CORSO AWS & MONGODB

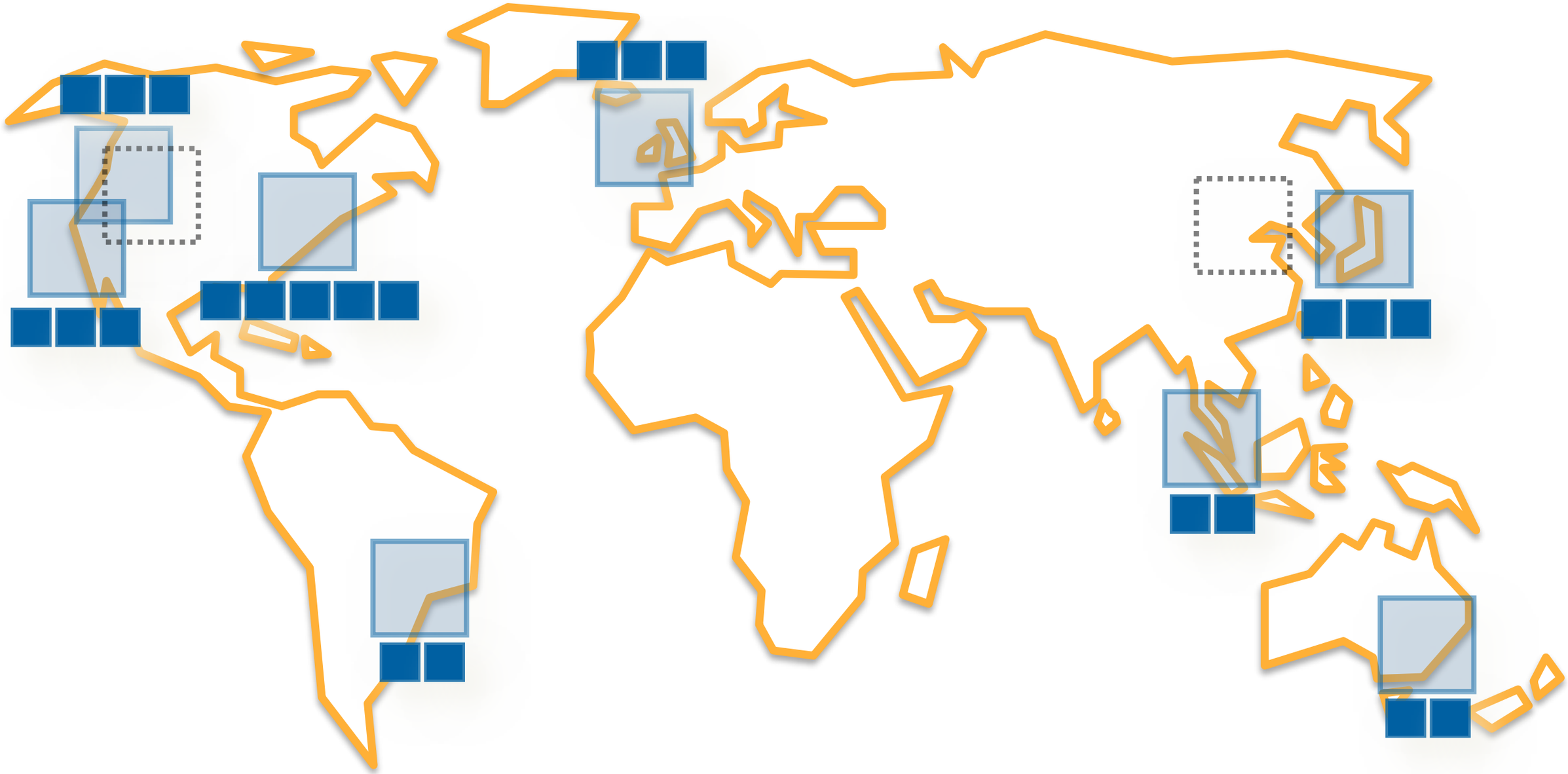
“Learn quickly and Think Well!”

Stefano Dindo
s.dindo@zero12.it
@stefanodindo

Amazon Web Services

ZERO12

@stefanodindo



ZERO12

@stefanodindo

ZERO12

@stefanodindo

Enterprise Applications

- Virtual Desktops
- Collaboration and Sharing

Platform Services

Databases

- Relational
- No SQL
- Caching

Analytics

- Hadoop
- Real-time
- Data Warehouse
- Data Workflows

App Services

- Queuing
- Orchestration
- App Streaming
- Transcoding
- Email
- Search

Deployment & Management

- Containers
- Dev/ops Tools
- Resource Templates
- Usage Tracking
- Monitoring and Logs

Mobile Services

- Identity
- Sync
- Mobile Analytics
- Notifications

Foundation Services

- **Compute** (VMs, Auto-scaling and Load Balancing)
- **Storage** (Object, Block and Archive)
- **Security & Access Control**
- **Networking**

Infrastructure

- Regions
- Availability Zones
- CDN and Points of Presence

ZERO12

@stefanodindo

Compute

EC2 · Lambda · EC2 Container Service · Elastic Beanstalk · Elastic Load Balancing

Networking

VPC · Direct Connect · Route 53

Analytics

EMR · Data Pipeline · Kinesis · Machine Learning · QuickSight · Elasticsearch Service

Developer Tools

CodeCommit · CodeDeploy · CodePipeline

Management Tools

CloudWatch · Cloud Formation · CloudTrail · Config · OpsWorks · Service Catalog

Security & Identity

Identity & Access Management · Directory Service · Trusted Advisor · Cloud HSM · Key Management Service · Web App Firewall

Storage & Content Delivery

S3 · CloudFront · EFS · Glacier · Storage Gateway · Snowball

Application Services

API Gateway · AppStream · CloudSearch · Elastic Transcoder · SES · SQS · SWF

Hubs

Mobile Hub

Mobile Services

Cognito · Device Farm · Mobile Analytics · SNS

Database

RDS · DynamoDB · ElastiCache · RedShift · Simple DB · Database Migration Service

IOT

IoT

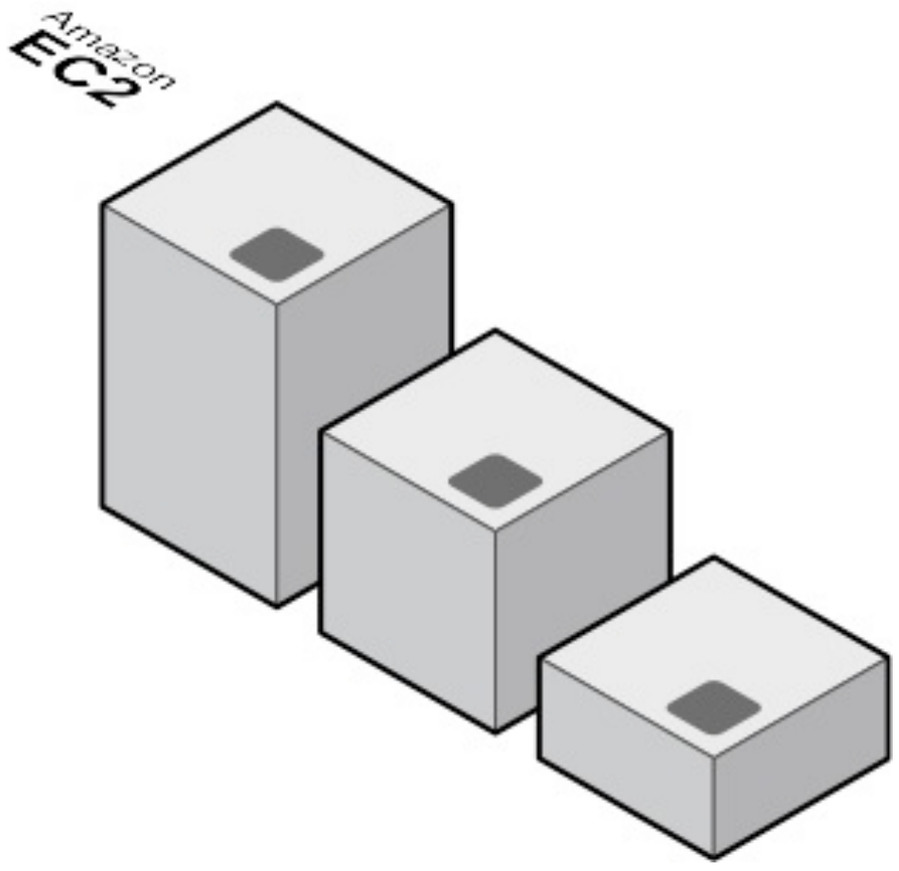
Enterprise Applications

WorkSpaces · WorkDocs · WorkMail

EC2

ZERO12

@stefanodindo



Stopped

Terminated

Running

Waiting

ZERO12

@stefanodindo

Tipologie istanza

General Purpose		Compute Optimized	Storage Optimized		GPU Purpose	Memory Optimized
T2	M3/4	C3/4	I2	HS1	G2	R3

ZERO12

@stefanodindo

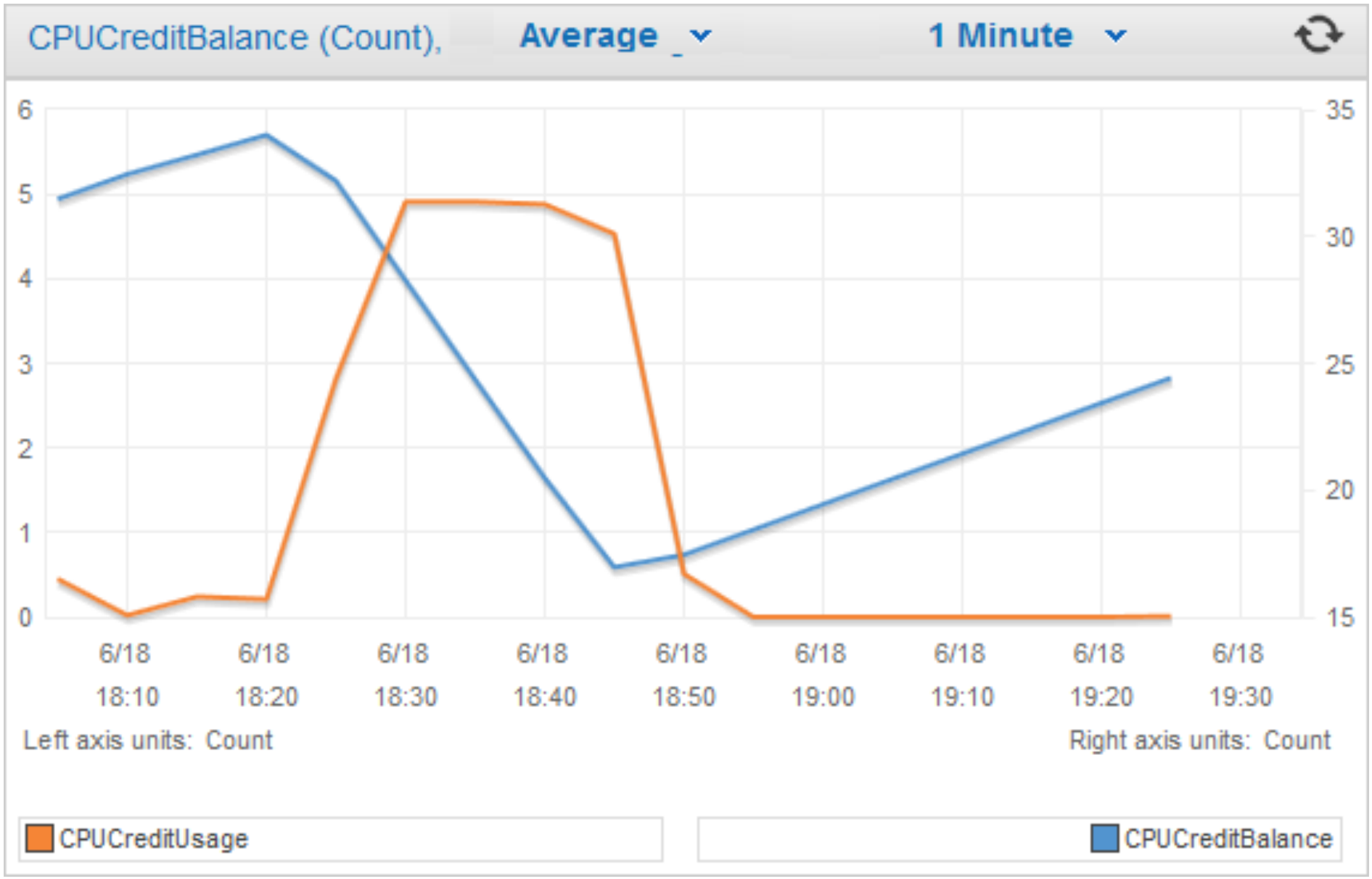
ZERO12

@stefanodindo

Nome	vCPU	Baseline performance	RAM	CPU credit / ora	Max Crediti
t2.micro	1	10%	1 GB	6	144
t2.small	1	20%	2 GB	12	288
t2.medium	2	40%	4 GB	24	576

ZERO12

@stefanodindo



ZERO12

@stefanodindo

- Persistente
- Boot in secondi
- I/O a Pagamento
- Stop a Pagamento

EBS

- Termina con istanza
- Boot in minuti
- I/O a gratis
- Stop gratuito

S3

General Purpose

Provisioned IOPS (SSD)

