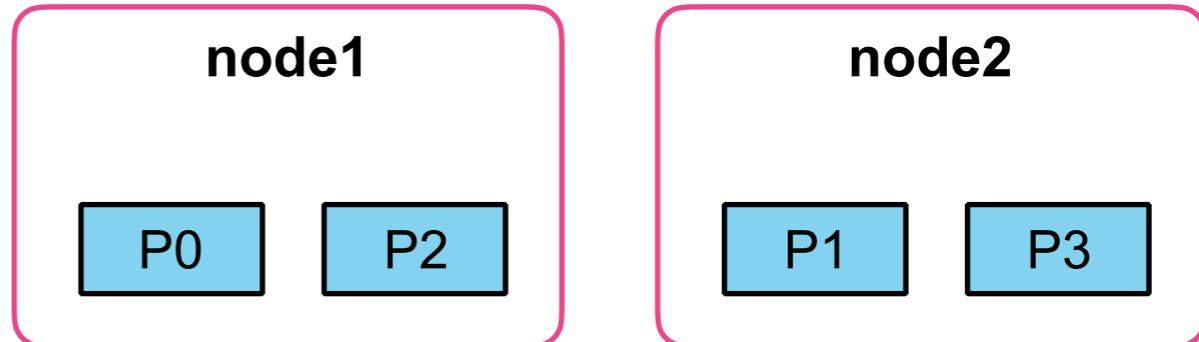
Scaling with Indices

- Using *multiple indices* also provides scaling
 - If you need to add capacity, consider just creating a new index
- Then search across both indices to search “new” and “old” data
 - you could even define a single alias for the multiple indices
- Searching 1 index with 50 shards is equivalent to searching 50 indices with 1 shard each

Puneeth Kumar Bodigutta - 17-Sep-2017

Scaling with Indices

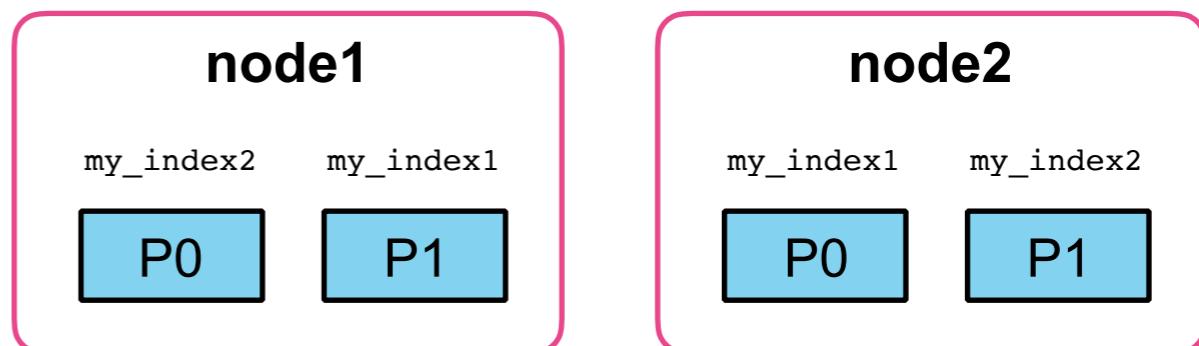
```
PUT my_index
{
  "settings": {
    "number_of_shards": 4
  }
}
```



```
PUT my_index1
{
  "settings": {
    "number_of_shards": 2
  }
}

PUT my_index2
{
  "settings": {
    "number_of_shards": 2
  }
}
```

Searching `my_index` hits 4 shards.
So does a search over
`my_index1,my_index2`



Capacity Planning Use Cases

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Capacity Planning Use Cases

- When planning your cluster and designing indices, it is important to understand:
 - what your data looks like,
 - and how that data is going to be searched
- Two very common use cases are:
 - **searching fixed-size data:** searching a large dataset that may grow slowly
 - **time-based data:** data that grows rapidly, like log files

Puneeth Kumar Bodigutta - 17Sep2017 - Optum



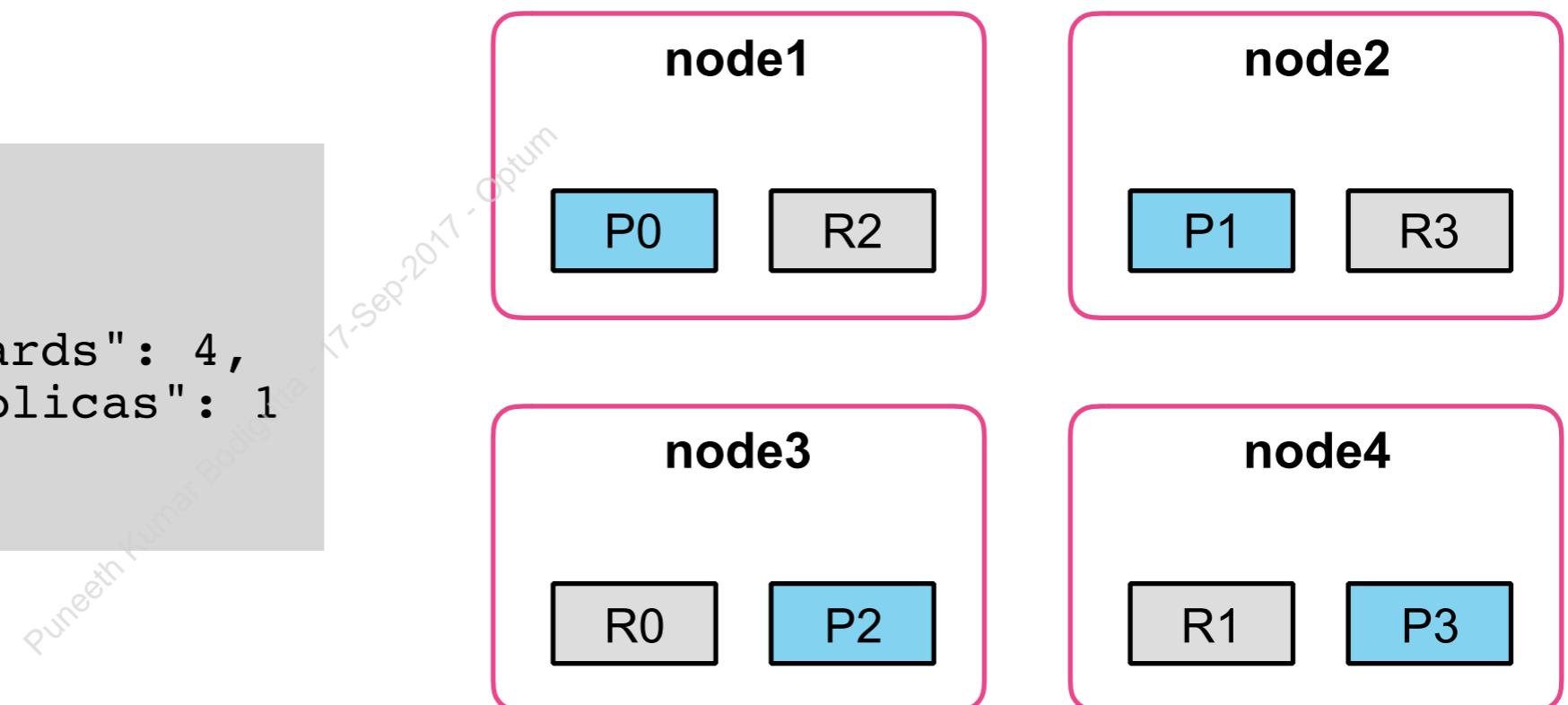
Fixed-size Data

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Fixed-size Data

- Your use case may involve searching a large dataset that only gradually grows and/or changes
 - relatively **fixed-size** collection of documents
 - search for relevant documents, no matter their age

```
PUT hotels
{
  "settings": {
    "number_of_shards": 4,
    "number_of_replicas": 1
  }
}
```

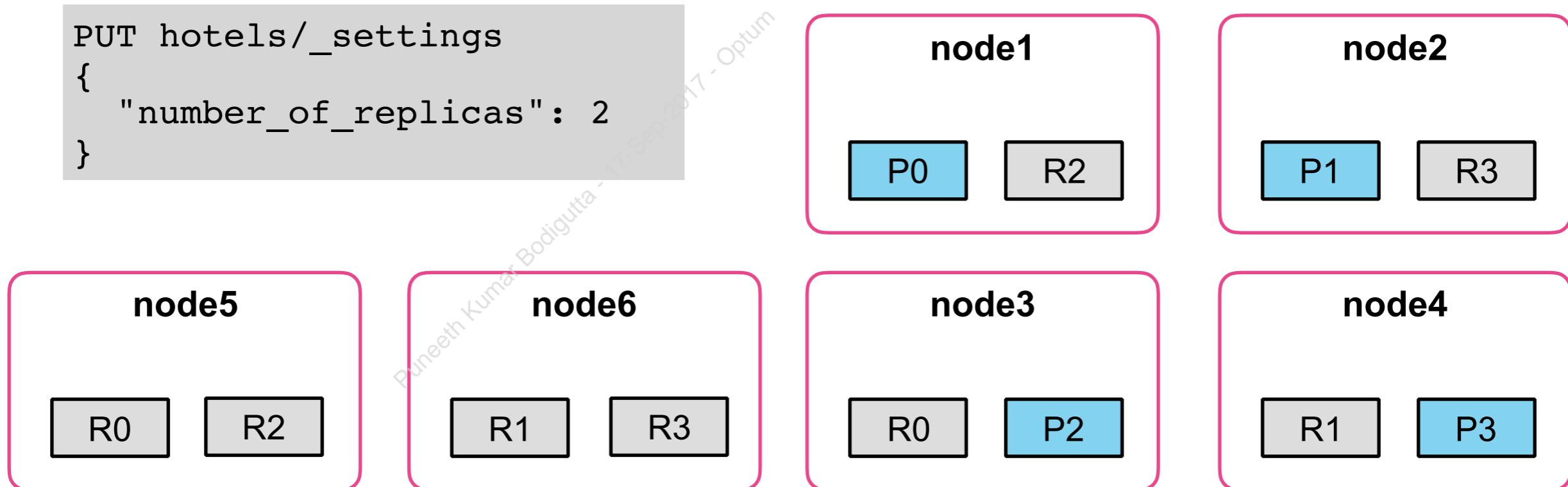


Planning for Fixed-size Data

- In this case, size your indices based on the maximum shard capacity vs. the amount of data you anticipate
 - If you need to increase capacity, either reindex or add multiple indices (but preferably not very often)
 - Easy to increase throughput by adding more nodes and replicas

```
PUT hotels/_settings
{
  "number_of_replicas": 2
}
```

Puneeth Kumar Bodigutta - 17 Sep 2017 - Optum



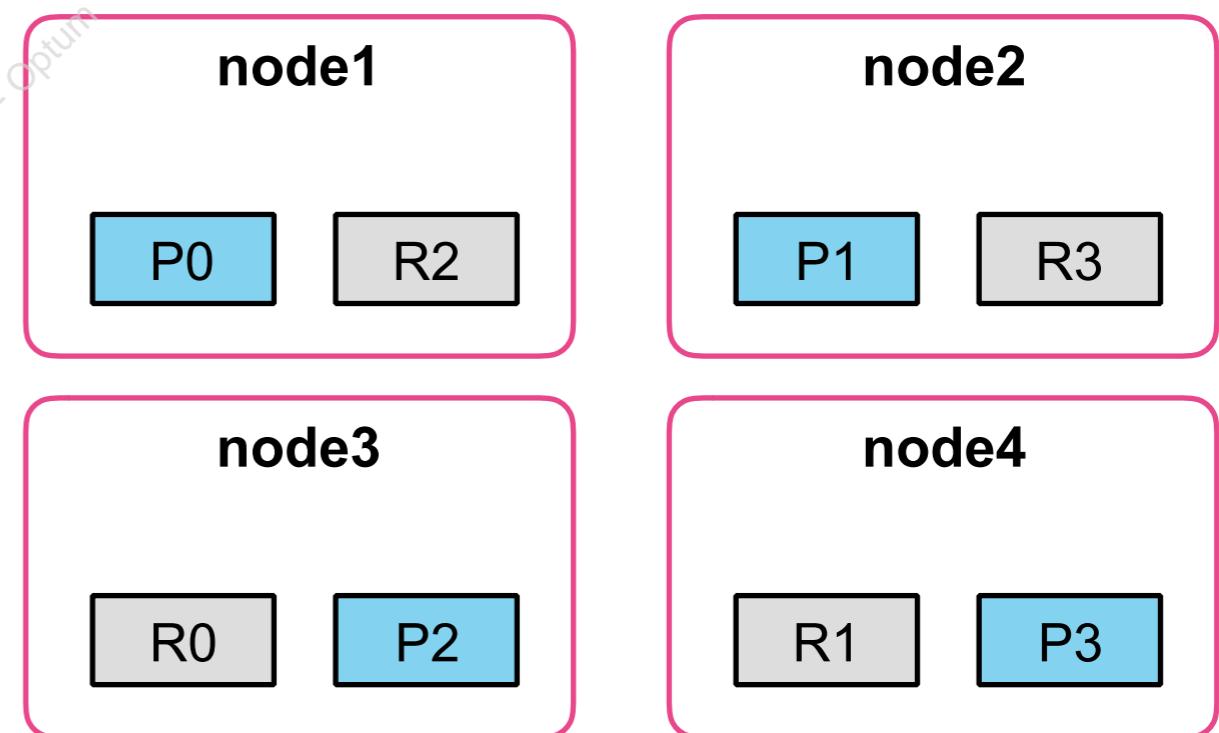
Time-based Data

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized

Time-based Data

- *Time-based data* includes:
 - logs
 - social media streams
 - time-based events
- These documents have a timestamp, and likely do not change

```
PUT tweets-2017-02-05
{
  "settings": {
    "number_of_shards": 4,
    "number_of_replicas": 1
  }
}
```



Planning for Time-based Data

- ***Searching*** on time-based data usually involves a timestamp
 - you typically search for recent events
 - older documents become less important
- ***Data ingestion*** is another key factor to consider
 - you typically have a lot of data coming in
 - you do not want indexing to become a bottleneck (it has to keep up)

Puneeth Kumar Bodigutta - 17-Sep-2017 Optum



Index per Time Frame

- Time-based data is best organized using *time-based indices*
 - create a new index each day, week, month, year (whatever is appropriate for the amount of data being ingested)
 - add the date to the name of the index

```
PUT tweets-2017-02-05
{
  "settings": {
    "number_of_shards": 4,
    "number_of_replicas": 1
  }
}

PUT tweets-2017-02-06
{
  "settings": {
    "number_of_shards": 4,
    "number_of_replicas": 1
  }
}

PUT tweets-2017-02-07
{
  "settings": {
    "number_of_shards": 4,
    "number_of_replicas": 1
  }
}
```

Puneeth Ramamurthy

Searching Time-based Indices

- Searches can use **wildcards** or **aliases** to search over multiple timeframes:

I want to search all tweets in February, 2017

```
GET tweets-2017-02*/_search
```

Puneeth Kumar Bodigutta - 17-Sep-2017

Data Ingestion for Time-based Indices

- Best practice is to create an alias for the *active index* for data ingestion
 - a **PUT** request can only target one index, so the alias can only point to a single index

```
POST _aliases
{
  "actions": [
    {
      "add": {
        "index": "tweets-2017-02-06",
        "alias": "tweets_write"
      }
    },
    {
      "remove": {
        "index": "tweets-2017-02-05",
        "alias": "tweets_write"
      }
    }
  ]
}
```

All indexing can be sent to
“**tweets_write**” without worrying
about the date

Managing Time-based Indices

- For optimal ingest rates:
 - Spread the shards of your active index over as many nodes as possible
 - for example, 20 nodes = 20 primary shards on the active index
- For optimal search and low resource usage:
 - shrink the older indices down to the optimal number of shards
 - close indices that are no longer being searched
- Use a hot/warm architecture
 - use hot nodes for indexing and warm nodes for querying

Puneeth Kumar Podigutta
17 Sep 2017 Optum

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- If you are expecting your cluster to grow, then it is good to plan for that by ***overallocating*** shards
- A little overallocation is good. A “kagillion” shards is not
- You can attempt to calculate the ***maximum shard size*** for your particular use case by pushing the limits of your index using one primary shard on a one-node cluster
- You can scale the query workload of your cluster by adding more nodes and ***increasing the number of replicas*** of your indices
- You can similarly provide scaling by distributing your documents across ***multiple indices***

Puneeth Kumar Bodigala - 11 Sep 2017



Quiz

- True or False:** The number of primary shards of an index is fixed at the time the index is created.
- Is it more optimal to search over 1 index with 10 primary shards, or 10 indices with 1 primary shard each?
- If you have a two node cluster, why would you ever create an index with more than two primary shards?
- True or False:** Creating an index with only one primary shard is not a good design.
- Suppose you calculated the max shard size for your dataset to be about 100,000 documents. How many shards should you use for a relatively fixed-size dataset of 900,000 documents?

