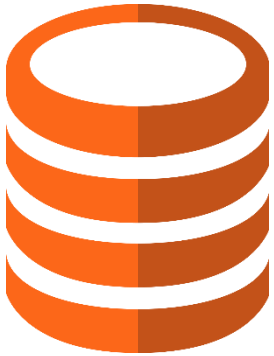
Clustering & Configuration Strategies

JP Toto

@jptoto | http://jptoto.jp

Traditional SQL Databases



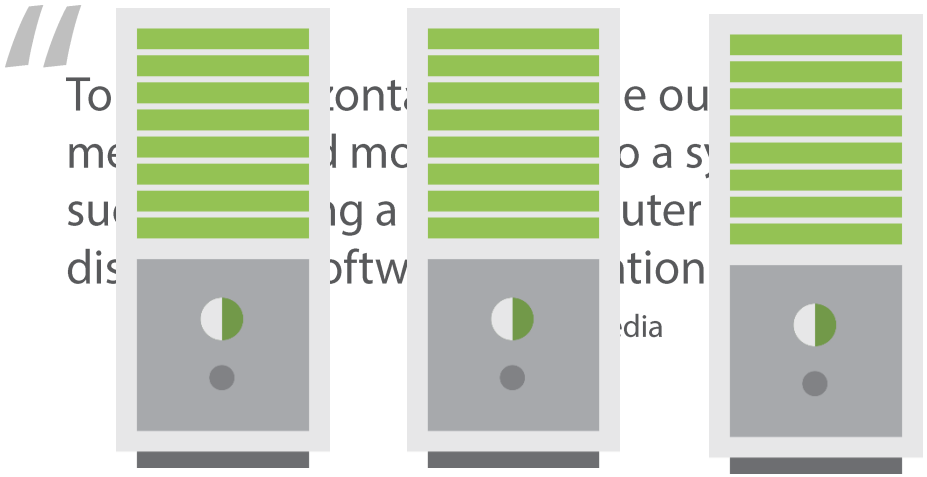
SQL Database

SERVER 01
(primary)

SERVER 02
(replica)

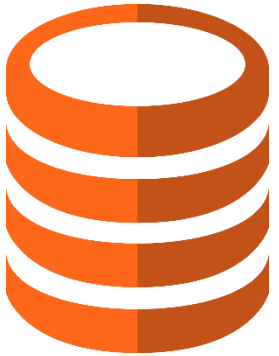
Elasticsearch Is a Distributed Database

Elasticsearch Index



Elasticsearch Concepts: Indexes

≈

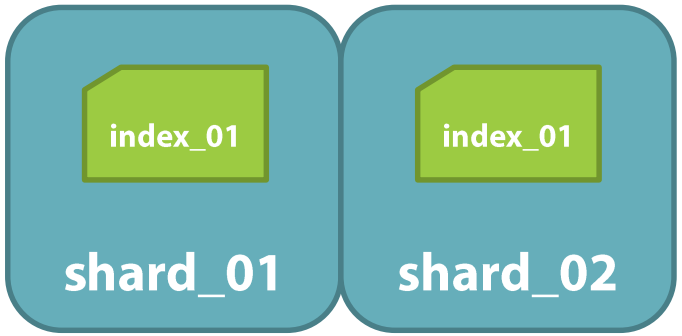


SQL Database

Logical separation of data

Inserting documents = “indexing”