SSD High IO

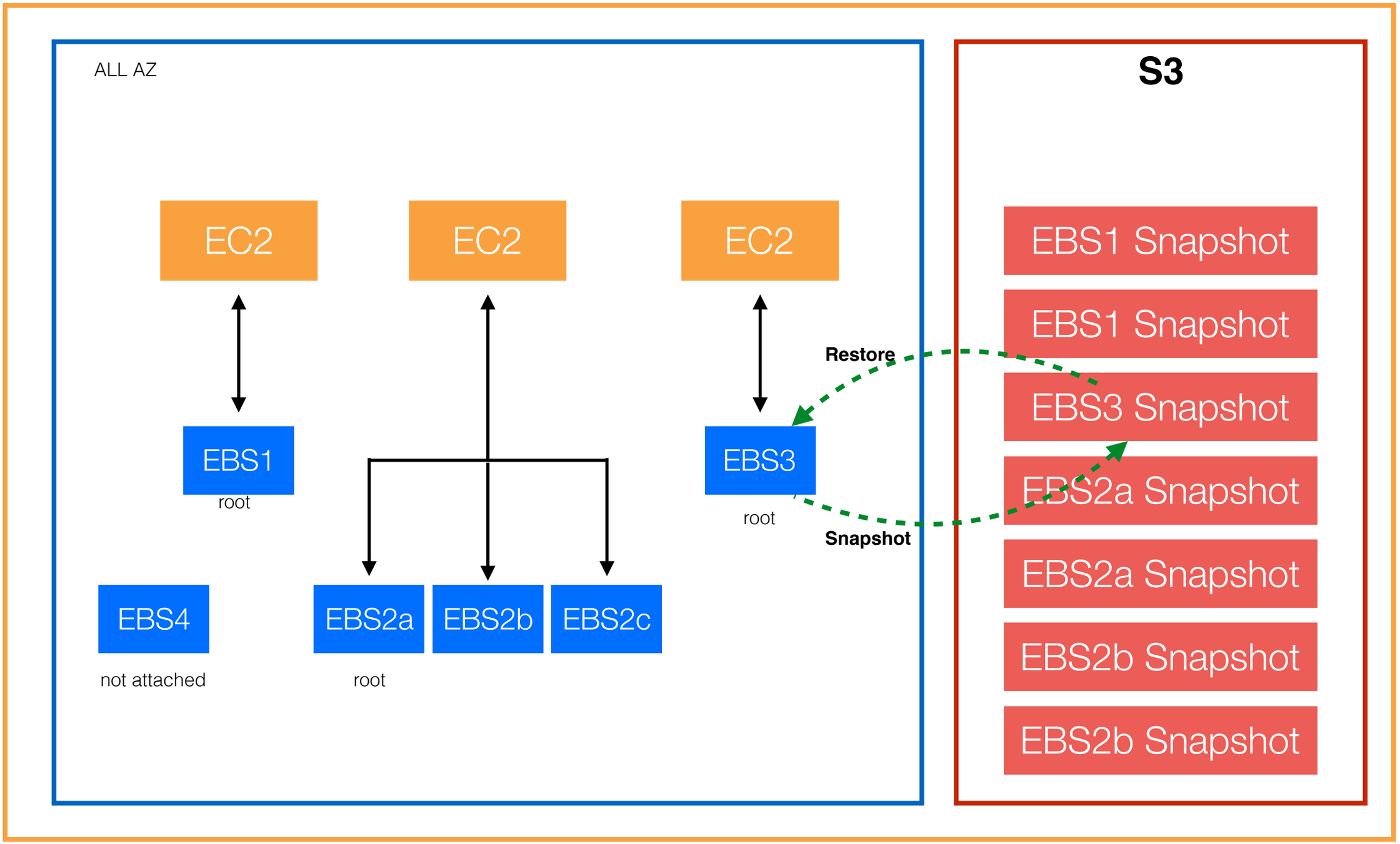
ZERO12

@stefanodindo

EBS Life cycle

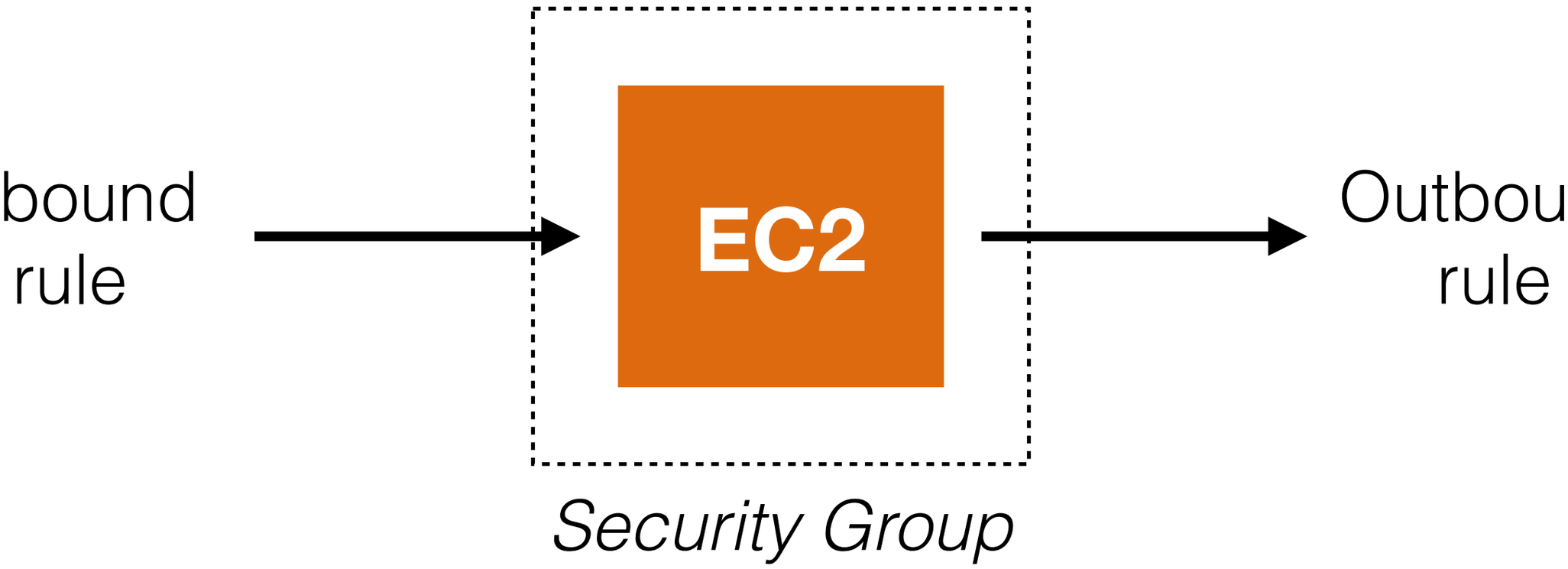
amazon web services | Partner Network

CONSULTING PARTNER



ZERO12

@stefanodindo



ZERO12

@stefanodindo

Spot

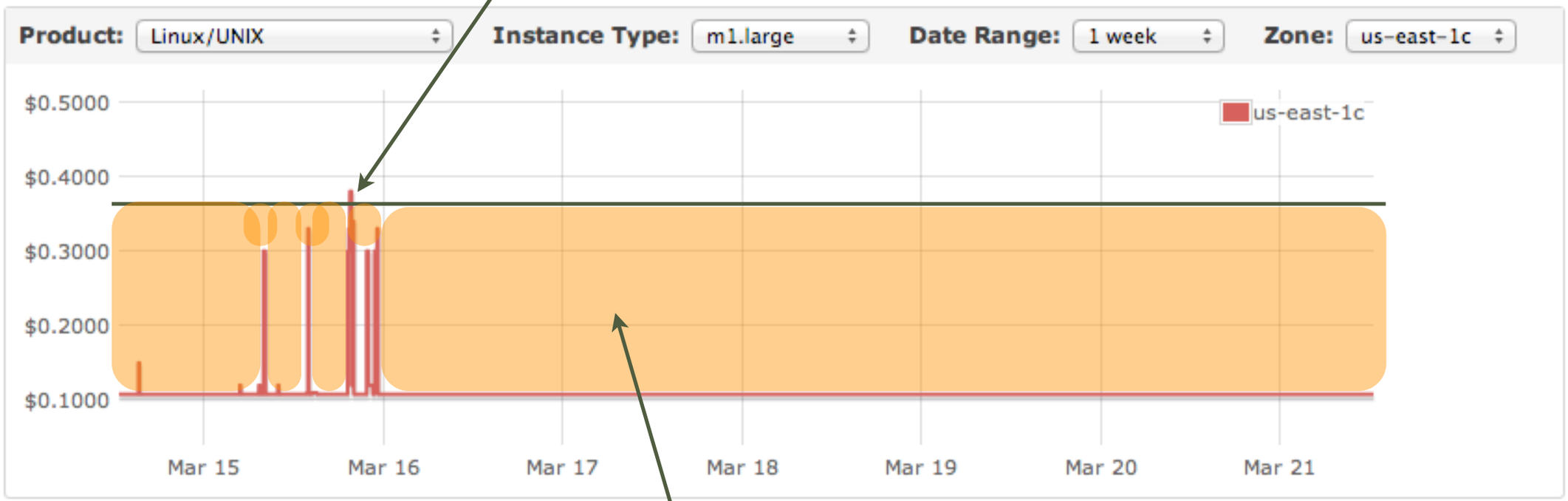
On-Demand

Reserved

ZERO12

@stefanodindo

$ spot > $ on-demand --> on-demand



$ Risparmiati

S3

Alta scalabilità per
l’accesso agli oggetti

Inserire un oggetto in S3

significa durabilità del 99,999999999%

Alta scalabilità per
l’accesso agli oggetti

Che cos'è S3 ?

