Afi Escuela

Ecosistema Hadoop

Máster en Data Science y Big Data

Jorge López-Malla Matute
jlmalla@geoblink.com

Febrero 2022

Presentación

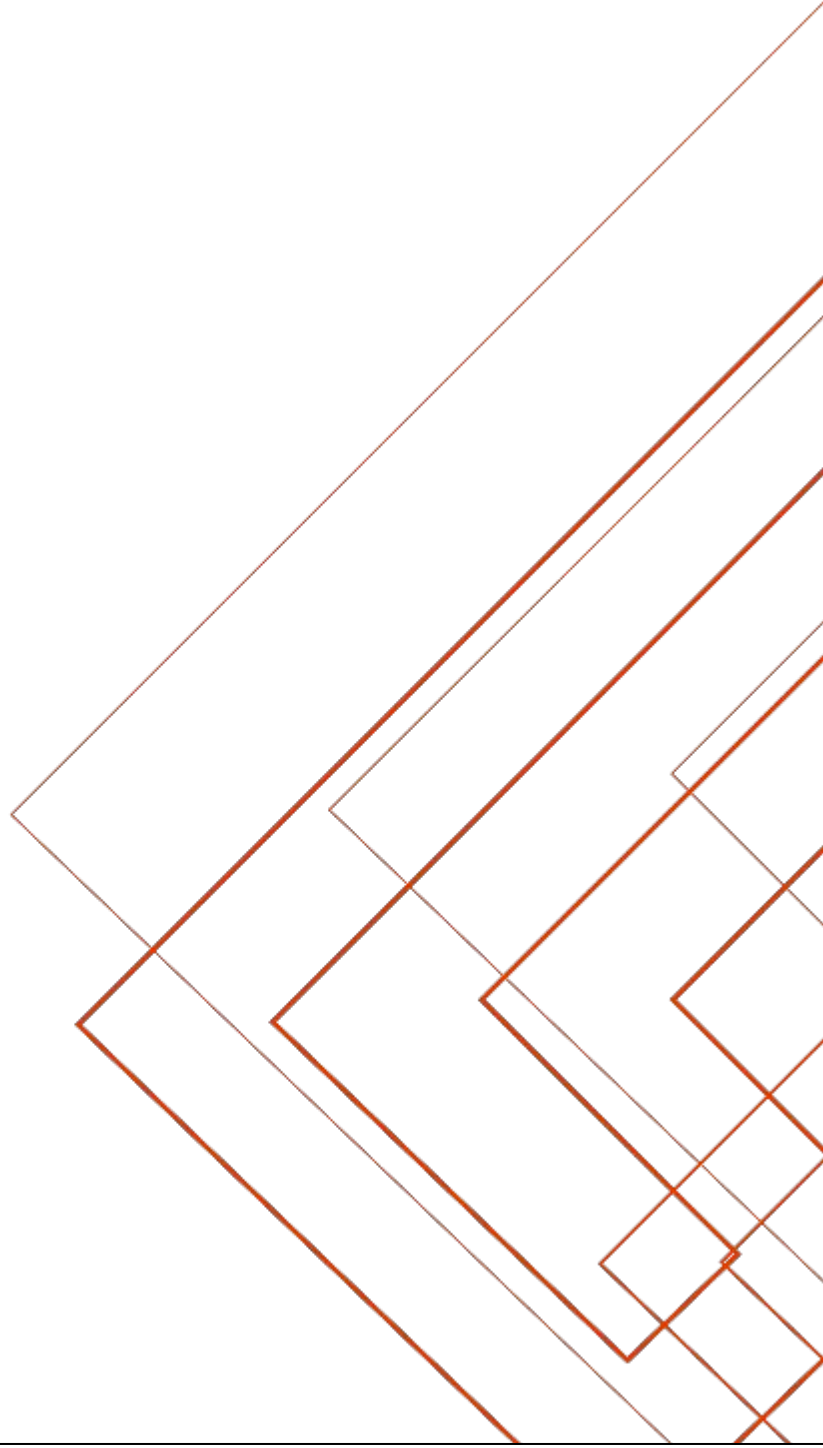
- Jorge López-Malla Matute
- Puesto actual:
  - Senior Data Engineer en Geoblink
- Experiencia docente:
  - Profesor en diversos Masters de Big Data durante los últimos 6 años
  - Profesor de Tecnologías Masivas en ICAI
- Años trabajados en Big Data: 9 años

Índice

Contenido

1. Ecosistema

2. Componentes
    - Apache Hive
    - Apache HBase
    - Apache Flume
    - Apache Mahout



Ecosistema

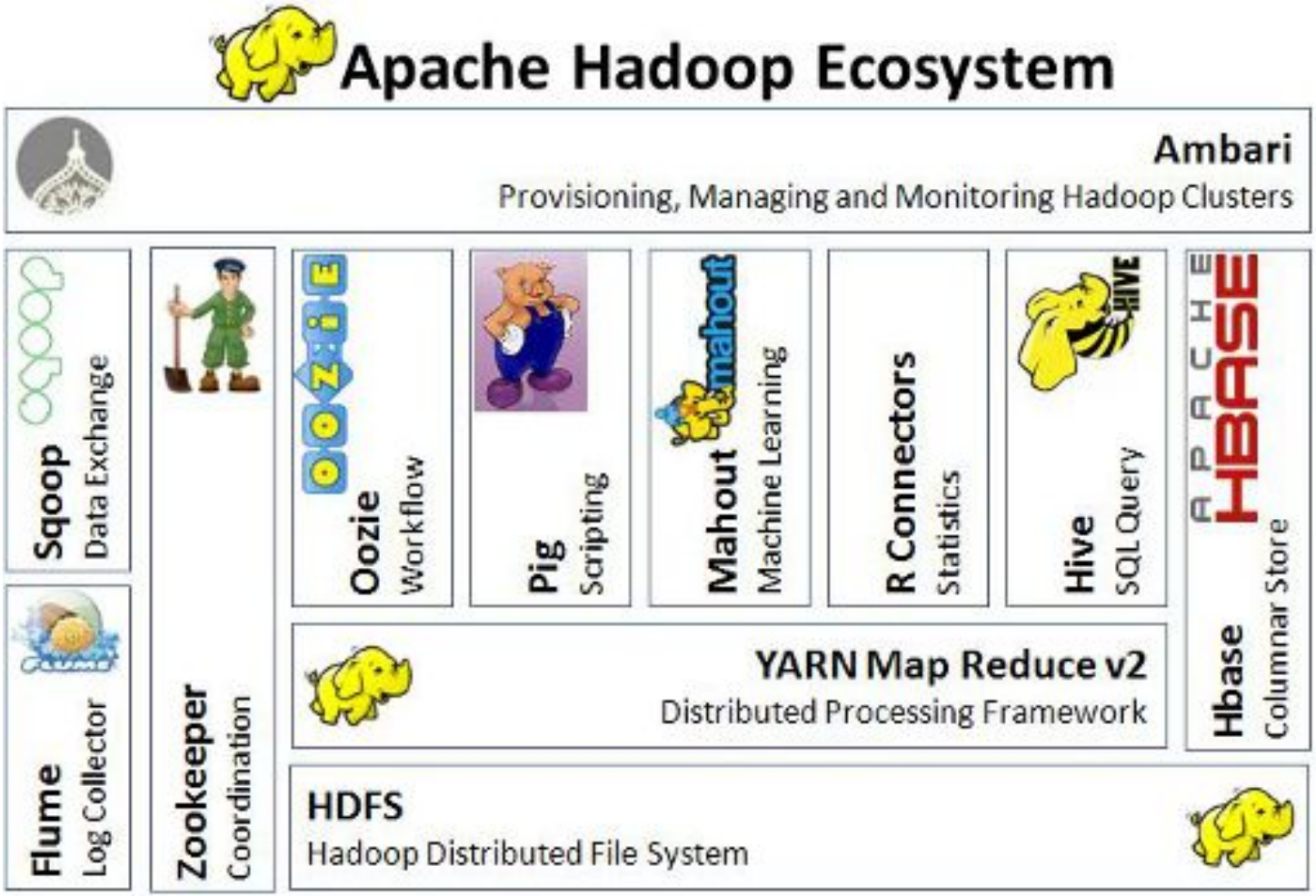
Ecosistema

- En el momento de naciemiento del Big Data existen multitud de problemas que se podrían resolver usando tecnologías Big Data
- Muchos de esos problemas requerían nuevas tecnologías que solventan nuevos casos de uso
- Las primeras casuisticas en resolverse fueron el almacenamiento y el procesamiento
- Aunque se mejora el procesamiento. los tiempos de procesamiento y cantidad de datos a procesar necesitaban de control de los procesamientos
- Asi es como nacieron HDFS, Map & Reduce y YARN

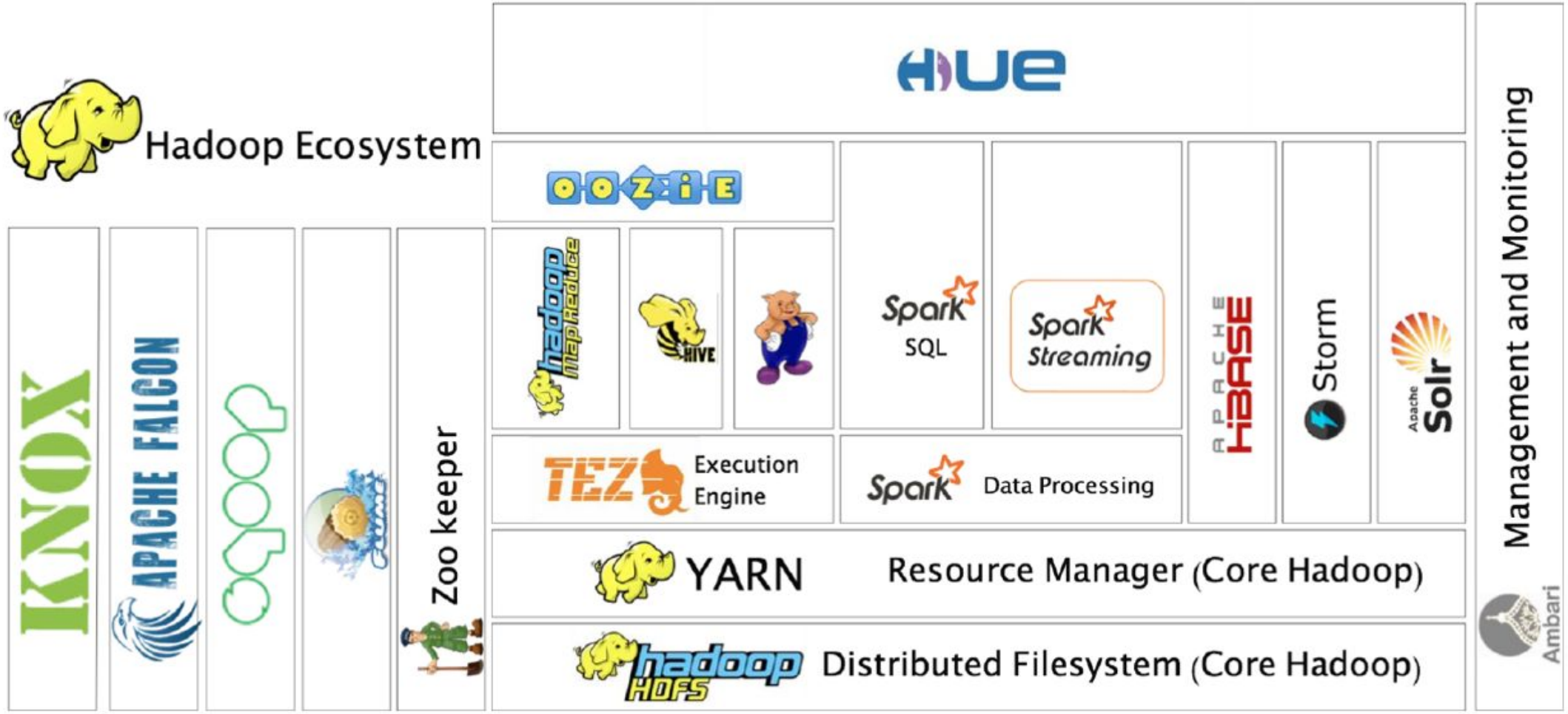
Ecosistema

- Con estas soluciones desarrolladas empiezan a surgir nuevas tecnologías para solucionar estos problemas
- Todas ellas se fundamentan en los tres proyectos que se conocen como Hadoop Core, YARN, HDFS y Map & Reduce
- Conforme van creciendo estos proyectos hay proyectos que no se basan directamente en estos tres proyectos pero los usan
- Todo esto hace que se hable de Ecosistema de Hadoop

Ecosistema

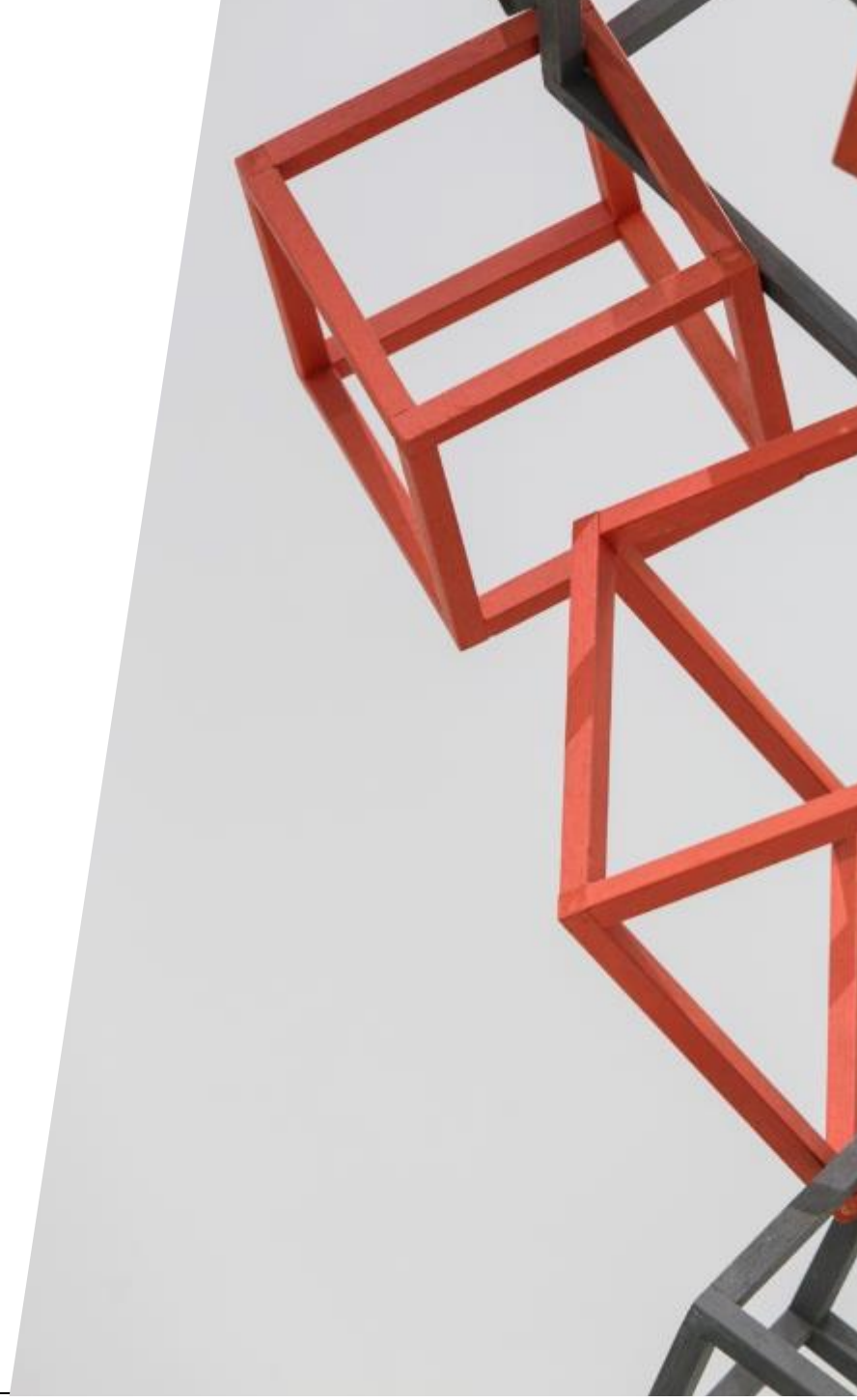

Apache Hadoop Ecosystem
Ambari
Provisioning, Managing and Monitoring Hadoop Clusters
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
Oozie
Workflow
Pig
Scripting
Mahout
Machine Learning
R Connectors
Statistics
Hive
SQL Query
APACHE HBASE
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System

Ecosistema


Hadoop Ecosystem
HUE
OOZIE
KNOX
APACHE FALCON
Zoo keeper
hadoop MapReduce
HIVE
Spark SQL
Spark Streaming
APACHE HBASE
Storm
Apache Solr
TEZ Execution Engine
Spark Data Processing
YARN Resource Manager (Core Hadoop)
hadoop HDFS Distributed Filesystem (Core Hadoop)
Management and Monitoring
Ambari

Componentes

Apache Hive



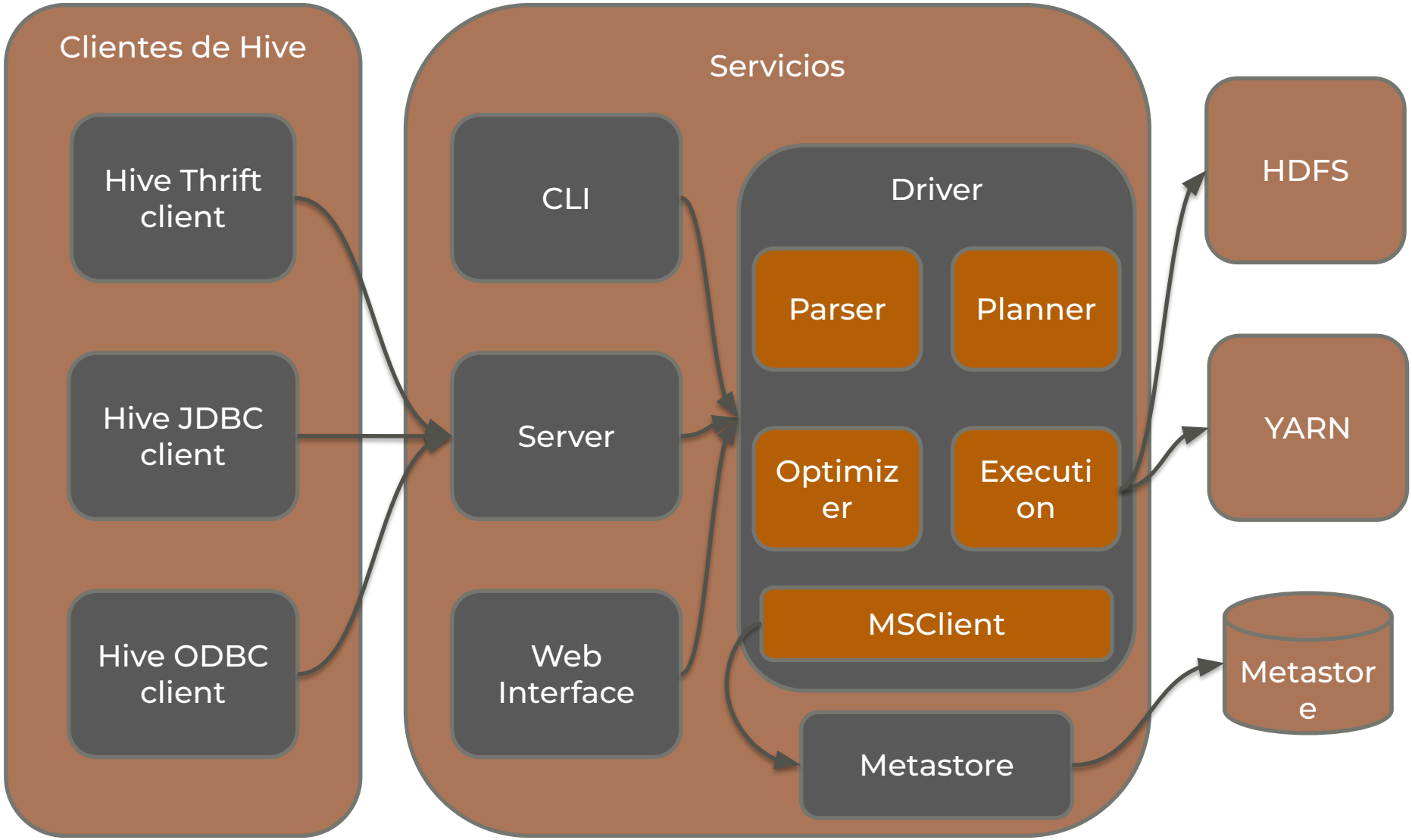
Hive- Introducción

- En el inicio **Map & Reduce** sólo estaba enfocado al desarrollo
- Con el tiempo más casos de uso se fueron desarrollando y se vio la necesidad de lenguajes de scripting
- El primer proyecto de Hadoop en scripting fue **Apache Pig** (2008)
- Los usuarios tenían que seguir aprendiendo otro lenguaje
- No podía conectarse con otros servicios externos con facilidad como lo hubiera hecho el SQL

Hive- Introducción

- En 2010 Facebook libera un proyecto que convierte sentencias SQL en trabajos de Map&Reduce
- Se libera Apache Hive y se extiende su uso
- Al ser un intérprete de SQL permite que herramientas que ya usaban SQL pudieran procesar cantidades masivas de datos
- Hive además permite la compartición de tablas de datos entre distintos procesos de una manera sencilla
- Los proyectos posteriores de procesamiento distribuido toman Hive como referencia

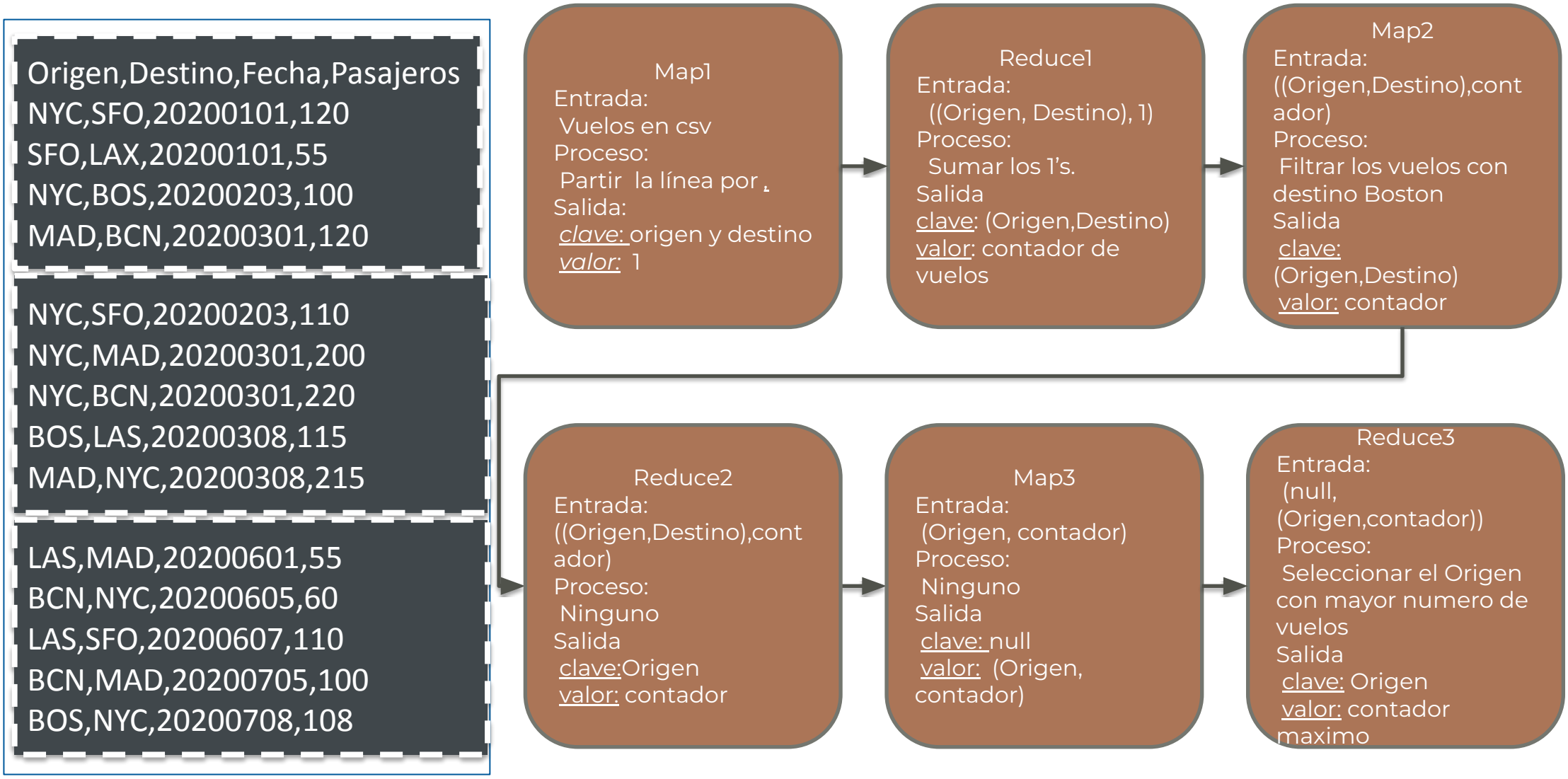
Hive- Arquitectura



Hive- Funcionamiento

- **Hive** interpreta sentencias SQL en un motor de procesamiento distribuido
- **Hive** necesita de una base de datos de metadatos
- En ella se guardan los metadatos de las tablas
- Las tablas tienen que almacenarse en un almacenamiento compatible con el motor de procesamiento
- **Hive** interpreta la secuencia y la convierte en trabajo distribuido más eficiente

