Designing Scalable and User-Intensive Enterprise Applications

Date (02 November 2022)
Suresh Kumar Mukhiya, PhD, Tryg Norge (www.skmukhiya.com.np)
Guest Lecture UIB

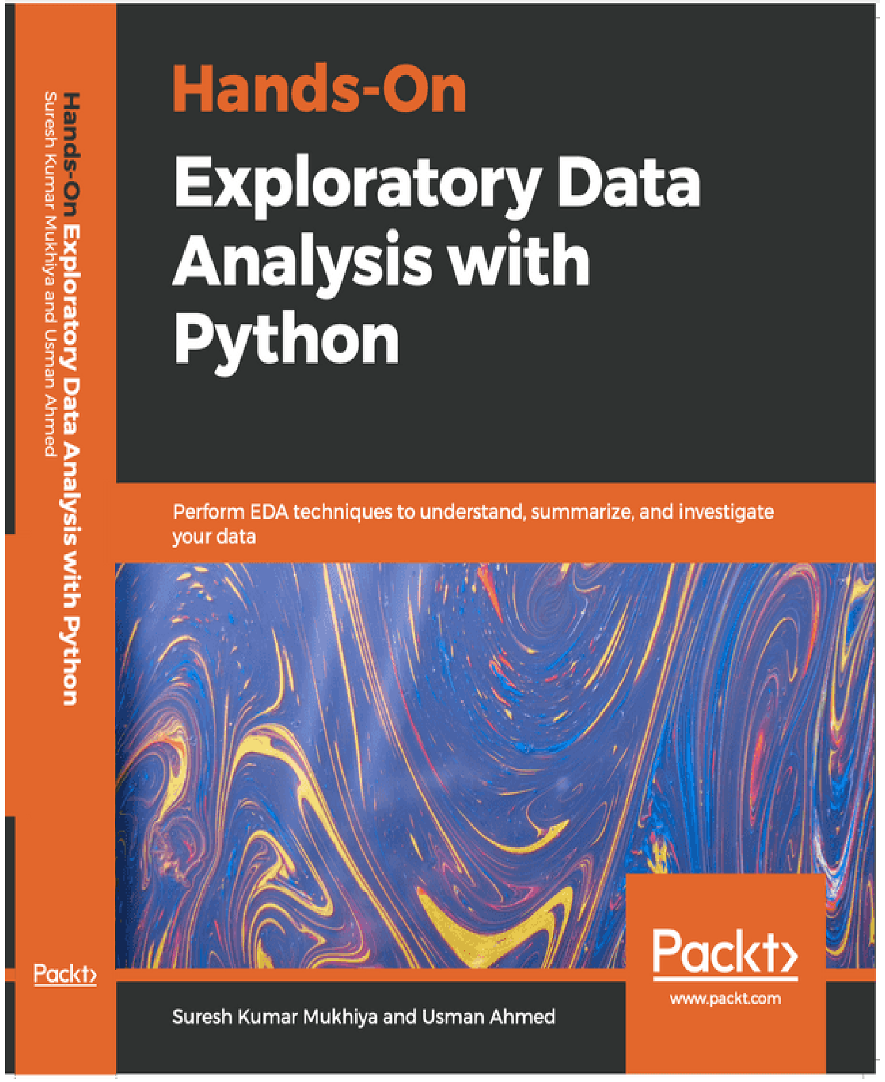
Agenda

1. Overview (User-intensive, Enterprise Application, Scalable Systems, Client, Server, DNS)
2. Single Server Setup
3. Database in Single Server Setup
4. Which Database to use?
5. Scalability
6. Load balancer
7. Database replication
8. Cache Tier
9. Content Delivery Network (CDN)
10. Data Centers
11. Take away

About Me

- From Nepal
- 2014: Moved to Norway
- 2016: Masters in Information System (NTNU, Trondheim)
- 2021: PhD in Software Engineering, HVL, Bergen
- Married, 4 years old daughter
- Started as full stack developer since 2010, almost 12 years in IT
- I like photography, fishing and writing books
- **Major focus:** Software Architecture, Frontend Technologies, and Data Science.

Published books



Hands-On Exploratory Data Analysis with Python

Suresh Kumar Mukhiya and Usman Ahmed

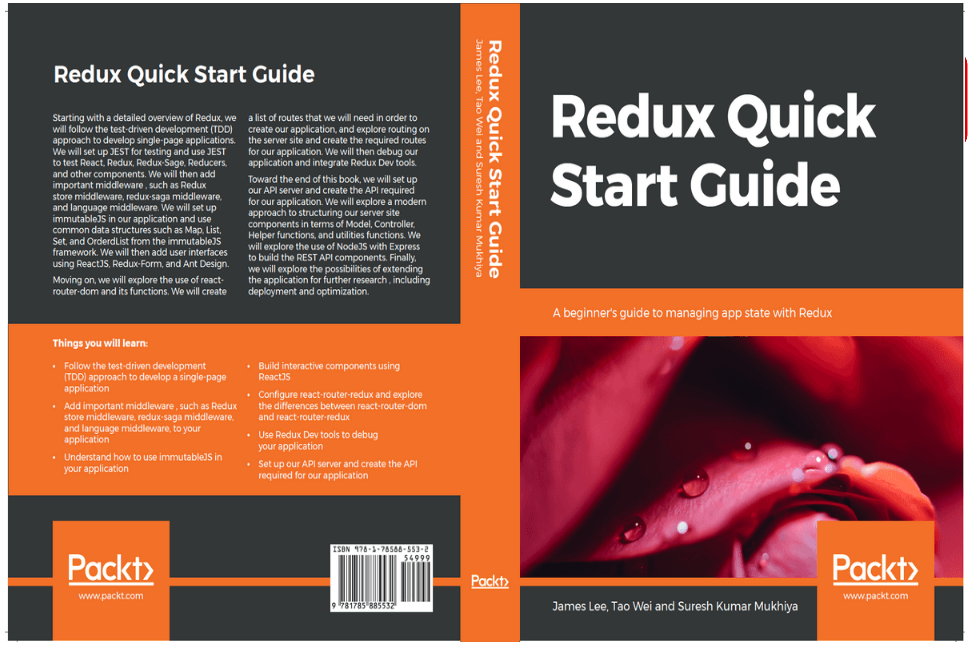
Hands-On Exploratory Data Analysis with Python

Perform EDA techniques to understand, summarize, and investigate your data

Packt

www.packt.com

Suresh Kumar Mukhiya and Usman Ahmed



Redux Quick Start Guide

Starting with a detailed overview of Redux, we will follow the test-driven development (TDD) approach to develop single-page applications. We will set up JEST for testing and use JEST to test React, Redux, Redux-Sage, Reducers, and other components. We will then add important middleware , such as Redux store middleware, redux-saga middleware, and language middleware. We will set up immutableJS in our application and use common data structures such as Map, List, Set, and OrderdList from the immutableJS framework. We will then add user interfaces using ReactJS, Redux-Form, and Ant Design.

Moving on, we will explore the use of react-router-dom and its functions. We will create a list of routes that we will need in order to create our application, and explore routing on the server site and create the required routes for our application. We will then debug our application and integrate Redux Dev tools.

Toward the end of this book, we will set up our API server and create the API required for our application. We will explore a modern approach to structuring our server site components in terms of Model, Controller, Helper functions, and utilities functions. We will explore the use of NodeJS with Express to build the REST API components. Finally, we will explore the possibilities of extending the application for further research , including deployment and optimization.

Things you will learn:

- Follow the test-driven development (TDD) approach to develop a single-page application
- Add important middleware , such as Redux store middleware, redux-saga middleware, and language middleware, to your application
- Understand how to use immutableJS in your application
- Build interactive components using ReactJS
- Configure react-router-redux and explore the differences between react-router-dom and react-router-redux
- Use Redux Dev tools to debug your application
- Set up our API server and create the API required for our application

Packt

www.packt.com

ISBN 978-1-78588-553-2

Redux Quick Start Guide

James Lee, Tao Wei and Suresh Kumar Mukhiya

Packt

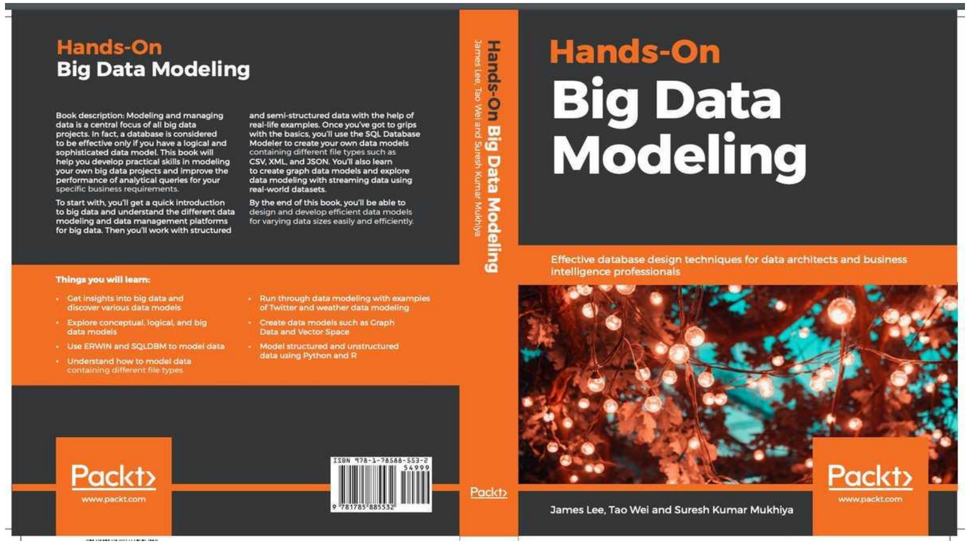
Redux Quick Start Guide

A beginner's guide to managing app state with Redux

James Lee, Tao Wei and Suresh Kumar Mukhiya

Packt

www.packt.com



Hands-On Big Data Modeling

Book description: Modeling and managing data is a central focus of all big data projects. In fact, a database is considered to be effective only if you have a logical and sophisticated data model. This book will help you develop practical skills in modeling your own big data projects and improve the performance of analytical queries for your specific business requirements.

To start with, you'll get a quick introduction to big data and understand the different data modeling and data management platforms for big data. Then you'll work with structured and semi-structured data with the help of real-life examples. Once you've got to grips with the basics, you'll use the SQL Database Modeler to create your own data models containing different file types such as CSV, XML, and JSON. You'll also learn to create graph data models and explore data modeling with streaming data using real-world datasets.

By the end of this book, you'll be able to design and develop efficient data models for varying data sizes easily and efficiently.

Things you will learn:

- Get insights into big data and discover various data models
- Explore conceptual, logical, and big data models
- Use ERWIN and SQLDBM to model data
- Understand how to model data containing different file types
- Run through data modeling with examples of Twitter and weather data modeling
- Create data models such as Graph Data and Vector Space
- Model structured and unstructured data using Python and R

Packt

www.packt.com

ISBN 978-1-78588-553-2

Hands-On Big Data Modeling

James Lee, Tao Wei and Suresh Kumar Mukhiya

Packt

Hands-On Big Data Modeling

Effective database design techniques for data architects and business intelligence professionals

James Lee, Tao Wei and Suresh Kumar Mukhiya

Packt

www.packt.com

Overview

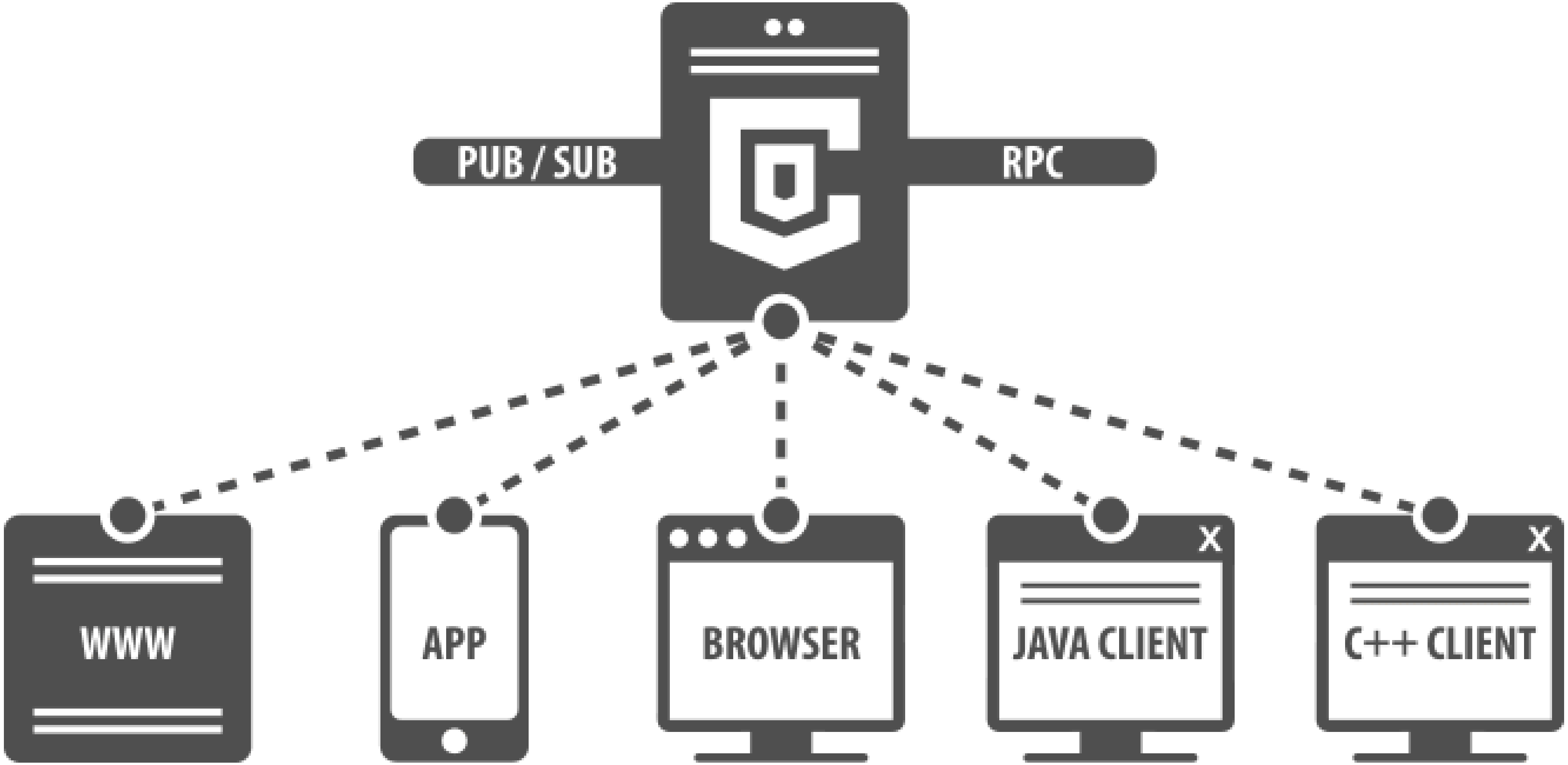
- **User-Intensive Application:** can handle zero to million users. Example: Facebook, Netflix

- **Enterprise Application:** An enterprise application (EA) is a large software system platform designed to operate in a corporate environment