Puneeth Kumar Podder - ES 2017 - Optimal Indexing

Lab 8

Capacity Planning

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Chapter 9

Cluster Management

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Shard Allocation Awareness
- Forced Awareness
- Installing Plugins
- Cluster Backup
- Repositories
- Taking a Snapshot
- Restoring from a Snapshot
- The Incremental Nature of Snapshots

Puneeth Kumar Duggutta - Sep-2017 - Optum

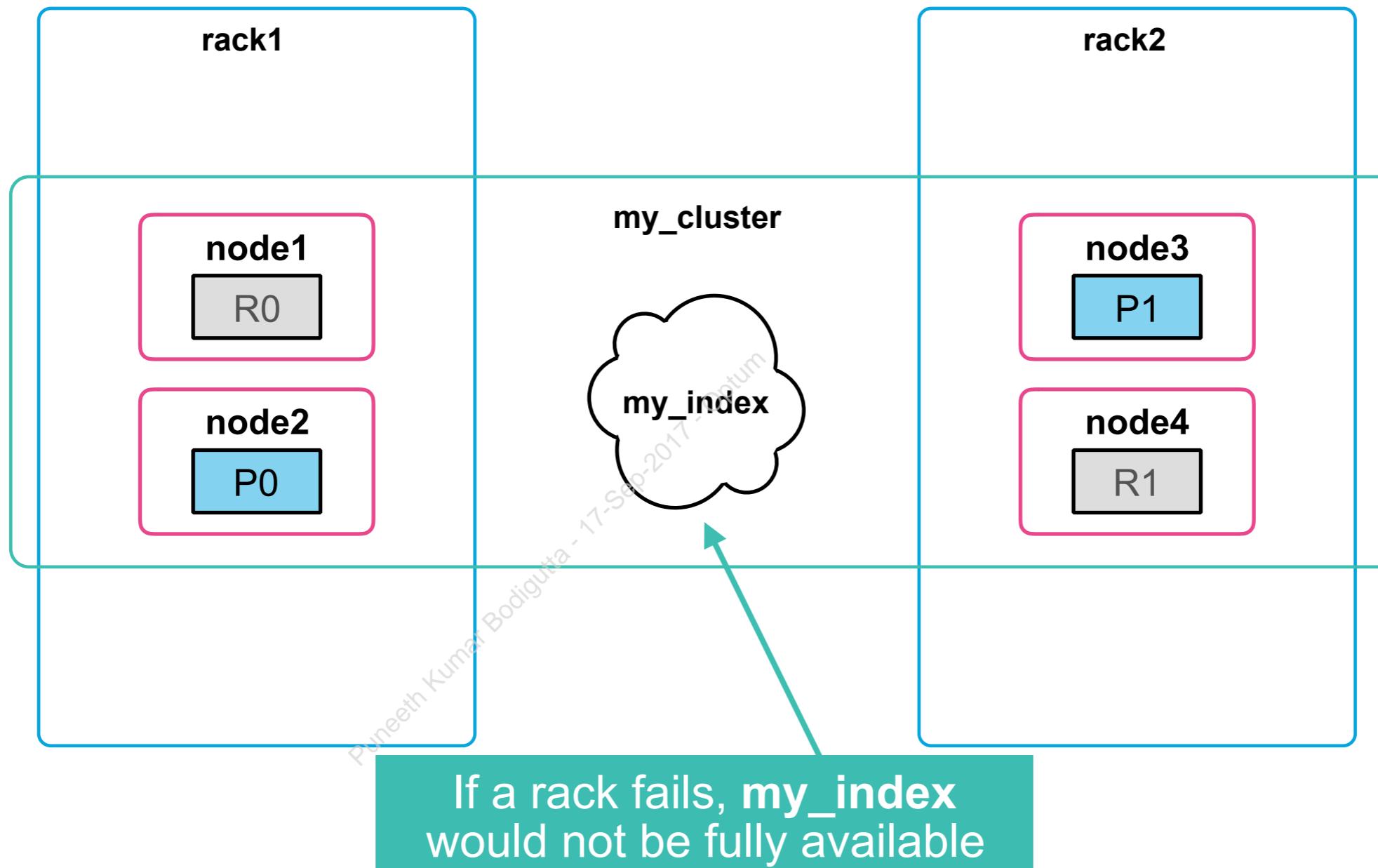
Shard Allocation Awareness

Puneeth Kumar Bodigutta - 17-Sep-2017 Optum



Awareness Example

- Suppose your hardware is spread across two racks (or zones, or VMs, or any grouping you have):



Shard Allocation Awareness

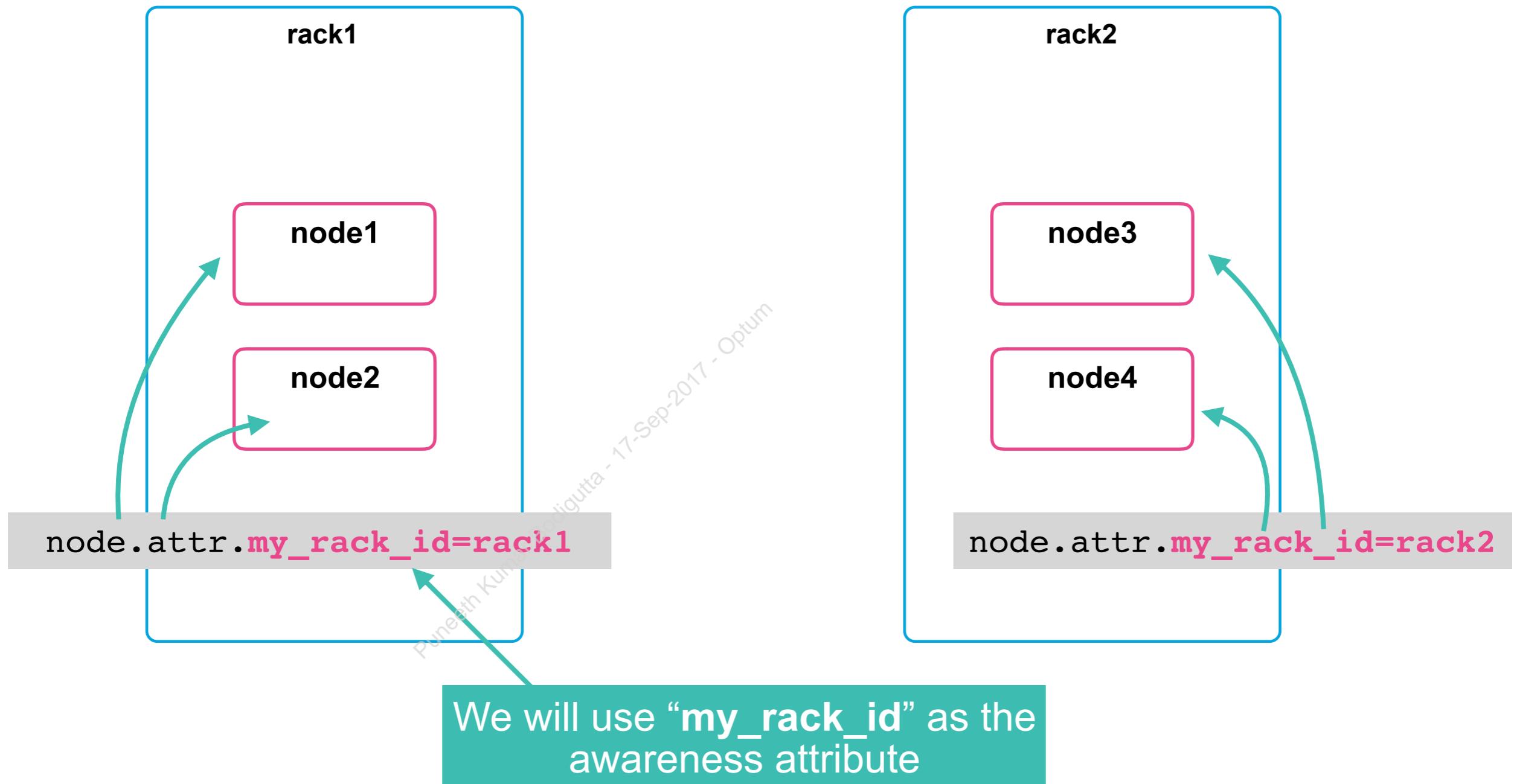
- You can make Elasticsearch aware of the physical configuration of your hardware
 - using **cluster.routing.allocation.awareness**, which is a cluster level setting
 - referred to as ***shard allocation awareness***
- Useful when more than one node shares the same resource (disk, host machine, network switch, rack, etc.)

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Step 1: Label Your Nodes

- Use `node.attr` to label your nodes:



Step 2: Configure Your Cluster

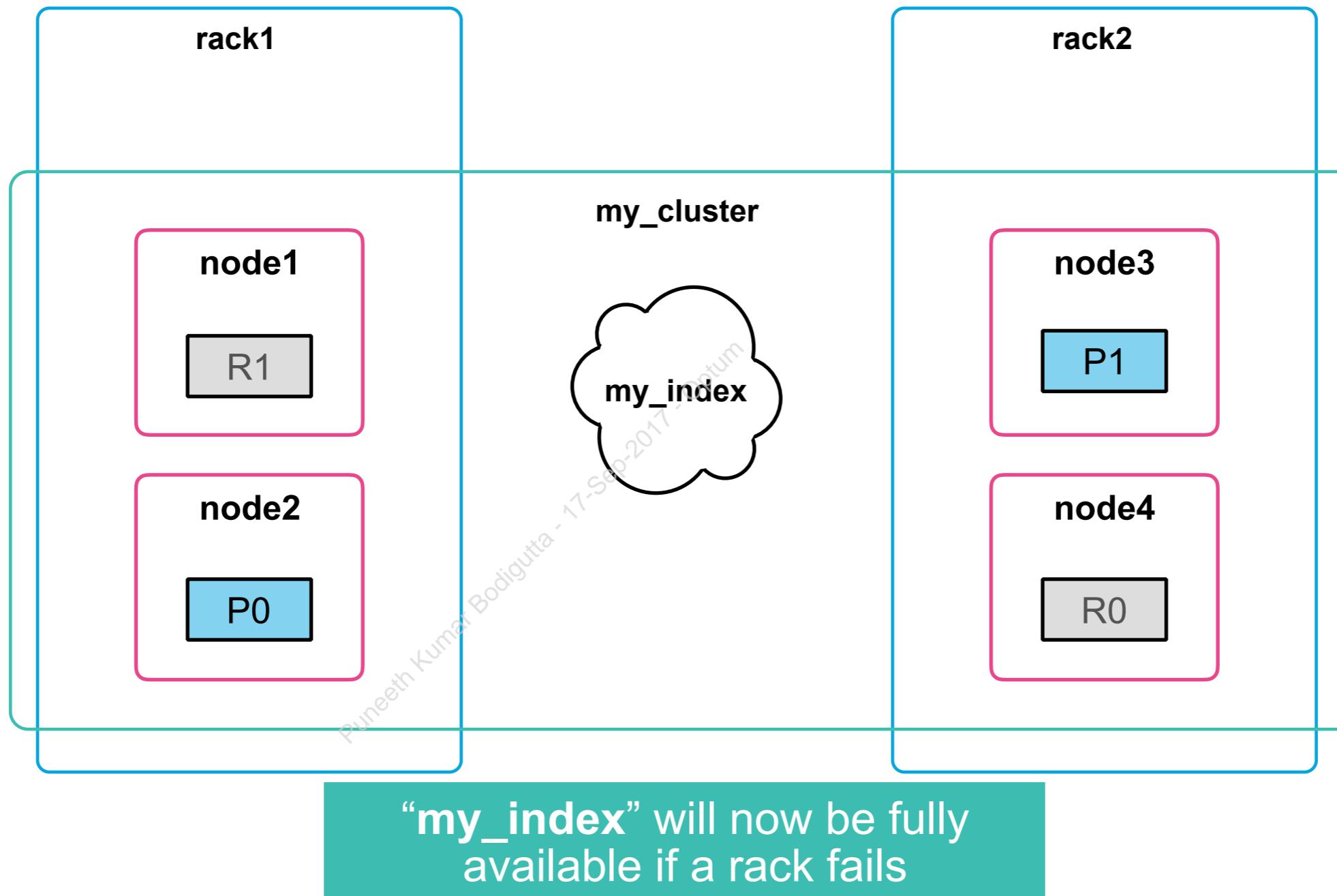
- You have to tell Elasticsearch which attribute (or attributes) are being used for awareness:

```
PUT _cluster/settings
{
  "persistent": {
    "cluster.routing.allocation.awareness.attributes": "my_rack_id"
  }
}
```

The name of the node
attribute you defined for
awareness

Awareness Example

- Now you are guaranteed that at least one copy of all shards will exist in each rack for each index:

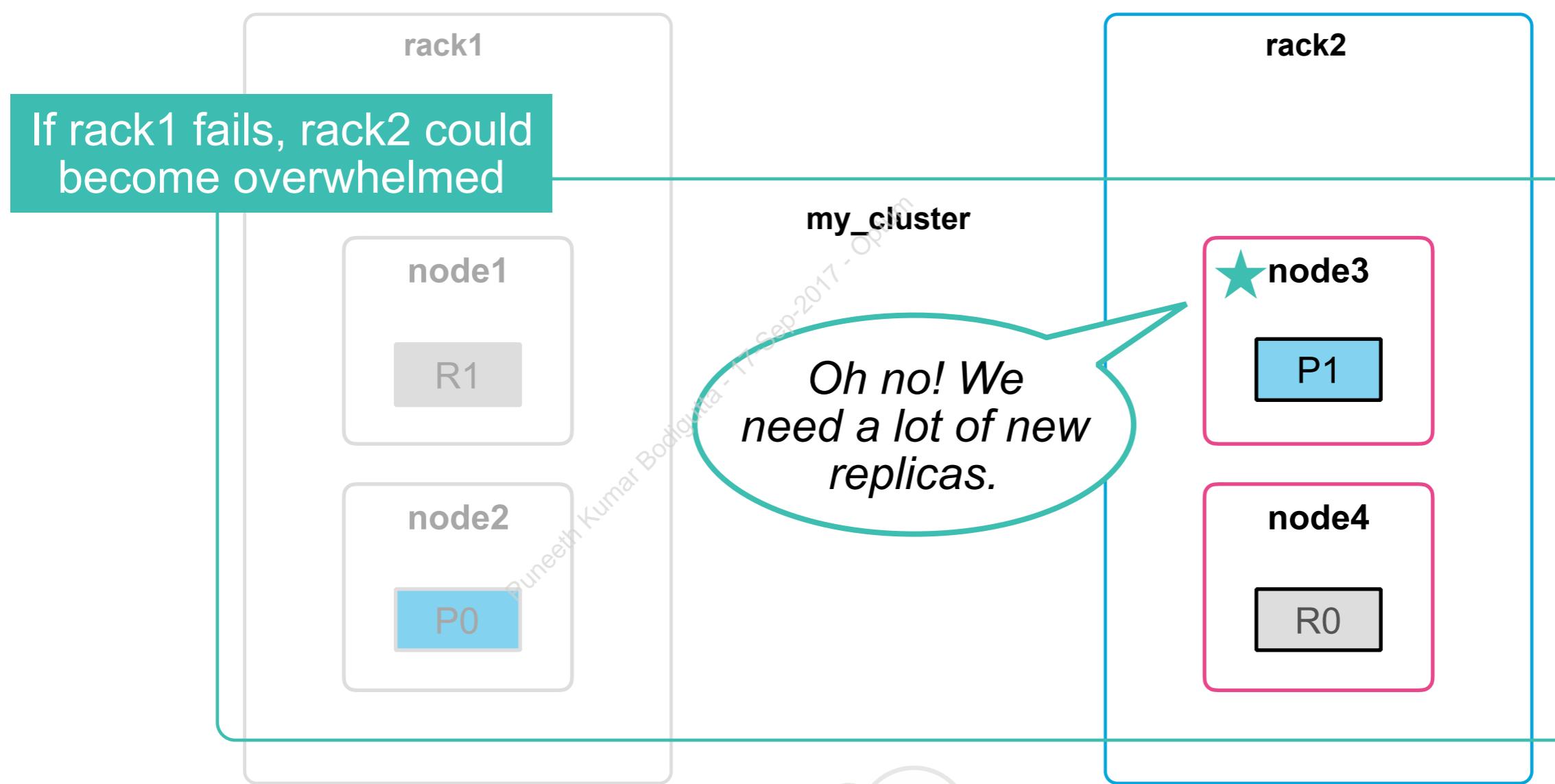


Forced Awareness

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

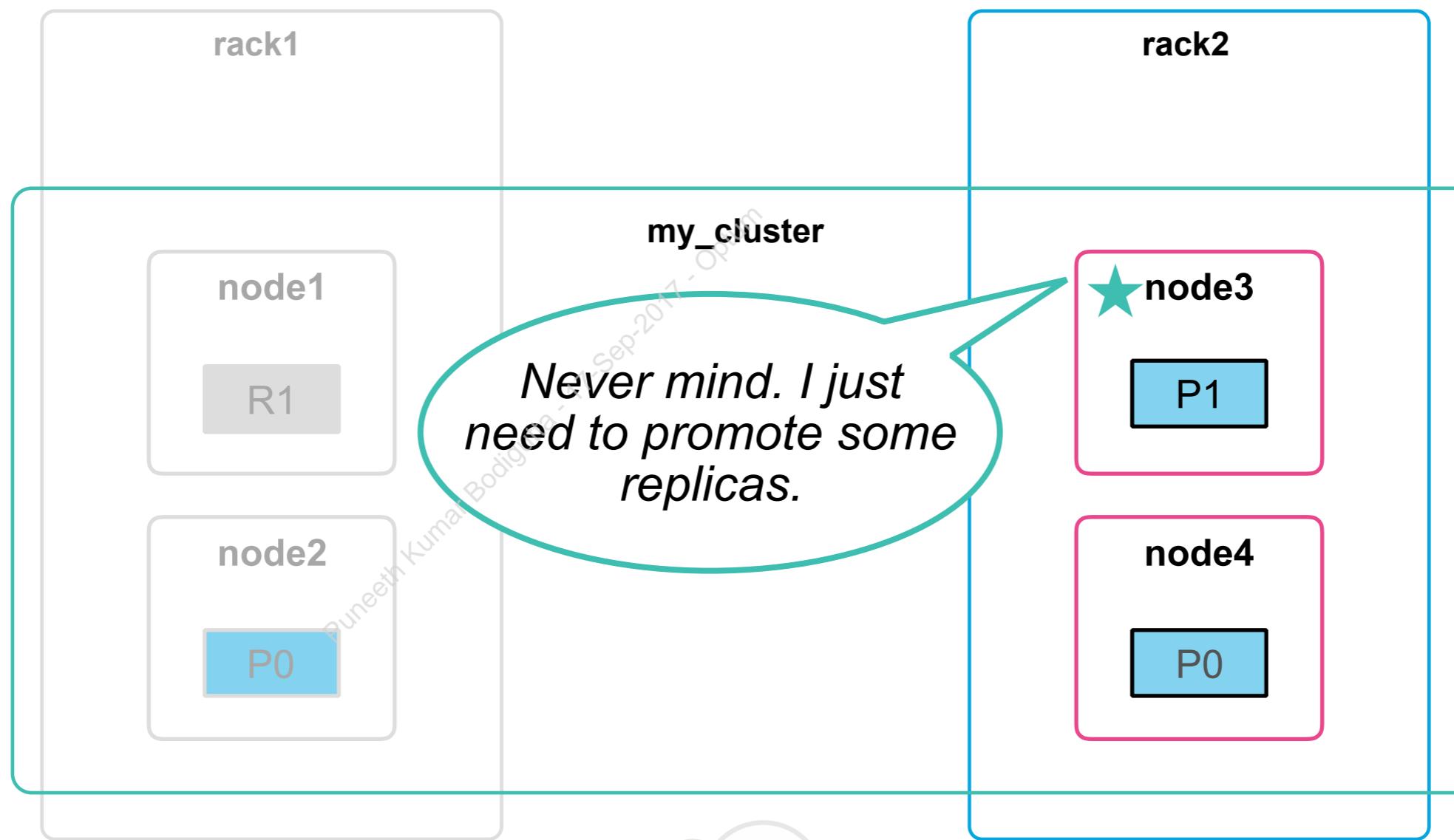
Why Forced Awareness?

- Suppose you have a rack (or zone) that fails
 - the remaining rack will attempt to reassign all the missing replicas
 - the single rack might not be able to handle that type of volume



Forced Awareness

- You can configure **forced awareness** to avoid overwhelming a zone
 - never allows copies of the same shard to be in the same zone



Configure Forced Awareness

- You have to tell Elasticsearch which attributes and values to use for forced awareness:
 - In this example, no copies of the same shard will appear on **rack1** and **rack2**

```
PUT _cluster/settings
{
  "persistent": {
    "cluster": {
      "routing": {
        "allocation.awareness.attributes": "my_rack_id",
        "allocation.awareness.force.my_rack_id.values": "rack1,rack2"
      }
    }
  }
}
```

Puneet Kumar Boddu

this setting is the **name** of
the attribute

this setting contains the
values of the attribute



Installing Plugins

Puneeth Kumar Bodigutta - 17-Sep-2017 - opsum

What are Plugins?