Hive- Funcionamiento

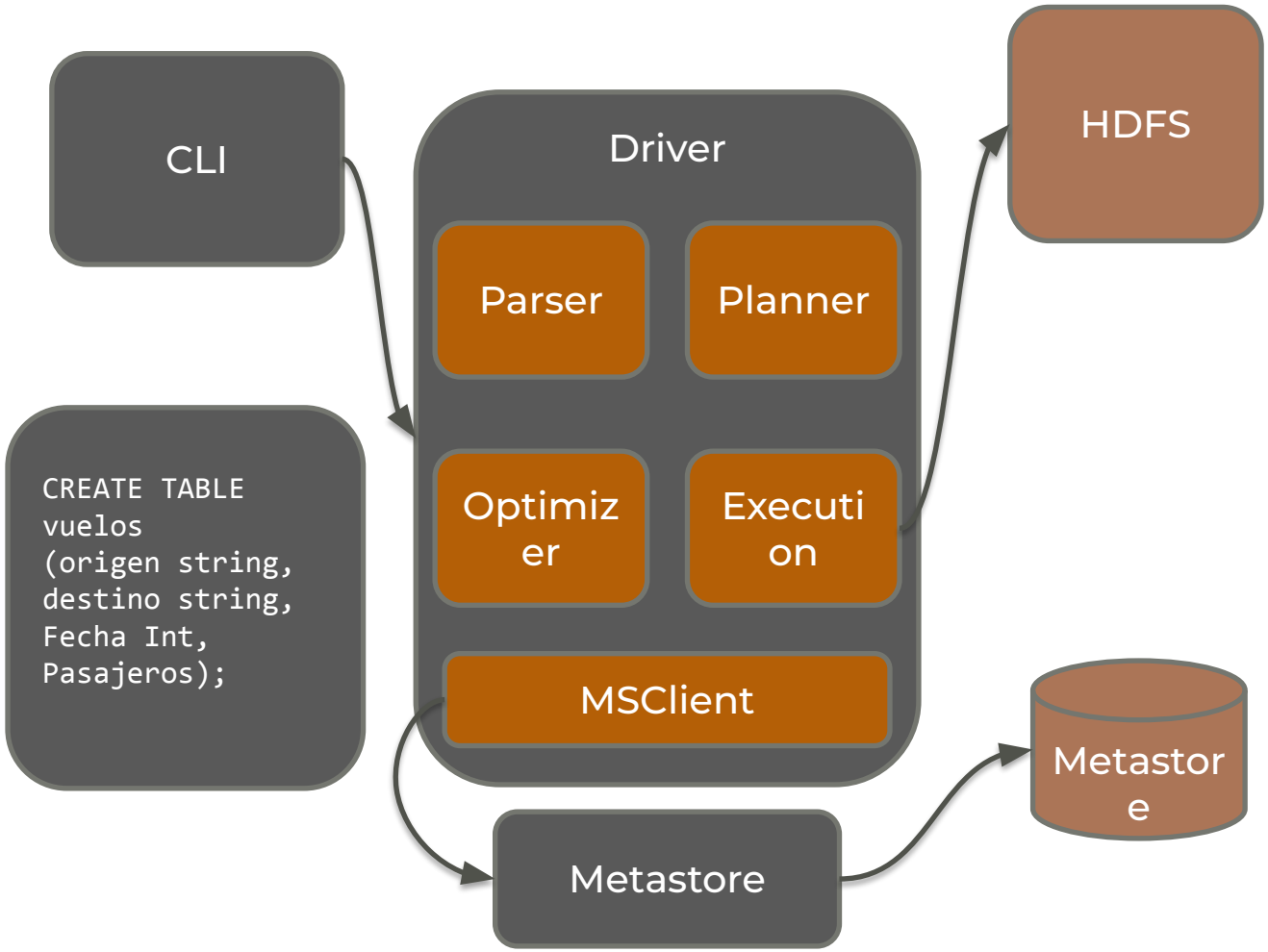


Hive- Funcionamiento

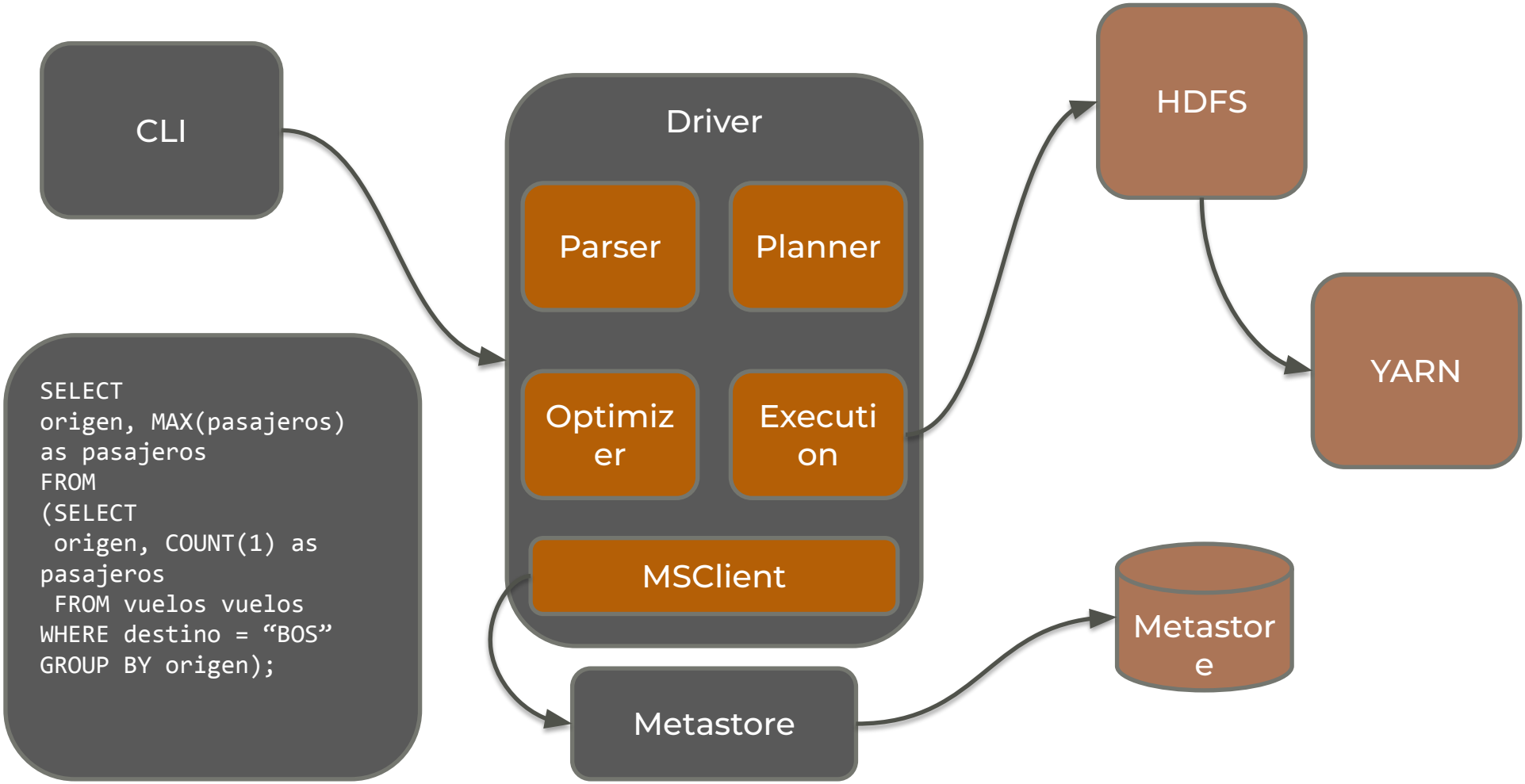
```
Origen,Destino,Fecha,Pasajeros
NYC,SFO,20200101,120
SFO,LAX,20200101,55
NYC,BOS,20200203,100
MAD,BCN,20200301,120
```

```
NYC,SFO,20200203,110
NYC,MAD,20200301,200
NYC,BCN,20200301,220
BOS,LAS,20200308,115
MAD,NYC,20200308,215
```

```
LAS,MAD,20200601,55
BCN,NYC,20200605,60
LAS,SFO,20200607,110
BCN,MAD,20200705,100
BOS,NYC,20200708,108
```



Hive- Funcionamiento



Hive- Funcionamiento

```
SELECT
origen, MAX(pasajeros) as
pasajeros
FROM
(SELECT
 origen, COUNT(1) as pasajeros
 FROM vuelos vuelos
WHERE destino = “BOS”
GROUP BY origen);
```

Map1
Entrada:
tabla vuelos en csv
Proceso:
filtrar los vuelos con destino BOS
Salida:
clave: origen
valor: 1

Combiner1
Entrada:
(Origen, [1’s])
Proceso:
Sumar los 1’s
Salida
clave: Origen
valor: contador

Reduce1
Entrada:
(Origen, [contador])
Proceso:
Sumar *contador*
Salida
clave: Origen
valor: Sum(*contador*)

Map2
Entrada:
(Origen,sum(*contador*))
Proceso:
Ninguno
Salida
clave: Null
valor: (Origen,sum(*contador*))

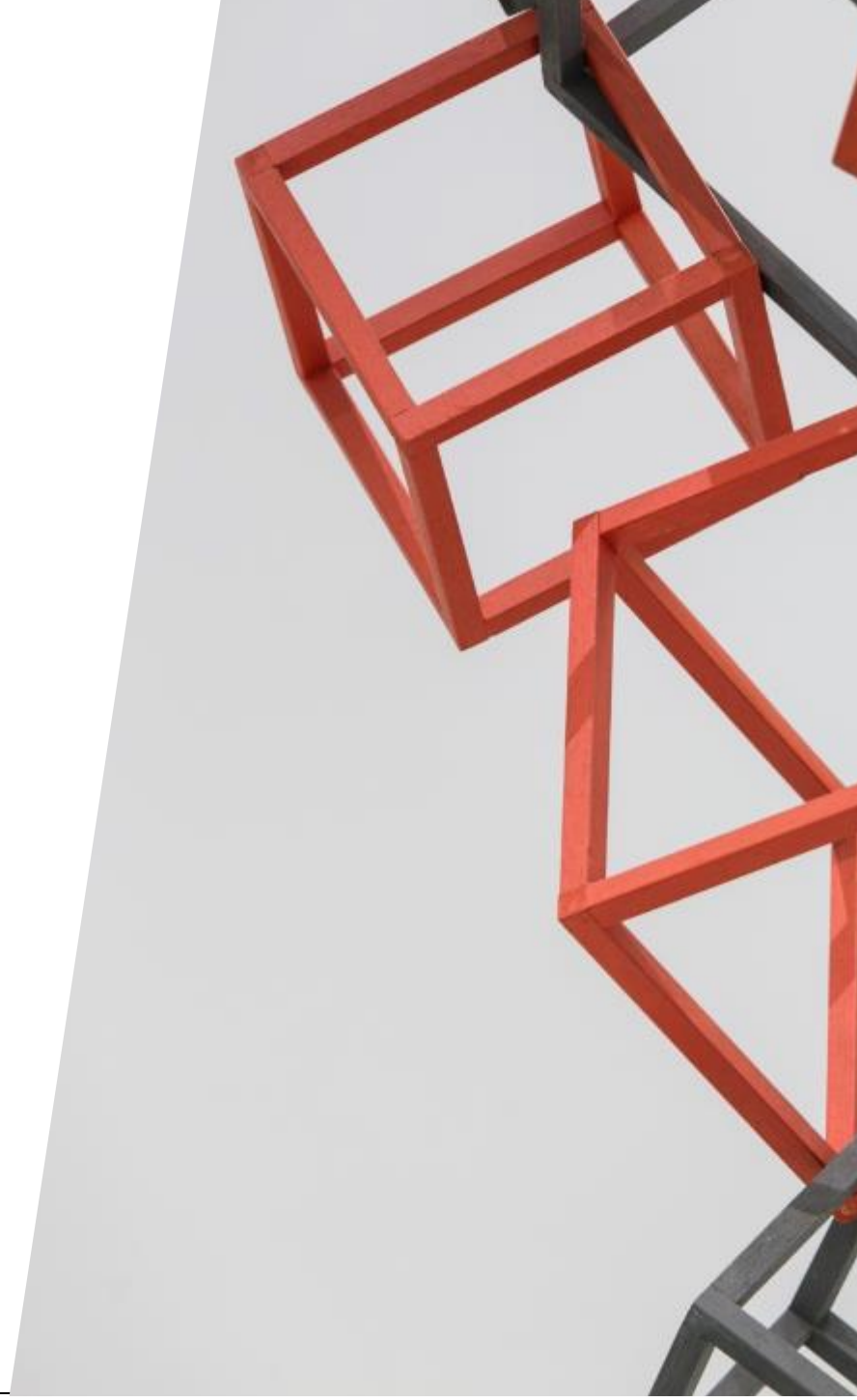
Combiner2
Entrada:
(Null,[(Origen, contador)])
Proceso:
Obtener el máximo
Salida
clave: Null
valor: max((Origen, contador))

Reduce2
Entrada:
(Null,[max((Origen, contador))])
Proceso:
Obtener el máximo
Salida
clave:Origen
valor:

Hive- Introducción

- La solución de **Hive** da la solución óptima mirando el planificador
- El planificador de queries no entiende de datos ni de escalabilidad
- Puede hacer algún tipo de optimización usando el almacenamiento
- ¿Y si los vuelos con destino a Boston suman tanto como para llenar la partición más grande de nuestro sistema?
- Se podría solucionar con tablas intermedias

Apache Hbase



HBase - Introducción

- Map & Reduce procesa grandes cantidades de datos en un tiempo aceptable
- Hive nos proporciona un lenguaje SQL
- Si juntamos las dos tecnologías, ¿tendríamos una Base de Datos relacional con todas ventajas del Big Data?
- La respuesta es NO
- Juntando una tecnología de cómputo que admita cantidades masivas de datos y un proyecto que nos permita hacer SQL **NO** nos permite hacer queries en tiempo *online* "sólo" facilita el procesamiento

HBase - Introducción

- A raíz de las nuevas formas de almacenar se empiezan a pensar nuevas formas de consultar esa información
- El principal problema que pretende resolver es el acceso aleatorio a unos datos concretos
- Para ello usa HDFS como sistema de ficheros subyacente y una estrategia **Columnar**
- Con ello se consigue un acceso “rápido” a datos aleatorios en colecciones masivas de información
- Posteriormente nacen tecnologías de almacenamiento basadas en su misma filosofía supliendo algunas de sus carencias (Apache Cassandra)

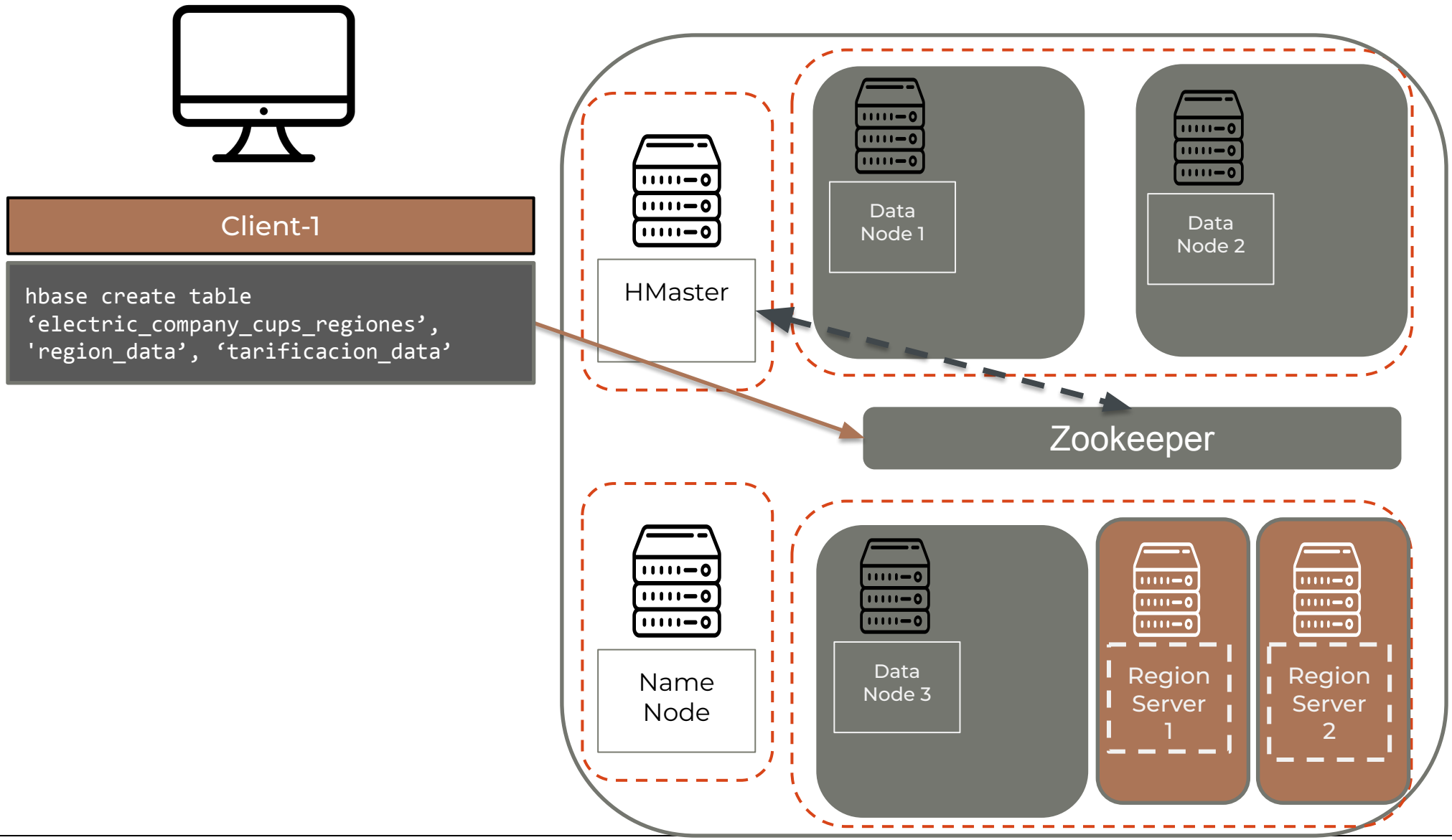
HBase - Introducción

- A raíz de las nuevas formas de almacenar se empiezan a pensar nuevas formas de consultar esa información
- El principal problema que pretende resolver es el acceso aleatorio a unos datos concretos
- Para ello usa HDFS como sistema de ficheros subyacente y una estrategia **Columnar**
- Con ello se consigue un acceso “rápido” a datos aleatorios en colecciones masivas de información
- Posteriormente nacen tecnologías de almacenamiento basadas en su misma filosofía supliendo algunas de sus carencias (Apache Cassandra)

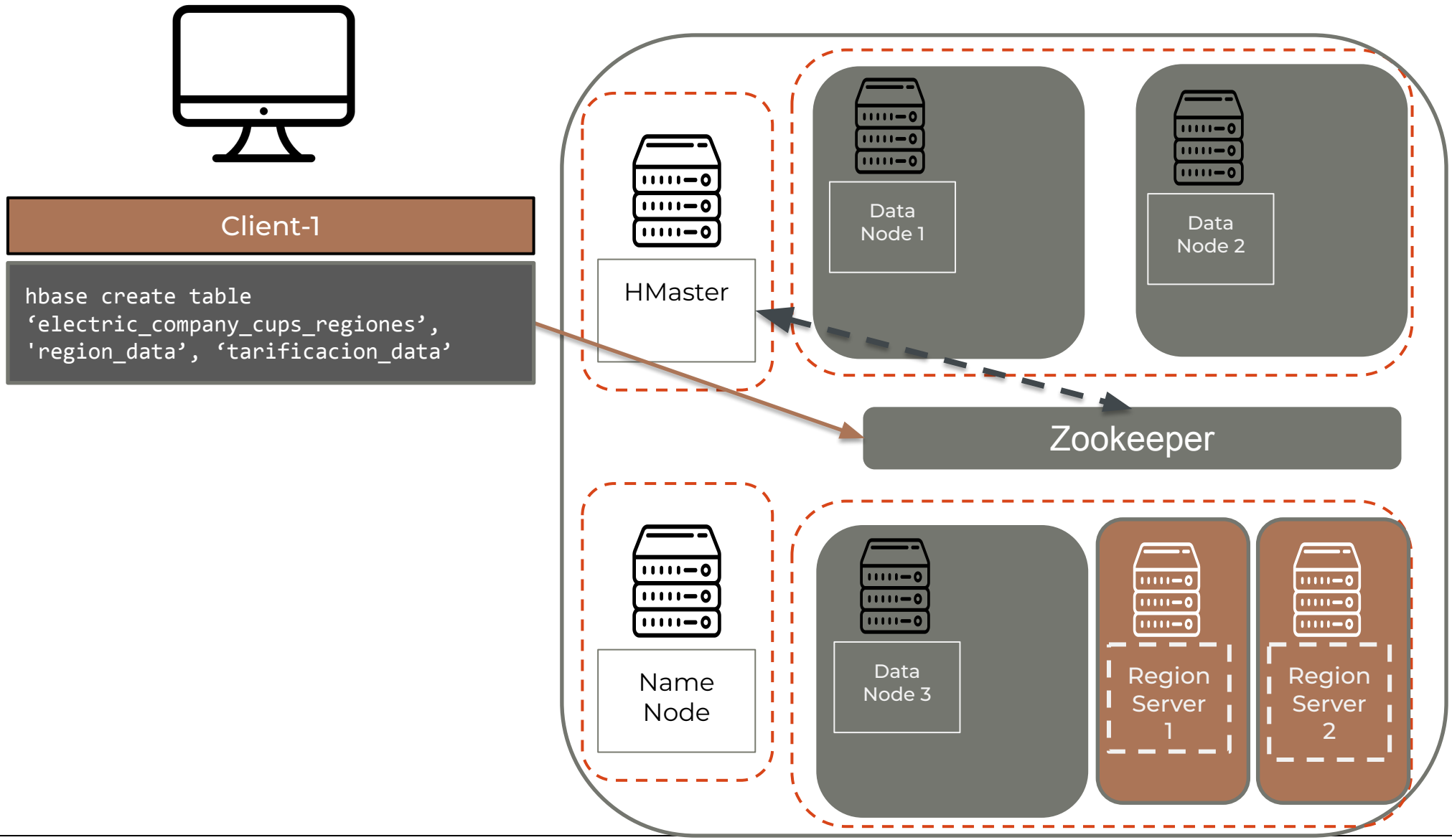
HBase - Terminología

- Hbase tiene la siguiente terminología:
  - **Table**: Conjunto de **rows**
  - **Row:** Conjunto de datos agrupados en columnas. Se compone de una **rowKey** y varias **column families**
  - **RowKey:** Column de una **row** que hace única a la misma**.**
  - **ColumnFamily:** Serie de Columns de una row. Su estrucutra no se tiene que compartir entre distintas **rows** de una misma **table**

