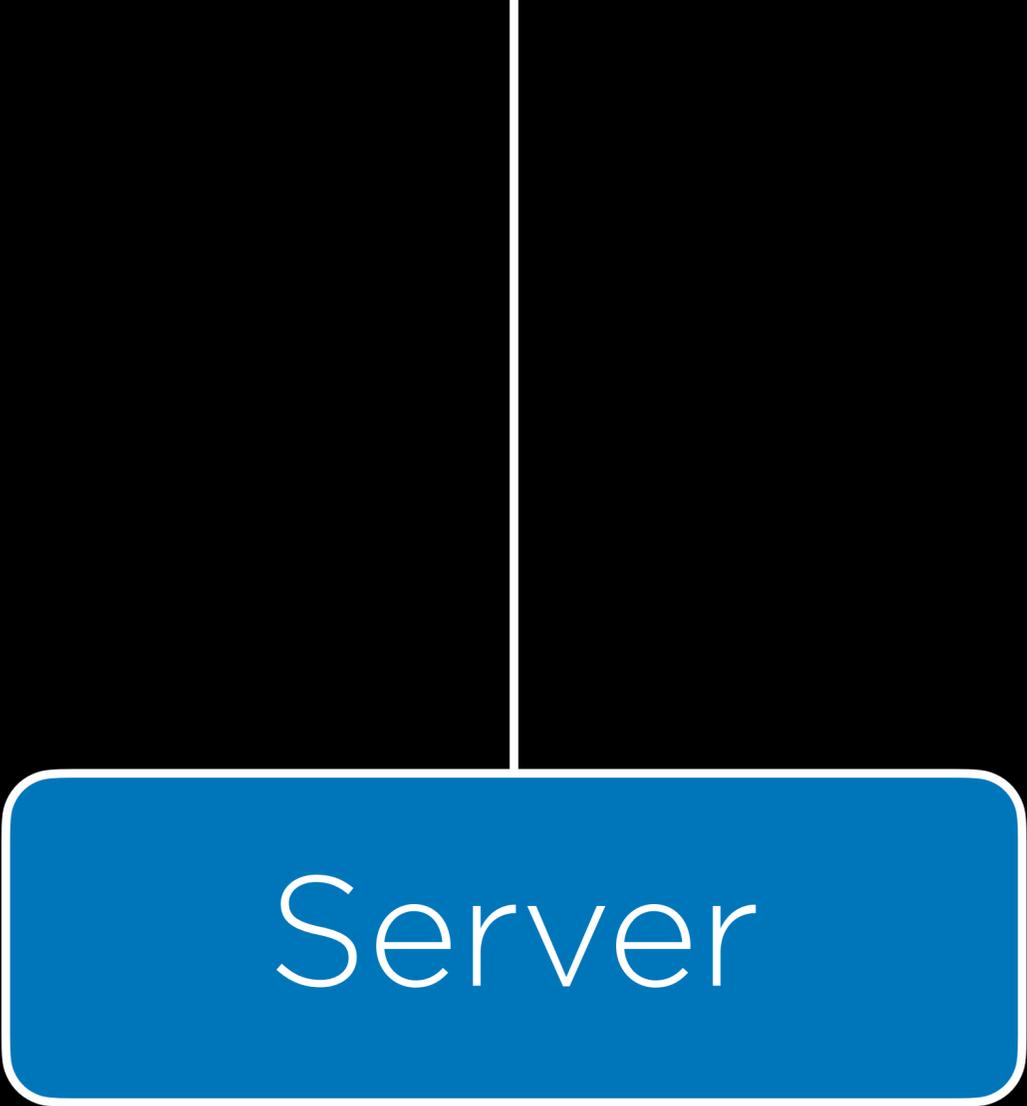


# Web Programming

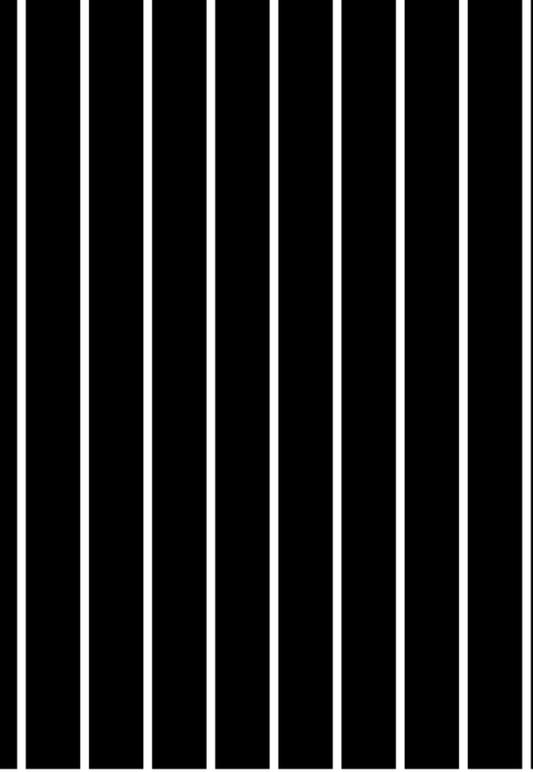
with Python and JavaScript

# Scalability and Security

**Scalability**



Server



Server

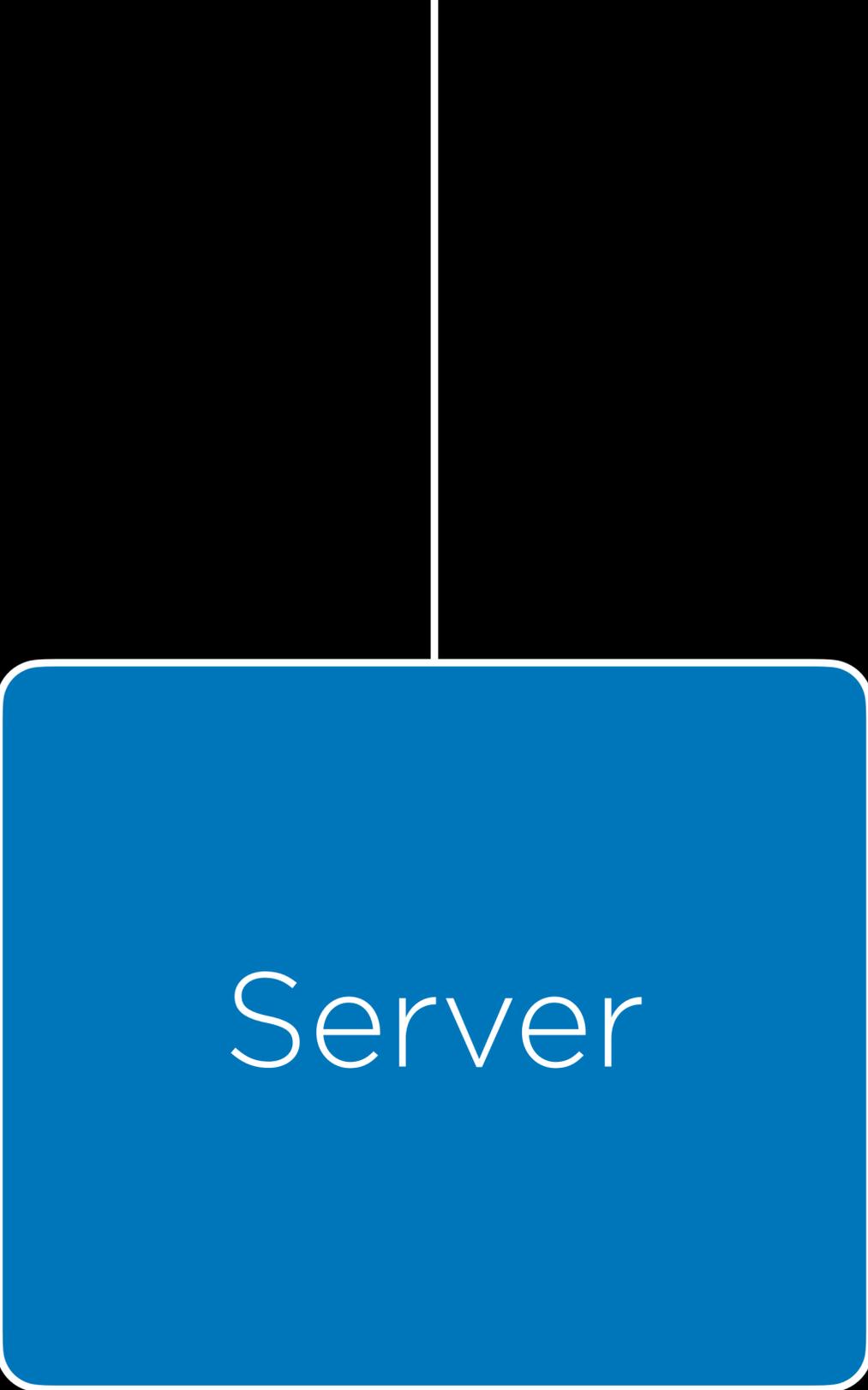
# Servers

- Cloud
- On Premise

# Benchmarking

# Vertical Scaling

Server



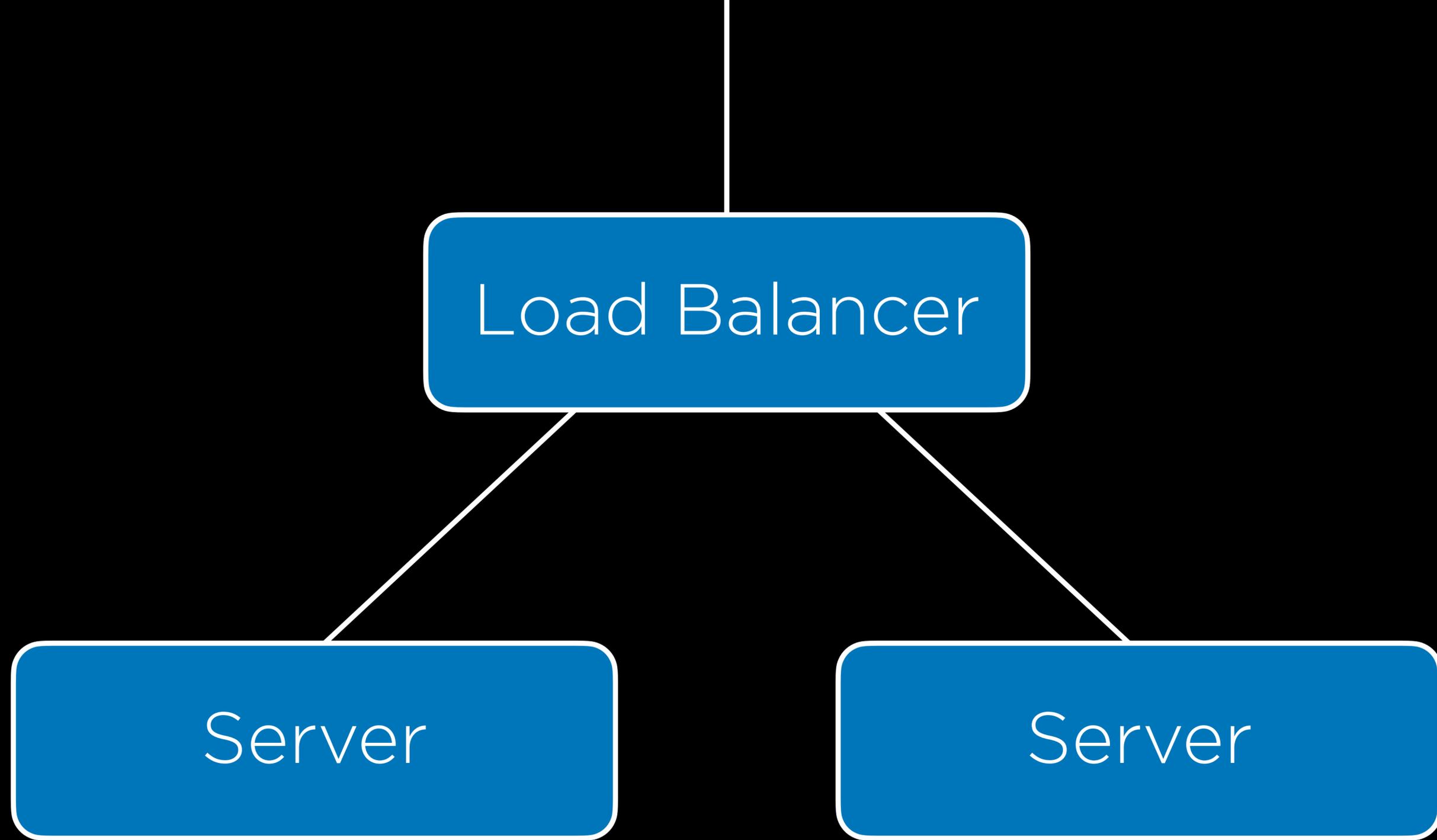
Server

# Horizontal Scaling

Server

Server

Server



# Load Balancing

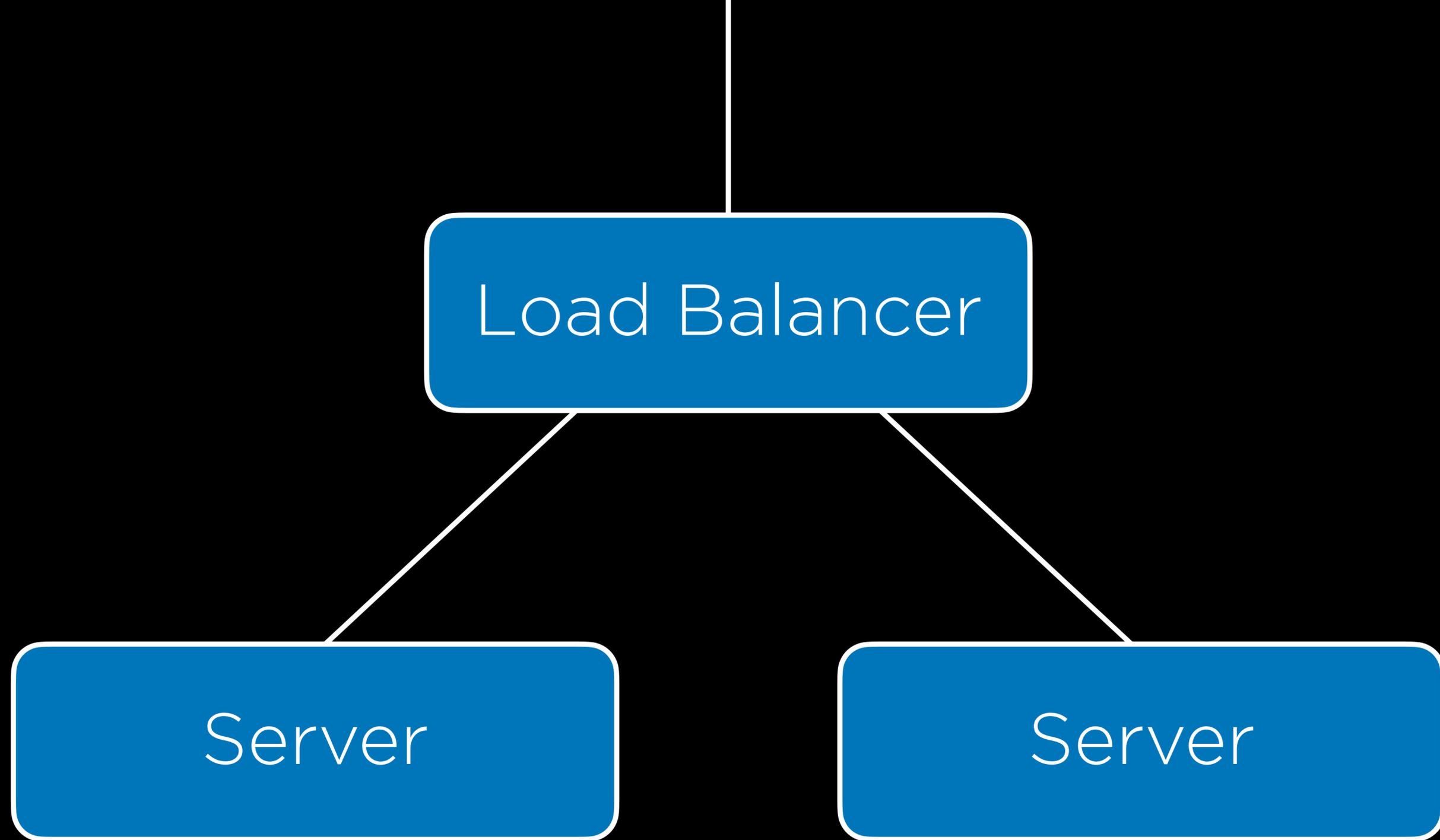
# Load Balancing Methods

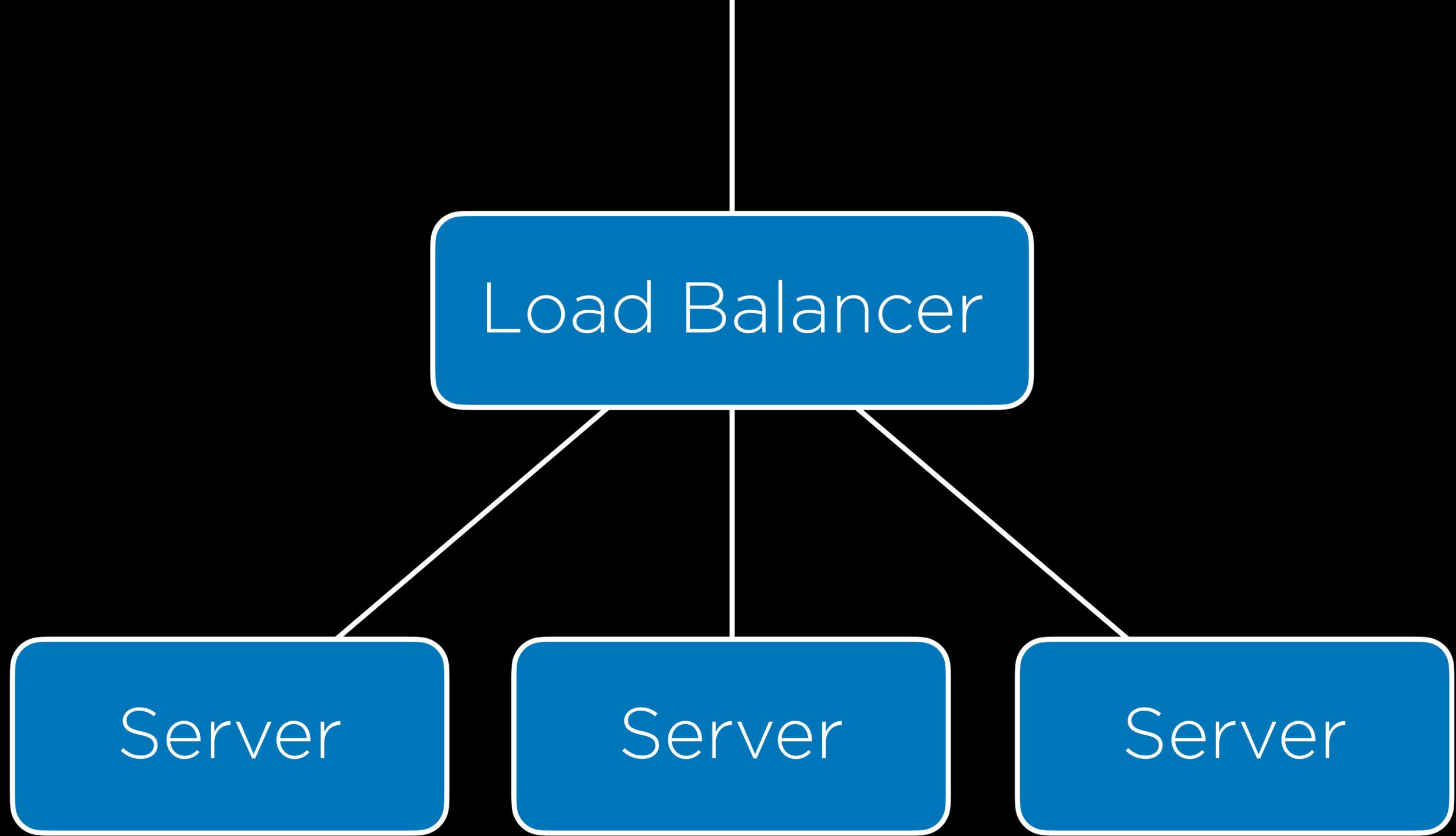
- Random Choice
- Round Robin
- Fewest Connections
- ...