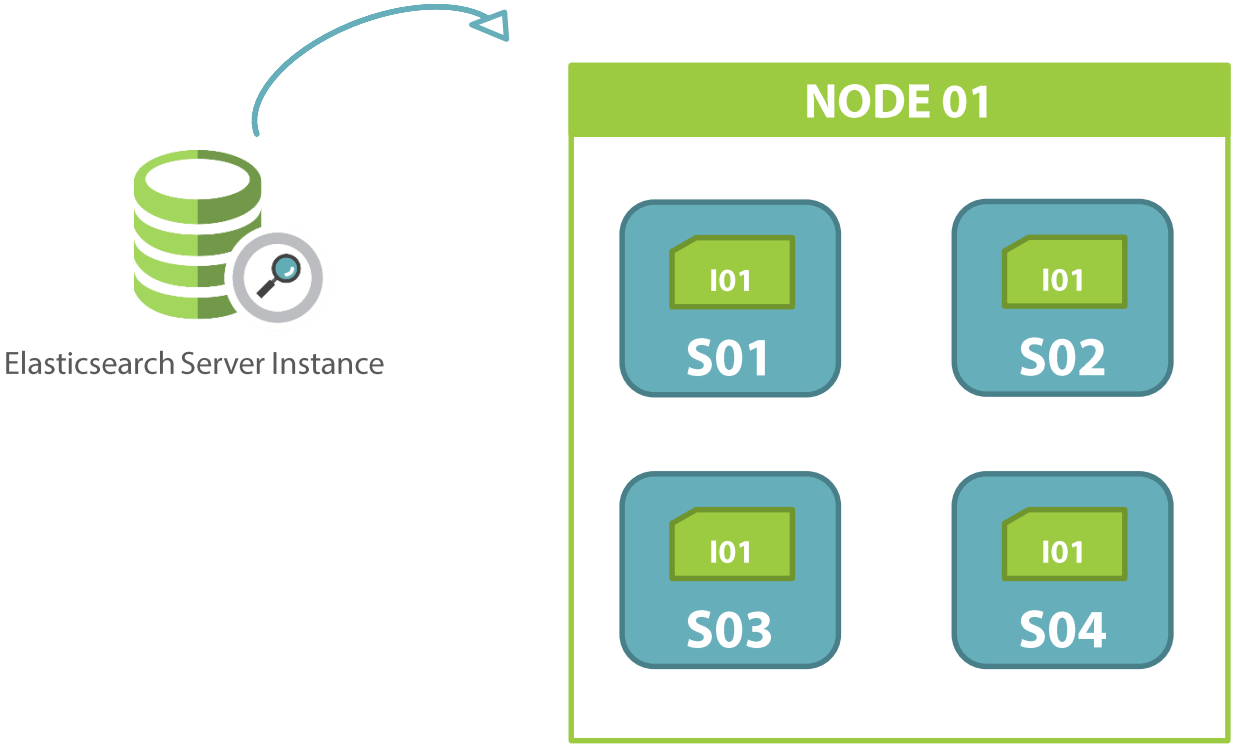
Elasticsearch Concepts: Shards



Indexes are stored in shards

Indexes can live in multiple shards

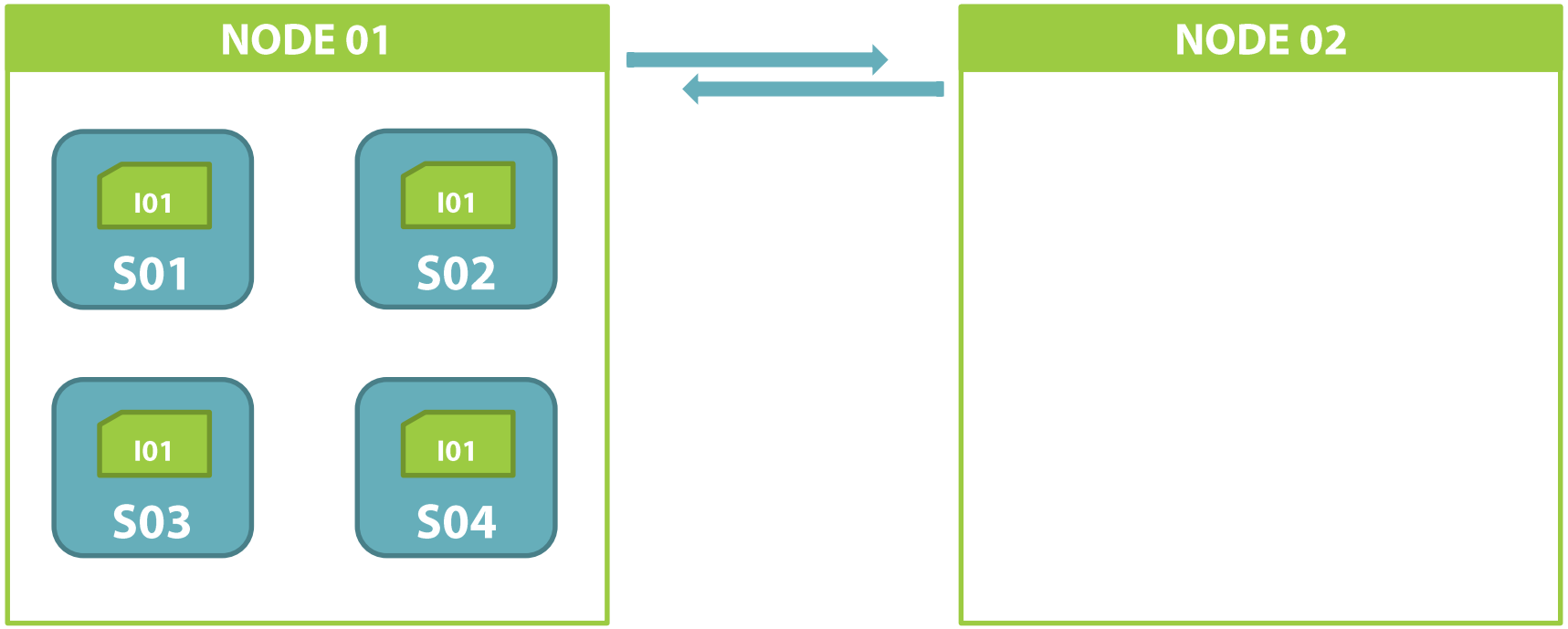
Elasticsearch Concepts: Shards



Index stored across all four shards

Elasticsearch Concepts: Shards

Free!