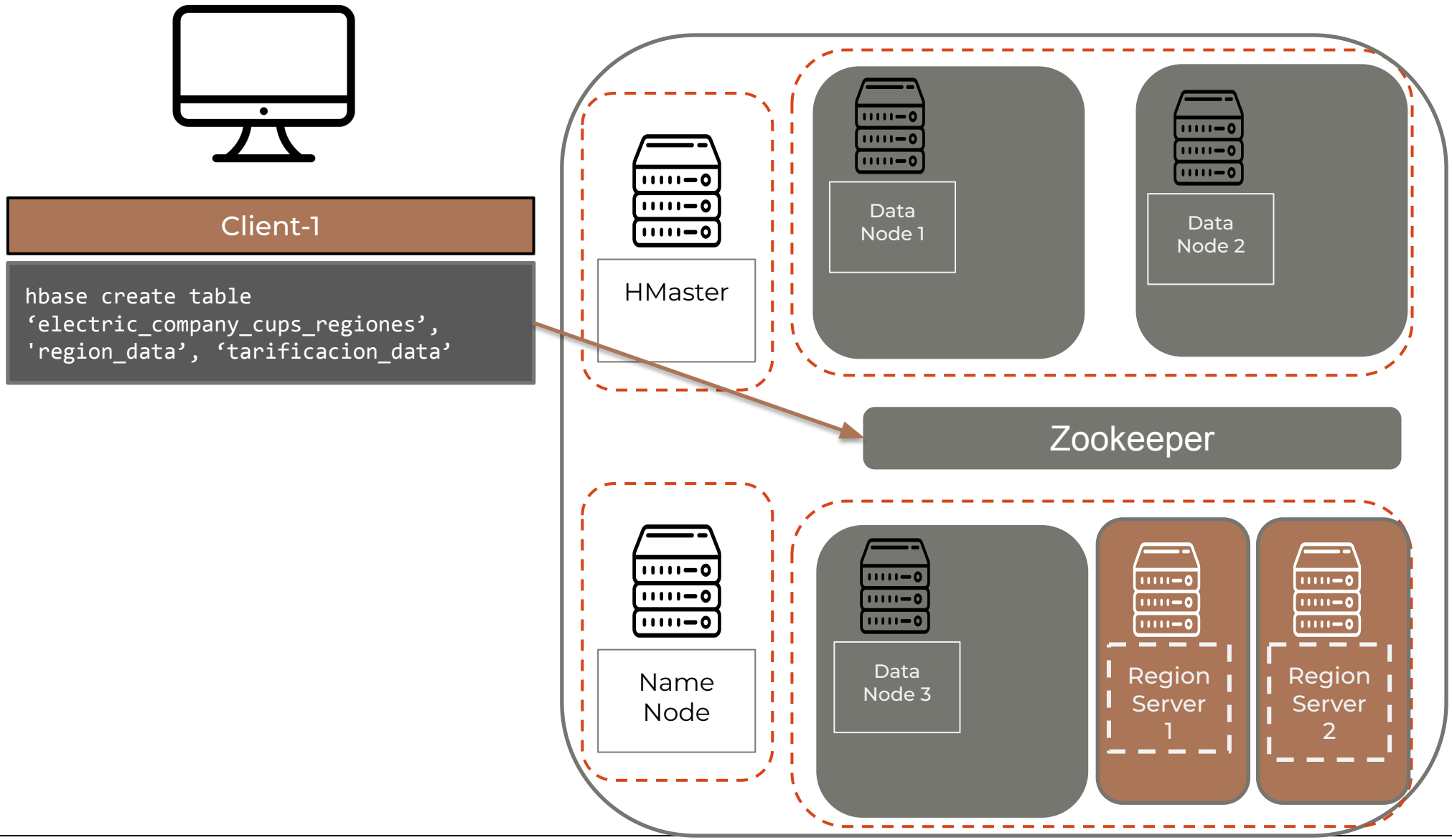
HBase Funcionamiento



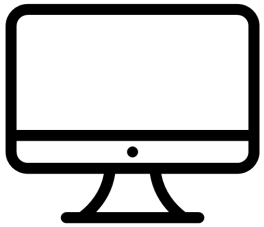
HBase Funcionamiento



HBase Funcionamiento



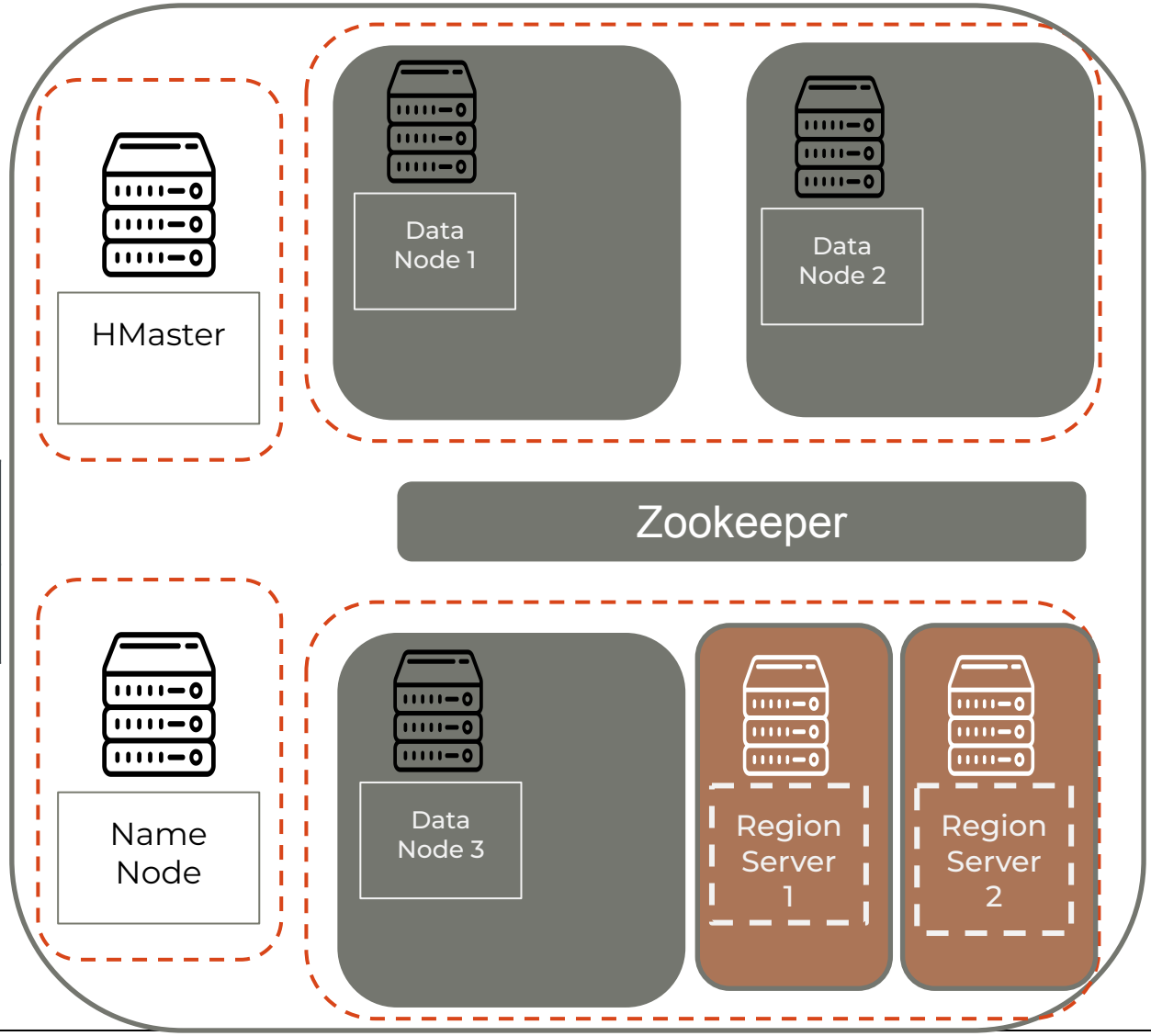
HBase Funcionamiento



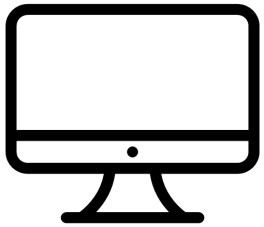
Client-1

```
hbase create table
‘electric_company_cups_regiones’,
'region_data’, ‘tarificacion_data’
```

Row	Region_data	Tarificacion_data	timestamp



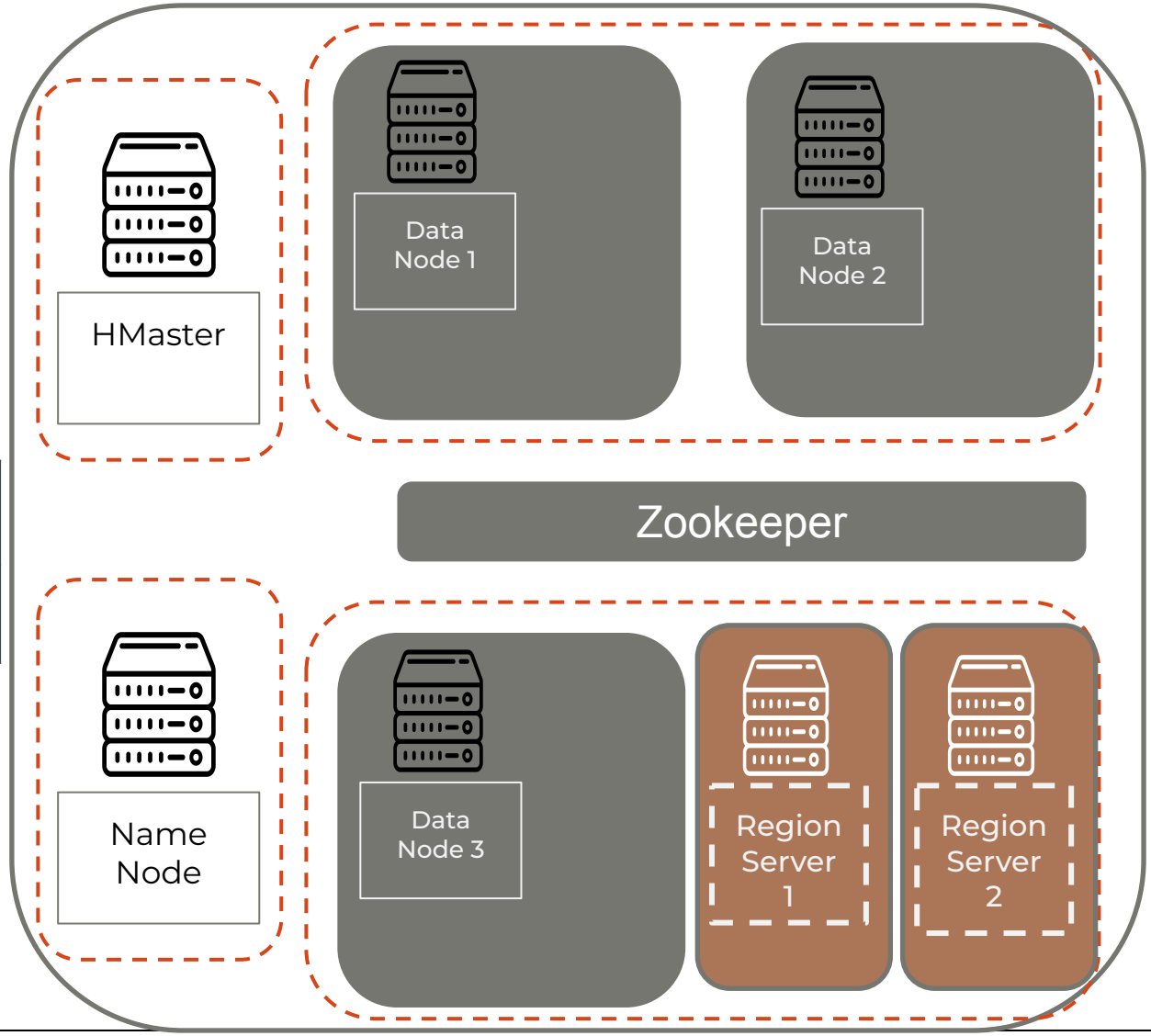
HBase Funcionamiento



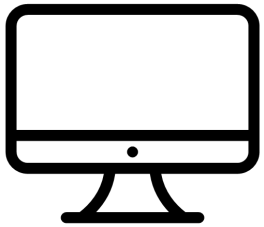
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, 'region_data:valor_region’
‘Colmenar Viejo’
```

Row	Region_data	Tarificacion_data	timestamp



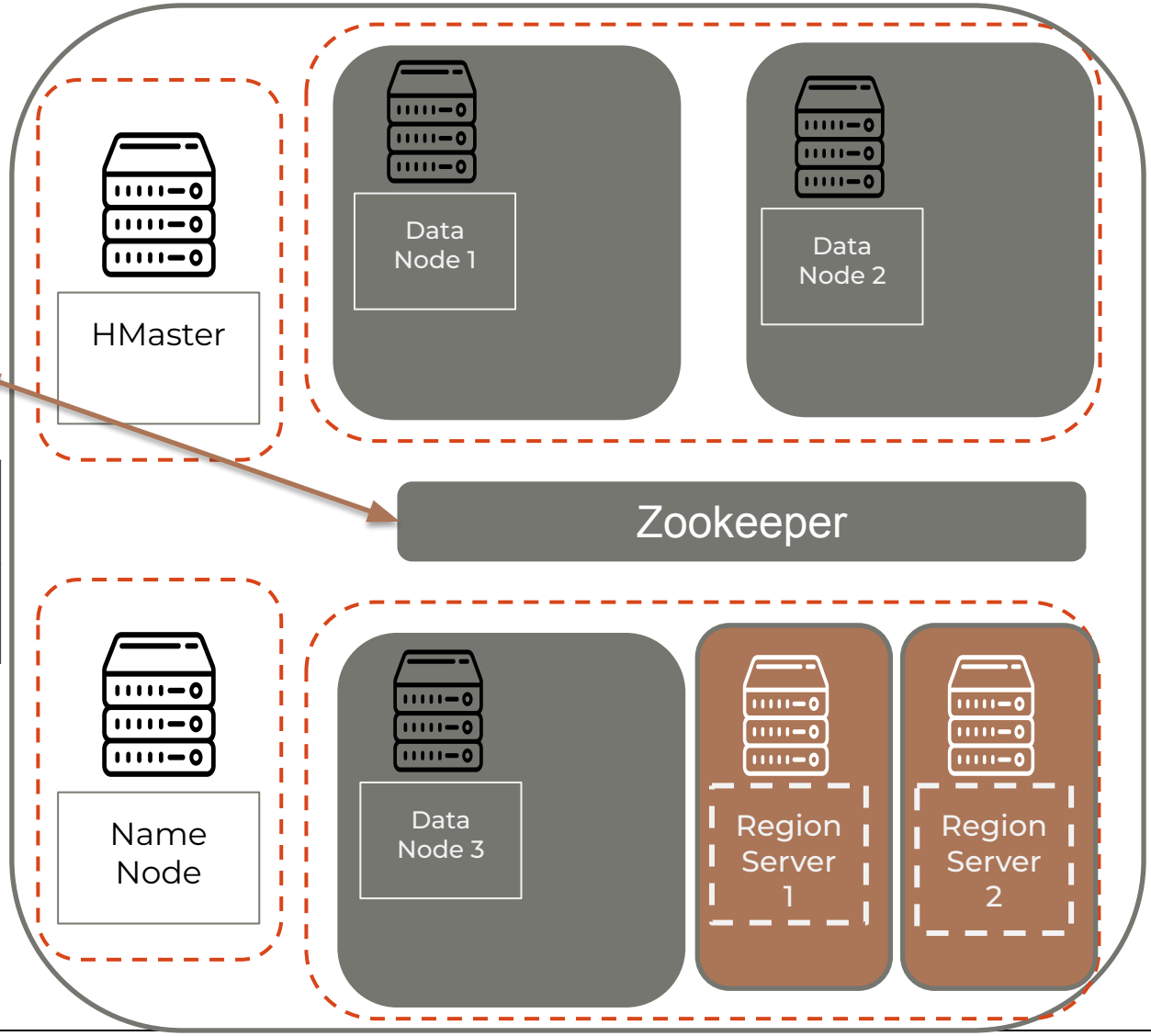
HBase Funcionamiento



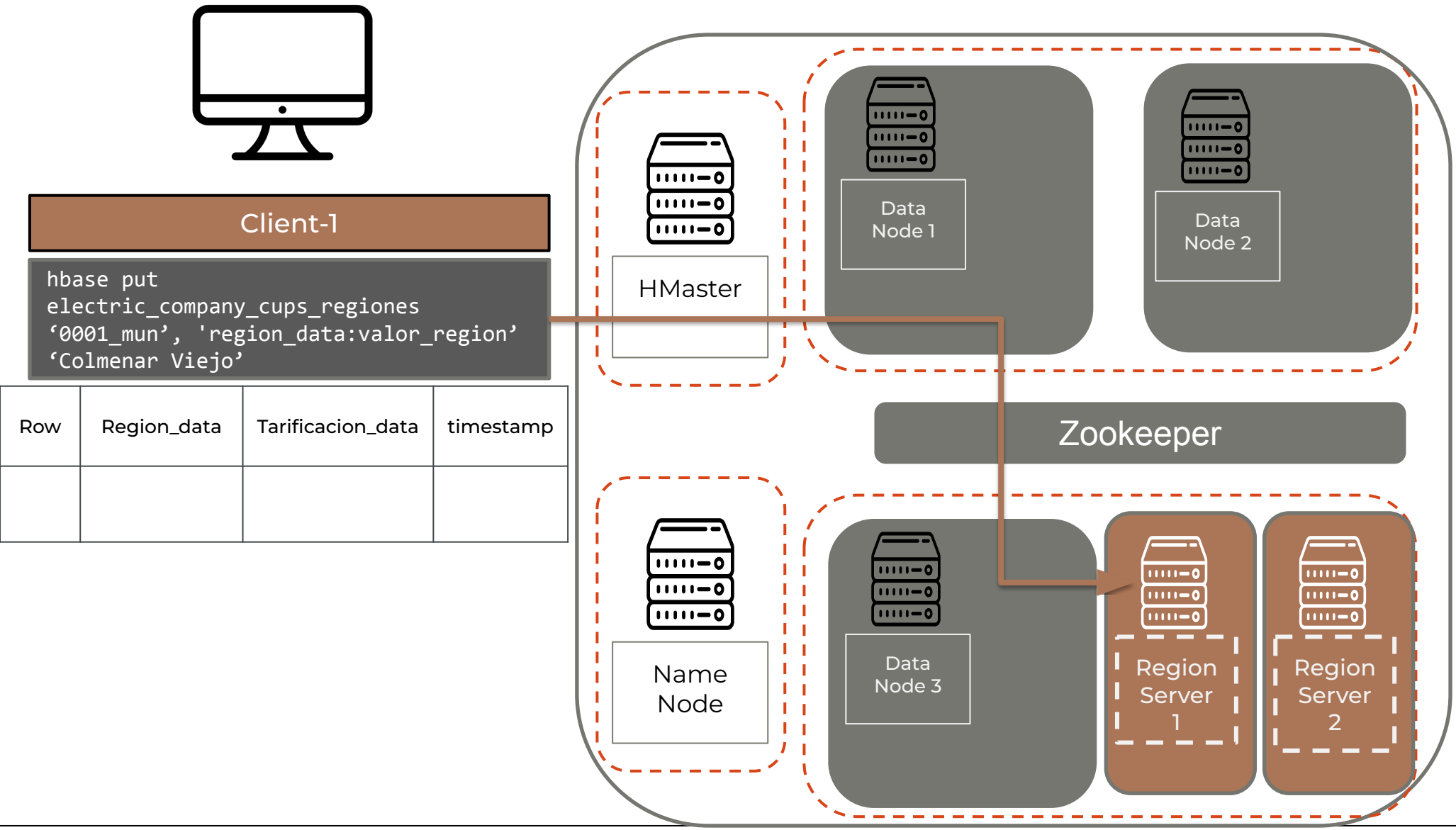
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, 'region_data:valor_region’
‘Colmenar Viejo’
```

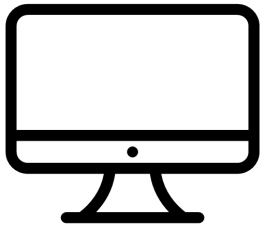
Row	Region_data	Tarificacion_data	timestamp



HBase Funcionamiento



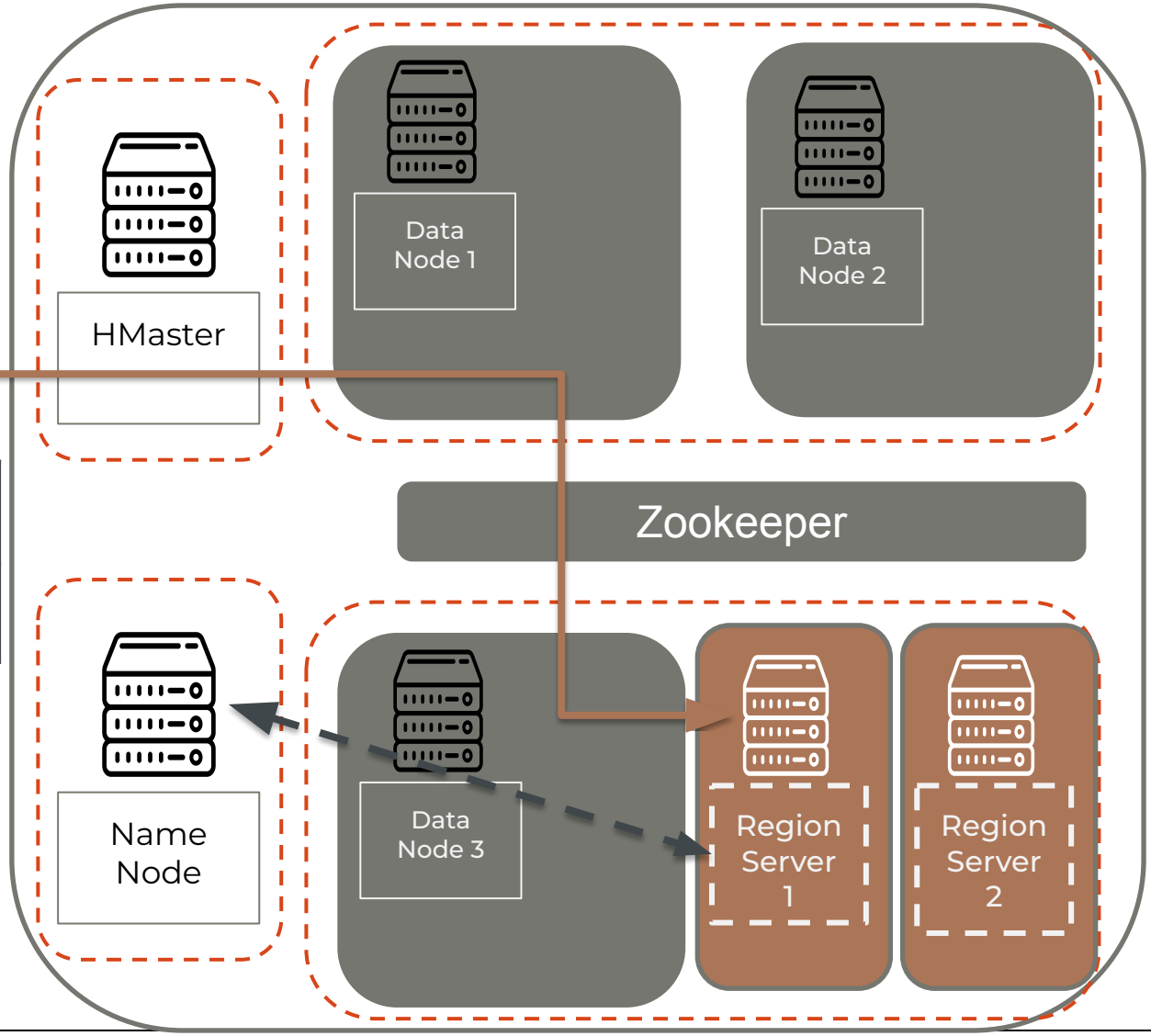
HBase Funcionamiento



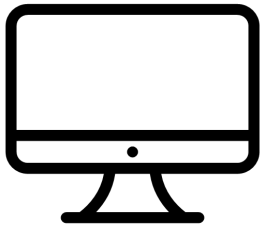
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, 'region_data:valor_region’
‘Colmenar Viejo’
```

Row	Region_data	Tarificacion_data	timestamp



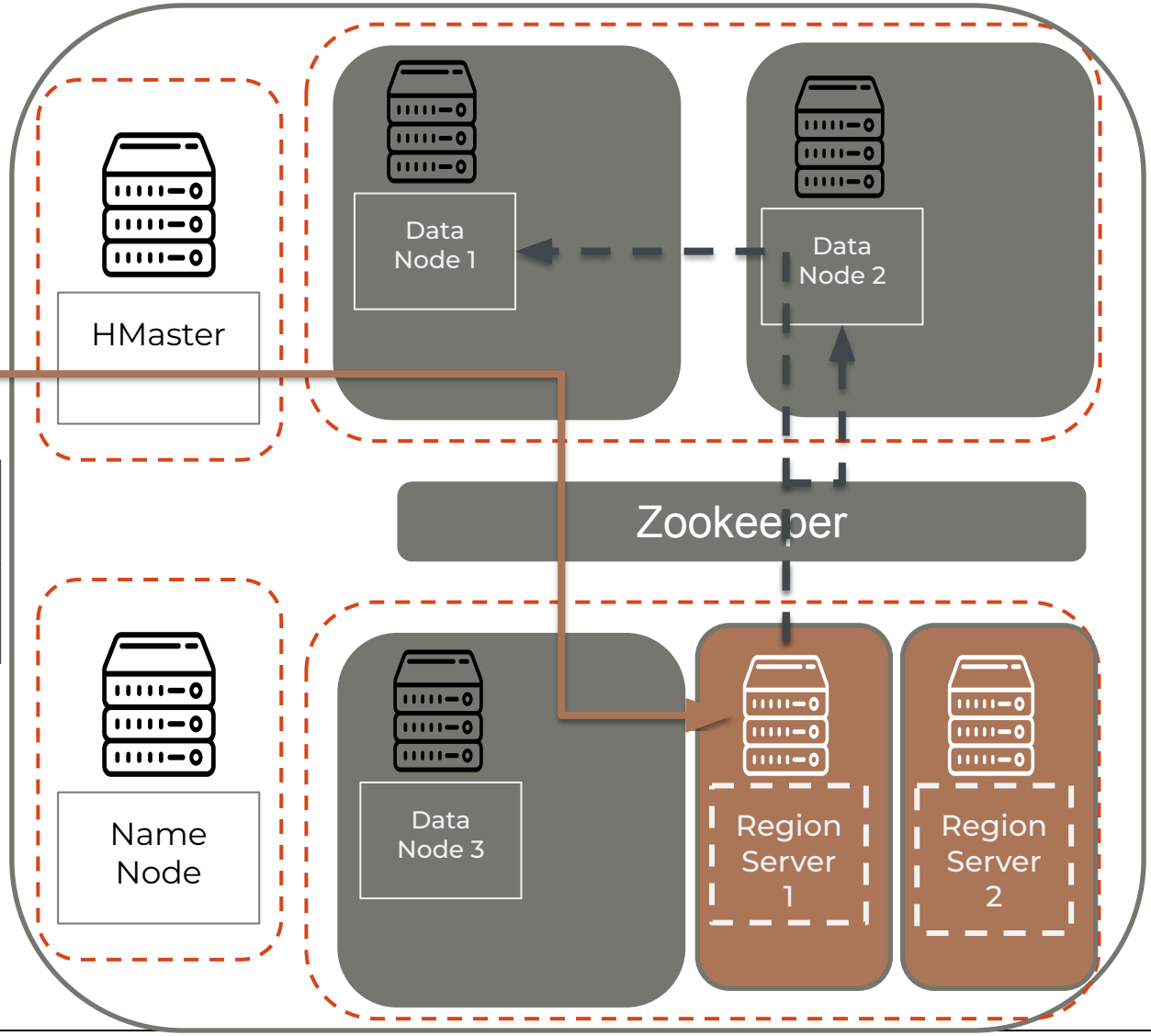
HBase Funcionamiento



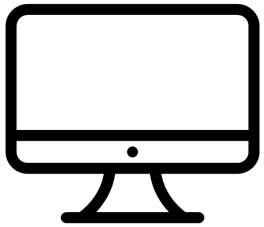
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, 'region_data:valor_region’
‘Colmenar Viejo’
```