Servizio di Storage
ad alta scalabilità

Web Store,
No File System

Accesso via API

Veloce

Alta disponibilità e durabilità

Economico

ZERO12

@stefanodindo

A Regional Service

I tuoi dati non lasciano la regione a meno che tu non decida di spostarli

ZERO12

@stefanodindo

AWS S3 Storage Class

Standard

Oggetti per cui si desidera avere alta disponibilità

Es. Media sempre accessibili

Reduced Redundancy Storage

Oggetti che si ipotizzare di perdere o ricreare

Es. Media con encondings diverso

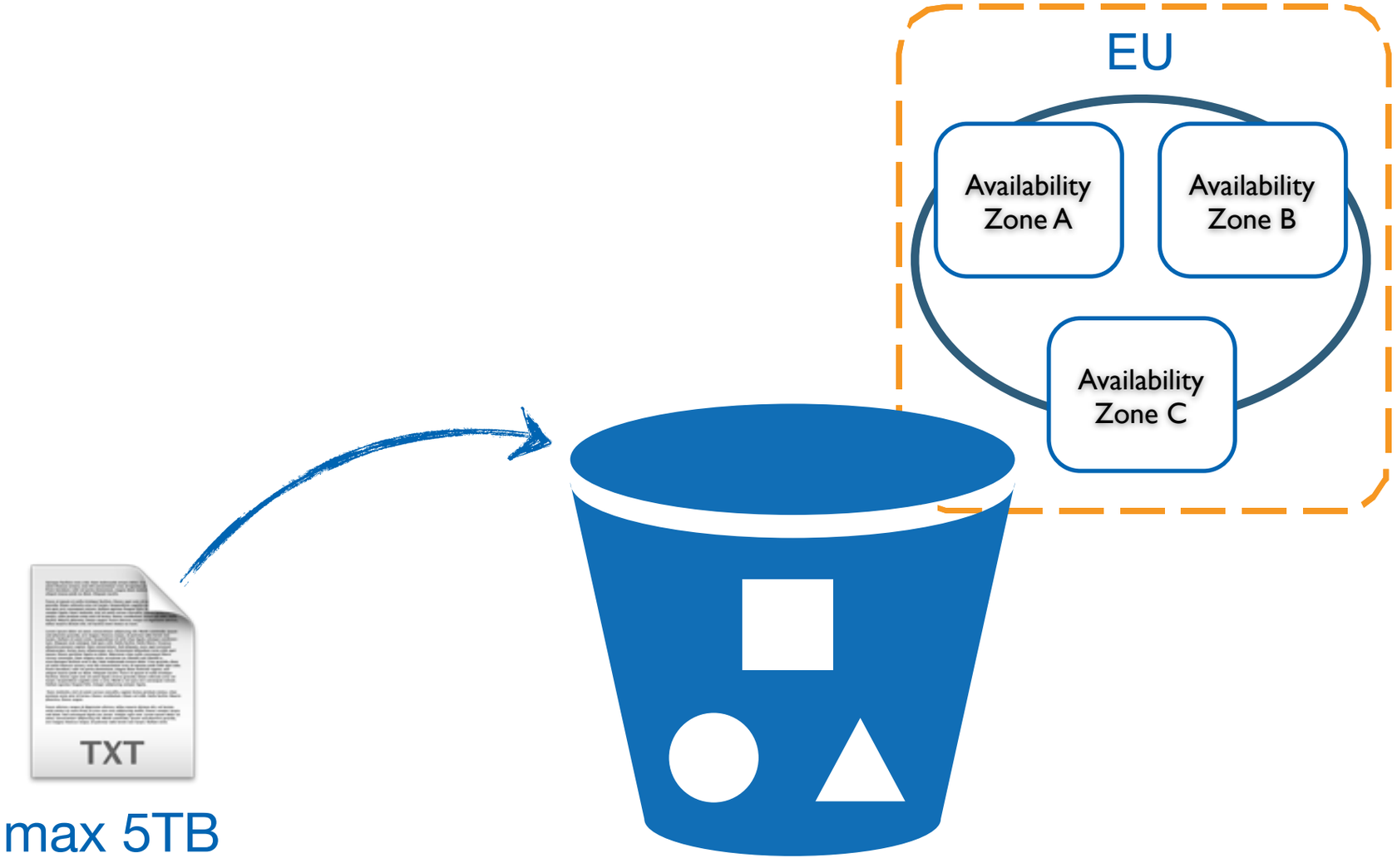
Glacier

Oggetti che si archiviare per lungo tempo

Es. Digital archives di dati aziendali

ZERO12

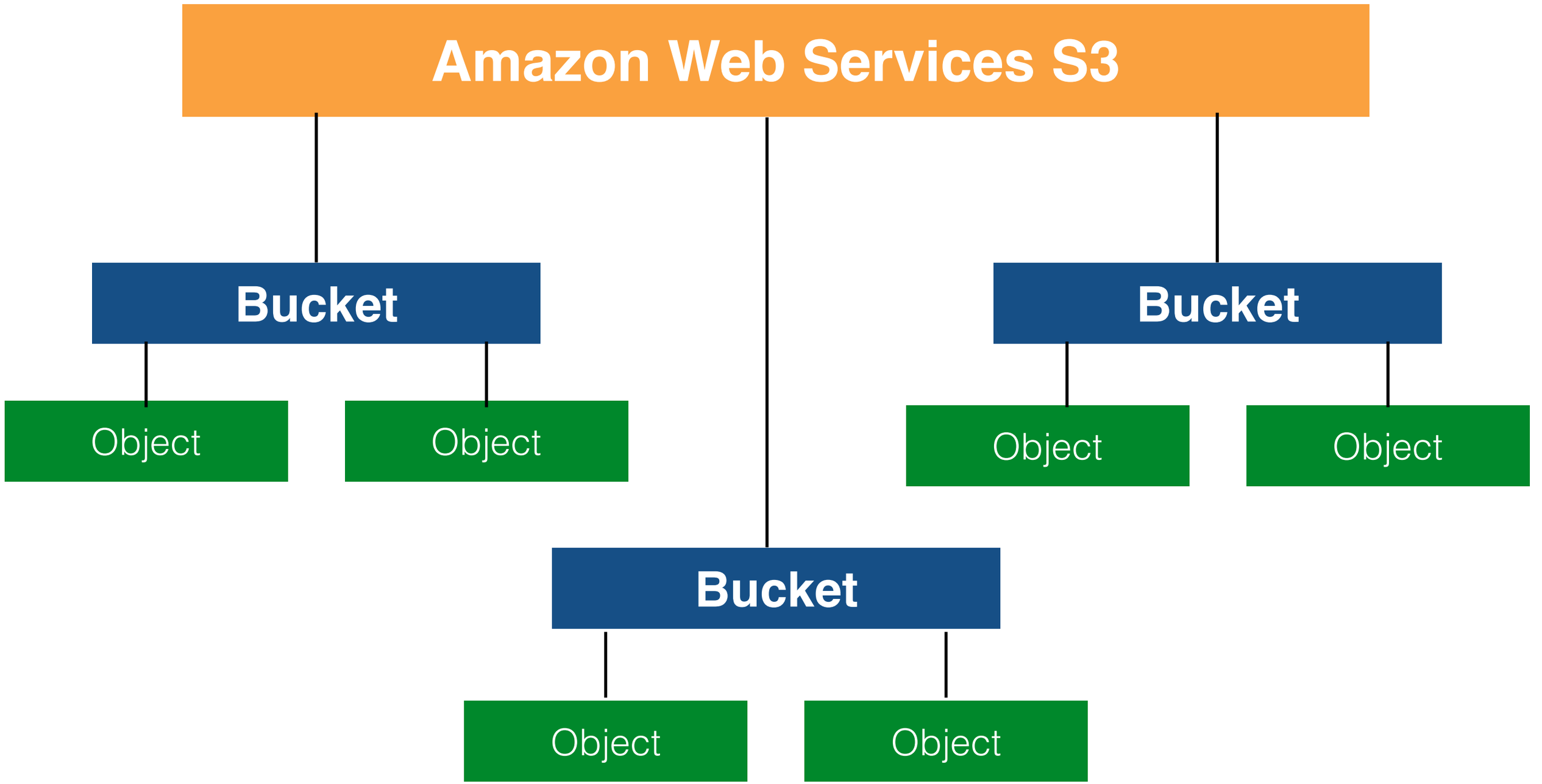
@stefanodindo



ZERO12

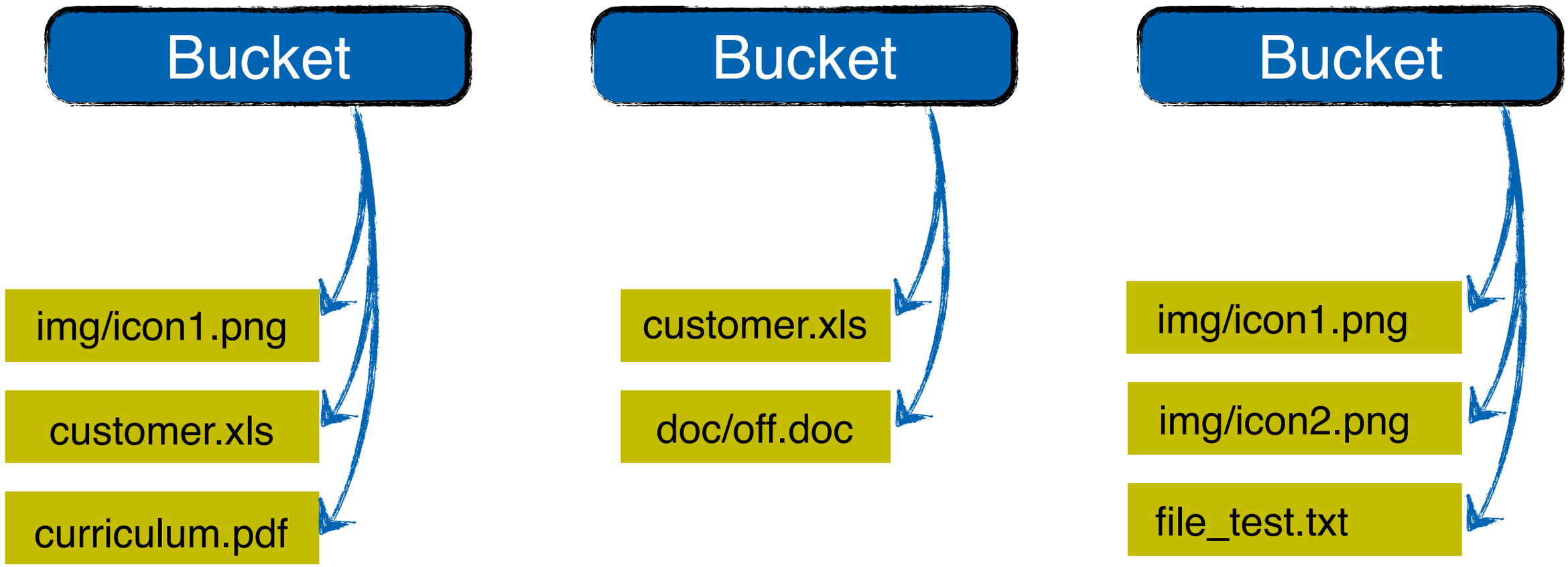
@stefanodindo

AWS S3 Namespace



ZERO12

@stefanodindo



ZERO12

@stefanodindo

AWS S3 Namespace

Object Key

Max 1024 UTF-8

Unica per il bucket

Incluso il path nel prefisso

/var/www/customers/html/index.php

Questa è una Object Key

Crittografia automatica
dei dati

Semplice

Durabilità

Server Side encryption

Strong
AES-256

Self Managed
Non è necessario applicare metodi di gestione delle chiavi

Sicuro
massimo 3 accessi multipli

Object deletion

Cancellazione permanente dell’oggetto S3

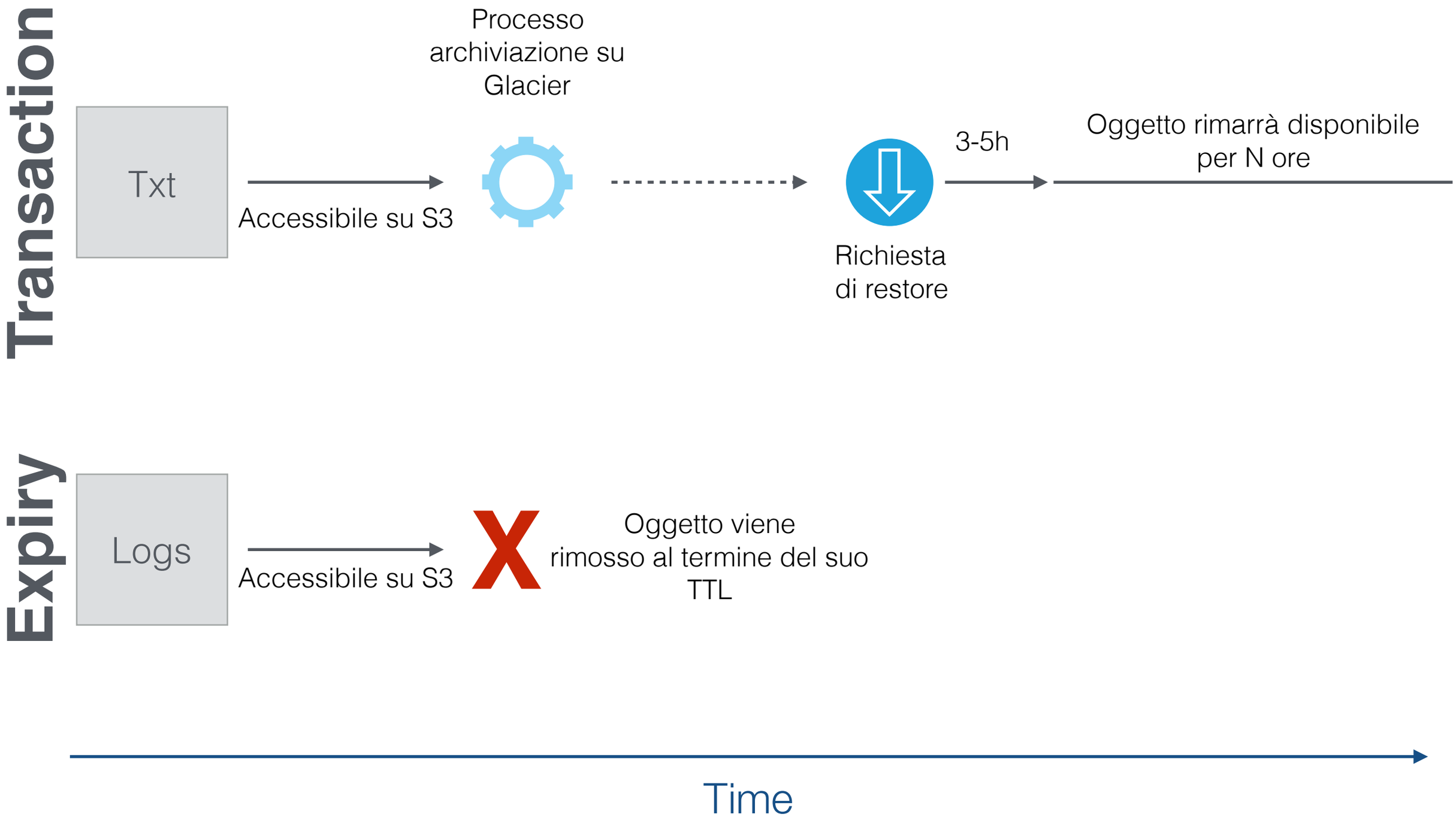
Lifecycle Management

Archiviazione degli oggetti

Spostare gli oggetti su Glacier e rimuoverli da S3

@stefanodindo

AWS S3 Lifecycle Mng



RDS

ZERO12

@stefanodindo

Database on-premises

- Ottimizzazione applicativi
- Scaling
- Alta disponibilità del servizio
- Database backup
- DB Patches
- DB installazione
- OS Patches
- Installazione Sistema Operativo
- Server maintenance
- Rack & Stack
- Energia elettrica, HVAC, rete

@stefanodindo

RDS - Full managed service
Significa

Scaling

Alta disponibilità del servizio

Database backup

DB Patches

DB installazione

OS Patches

Installazione Sistema Operativo

Server maintenance

Rack & Stack

Energia elettrica, HVAC, rete

DEMANDARE LO STACK ad AWS

ZERO12

@stefanodindo

RDS - Full managed service
Significa

Il tuo focus è l'ottimizzazione dell'applicativo

Ottimizzazione applicativi

ZERO12

@stefanodindo

Self Managed vs Full managed

Self-Managed database	AWS-Managed Database
Hai la responsabilità di aggiornamenti e backup dell'intero stack	Update, backup e failover sono gestiti dal servizio
Completa responsabilità delle funzioni di sicurezza	AWS fornisce un'alto livello di sicurezza di infrastruttura, certificazione. Devi gestire sicurezza accesso al database
Replicazione è impegnativa e deve essere fatta manualmente	Replicazione e failover sono parte del servizio AWS

ORACLE®

Oracle Database SE1, SE, EE + options and packs
Bring-your-own-license or pay-as-you-go (hourly)

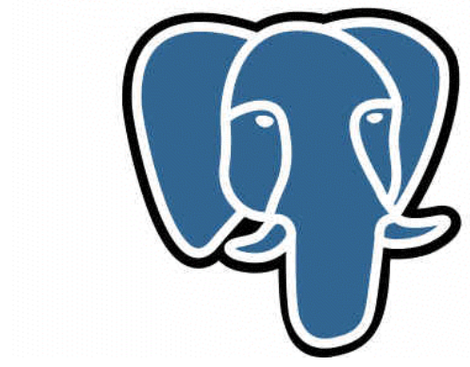
MySQL®

Amazon Aurora

MariaDB

Microsoft® SQL Server® 2008

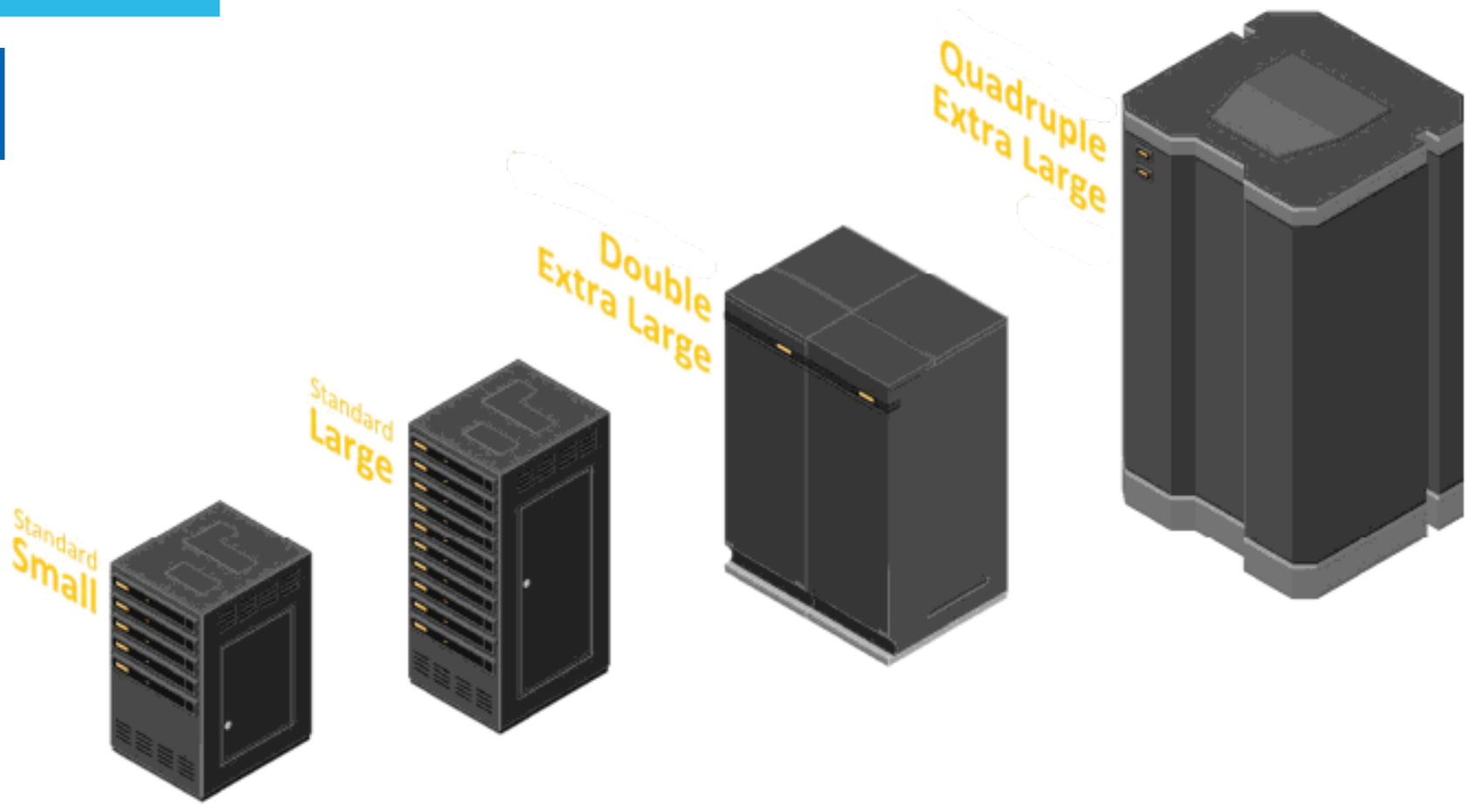
PostgreSQL



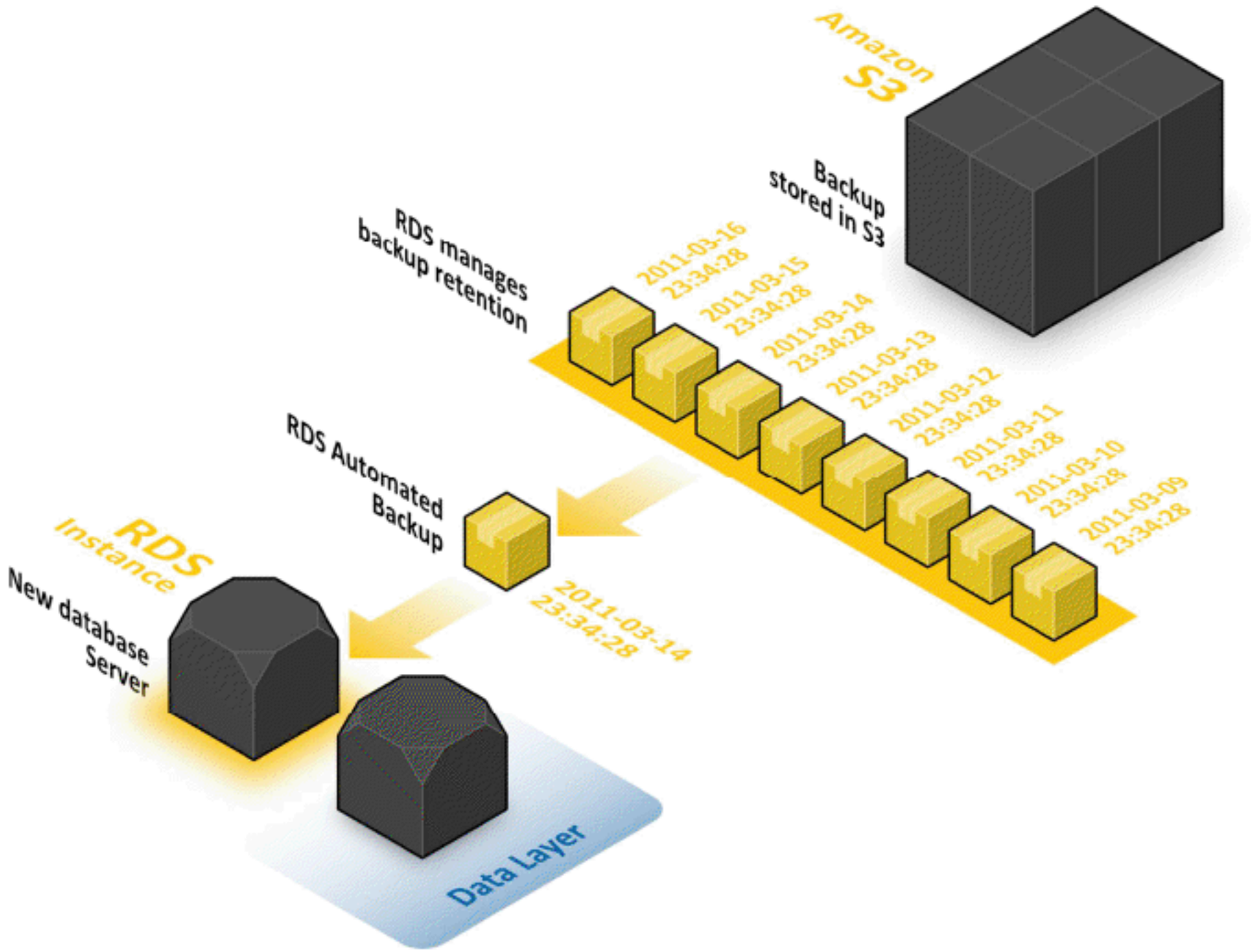
RDS

Backup schedulati

Multi-AZ



RDS


Amazon
S3
Backup
stored in S3
RDS manages
backup retention
2011-03-16
23:34:28
2011-03-15
23:34:28
2011-03-14
23:34:28
2011-03-13
23:34:28
2011-03-12
23:34:28
2011-03-11
23:34:28
2011-03-10
23:34:28
2011-03-09
23:34:28
RDS Automated
Backup
2011-03-14
23:34:28
RDS
Instance
New database
Server
Data Layer

ZERO12

@stefanodindo

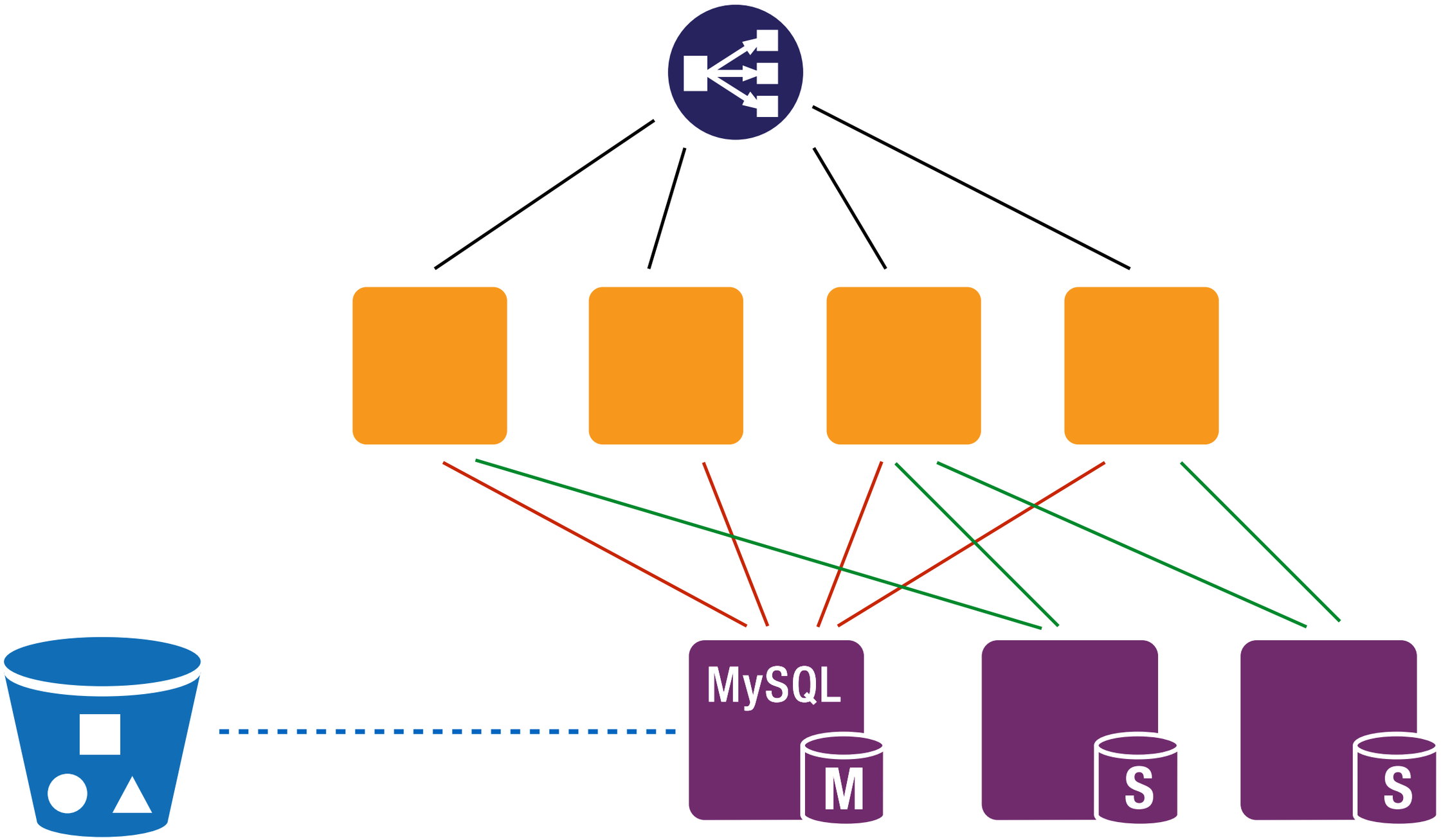
La configurazione Multi-AZ garantisce: alta disponibilità e durabilità