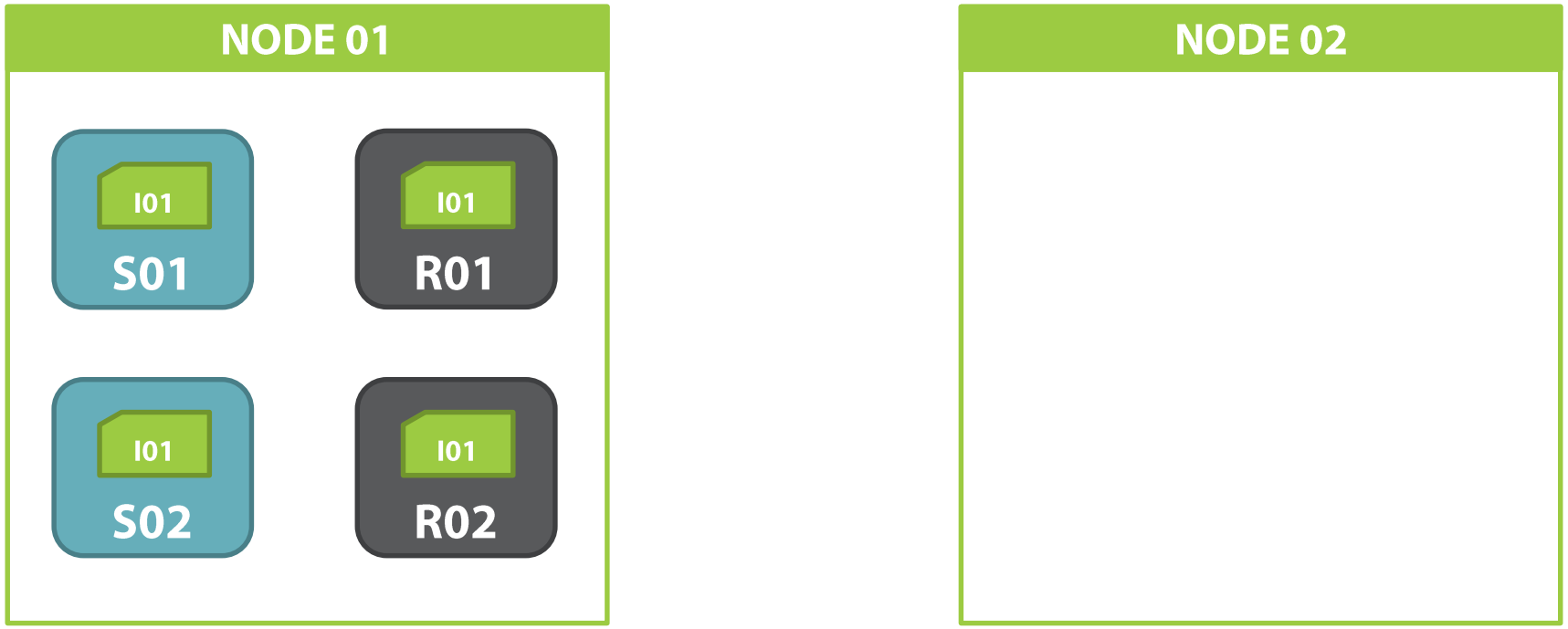


1) Node 02 joins cluster as peer

2) Nodes "gossip" to exchange information

3) Shards "rebalance" and migrate to even the load

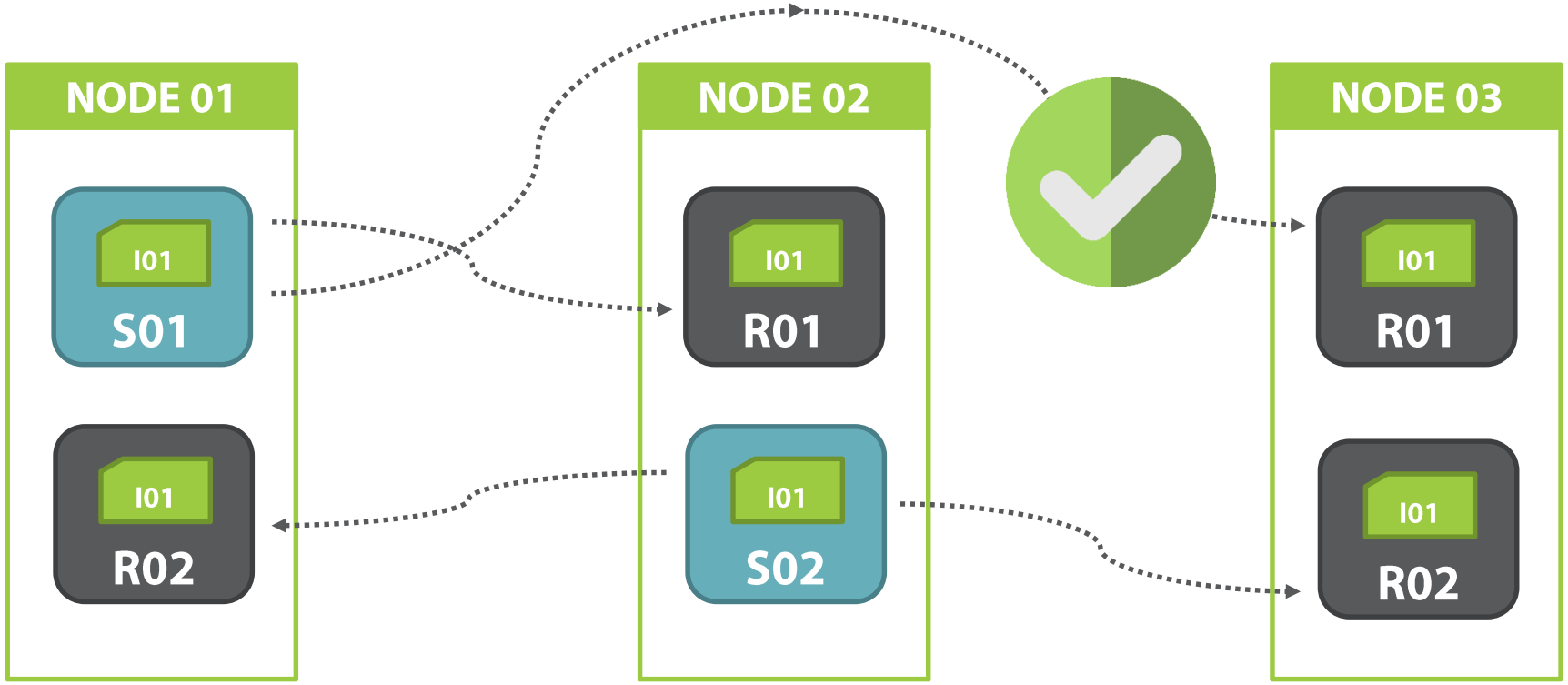
Elasticsearch Concepts: Replicas



Replicas are duplicates of a shard

They also distribute themselves among nodes

Elasticsearch Concepts: Replicas

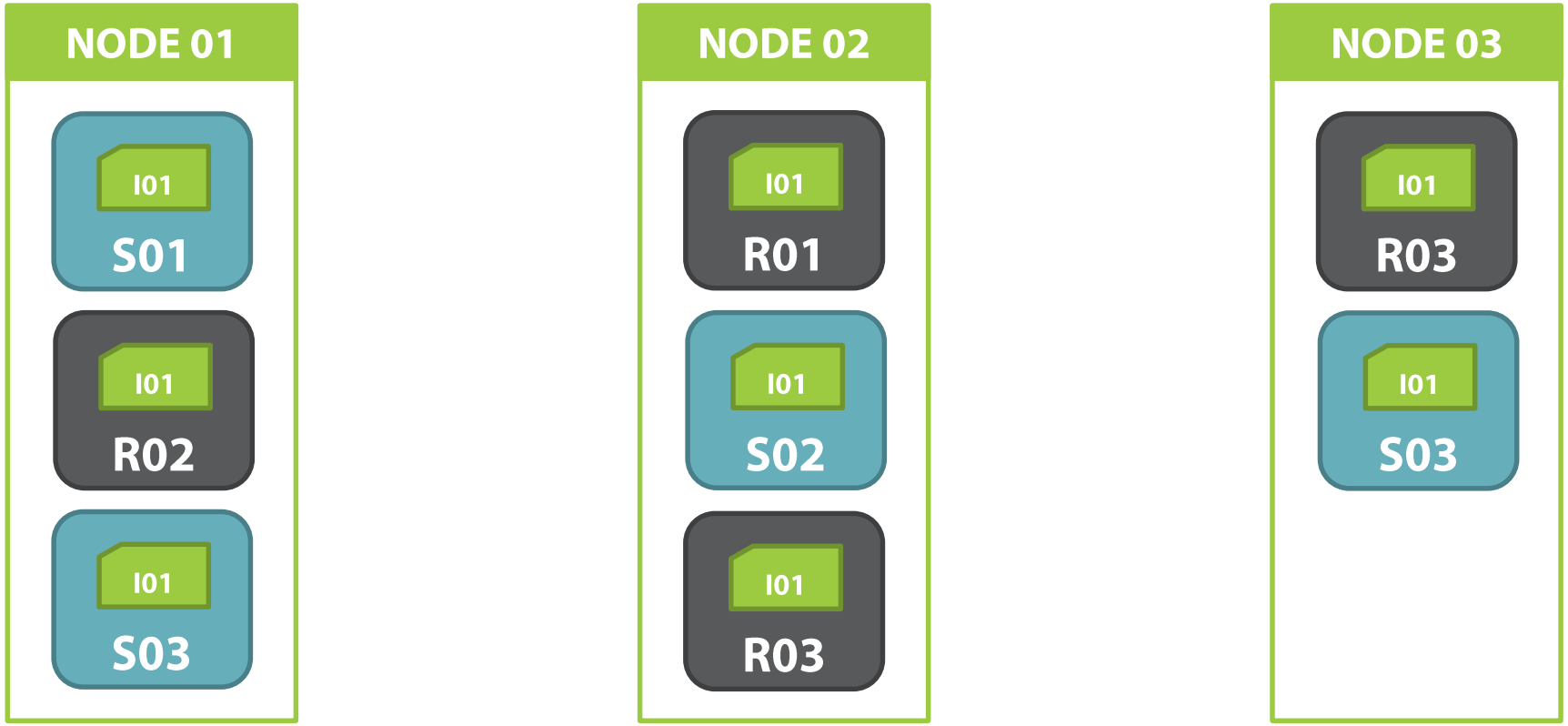


Two shards plus two replicas per shard

Replicas can also serve data

At least one copy of each shard or replica remains. We're still serving 100% of the data

Elasticsearch Concepts: Rebalancing



Elasticsearch automatically detects node failures

Attempts to replace the missing shards and replicas

Elasticsearch Concepts: Node Roles

Planning makes for a fast, efficient cluster!