# Session-Aware Load Balancing

- Sticky Sessions
- Sessions in Database
- Client-Side Sessions
- ...

# Autoscaling





```
graph TD; LB[Load Balancer] --- S1[Server]; LB --- S2[Server]; LB --- S3[Server]; LB --- S4[Server];
```

Load Balancer

Server

Server

Server

Server

```
graph TD; LB[Load Balancer] --- S1[Server]; LB --- S2[Server]; LB --- S3[Server]; LB --- S4[Server];
```

Load Balancer

Server

Server

Server

Server

```
graph TD; LB[Load Balancer] --- S1[Server]; LB --- S2[Server]; LB --- S3[Server]; LB --- S4[Server];
```

Load Balancer

Server

Server

Server

Server

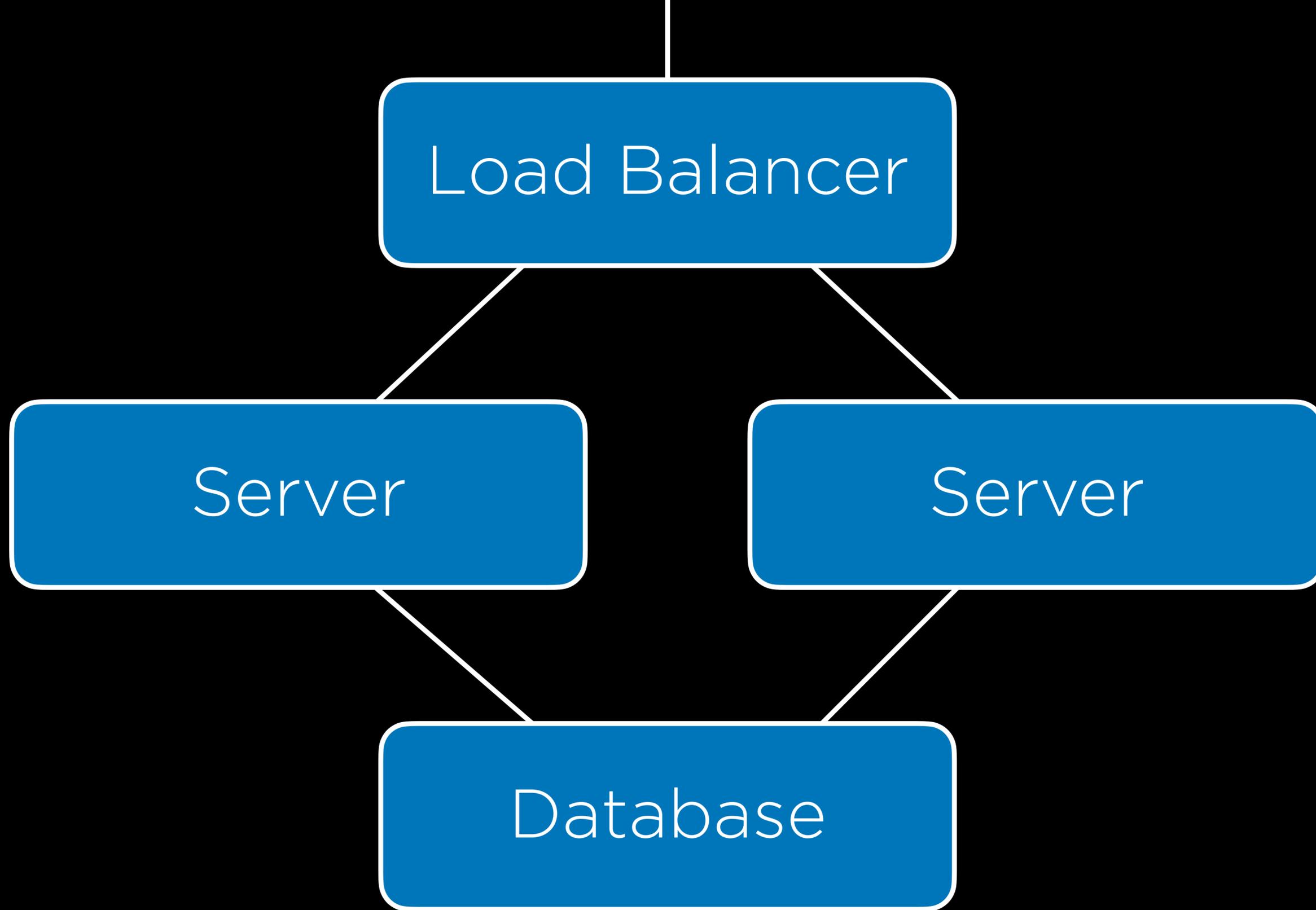
# Scaling Databases

Load Balancer

```
graph TD; LB[Load Balancer] --- S1[Server]; LB --- S2[Server];
```

Server

Server



# Database Partitioning

# flights

id	origin	origin_code	destination	destination_code	duration
1	New York	JFK	London	LHR	415
2	Shanghai	PVG	Paris	CDG	760
3	Istanbul	IST	Tokyo	NRT	700
4	New York	JFK	Paris	CDG	435
5	Moscow	SVO	Paris	CDG	245
6	Lima	LIM	New York	JFK	455

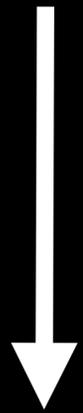
# airports

id	code	city
1	JFK	New York
2	PVG	Shanghai
3	IST	Istanbul
4	LHR	London
5	SVO	Moscow
6	LIM	Lima
7	CDG	Paris
8	NRT	Tokyo