- Availability zone sono infrastrutture fisiche distinte e indipendenti
- Le operazioni su una configurazione Multi-AZ sono replicante, in modo sincrono, con le altre zone della stessa AWS Region
- La gestione del failover è automatica
- Le attività di manutenzione pianificate sono applicate prima ai backup e poi al master

ZERO12

@stefanodindo

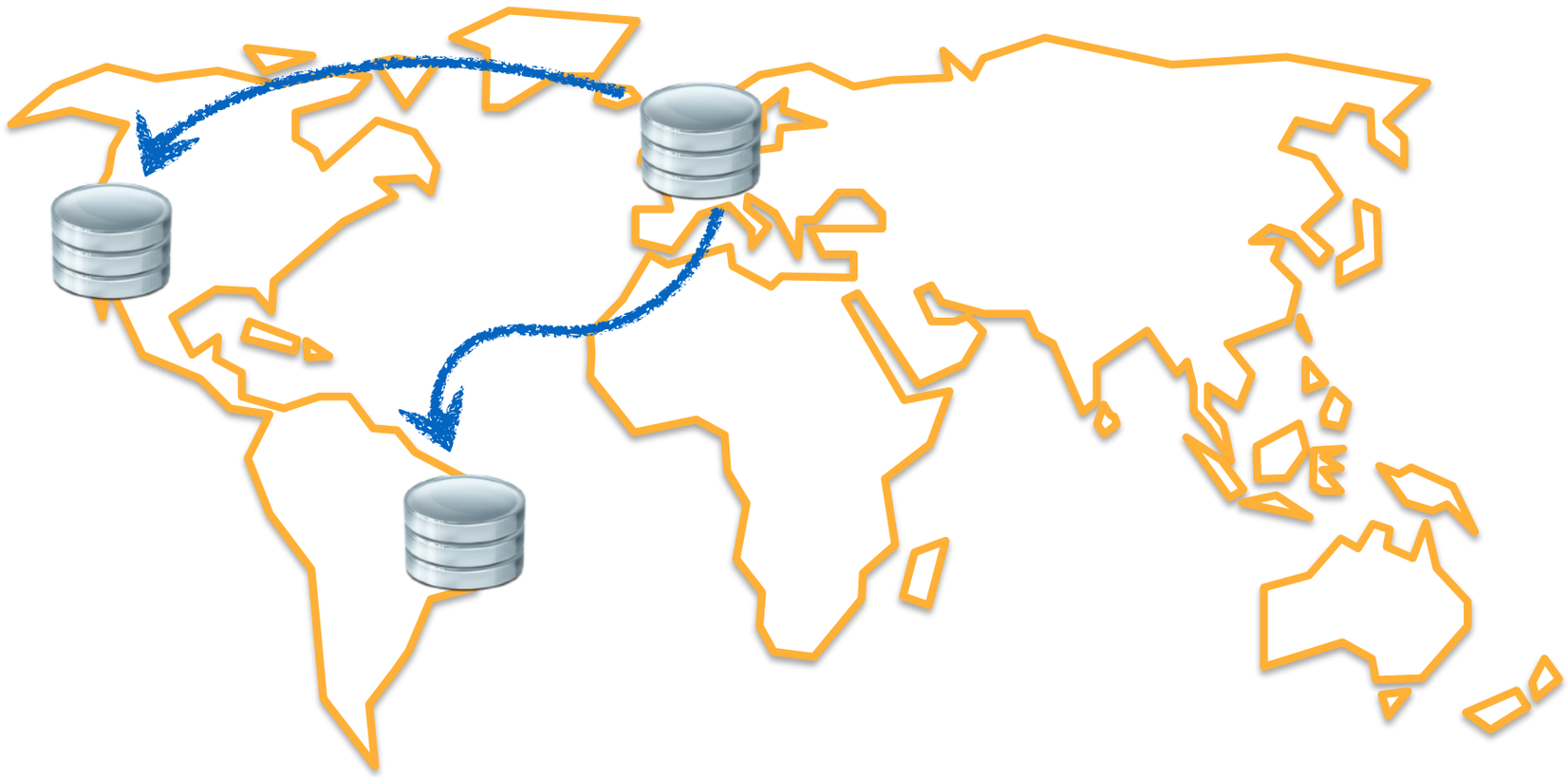
Scenario architettura



ZERO12

@stefanodindo

Cross-Region snapshot Copy

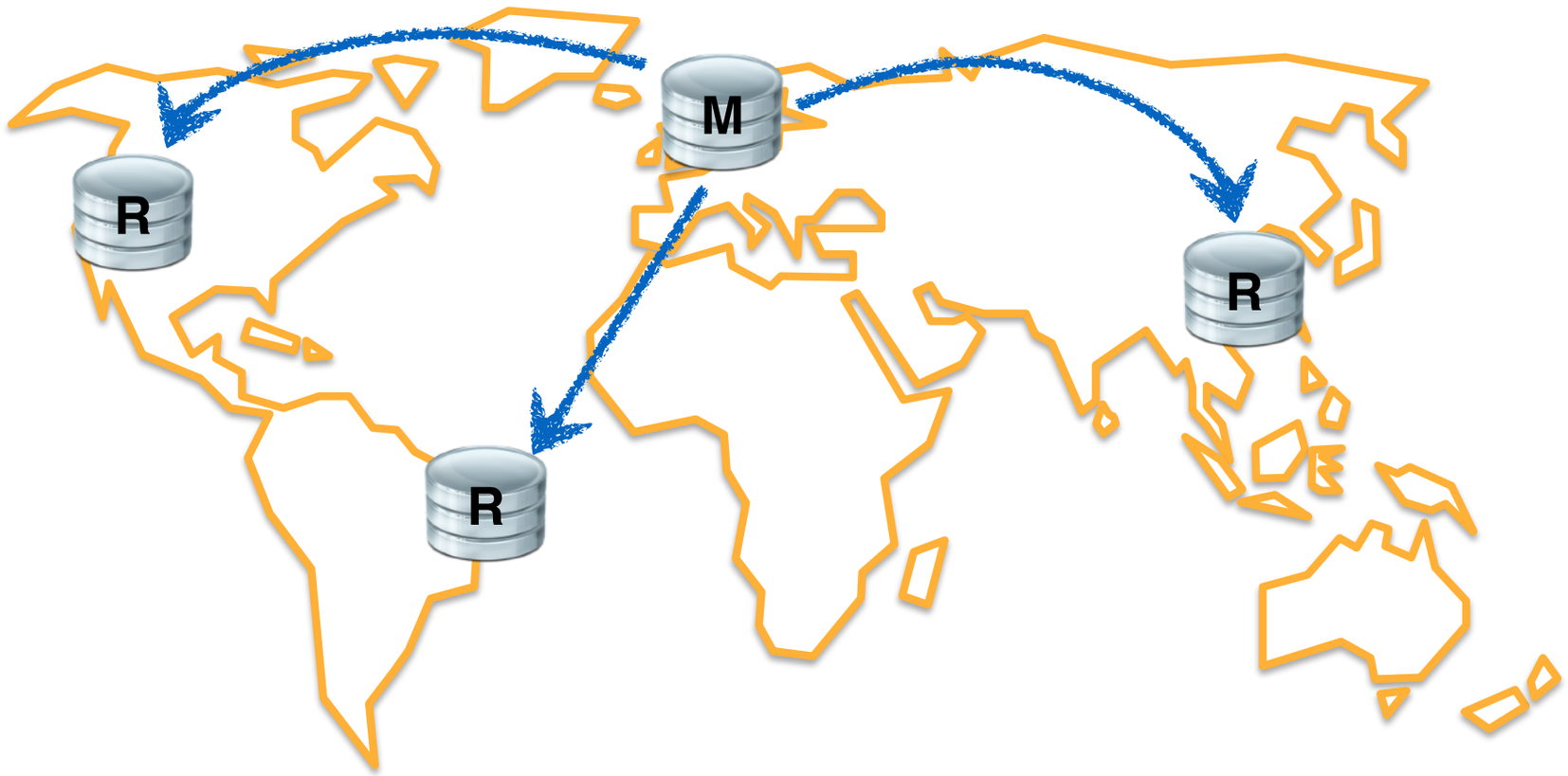


- Maggiore durabilità e facilità di migrazione
- Possibilità di copiare snapshot in regioni diverse
- Warm standby Disaster recovery
- Funzione utilizzata per migrare anche regione del servizio

ZERO12

@stefanodindo

Cross-Region Read Replica



- Semplice e veloce soluzione di ricovero in caso di disatro
- Velocità di accesso ai dati in lettura per applicazioni distribuite WW
- Facile promuovere a master un nodo in caso di migrazione

ZERO12

@stefanodindo

Come funziona la fatturazione

=

db.m3.large

+

100GB

+

1000 IOPS

$ 267,8 = 720h * $ 0,2 + 100 * $ 0,138 + 1000* $ 0,11

VPC

ZERO12

@stefanodindo

[Concetti fondamentali]

- Amazon VPC è una rete isolata all'interno della rete pubblica di Amazon Web Services
- In una VPC è possibile:
  - Creare più reti sottoreti pubbliche / private
  - Avviare risorse con la classe di IP di sottorete desiderato
  - Definire Security Group, Access Control List (ACL) e Subnet

    Route specifiche per la VPC

ZERO12

@stefanodindo

[Perchè usare una VPC]

- I motivi per usare una VPC sono:
  - La possibilità di isolare la sottorete dagli altri account AWS
  - La maggiore sicurezza disponibile con VPC
  - La possibilità di creare un’estensione della propria rete aziendale
  - Gli indirizzi IP non cambiano durante le operazioni di STOP/START delle istanze