Elasticsearch Concepts: Node Roles

Data nodes, master nodes, and client nodes

This works ok, but it's not terribly efficient

Setting up nodes for dedicated functions offers more performance and stability

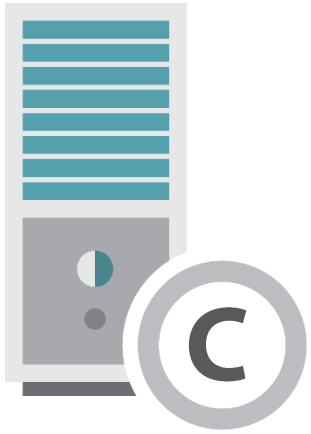
Elasticsearch Nodes: Data Node

Hard working since they physically house all the shards and data

Don't typically service query requests

Tend to have the "beefiest" resources

Elasticsearch Nodes: Data Node

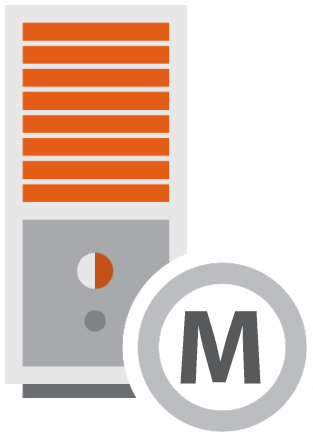


Gateway to the cluster

Handles all query requests and directs them to data nodes

They're important but don't need a lot of muscle

Elasticsearch Nodes: Data Node

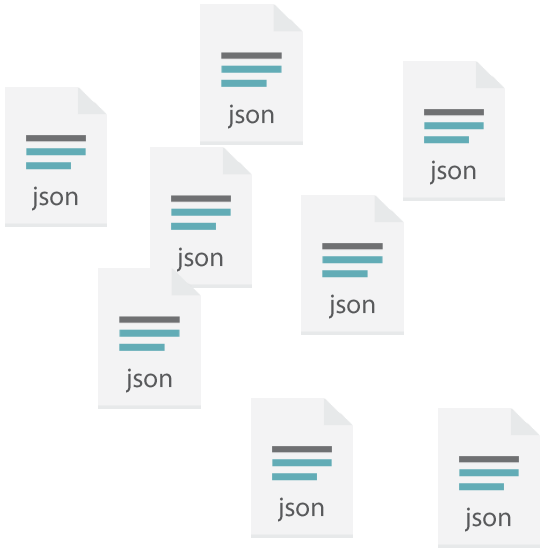


- Brains of the whole operation
- In charge of maintaining the "cluster state"
- It's the only node allowed to update the state
- Without a master the cluster won't function

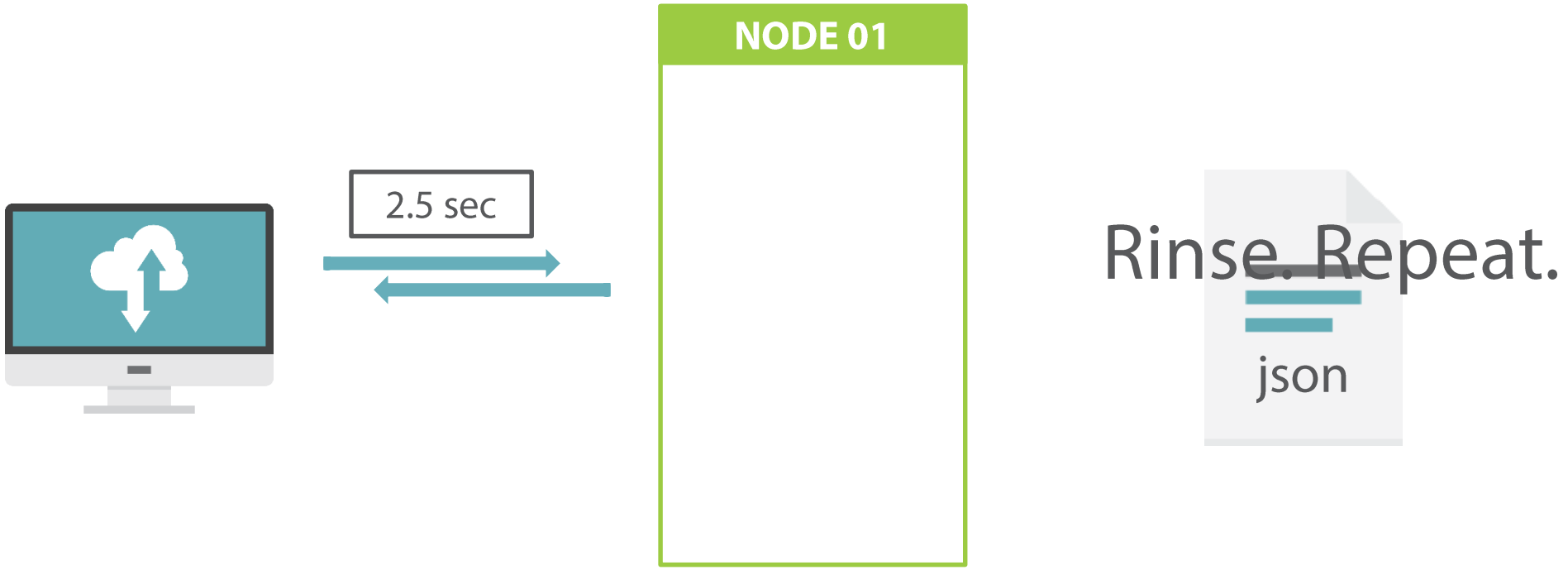
Capacity Planning

Tricky business

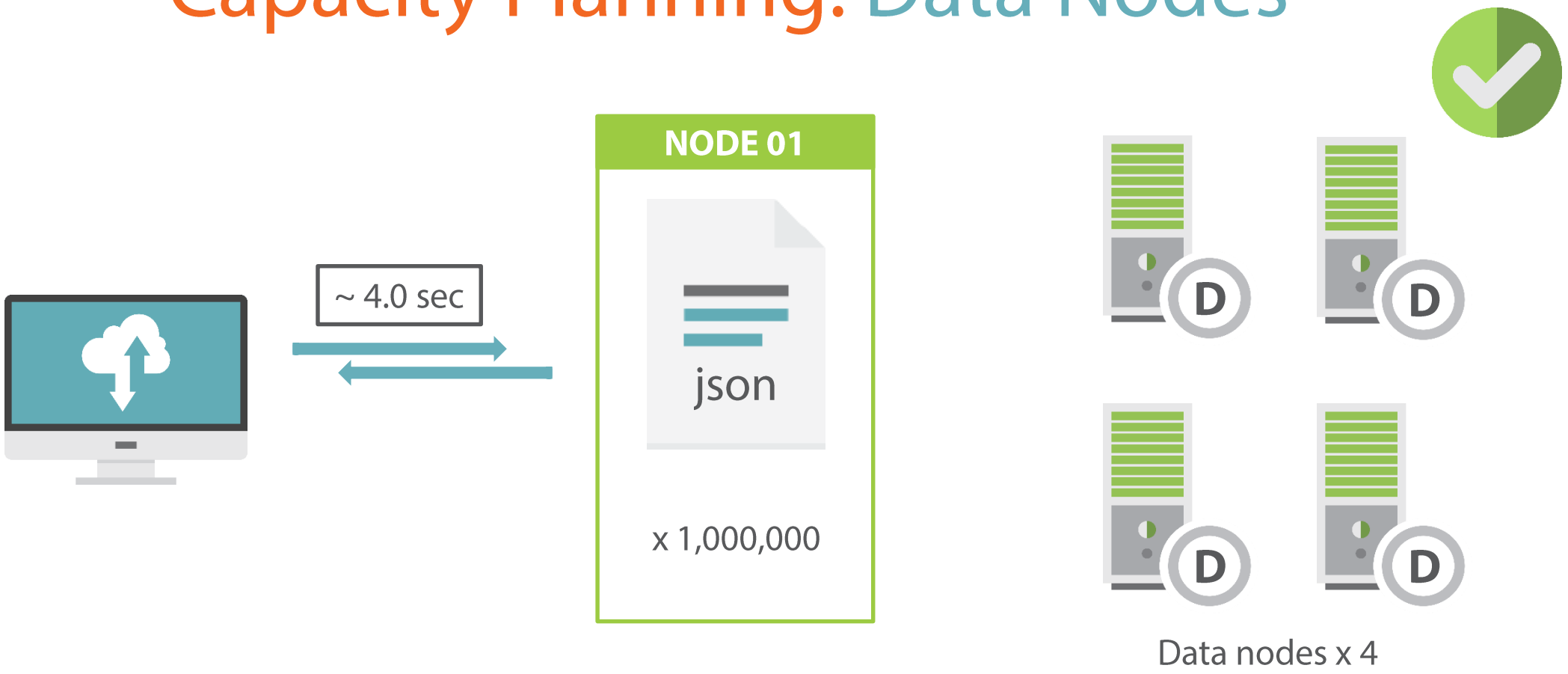
Size and number of documents plays a role in performance