# flights

id	origin_id	destination_id	duration
1	1	4	415
2	2	7	760
3	3	8	700
4	1	7	435
5	5	7	245
6	6	1	455

flights



flights\_domestic

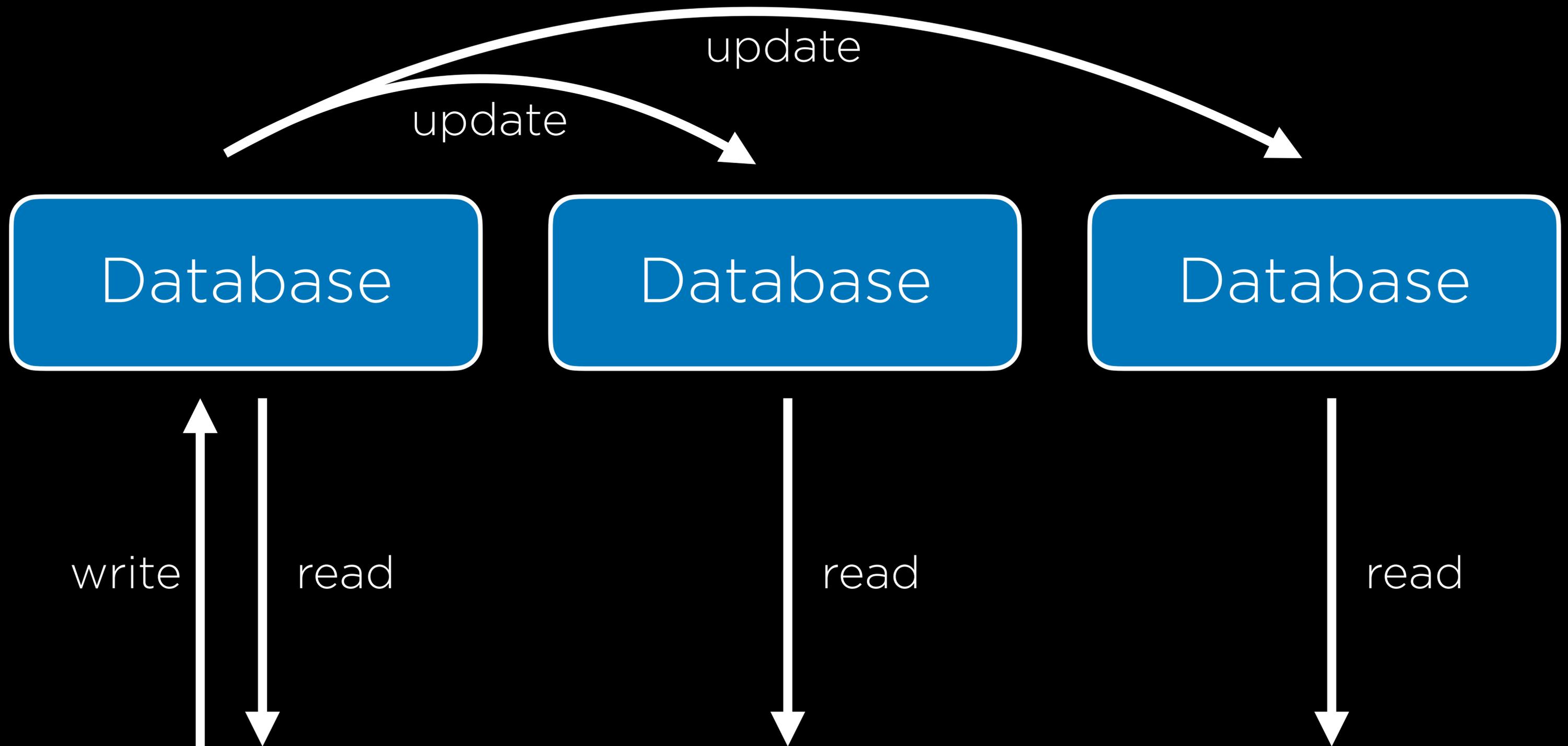
flights\_international

# Database Replication

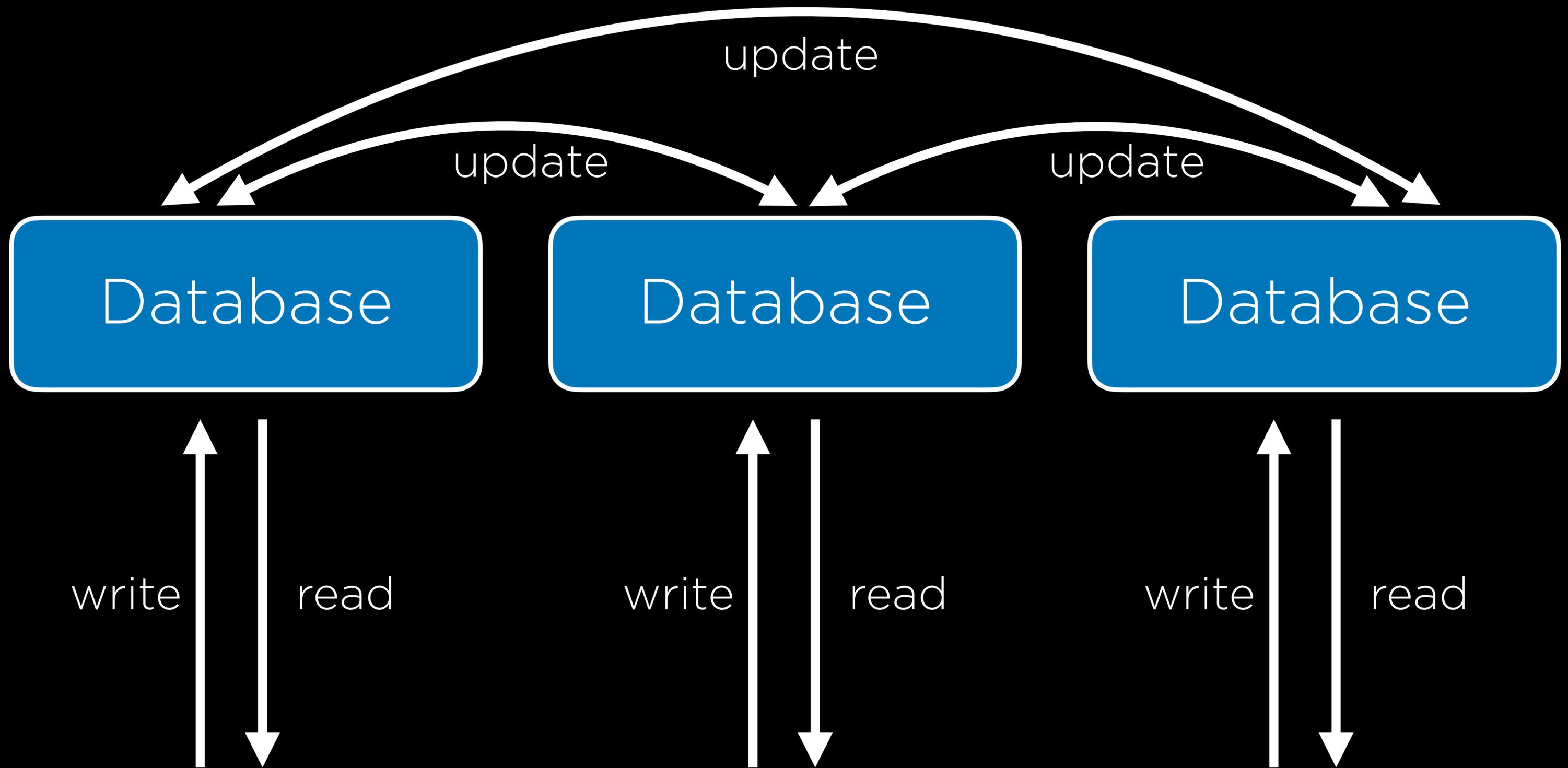
# Database Replication

- Single-Primary Replication
- Multi-Primary Replication

# Single-Primary Replication



# Multi-Primary Replication



# Caching

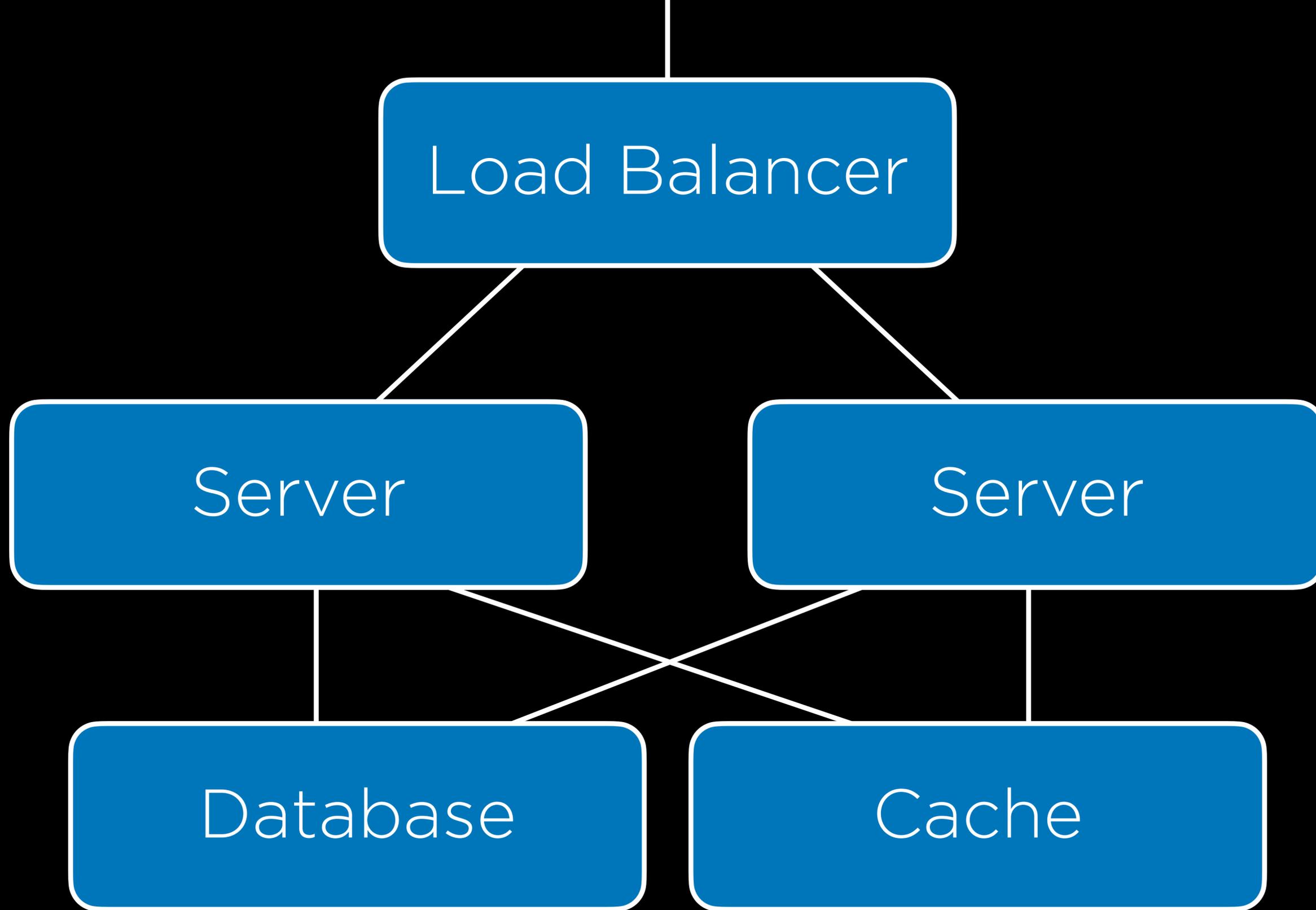
# Client-Side Caching

Cache-Control: max-age=86400

Cache-Control: max-age=86400

ETag: "7477656E74796569676874"

# Server-Side Caching



# Django Cache Framework

- Per-View Caching