Row	Region_data	Tarificacion_data	timestamp



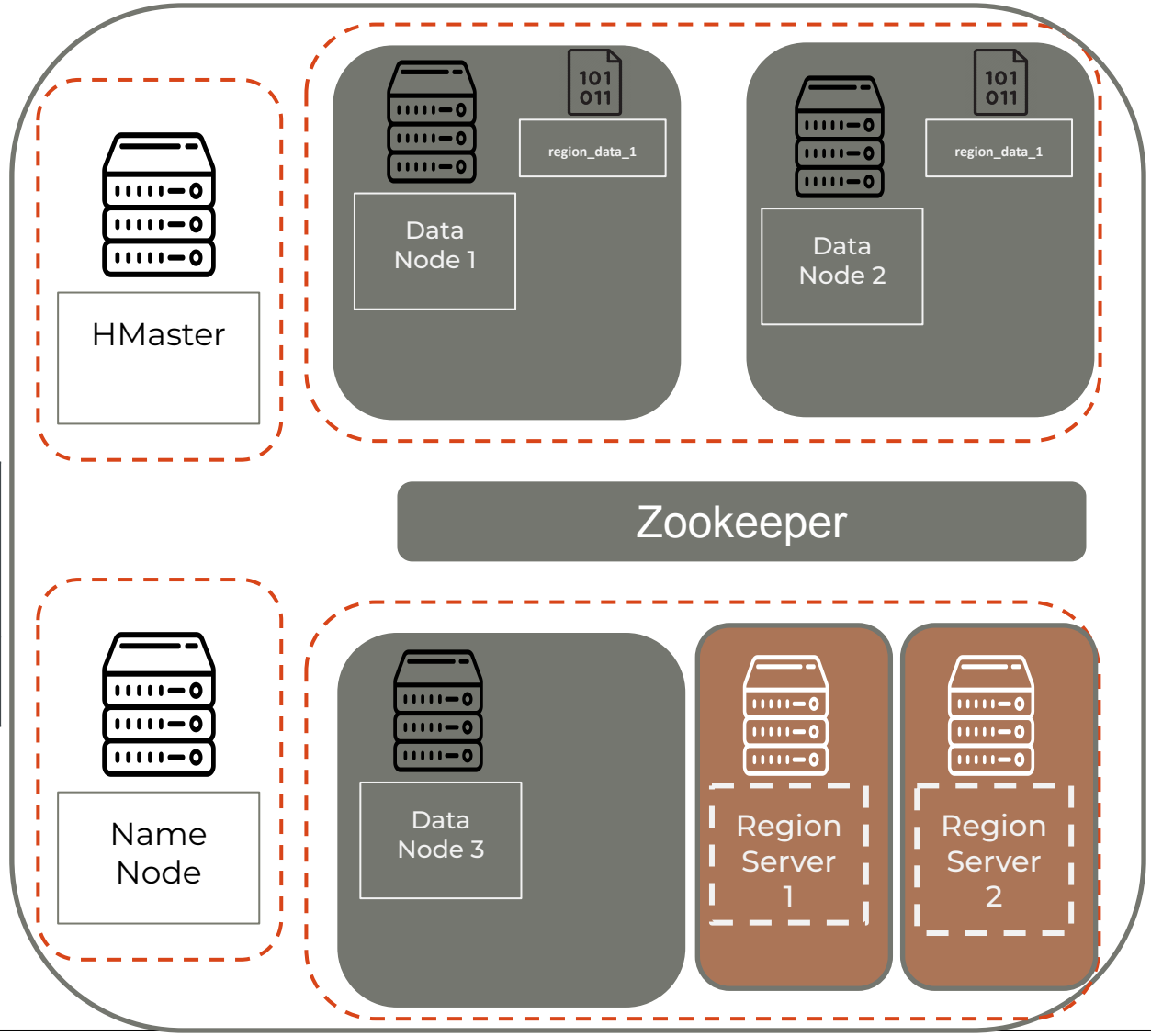
HBase Funcionamiento



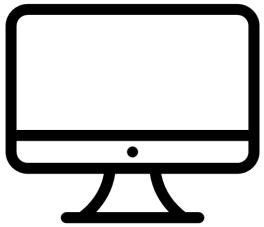
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, 'region_data:valor_region’
‘Colmenar Viejo’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000



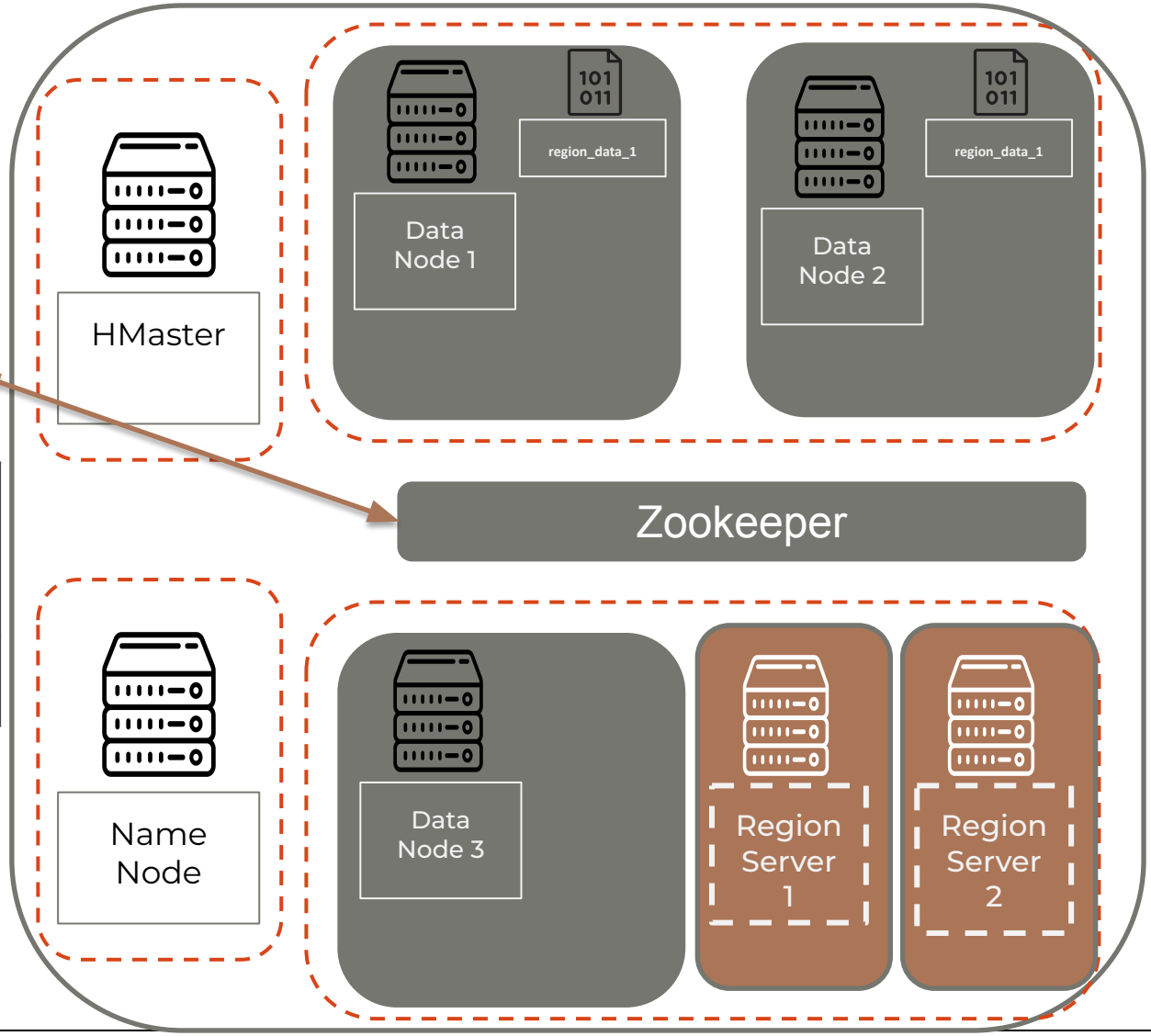
HBase Funcionamiento



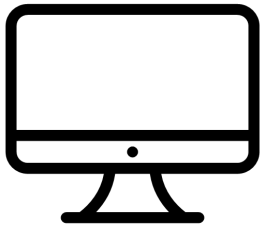
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data’:’valle
10.34’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000



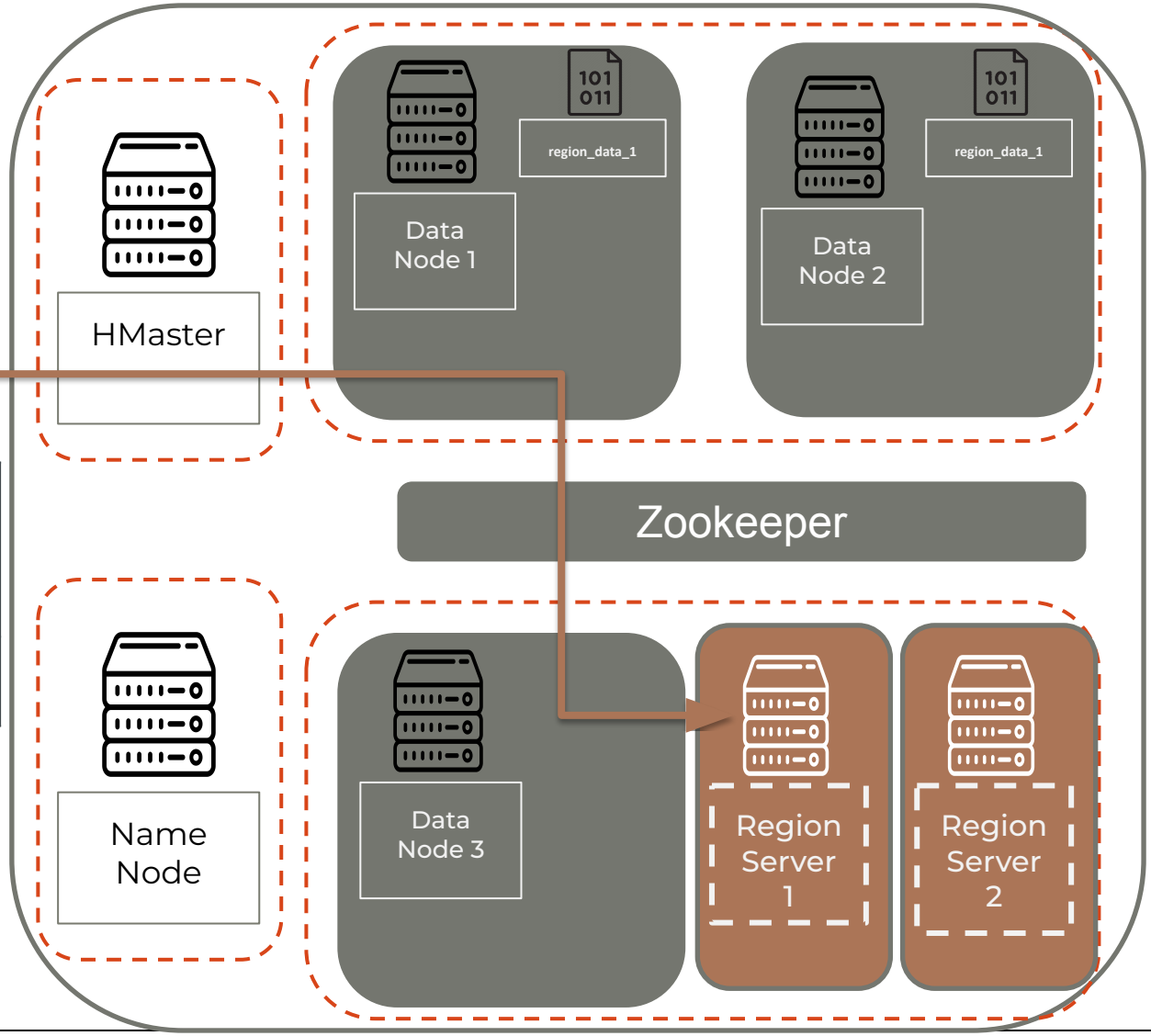
HBase Funcionamiento



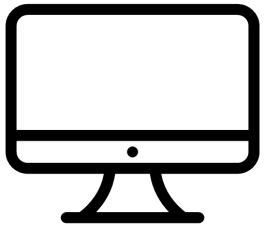
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data’:’valle
10.34’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000



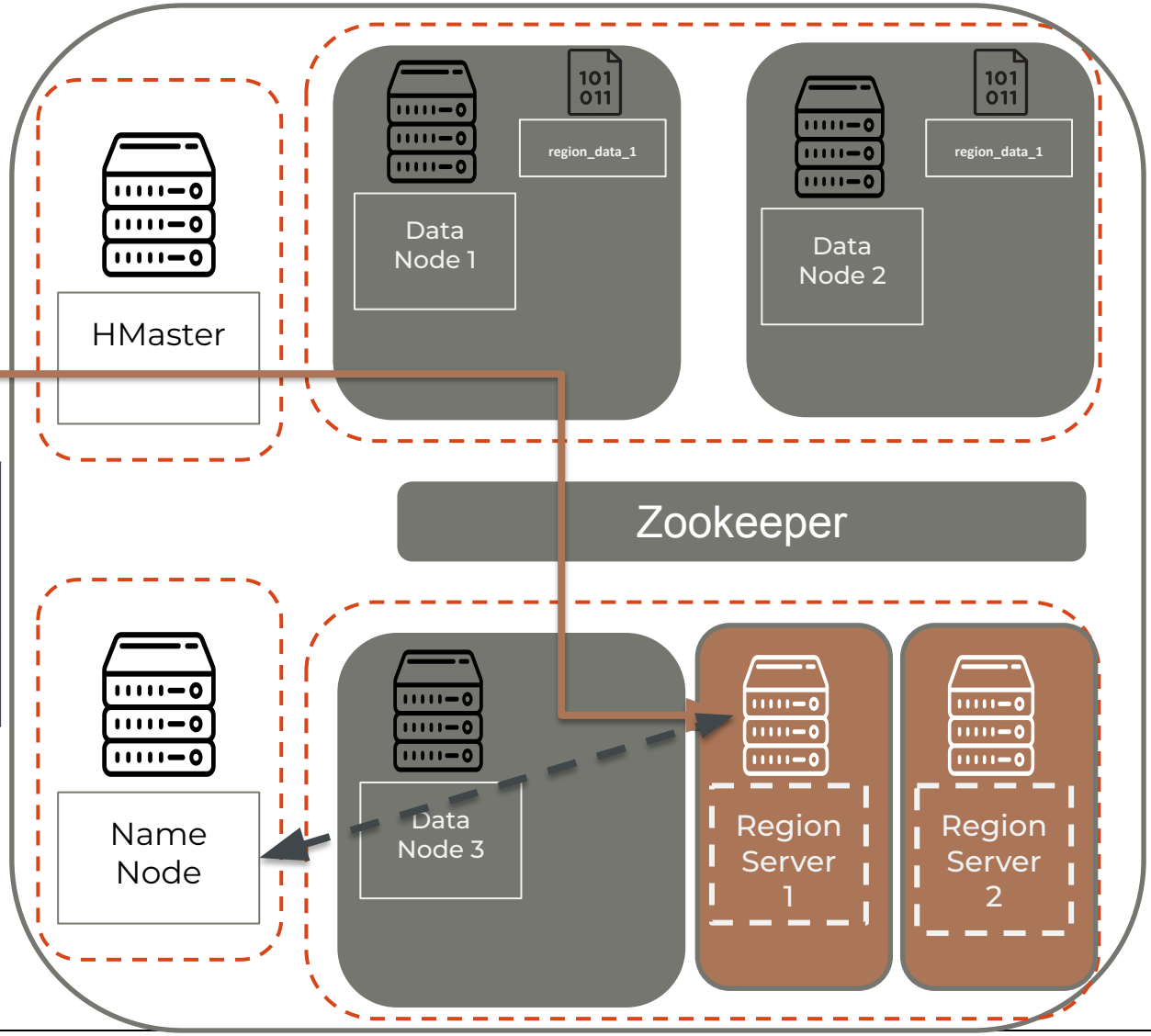
HBase Funcionamiento



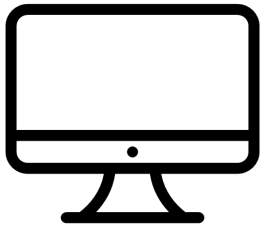
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data’:’valle
10.34’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000



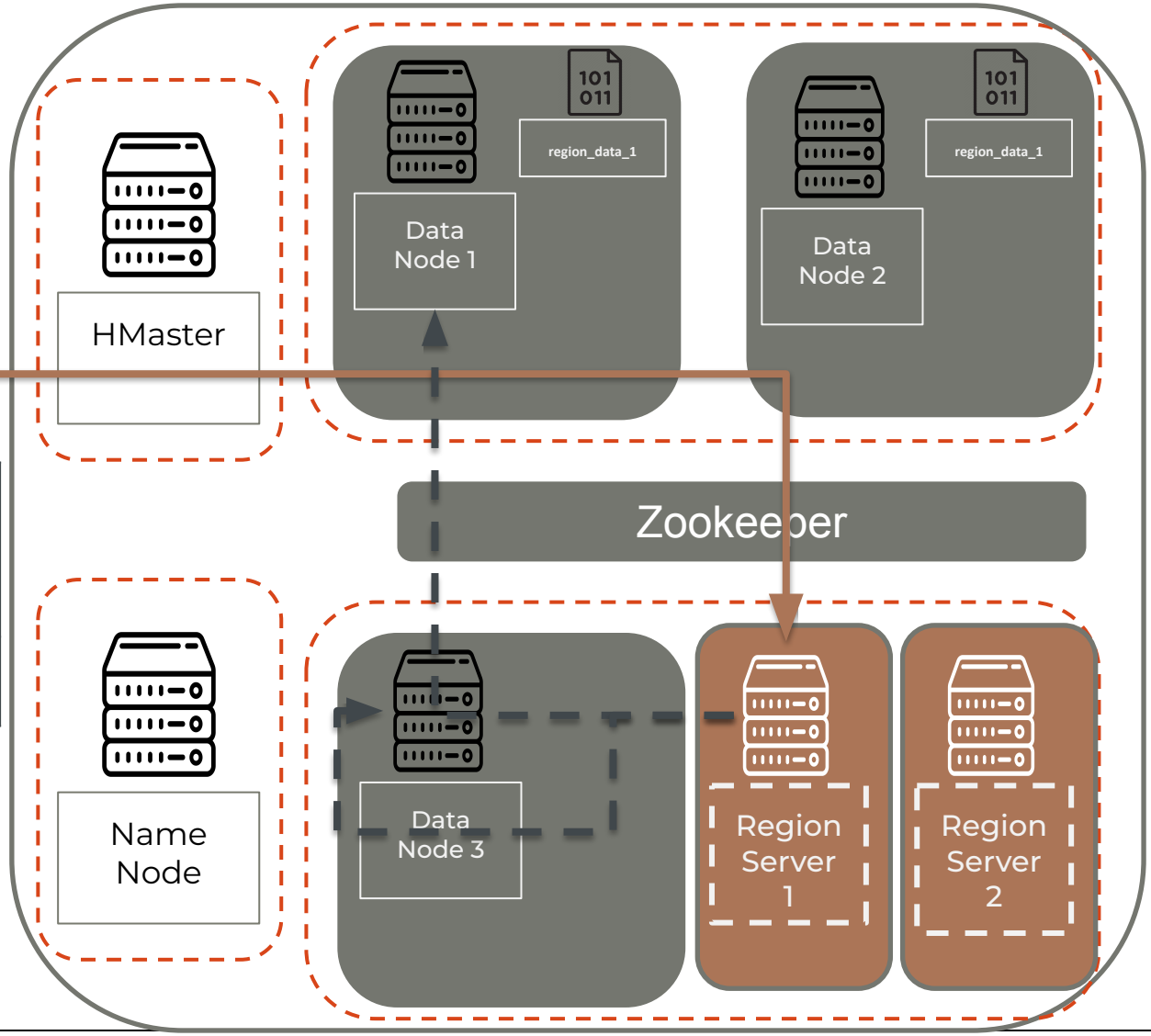
HBase Funcionamiento



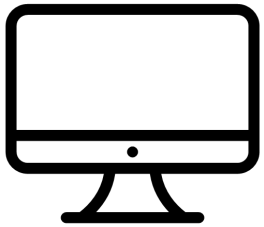
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data’:’valle
10.34’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000



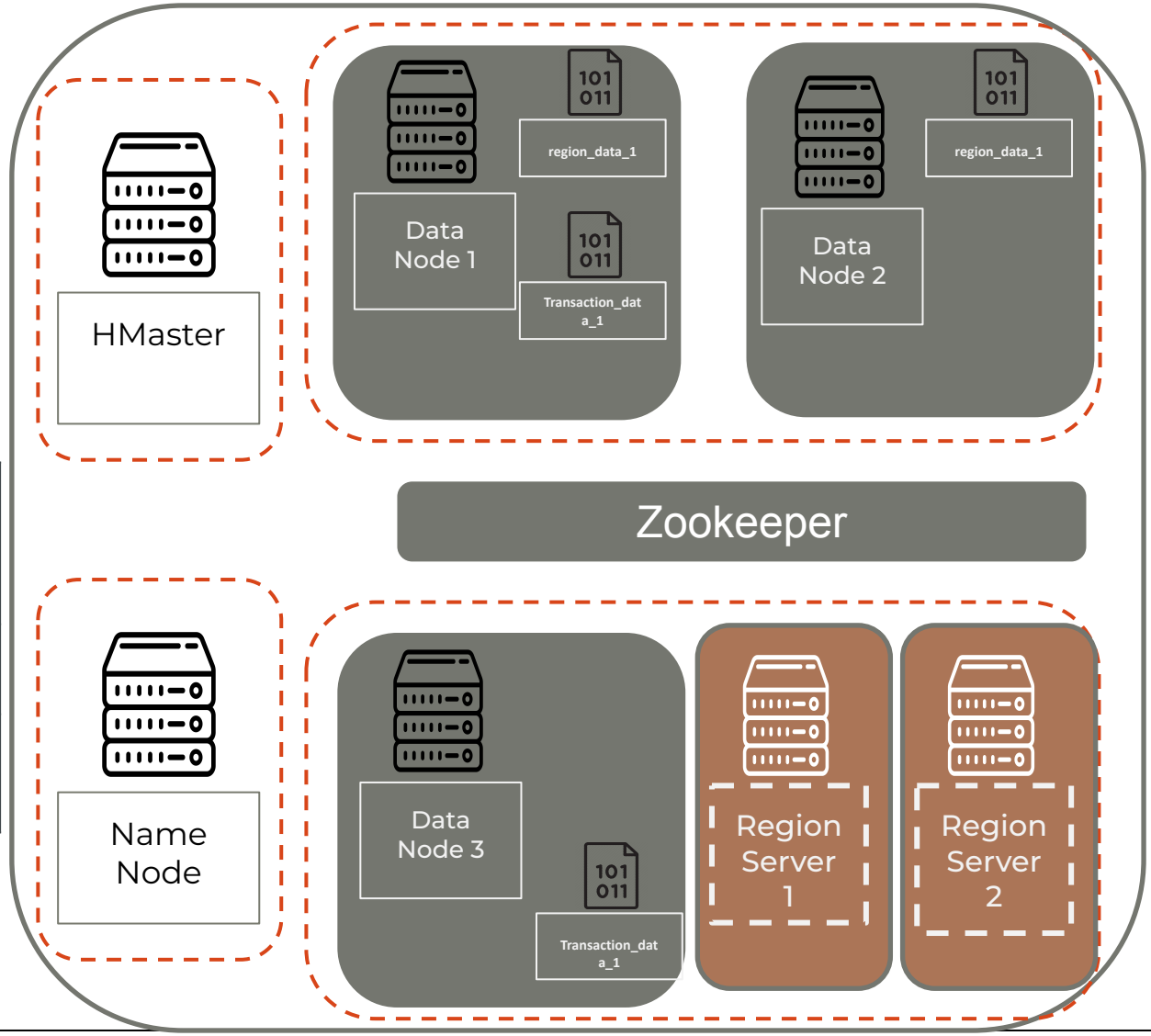
HBase Funcionamiento



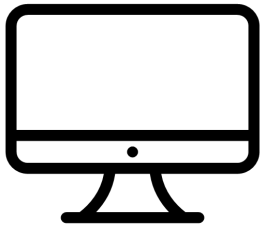
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data’:’valle
10.34’
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000
0001_mun		10.34	1631952400000