- **Scalable System:** An application that can handle a growing number of users and load, without compromising on performance

Kahoot

Overview – Client/ Server



Overview – Domain Name Server (DNS)

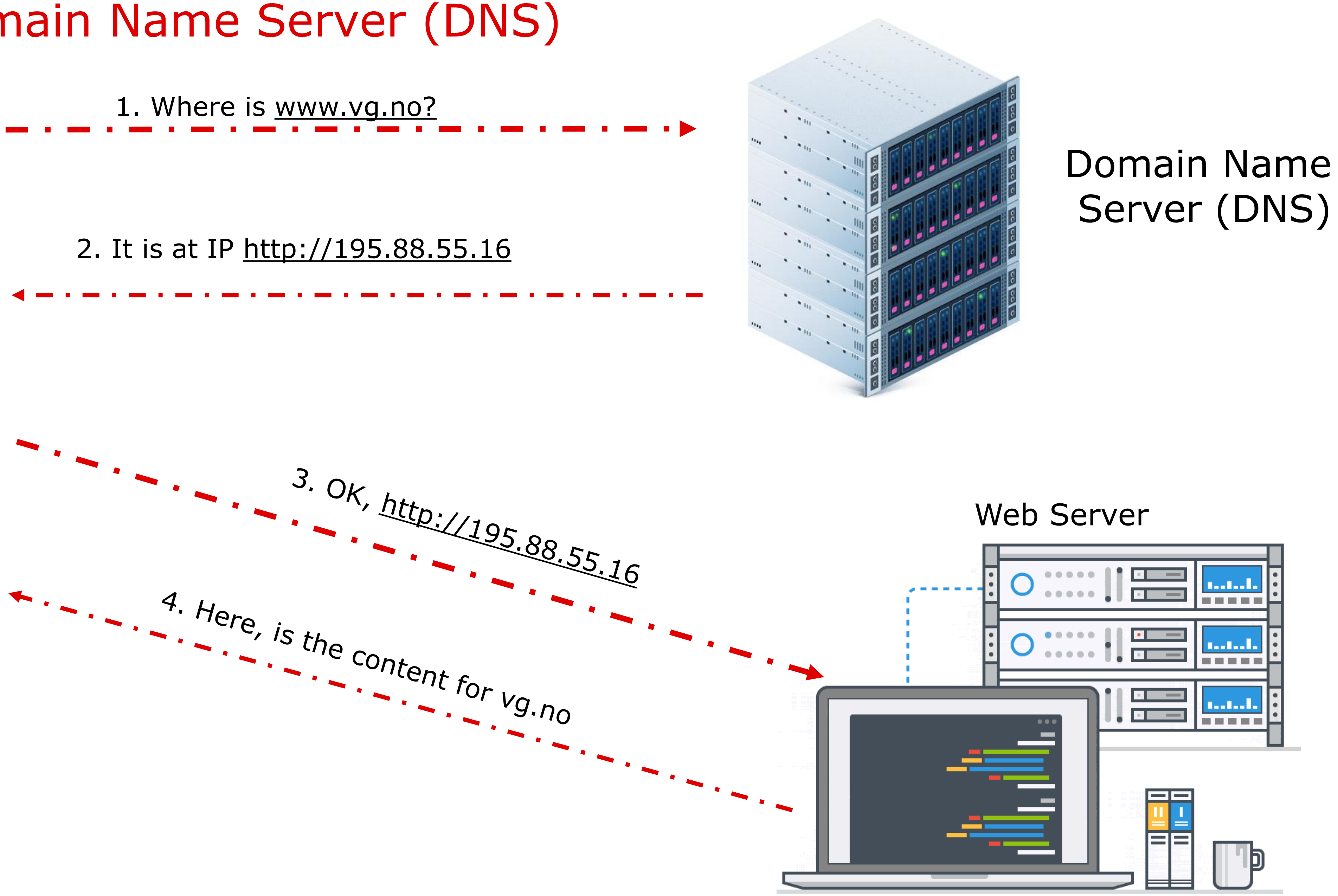
Schibsted er ansvarlig for dine data på denne siden. Les mer her

Beskjeden til Viggo Kristiansen:

- Hvis du er skyldig, da blir du tatt

01:19

vg.no



1. Where is www.vg.no?

2. It is at IP http://195.88.55.16

3. OK, http://195.88.55.16

4. Here, is the content for vg.no

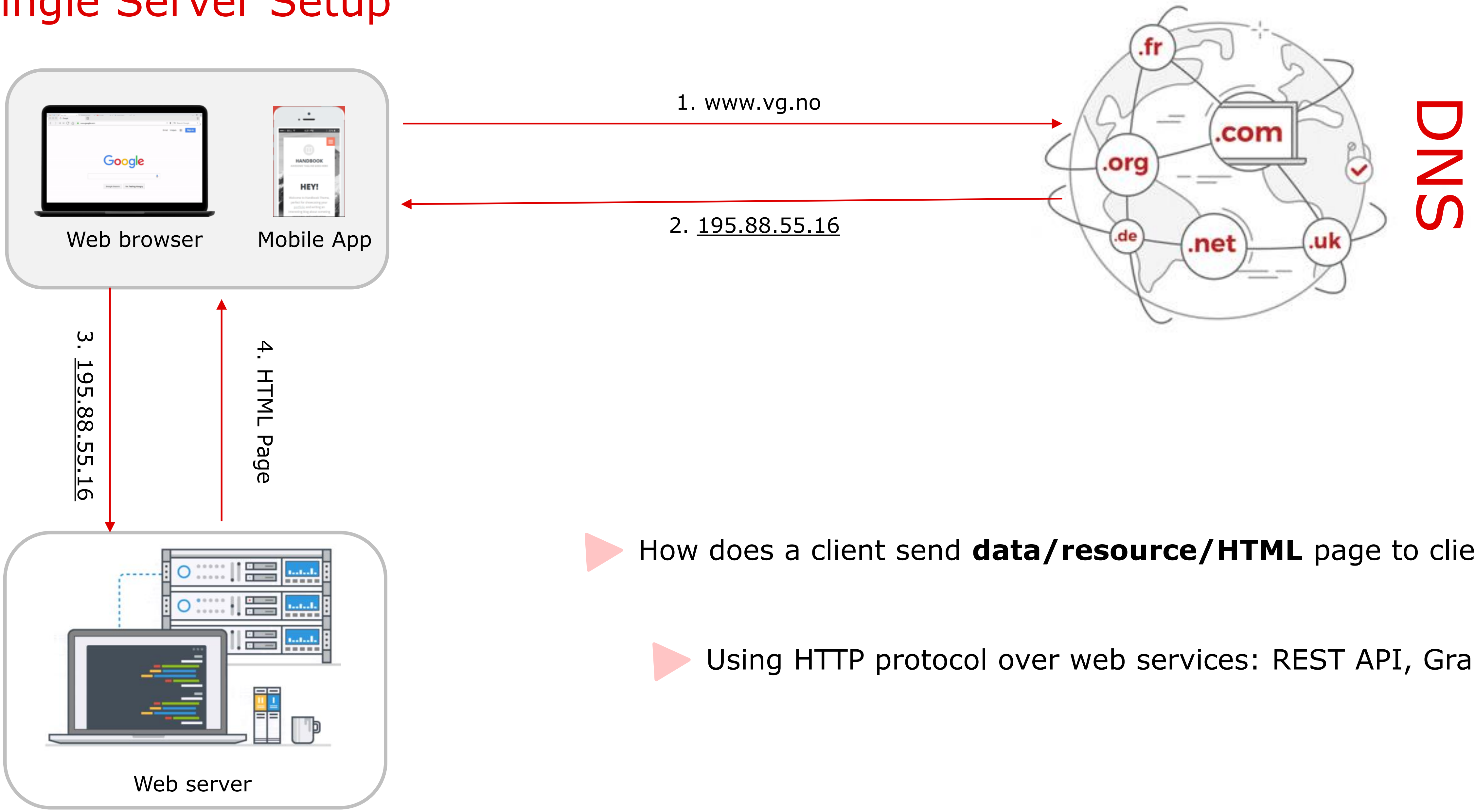
Domain Name Server (DNS)

Web Server

A journey of thousand miles begins with a single step.

Let us build an architecture for small group of user.

Single Server Setup



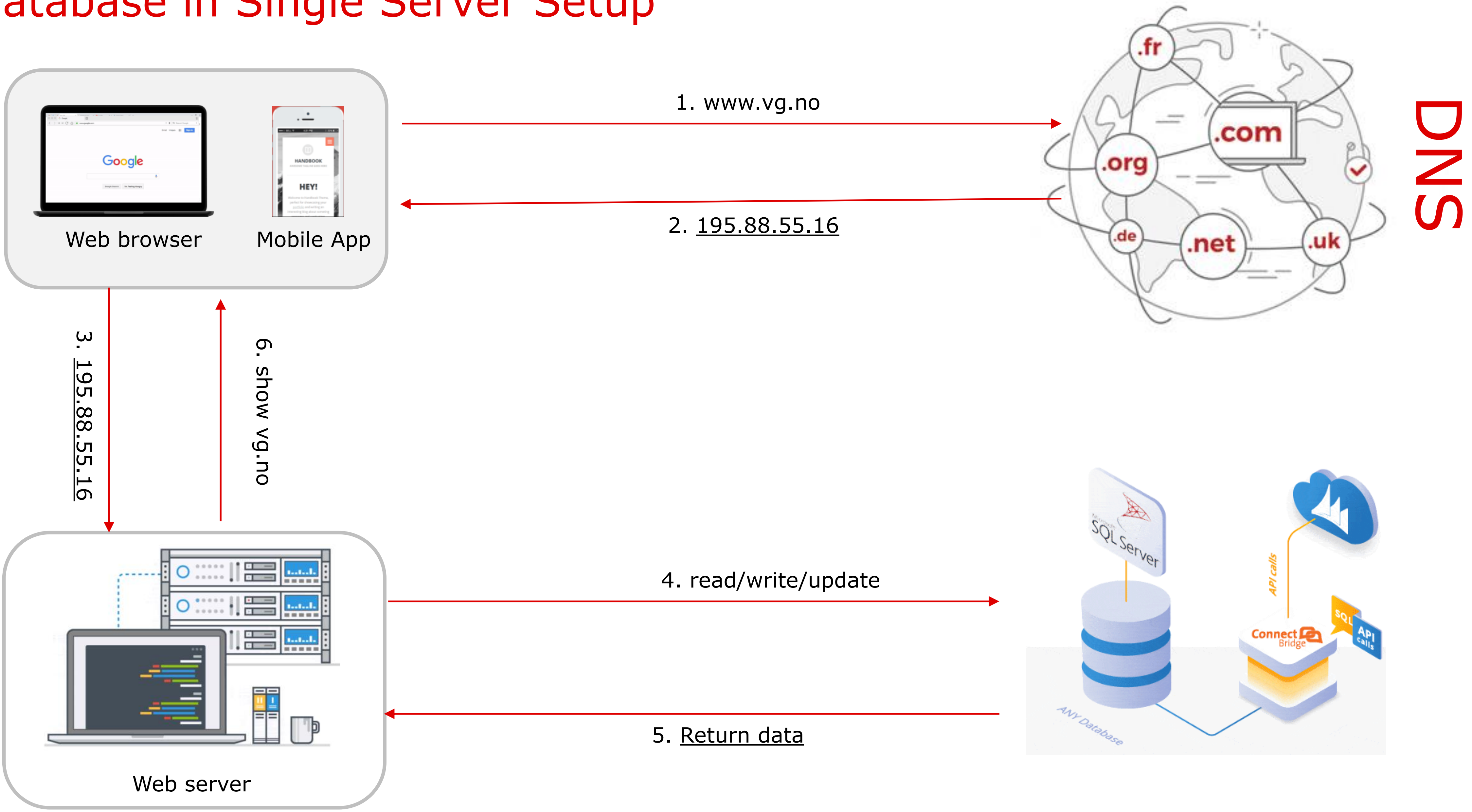
- How does a client send **data/resource/HTML** page to client?
- Using HTTP protocol over web services: REST API, GraphQL

How does any server stores information?

- Database

- Let us include DATABASE in the single server setup.

Database in Single Server Setup



Web browser

Mobile App

1. www.vg.no

2. 195.88.55.16

DNS

3. 195.88.55.16

6. show vg.no

4. read/write/update

5. Return data

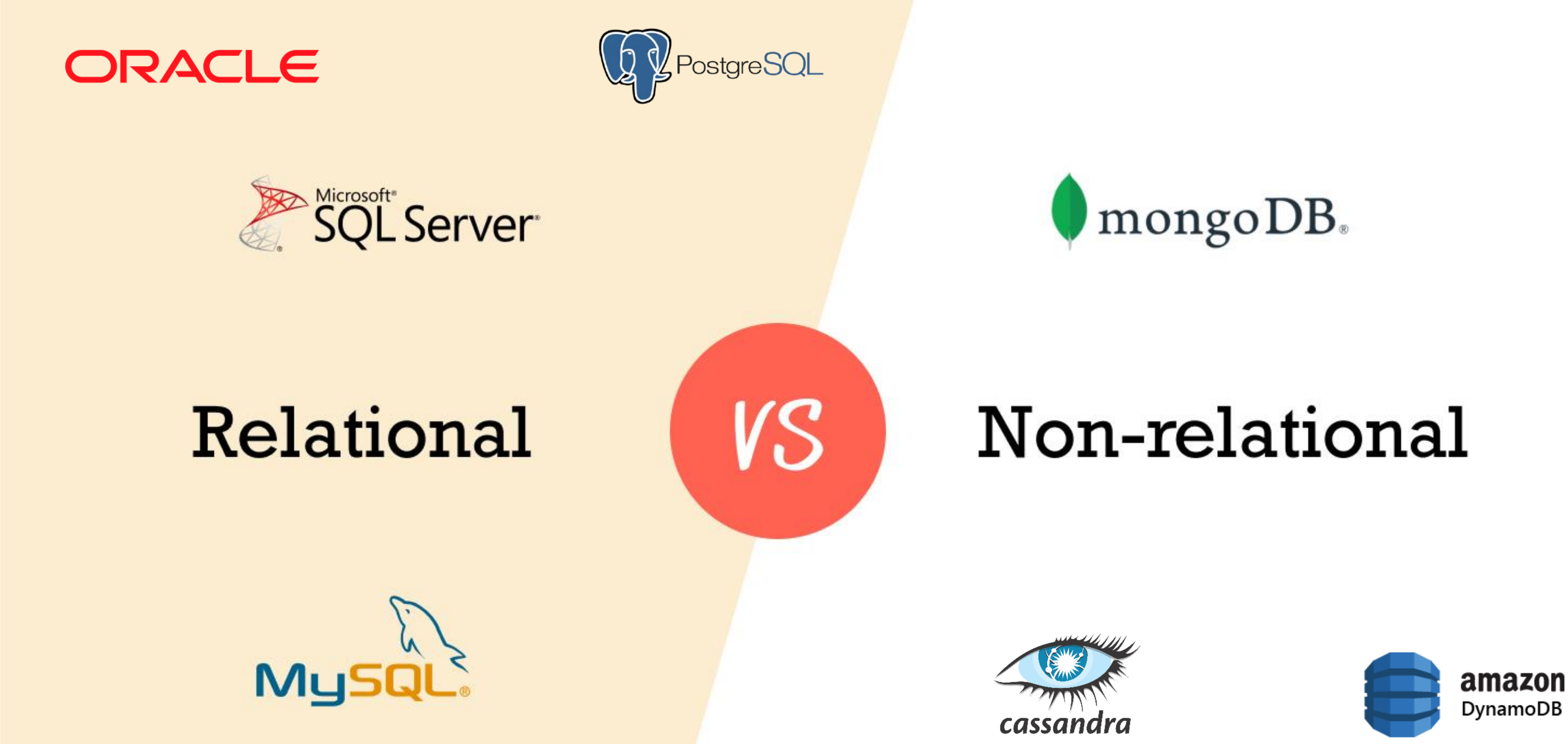
Web server

A thought would be..