- Template Fragment Caching
- Low-Level Cache API

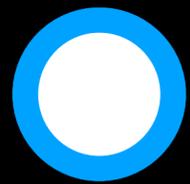
**Security**

# Topics

- HTML and CSS
- Git
- Python
- Django
- SQL
- JavaScript
- User Interfaces
- Testing and CI/CD

**Git**

# Open-Source Software

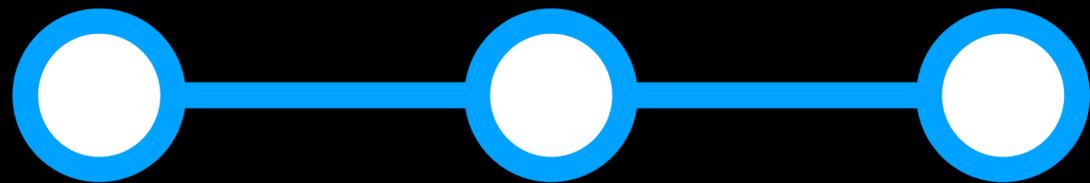


first  
commit



first  
commit

credentials  
exposed



first  
commit

credentials  
exposed

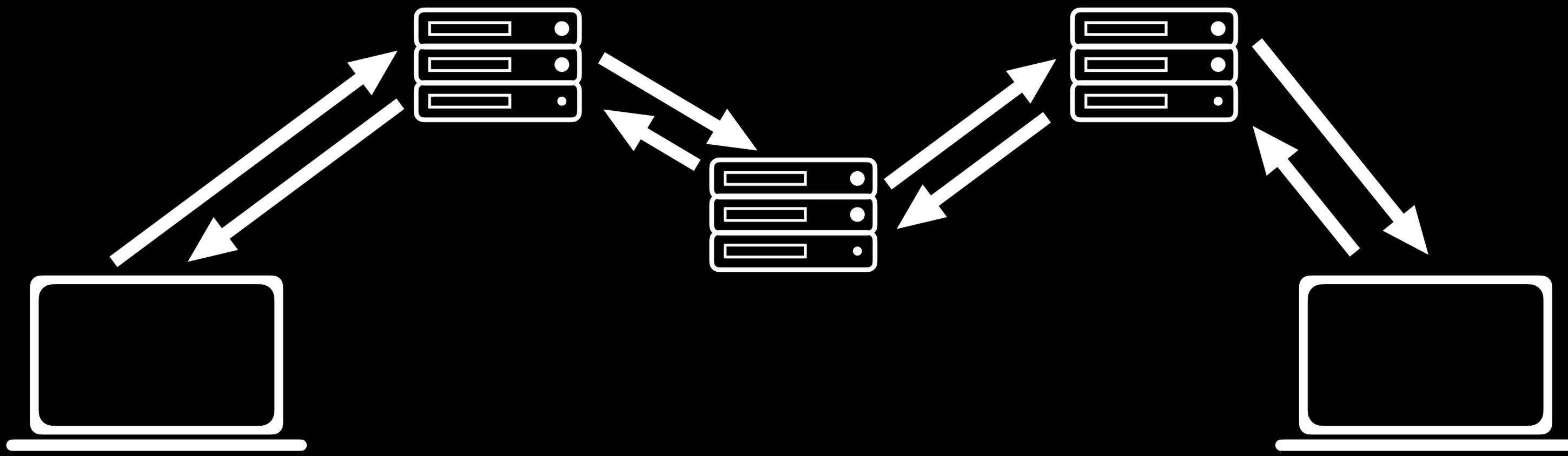
credentials  
removed

**HTML**

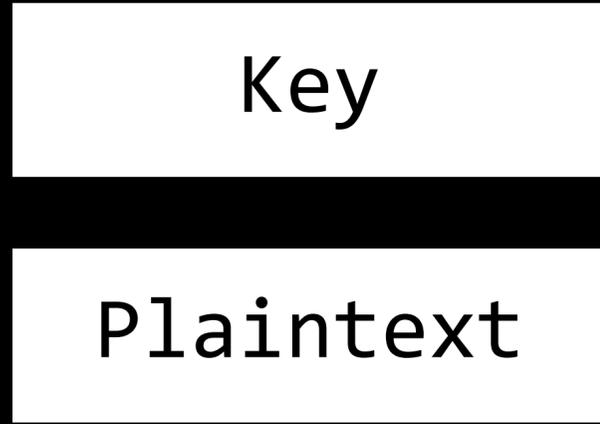
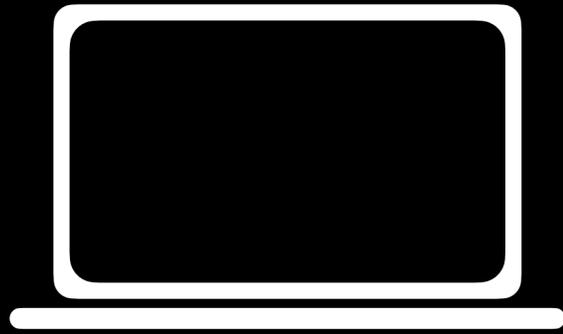
```
<a href="ur11">  
    ur12  
</a>
```