- **Plugins** are a way to enhance the core functionality of Elasticsearch in a custom manner
- **Core plugins** are a part of the Elasticsearch project
 - <https://www.elastic.co/guide/en/elasticsearch/plugins/current/intro.html>
 - there are community plugins available as well (use carefully!)
- Plugins consist of JAR files
 - and possibly scripts and config files

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Some Examples of Core Plugins

- **X-Pack**
 - extension of the Elastic Stack that includes Security, Alerting, Monitoring, Reporting and Graph
- **Discovery Plugins**
 - including EC2, Azure and Google Compute Engine
- **Analysis Plugins**
 - for adding new text analysis capabilities
- **Snapshot/Restore Plugins**
 - including S3, Azure, HDFS and Google Cloud Storage
- ...and many more

Installing Plugins

- Plugins are installed using the **elasticsearch-plugin** script
 - found in the **bin** folder of your Elasticsearch installation
 - used to **install**, **list** and **remove** plugins
 - plugins must be installed on every node
- Core plugins can be installed from elastic.co by simply providing the name of the plugin:

```
bin/elasticsearch-plugin install analysis-icu
```

one of the core plugins

Puneeth Kumar Boddu
17-Sep-2017 - Optum

Install from a File

- If the plugin has been downloaded, you can specify its location using a URL:

```
bin/elasticsearch-plugin install file:///home/elastic/x-pack.zip
```

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum



Viewing and Deleting Plugins

- Use **list** to view currently installed plugins:

```
bin/elasticsearch-plugin list
```

- Use **remove** to delete a plugin:

```
bin/elasticsearch-plugin remove analysis-icu
```

- After installing or removing a plugin, you will need to restart the node to complete the installation/removal process

Puneeth Kumar Bodigunta
ESL 2017 - Optum

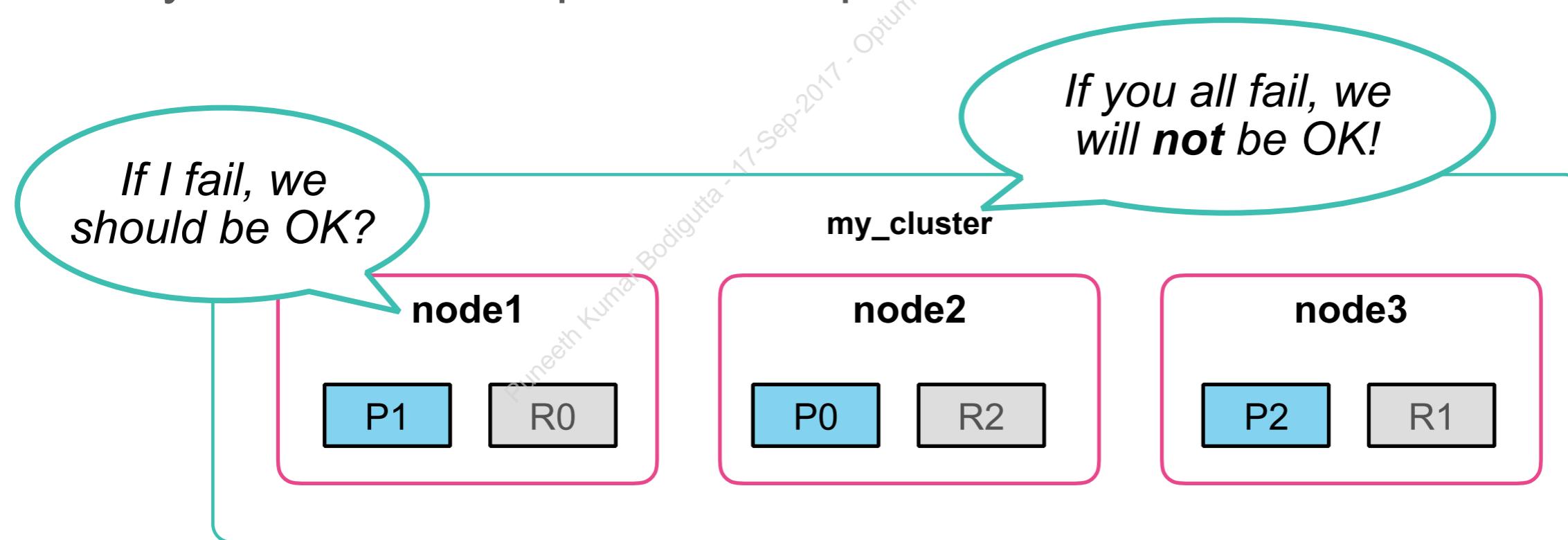


Cluster Backup

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

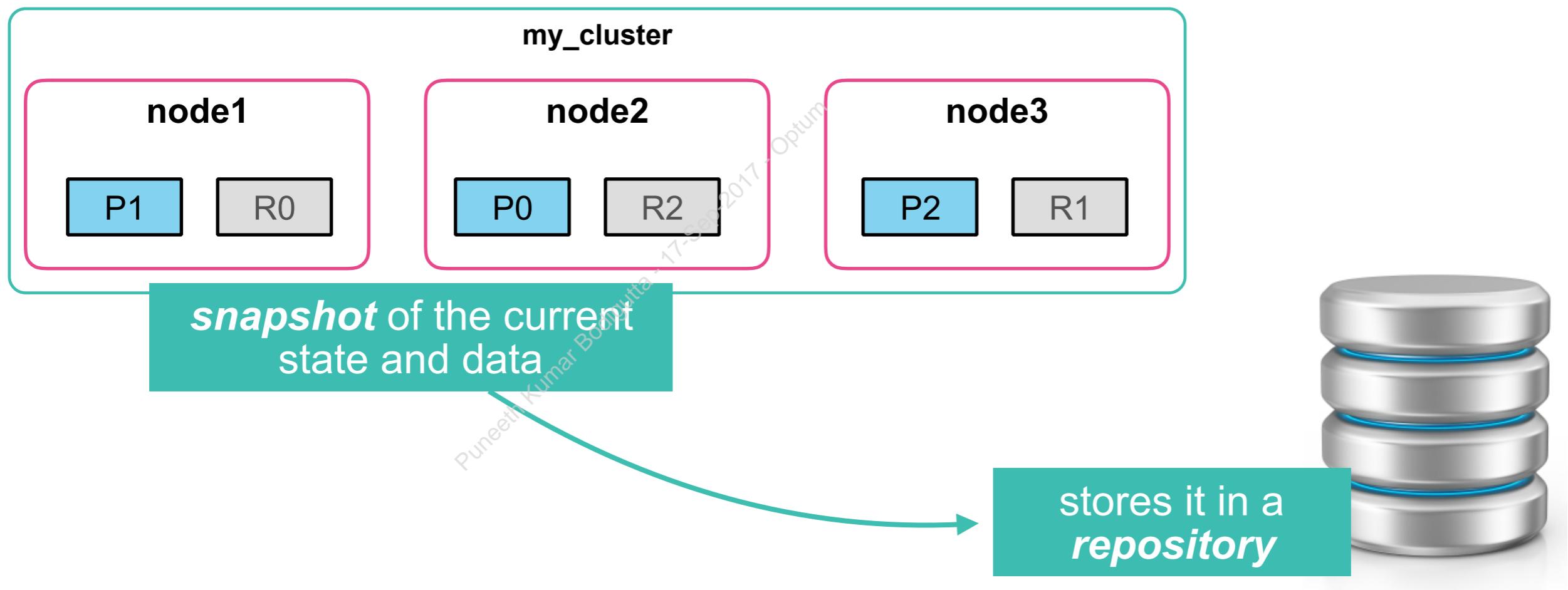
Cluster Backup

- We have talked a lot about *replica shards*
 - they provide replication of your documents
 - that is not the same as a backup!
- Replicas do not provide protection against catastrophic failure
 - you need a complete backup for those situations



Snapshot and Restore

- The ***Snapshot and Restore API*** provides a cluster backup mechanism
 - takes the current state and data in your cluster and saves it to a ***repository***
 - uses an incremental approach (discussed later)



Repositories

Puneeth Kumar Bodigutta - 17-Sep-2017 - Open

Repositories

- The backup process starts with the creation of a **repository**
 - You can configure multiple repositories per cluster
- Different types are supported:

Repository	Configuration type	
Shared file system	"type" : "fs"	
Read-only URL	"type" : "url"	
S3	"type" : "s3"	
HDFS	"type" : "hdfs"	
Azure	"type" : "azure"	
Google Cloud Storage	"type" : "gcs"	S3, HDFS, Azure and GCS require the appropriate plugin to be installed

Registering a Repository

- Before a snapshot can be performed, the repository needs to be *registered*
 - using the `_snapshot` endpoint
 - the folder must be accessible from all nodes in the cluster
 - `path.repo` must be configured on all nodes for an “fs” repo

register a filesystem repository
named “`my_repo`”

```
PUT _snapshot/my_repo
{
  "type": "fs",
  "settings": {
    "location": "/mnt/my_repo_folder"
  }
}
```

`path.repo: /mnt/my_repo_folder`
needs to be added to
`elasticsearch.yml` on all nodes

File System Repository Settings

- Other settings for an “fs” repo include:

```
PUT _snapshot/my_repo
{
  "type": "fs",
  "settings": {
    "location": "/mnt/my_repo_folder",
    "compress": true,
    "max_restore_bytes_per_sec": "40mb",
    "max_snapshot_bytes_per_sec": "40mb"
  }
}
```

compress metadata files

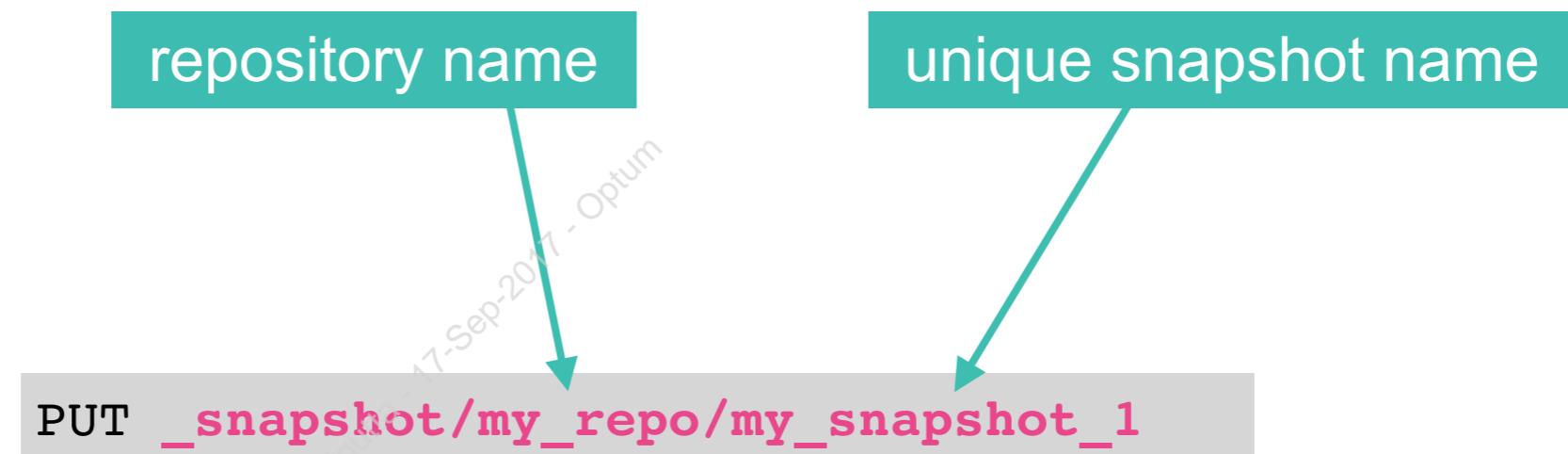
throttles per-node snapshot
and restore rates

Taking a Snapshot

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Snapshotting All Indices

- Once the repository is configured, you can take a snapshot using the `_snapshot` endpoint:
 - snapshots are a “point-in-time” copy of the data
 - only changes since the last snapshot are copied



No specific indices were listed, so this snapshot includes ***all open indices***

Specifying Indices

- Add the “**indices**” parameter to pick specific indices for the snapshot:

```
PUT _snapshot/my_repo/my_logs_snapshot_1
{
  "indices": "logs-*",
  "ignoreUnavailable": true,
  "includeGlobalState": true
}
```

snapshot of all open
indices that start with
“**logs-**”

This snapshot also ignores any
unavailable indices, and includes
the global state metadata

Monitoring Progress

- Use the `_status` endpoint to view the progress of a snapshot:

```
GET _snapshot/my_repo/my_snapshot_2/_status
```

- You can also wait for the snapshot to complete (but it may take a while):

```
PUT _snapshot/my_repo/my_logs_snapshot_2?wait_for_completion=true
{
  ...
}
```

Puneeth Kumar Bodigatta - 17-Sep-2017
Optum



Managing Snapshots

- Get information about *all snapshots* in a repo:

```
GET _snapshot/my_repo/_all
```

- Get information about a *specific snapshot*:

```
GET _snapshot/my_repo/my_snapshot_1
```

- *Delete* a snapshot:

```
DELETE _snapshot/my_repo/my_snapshot_1
```

Puneeth Kumar - Cutta - 17-Sep-2017 - Optum



Restoring from a Snapshot

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

Restoring from a Snapshot

- Use the `_restore` endpoint on the ID of the snapshot to restore all indices from a snapshot:

```
POST _snapshot/my_repo/my_snapshot_2/_restore
```

- You can also restore specific indices:

```
POST _snapshot/my_repo/my_snapshot_2/_restore
{
  "indices": "logs-*",
  "ignore_unavailable": true,
  "include_global_state": false
}
```

Renaming Indices

- You can restore old indices into new ones without replacing existing data
 - specify a “`rename_pattern`” and a “`rename_replacement`”:

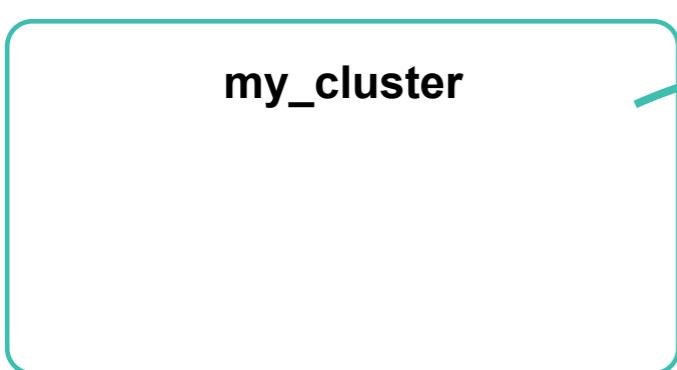
```
POST _snapshot/my_repo/my_snapshot_2/_restore
{
  "indices": "logs-*",
  "ignore_unavailable": true,
  "include_global_state": false,
  "rename_pattern": "logs-(.+)",
  "rename_replacement": "restored-logs-$1"
}
```

If an index starts with “`logs-*`”,
create a new index as
“`restored-logs-*`”

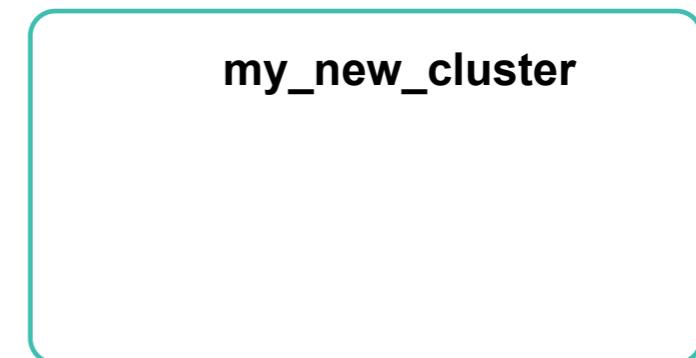
Restore to Another Cluster

- You can restore a snapshot made from one cluster into another cluster
 - you need to *register the repository* in the new cluster first

```
POST _snapshot/my_repo/snap1
```



my_repo



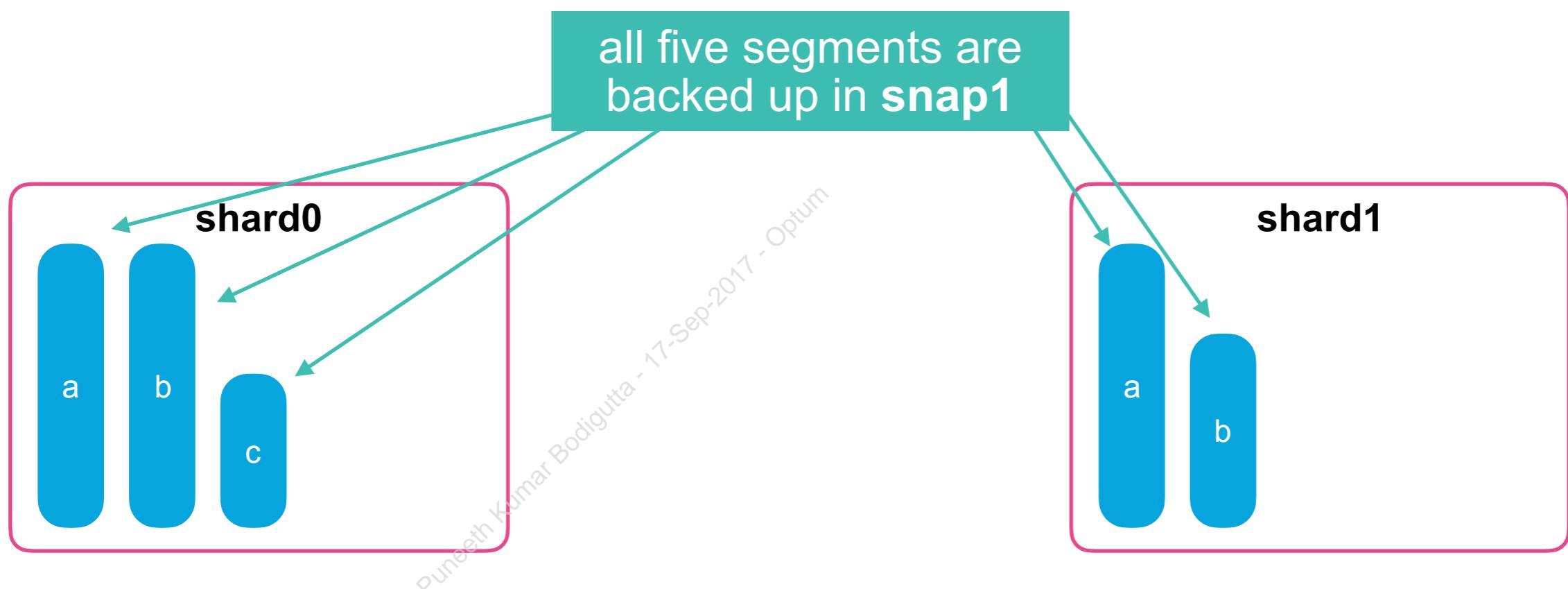
```
POST _snapshot/my_repo/snap1/_restore
```