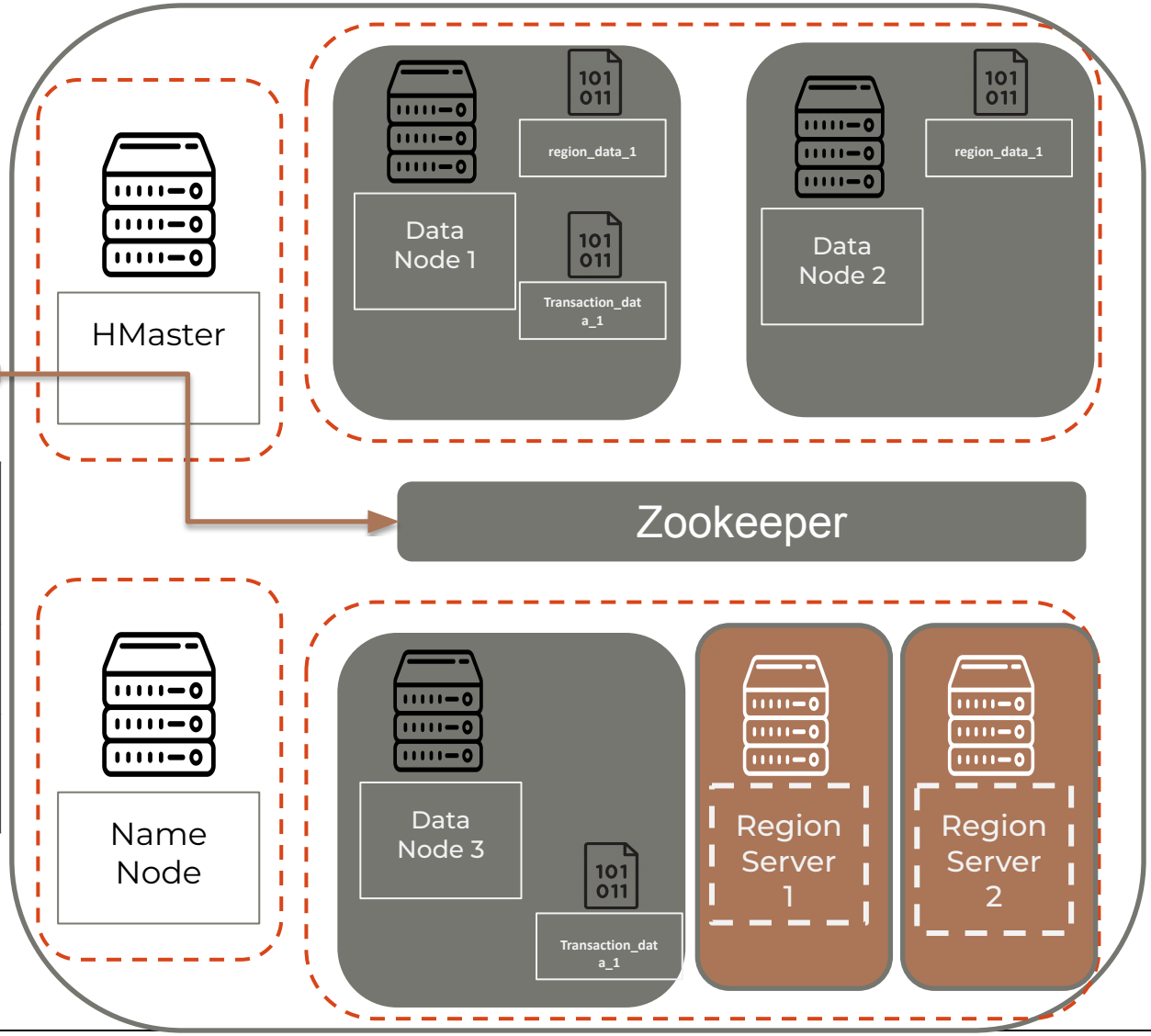
HBase Funcionamiento



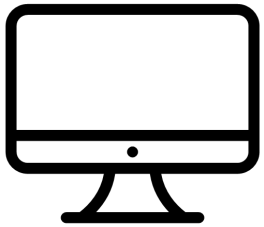
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data:pico‘
‘11.34‘
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000
0001_mun		10.34	1631952400000



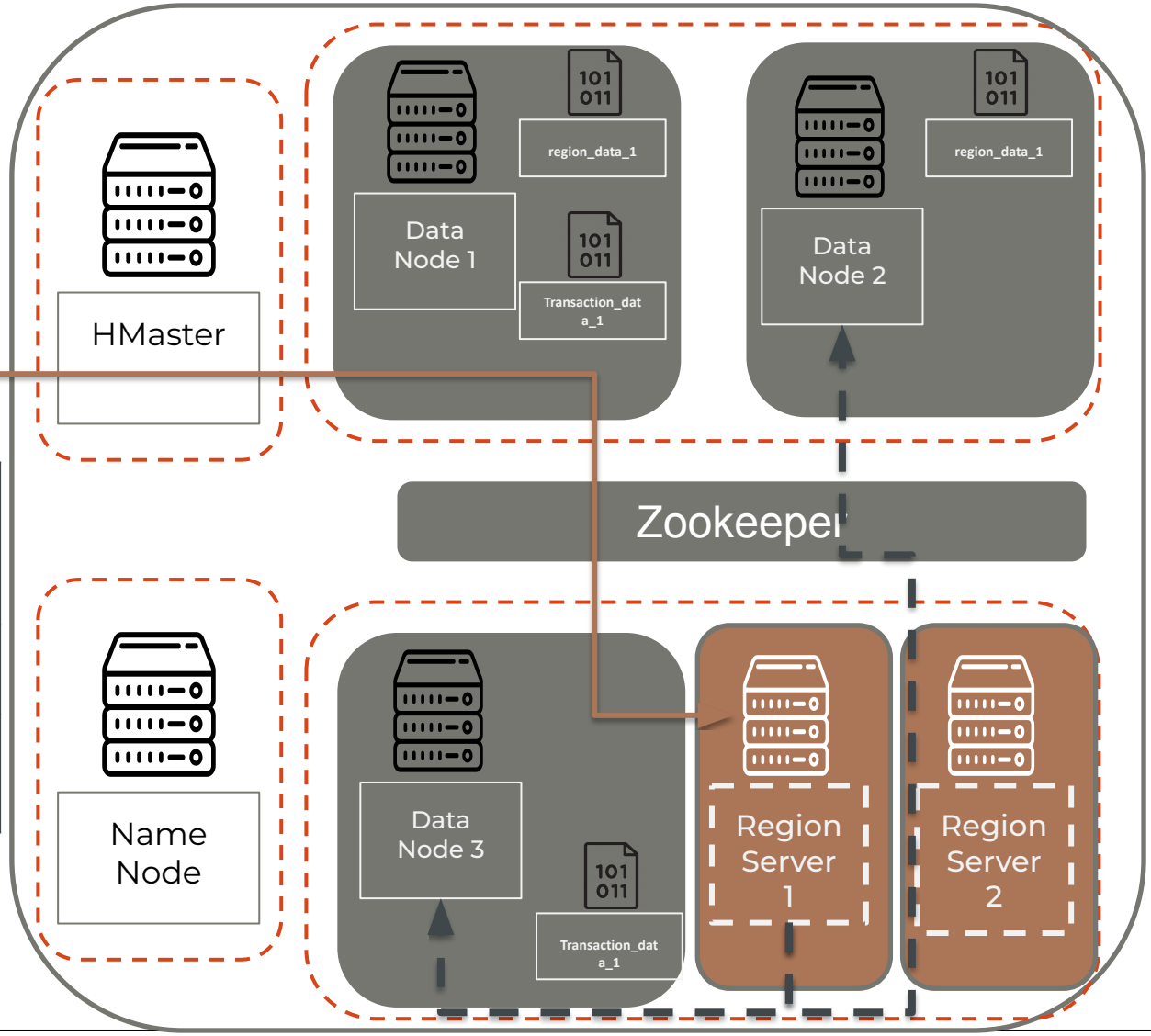
HBase Funcionamiento



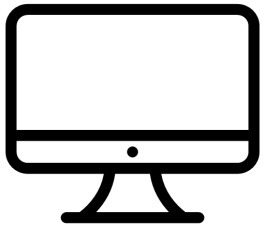
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data:pico‘
‘11.34‘
```

Row	Region_data	Tarificacion_data	timestamp
	valor_region		
0001_mun	Colmenar Viejo		1631952266000
0001_mun		10.34	1631952400000



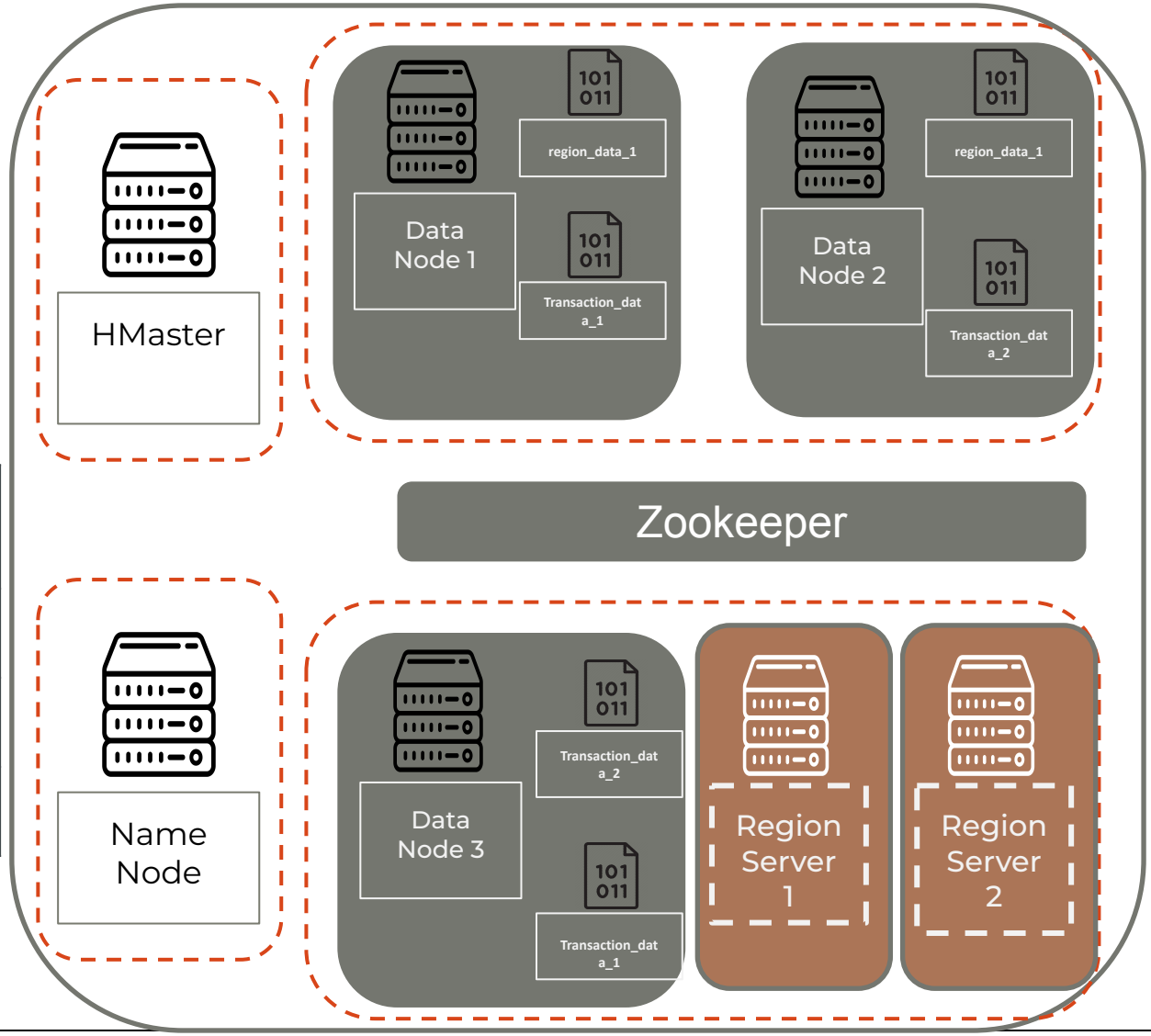
HBase Funcionamiento



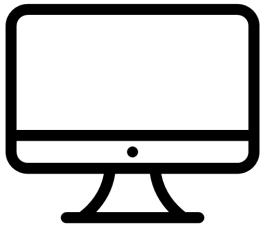
Client-1

```
hbase put
electric_company_cups_regiones
‘0001_mun’, ‘Tarificacion_data:pico‘
‘11.34‘
```

Row	Region_data	Tarificacion_data		timestamp
	valor_region	valle	pico	
0001_mun	Colmenar Viejo			1631952266000
0001_mun		10.34		1631952400000
0001_mun			11.34	1631952500000



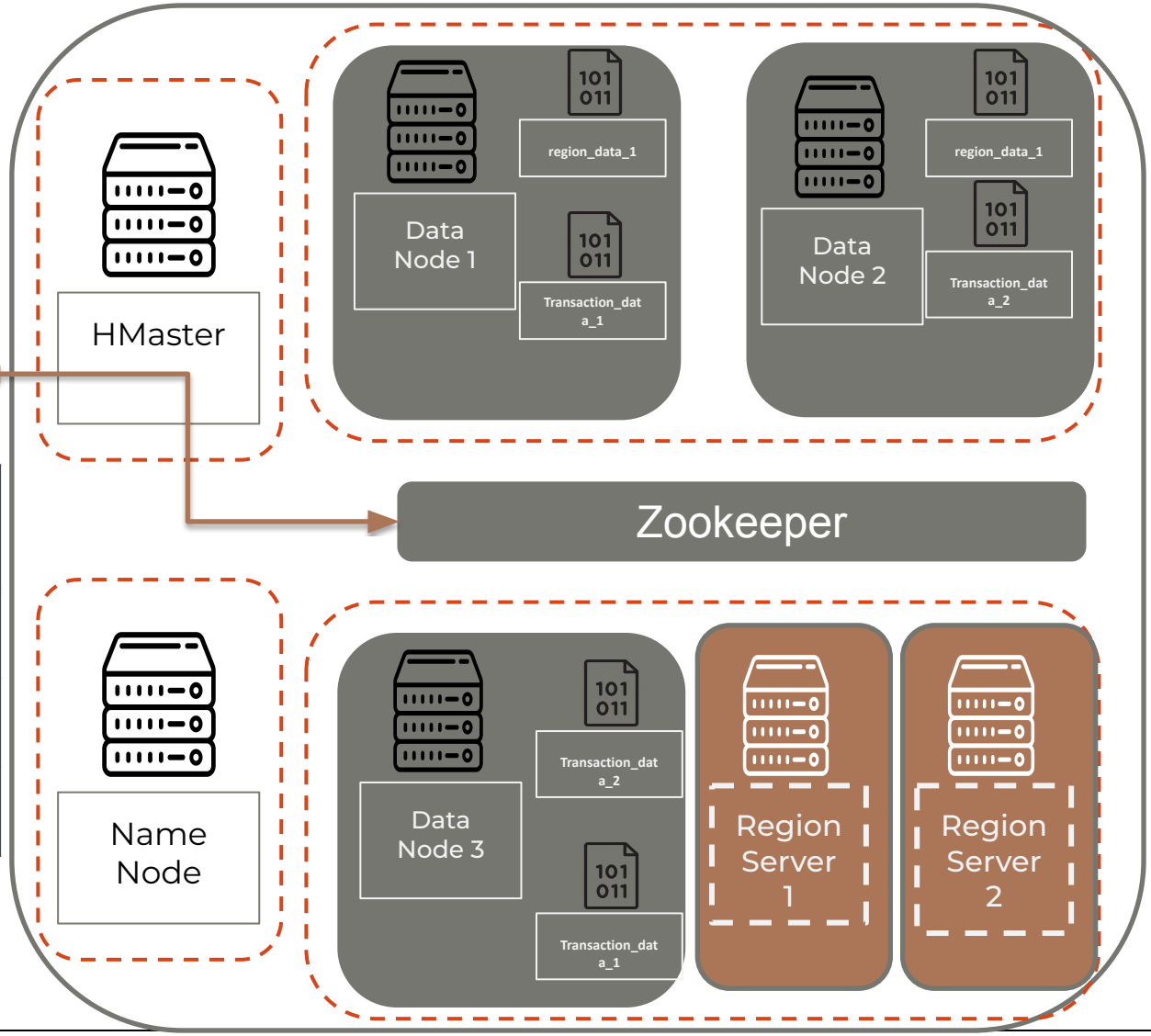
HBase Funcionamiento



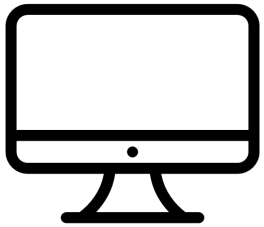
Client-1

```
hbase put
electric_company_cups_regiones
‘0002_mun’, region_data:valor_region
‘Alcalá’
```

Row	Region_data	Tarificacion_data		timestamp
	valor_region	valle	pico	
0001_mun	Colmenar Viejo			1631952266000
0001_mun		10.34		1631952400000
0001_mun			11.34	1631952500000



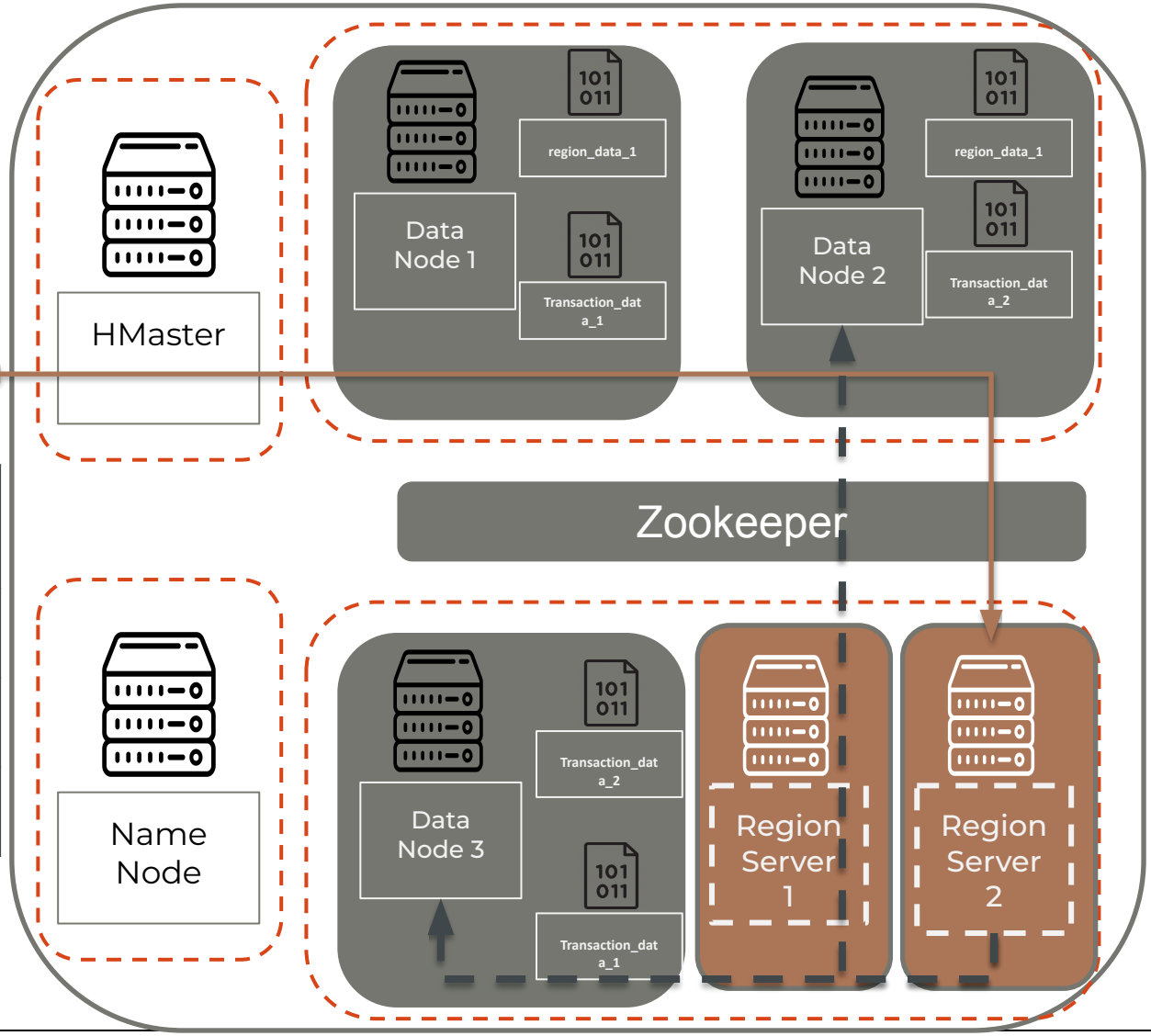
HBase Funcionamiento



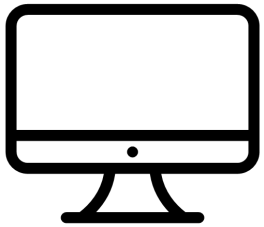
Client-1

```
hbase put
electric_company_cups_regiones
‘0002_mun’, region_data:valor_region
‘Alcalá’
```

Row	Region_data	Tarificacion_data		timestamp
	valor_region	valle	pico	
0001_mun	Colmenar Viejo			1631952266000
0001_mun		10.34		1631952400000
0001_mun			11.34	1631952500000



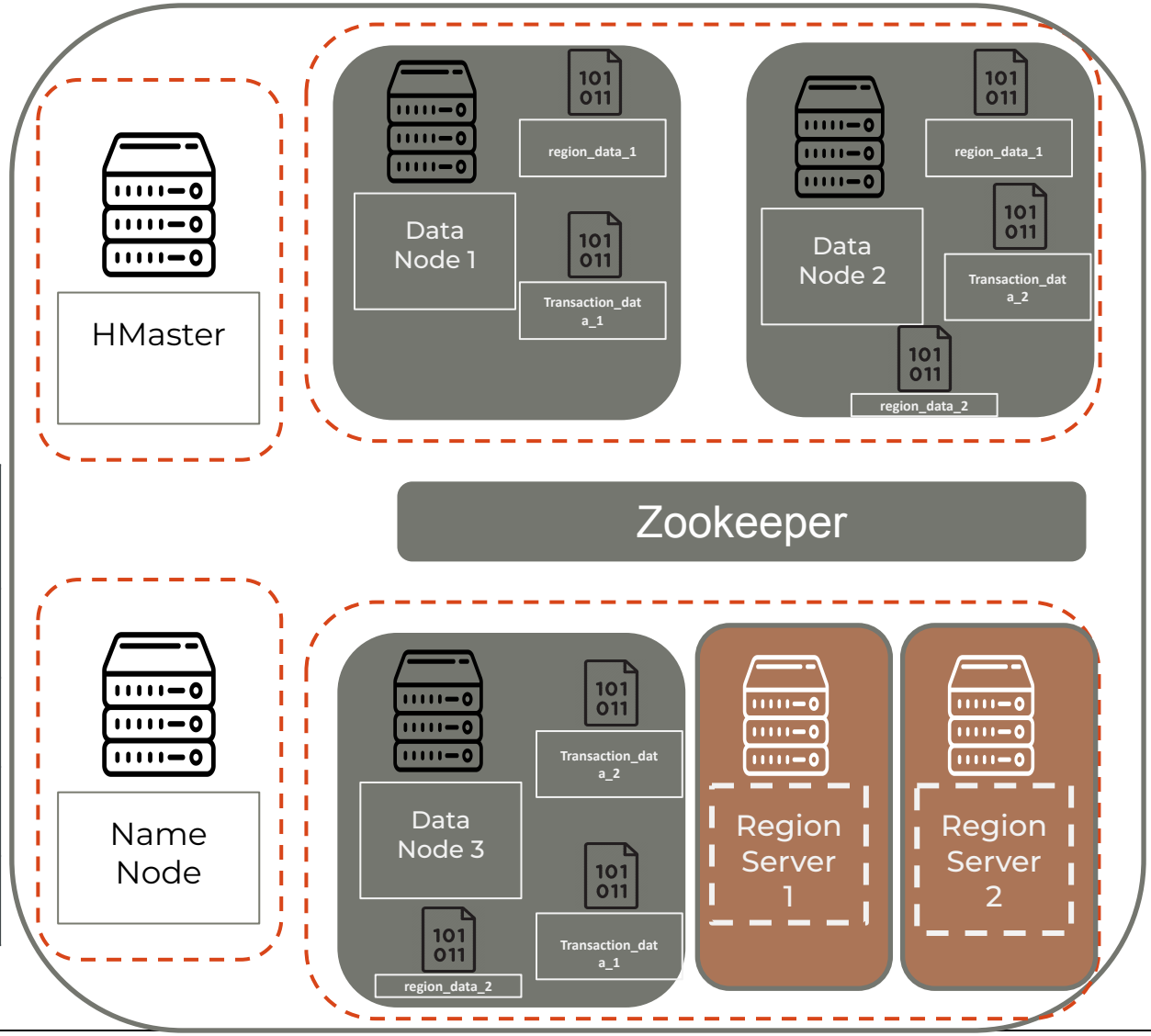
HBase Funcionamiento



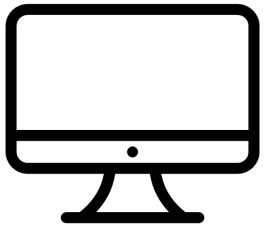
Client-1