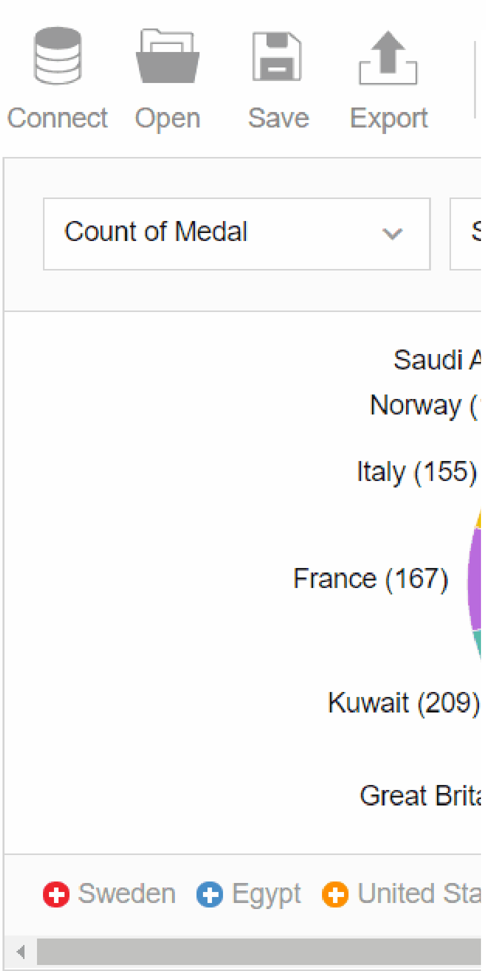
Which Database to use?

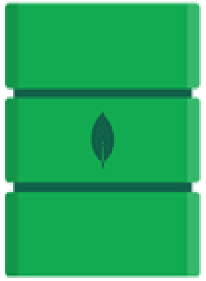
What are the different types of Databases?

Relational Databases VS Non-relational database



Relational Databases VS Non-relational database





When to use non-relational database?

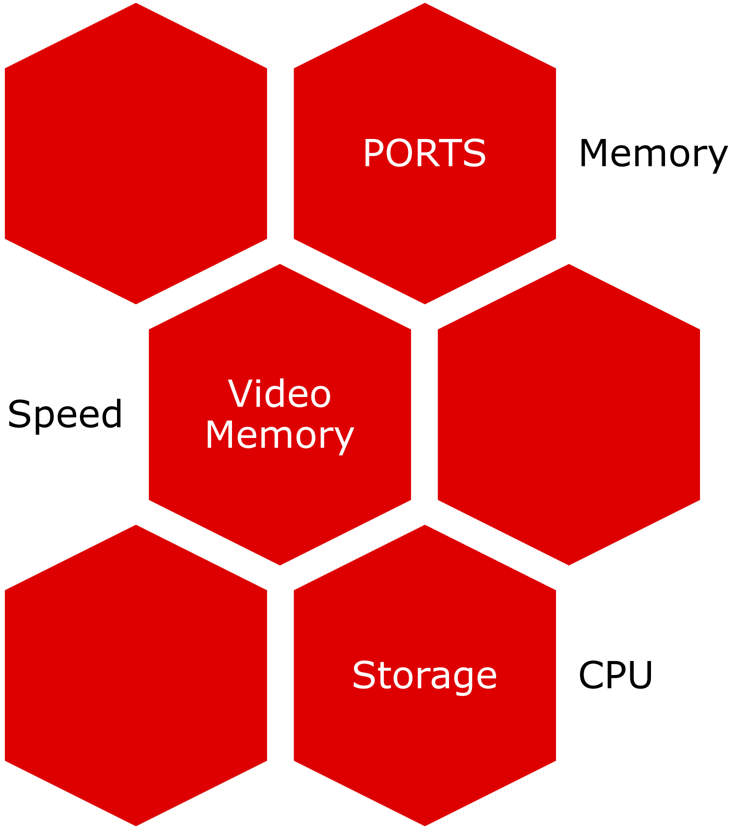
- If your application require **super-low latency**.
- Your data is **unstructured**.
- You need to **store a massive amount** of data.
- You need to **serialize and deserialize** data (JSON, XML, YAML)

We were discussing Single Server Setup with Database.

Back to the architecture

When you talk about servers?

- **It is just a computer with higher specification.**



PORTS

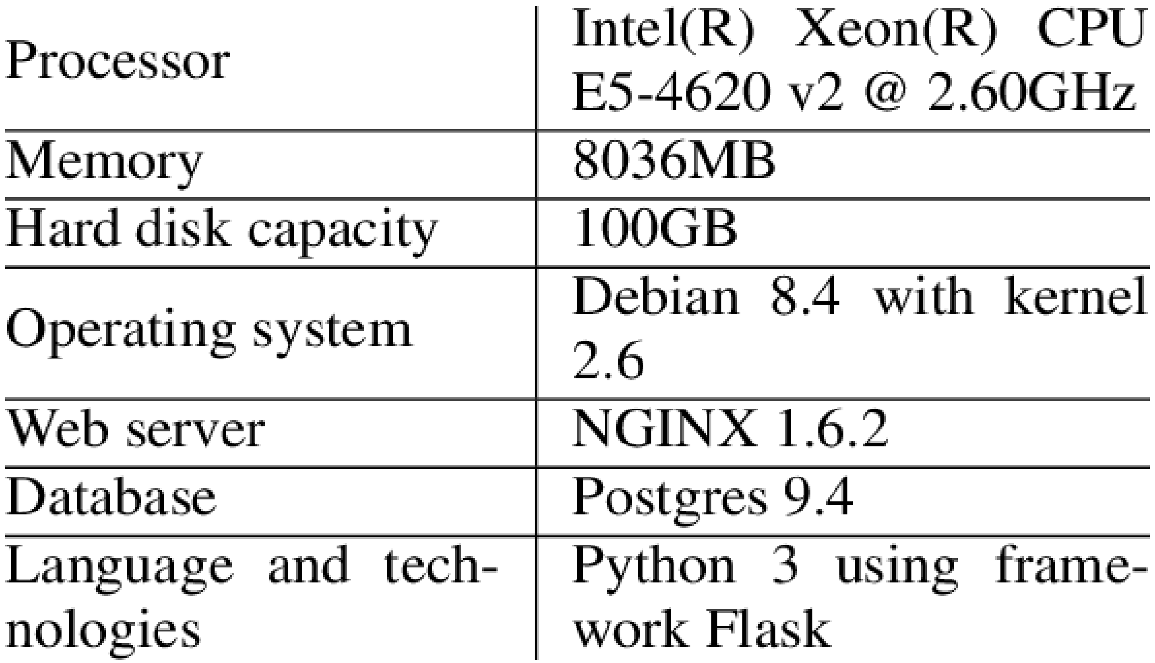
Memory

Speed

Video Memory

Storage

CPU



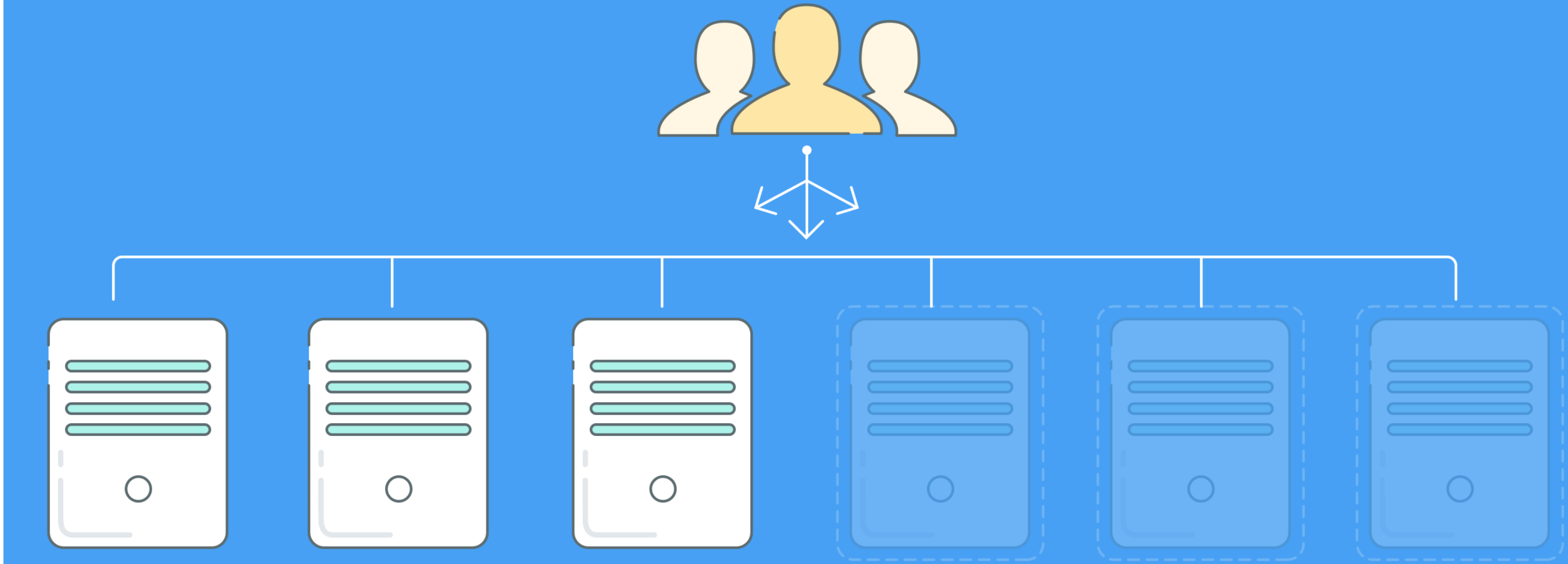
Processor	Intel(R) Xeon(R) CPU E5-4620 v2 @ 2.60GHz
Memory	8036MB
Hard disk capacity	100GB
Operating system	Debian 8.4 with kernel 2.6
Web server	NGINX 1.6.2
Database	Postgres 9.4
Language and technologies	Python 3 using framework Flask

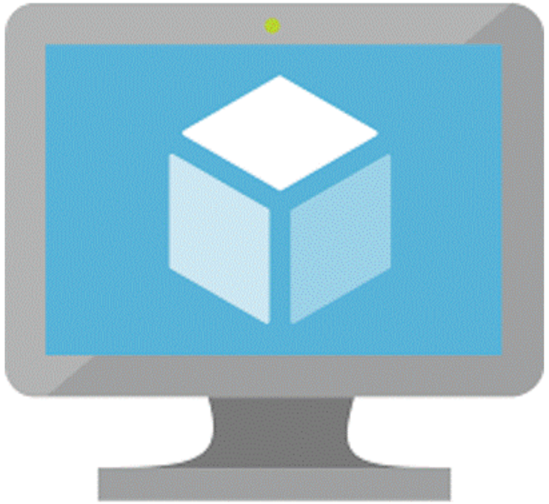
What will you do if?

Number of users increase, or you run out of specification?

- You will need to grow/increase your specification.
- This concept is called as: ?

Scalability





Vertical Scaling

Horizontal Scaling

Scalability thoughts ……. ????

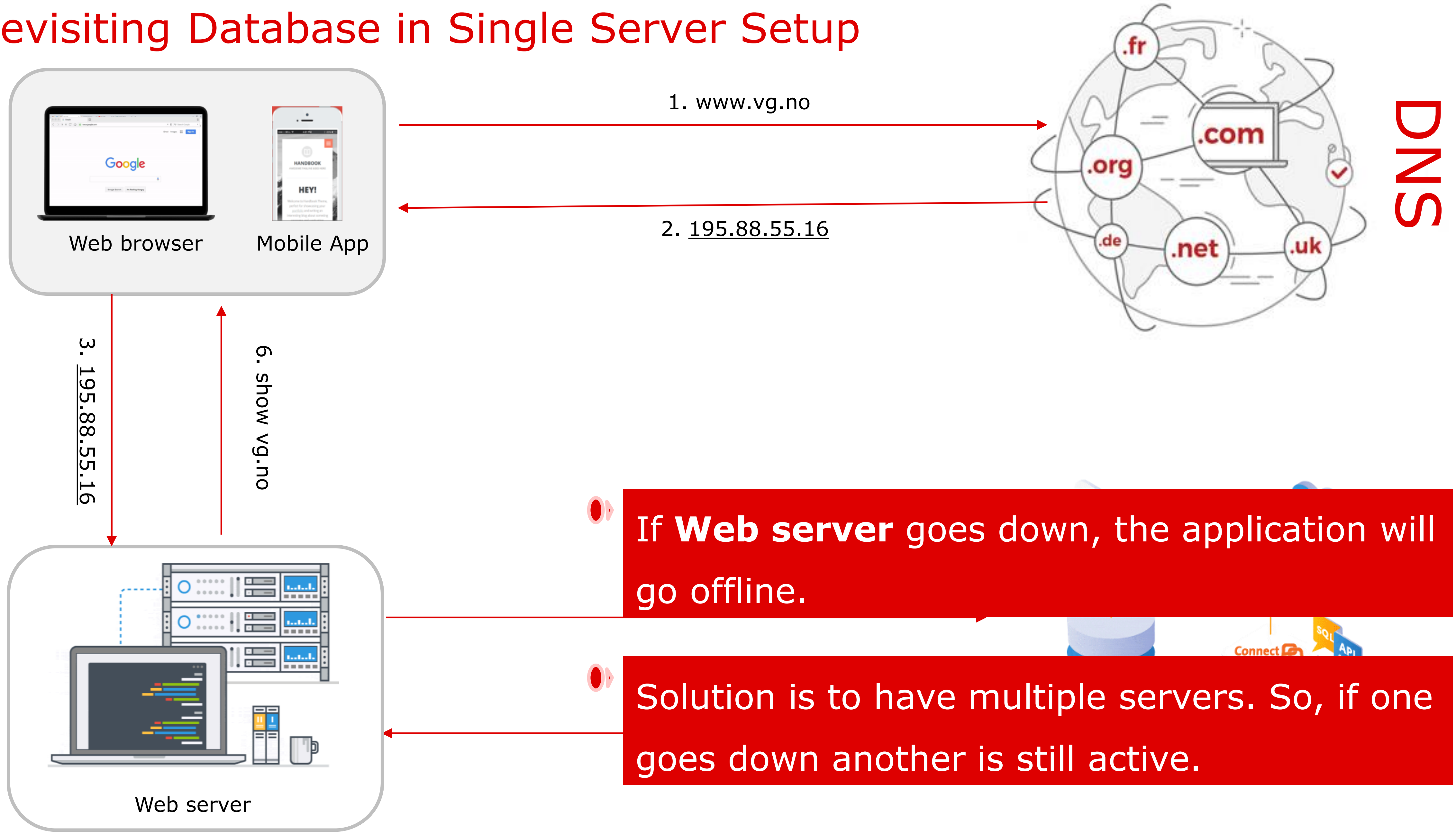
- **Vertical scalability** works when traffic is low. It is simple to implement.
  - Impossible to add unlimited CPU and memory to any computer.
  - If server goes down, entire application goes down.