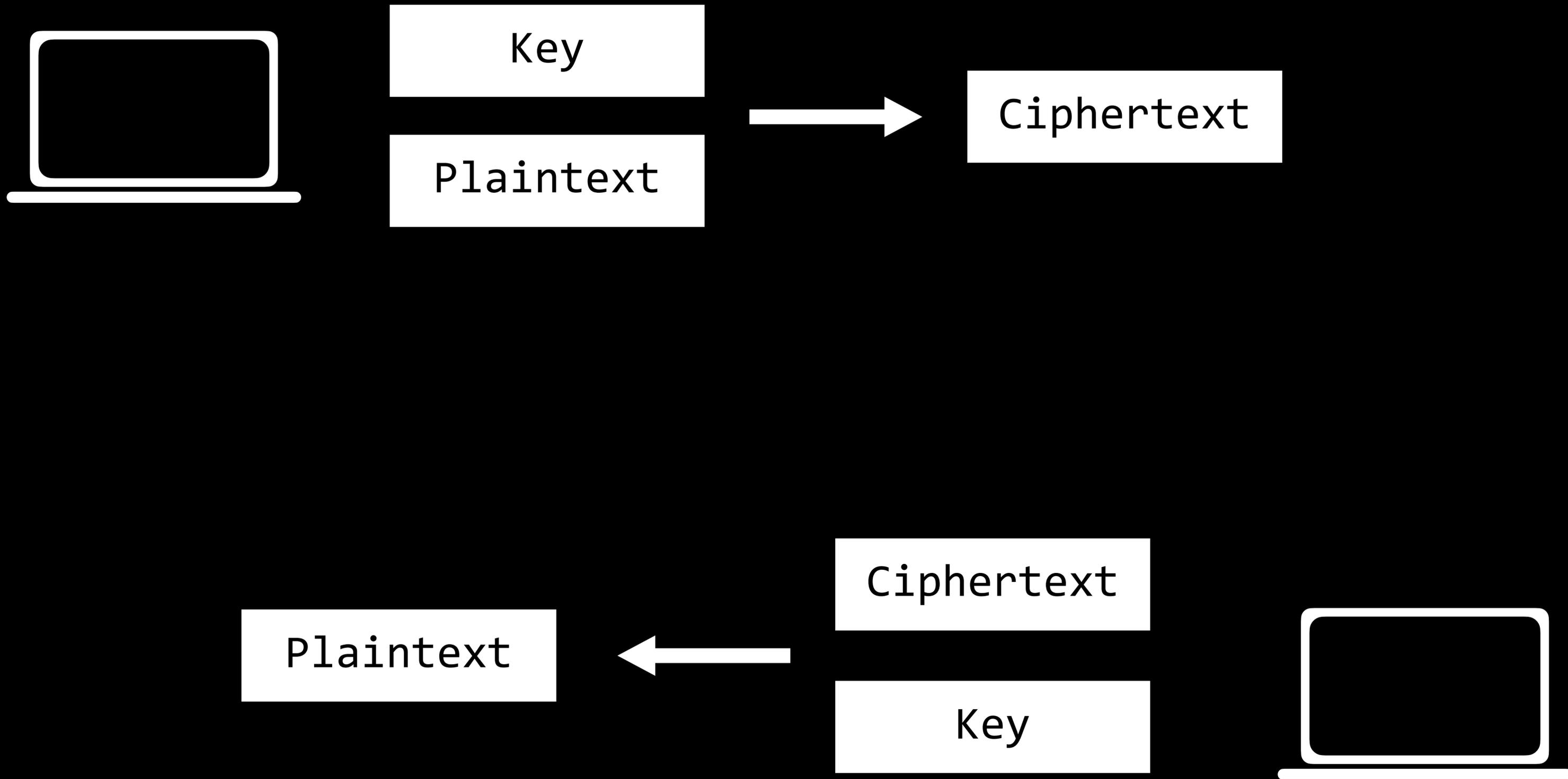
**Django**

# HTTP and HTTPS



# Secret-Key Cryptography

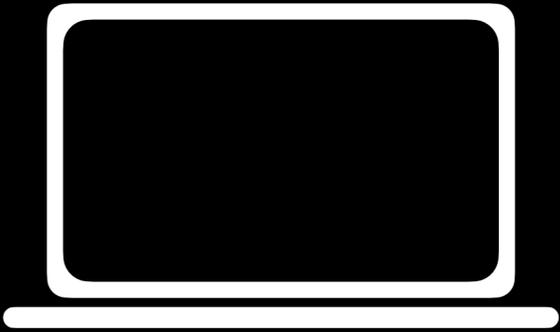




# Public-Key Cryptography

Public Key

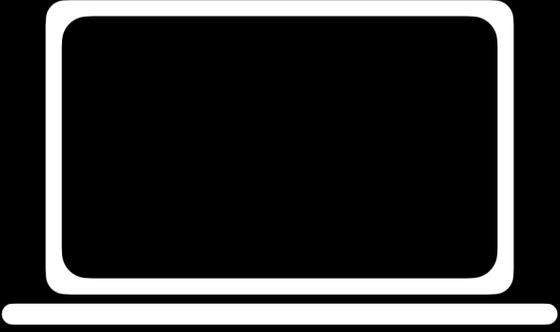
Private Key

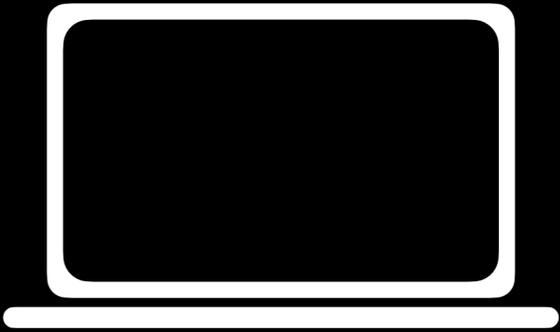


Plaintext

Public Key

Private Key





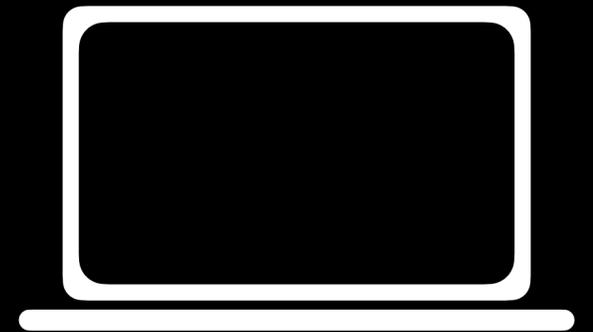
Plaintext

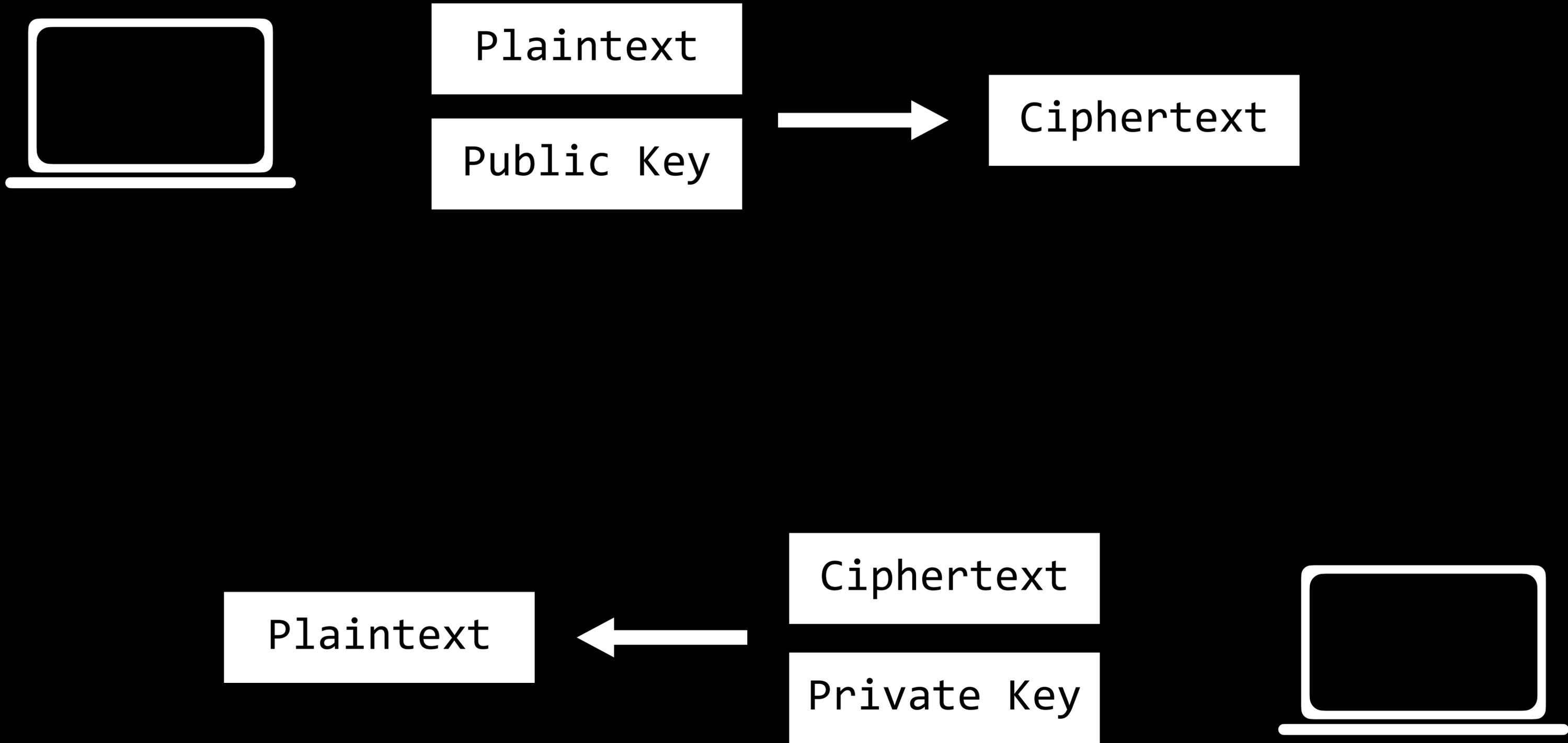
Public Key



Ciphertext

Private Key





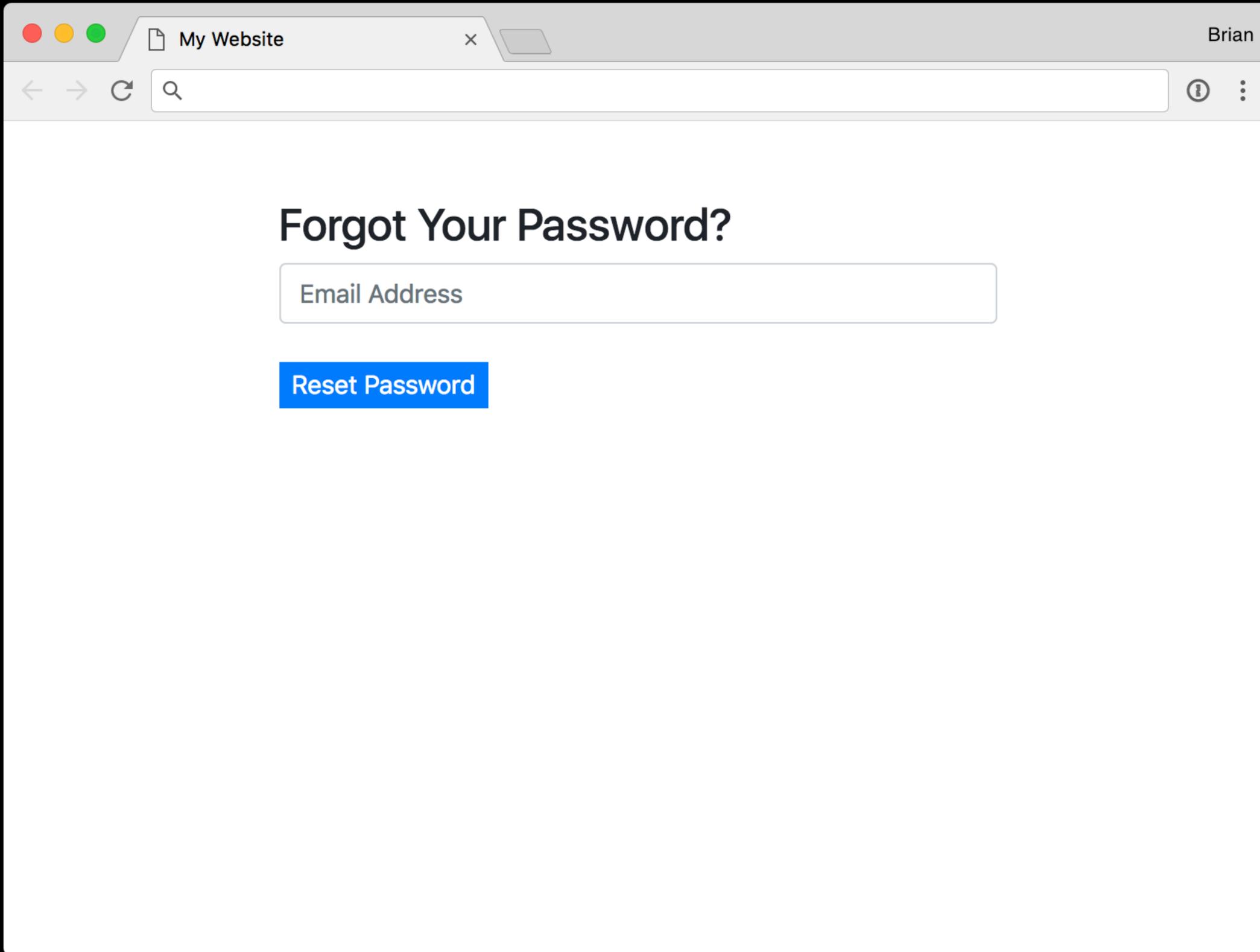