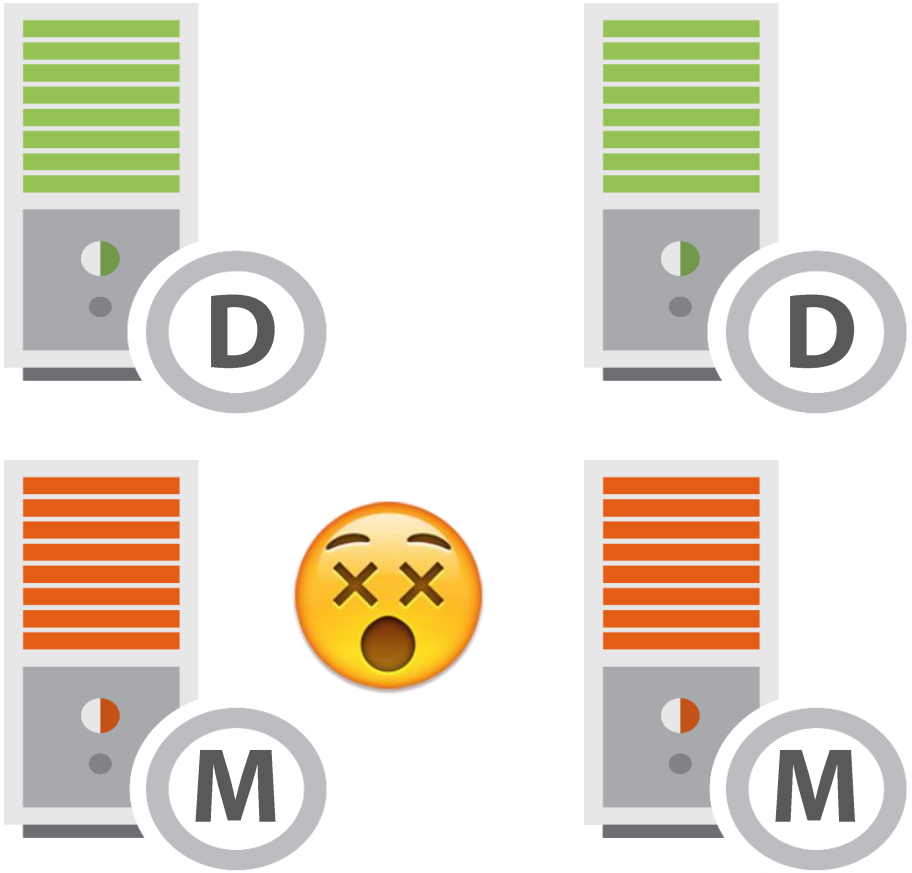
Capacity Planning



Capacity Planning: Data Nodes



Capacity Planning: Master Nodes



Split brain scenarios are not good

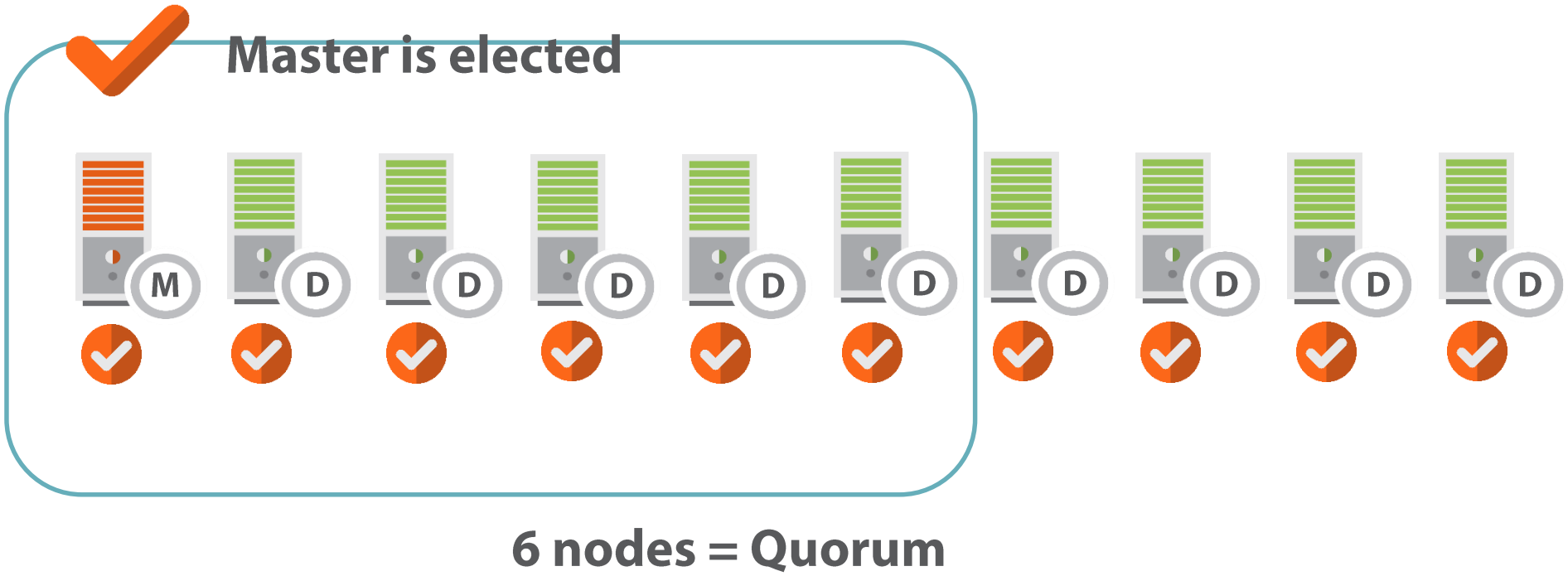
Capacity Planning: Master Nodes

`minimum_master_nodes: x`

How many master nodes must be present to elect a master?