The Incremental Nature of Snapshots

Puneeth Kumar Bodigutta - 17-Sep-2011, Optum

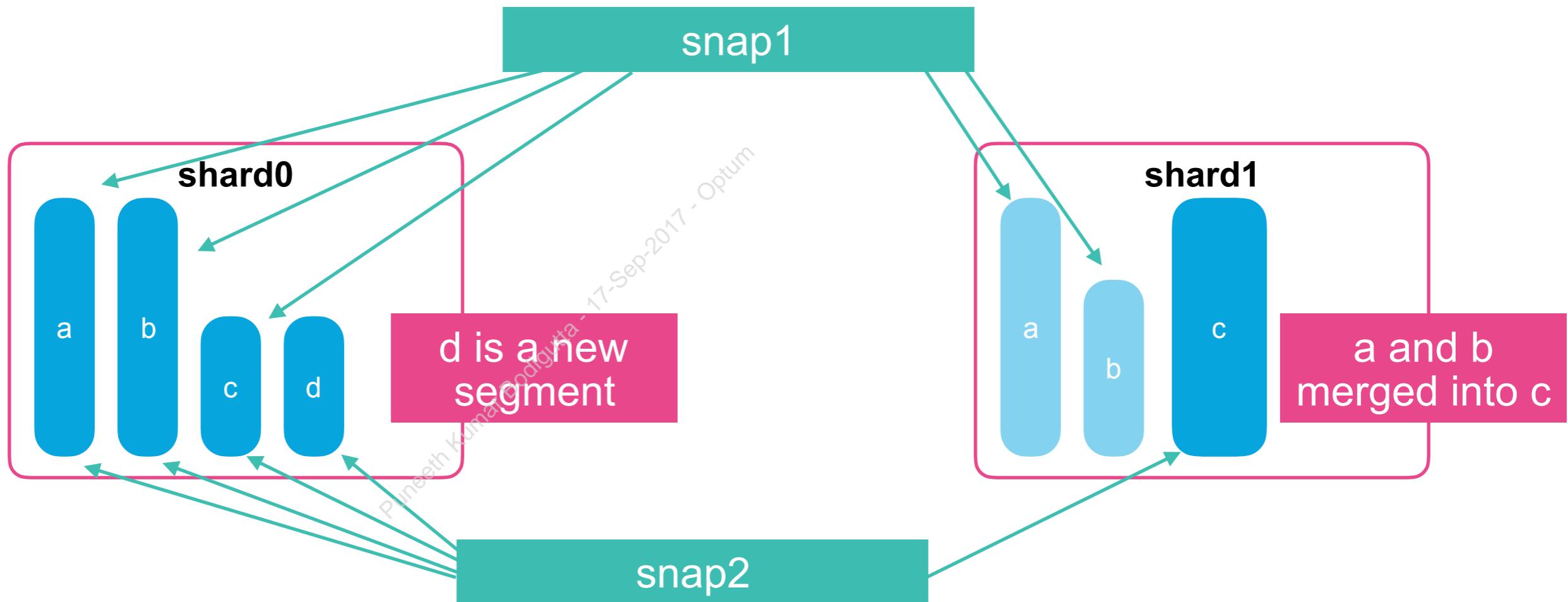
Incremental Nature of Snapshots

- Each snapshot is an increment of the previous one
 - It is important to understand what we mean by “*increment*”
- Suppose we take a snapshot (**snap1**) of a simple index with 2 shards and 5 total segments:



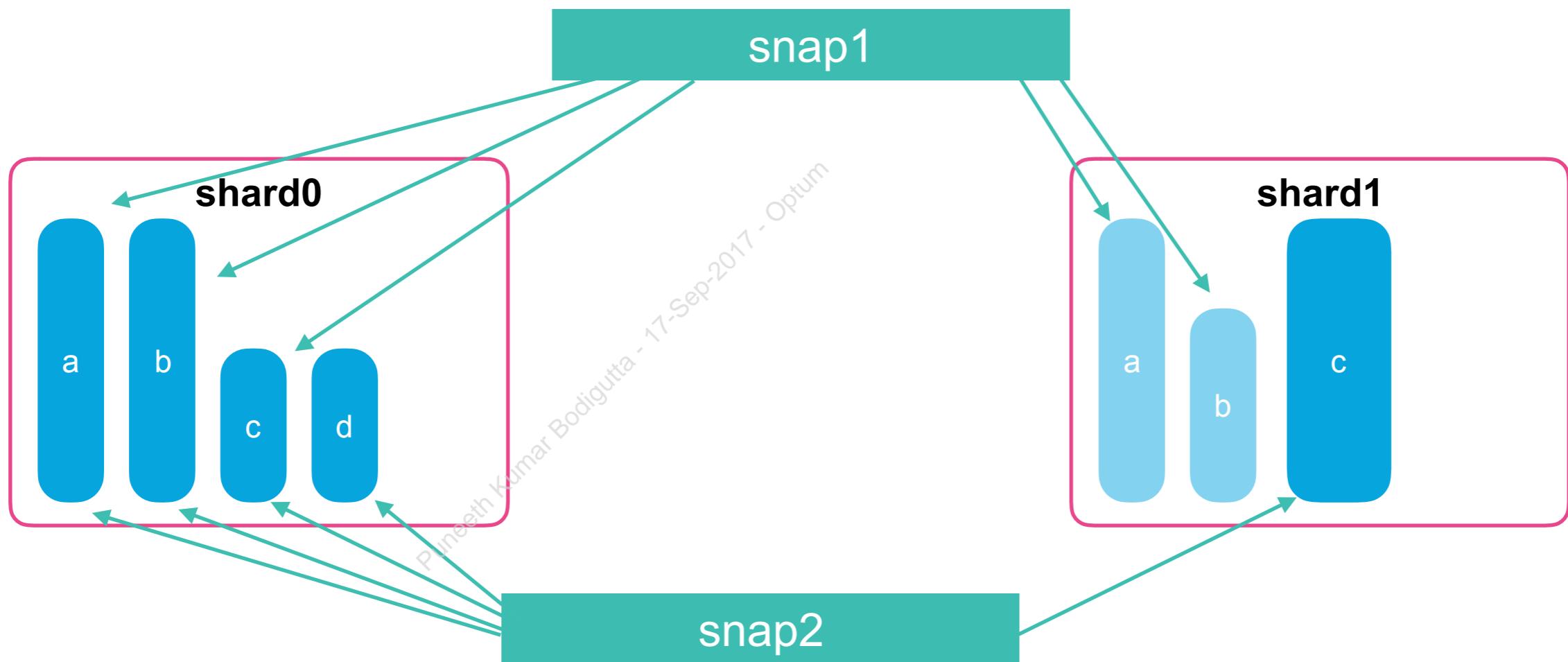
Incremental Nature of Snapshots

- As time passes, new segments are created and existing segments are merged:
 - suppose we take another snapshot (**snap2**)



Incremental Nature of Snapshots

- The “*increment*” for snap2 will be:
 - segment d on shard0
 - segment c on shard1



Increments are at the Segment Level

- The increments (in terms of documents) in a snapshot are the changes in **segments**
 - The difference between **snap1** and **snap2** is not the documents that were added or deleted
- Consider the scenario where no documents were indexed or deleted between **snap1** and **snap2**
 - **snap2** could still be quite large
 - Suppose **snap1** was 10 segments of 1gb each
 - Then a merge occurs that creates 1 segment of 7gb
 - **snap2** would have to copy the new segment containing 7gb

Puneeth Kumar Bodigunta 17-Sep-17 - Optimizing Elasticsearch

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- You can make Elasticsearch aware of the physical configuration of your hardware using **cluster.routing.allocation.awareness**
- You can configure **forced awareness** to avoid overwhelming a rack or zone of servers
- Plugins are installed using the **elasticsearch-plugin** script
- Core plugins can be installed from elastic.co by simply providing the name of the plugin
- The ***Snapshot and Restore API*** provides a cluster backup mechanism
- Before a snapshot can be performed, its repository needs to be registered
- You can snapshot and restore all indices, or you can choose specific indices
- The snapshot increments of a cluster's data are the changes in **segments**

Puneeth Kumar Bodigutta - ITES 2017, Guntur

Quiz

1. Why configure shard allocation awareness if you have already configured shard allocation filtering?
2. What is the benefit of forced awareness over simply configuring shard allocation awareness?
3. **True or False:** Configuring all indices to have 2 or more replicas provides a reliable backup mechanism for a cluster.
4. What is the first step in creating a snapshot?
5. **True or False:** A snapshot is a point-in-time copy of the data; changes made to the cluster while the snapshot is being created will not appear in the snapshot.
6. **True or False:** Suppose you take a snapshot, delete two indices, then take another snapshot. The second snapshot will be smaller in size than the first one due to the incremental nature of snapshots.



Lab 9

Cluster Management

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open

Chapter 10

Monitoring

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Monitoring Options
- The Stats APIs
- Task Monitoring
- The cat API
- The X-Pack Monitoring Component
- The Monitoring UI
- Monitoring Guidelines

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Monitoring Options

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

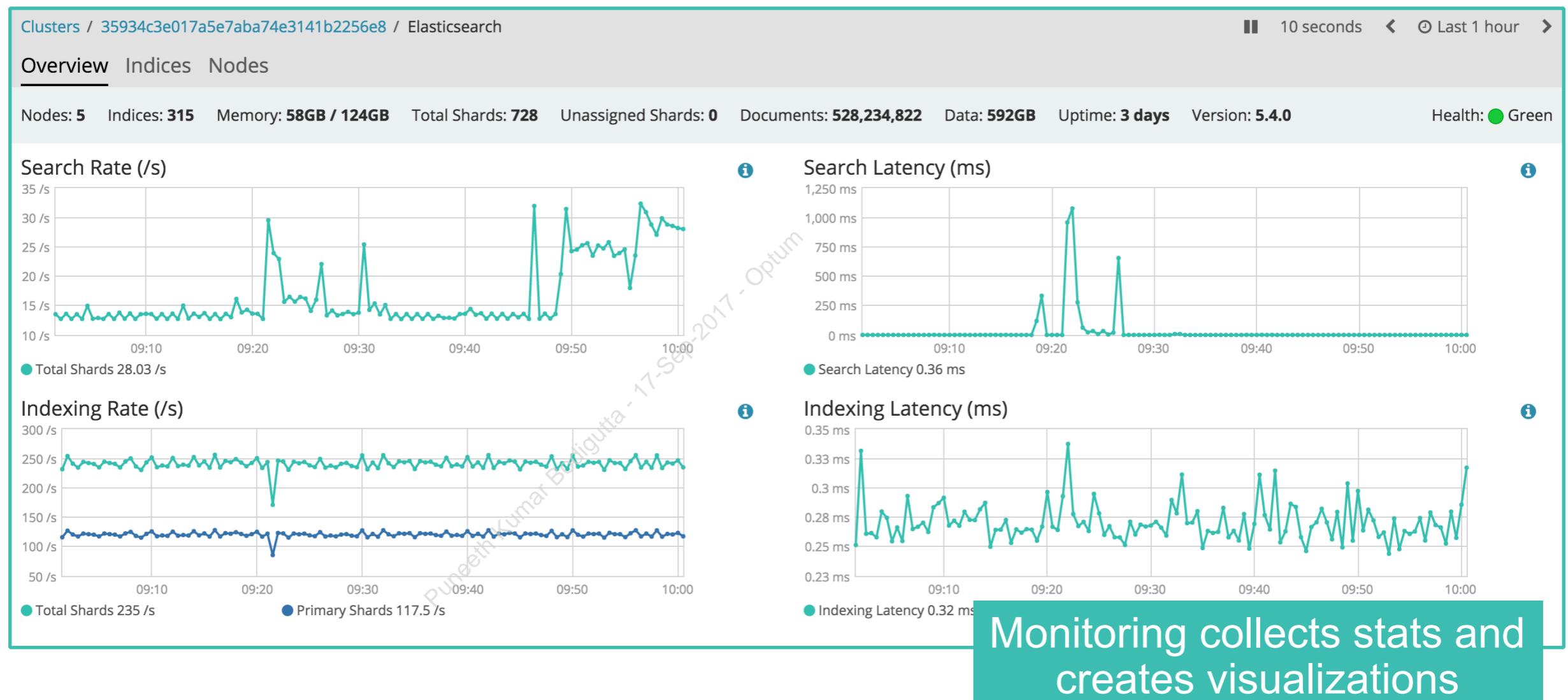
Monitoring Options

- Elasticsearch has several APIs for monitoring:
 - **Node Stats:** `_nodes/stats`
 - **Cluster Stats:** `_cluster/stats`
 - **Index Stats:** `my_index/_stats`
 - **Pending Cluster Tasks API:** `_cluster/pending_tasks`
- The above APIs return JSON objects (no surprise)
 - **cat API:** a human-readable alternative to the JSON APIs above
 - similar results, just formatted differently

Puneet Kumar Boddu - 17-Sep-2017 - Optum

Monitoring (the Component)

- While useful, the stats APIs are *point-in-time*
 - That is where *Monitoring* comes in to play (not the verb “monitoring”, but the X-Pack component called “Monitoring”)



The Stats APIs

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Cluster Stats

- The **Cluster Stats API** provides stats at the cluster level:

GET `_cluster/stats`

Only a very small subset of
the large amount of stats

```
{  
  "_nodes": {  
    "total": 2,  
    "successful": 2,  
    "failed": 0  
  },  
  "cluster_name": "my_cluster",  
  "timestamp": 1486701112018,  
  "status": "red",  
  "indices": {  
    "count": 26,  
    "shards": {  
      "total": 162,  
      "primaries": 87,  
      "replication": 0.8620689655172413,  
      "index": {  
        "shards": {  
          "min": 2,  
          "max": 10,  
          "avg": 6.230769230769231  
        },  
        "primaries": {  
          "min": 1,  
          "max": 6,  
          "avg": 3.3461538461538463  
        },  
        "replication": {  
          "min": 1,  
          "max": 6,  
          "avg": 3.3461538461538463  
        }  
      }  
    }  
  }  
}
```

Node Stats

- The **Node Stats API** provides stats at the node level:

GET `_nodes/stats`

You can specify a list of nodes also

GET `_nodes/node1/stats`

```
{  
  "_nodes": {  
    "total": 2,  
    "successful": 2,  
    "failed": 0  
  },  
  "cluster_name": "my_cluster",  
  "nodes": {  
    "OmWJPhToQ0iyNfz-Qd9i8g": {  
      "timestamp": 1486701705225,  
      "name": "node1",  
      "transport_address": "192.168.1.6:9300",  
      "host": "192.168.1.6",  
      "ip": "192.168.1.6:9300",  
      "roles": [  
        "master",  
        "data"  
      ],  
      "attributes": {  
        "temp": "hot",  
        "server_size": "small",  
        "zone": "zoneA"  
      },  
      ...  
    }  
  }  
}
```

Indices Stats

- The *Indices Stats API* provides stats at the index level:

GET `my_index/_stats`

You can get the stats from all indices in one request:

GET `_stats`

```
{  
  "_shards": {  
    "total": 10,  
    "successful": 10,  
    "failed": 0  
  },  
  "_all": {  
    "primaries": {  
      "docs": {  
        "count": 2,  
        "deleted": 0  
      },  
      "store": {  
        "size_in_bytes": 7399,  
        "throttle_time_in_millis": 0  
      },  
      "indexing": {  
        "index_total": 0,  
        "index_time_in_millis": 0,  
        "index_current": 0,  
        "index_failed": 0,  
        "delete_total": 0,  
        "delete_time_in_millis": 0,  
        ...  
      }  
    }  
  }  
}
```

Task Monitoring

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Pending Tasks

- The *Pending Tasks API* shows cluster-level changes that have not been executed yet:

```
GET _cluster/pending_tasks
```

```
{  
  "tasks": [  
    {  
      "insert_order": 101,  
      "priority": "URGENT",  
      "source": "create-index [my_index], cause [api]",  
      "time_in_queue_millis": 86,  
      "time_in_queue": "86ms"  
    }  
  ]  
}
```

The response is often empty because cluster-level changes are fast

Task Management API

- The **Task Management API** shows tasks currently executing on the nodes
 - provides a nice view of how busy the cluster is

GET _tasks



You can also use this API to cancel a task

Puneeth Kumar Bodiguttu - 17-Sep-2017

```
{  
  "nodes": {  
    "OmWJPhToQ0iyNfz-Qd9i8g": {  
      "name": "node1",  
      "transport_address": "192.168.1.6:9300",  
      "host": "192.168.1.6",  
      "ip": "192.168.1.6:9300",  
      "roles": [  
        "master",  
        "data"  
      ],  
      "tasks": {  
        "OmWJPhToQ0iyNfz-Qd9i8g:37432": {  
          "node": "OmWJPhToQ0iyNfz-Qd9i8g",  
          "id": 37432,  
          "type": "direct",  
          "action": "cluster:monitor/tasks/lists[n]",  
          "start_time_in_millis": 1486702376488,  
          "running_time_in_nanos": 2157349,  
          "cancellable": false,  
          "parent_task_id": "OmWJPhToQ0iyNfz-Qd9i8g:37431"  
        },  
        "OmWJPhToQ0iyNfz-Qd9i8g:37433": {  
          "node": "OmWJPhToQ0iyNfz-Qd9i8g",  
          "id": 37433,  
          "type": "direct",  
          "action": "cluster:monitor/tasks/lists[n]",  
          "start_time_in_millis": 1486702376488,  
          "running_time_in_nanos": 2157349,  
          "cancellable": false,  
          "parent_task_id": "OmWJPhToQ0iyNfz-Qd9i8g:37431"  
        }  
      }  
    }  
  }  
}
```