```
hbase put
electric_company_cups_regiones
‘0002_mun’, region_data:valor_region
‘Alcalá’
```

Row	Region_data	Tarificacion_data		timestamp
	valor_region	valle	pico	
0001_mun	Colmenar Viejo			1631952266000
0001_mun		10.34		1631952400000
0001_mun			11.34	1631952500000
0002_mun	Acalá			1631952600000



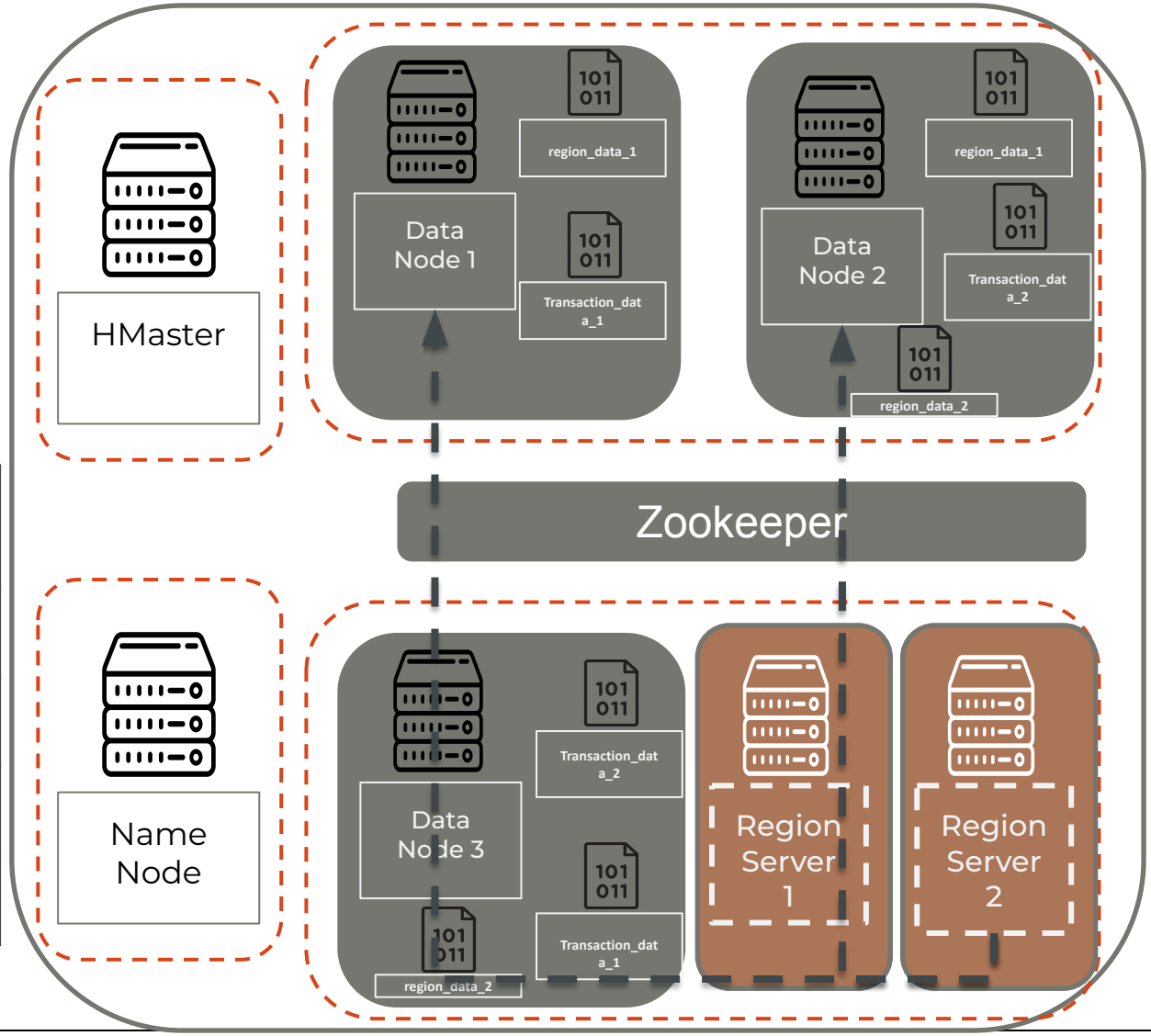
HBase Funcionamiento



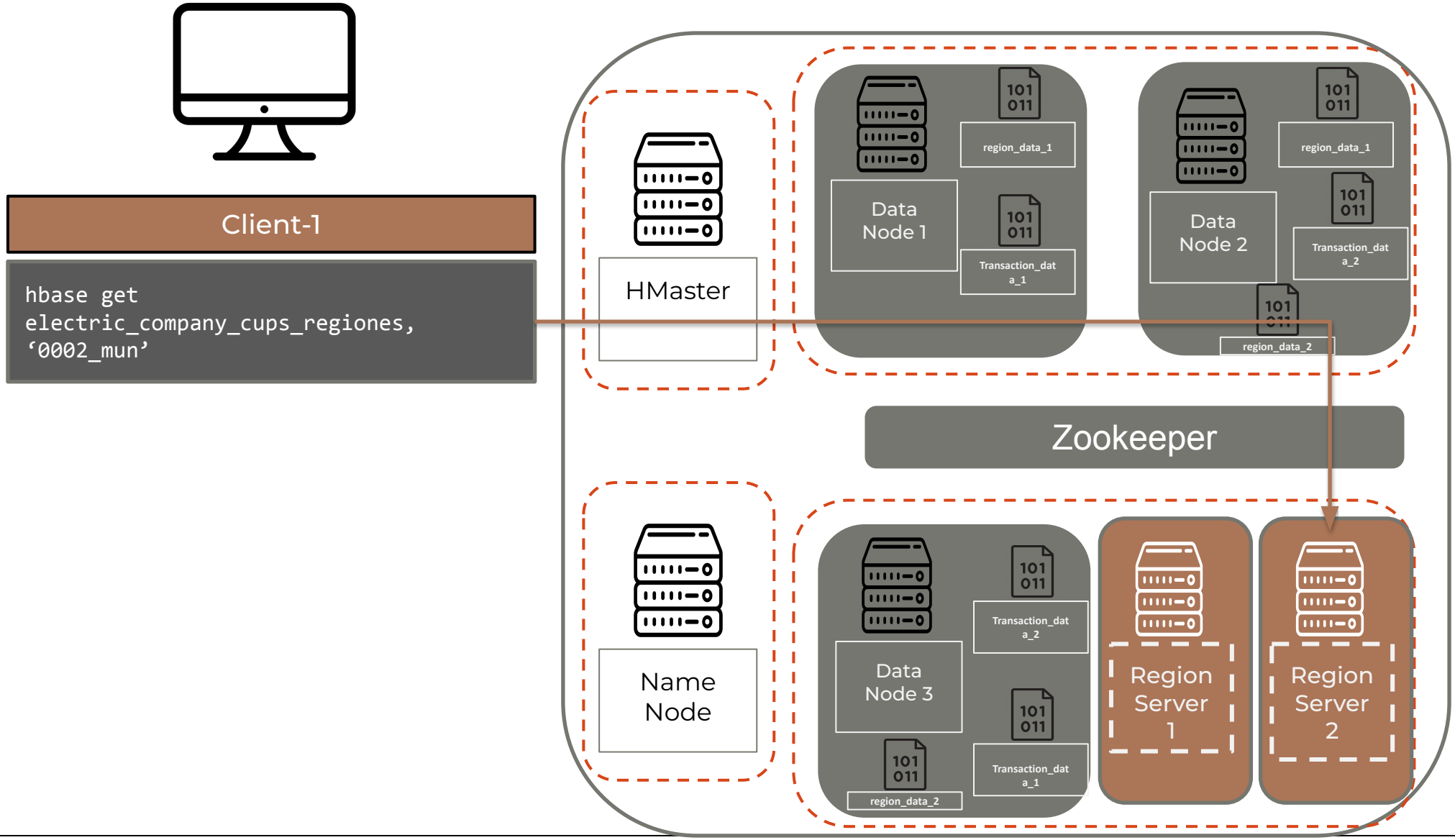
Client-1

```
hbase put
electric_company_cups_regiones
‘0002_mun’, region_data:valor_region
‘Alcalá’
```

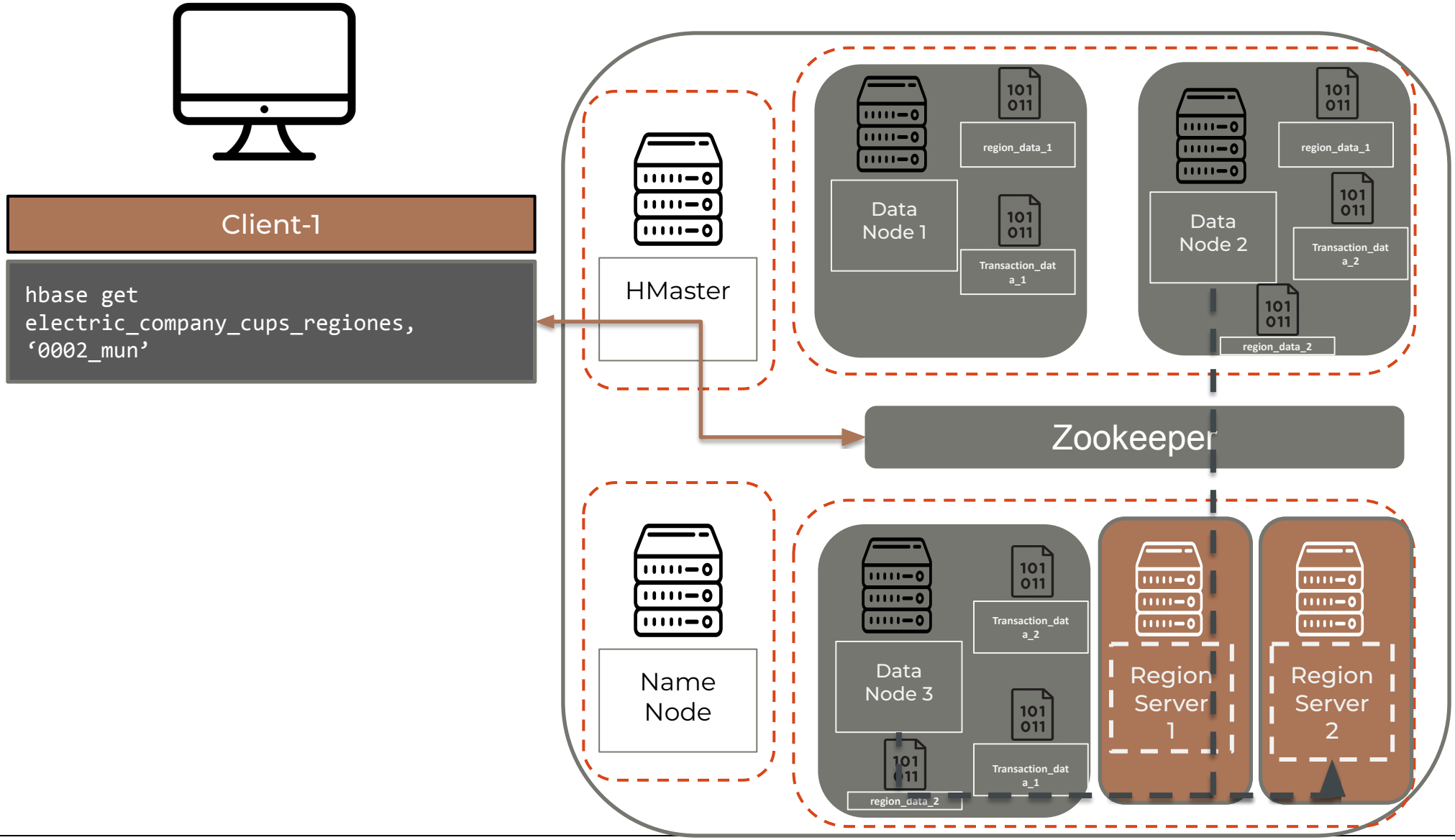
Row	Region_data	Tarificacion_data		timestamp
	valor_region	valle	pico	
0001_mun	Colmenar Viejo			1631952266000
0001_mun		10.34		1631952400000
0001_mun			11.34	1631952500000
0002_mun	Acalá			1631952600000



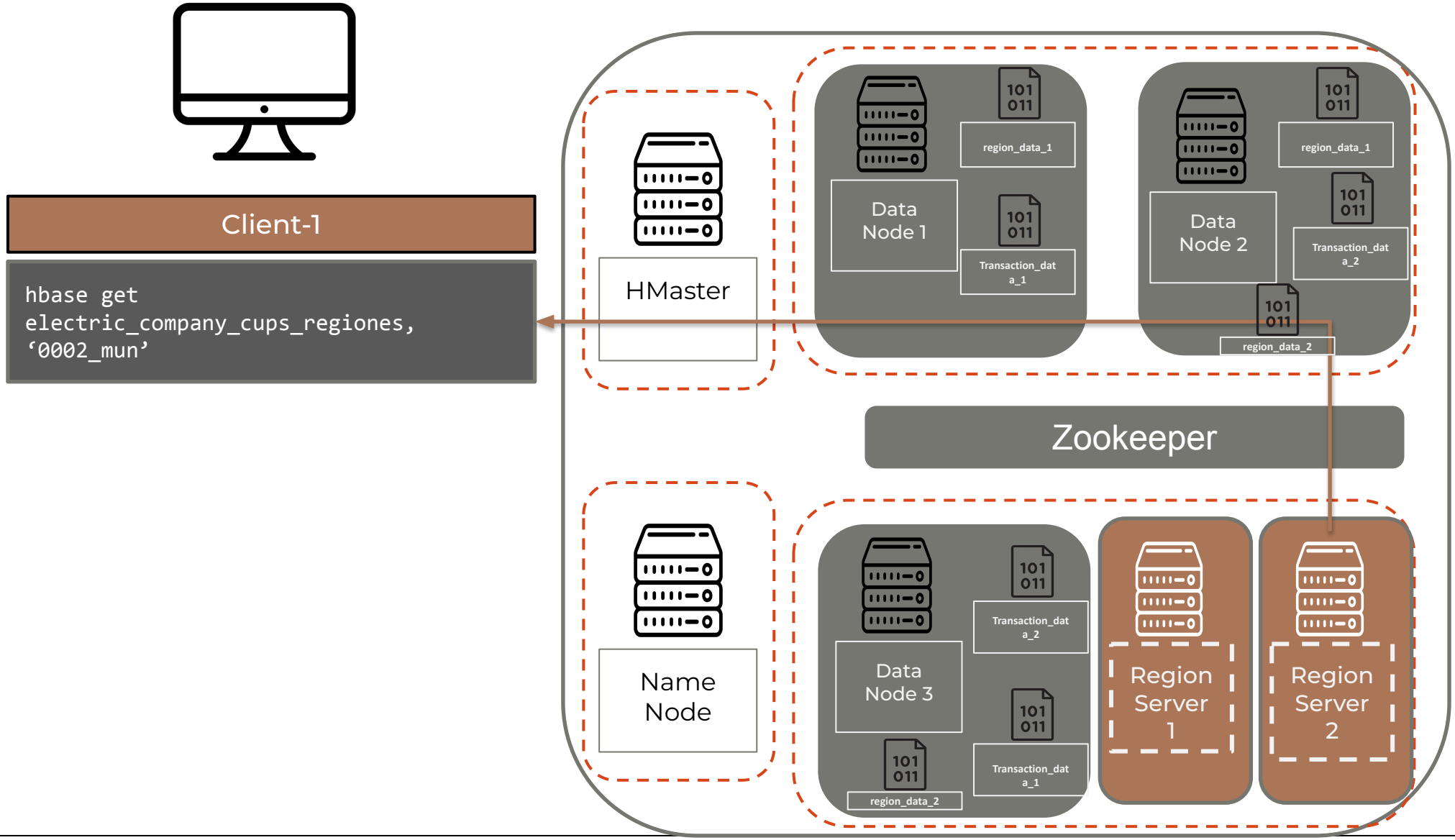
HBase Funcionamiento



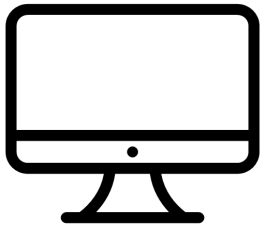
HBase Funcionamiento



HBase Funcionamiento

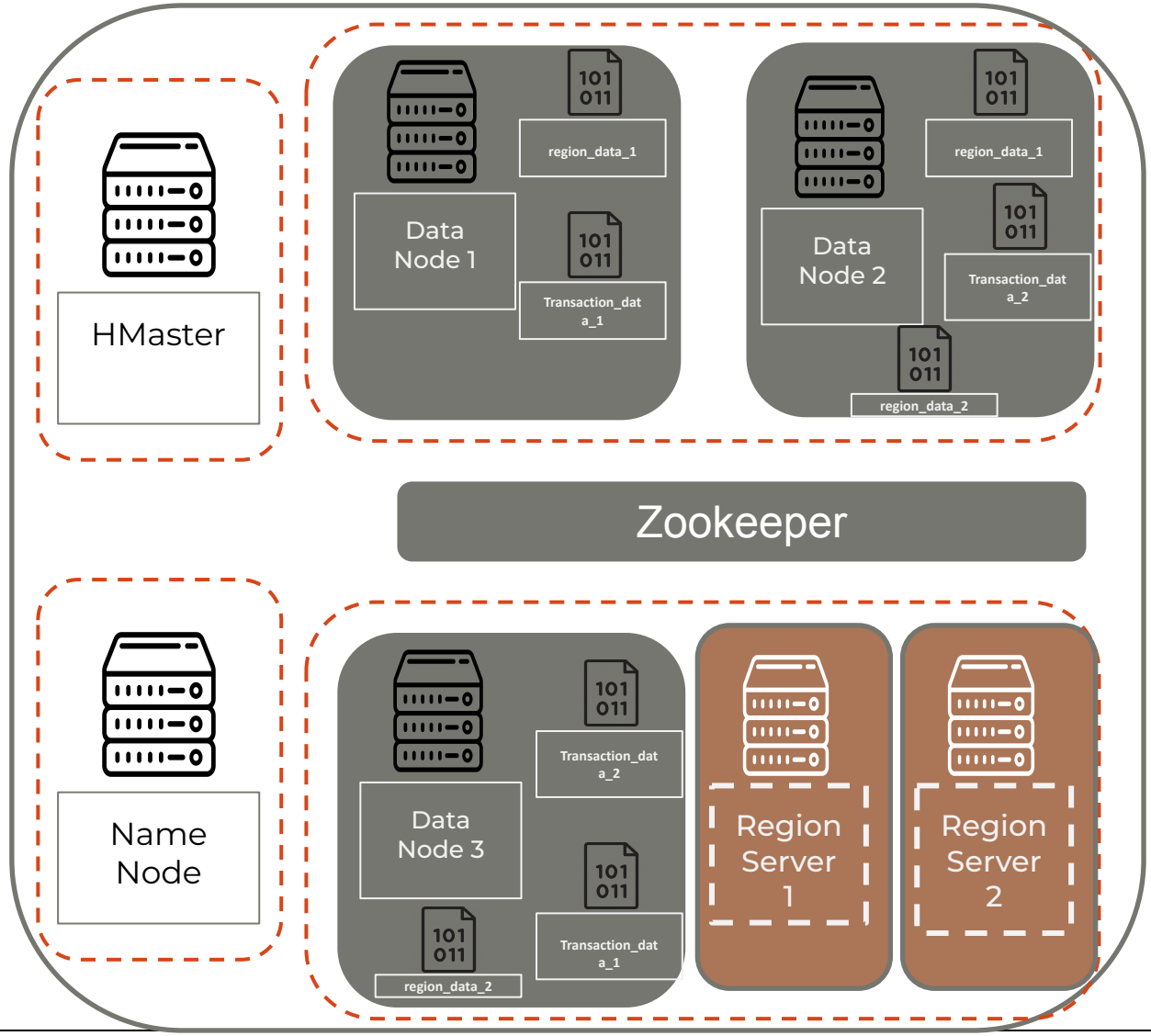


HBase Funcionamiento

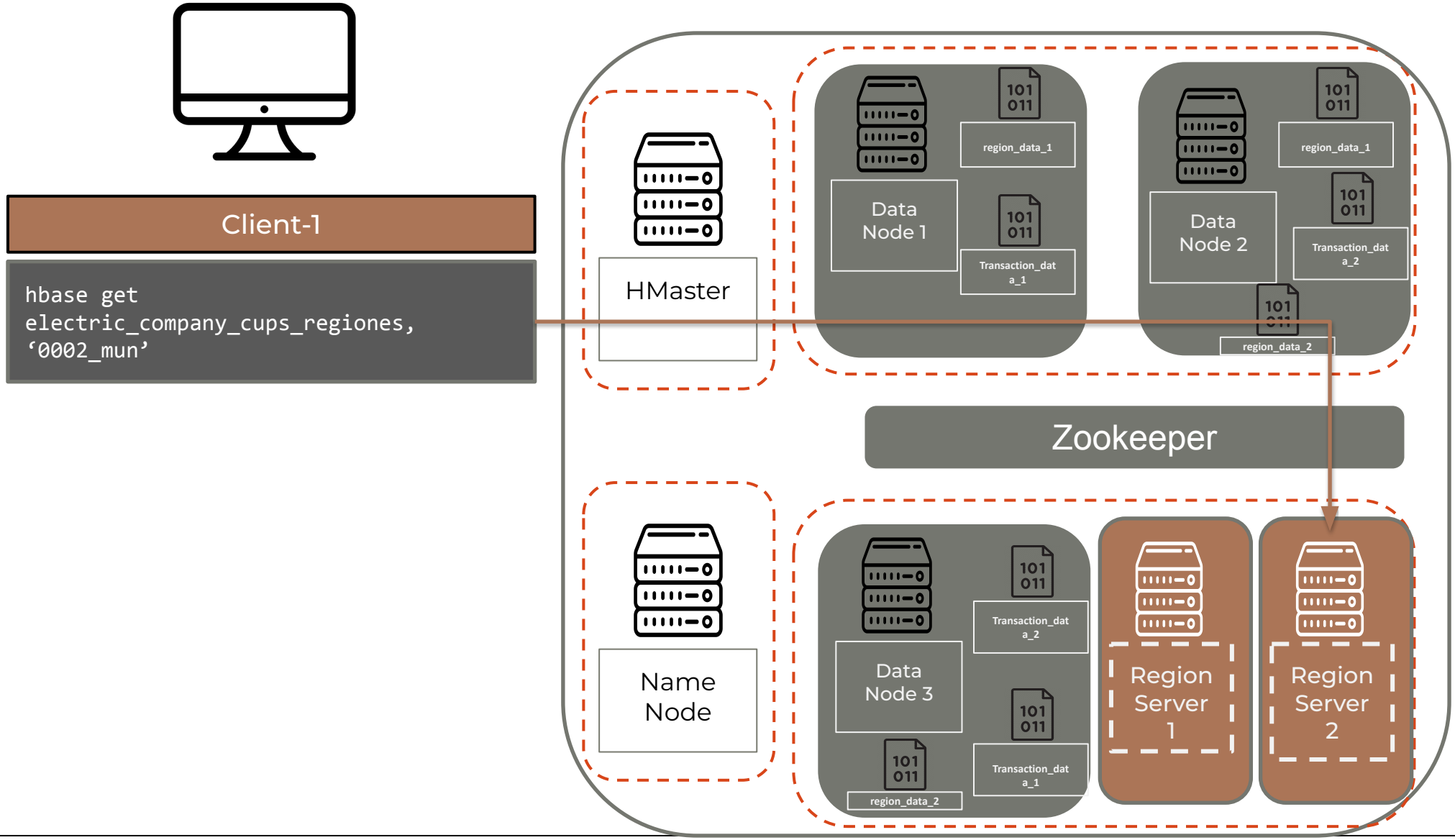


Client-1

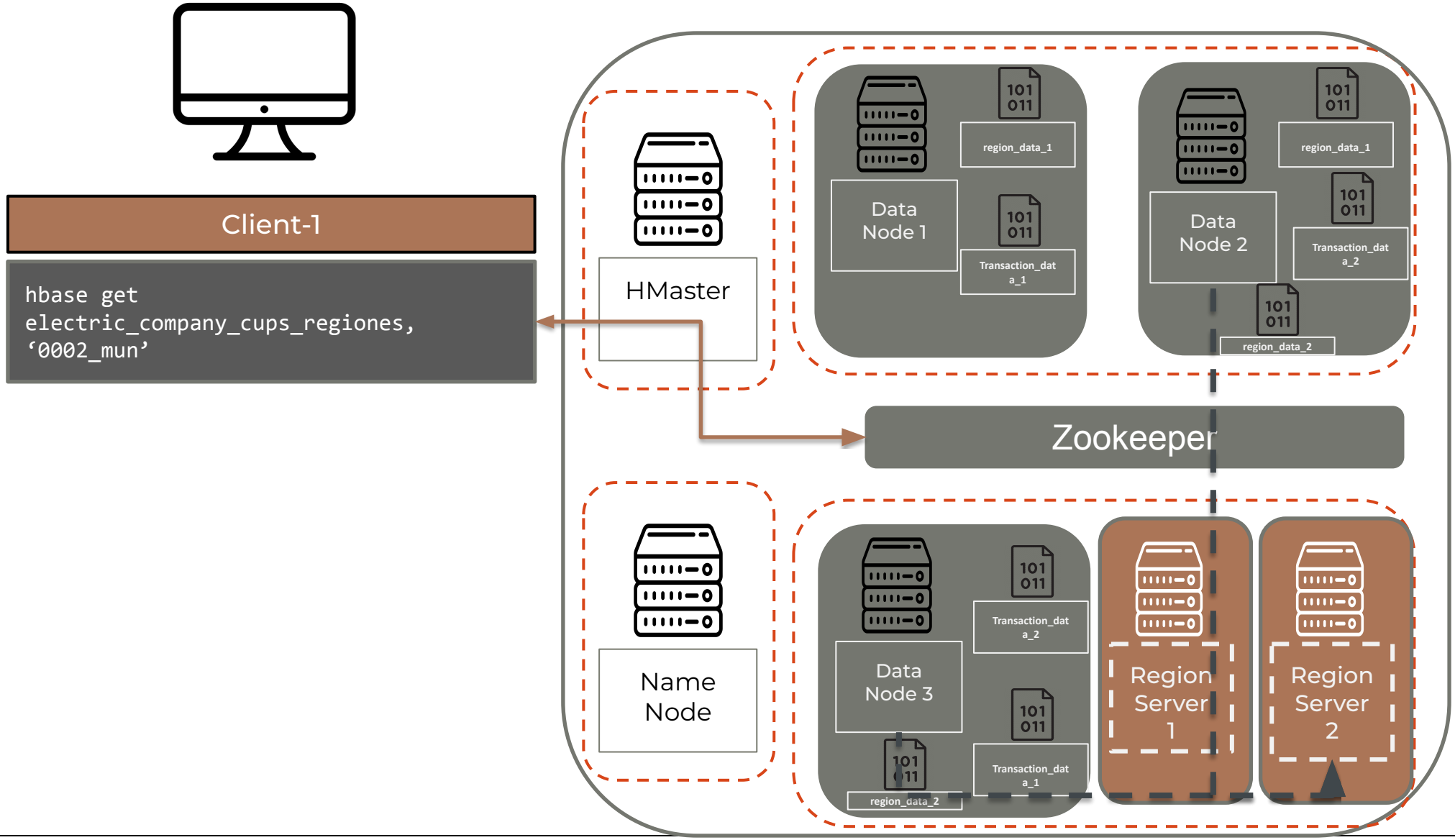
```
hbase get
electric_company_cups_regiones,
‘0002_mun’