ZERO12

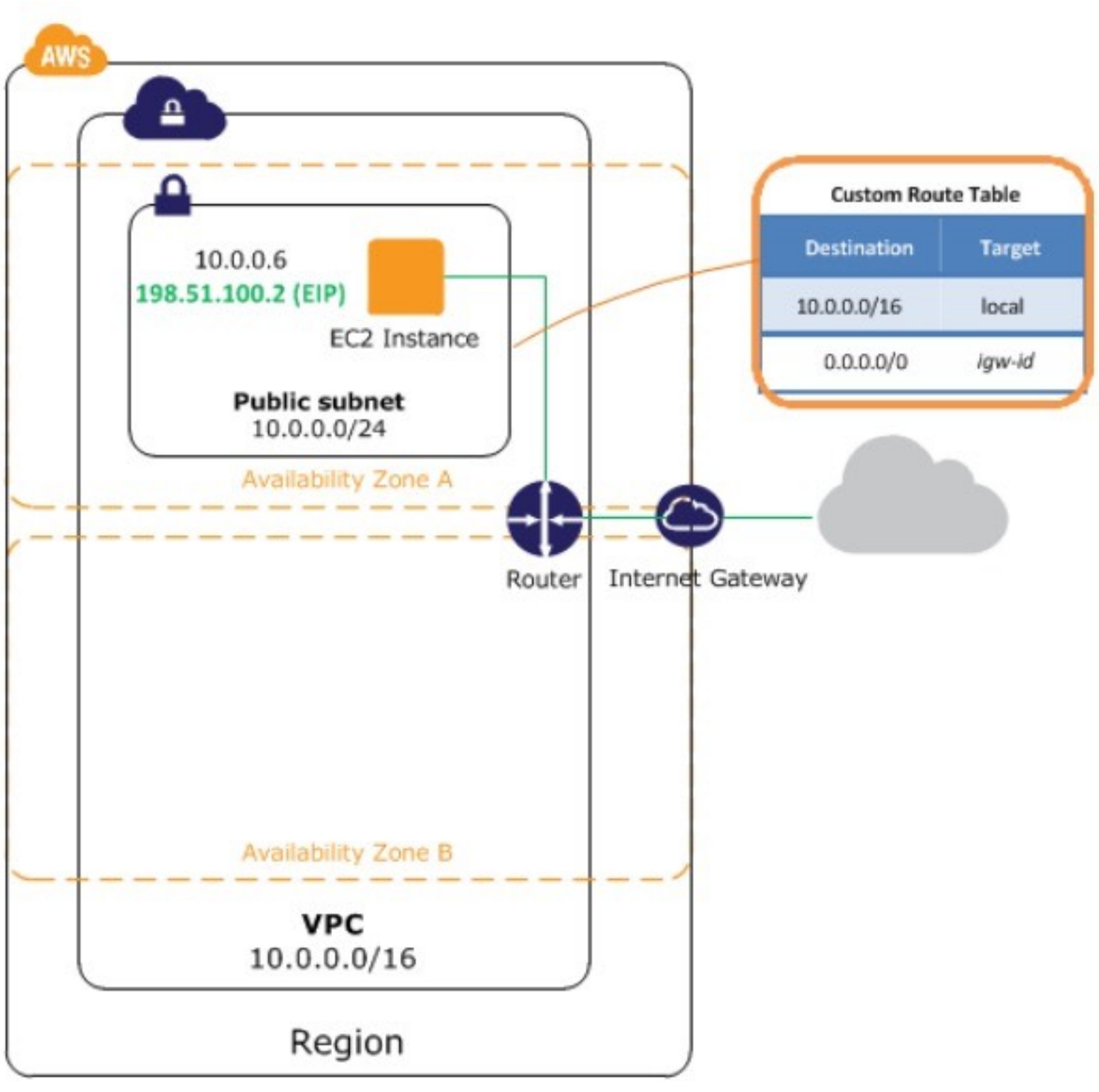
@stefanodindo

SCENARIO 1

Public Subnet Only

ZERO12

@stefanodindo



ZERO12

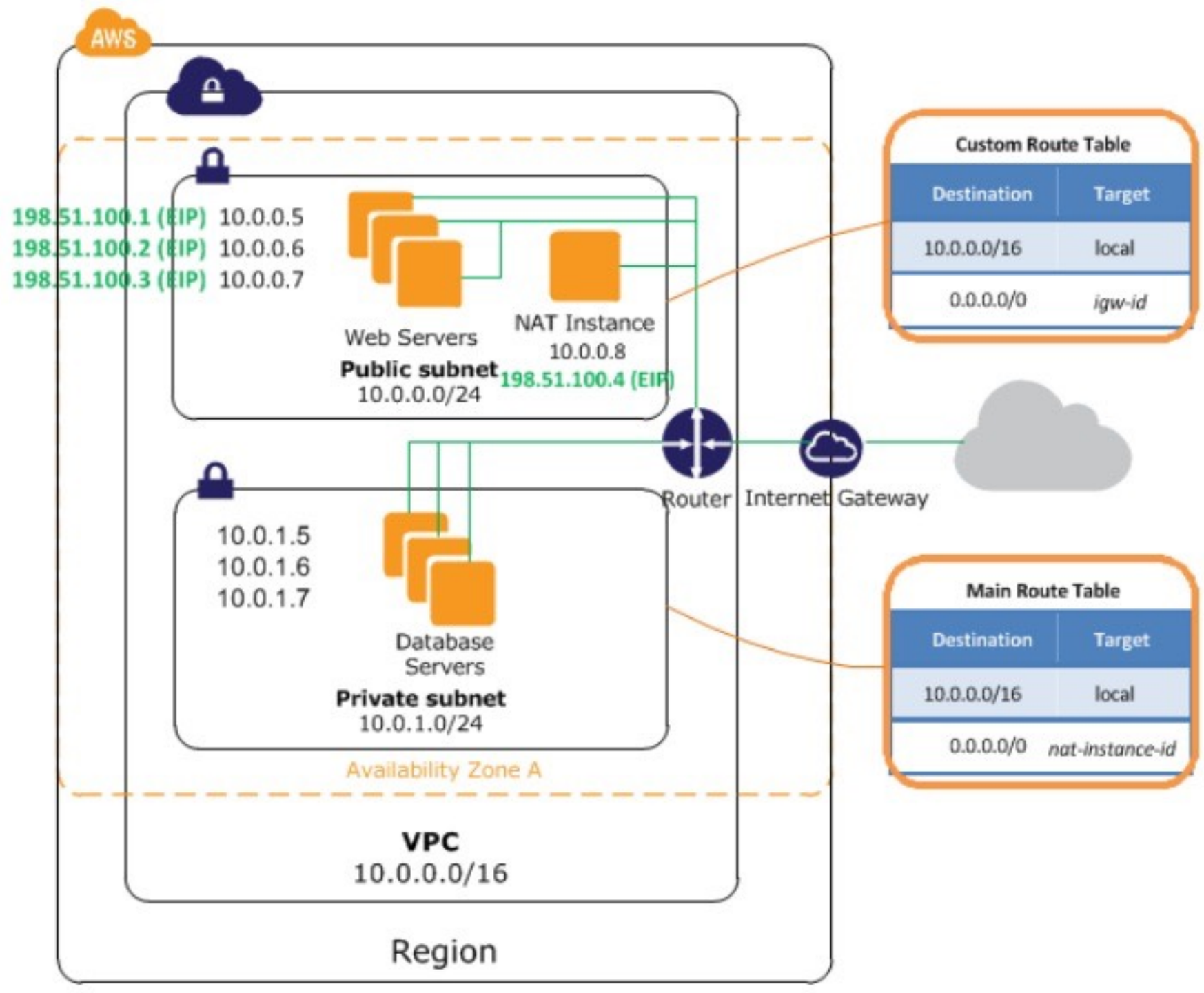
@stefanodindo

SCENARIO 2

Public & Private Subnet

ZERO12

@stefanodindo



ZERO12

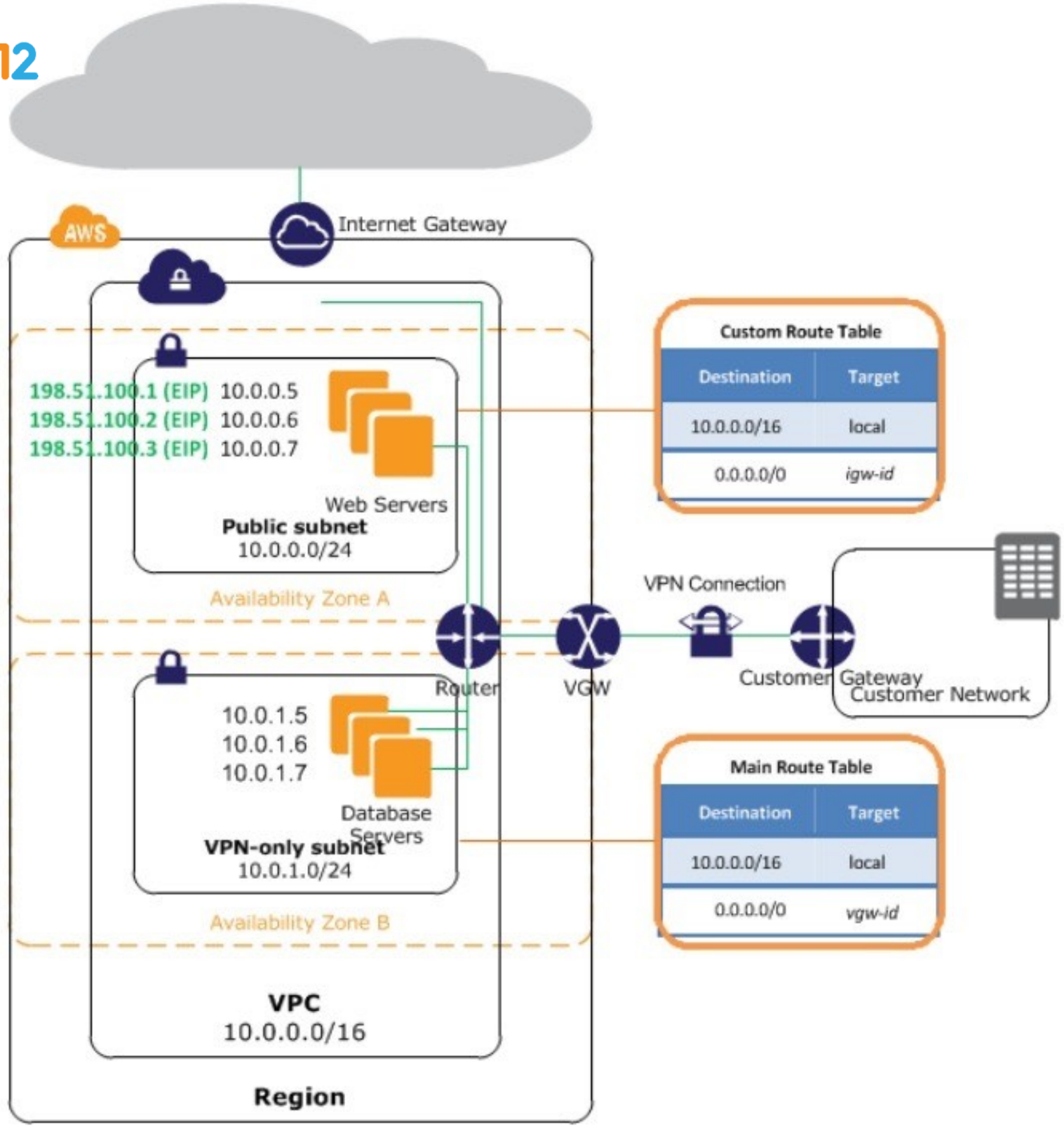
@stefanodindo

SCENARIO 3

Public & Private Subnet e Hardware VPN Access

ZERO12

@stefanodindo




mongoDB

Cos'è un documento per MongoDB?

Un documento è una struttura dati in stile JSON formata da un numero variabile di coppie campo - valore

```
{
   “nome”: “Giorgio”,
   “cognome”: “Rossi”,
   “email”: “giorgio.rossi@gmail.com”,
   “cell”: 3281432896,
   “sport”: [“nuoto”, “calcio”]
}
```

MongoDB salva i documenti su disco in formato BSON (http://bsonspec.org).
La dimensione massima di un documento BSON è 16MB.

Terminologia

SQL - Like	Corrispettivo in MongoDB
Database	Database
Tabella	Collection
Riga	Document
Colonna	Field

Come accedo ai dati

Shell javascript

db.collection.find()

db.collection.insert()

db.collection.update()

db.collection.delete()

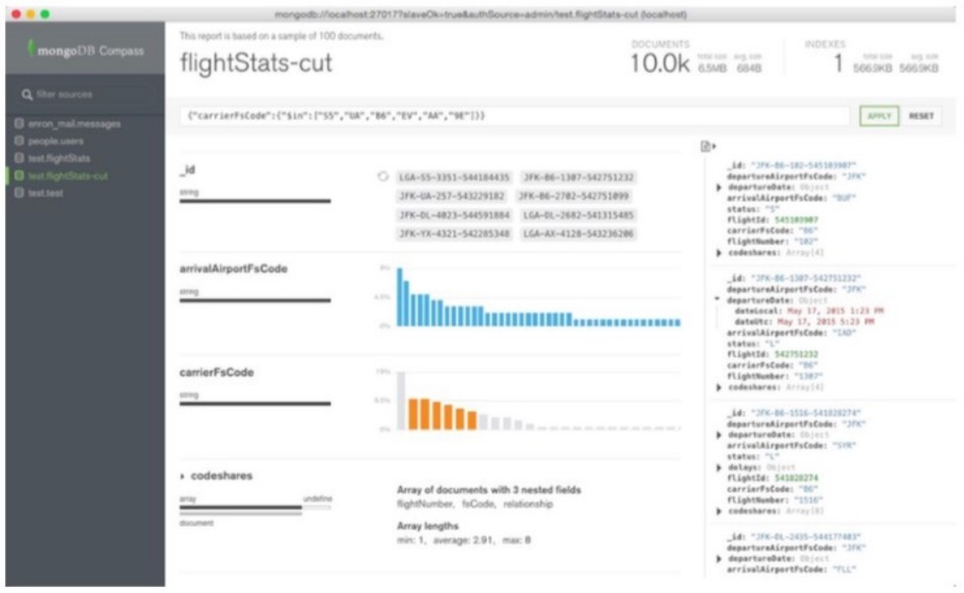
Driver

C, C++, C#, Java, Node.js, Perl, Php, Python, Ruby, Scala

supportati dalla community.....

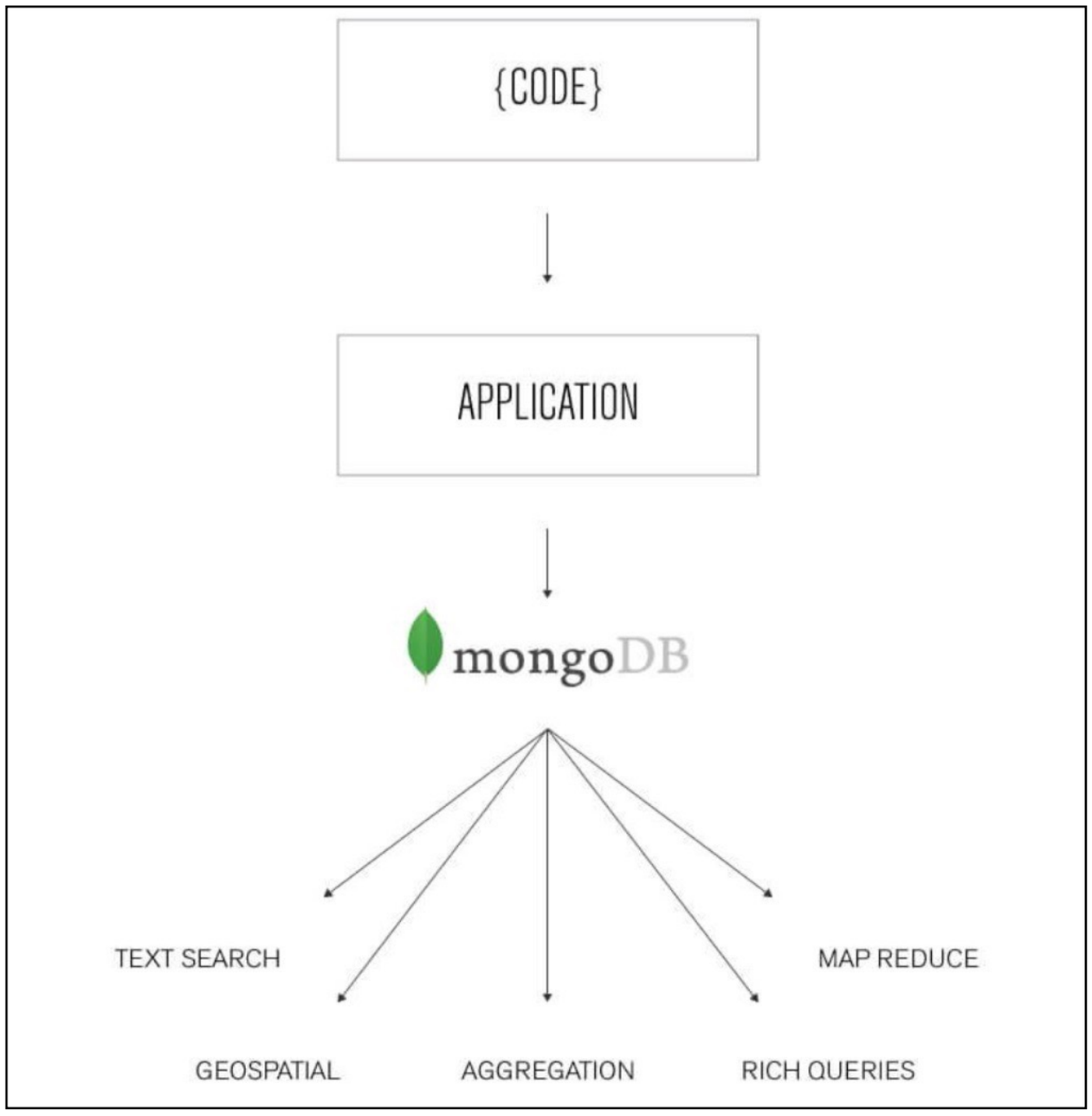
Go, Erlang, F#, R, ...

Compass

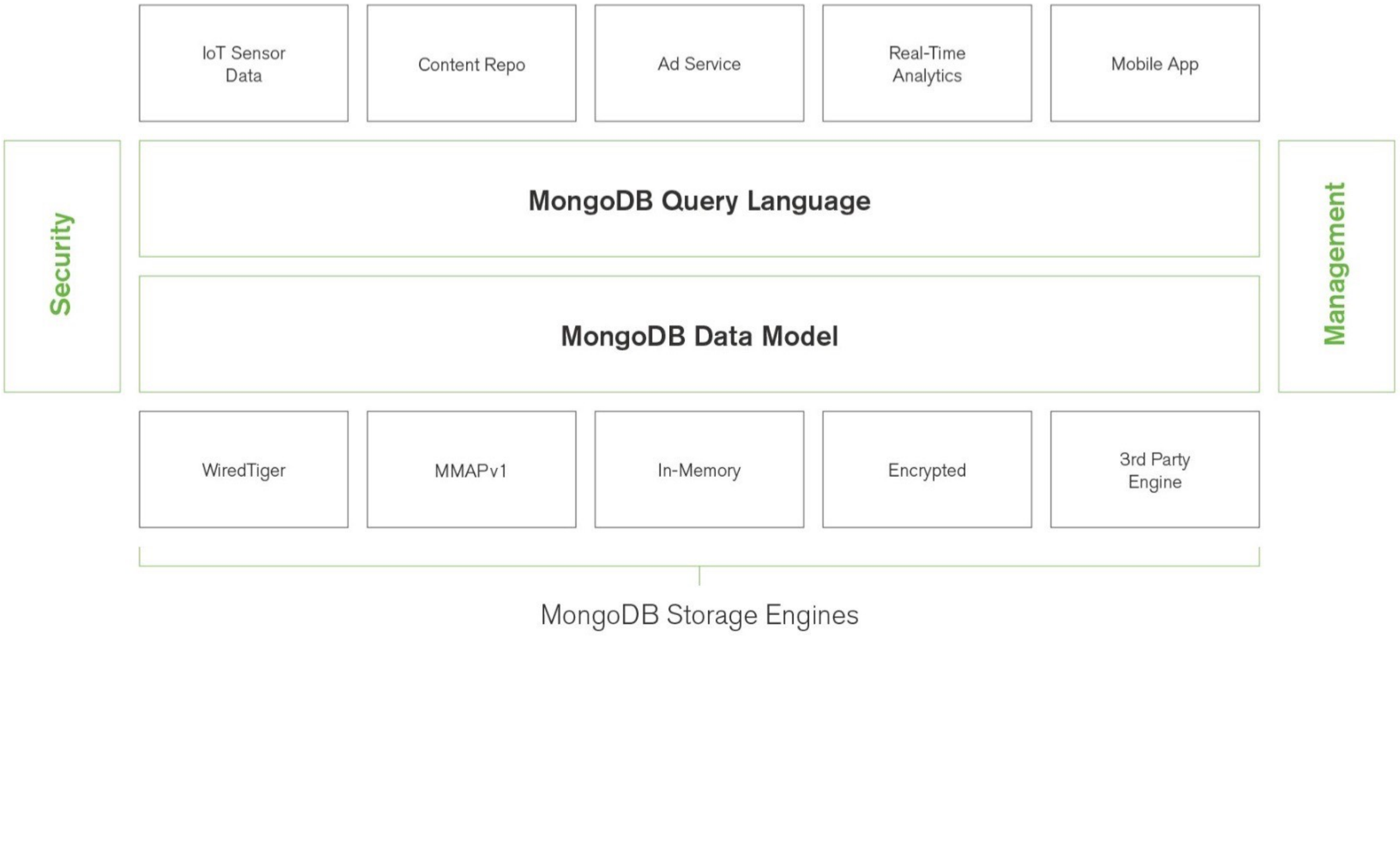


Query Model

- Key-value queries per ottenere il valore anche di un solo campo del documento
- Range queries (>, <, >=, <=)
- Geospatial queries per ottenere documenti col criterio della prossimità
- Text Search queries per ottenere documenti contenenti porzioni di testo
- Aggregation Framework queries (count, min, max, average, …..)
- MapReduce queries per eseguire operazioni di map e reduce direttamente sul database



Pluggable Storage Architecture



MongoDB: Modello a Documenti

PERSON

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rome

CAR

Car_ID	Model	Year	Value	Pers_ID
101	Bently	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2


```
{
  _id: ‘Objectid(“4b2b9…”)’,
  first_name: ‘Paul’,
  surname: ‘Miller’,
  city: ‘London’,
  location: [45.123,47.232],
  cars: [
    { model: ‘Bentley’,
      year: 1973,
      value: 100000, … },
    { model: ‘Rolls Royce’,
      year: 1965,
      value: 330000, … }
  ]
}
```

Documento

Esempio

```
{
  _id: ‘Objectid(“4b2b9…”)’,
  first_name: ‘Paul’,
  surname: ‘Miller’,
  city: ‘London’,
  location: [45.123,47.232],
  cars: [
    { model: ‘Bentley’,
      year: 1973,
      value: 100000, … },
    { model: ‘Rolls Royce’,
      year: 1965,
      value: 330000, … }
  ]
}
```

Tipi di dati

Null `{ x: null }`

Boolean `{ x: true }`

Number `{ x: 3.14 } { x: 3 }`

String `{ x: “zero12” }`

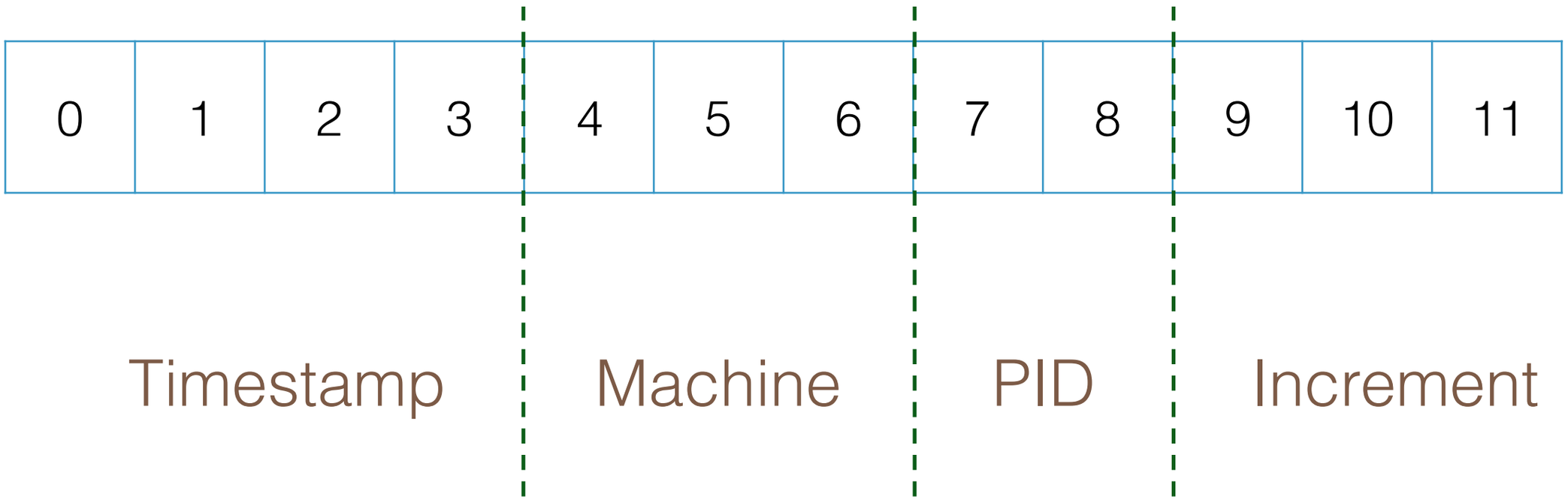
Date `{ x: new Date() }`

Array `{ x: [“a”,”b”, “c”] }`

Embedded documents

`{ x: {y: “a” } }`

Object id



Terminologia

Mongod:

E' il processo server che esegue il database e che rende disponibile tutti i servizi per la creazione di database, collections, l'inserimento di documenti e le relative funzioni di interrogazione

Mongos:

E' il servizio di routing per processare le query provenienti dallo strato applicativo e che determina la locazione dei dati richiesti in una configurazione Sharding.