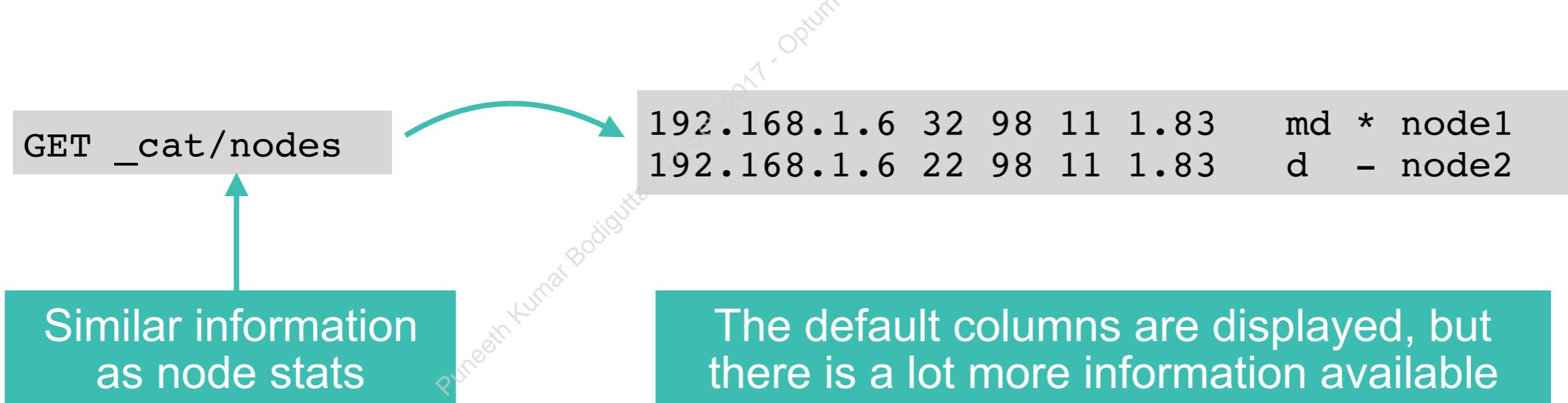


The cat API

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized

The cat API

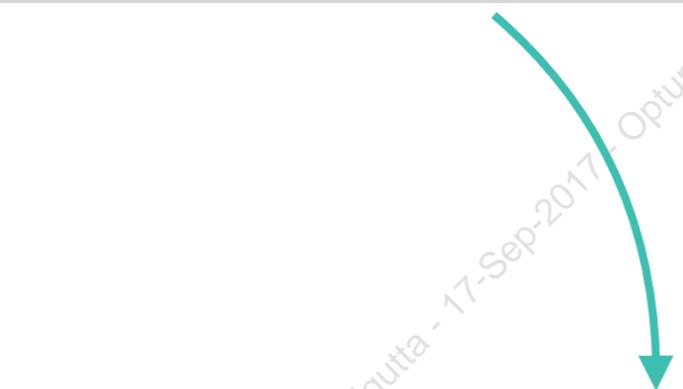
- You have seen the `_cat` API throughout the labs
 - it is a wrapper around many of the Elasticsearch JSON APIs, including the stats APIs discussed so far in this chapter
 - command-line tool friendly
 - helpful for simple monitoring (e.g. Nagios)



Specifying Columns

- Add the “h” parameter to specify which columns to return
 - use “*” to retrieve all columns
 - add the “v” parameter to display the column names in the response

```
GET _cat/nodes?v&h=name,disk.avail,search.query_total,heap.percent
```



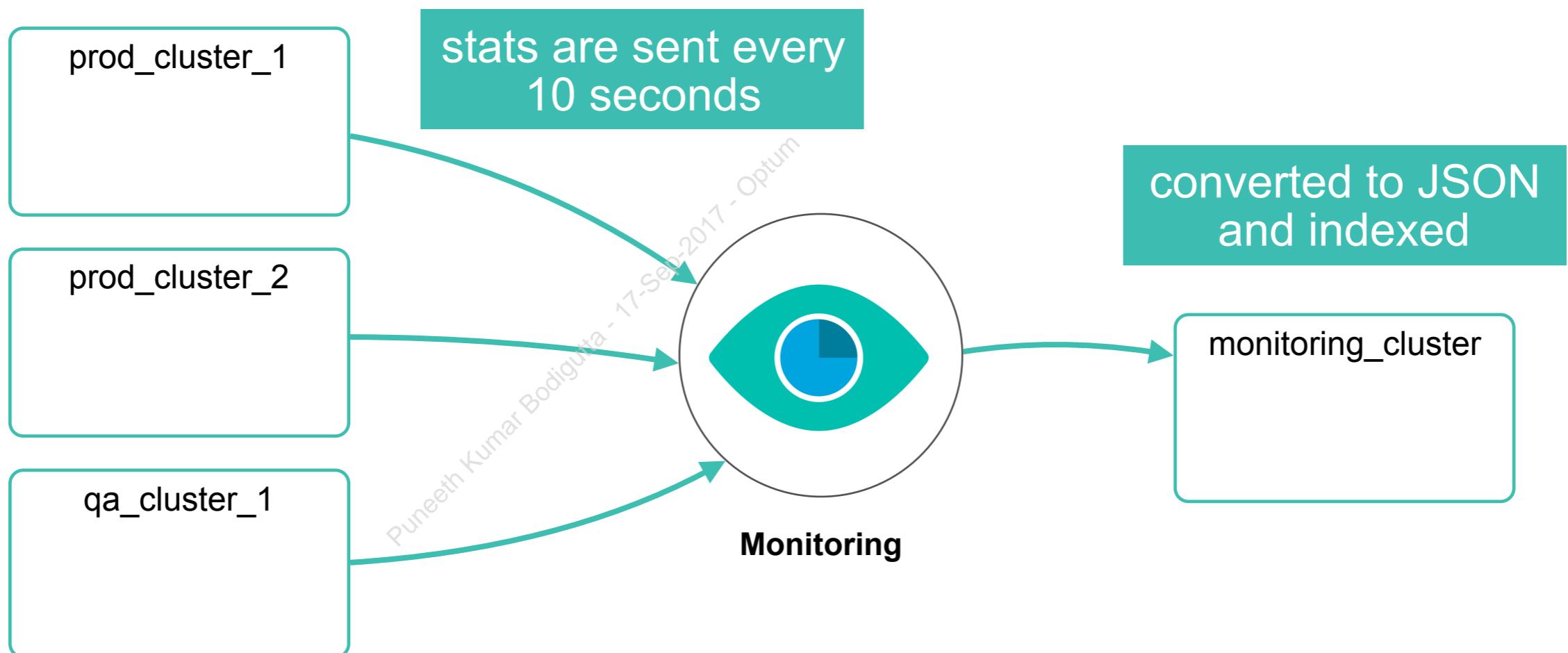
name	disk.avail	search.query_total	heap.percent
node1	144.4gb	3800	43
node2	144.4gb	0	26

The X-Pack Monitoring Component

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

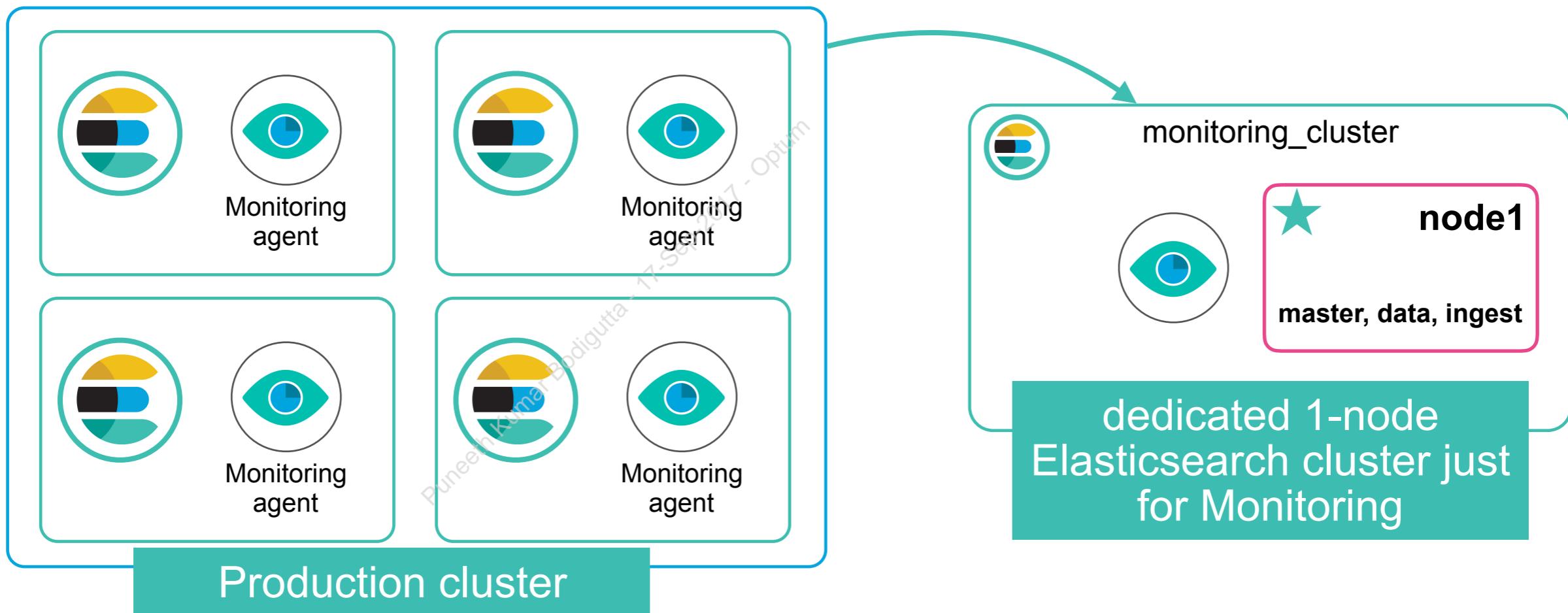
Monitoring

- **Monitoring** uses Elasticsearch to monitor Elasticsearch
 - the stats of all the monitored clusters are indexed into Monitoring's Elasticsearch cluster
 - **xpack.monitoring.collection.interval** defaults to 10 seconds



Dedicated Cluster

- Recommend using a ***dedicated cluster*** for Monitoring
 - reduce the load and storage on your other clusters
 - access to Monitoring even when other clusters are unhealthy
 - separate security levels from Monitoring and production clusters



Install Monitoring

- Installed as a part of the X-Pack plugin

```
ES_HOME/bin/elasticsearch-plugin install x-pack
```

- You will also need to install the X-Pack plugin for Kibana:

```
KIBANA_HOME/bin/kibana-plugin install x-pack
```

- Installing X-Pack starts a monitoring agent on each node
 - for collecting the metrics

Puneeth Kumar Duggutta - 11 Sep 2017 - Optum

Configuring Monitoring

- If Monitoring is on a different cluster than the one being monitored, you need to tell the monitored cluster where to send its stats:
 - if X-Pack security is enabled on the Monitoring cluster, then provide credentials
 - you can also use SSL/TLS (see docs for details:
<https://www.elastic.co/guide/en/x-pack/current/monitoring-cluster.html>)

configure in `elasticsearch.yml`
of each node

```
xpack.monitoring.exporters:  
  id1:  
    type: http  
    host: ["http://monitoring_cluster:9200"]  
    auth.username: username  
    auth.password: changeme
```

Configuring Monitoring

- Here are a few common Monitoring settings
 - which can be configured in `elasticsearch.yml`:

<code>xpack.monitoring.collection.indices</code>	The indices to collect data from. Defaults to all indices, but can be a comma-separated list.
<code>xpack.monitoring.collection.interval</code>	How often data samples are collected. Defaults to 10s
<code>xpack.monitoring.history.duration</code>	How long before indices created by Monitoring are automatically deleted. Defaults to 7d

- The complete list of Monitoring settings is at
 - <https://www.elastic.co/guide/en/x-pack/current/monitoring-settings.html>

The Monitoring UI

Puneeth Kumar Bodigutta - 17-Sep-2017 . Open

The Monitoring UI

- Open up Kibana
 - you will have to login now because X-Pack comes with Security
 - username is “**elastic**” and password is “**changeme**”
- Click on the **Monitoring** shortcut in the left toolbar:

A screenshot of the Elasticsearch Monitoring UI. On the left, a vertical toolbar lists several monitoring shortcuts: Clusters, my_cluster, Your Trial license will expire on March 11, 2017, Elasticsearch (highlighted in teal), Status, Overview, Version: 5.2.0, Uptime: 9 minutes, Kibana (highlighted in pink), Status, Overview, Requests: 106, Max. Response Time: 186 ms. A red box with the text "Click here:" and an arrow points to the "Elasticsearch" icon in the toolbar. The main dashboard shows cluster statistics: Nodes: 2 (Disk Available: 144GB / 233GB (61.97%), JVM Heap: 29.53% (585MB / 2GB)), Indices: 29 (Documents: 3,218, Disk Usage: 3MB, Primary Shards: 90, Replica Shards: 78), and Kibana instances: 1 (Connections: 12, Memory Usage: 13.27% (190MB / 1GB)).

Clusters Dashboard

- The initial Monitoring page is the “Clusters” dashboard
 - shows all the clusters being monitored
 - notice it is monitoring your Kibana instances as well

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Clusters

my_cluster

Your Trial license will expire on [March 11, 2017](#).

Elasticsearch

Status

Overview

Version: 5.2.0
Uptime: 3 hours

Kibana

Status

Overview

Requests: 1188
Max. Response Time: 93 ms

Nodes: 2

Disk Available: 143GB / 233GB (61.39%)
JVM Heap: 29.37% (581MB / 2GB)

Indices: 29

Documents: 36,412
Disk Usage: 40MB
Primary Shards: 90
Replica Shards: 78

Instances: 1

Connections: 12
Memory Usage: 6.95% (100MB / 1GB)

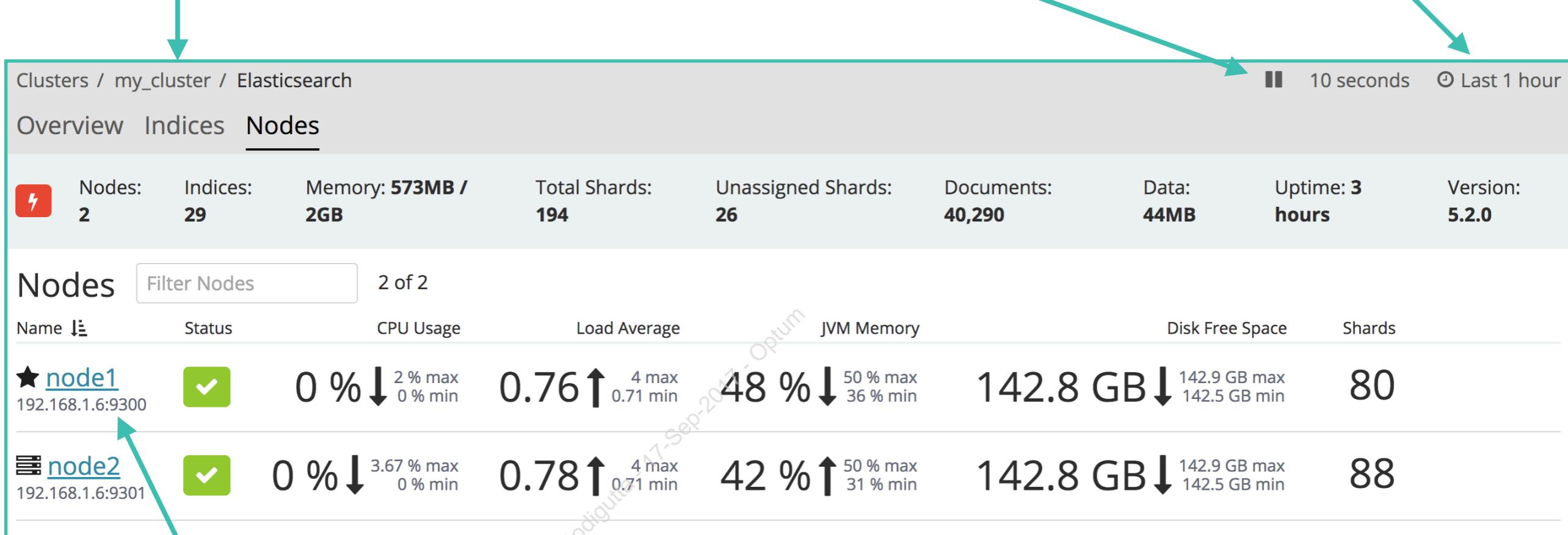
Note the free version of Monitoring can only monitor a single cluster

Nodes Dashboard

breadcrumbs for easy navigation

polling interval (and a handy “pause” button)

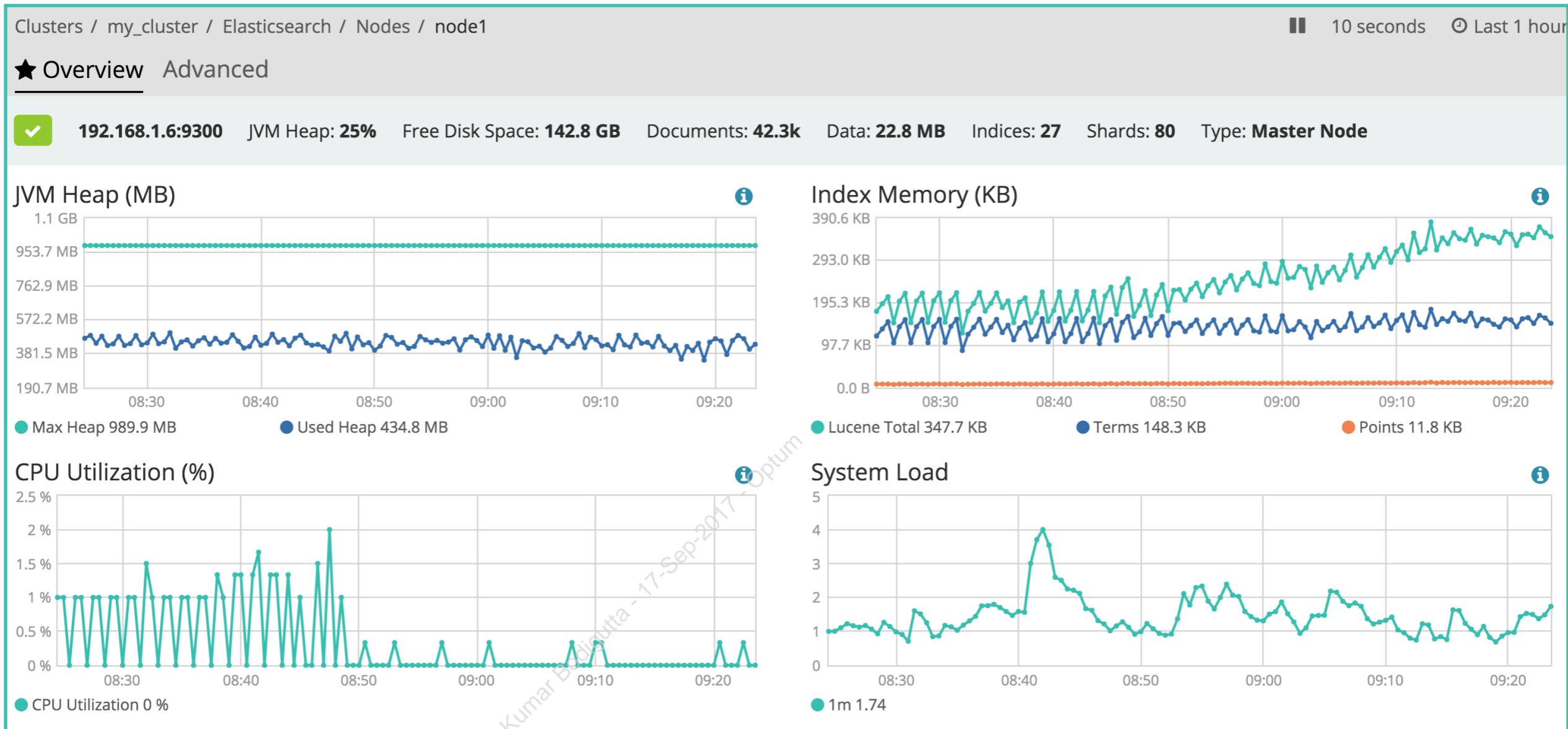
Time interval of what you are currently viewing



master node has the “star”

stats update in real-time

Individual Node Dashboard



the actual page has more details

Indices Dashboard

quick way to find unhealthy indices

Clusters / my_cluster / Elasticsearch								10 seconds	Last 1 hour			
Overview		Indices		Nodes								
Nodes:	Indices:	Memory:	867MB / 2GB	Total Shards:	194	Unassigned Shards:	26	Documents:	46,791	Data:	51MB	
Nodes: 2	Indices: 29	Memory: 867MB / 2GB	Total Shards: 194	Unassigned Shards: 26	Documents: 46,791	Data: 51MB	Uptime: 3 hours	Version: 5.2.0				
Indices	Filter Indices		20 of 25	<input type="checkbox"/> Show system indices	Puneeth Kumar Bodiguttu 17-Sep-2017 - Optum	Name	Status	Document Count	Data	Index Rate	Search Rate	Unassigned Shards
my_tweets		!	4	23.7 KB	0 /s	0 /s	2					
logs-old		✓	2	8.3 KB	0 /s	0 /s	0					
my_new_index		✓	2	13.1 KB	0 /s	0 /s	0					
restored-logs-old		✓	2	8.3 KB	0 /s	0 /s	0					

Lots of Stats!