- **Horizontal scalability** is more desirable for large scale applications.

Let us assume our userbase increases. Single Server Setup with Database will fail. We need to scale up.

Back to the single server setup.

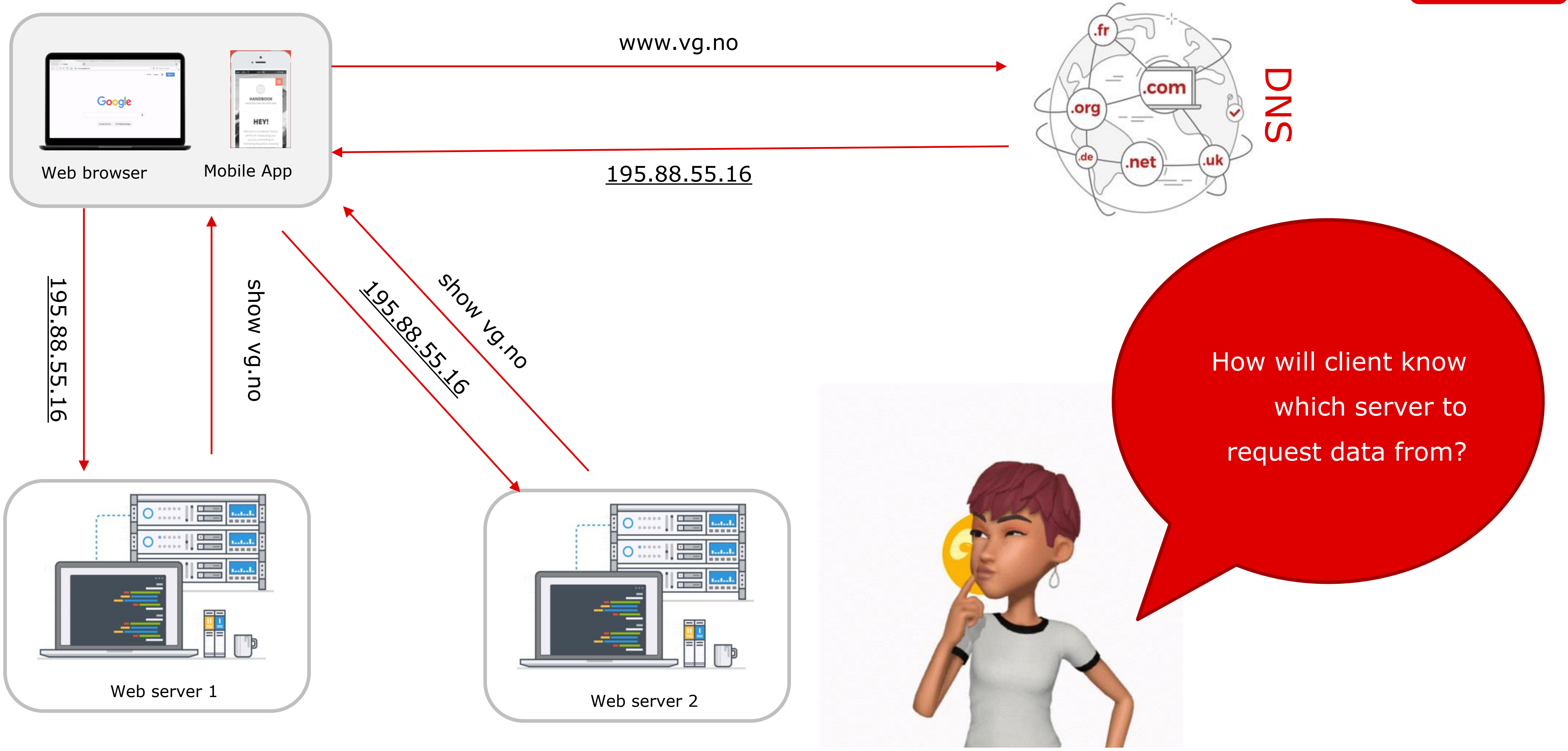
Revisiting Database in Single Server Setup



If **Web server** goes down, the application will go offline.

Solution is to have multiple servers. So, if one goes down another is still active.

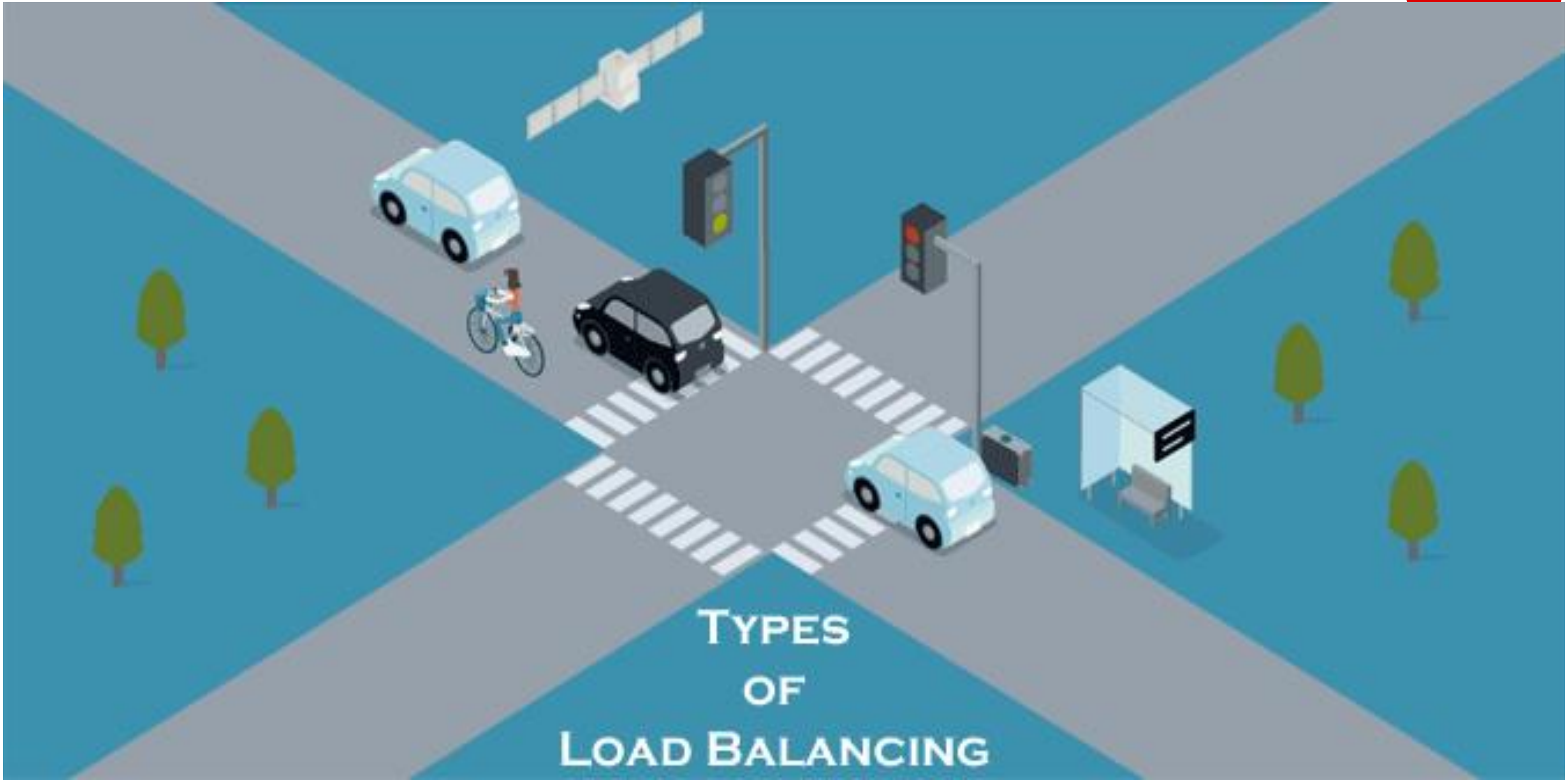
Let us increase one more server



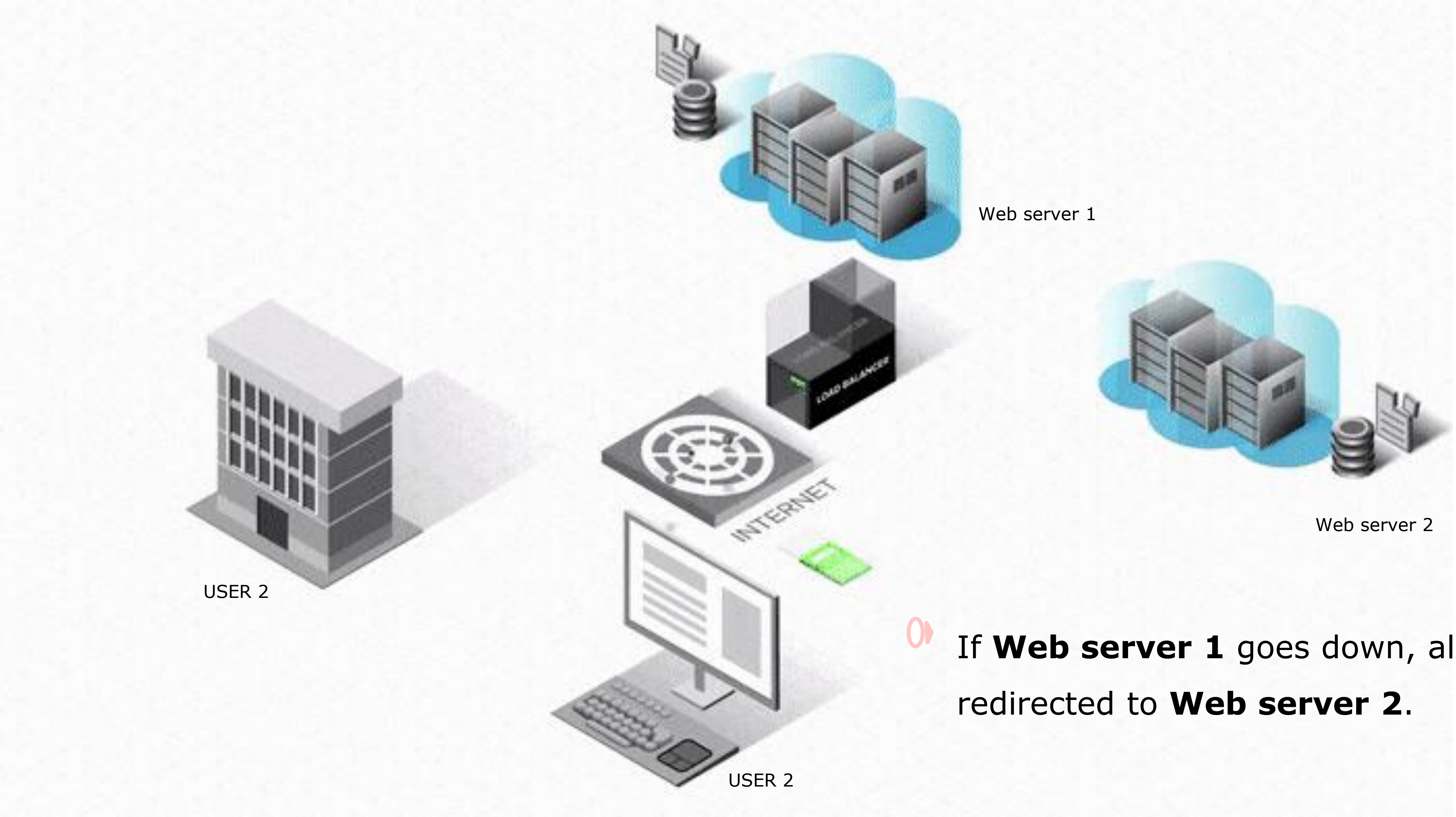
How will client know which server to request data from?

This is where the concept of **Load Balancer** comes in.

Real life load balancer: Traffic System



Let us put this concept into our architecture …

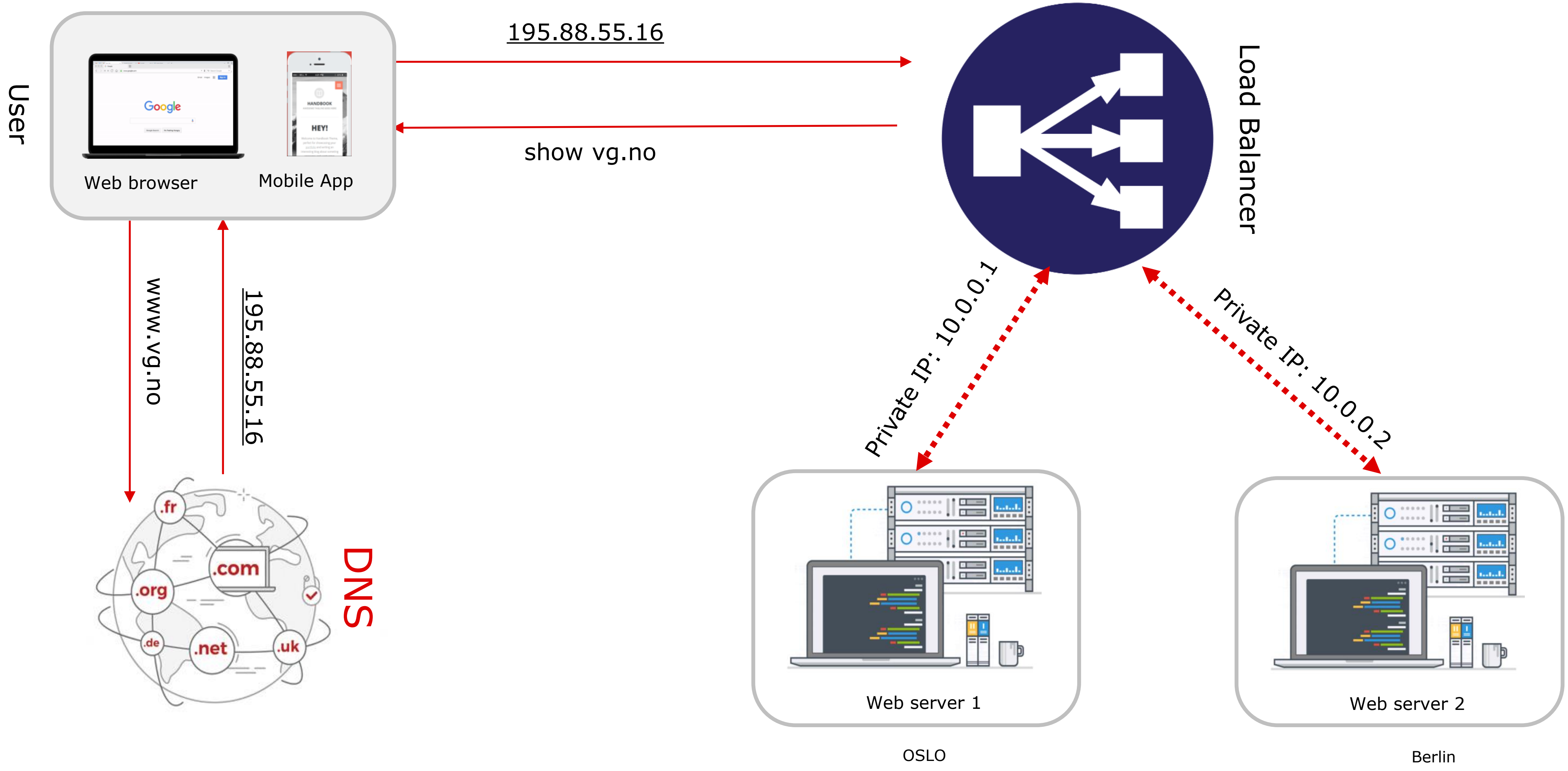


If **Web server 1** goes down, all traffic will be redirected to **Web server 2**.