SQL

# users

id	username	password
1	harry	hello
2	ron	password
3	hermione	12345
4	ginny	abcdef
5	luna	qwerty

# users

id	username	password
1	harry	48c8e8c3f9e80b68ac67304c7c510e9fcb
2	ron	6024aba15e3f9be95e3c9e6d3bf261d78e
3	hermione	90112701066c0a536f2f6b2761e5edb09e
4	ginny	b053b7574c8a25751e2a896377e5d477c5
5	luna	a4048eaaee50680532845b2025996b44a9



My Website

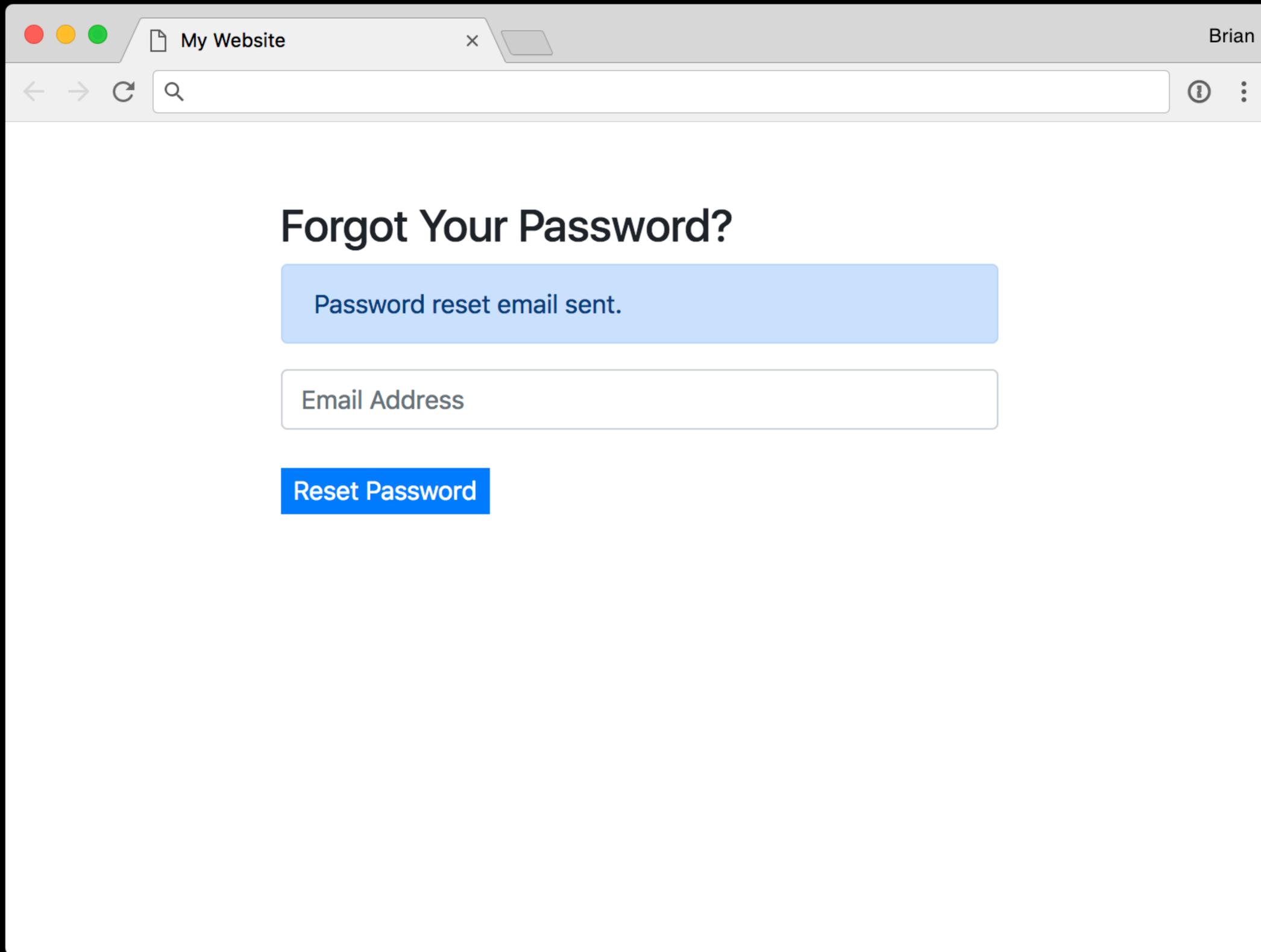


Brian



## Forgot Your Password?

[Reset Password](#)