- Between the Stats APIs and the Monitoring UI
 - there are a lot of statistics to monitor!
- It can be overwhelming to have access to this much information
 - so let's go over some general guidelines on what and how to monitor...

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Monitoring Guidelines

Puneeth Kumar Bodigutta - 17-Sep-2017, Optum

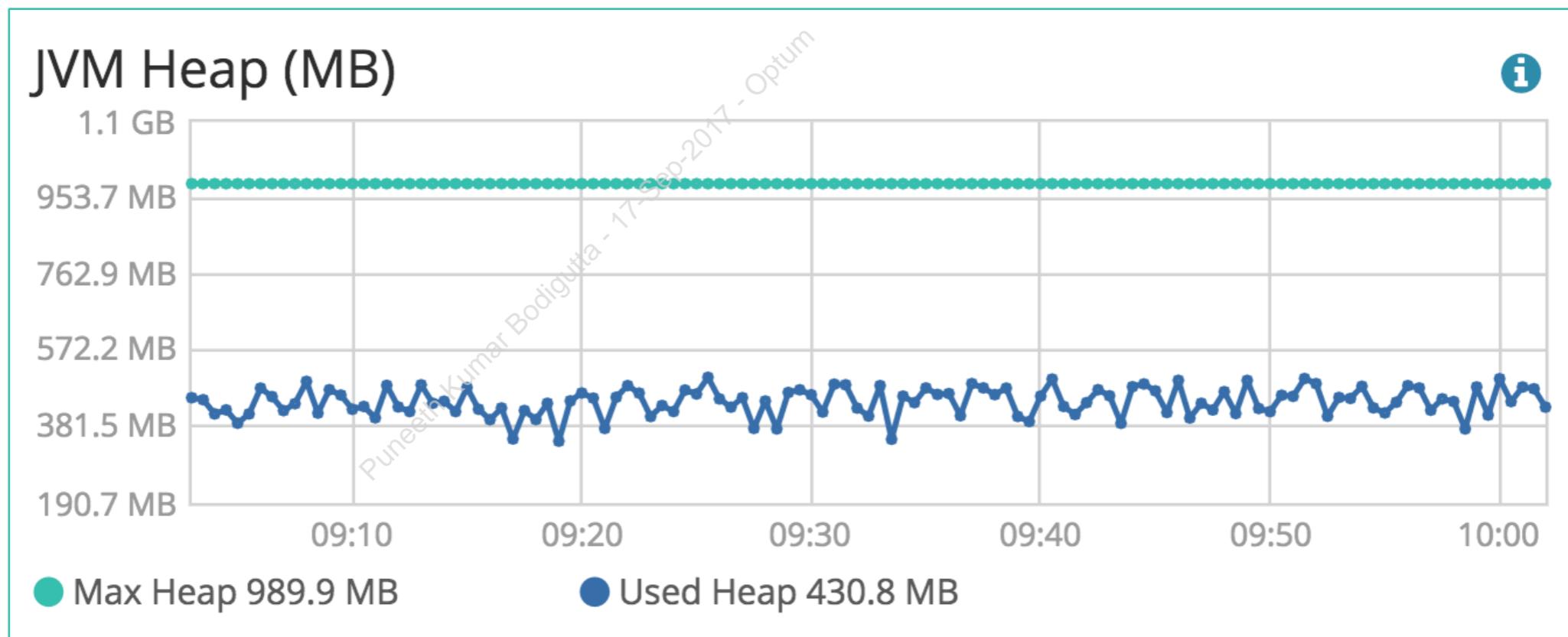
Configure Alerts

- You should use some type of monitoring tool to ***fire alerts*** for unexpected or extreme situations
 - send an alert, email, page, etc.
 - Nagios is a popular tool
 - lots of third-party tools available
- Or you can use ***Alerting***, which is a part of the X-Pack plugin
 - <https://www.elastic.co/guide/en/x-pack/current/xpack-alerting.html>

Puneeth Kumar Bodigutta - APR 2017 - Optum

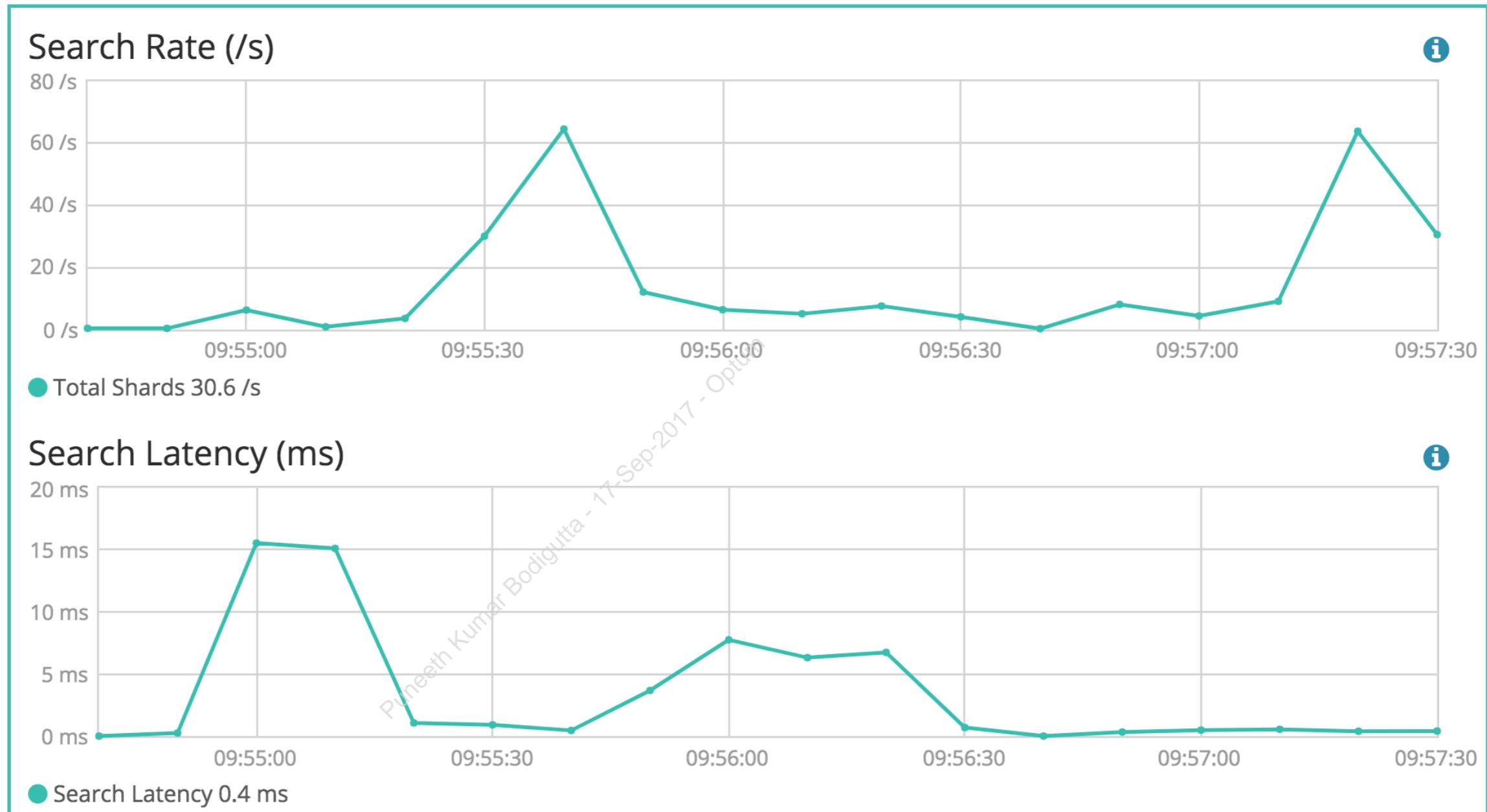
JVM Heap Usage

- Monitor **JVM Heap** usage on every node
 - if it goes above a certain percentage, fire a **warning**
 - if it goes above a higher percentage, fire a **page**
- Use Monitoring to establish a baseline percentage
 - also watch for healthy vs. unhealthy garbage collection



Monitoring for Query Performance

- Monitoring has time series charts for viewing the performance of your queries:



Handy Query Performance Commands

- To get the **wire** time:

```
(time curl -s -XGET host:9200/tweets*/_search) 2>&1 > /dev/null | grep real | awk '{print $2}'
```

0m0.025s

- To get the **took** time:

```
curl -s -XGET 'host:9200/logstash-*/_search?pretty' | head | grep '"took"' | awk '{print $3}'
```

10

Puneeth Kumar Bodigutta - 11/11/2017 - QnA

The Search Slow Log

- The **Search Slow Log** captures information about *long-running searches* (query and fetch phases) into a log file
 - The log file is already configured in **log4j2.properties**, but disabled by default
 - Useful, but can be limited since it logs per shard
 - Packetbeat may be a better solution

```
PUT my_index/_settings
{
  "index.search.slowlog.threshold" : {
    "query.warn" : "10s",
    "query.info" : "5s",
    "query.debug" : "2s",
    "query.trace" : "0s",
    "fetch.warn" : "1s",
    "fetch.info" : "800ms",
    "fetch.debug" : "500ms",
    "fetch.trace" : "0s"
  }
}
```

Define thresholds at different levels (warn, debug, info, trace)

“0s” logs all queries

The Indexing Slow Log

- The ***Indexing Slow Log*** is a similar but separate file to the search slow log
 - logs ***indexing events*** that take longer than configured thresholds
 - already configured in **log4j2.properties**

```
PUT my_index/_settings
{
  "index.indexing.slowlog" : {
    "threshold.index" : {
      "warn" : "10s",
      "info" : "5s",
      "debug" : "2s",
      "trace" : "0s"
    },
    "level" : "trace",
    "source" : 1000
  }
}
```

The first 1,000 characters of the document's source will be logged

Thread Pool Queues

- Many cluster tasks (bulk, index, get, search, etc.) use thread pools to improve performance
 - these thread pools are fronted by queues
 - when a queue is full, 429 status code is returned

GET `_nodes/thread_pool`

```
...  
"bulk": {  
  "type": "fixed",  
  "min": 8,  
  "max": 8,  
  "queue_size": 50  
}  
...
```

GET `_nodes/stats/thread_pool`

```
...  
"bulk": {  
  "threads": 8,  
  "queue": 0,  
  "active": 0,  
  "rejected": 0,  
  "largest": 8,  
  "completed": 177  
}  
...
```

Puneet Kaur Bodgutta 17-Sep-2017 - Optum

Thread Pool Queues

- The `_cat` API provides a nice view of the thread pools:

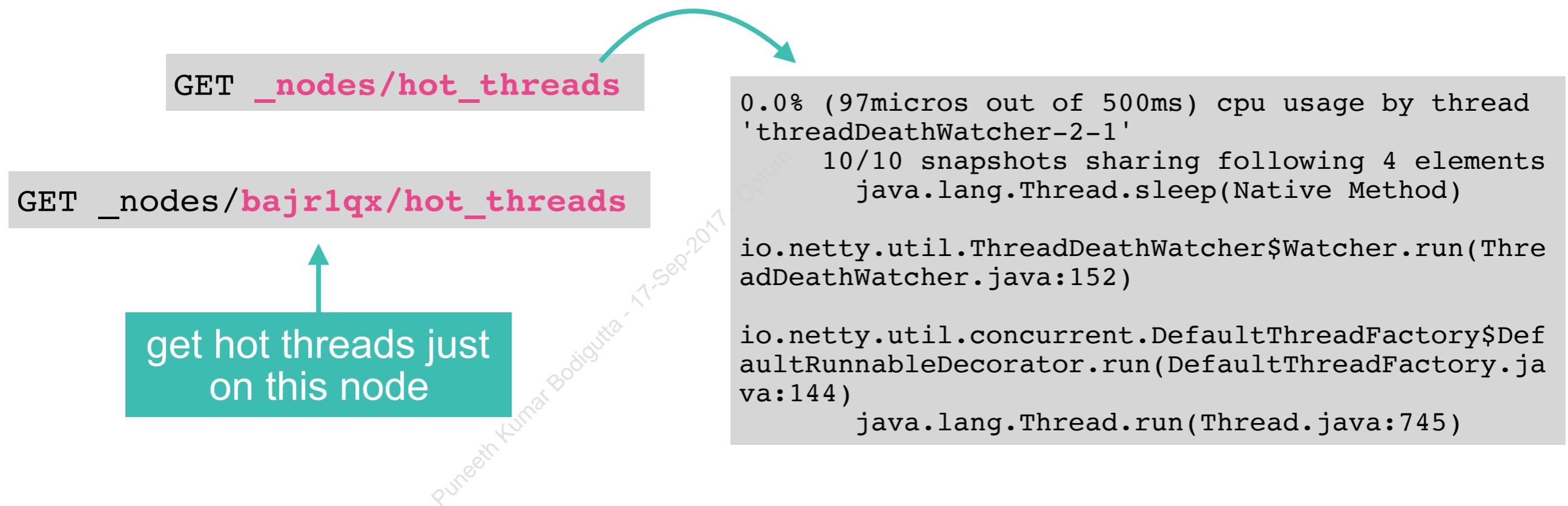
```
GET _cat/thread_pool?v
```

node_name	name	active	queue	rejected
node1	bulk	0	0	0
node1	get	0	0	0
node1	index	0	0	0
node1	management	1	0	0
node2	bulk	0	0	0
node2	get	0	0	0
node2	index	0	0	0
node2	management	1	0	0

- A full queue may be good or bad (“It depends!”)
 - OK if bulk indexing is faster than ES can handle
 - Bad if search queue is full

The hot_threads API

- The *Nodes hot_threads API* allows you get to view the current hot threads on each node
 - invoke it on all nodes or a specific node



Logstash Monitoring

Puneeth Kumar Bodigutta - 17-Sep-2017 , Optum



Logstash Monitoring

- With X-Pack, you can monitor Logstash
 - install the X-Pack plugin for Logstash:

```
bin/logstash-plugin install x-pack
```

- configure your Logstash nodes to send metrics to your production cluster (not your Monitoring cluster):

```
xpack.monitoring.elasticsearch.url: "http://PRODUCTION:9200"  
xpack.monitoring.elasticsearch.username: "logstash_system"  
xpack.monitoring.elasticsearch.password: "changeme"
```

logstash_system user gets created during X-Pack install

Logstash Monitoring

 Elasticsearch

Status

Overview

Version: 5.2.2
Uptime: 22 minutes

Nodes: 2
Disk Available: 136GB / 233GB (58.37%)
JVM Heap: 38.45% (761MB / 2GB)

Indices: 26
Documents: 14,232
Disk Usage: 18MB
Primary Shards: 67
Replica Shards: 63

 Kibana

Status

Overview

Requests: 2
Max. Response Time: 36 ms

Instances: 1
Connections: 2
Memory Usage: 10.02% (143MB / 1GB)

 Logstash

Overview

Events Received: 2
Events Emitted: 2

Nodes: 1
Uptime: a minute
JVM Heap: 13.54% (268MB / 2GB)

Puneeth Kumar Bodigutta - ESConf 2017 - Optum

New section appears now in Monitoring

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- Elasticsearch has **Stats APIs** for retrieving statistics about your cluster, nodes, indices and pending tasks
- The **cat API** provides a human-readable wrapper around the Stats API (and other Elasticsearch APIs)
- The **X-Pack Monitoring** component uses Elasticsearch to monitor Elasticsearch
- Best practice is to use a **dedicated cluster** for Monitoring
- Monitoring has a Kibana plugin that provides useful dashboards and visualizations of the statistics collected by Monitoring
- We discussed several guidelines of areas that may require close monitoring and alerting

Quiz

1. **True or False:** The Monitoring component of X-Pack can monitor multiple clusters.
2. What are the benefits of using a dedicated cluster for the Monitoring component?
3. The default Monitoring collection interval is _____ seconds.
4. How would you check to see if the queue for the index thread pool was full?
5. Suppose your cluster started to have performance issues while processing search queries. What could you do to monitor the problem?