We know what a Load Balancer is:

Let us include load balancer into our architecture

Let us include Load Balancer into our architecture.



User

Web browser

Mobile App

195.88.55.16

show vg.no

Load Balancer

www.vg.no

195.88.55.16

DNS

Private IP: 10.0.0.1

Private IP: 10.0.0.2

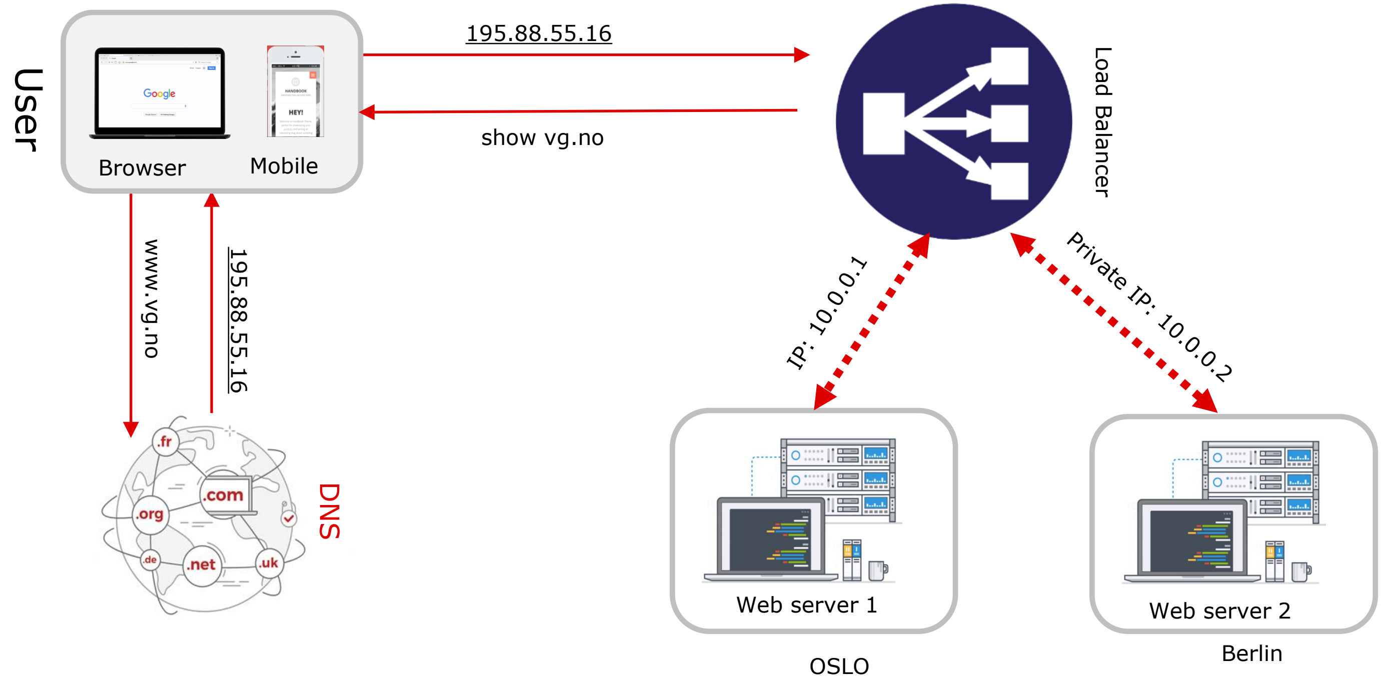
Web server 1

OSLO

Web server 2

Berlin

Let us see benefits of this architecture

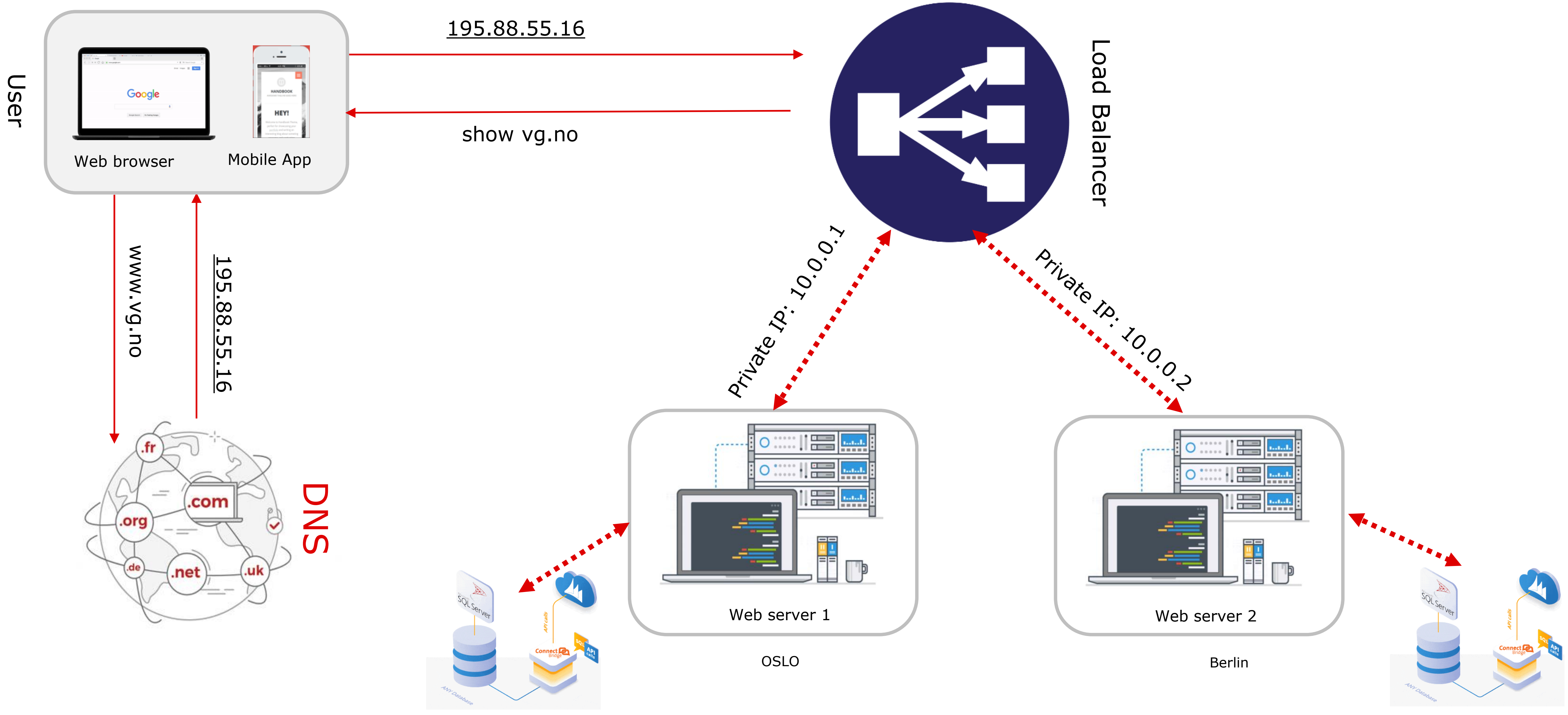


Software Quality Attributes

- Availability
- Scalability
- Security

- If **Web server 1** goes down, all traffic will be redirected to **Web server 2**.
- Web servers are unreachable directly. It provides better security.
- If two servers are not enough, we only need to add more servers.

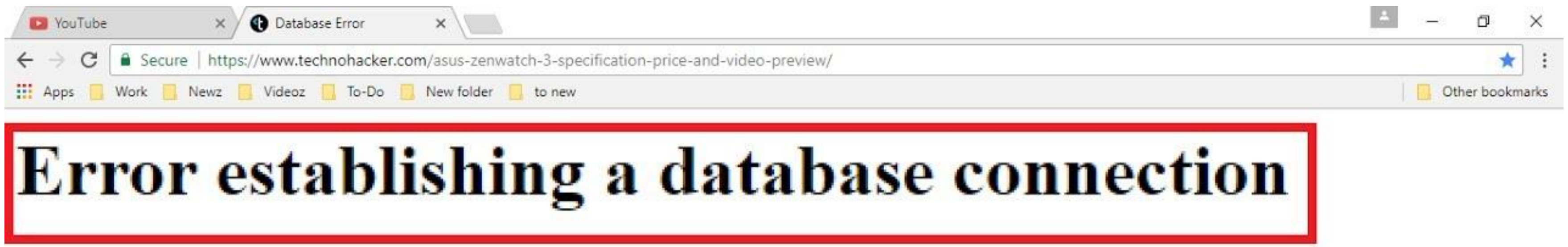
Two-server architecture with Database.



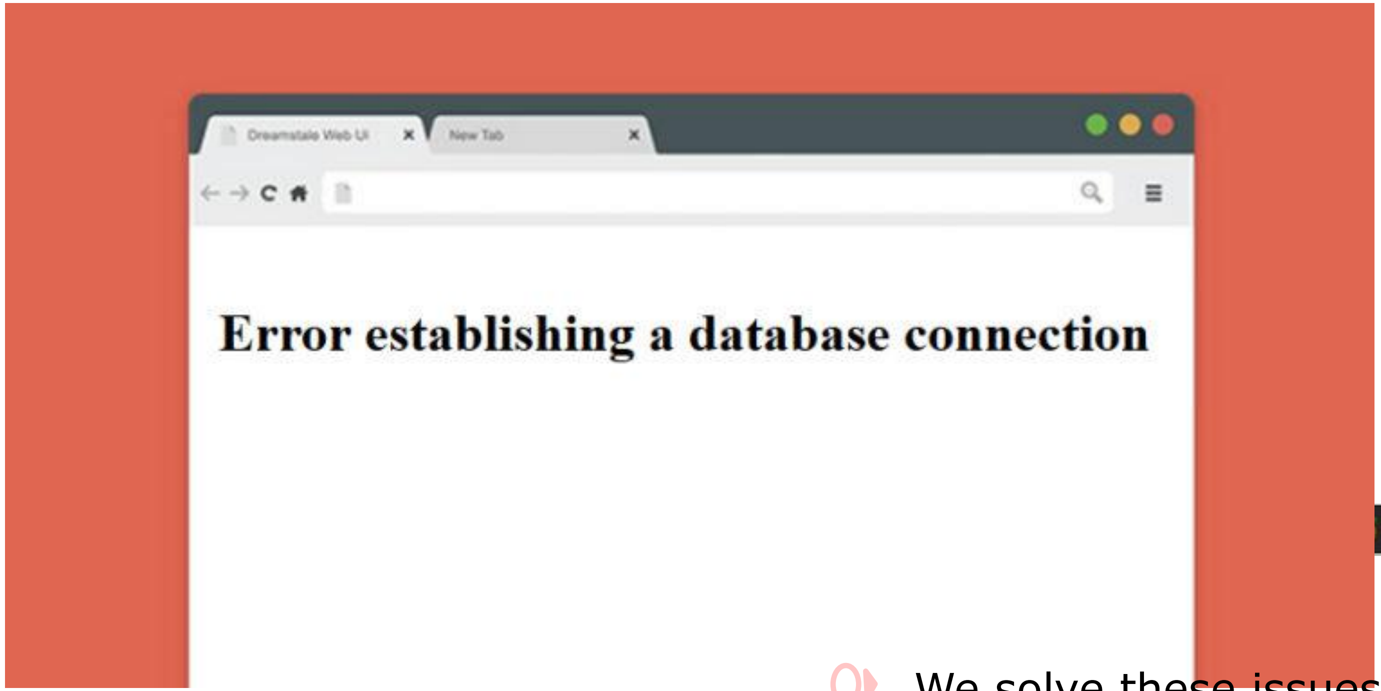
We solved server failure situation (By having multiple servers)

P1: What about database failure…?

P2: What if database gets corrupted?


YouTube
Database Error
Secure
https://www.technohacker.com/asus-zenwatch-3-specification-price-and-video-preview/
Apps
Work
Newz
Videoz
To-Do
New folder
to new
Other bookmarks