Good Rule: number of master nodes / 2 + 1

Capacity Planning: Master Nodes

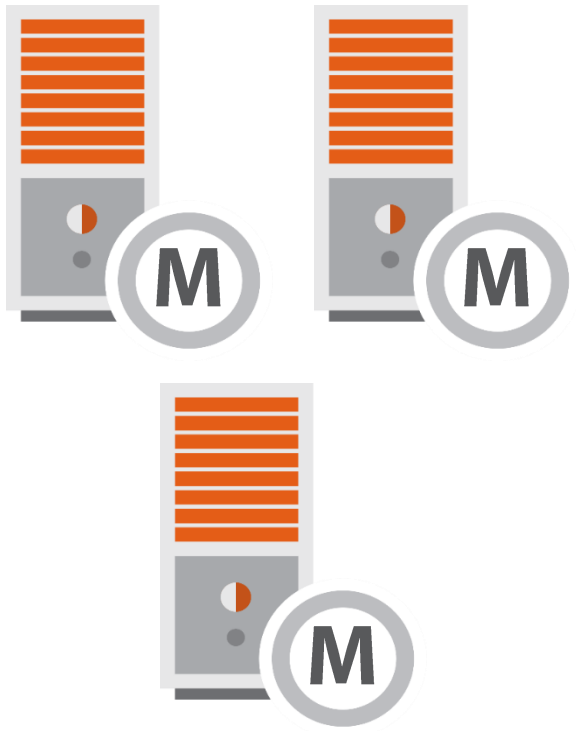


6 nodes = Quorum

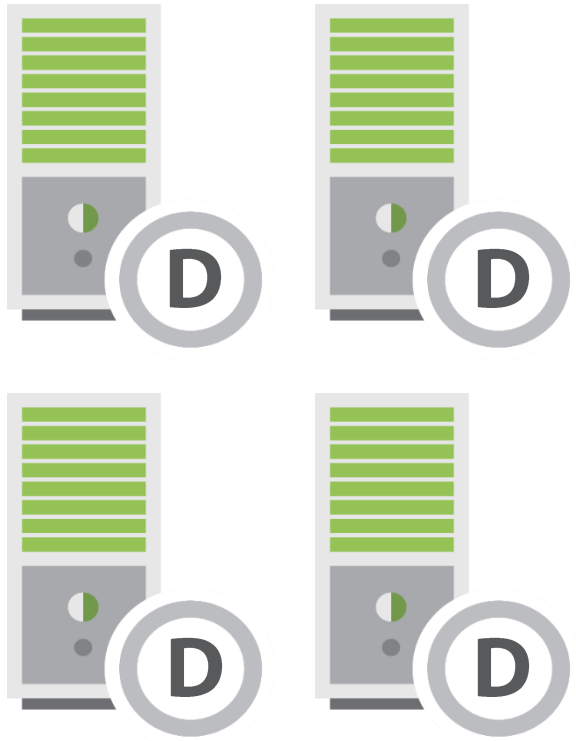
Capacity Planning: Master Nodes

2 nodes = Quorum

Capacity Planning: Master Nodes

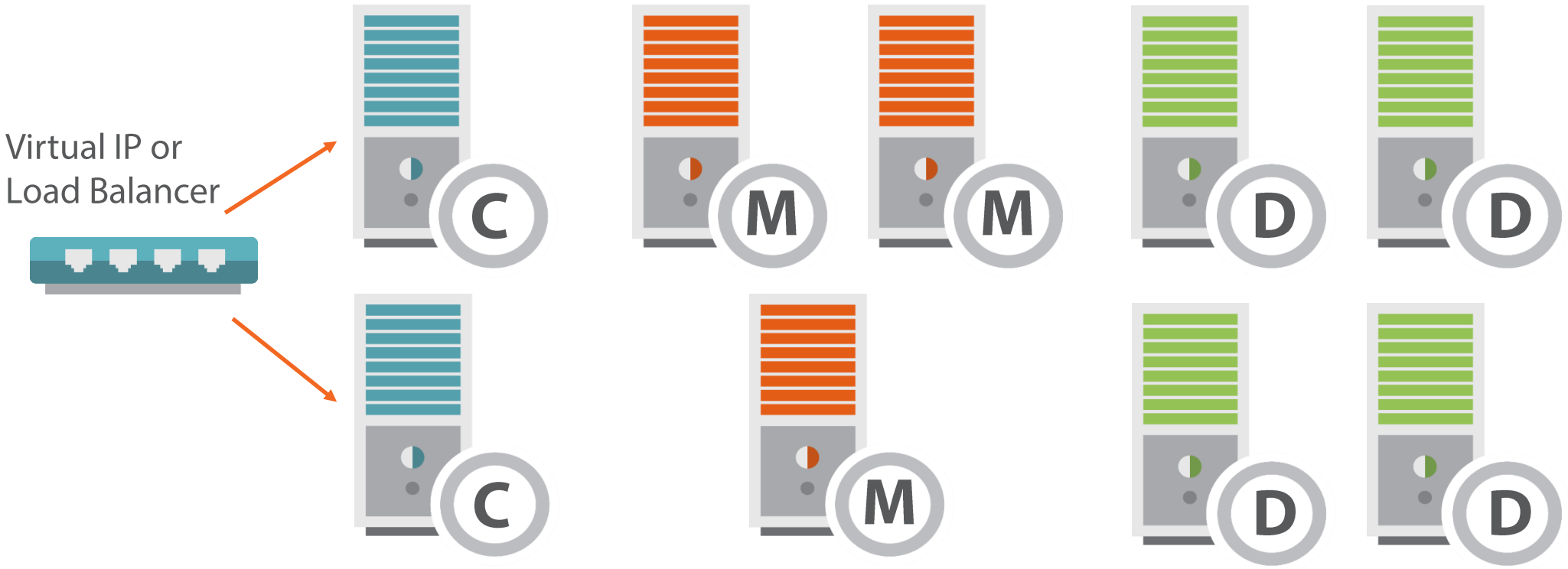


Dedicated master-eligible nodes x3



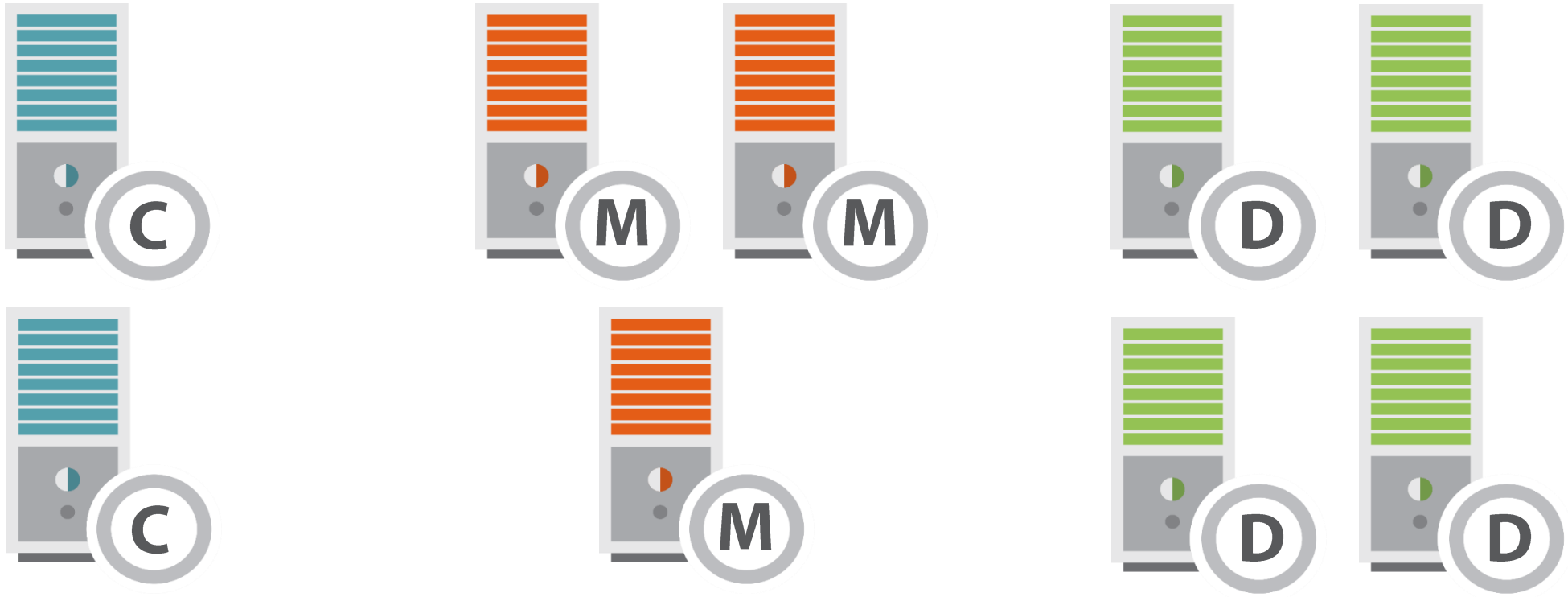
Data nodes x4

Capacity Planning: Client Nodes



Helps direct queries and take load off of the data nodes

Complete Production Cluster



Nine total Elasticsearch nodes

Server Requirements

Physical servers?

Virtual machines?

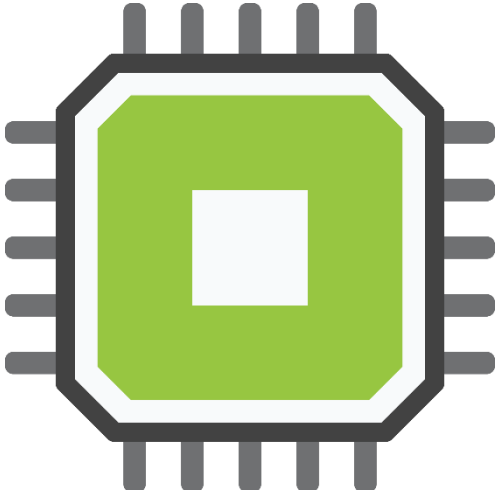
CPU

RAM

Disk

Depends on node role and data

Server CPU Selection

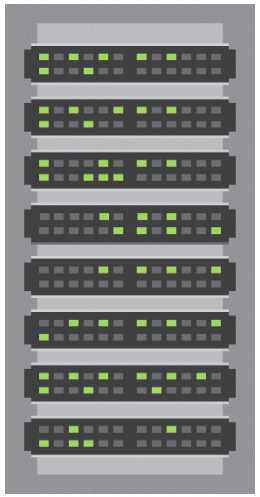


The more cores the better

Favor cores over clock speed

Better to run concurrent operations than each operation slightly faster

Server RAM

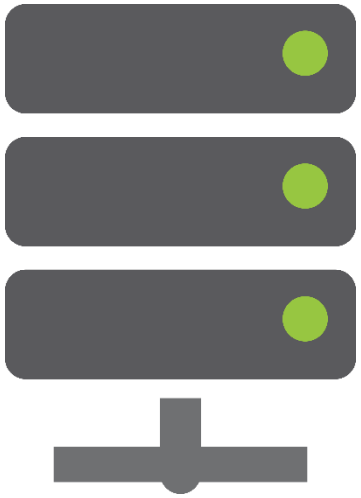


JVM has heap limitations to consider

64 GB is the sweet spot for data nodes

Client and master nodes can run 32 GB or less

Server Disks



- Get the fastest disks possible