Mongo:

Rappresenta l'eseguibile per avviare la shell Javascript per interagire con database, collezioni e documenti che possono essere interrogati e manipolati da riga di comando.

MongoDB shell

MongoDB dispone nativamente di una shell javascript per l'amministrazione del database

mongo 172.31.21.36:27017/firstDB

```
ubuntu@ip-172-31-21-36:~$ mongo 172.31.21.36:27017/firstDB
MongoDB shell version: 2.6.3
connecting to: 172.31.21.36:27017/firstDB
>
```

Principali Comandi

use <nome db>

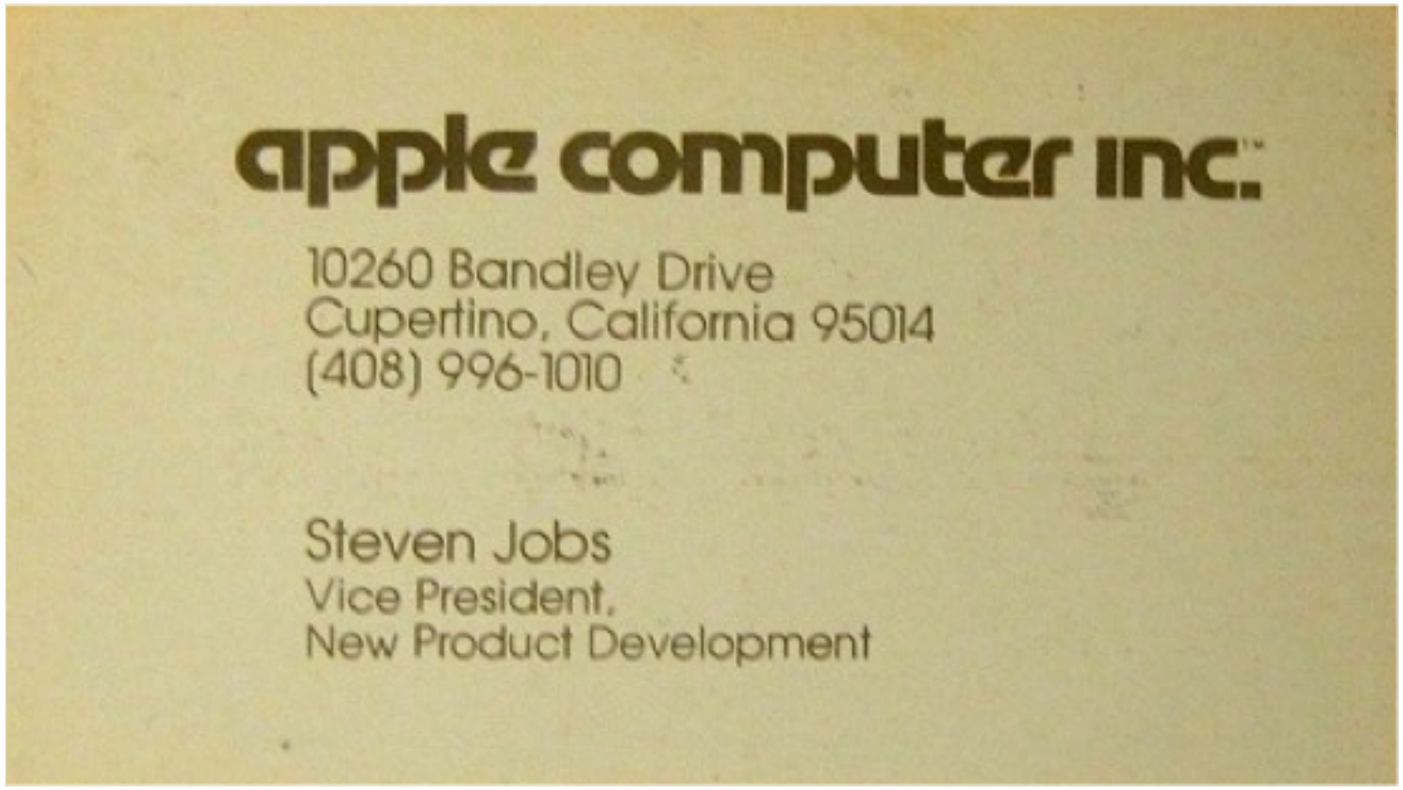
show collections

db.<nomeCollection>.insert({…})

db.<nomeCollection>.findOne()

db.<nomeCollection>.update(<criterio>, {…})

db.<nomeCollection>.remove(<criterio>, {…})


apple computer inc.
10260 Bandley Drive
Cupertino, California 95014
(408) 996-1010
Steven Jobs
Vice President,
New Product Development

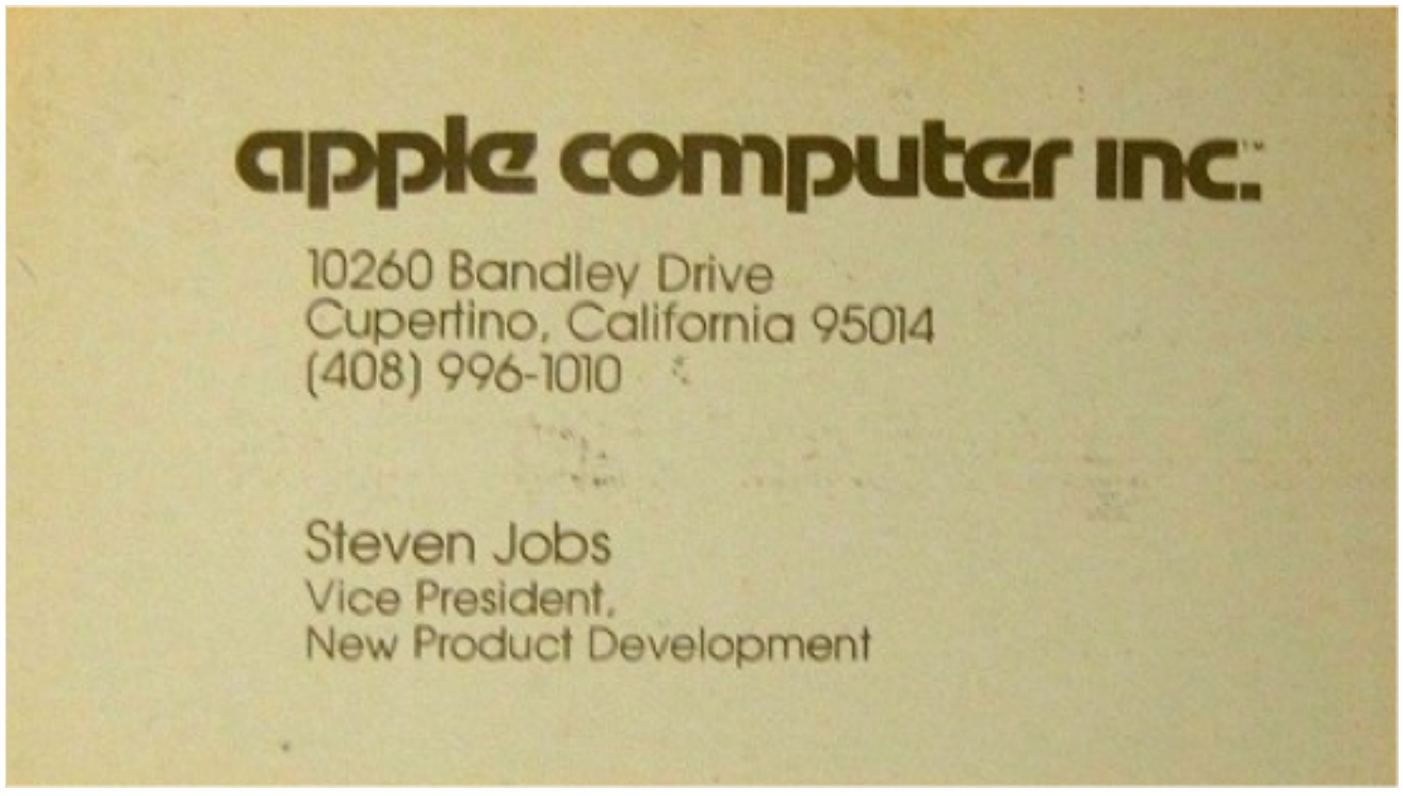
References

Contacts

```
{
    “_id”: 2,
    “name”: “Steven Jobs”,
    “title”: “VP, New Product Development”,
    “company”: “Apple Computer”,
    “phone”: “408-996-1010”,
    “address_id”: 1
}
```

Addresses

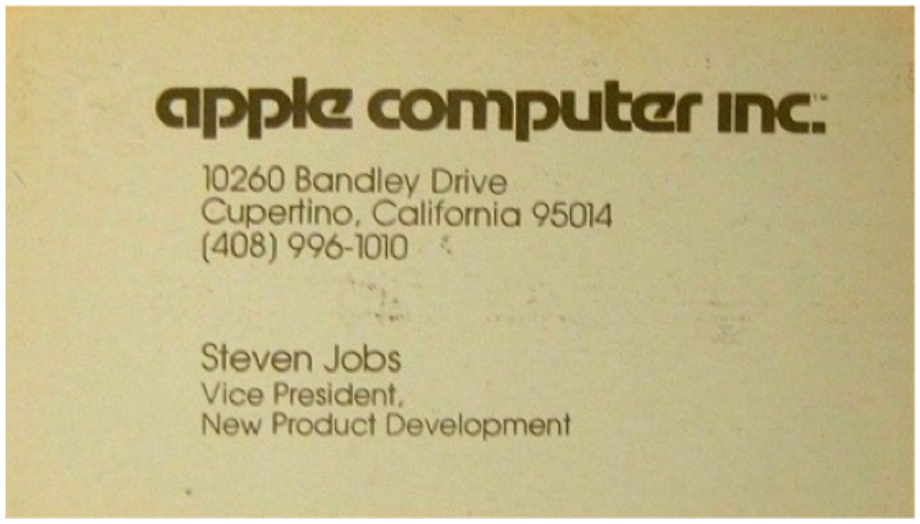
```
{
    “_id”: 1,
    “street”: “10260 Bandley Dr”,
    “city”: “Cupertino”,
    “state”: “CA”,
    “zip_code”: ”95014”,
    “country”: “USA”
}
```



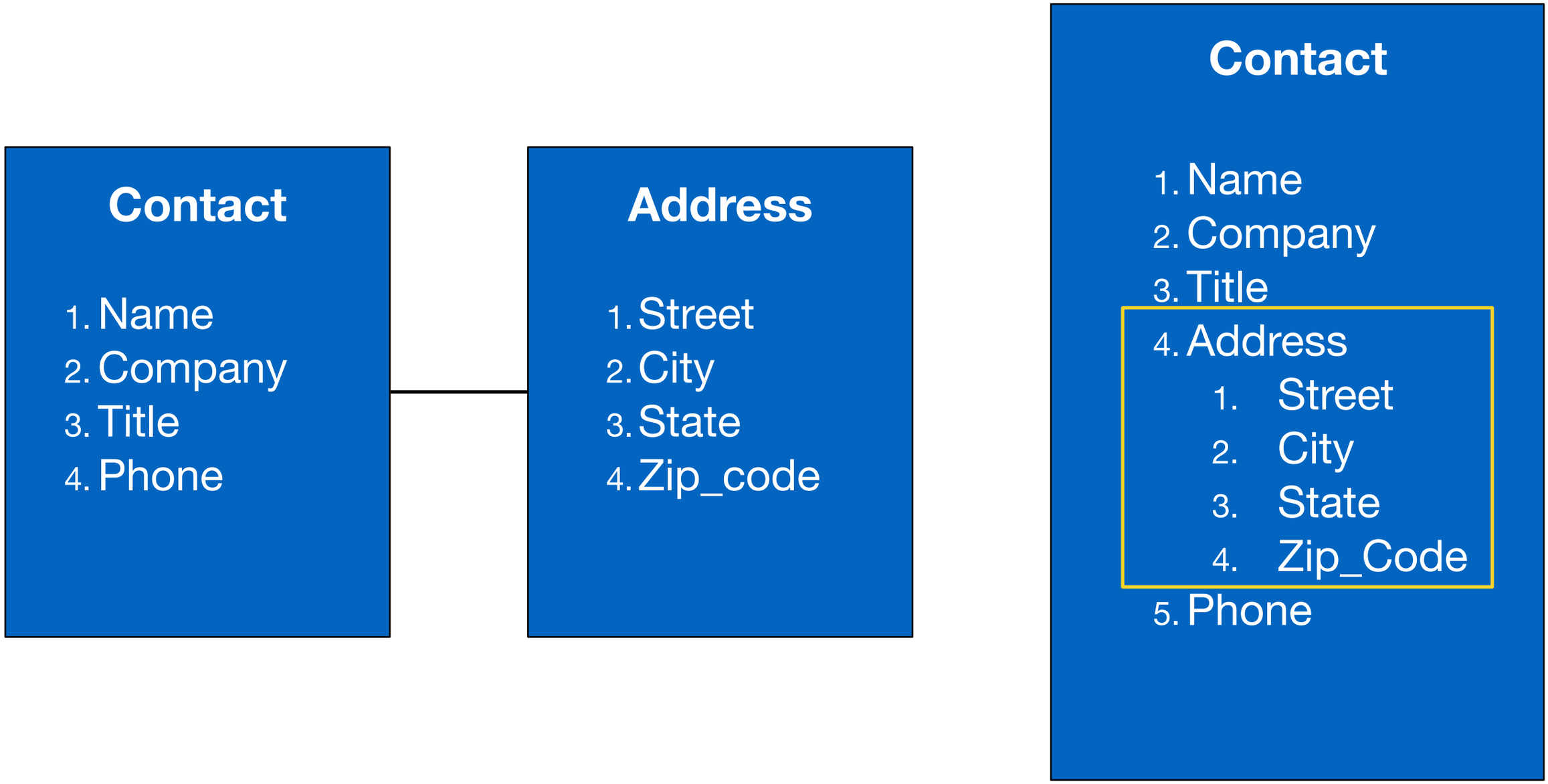
Embedding

Contacts

```
{
    “_id”: 2,
    “name”: “Steven Jobs”,
    “title”: “VP, New Product Development”,
    “company”: “Apple Computer”,
    “address”: {
     “street”: “10260 Bandley Dr”,
     “city”: “Cupertino”,
     “state”: “CA”,
     “zip_code”: ”95014”,
     “country”: “USA”
    },
    “phone”: “408-996-1010”
}
```



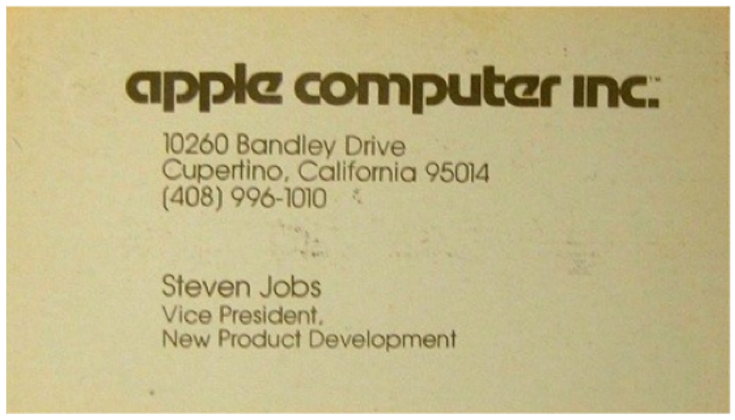
Relazionale vs Documents



Quali sono le differenze e perchè?