Error establishing a database connection

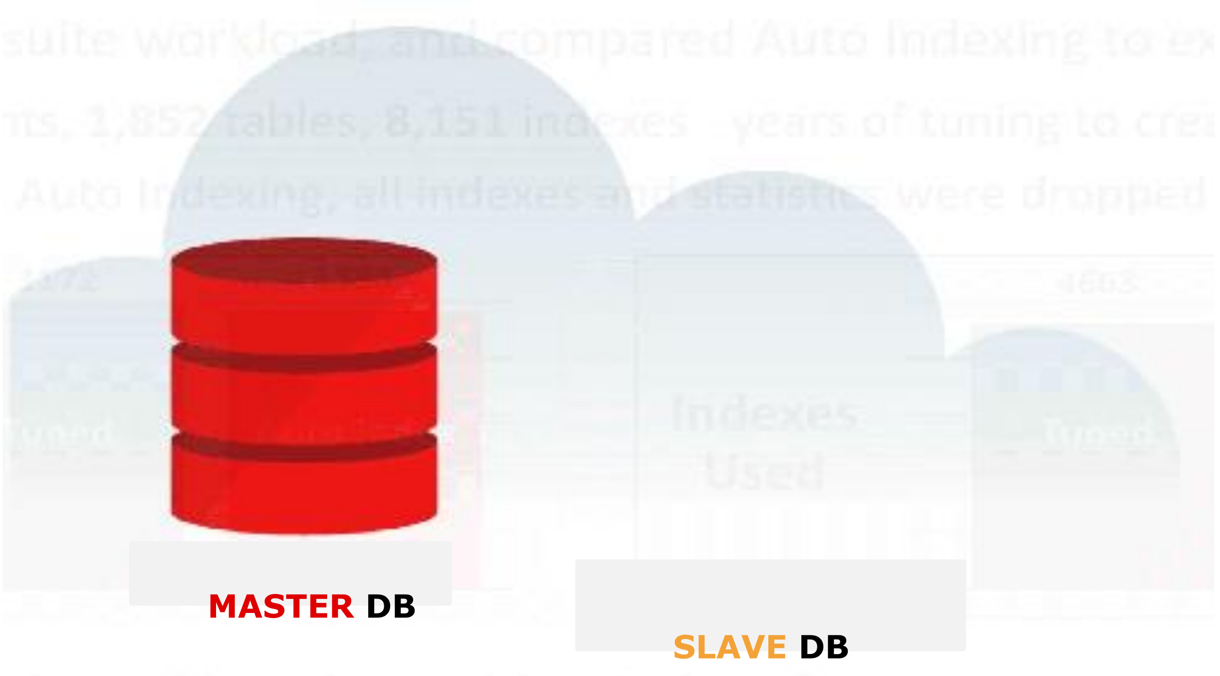


Error establishing a database connection

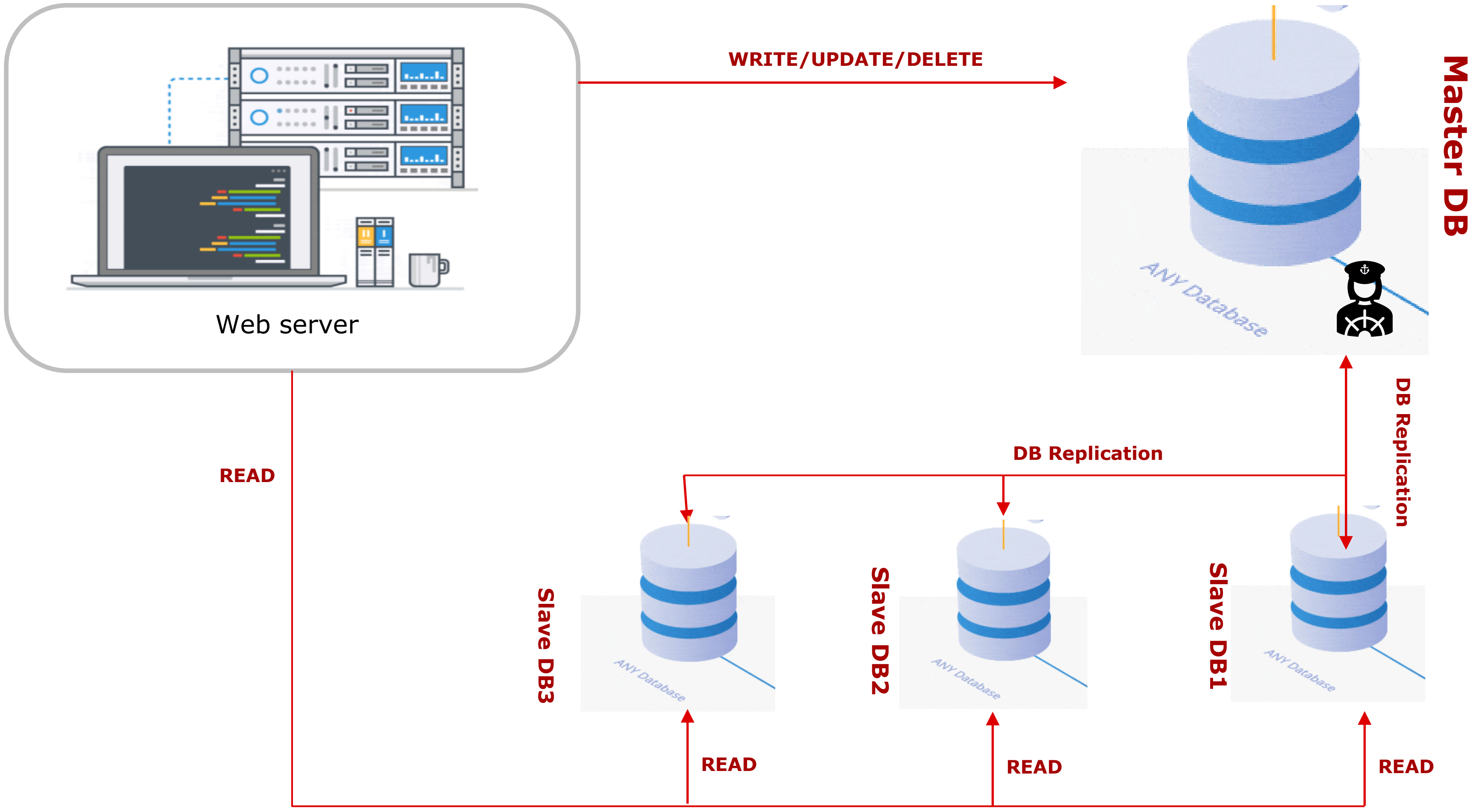
- We solve these issues by a technique called database replication.

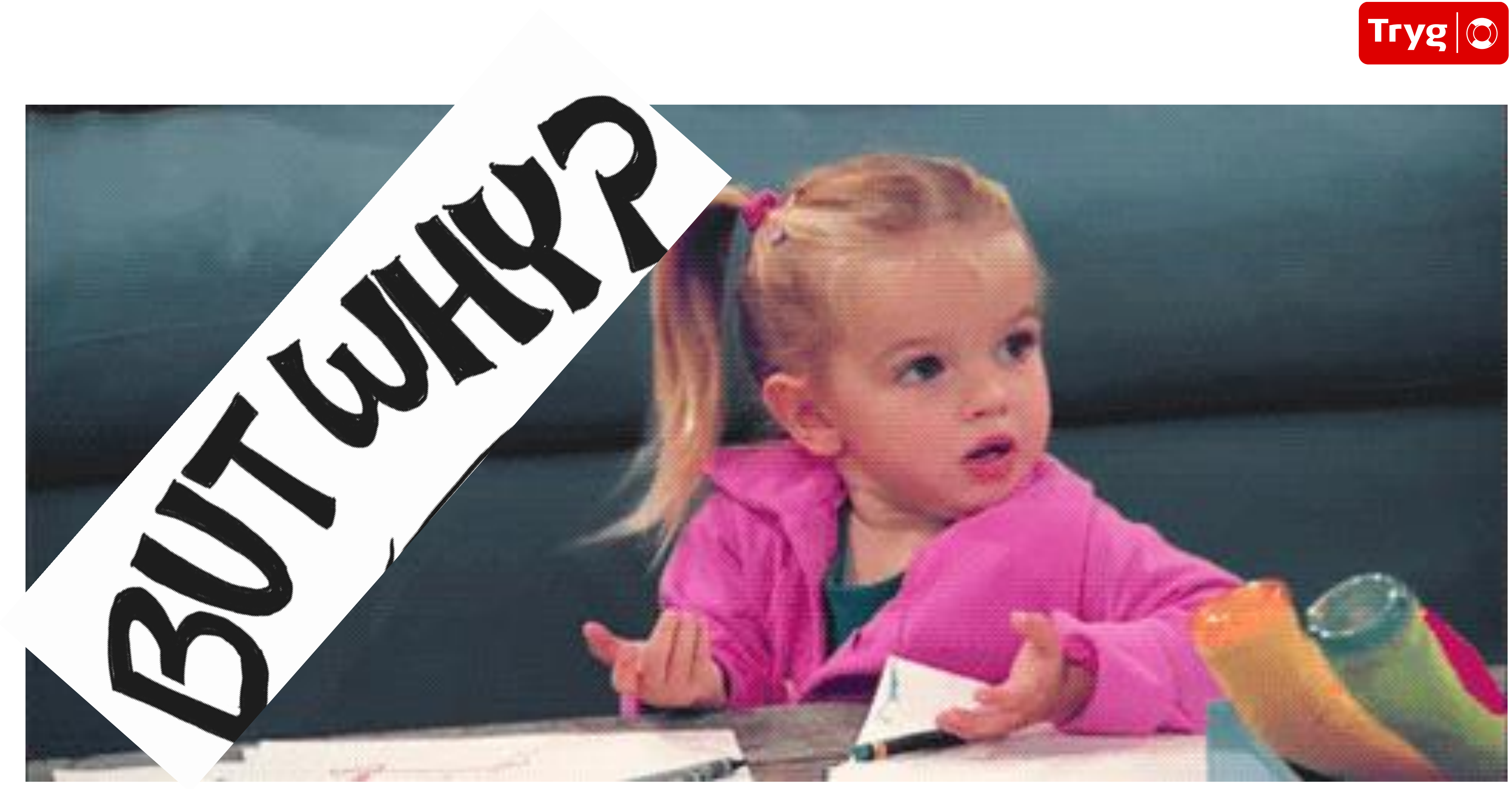
Database Replication

- Database replication means making copies of the original database.
  - The original database is called **master**.
    - The database copy is called **slave**.

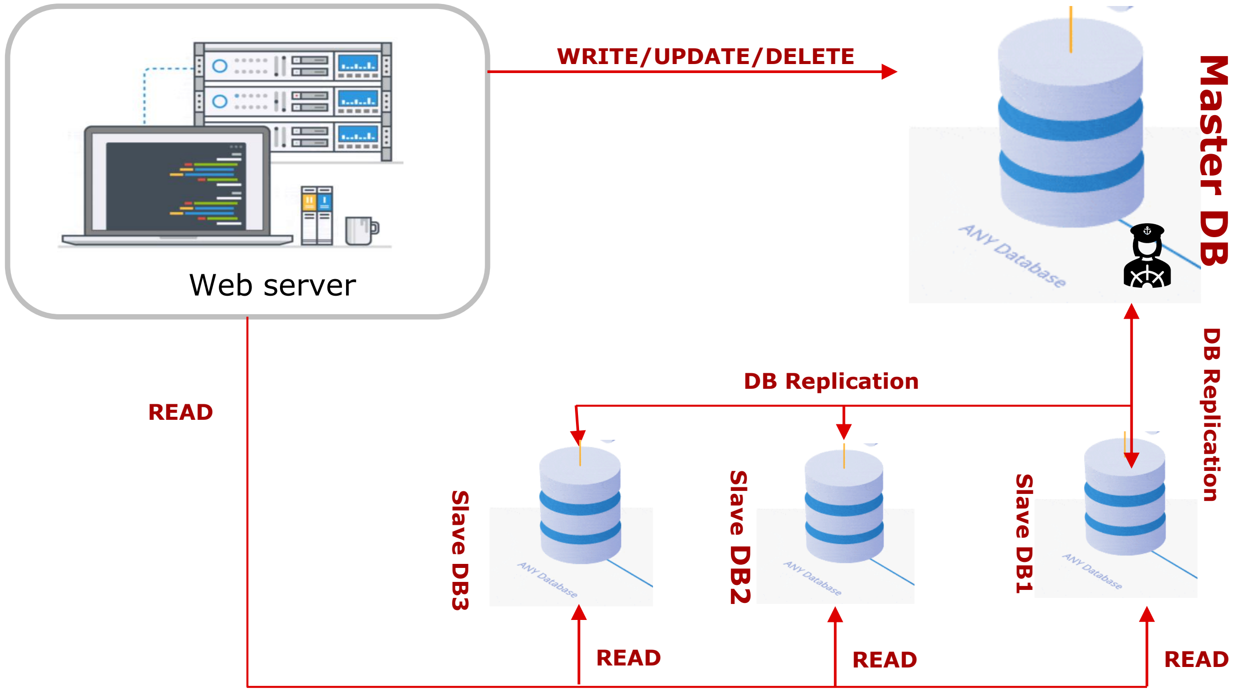


Database Replication – Master and Slave




BUT WHY?

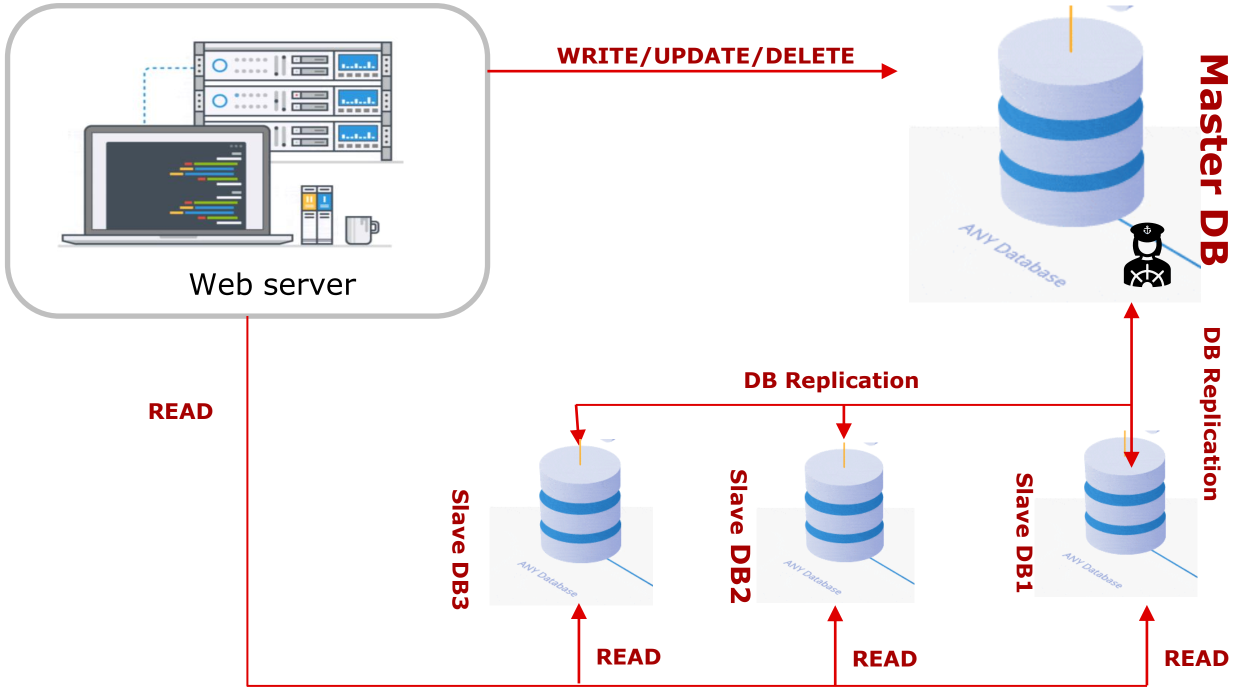
Database Replication – benefits



- Most application requires higher ratio of read to write. Hence n(slave) > n(master).
- **Better Performance:** Read happens on slave, write happens on master, hence queries happens in parallel.
- **Reliability:** If one server is destroyed by natural disaster, data is still preserved. No data loss.
- **Availability:** Databases are replicated across different locations. Web application remains active even if some database server fails.

But what if a database goes offline?

What if one of the DBs goes offline?



- **If there is only 1 slave**, and if it goes offline, read operations will be redirected to master DB temporarily.

- **If there is multiple slaves,** and if one slave goes down, read operations are redirected to healthy slaves.