Puneeth Kumar Podigutla - ES 2017 - Optum



Lab 10

Monitoring

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized

Chapter 11

Upgrading a Cluster

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Overview of Upgrades
- Rolling Upgrade
- Full Cluster Restart
- 2.x -> 5.x Considerations

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Overview of Upgrades

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optim

Versions

- Elasticsearch versions are denoted as X.Y.Z
 - X is the *major version*
 - Y is the *minor version*
 - Z is the *patch* level or *maintenance* release
- As we saw in Chapter 1, the latest major change was 2 to 5
 - to get all the Elastic Stack components on the same version number
 - but “2 to 5” only counts as “1” in terms of the distance to the prior version

Puneeth Kumar Bodigutta - 17 Sep 2017 - Optimized

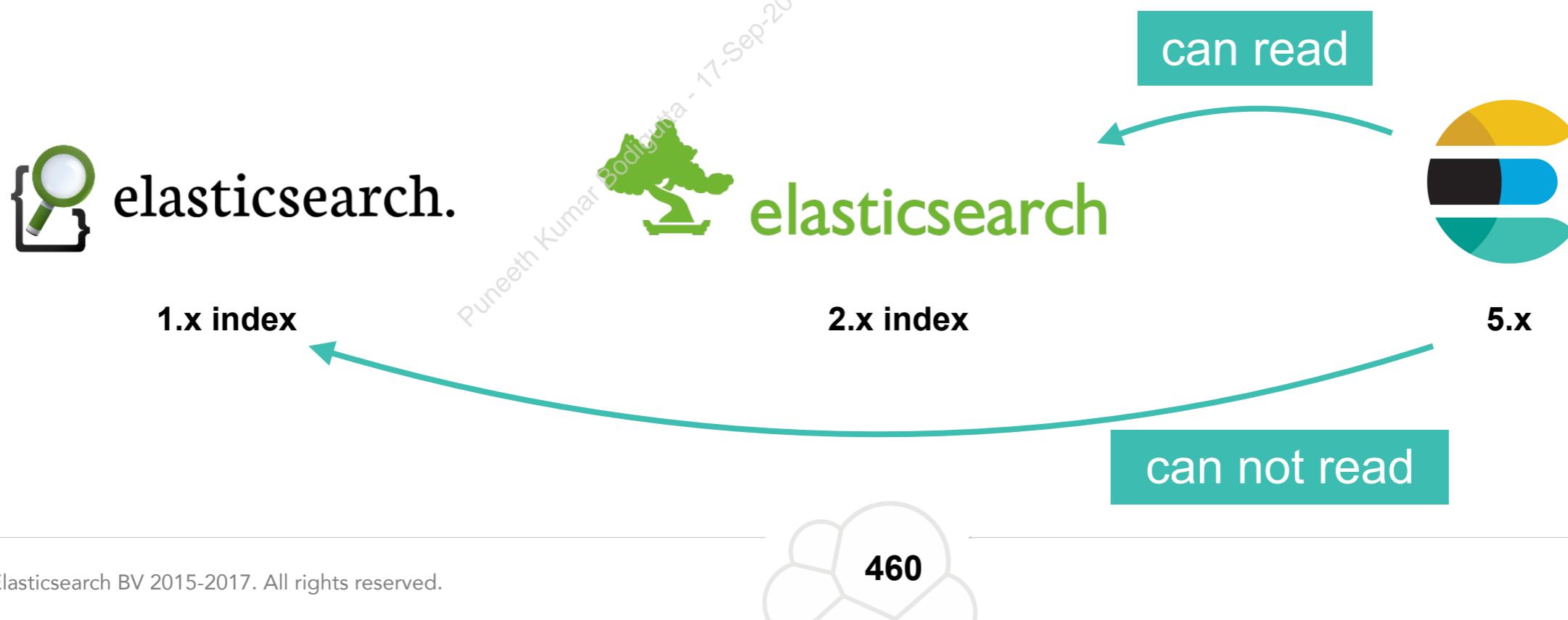
Overview of Upgrades

- Elasticsearch has a fairly rapid release cycle
 - at some point, you will want to upgrade a cluster to a newer version
- In general, there are two possible scenarios that occur when upgrading a cluster:
 - a ***rolling upgrade*** can occur (no downtime)
 - a ***full cluster restart*** is required (some downtime)



Upgrading from 1.x

- Elasticsearch can read indices from the *previous major version only*
 - so an index in 2.x can be used in a 5.x cluster
 - but an index in 1.x can *not* be used in a 5.x cluster
- Elasticsearch 5.x will fail to start if 1.x indices appear
 - therefore, upgrading from 1.x to 5.x *requires all your old indices to be reindexed*



Rolling Upgrade

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Steps for a Rolling Upgrade

- A rolling upgrade allows the nodes in the cluster to be upgraded one at a time
 - by using a *rolling restart*
- To perform a rolling upgrade:
 1. stop non-essential indexing
 2. disable shard allocation
 3. stop and upgrade one node (including plugins)
 4. start the node
 5. re-enable shard allocation and wait
 6. GOTO step 2 (until all nodes are updated)

Step 0: backup your cluster!

Rolling Upgrade

- **Step 1:** stop indexing data (if possible)
 - if you still need to index, that is OK but shard recovery will take longer

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Rolling Upgrade

- **Step 2:** Disable shard allocation
 - and perform a synced flush
 - recall the default is “1m”, but better to just disable it entirely:

```
PUT _cluster/settings
{
  "transient": {
    "cluster.routing.allocation.enable" : "none"
  }
}

POST _flush/synced
```

Rolling Upgrade

- **Step 3:** Install the new version:
 - stop the node you want to upgrade (upgrade master nodes first!)
 - install the new version (.zip, RPM, DEB, MSI)
 - copy the **config** directory from the old location to the new one
 - configure the new node to use the old **data** directory
 - upgrade any plugins if applicable

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimize

Rolling Upgrade

- **Step 4:** Start the node up again
 - and wait for it to join the cluster

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Rolling Upgrade

- Step 5: Reenable shard allocation,

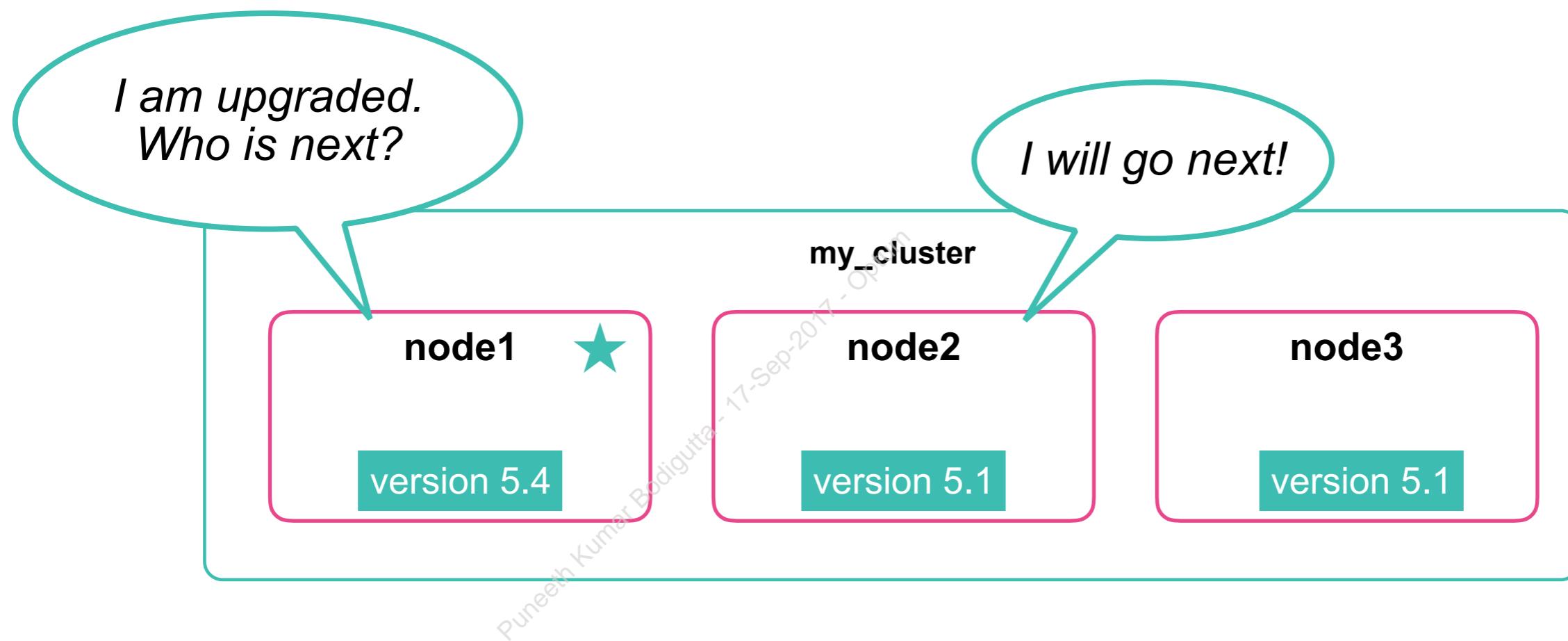
```
PUT _cluster/settings
{
  "transient": {
    "cluster.routing.allocation.enable" : "all"
  }
}
```

- then wait for the cluster to be green again:
 - if green is not possible, check there are no initializing or relocating shards before continuing

```
GET _cat/health
```

Repeat for Each Node

- Step 6 is to start the process over for the next node



Full Cluster Restart

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Full Cluster Restart

- Upgrading to a new major version of Elasticsearch requires a ***full cluster restart***
 - the cluster will be unavailable during the upgrade
- The steps are very similar to a rolling upgrade:
 - stop indexing (e.g. disable writes, or make ES unreachable)
 - disable shard allocation
 - perform a synced flush
 - shutdown and upgrade ***all*** nodes
 - start all dedicated master nodes
 - start the data nodes
 - wait for yellow
 - reenable shard allocation

Puneeth Kumar Kodigutla 17-Sep-2017 - Optum

make sure to use “persistent”

downtime is here

and wait for the election



2.x -> 5.x Considerations

Puneeth Kumar Bodigutta - 17-Sep-2017 Optum

2.x -> 5.x Considerations

- Install and use the *Migration Plugin*
 - <https://github.com/elastic/elasticsearch-migration>
 - You can execute it from a remote cluster as well
 - It performs a Cluster Checkup
 - upgrades old indices (<2.0) at the click of a button
 - enables or disables deprecation logging on your cluster

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimize

2.x -> 5.x Considerations

- Run a series of checks on your cluster, nodes, and indices and alerts you to any known problems that need to be rectified before upgrading.
 - **green**: all OK
 - **blue**: advisory note; something has changed; no action needed
 - **yellow**: can upgrade directly, but you are using deprecated functionality which will not be available in version 5.x
 - **red**: cannot upgrade without fixing this problem

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimal Cluster Configuration

2.x -> 5.x Considerations

- Particular items to look out for include:
 - bootstrap checks are new to 5.x
 - JVM options
 - support for `_site` plugins has been removed
 - default scripting language is now **Painless**
 - **string** data type replaced by **text** and **keyword** data types

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimize

Full Upgrade Best Practices

- Test, Test, Test!
 - perform upgrade for non-prod environment and test with apps
 - in cloud, can create entire new cluster and restore into it to test
- Prepare!
 - snapshot prior
 - disable access to the cluster and then snapshot one more time
 - remove migration plugin
 - perform full cluster shutdown

Puneet Kumar Boddu - 17-May-2017 - Optum

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- There are typically two possible scenarios that occur when upgrading a cluster:
 - a *rolling upgrade* can occur (no downtime)
 - a *full cluster restart* is required (some downtime)
- A *rolling upgrade* allows the nodes in the cluster to be upgraded one at a time
- Upgrading to a new major version of Elasticsearch requires a *full cluster restart*
- Elasticsearch can read indices from the *previous major version only*
- At some point, you are going to want to upgrade your Elasticsearch clusters!

Puneeth Kumar Bhatta - 17-Sep-2017 Optum



Quiz

1. **Rolling or Restart:** an upgrade from 2.0 to 2.3
2. **Rolling or Restart:** an upgrade from 2.2 to 5.2
3. **Rolling or Restart:** an upgrade from 1.x to 5.x
4. **True or False:** You can index documents during a rolling upgrade.
5. **True or False:** All of the data in your cluster is available for searches during a rolling upgrade.
6. What is the benefit of performing a synced flush right before a node restart?

Puneeth Kumar Bodiguttula
27 Sep 2017 Optum



Lab 11

Perform a Rolling Restart

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Chapter 12

Production Checklist

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

- 1 Introduction to Elasticsearch
- 2 Installation and Configuration
- 3 Working with Nodes
- 4 Indexes
- 5 Cluster Health
- 6 Mappings and Analysis
- 7 Index Management
- 8 Capacity Planning
- 9 Cluster Management
- 10 Monitoring
- 11 Upgrading a Cluster
- 12 Production Checklist

Topics covered:

- Security Considerations
- Bootstrap Checks
- Best Practices

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Security Considerations

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

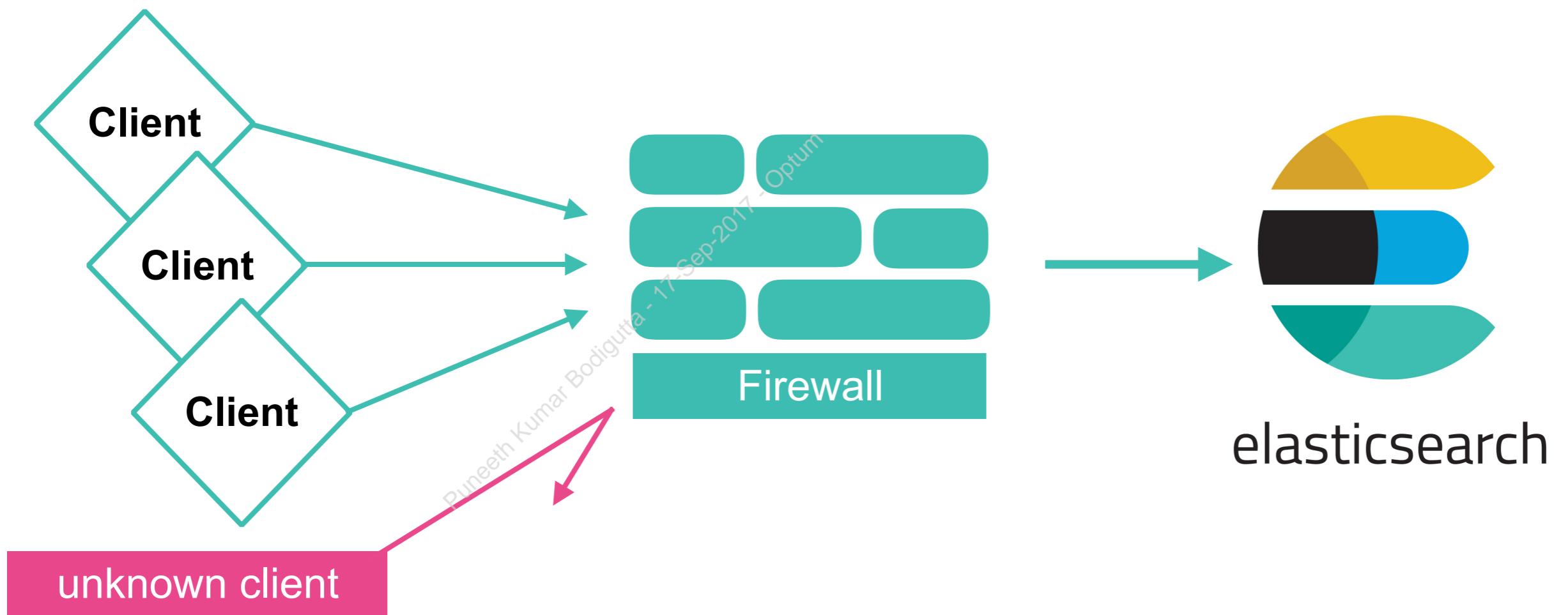
Security Considerations

- Out-of-the-box, Elasticsearch does not have any authentication, authorization or encryption services
- **X-Pack Security** provides a complete solution for securing Elasticsearch
 - but depending on your security requirements, there are a few other options to consider...

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

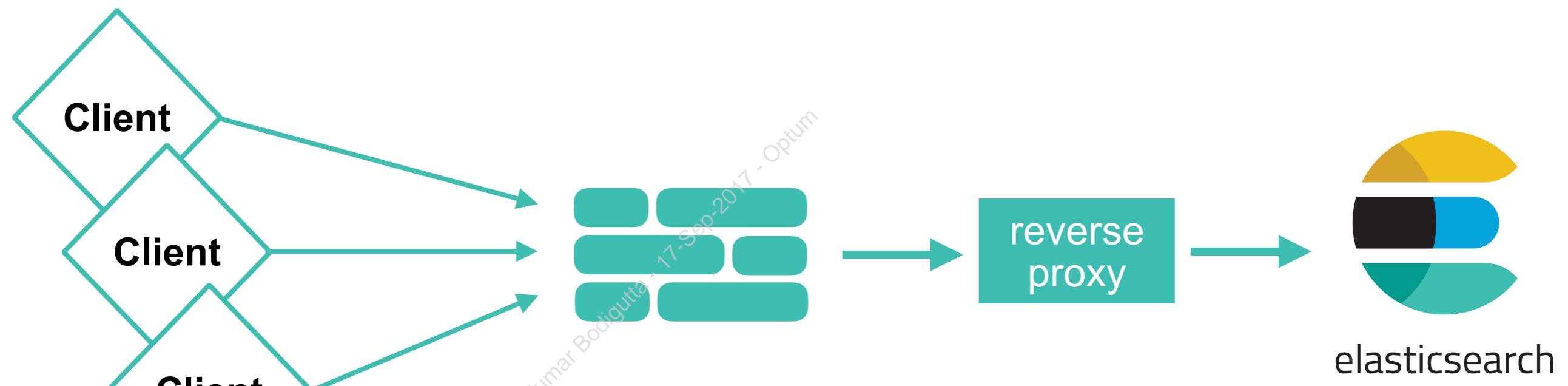
Firewall

- Add **firewall rules** with IP filtering to only allow access from the machines running your application



Reverse Proxy

- You can put Elasticsearch behind a *reverse proxy*
 - gain basic authentication and authorization
 - helpful blog at <https://www.elastic.co/blog/playing-http-tricks-nginx>



Secure REST with SSL

- Use an application like HAProxy, NGINX, Apache httpd, etc. to provide SSL for REST API
 - For example, you could run HAProxy locally on each node
 - deploy a backend pool with one entry of 'localhost:9200'
 - bind HTTP on each Elasticsearch node to localhost only

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Read-only Configurations

- Using a layer 7 web proxy (HAProxy, NGINX, etc.), you can configure:

1. read-only cluster vs. write cluster

- add rules that parse the URL path restricting requests to certain URL paths

2. read-only indices vs. write indices:

- only allow GET requests for certain logins/IPs

3. read-only Kibana dashboards and visualizations

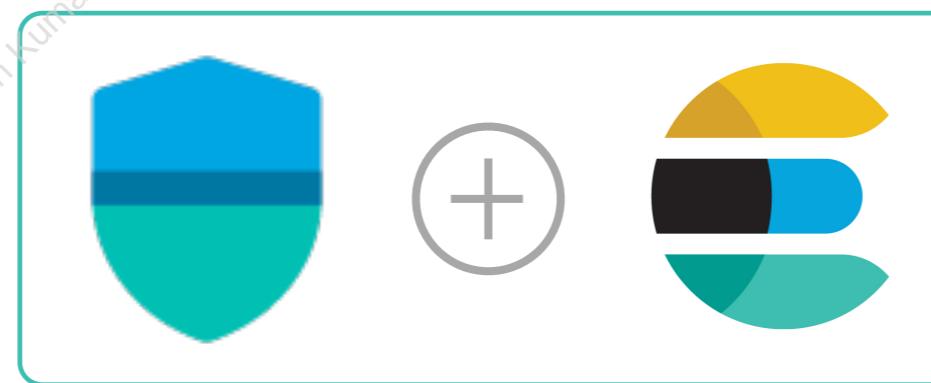
- only allow GET, HEAD, and OPTIONS requests for the particular Kibana instances that are desired to be read-only

Puneeth Kumar Budigutta - Map-2017.pdf

X-Pack Security

- The **Security** component of **X-Pack** provides all of the security solutions discussed on the previous slides
- As well as:
 - role-based security
 - document-level and field-level security
 - encryption of data between nodes
 - username/password access to Kibana
 - role-based security with Kibana
 - audit logging

Puneeth Kumar Bodiguttla / 17 Sep 2017 / Optum



Bootstrap Checks

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Remember the Bootstrap Checks?