COLUMN                          CELL
region_data : valor_region
timestamp=1631952266000,
value=’Colmenar Viejo’
Tarificacion_data:pico
 timestamp=1631952500000, value=11.34
Tarificacion_data:valle
 timestamp=1631954000000, value=100.34
```



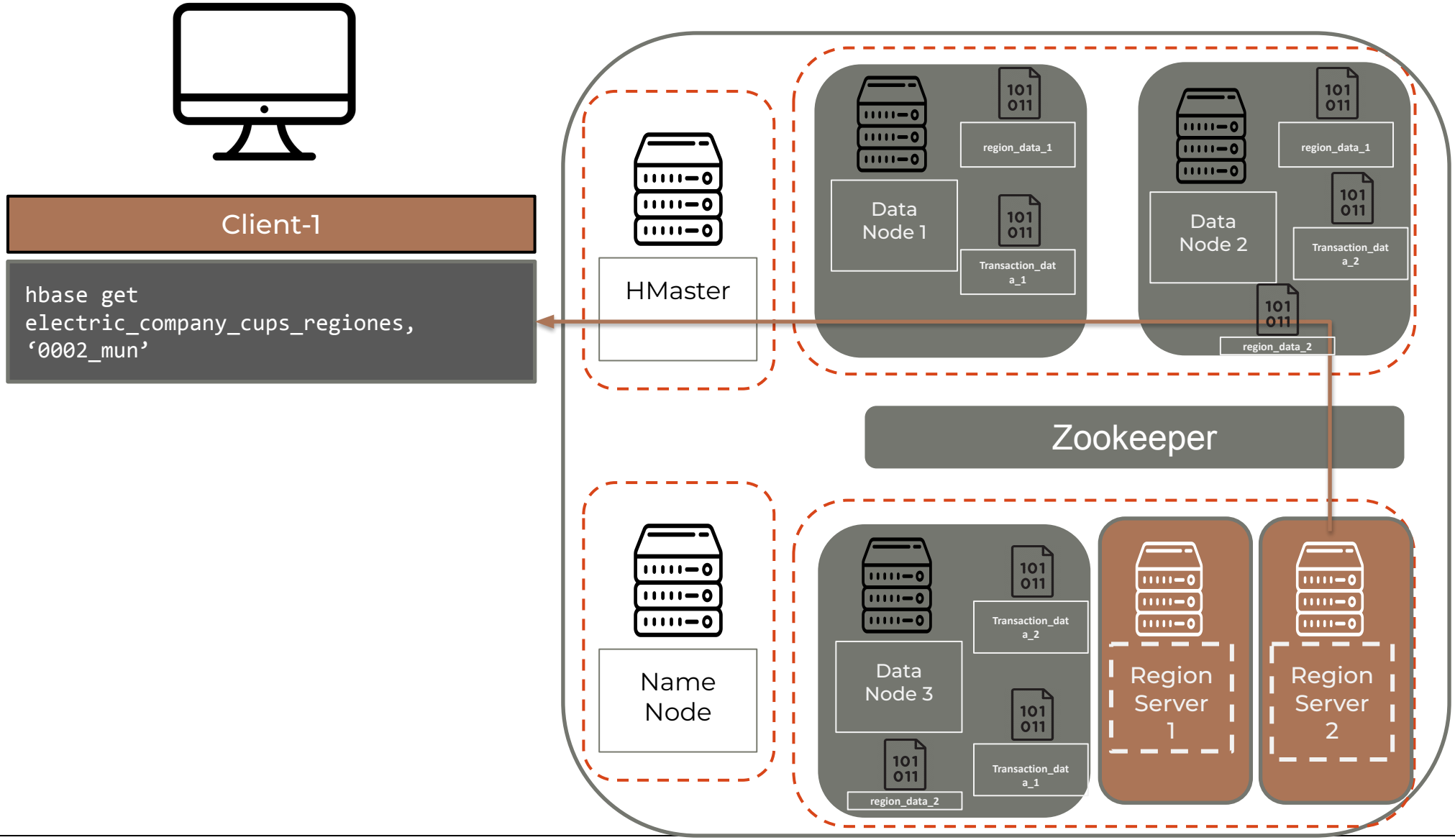
HBase Funcionamiento



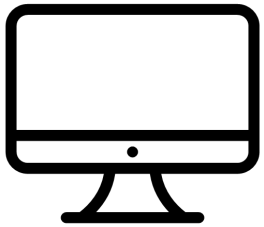
HBase Funcionamiento



HBase Funcionamiento

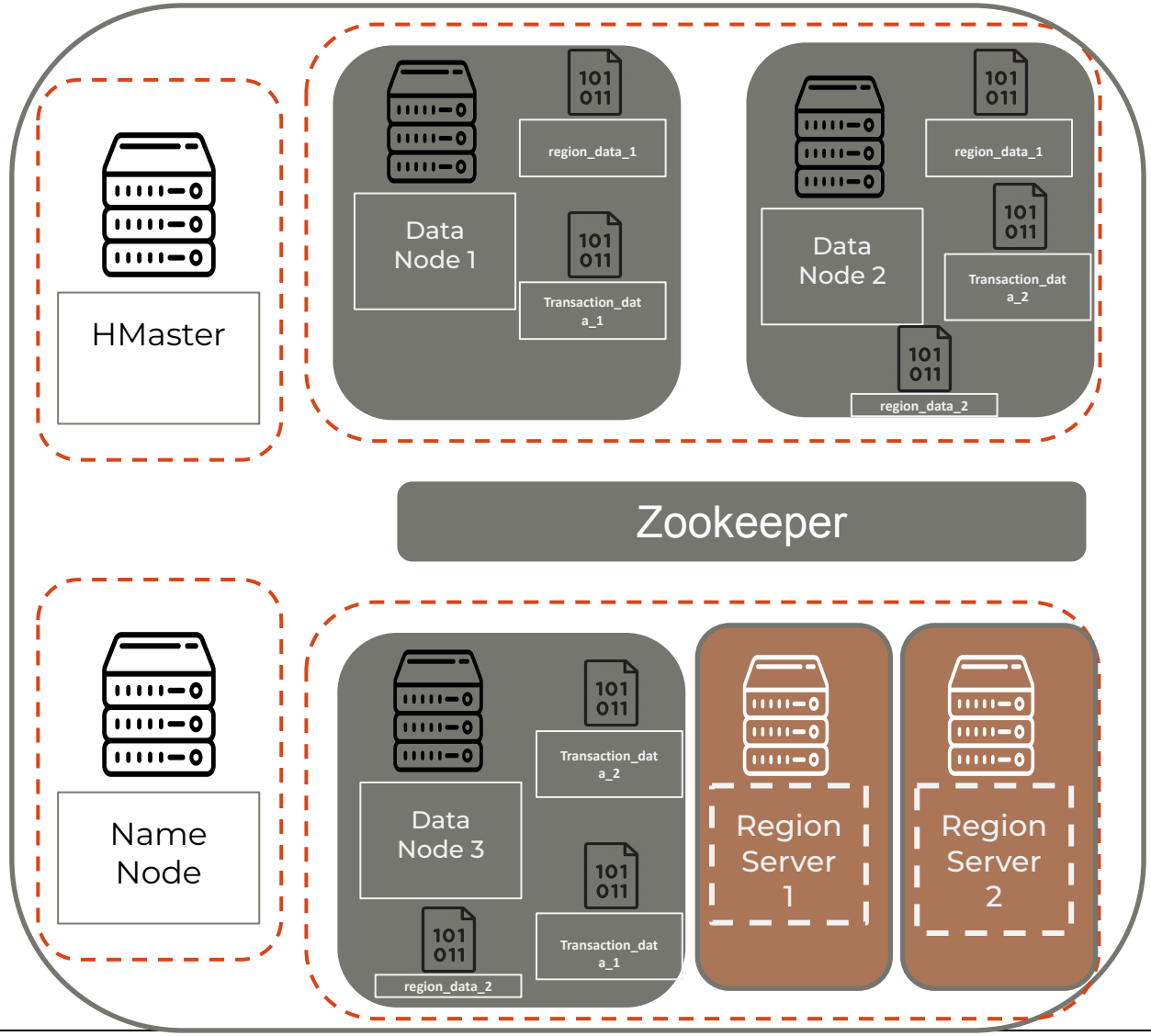


HBase Funcionamiento

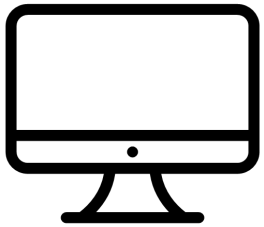


Client-1

```
hbase scan
electric_company_cups_regiones
```

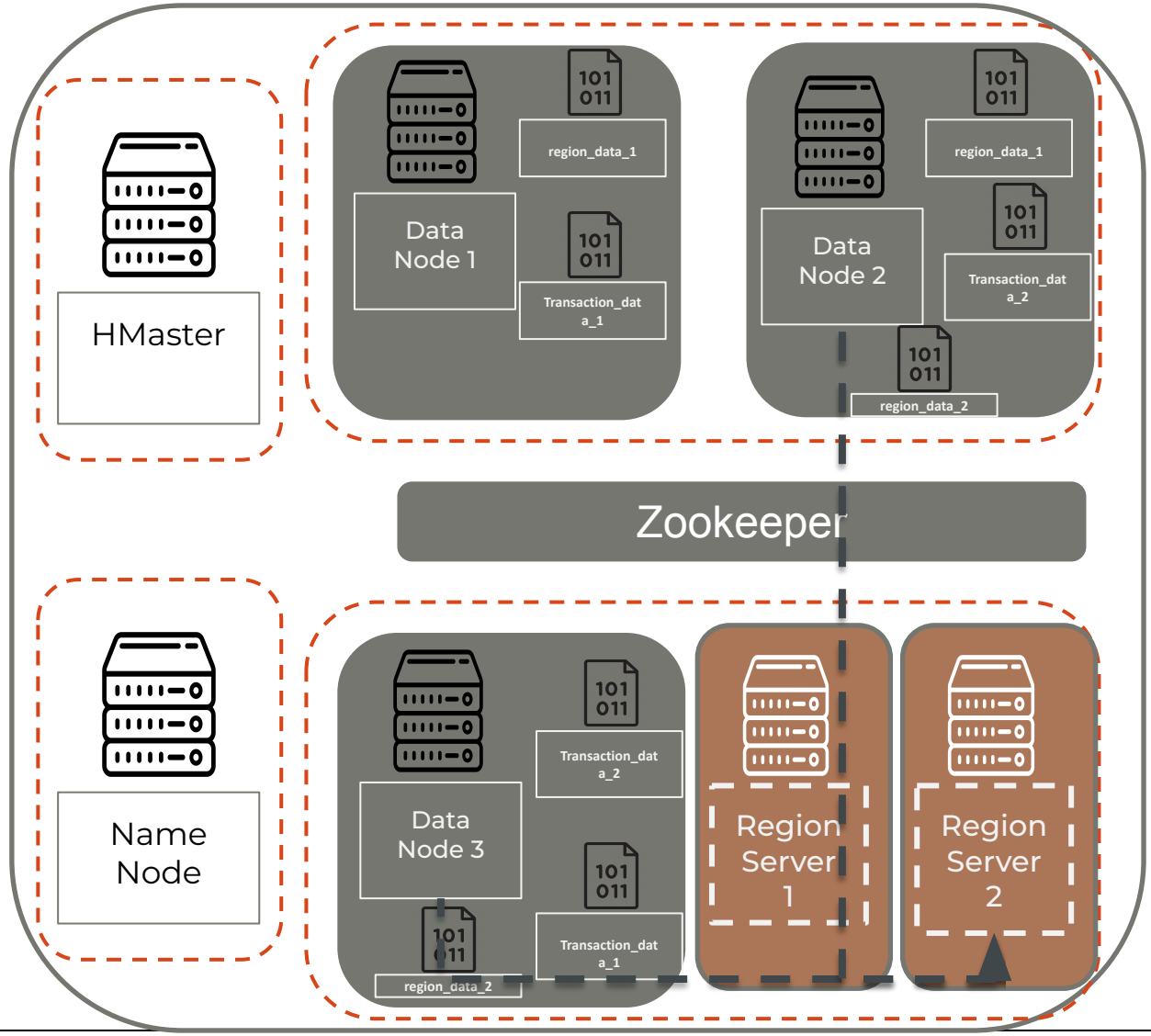


HBase Funcionamiento

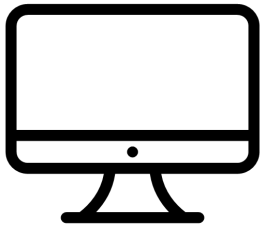


Client-1

```
hbase scan
electric_company_cups_regiones
```

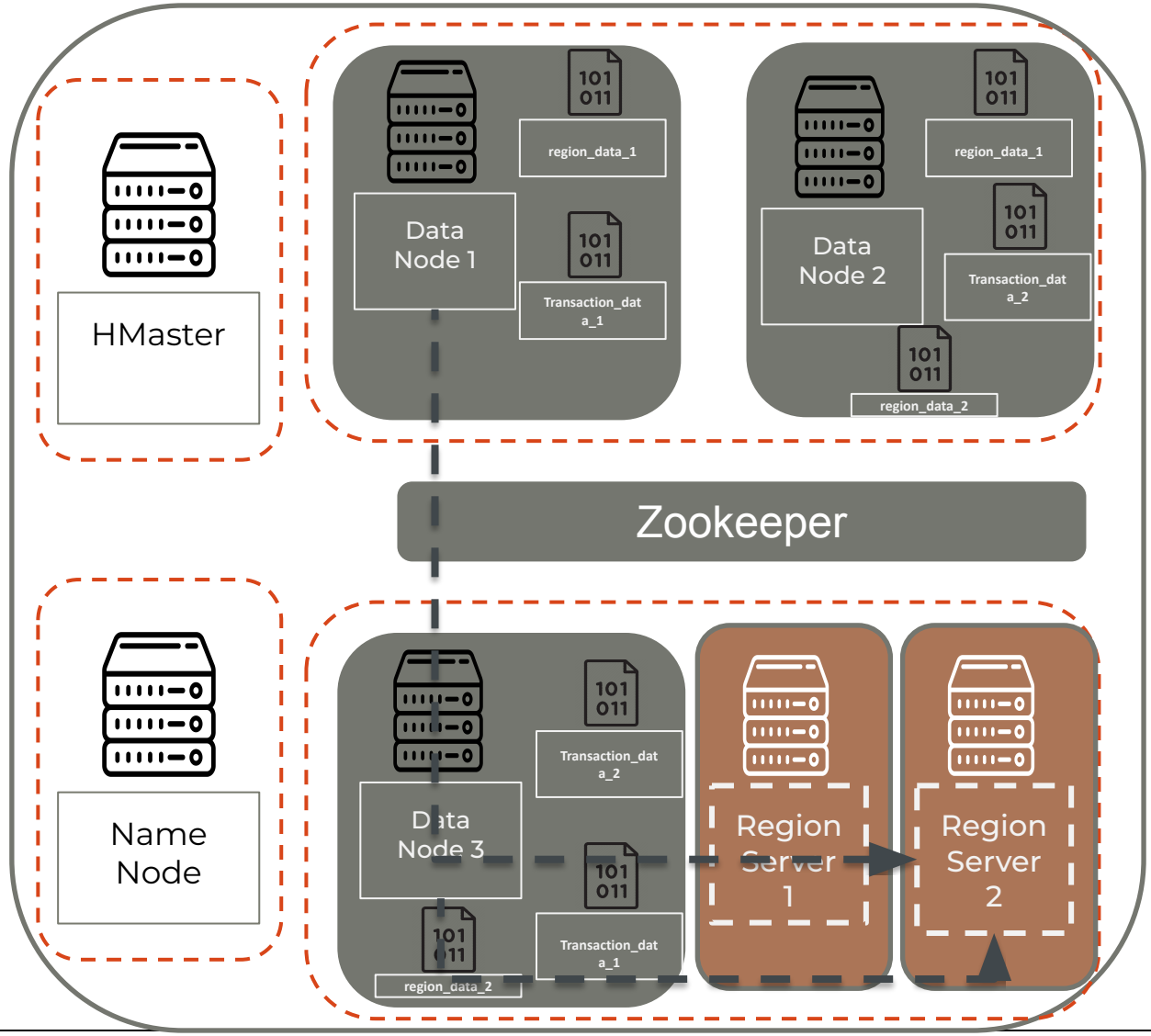


HBase Funcionamiento

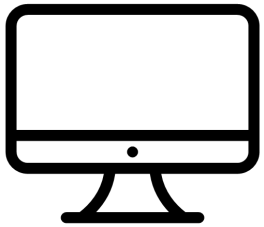


Client-1

```
hbase scan
electric_company_cups_regiones
```

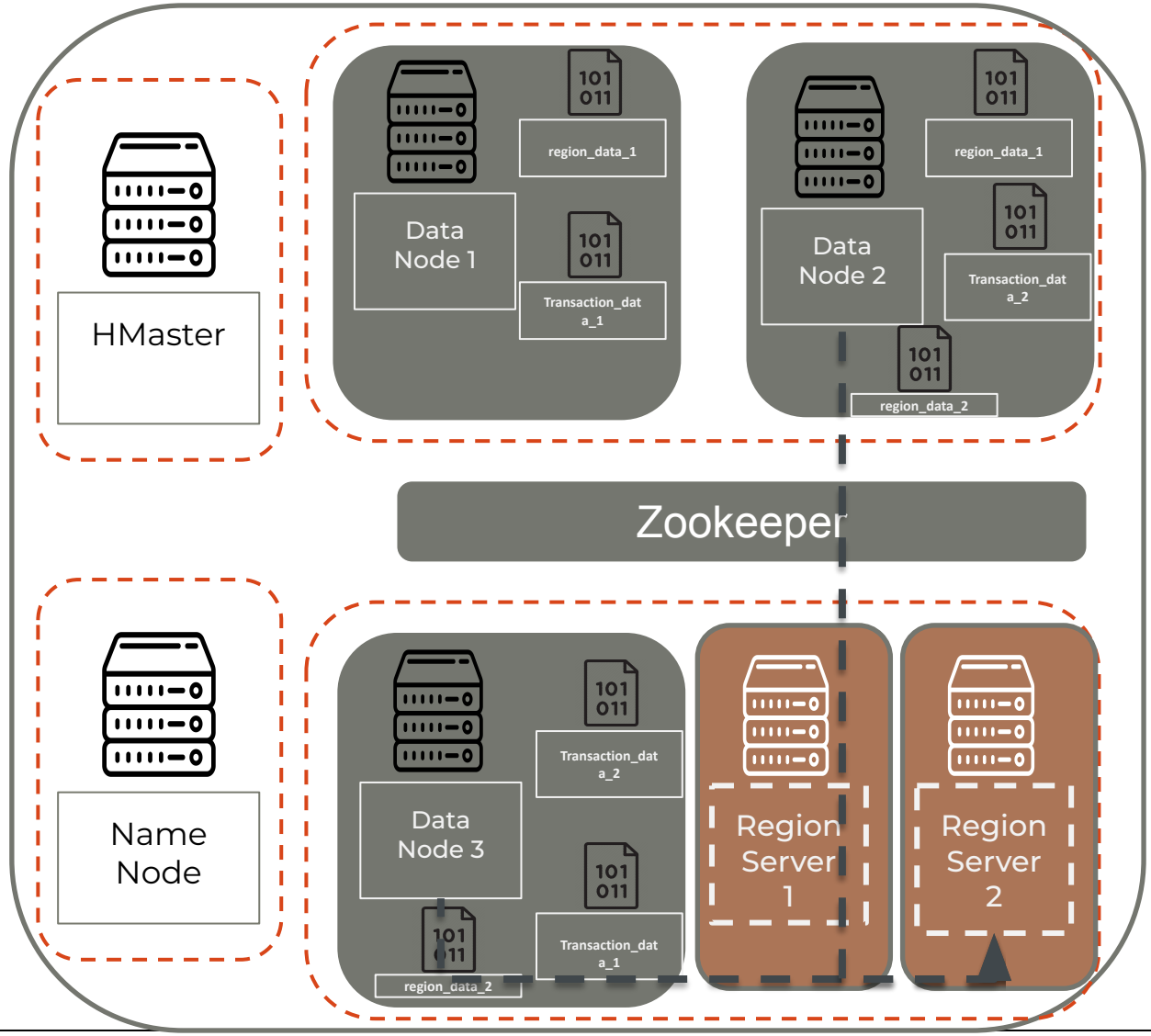


HBase Funcionamiento

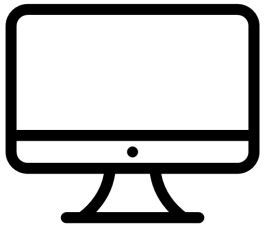


Client-1