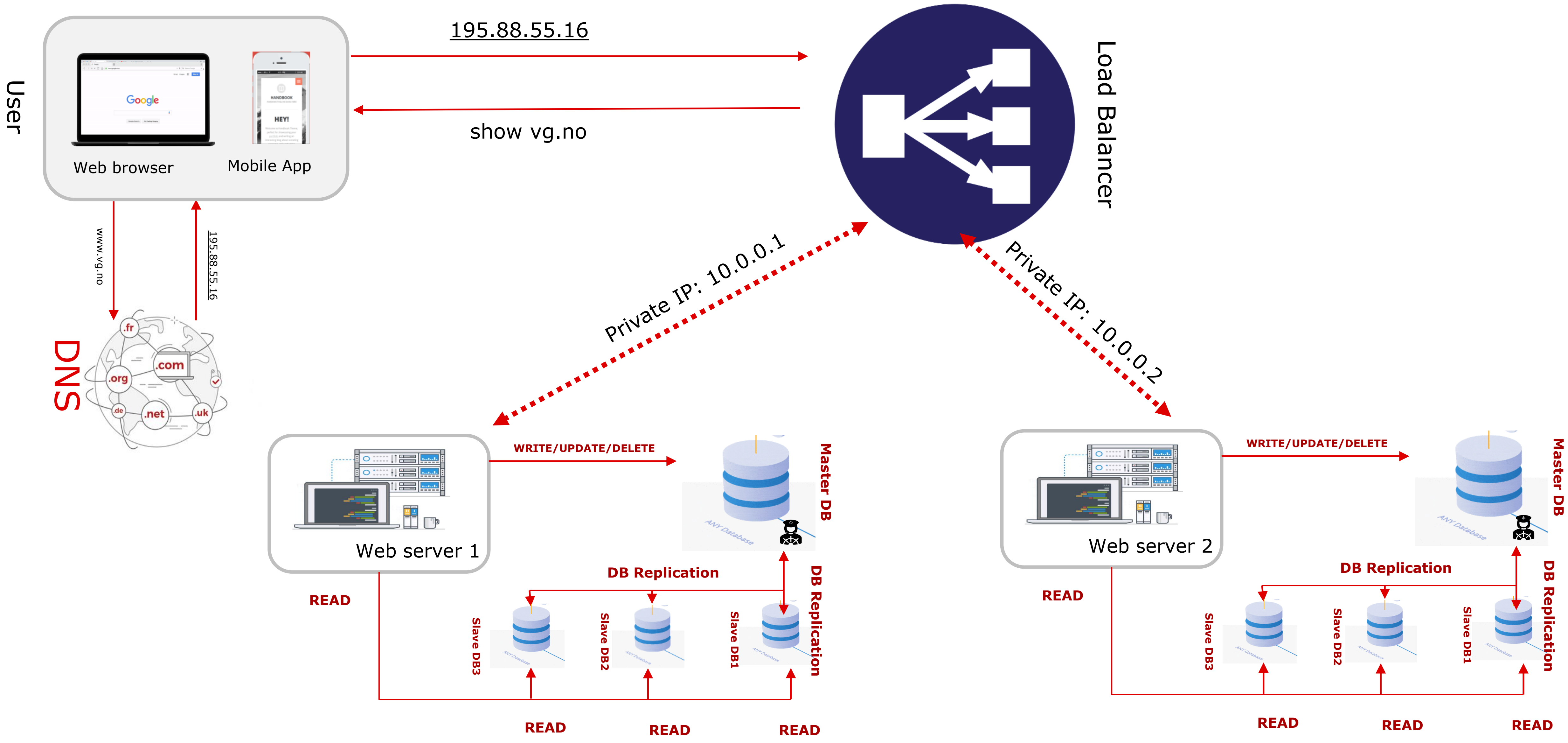
- **If master DB goes offline,** a slave DB is promoted to master DB. Faulty is replaced/repaired and goes back to former process.

- **Master/slave** management requires complex configuration.

We discussed database replication.

Let us include DB replication into our 2-server architecture

Two-server architecture with LB, DB replication

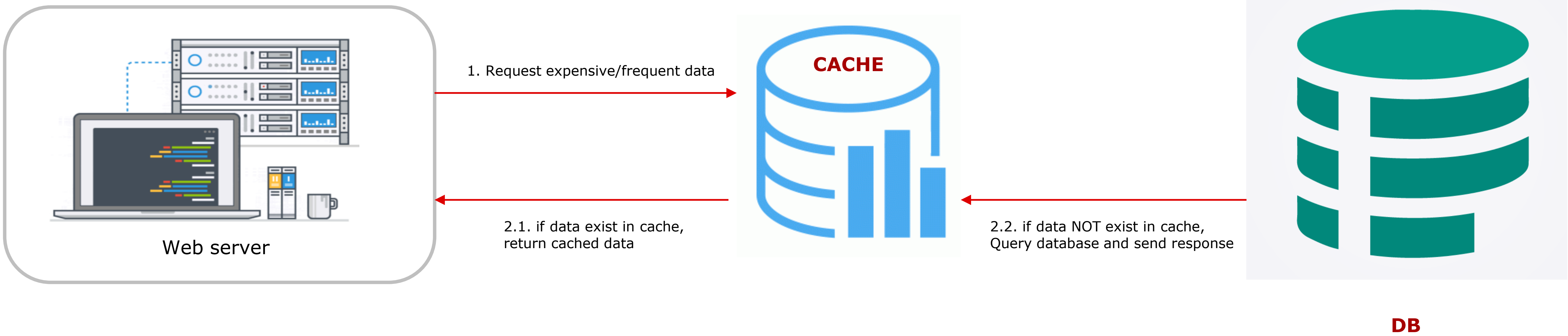


Now you have a solid understanding of Web Tiers and Database Tiers.

Let us discuss how to improve the load / response time.

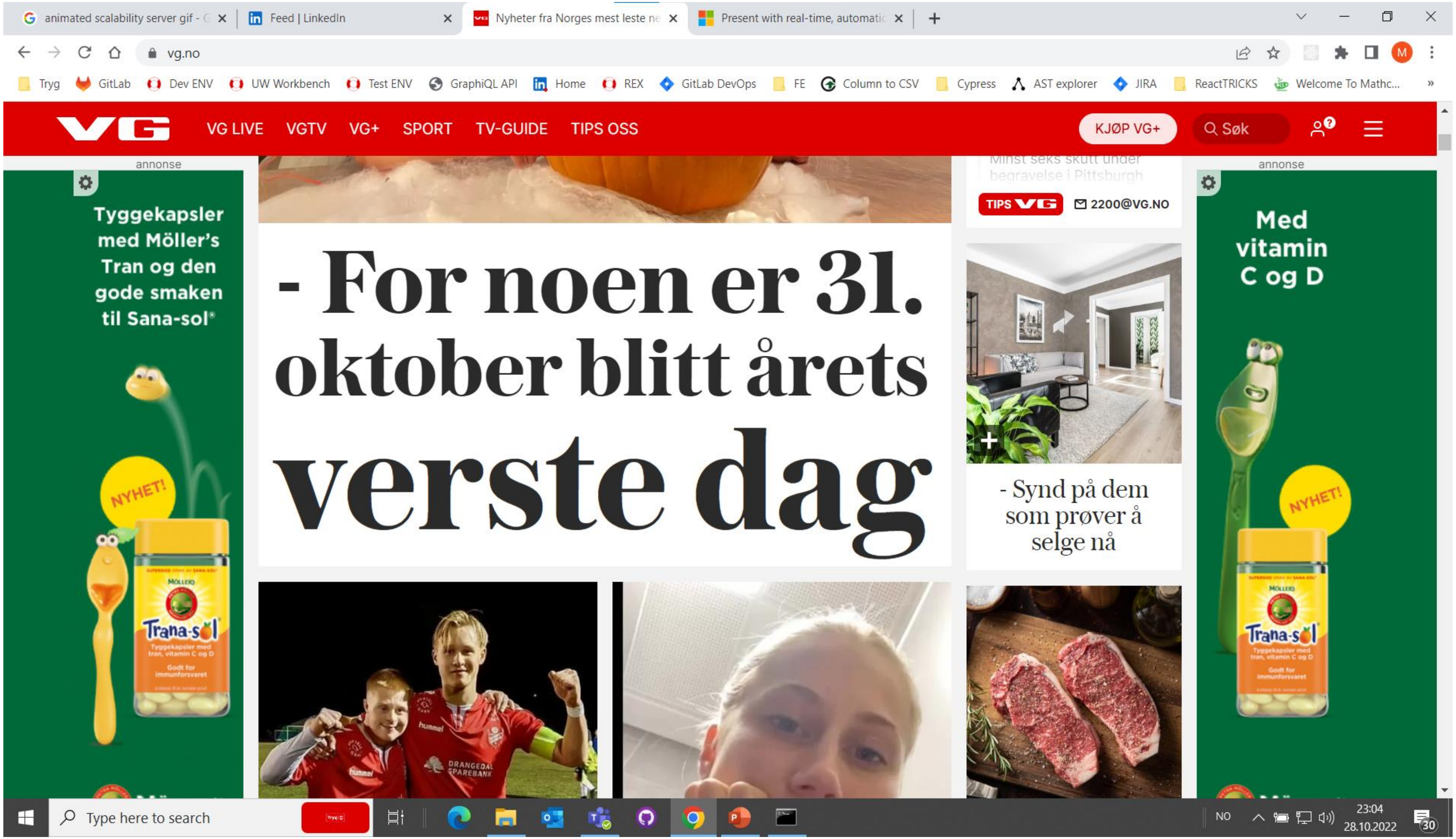
Cache – temporary storage

A temporary storage that stores the result of <u>**most expensive responses**</u> or most frequently accessed data.


Web server
1. Request expensive/frequent data
CACHE
2.1. if data exist in cache,
return cached data
2.2. if data NOT exist in cache,
Query database and send response
DB

Cache is much faster than actual database. The aim of cache is to increase system performance.

Static content in any web application



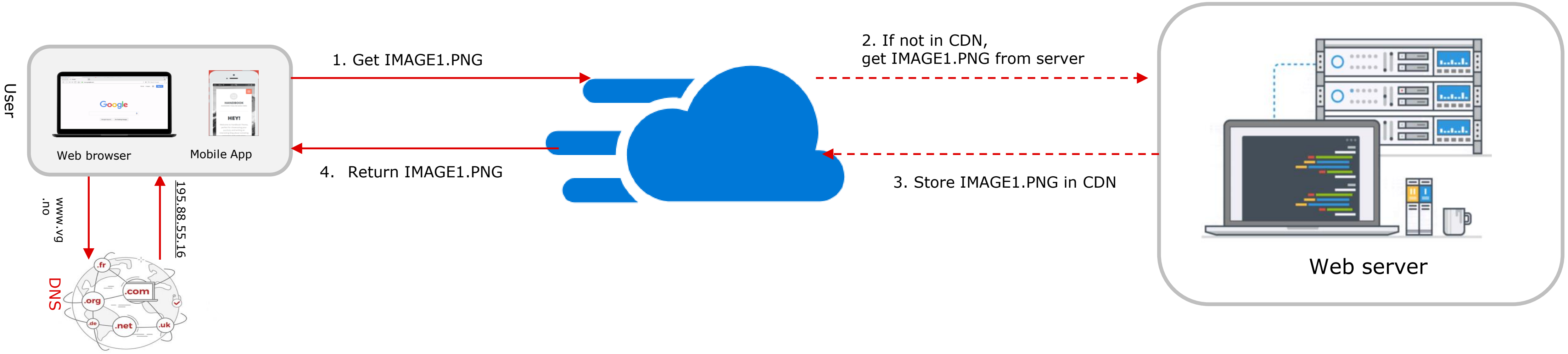
Minst seks skutt under begravelse i Pittsburgh

- For noen er 31. oktober blitt årets verste dag

- Synd på dem som prøver å selge nå

Content Delivery Network (CDN)

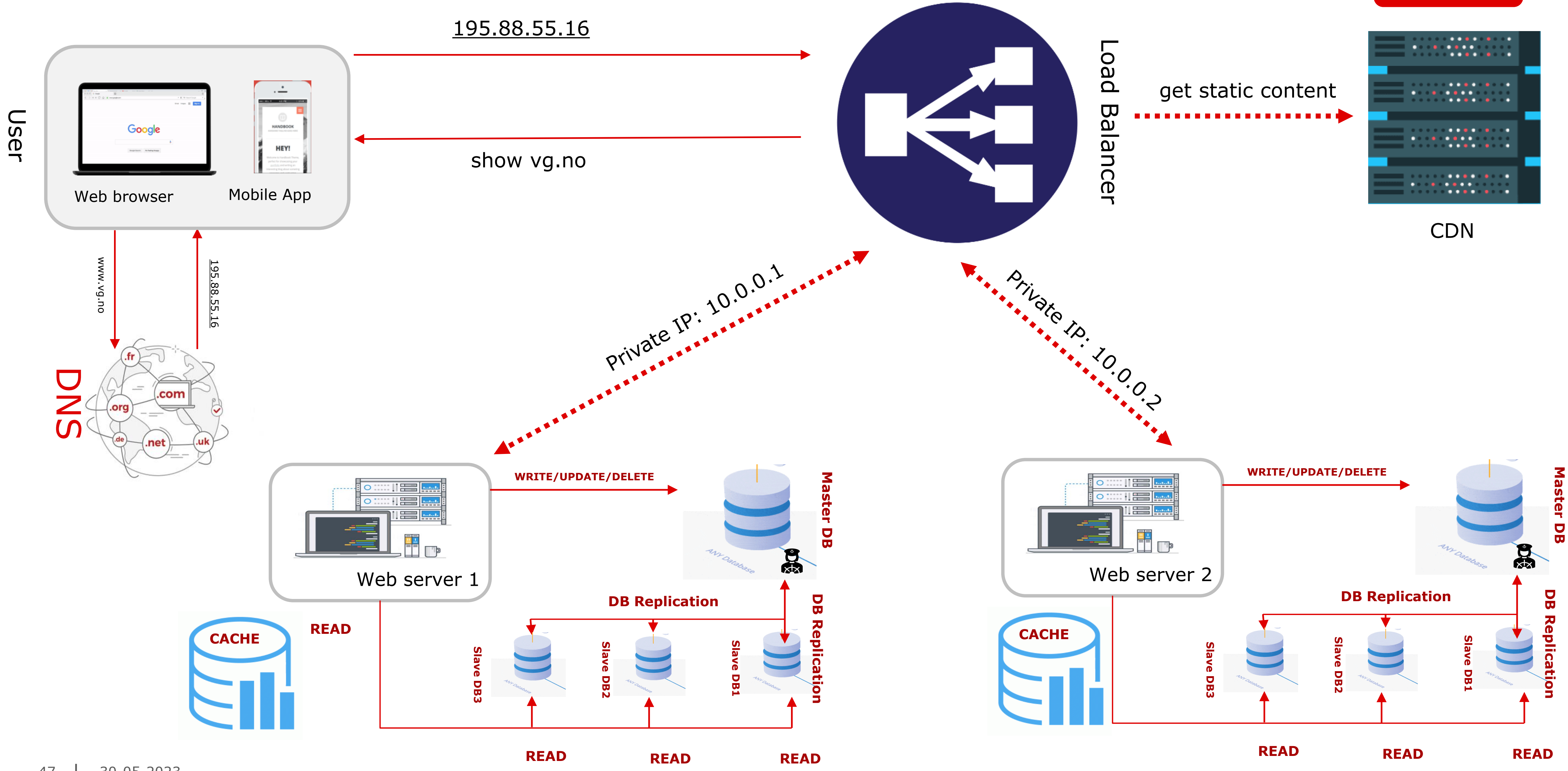
- Static contents: Images, Videos, Icons, CSS, JavaScript Files etc.
- The idea is to store static contents in one ore more CDN servers.
- When a user visits a web application, CDN server closest to the user delivers static content. Dynamic content is fetched from Web Servers.