Flessibilità

```
{
    “name”: “Steven Jobs”,
    “title”: “VP, New Product Development”,
    “company”: “Apple Computer”,
    “address”: {
     “street”: “10260 Bandley Dr”,
     “city”: “Cupertino”,
     “state”: “CA”,
     “zip_code”: ”95014”
    },
    “phone”: “408-996-1010”
}
```




```
{
    “name”: “Larry Page”,
    “url”: “http://google.com/”,
    “title”: “CEO”,
    “company”: “Google!”,
    “email”: “larry@google.com”,
    “address”: {
        “street”: “555 Bryant, #106”,
        “city”: “Palo Alto”,
        “state”: “CA”,
        “zip_code”: “94301”
    }
    “phone”: “650-618-1499”,
    “fax”: “650-330-0100”
}
```

Schema Design

1:1

1:N

N:N

Embedding è meglio ! :-)

Embedding over Referencing

1. Eseguire l'Embedding è come avere delle Join preparate:
    1. BSON document sono facili da gestire per i server
2. Embedd:
    1. Quando "pochi" o "molti" oggetti sono visualizzati insieme al padre
    2. Per motivi di performance
    3. Per atomicità delle informazioni
3. Reference
    1. Quando si necessità di maggiore scalabilità
4. Per maggiore "consistenza" nella associazioni N:N senza duplicare i dati

Architetture

Perchè replica dei dati

- Quanto punti di failure vuoi avere?
- Quanti sonni tranquilli vuoi dormire?
- Mai avuto problemi di connettività?
- Più nodi facilitano l’uso dei dati in modo diverso

Replica Set

Creazione ReplicaSet

Nodo 1

Nodo 2

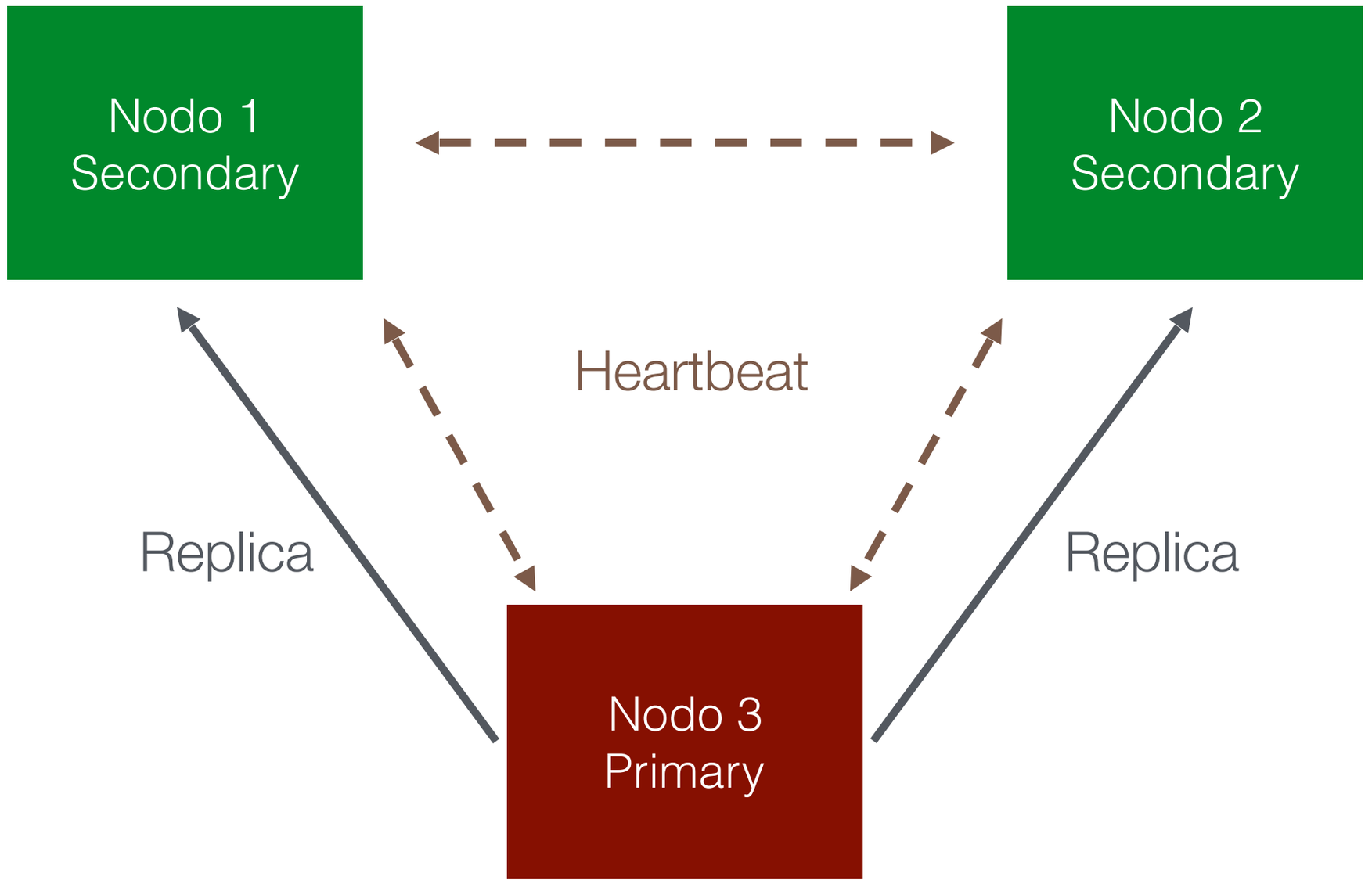
Nodo 3

Configurazione ReplicaSet - 2 di 2

```
> conf = {
    _id : "zero12rs",
    members : [
        {_id : 0, host : "A"},
        {_id : 1, host : "B"},
        {_id : 2, host : "C"},
    ]
}

> rs.initiate(conf)
```

ReplicaSet - 1 di 3



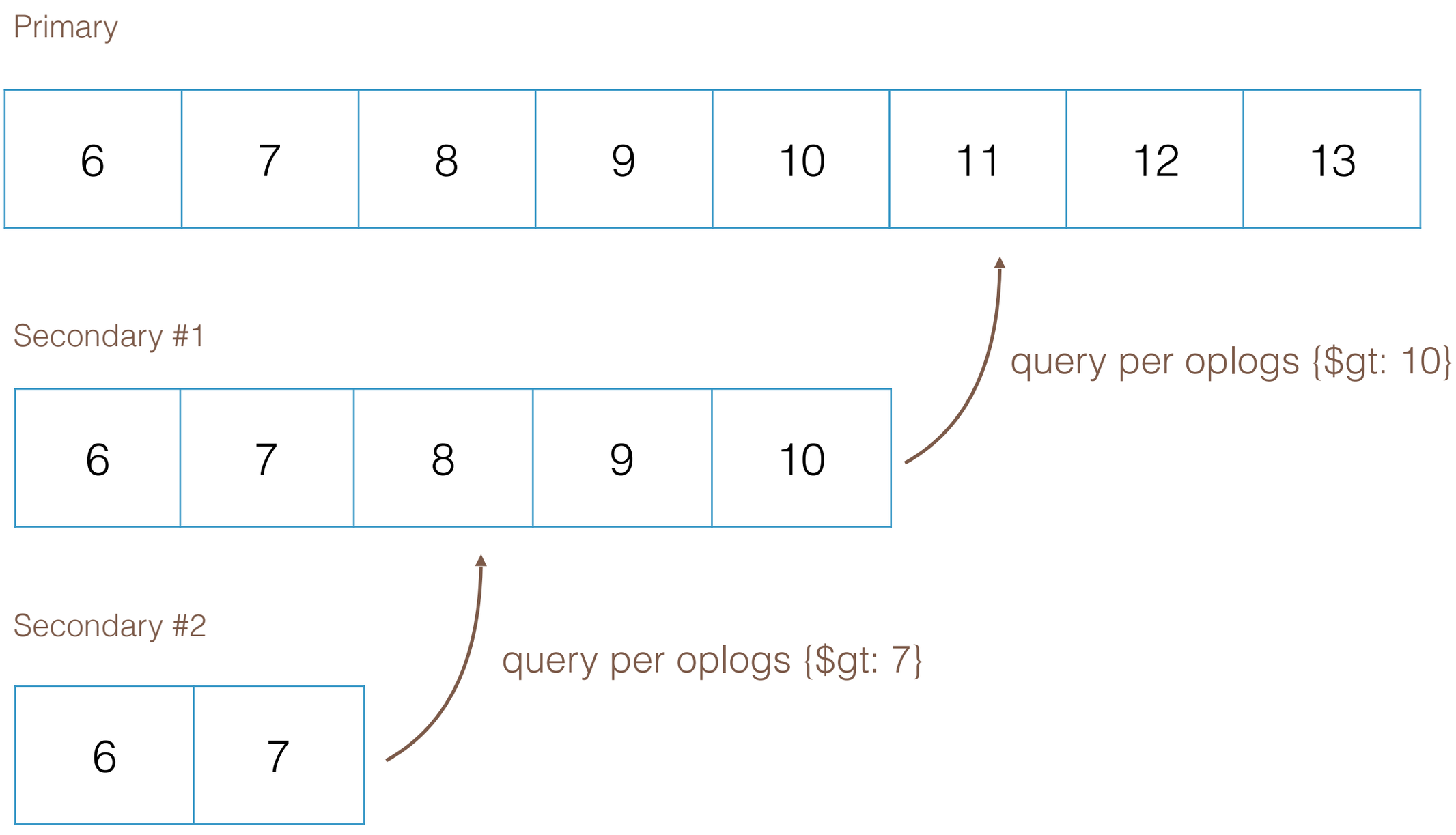
ReplicaSet - 2 di 3

ReplicaSet - 3 di 3

Come funziona la replica

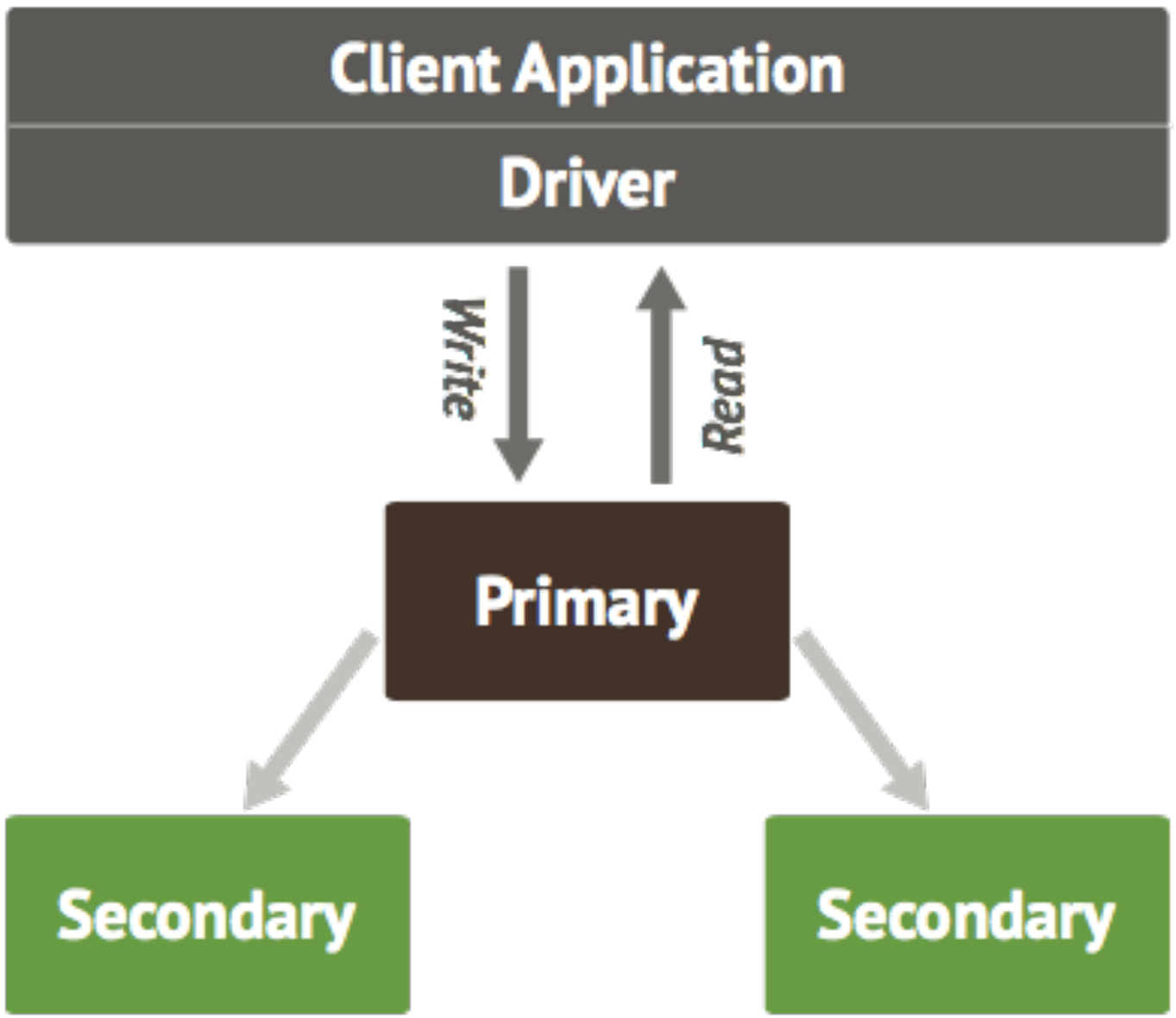
Come funziona la replica ?

oplog: contiene la lista di tutte le operazioni di scrittura e aggiornamenti fatti nel nodo primario

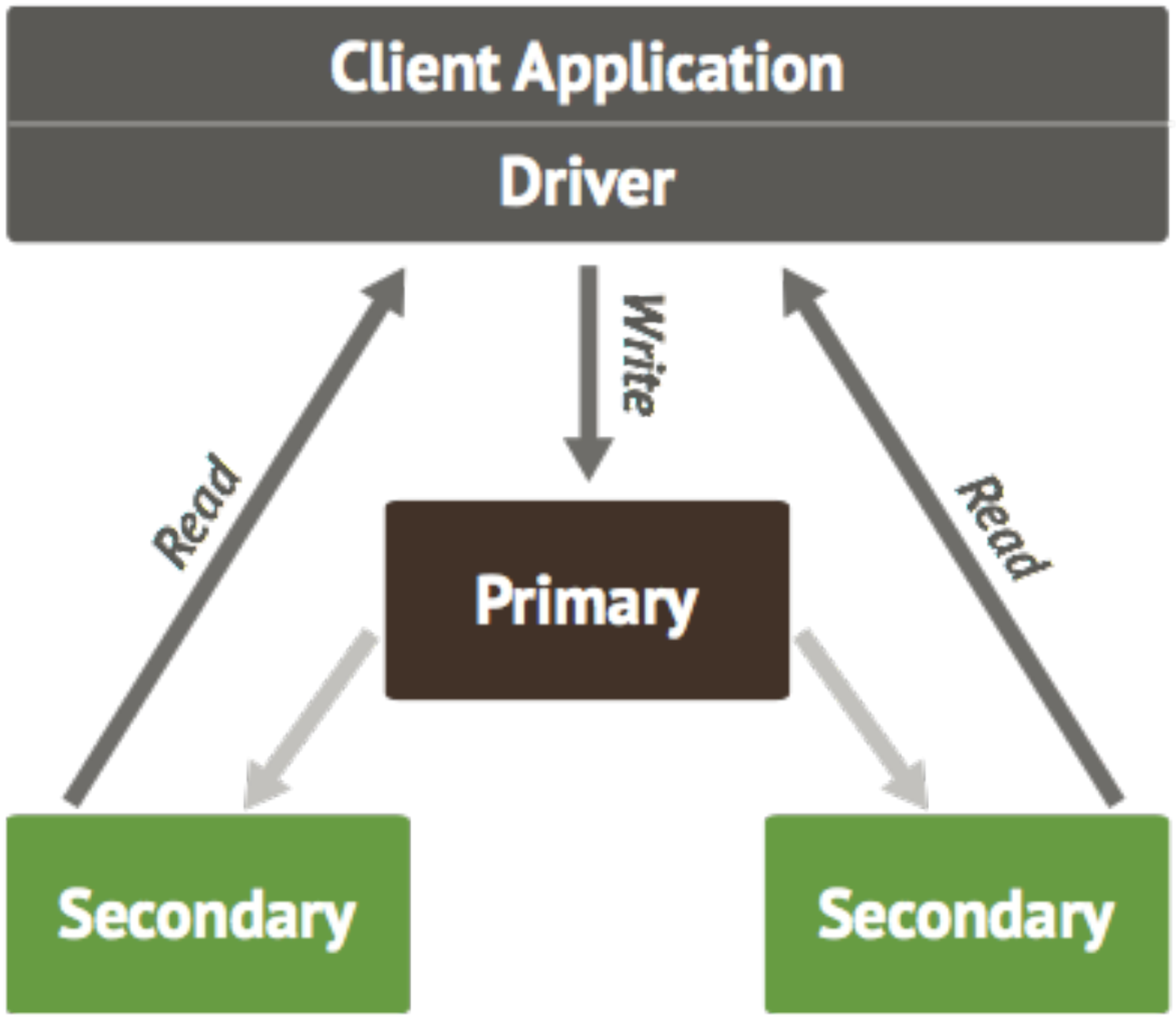


Sviluppare con replicaSet

ReplicaSet per gli sviluppatori



ReplicaSet per gli sviluppatori



Tagging

- Controllare dove i dati vengono scritti e/o letti
- Ogni membro del RS può avere uno o più TAG
  - tags: {dc: “ny”}
  - tags: {dc: “ny”, rack: “zero12rack”}
- Permette di definire le regole di Write Concerns
- E’ possibile cambiare le politiche senza modificare il codice

Esempio di Tagging

```
{
    _id : "zero12rs",
    members : [
        {_id : 0, host : "A", tags : {"dc": "ny"}},
        {_id : 1, host : "B", tags : {"dc": "ny"}},
        {_id : 2, host : "C", tags : {"dc": "sf"}},
        {_id : 3, host : "D", tags : {"dc": "sf"}},
        {_id : 4, host : "E", tags : {"dc": "cloud"}}],
    settings : {
        getLastErrorModes : {
            allDCs :  {"dc" : 3},
            someDCs : {"dc" : 2}} }
}
> db.blogs.insert({...})
> db.runCommand({getLastError : 1, w : "someDCs"})
```