```
hbase scan
electric_company_cups_regiones
```

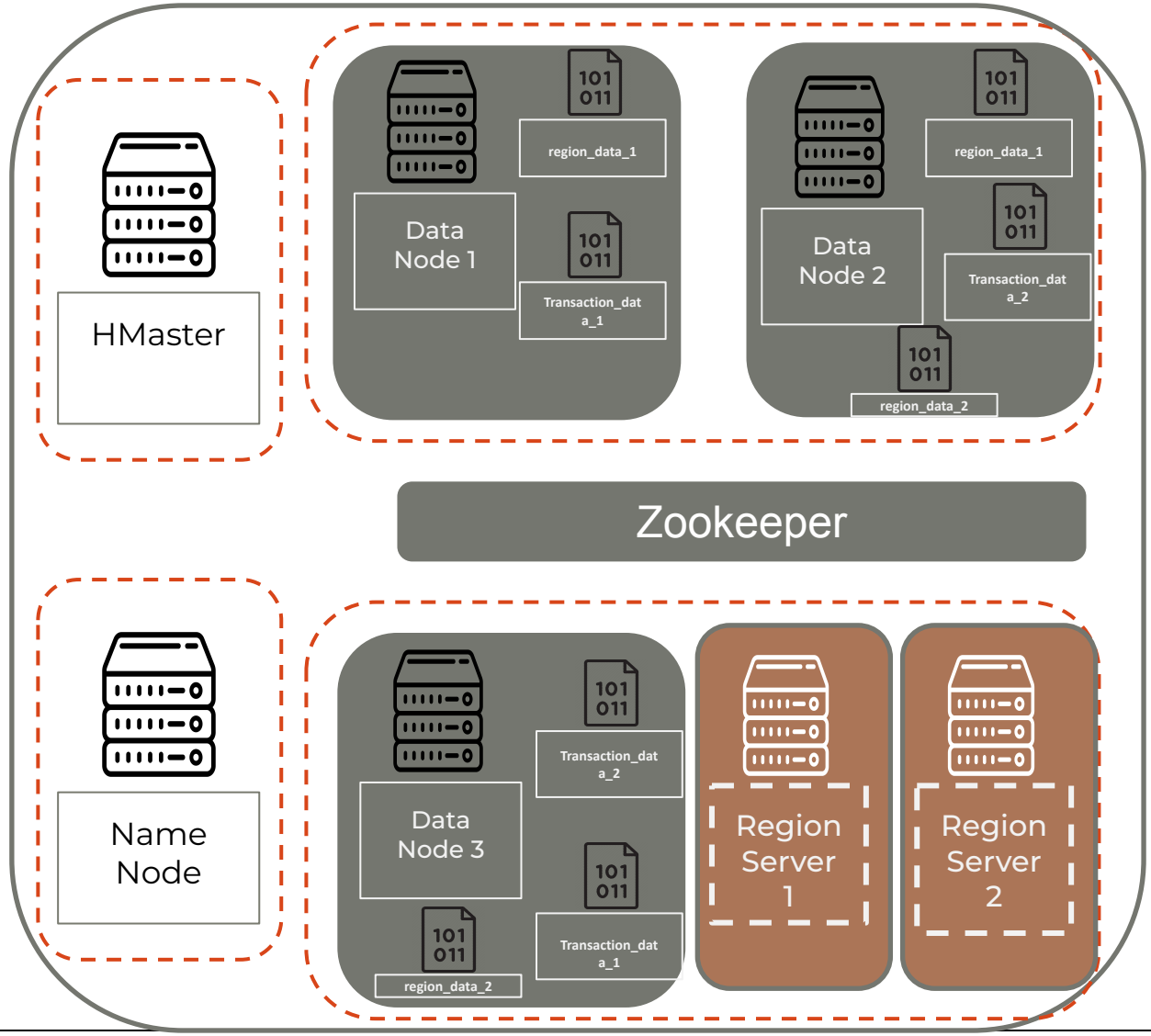


HBase Funcionamiento



Client-1

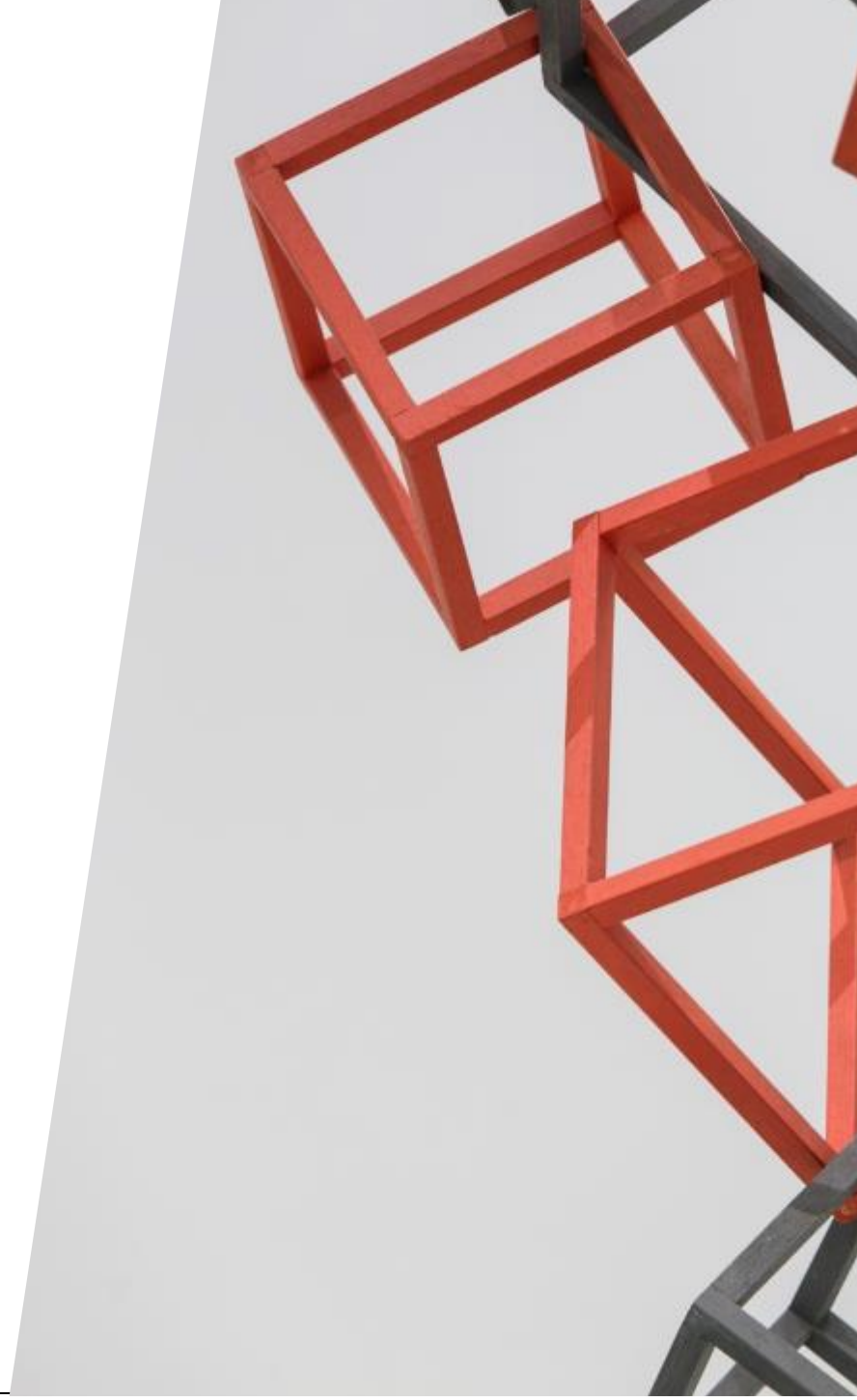
```
hbase scan
electric_company_cups_regiones

                ROW COLUMN+CELL
            '0001-municipio'
        region_data : valor_region
          timestamp=1631952266000,
          value='Colmenar Viejo'
            '0001-municipio'
        Tarificacion_data:pico
    timestamp=1631952500000, value=11.34
            '0001-municipio'
        Tarificacion_data:valle
    timestamp=1631954000000, value=100.34
            '0002-municipio'
        region_data : valor_region
          timestamp=1631952600000,
          value='Alcalá'
```



HBase - Introducción

- A raíz de las nuevas formas de almacenar se empiezan a pensar nuevas formas de consultar esa información
- El principal problema que pretende resolver es el acceso aleatorio a unos datos concretos
- Para ello usa HDFS como sistema de ficheros subyacente y una estrategia **Columnar**
- Con ello se consigue un acceso “rápido” a datos aleatorios en colecciones masivas de información
- Posteriormente nacen tecnologías de almacenamiento basadas en su misma filosofía supliendo algunas de sus carencias (Apache Cassandra)

Apache Flume



Flume - ETL

- En los primeros años del Big Data multitud de fuentes de datos y de procesos no podían ser cargadas directamente en HDFS
- Uno de los procesos más usados para este proceso es ETL
- ETL son siglas de:
  - **Extract:** Esta fase se encarga de extraer los datos de distintas de almacenamiento
  - **Transform**: Transforma los datos obtenidos en la fase de Extract
  - **Load**: Carga los datos transformados en otro tipo de tecnología de almacenamiento

Flume - ETL

- **Apache Flume** nace con el objetivo de satisficer el caso de uso de ETL y cargar en HDFS
- Además al tratar de cantides muy grades de datos se tiene que dar una capa intermedia que se encargará de guardar los datos antes volcarlos en una instancia de **HDFS**
- Estos dataset estaban almacenados en distintos solucionesno sólo soluciones de almacenmiento Big Data
- Para las transformaciones de datos sólo se permiten transformaciones básicas, no podemos mezclar datos ni realizar procesamientos usando **Map & Reduce**
- Antes de las que los sistemas esuvieran preoparados para el streming era la puerta de entrada de **Apache Kafka**

Flume - Arquitectura

- **Events**
  - An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

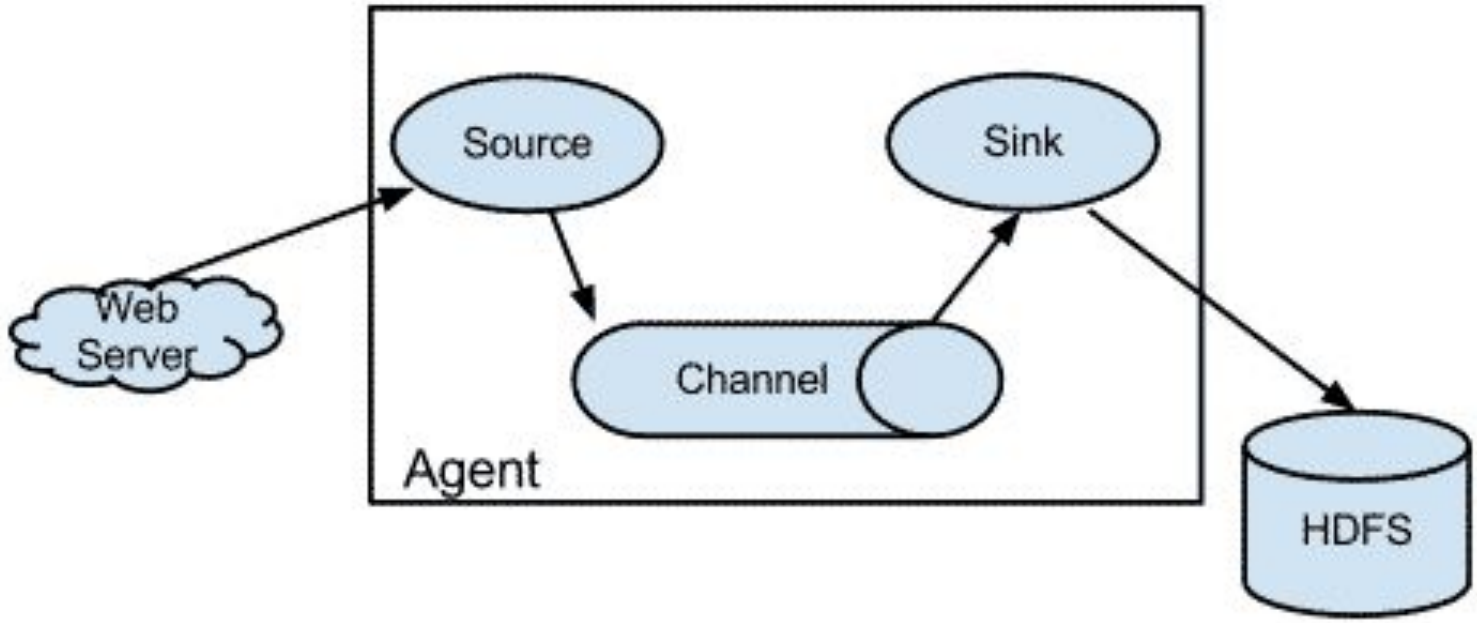
- **Sources**
  - The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source

- **Channels**
  - A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage lets source and sink run asynchronously.

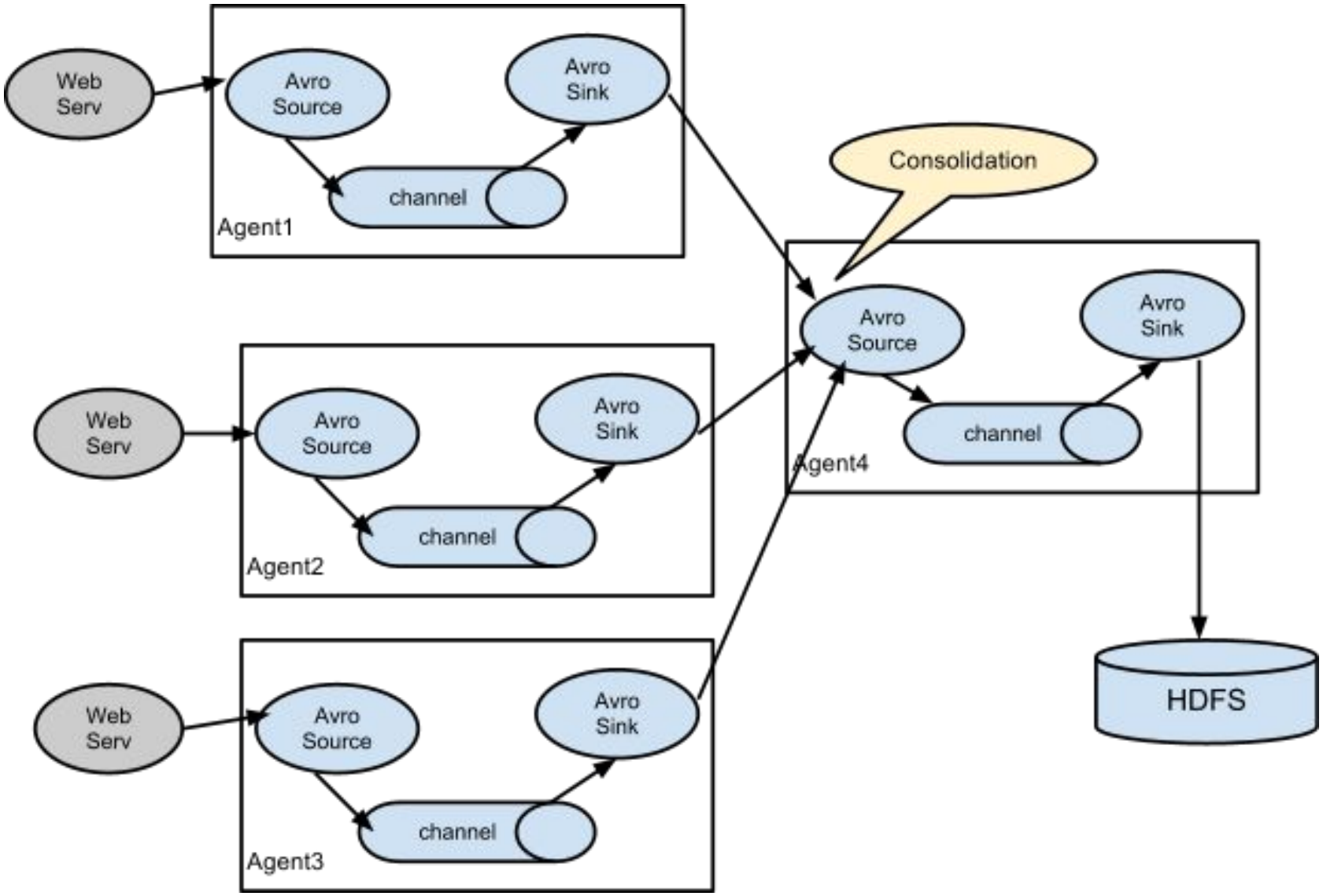
Flume - Arquitectura

- **Sinks**
  - The sink removes the event from the channel and forwards it on either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.
- **Agents**
  - An agent is the container for a Flume data flow. It is any physical JVM running Flume. An agent must contain at least one source, channel, and sink, but the same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

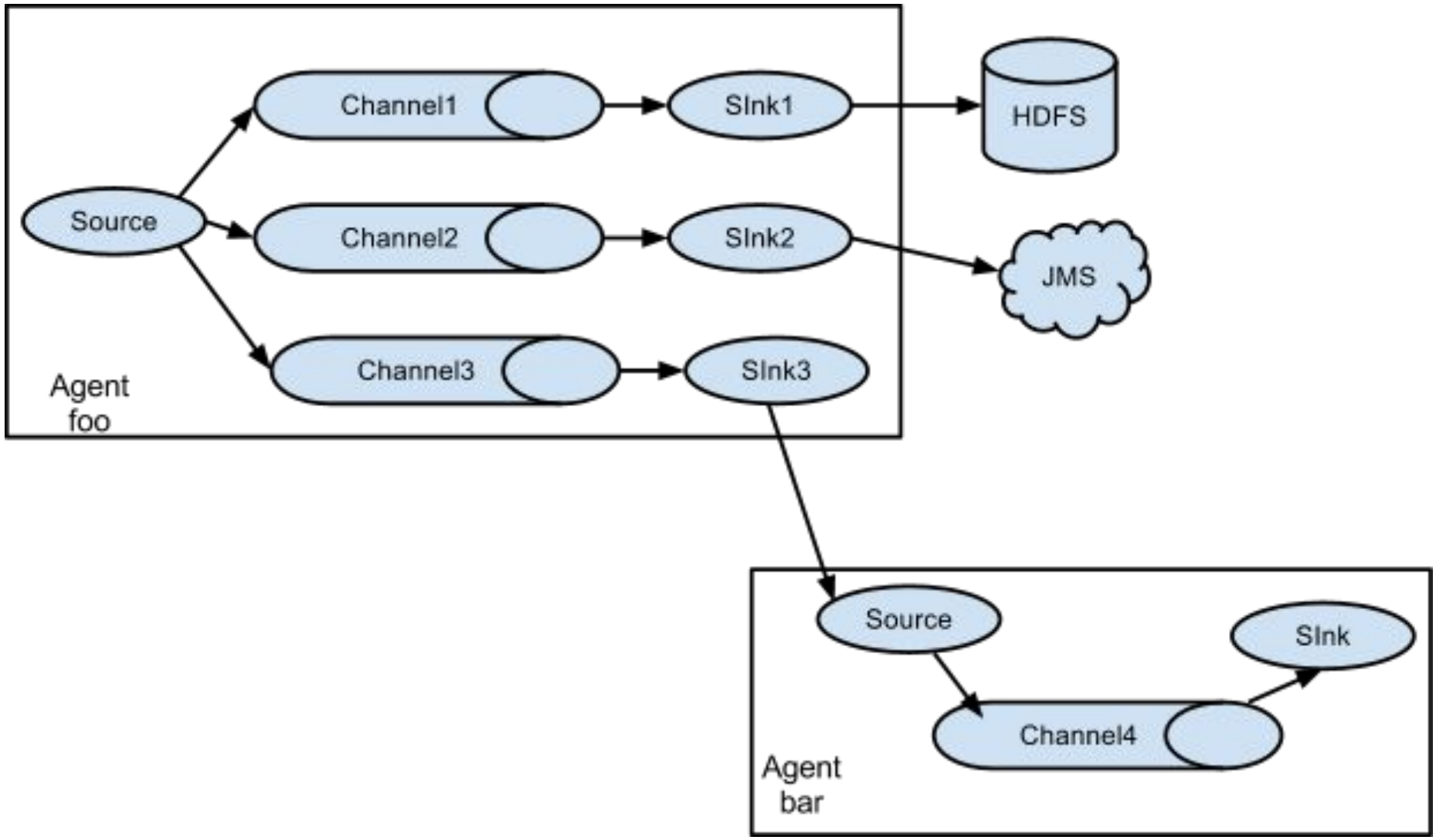
Flume - Arquitectura



Flume - Arquitectura



Flume - Arquitectura



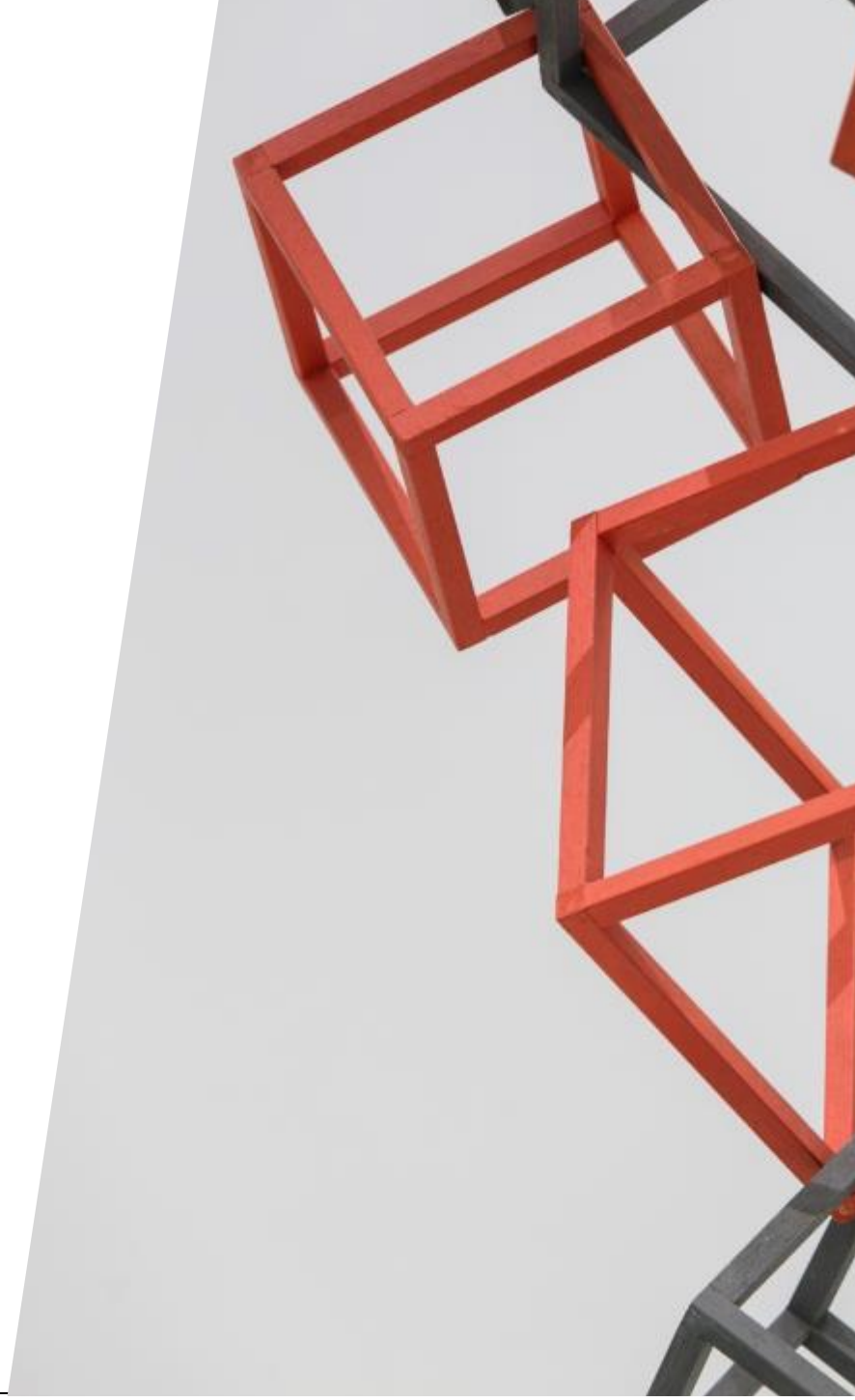
Flume - Componentes

- **Sources**:
  - Avro Source
  - Thrift Source
  - JMSSource
  - Converter
  - Spooling Directory Source
  - BlobDeserializer
  - NetCat Source
  - Sequence Generator
  - Syslog Sources
  - HTTP Source
  - .....
- **Channels**:
  - Memory Channel
  - JDBC Channel
  - File Channel
  - Kafka Channel
  - Spillable Memory Channel
  - Pseudo Transaction Channel
  - Custom Channel
  - .....
- **Sinks**:
  - HDFS Sink
  - Logger Sink
  - Avro Sink
  - Thrift Sink
  - IRC Sink
  - File Roll Sink
  - Null Sink
  - HBaseSink
  - MorphlineSolr Sink
  - ElasticSearch Sink
  - .....

Flume - Competidores

- **Apache Flume** nace como parte de un todo que es Hadoop y principalmente para poder ingestar datos a sus tecnologías de almacenamiento
- En la misma época
- Estos dataset estaban almacenados en distintos solucionesno sólo soluciones de almacenmiento Big Data
- Para las transformaciones de datos sólo se permiten transformaciones básicas, no podemos mezclar datos ni realizar procesamientos usando **Map & Reduce**
- Antes de las que los sistemas esuvieran preoparados para el streming era la puerta de entrada de **Apache Kafka**

Apache Mahout



Apache Mahout

- Tras la explosión del Big Data vino la siguiente gran explosión del **Machine Learning**
- Si una empresa tiene cantidades masivas de datos y tecnologías para procesarlos y almacenarlos de manera eficiente, ¿por qué no tomar decisiones con estos datos?
- El Machine Learning ya existía mucho antes de que llegaran las tecnologías Big Data
- Si tenemos datos nuevos ¿por qué no entrenar nuestros modelos constantemente?
- Antes de las existencia de los *Data Scientist* los Estadistas lanzaban modelos que duran varios días y que posiblemente no se volverían a entrenar en mucho tiempo
- Apache Mahout se crea para que estos entrenamientos se puedan lanzar periodicamente

Apache Mahout

- Para procesar cantidades masivas de datos
- Si una empresa tiene cantidades masivas de datos y tecnologías para procesarlos y almacenarlos de manera eficiente, ¿por qué no tomar decisiones con estos datos?
- El Machine Learning ya existía mucho antes de que llegaran las tecnologías Big Data
- Si tenemos datos nuevos ¿por qué no entrenar nuestros modelos constantemente?
- Antes de las existencia de los *Data Scientist* los Estadistas lanzaban modelos que duran varios días y que posiblemente no se volverían a entrenar en mucho tiempo
- Apache Mahout se crea para que estos entrenamientos se puedan lanzar periódicamente

Apache Mahout

- Apache Mahout nos ofrece una interfaz par poder lanzar trabajos de Machine Learning usando tecnología de almacenamiento y procesamiento Hadoop

- Todos los dataset para nuestra algoritmia (entrenamiento, test, predicción, ...) tienen que estar
  - Almacenados en un formato legible para mahout
  - Almacenados en un sistema de ficheros compatible con Hadoop

- Si un algoritmo no está presente en Apache Mahout, hay que implementarlo, lo que no suele ser una tarea sencilla

- Las tecnologías de computo másivo más modernas implementan sus propias interfaces para ML por lo que Mahaout está perdiendo tracción

Apache Mahout

- En apache Mahout están disponibles los siguientes algoritmos:
  - clustering
    - Canopy Clustering
  - linear-algebra
  - recommenders
  - regression
    - Closed Form Solutions
      - Ordinary Least Squares
      - Ridge Regression
    - Autocorrelation Regression
      - Cochrane Orcutt Procedure
      - Durbin Watson Test

Afi Escuela