- Safe to use RAID 0 for more speed even though it's not fault tolerant
- In case of a disk failure, Elasticsearch picks up the slack
- Avoid using network storage

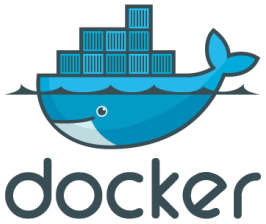
Networking

1 Gigabit Ethernet will suffice

10 Gigabit Ethernet may be better if shards are large

Avoid clustering across datacenters or different geographical locations

Virtual Machines and Docker



Great for easily adding nodes

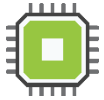
Make sure the hosts aren't too busy and slow down the virtual machines

Do not use for data nodes in production

Try not to use too many, could cause operations management difficulties

Final Specs: Data Nodes

64 GB of RAM



4 core CPU

4 x 1 TB SSD disks in RAID 0

Optional: OS on its own disk array

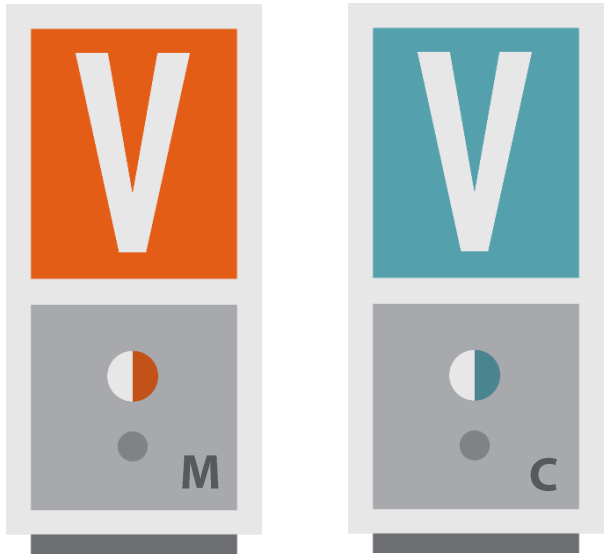
Final Specs: Master and Client Nodes

32 GB of RAM

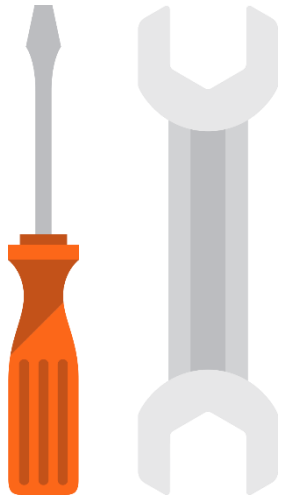
2 CPU cores

20 GB disk space

Avoid putting more than one node of the same role on the same virtual machine host!



Operating System and Elasticsearch Settings



Almost ready!