- A node in *production mode* must pass all of the checks, or the node will not start
 - the bootstrap checks fit into two categories:

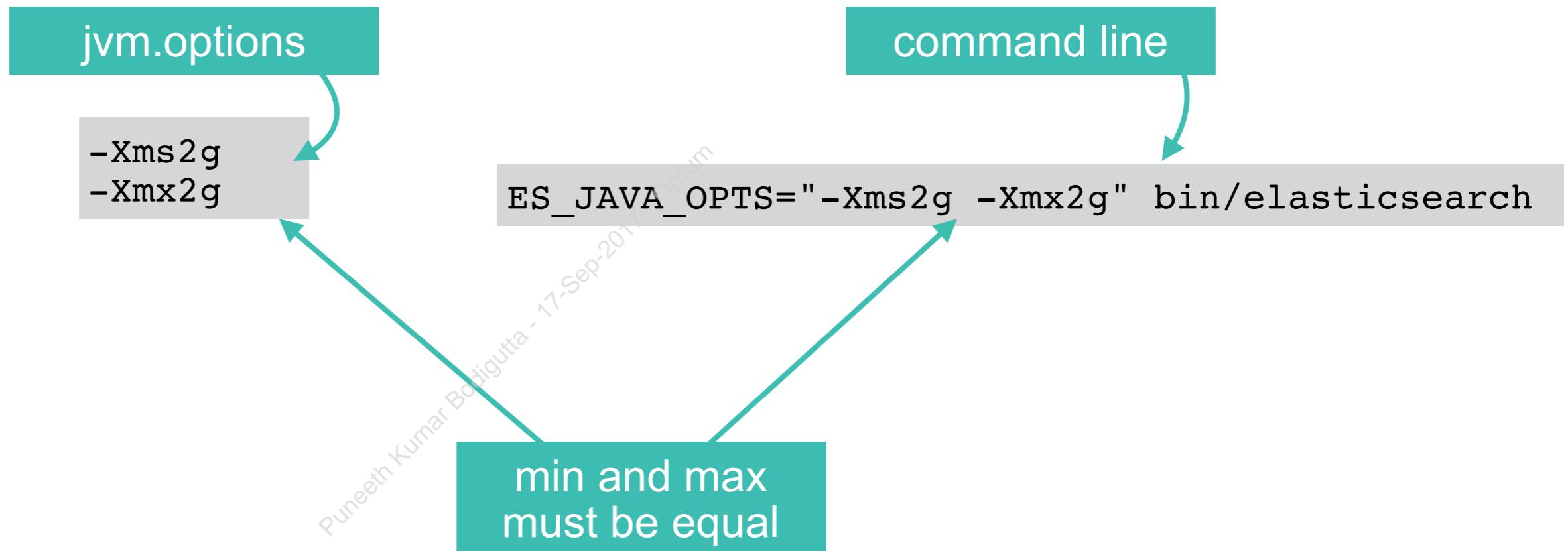
JVM Checks	Linux Checks
<input checked="" type="checkbox"/> heap size	<input checked="" type="checkbox"/> maximum map count
<input checked="" type="checkbox"/> disable swapping	<input checked="" type="checkbox"/> maximum size virtual memory
<input checked="" type="checkbox"/> not use serial collector	<input checked="" type="checkbox"/> maximum number of threads
<input checked="" type="checkbox"/> OnError and OnOutOfMemoryError	<input checked="" type="checkbox"/> file descriptor
<input checked="" type="checkbox"/> G1GC	<input checked="" type="checkbox"/> system call filter
<input checked="" type="checkbox"/> client JVM	

Puneeth Kumar Bodigutta - 17-May-2017 - Optum

JVM Checks

Heap size

- set min (**-Xms**) and max (**-Xmx**) heap sizes equal
- in **jvm.options** or using **ES_JAVA_OPTS** on the command line



JVM Checks

Not use serial collector

- ensures that Elasticsearch is *not* configured to run with the serial garbage collector

Disable swapping

- configure the JVM to lock the heap in memory using **mlockall** (Linux) or **virtual lock** (Windows)
- If **bootstrap.memory_lock** is **true**, ES will verify locking

GET `_nodes/process`

You can verify that the
memory lock was
successful

```
"cluster_name": "my_cluster",
"nodes": {
    "OmWJPhToQ0iyNfz-Qd9i8g": {
        "name": "node1",
        ...
        "process": {
            "refresh_interval_in_millis": 1000,
            "id": 3079,
            "mlockall": true
        }
    },
}
```

JVM Checks

OnError and OnOutOfMemoryError

- To pass this check, do not enable **OnError** nor **OnOutOfMemoryError** in our JVM options
- instead, upgrade to **Java 8u92+** and use the JVM flag **ExitOnOutOfMemoryError**

G1GC

- the G1GC garbage collector can not be enabled on versions of the JDK prior to JDK8u40 (and is not recommended anyway)

Client JVM

- OpenJDK has a client JVM and a server JVM
- you must start Elasticsearch with the server VM

Linux Checks



Maximum map count

- you must configure `vm.max_map_count` via `sysctl` to be at least 262,144



Maximum size virtual memory

- configure your system to allow the Elasticsearch process the ability to have unlimited address space
- set “as” to “unlimited” in `/etc/security/limits.conf`



Maximum number of threads

- configure your system to allow the Elasticsearch process the ability to create at least 2,048 threads
- set the `proc` setting in `/etc/security/limits.conf`



Linux Checks

File descriptor

- increase the limit on the number of open files descriptors for the user running Elasticsearch to 65,536 or higher
- set **ulimit -n 65536**, or set **nofile** to **65536** in **/etc/security/limits.conf**
- (RPM/Deb installs already have done this)

```
GET _cat/nodes?v&h=name,file_desc.max
```

name	file_desc.max
node1	10240
node2	10240

Linux Checks

System call filter

- either fix any configuration errors on your system that prevented system call filters from installing
- or, at your own risk disable system call filters by setting **bootstrap.system_call_filter** to **false**

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Best Practices

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Networking Best Practices

- Avoid running over WAN links between datacenters
 - not officially supported by Elastic
- Try to have zero (or very few) hops between nodes
- Separate **transport** and **http** traffic
 - bind to different network interfaces
 - use separate firewall rules for each kind of traffic
- Use long-lived HTTP connections
 - client libraries support this
 - or use a proxy/load-balancer

Puneet Kumar Bodapati - 17-Sep-2017 - Optum

Storage Best Practices

- Prefer solid state disks (SSDs)
 - segments are immutable, so the *write amplification factor* approaches one and is a non-issue
- Prefer local storage
 - in other words, avoid NFS or SMB
- You can set multiple **path.data** directories
 - allows you to stripe your index across multiple SSDs

```
path.data: /path1,/path2
```

Note that files belonging to a single shard will all be stored on the same data path

Storage Best Practices

- Use **noop** or deadline scheduler in the OS when using SSD
 - For details, see:
https://www.elastic.co/guide/en/elasticsearch/guide/current/hardware.html#_disks
- Spinning disks are OK for **warm** nodes
 - disable concurrent merges
`index.merge.scheduler.max_thread_count: 1`
- Trim your SSD's:
 - <https://www.elastic.co/blog/is-your-elasticsearch-trimmed>

Storage Best Practices

- Local disk is king!
 - avoid **NFS, AWS EFS, Azure filesystem**
- Elasticsearch does not need reliable storage
 - replicas/software provide HA
 - local disks are better than **SAN**
 - **RAID1/5/10** is not necessary
- **path.data** vs. **RAID0**
 - **RAID0** will be slightly more performant
 - **path.data** will allow a node to continue to function
 - e.g. a machine with 4 2TB drives and at most only 25% (2TB) of data will need to relocate

Hardware Selection

- In general, choose ***medium machines over large machines***
 - loss of a large node has a greater impact
 - ***prefer*** six 4cpu x 64gb x 4 1tb drives
 - ***avoid*** 2 12cpu x 256gb x 12 1tb drives
- Avoid running multiple nodes on one server
 - one Elasticsearch instance can fully consume a machine
- Larger machines can be helpful as ***cold nodes***
 - configure shard allocation filtering as previously discussed

Puneeth Kumar Bodiguttula
17-Sep-2017 Optum

Throttles

- Elasticsearch has relocation and recovery throttles to ensure these tasks do not have a negative impact
- **Recovery:**

- for faster recovery, temporarily increase the number of concurrent recoveries:

```
PUT _cluster/settings
{
  "transient": {
    "cluster.routing.allocation.node_concurrent_recoveries": 2
  }
}
```

- **Relocation:**
- for faster rebalancing of shards, increase:

```
"cluster.routing.allocation.cluster_concurrent_rebalance" : 2
```

Chapter Review

Puneeth Kumar Bodigutta - 17-Sep-2017 . Optum

Summary

- A node in ***production mode*** must pass all of the bootstrap checks, or the node will not start
- The JVM bootstrap checks apply to all OS's. The other bootstrap checks only apply to Linux.
- For best performance, choose ***SSD over spinning disks***
- Local disks are preferred - the software provides HA
- In general, choose ***medium machines over large machines***
- Temporarily decrease the recovery and reallocation throttles to increase the speed of these tasks if necessary

Puneeth Kamaraju - 17-Sep-2017 Optum

Quiz

- True or False:** If your hardware has 128GB of RAM, you should set the min and max heap size of the JVM to 64GB.
- What is the benefit of setting the **min** and **max** heap size to the same values?
- True or False:** SAN storage is preferred over local disks to provide high availability of data.
- What is the purpose of setting **bootstrap.memory_lock** to **true**?
- True or False:** It is a good idea to separate the **transport** and **HTTP** traffic over different network interfaces.

Puneeth Kumar
digvijay17-Sep-2017 - CTF

Conclusions

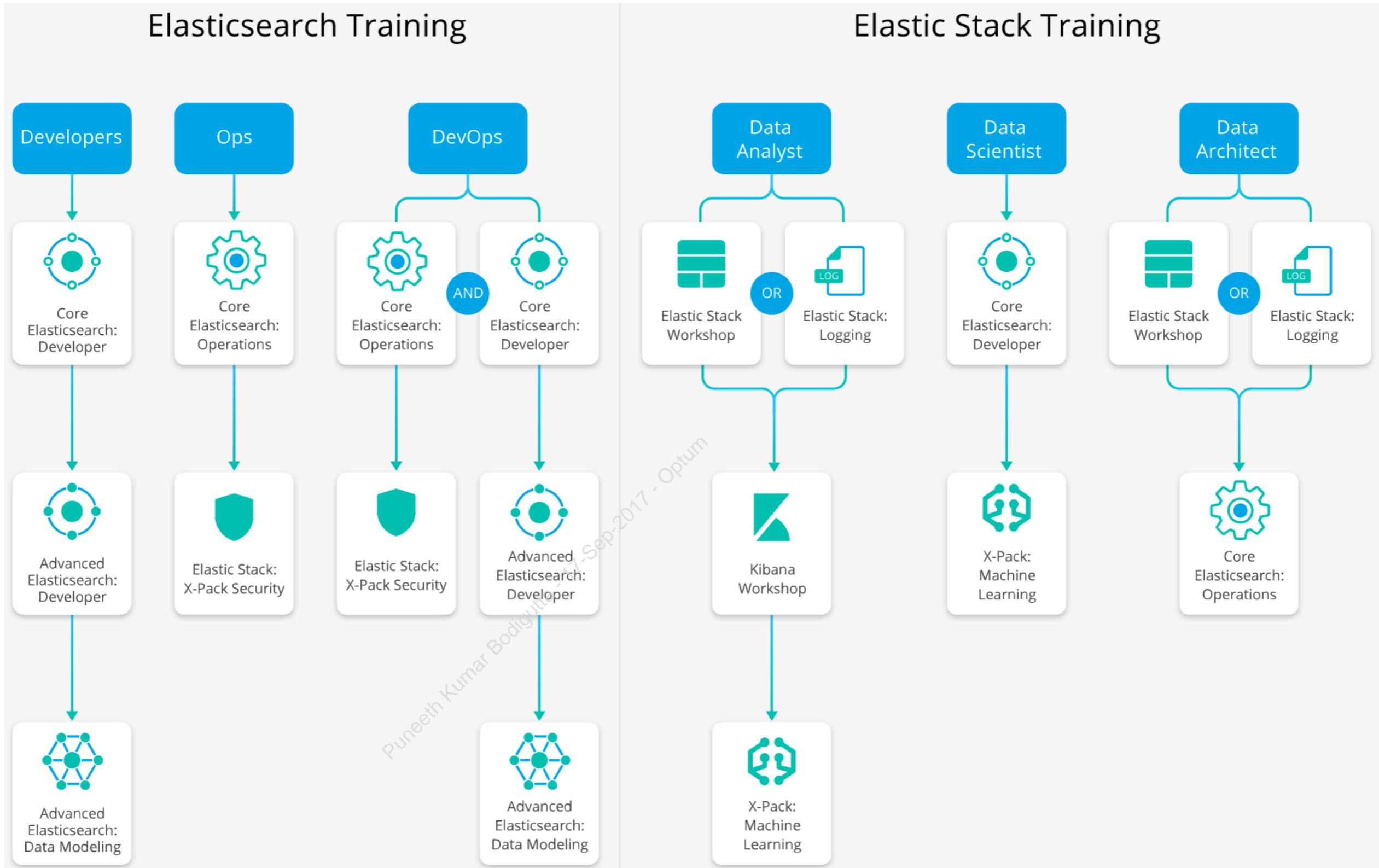
Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Documentation and Help

- Discussion forums: <http://discuss.elastic.co/>
- Meetups: <http://elasticsearch.meetup.com>
- Docs: <https://elastic.co/docs>
- Community: <https://elastic.co/community>
- More resources: <https://elastic.co/learn>

Puneeth Kumar Bodigutta - Elasticsearch 2017 - Optum

Elastic Training Paths



Quiz Answers

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Chapter 1 Quiz Answers

1. True
2. As a **noun**, an index is what ES creates for your documents. As a **verb**, we “index” documents into an index
3. True
4. Elasticsearch will define a new index for you
5. No. If you want ES to generate an ID, you have to use POST
6. Five is the default number of shards

Puneeth Kumar Bodiguttla - 11-Sep-2017 - Optimize



Chapter 2 Quiz Answers

1. path.data
2. False. HTTP is for clients, and is optional for any node
3. Both HTTP and transport will bind to any globally-scoped IP addresses by default (unless overridden), which results in a production mode node
4. True
5. True

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Chapter 3 Quiz Answers

1. False - they use unicast by default
2. master eligible, data and ingest
3. False - the master node has to create the index, but the client can send the request to any node
4. You could disable it only by disabling HTTP on that node
5. Set node.master and node.ingest = false, node.data = true
6. Disable master, data and ingest
7. Three master eligible nodes, with minimum_master_nodes set to 2
8. False. Assuming you have multiple master eligible nodes, then an election occurs.



Chapter 4 Quiz Answers

1. namespace, shards
2. 1
3. False! It actually is more expensive in terms of performance, but you gain about 25% more compression than LZ4
4. This would delete every index and document in your cluster
5. True: higher orders override the settings in lower order templates
6. True

Puneeth Kumar Bodigutta - 17-Sep-2017 Optum

Chapter 5 Quiz Answers

1. Can't get any better than yellow
2. False. Yellow has a potential for lost data because not all replicas exist
3. INITIALIZING
4. Must be red
5. The health status is either yellow or green, or 10 seconds passes

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optimized

Chapter 6 Quiz Answers

1. 582 = long, “I love this restaurant” is both “text” and “keyword”. the date maps to “date”
2. analysis, or text analysis
3. Exact values are not analyzed vs. full text is analyzed
4. True
5. Coordinating
6. a large number of uncommitted segments hence causing a very large **trans_log** that will need to be replayed at startup; to avoid this we could **_flush** indic(es) prior to the node restart.

Puneeth Kumar Bodumalla 21 Sept 2017 - Optum

Chapter 7 Quiz Answers

1. False
2. Existing documents in the target index are overwritten if they have an older version number
3. False: a closed index can not be queried or accessed in any way
4. The current day's index could be a hot node handling indexing and queries, while all previous indices could be on warm nodes (assuming they do not get queried as often)
5. The index will be defined, but all of its shards will be unallocated and the cluster will go into a red status

Puneeth Kumar Bodigunta
17-Sep-2017



Chapter 8 Quiz Answers

1. True
2. In terms of searching, it would essentially be equivalent
3. Overallocate! It allows for future scaling of the cluster
4. False. There are many scenarios where 1 shard might actually be optimal, especially if the data all fits in a single shard
5. 9 or 10 depending upon if you want some extra buffer room. And that depends upon other requirements: it may be a single index or multiple indices that total the 9 or 10 shards

Puneeth Kumar Bodigunda
17 Sept 2017 Optum

Chapter 9 Quiz Answers

1. Filtering is not “aware” of the physical configuration of your hardware. Awareness ensures that shards are distributed across “zones” that you define
2. Forced awareness never allows copies of the same shard to be in the same zone - shard allocation awareness does
3. False! Replicas provide no backup mechanism of any kind
4. You need to register the repository
5. True
6. False (or True): it could be smaller, but it also could be bigger depending on how the segments changed between snapshots

Puneeth Kumar Bhatnagar - 17-Sep-2017 - Onlinelabs



Chapter 10 Quiz Answers

1. True, but only the commercial version does. The free version can only monitor one
2. If a cluster fails, you will be able to view its history and perhaps diagnose the issue. There are also performance and security benefits.
3. 10 seconds
4. Look at the thread pool queues either with `_nodes/thread_pool` or `_cat/thread_pool`
5. Enable and configure the slow log, and also monitor the JVM heap usage and garbage collection

Puneeth Kumar Reddy @ puneethkr.com
17-Sep-2017

Chapter 11 Quiz Answers

1. Rolling
2. Restart
3. Restart and Reindex!
4. True. You can, but it slows down the recovery time
5. It depends! Typically yes - but you may have an index with no replicas, or you may have shard filtering configured so that only 1 particular node can hold a shard, which means that shard would not be available during the restart
6. It greatly speeds up the recovery time of indices that have not changed while the node (or nodes) was down

Puneeth Ramamurthy - 17-Nov-2017 Optimized Indexing



Chapter 12 Quiz Answers

1. False. Do not set the heap to bigger than about 30gb
2. Avoid the pauses that can occur if the heap resizes
3. False: SAN is not needed
4. The memory lock check verifies that if the **bootstrap.memory_lock** setting is enabled, that the JVM was successfully able to lock the heap.
5. True

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Thank you!

Please complete the online survey

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum

Course: Core Elasticsearch Operations

Version 5.4.1

© 2015-2017 Elasticsearch BV. All rights reserved. Decompiling, copying, publishing and/or distribution without written consent of Elasticsearch BV is strictly prohibited.

Puneeth Kumar Bodigutta - 17-Sep-2017 - Optum