Now you know the concept of cache, and CDN.

Let us include cache and CDN into our architecture.

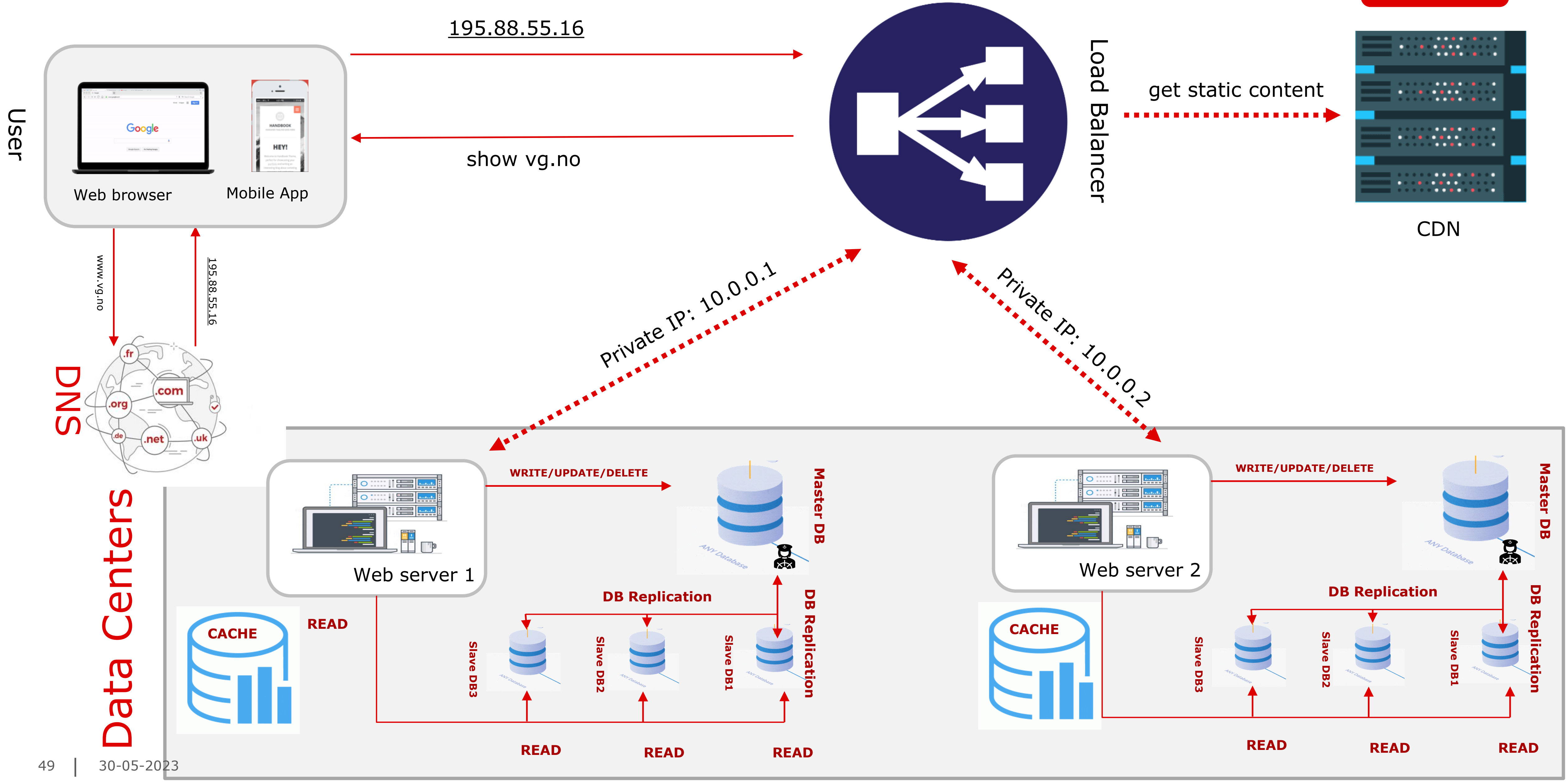
Two-server architecture with LB, DB replication, cache, CDN



Now you know the concept of client, server, cache, DB, CDN.

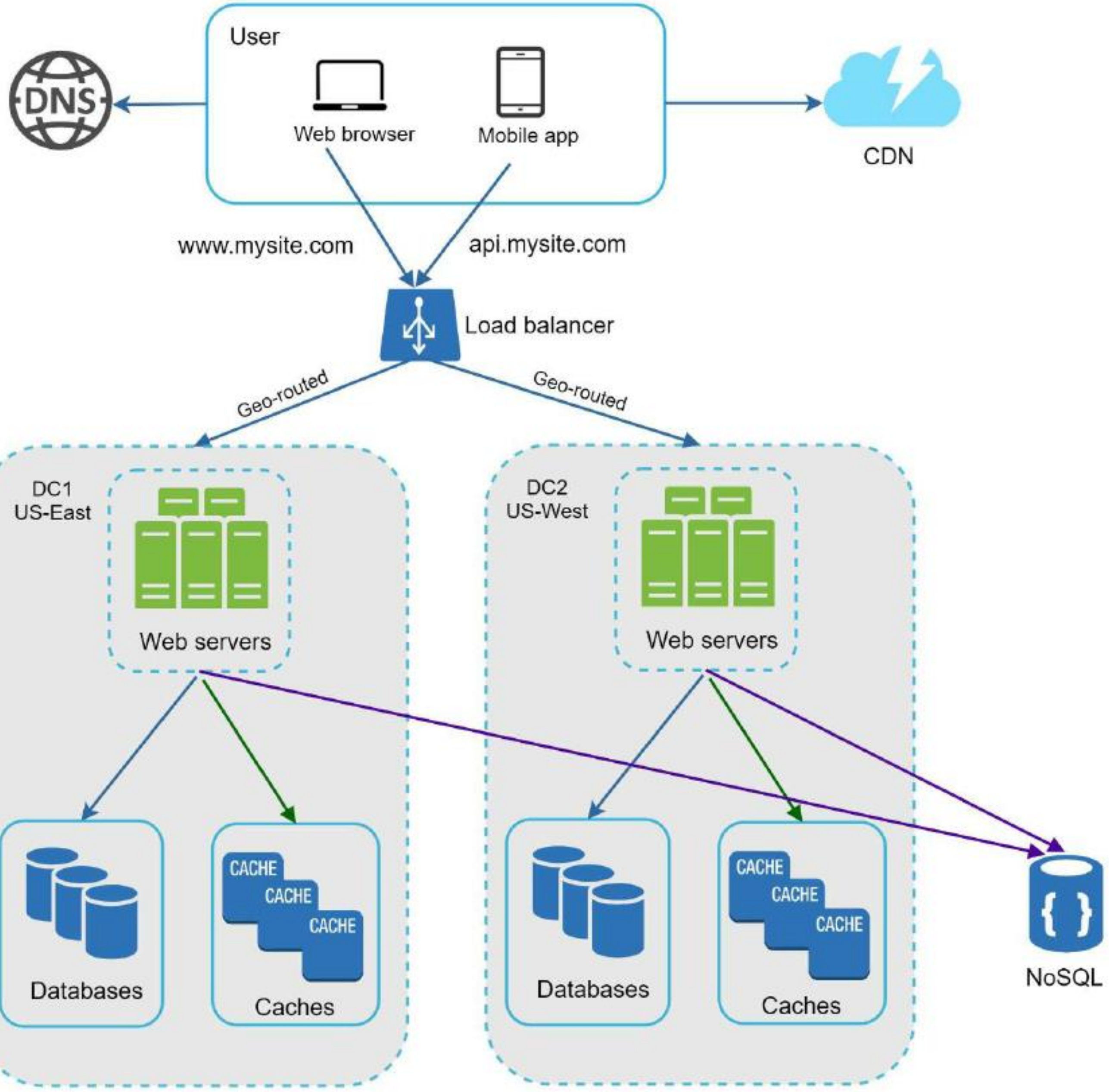
How about millions of users? Two servers are not enough. Need more.

Multi-server architecture – Data Centers



Architecture with shared DB

- Two Data centers and with shared DB
- Shared DB is used for most frequently stored Data like Authorization / Authentication

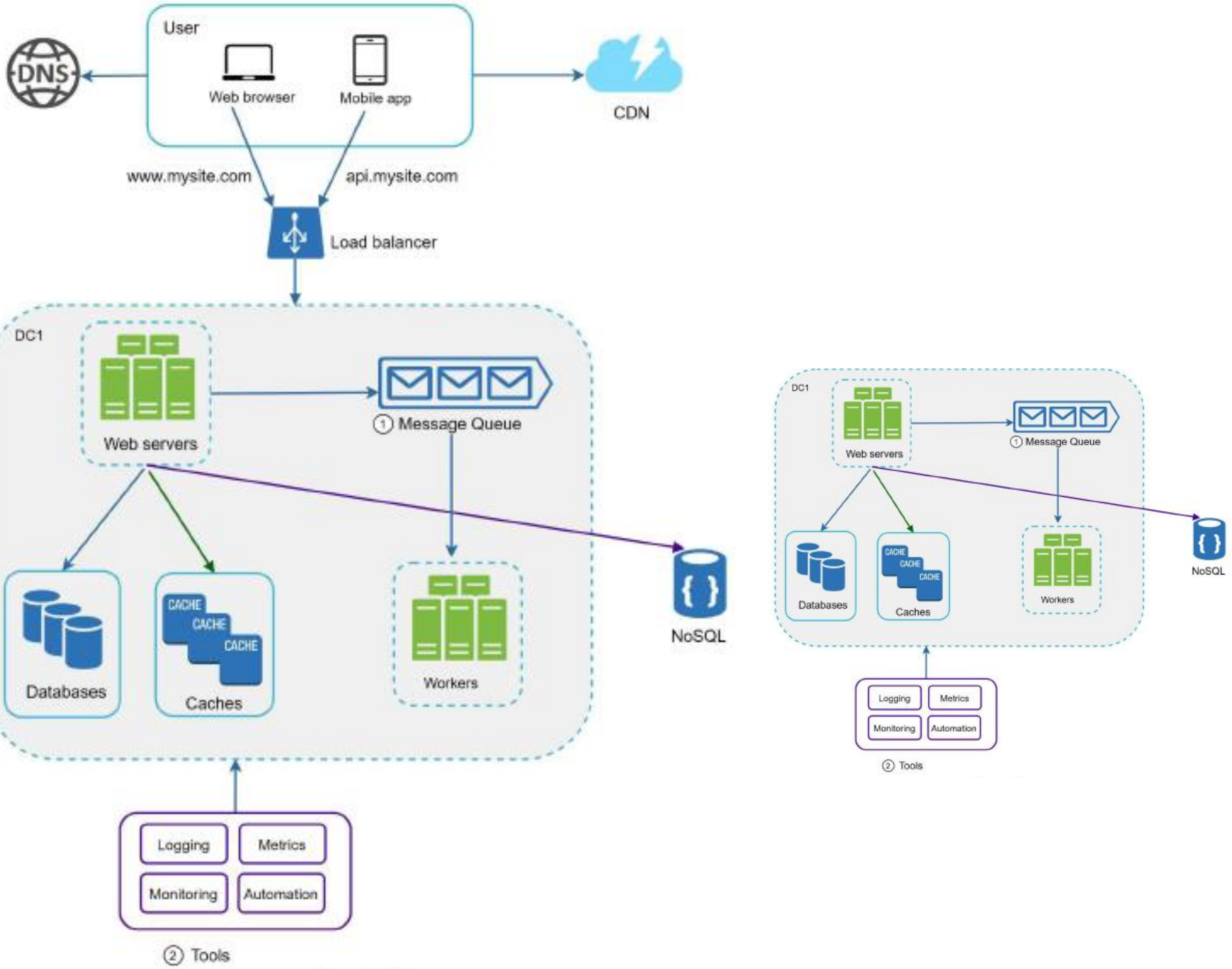


Architecture with shared DB / Loggings

- Two Data centers and with shared DB / Workers and Logging

- Every events are logged, monitored and automated.

- These events are used for workflows, fraud detection, network analysis, and many more.

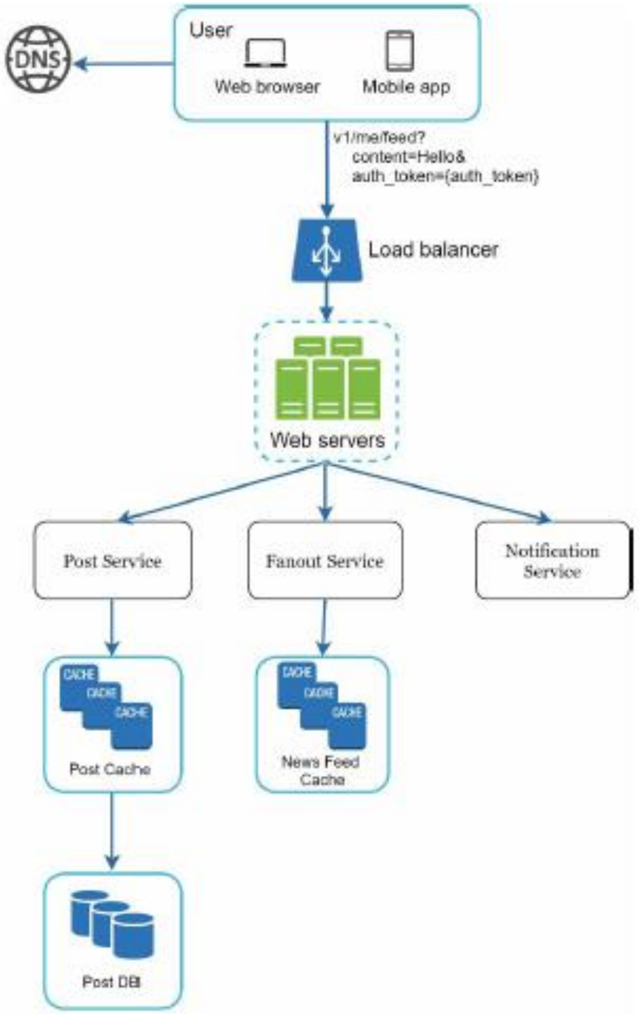


Typical Server with SOA

- ➢ Post service stores post in database and cache.

- ➢ Fanout service pushes new content to friend’s news feed. News feed is stored in the cache.

- ➢ Notification service informs friend that the new content is available.



… And there is endless possibilities.

… Always remember, your architecture depends on your requirements.

All models are wrong, but some are useful. - George Box

Take Away

- Designing any system depends on the requirements.
- There is not perfect architecture for any enterprise. It depends on use cases.
- Non-functional requirements often decides technical requirements.
- Requirements, user flows and processes are often explored using tools like BPMN, UML, Process Mining etc.
- Scalability, Performance, Security, Reliability and availability is essential quality attributes for enterprise applications.
- Security is an essentials software Quality attributes which is out of scope of this lecture.
- Lecture is inspired by book: System Design Interview by *Alex Xu.*

Further Questions

www.skmukhiya.com.np
(itsmeskm99@gmail.com)

- ➢ We are continuously hiring Developers in Tryg Norge. Reach out to me for any IT roles.