There are a few very important settings

Mostly Elasticsearch but some OS level

Orchestration and Automation is Helpful

Configuration management will save you

For this course, we're configuring manually

Consider using a good tool to help automate all configuration

Configuration: elasticsearch.yml

```
cluster.name:     “globomantics_kb_production”
node.name:        “es-master-0x”
```

Use a cluster name that makes sense for your application and your organization

Using node names that give semantic hints of their function is also a big help

Configuration: elasticsearch.yml

```
path.data:          “/volume/data”
path.logs:          “/var/logs/special_folder”
path.plugins:       “/volume/plugins”
```

You probably only need to modify these if you customize the installation

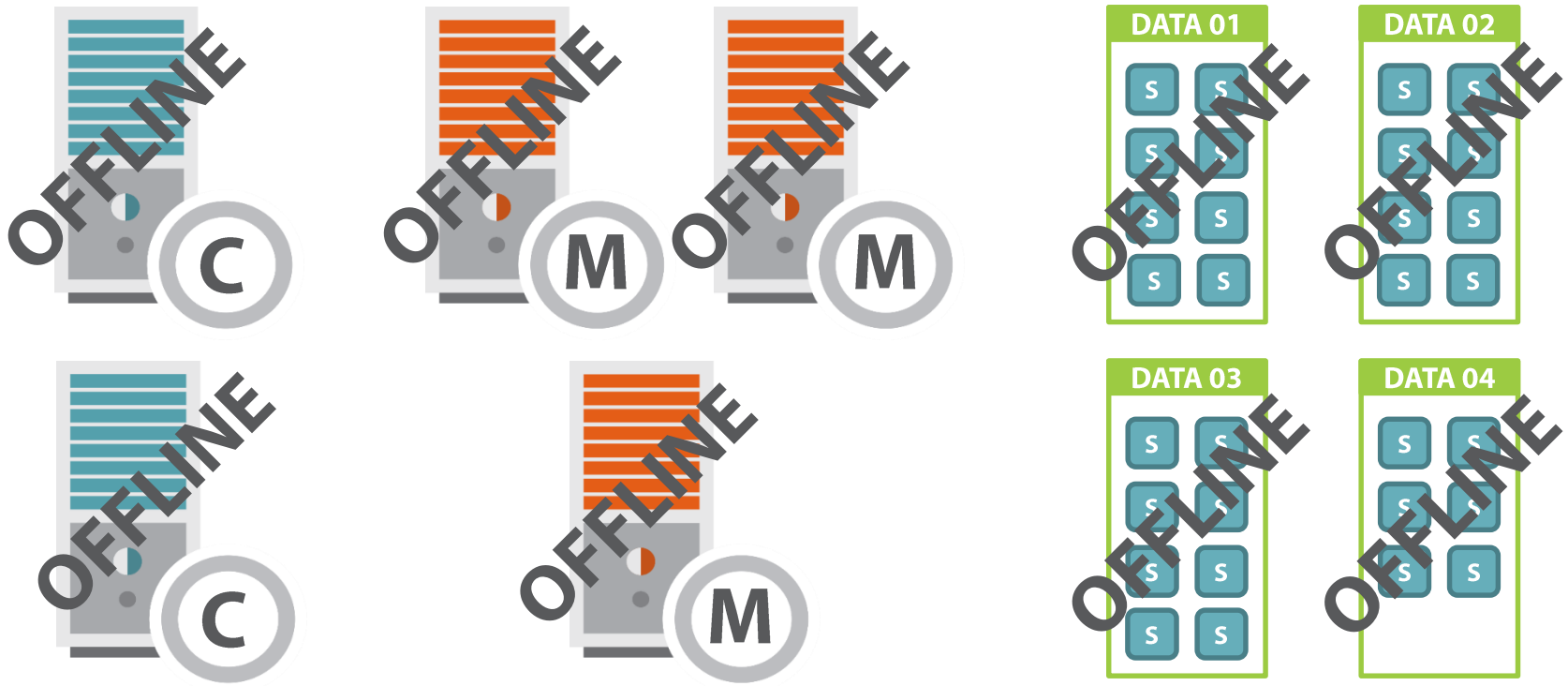
If you setup multiple disks or arrays and plan to store data on them, you may also want to modify these accordingly

Configuration: elasticsearch.yml

```
minimum_master_nodes: 2
```

Important! We want two master eligible nodes before we elect a master

Configuration: elasticsearch.yml



Tons of unnecessary disk and network activity!

Configuration: elasticsearch.yml

```
gateway.recovery_after_nodes: 8
gateway.expected_nodes: 9
gateway.recover_after_time: 3m
```

Don't worry, this is an avoidable situation

Wait until at least 8 nodes are present, and start recovery when either 3 minutes passes or all 9 nodes are present and active

Configuration: elasticsearch.yml

This is almost never good

Configuration: elasticsearch.yml

```
discovery.zen.ping.unicast.hosts: “host1”, “host2”…
discovery.zen.ping.multicast.enabled: false
```

List your nodes by host or by IP address

Should be setup on each node in the same way

Configuration: elasticsearch.yml

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

Makes node "master eligible"

Will not allow the node to service any shards

Configuration: elasticsearch.yml

```
node.client:    true
node.data:      false
```

Makes node "master ineligible"

Will not allow the node to service any shards

Configuration: elasticsearch.yml

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

Makes node "master ineligible"

Will allow the node to service shards

Configuration: elasticsearch.yml

```
http.enabled:  false
```

We don’t want to allow data nodes to be queried

We _could_ disable http on the master nodes too

Configuration: Java JVM

The JVM has lots of tunables – and you really shouldn't mess with them

Don't change garbage collection or threadpools

DO change heap settings

Configuration: Java JVM

Default heap = 1 GB (way too small)

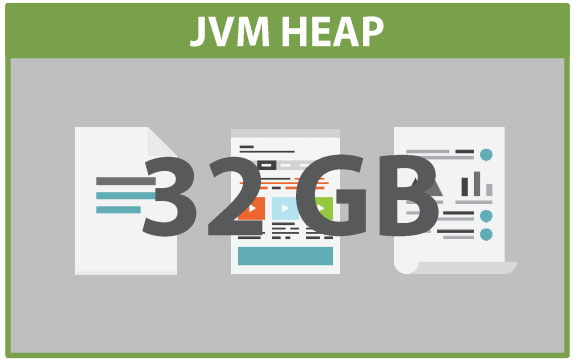
Data nodes = 64 GB RAM / 2 = 32 GB heap

Why?

1) 32 GB is enough room to work with

2) Lucene needs the rest of the RAM

3) Heap sizes > 32 GB are inefficient




```
ES_HEAP_SIZE=  32g
```

Configuration: Swap

Most systems come with a swap file configured

Can harm performance drastically

Consult OS docs and remove /swapfile

Alternatively, modify elasticsearch.yml

```
bootstrap.mlockall:     true
```

Configuration: MMap and File Descriptors

```
File Descriptors:        64,000
MMap:                    262144
```

File descriptors: How many files can a process use?

Settings can vary by OS

Typically set way too low

Planning and Configuration: Done!

- Learned how Elasticsearch works
- Planned roles for nodes
- Decided on hardware and specs
- Finalized configurations for each node