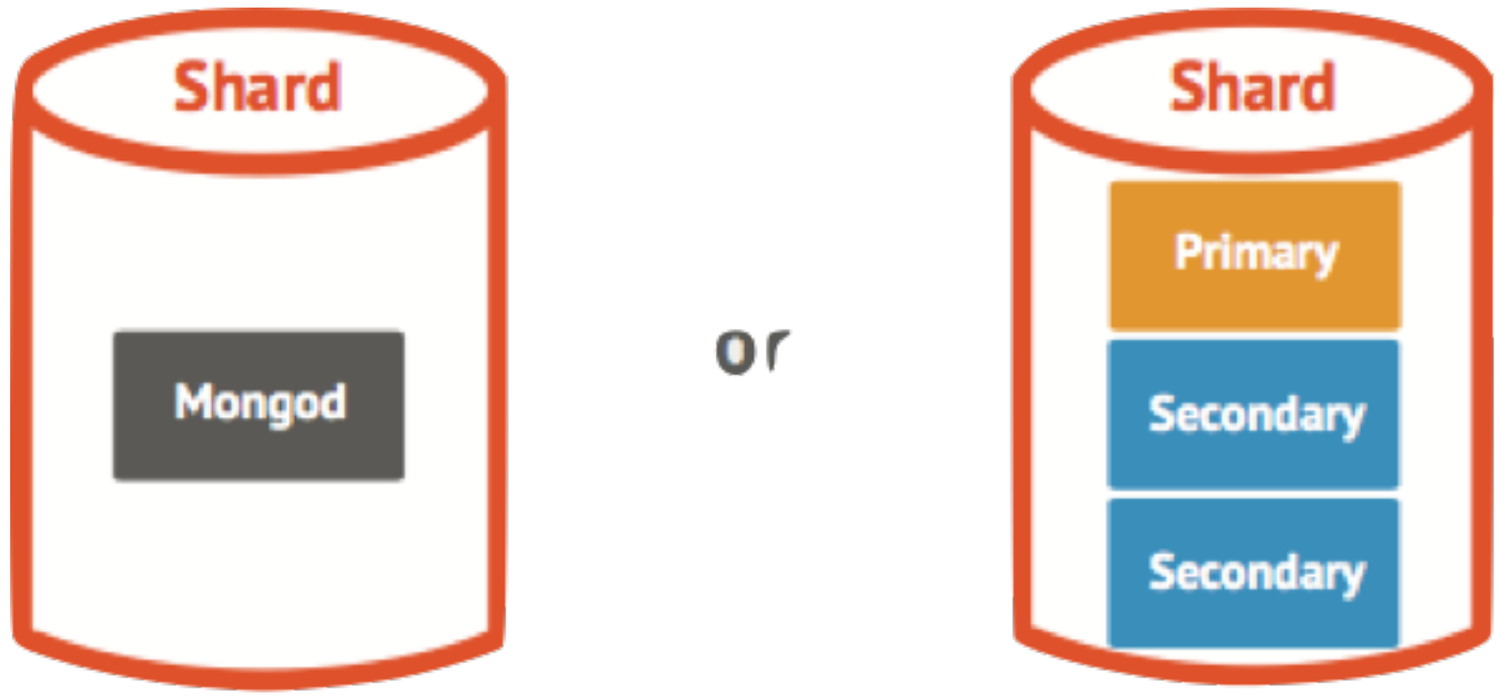
Slave Delay

```
> conf = {
    _id : "mySet”,
    members : [
        {_id : 0, host : "A”},
        {_id : 1, host : "B"},
        {_id : 2, host : "C”},
        {_id : 3, host : "D", hidden : true},
        {_id : 4, host : "E", hidden : true, slaveDelay : 3600}
    ]
}

> rs.initiate(conf)
```

Sharding

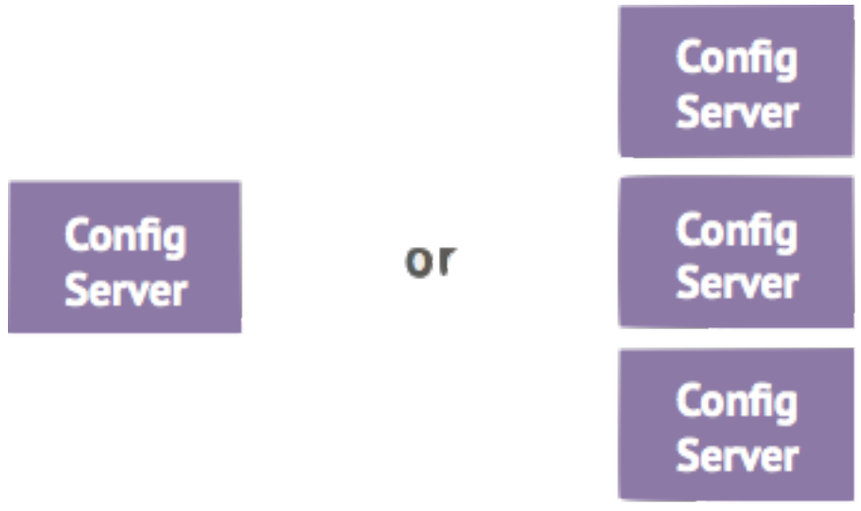
Distribuzione dei dati
(Sharding)



- Uno Shard è un nodo di un cluster
- Uno Shard può essere un singolo mongod oppure un Replica Set

Config Server

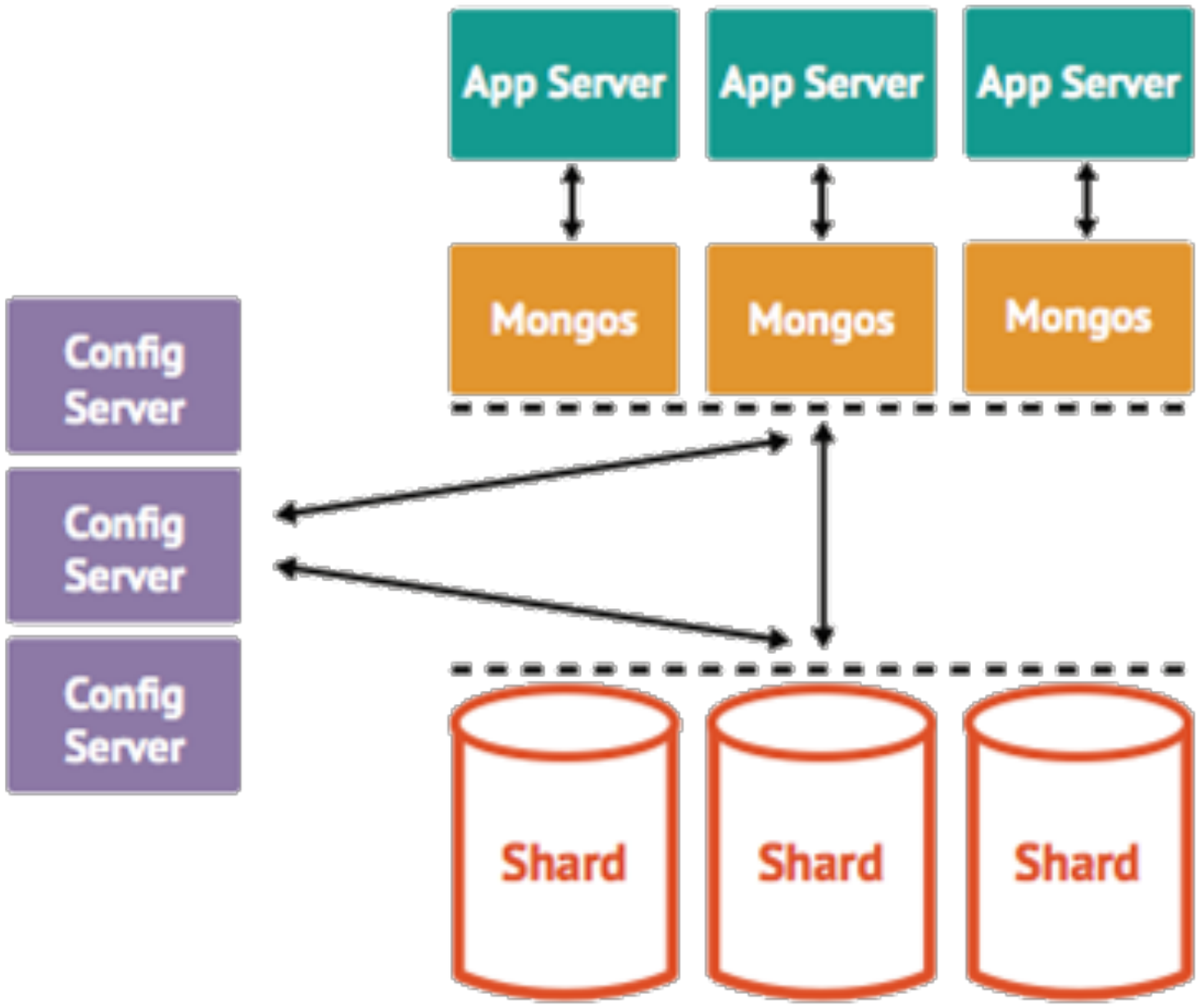
- Conserva il range di chunck e la loro localizzazione
- In un’architettura possono esserci 1 o 3 nodi Config Server
- Non possono essere configurati come Replica-Set



Mongos

- Funge da router / balancer
- Non ci sono dati presenti localmente (Informazioni utili sono conservati nel config server)
- Un’architettura può essere composta da 1 o più Mongos

Esempio architettura Sharding



Sharding Key

Sharding Key

- La chiave di Shard è immutabile
- I valori della chiave di Shard sono immutabile
- La chiave di Shard deve essere indicizzata
- La chiave di Shard è limitata ad una dimensione di 512 bytes
- La chiave di Shard è usata per definire il routing delle query

  - Usare un campo usato nelle query
- Solamente la shard key può essere unica tra tutti gli shards

  - `_id` è unico all'interno del singolo shard

Impatti della sharding key sullo sharding

La chiave di sharding determina la distribuzione dei documenti di una collezione nei diversi nodi di shard

- Scaling delle scritture
- Query Isolation
- Sorting
- Localizzazione degli indici

Funzionamento

Partitioning

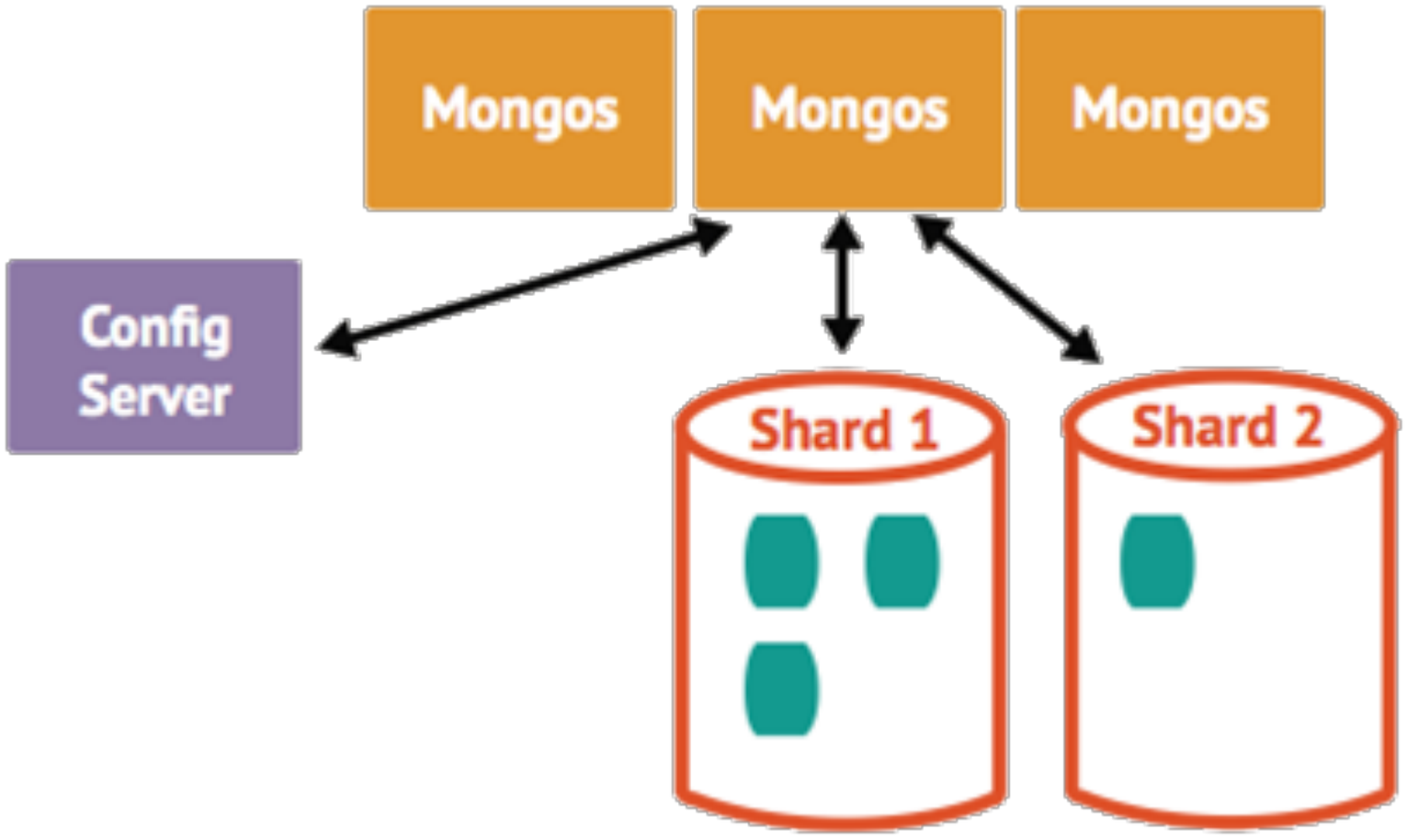
E’ importante ricordare che il partizionamento degli oggetti
è basato su un range

I chunk sono una partizione di questo range

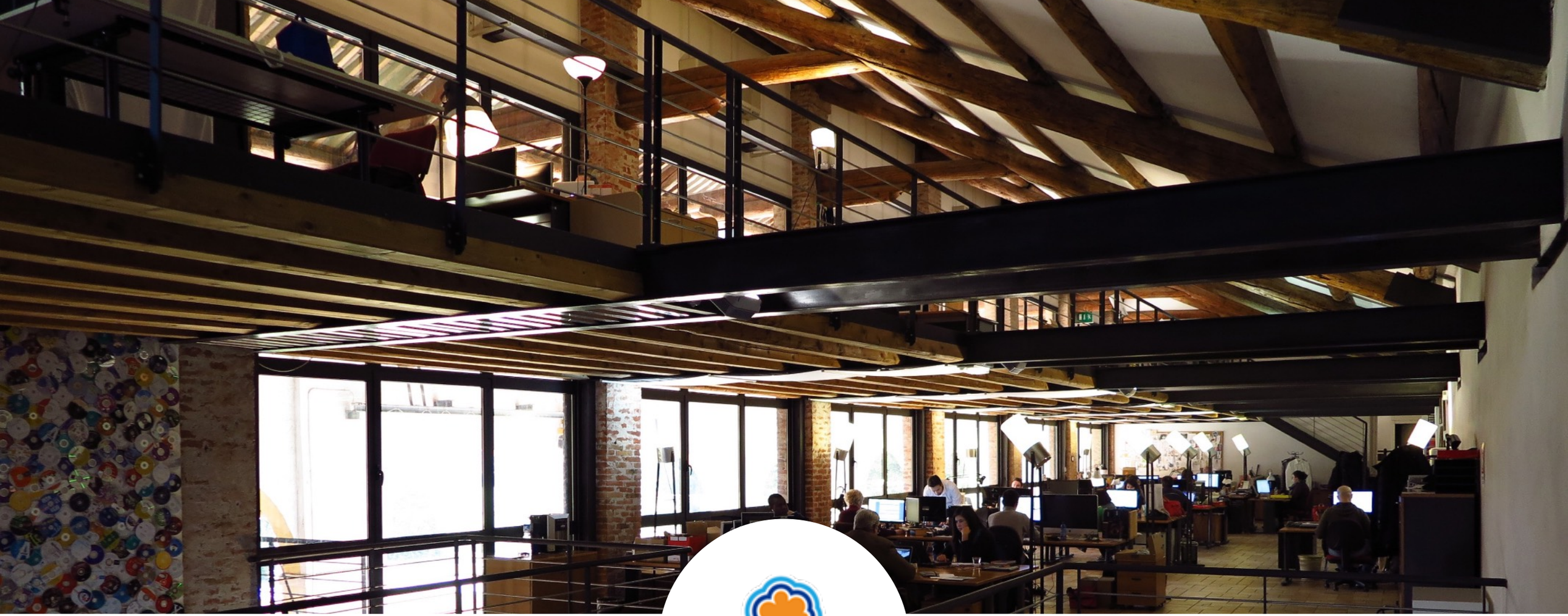
Divisione Chunk

- Un chunk viene suddiviso quando supera la dimensione massima
- Non ci sono punti di divisione se i documenti hanno la stessa shared key
- Lo split di un chunk è un’operazione logica (nessun dato viene spostato)

Balancing



- Il Balancer è eseguito all'interno di Mongos
- Quando la differenza tra lo shard più denso e la densità degli altri shard supera la soglia di migrazione, il processo di bilanciamento viene avviato



GRAZIE !

“Learn quickly and Think Well!”