My Website

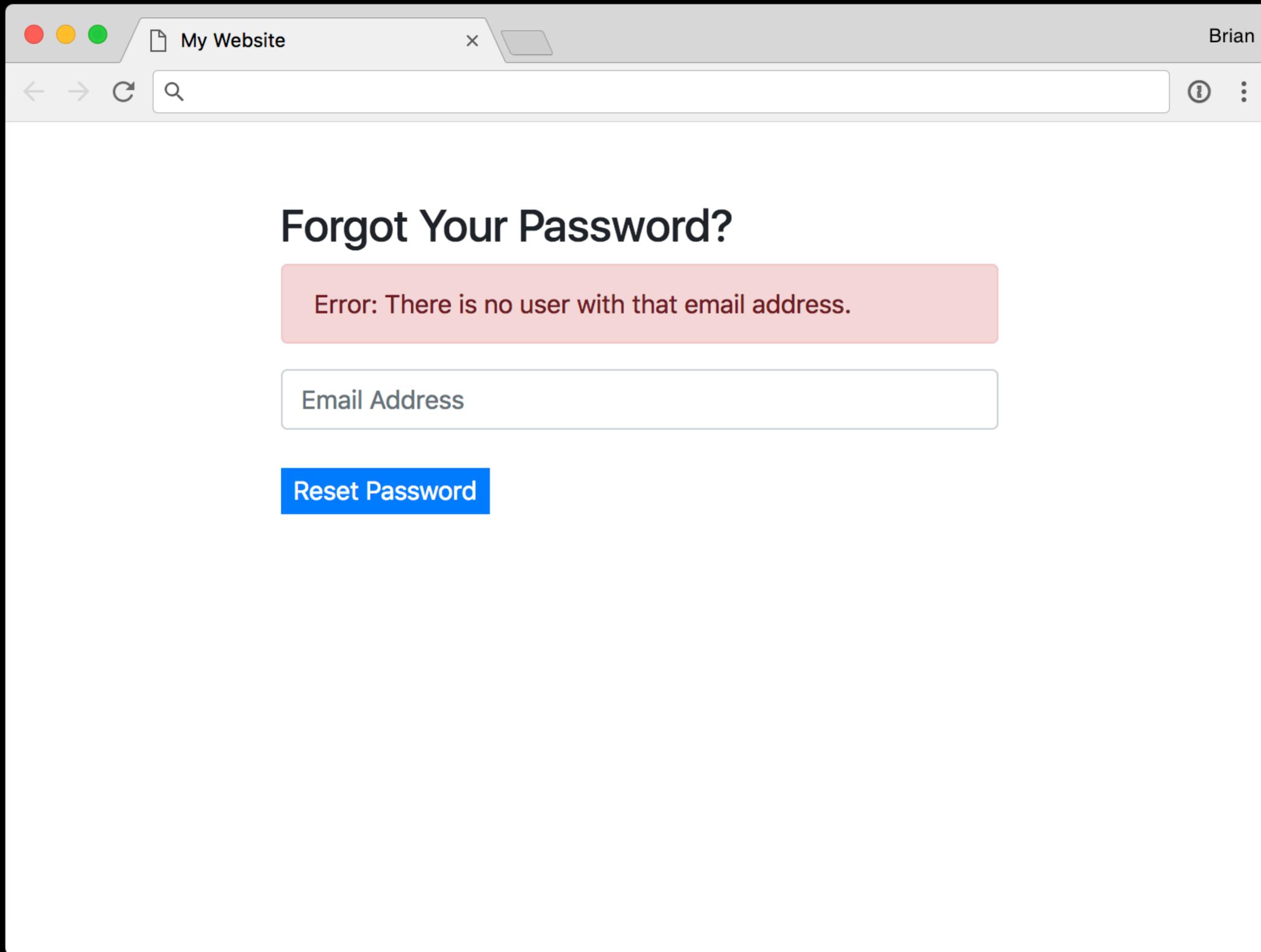
Brian



## Forgot Your Password?

Password reset email sent.

Reset Password



## Forgot Your Password?

Error: There is no user with that email address.

[Reset Password](#)

# SQL Injection

**Username:**

**Password:**

```
SELECT * FROM users  
WHERE username = username AND password = password;
```

**Username:**

harry

**Password:**

12345

```
SELECT * FROM users  
WHERE username = username AND password = password;
```

```
SELECT * FROM users  
WHERE username = "harry" AND password = "12345";
```

**Username:**

hacker" --

**Password:**

```
SELECT * FROM users  
WHERE username = username AND password = password;
```

```
SELECT * FROM users  
WHERE username = "hacker"--" AND password = "";
```

```
SELECT * FROM users  
WHERE username = "hacker" --" AND password = "";
```

**APIs**

# API Keys

- Rate Limiting
- Route Authentication

# Cross-Site Scripting

# Cross-Site Request Forgery

```
<body>  
  <a href="http://yourbank.com/transfer?to=brian&amt=2800">  
    Click Here!  
  </a>  
</body>
```

```
<body>  
    
</body>
```

```
<body>
  <form action="https://yourbank.com/transfer"
        method="post">
    <input type="hidden" name="to" value="brian">
    <input type="hidden" name="amt" value="2800">
    <input type="submit" value="Click Here!">
  </form>
</body>
```

```
<body onload="document.forms[0].submit()">
  <form action="https://yourbank.com/transfer"
    method="post">
    <input type="hidden" name="to" value="brian">
    <input type="hidden" name="amt" value="2800">
    <input type="submit" value="Click Here!">
  </form>
</body>
```

```
<form action="/transfer" method="post">
  {% csrf_token %}
  <input name="to" value="brian">
  <input name="amt" value="2800">
  <input type="submit" value="Transfer">
</form>
```

# Web Programming

**What's next?**

# Other Web Frameworks

- Server-Side
  - Express.js
  - Ruby on Rails
  - ...
- Client-Side
  - AngularJS
  - React
  - Vue.js
  - ...

# Deploying Websites

- Amazon Web Services
- GitHub Pages
- Google Cloud
- Heroku
- Microsoft Azure
- ...

# Web Programming

**HTML and CSS**

**HTML**



**Git**



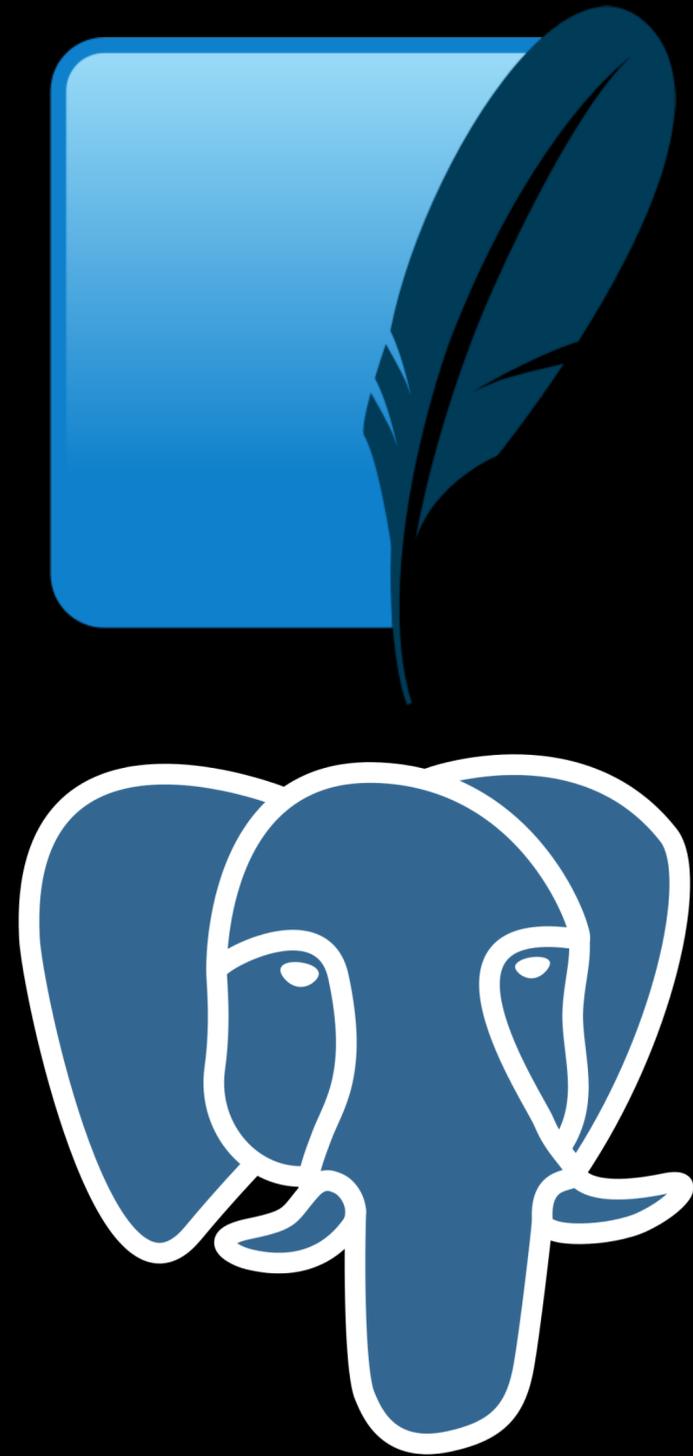
Python



Django

django

# SQL, Models, and Migrations

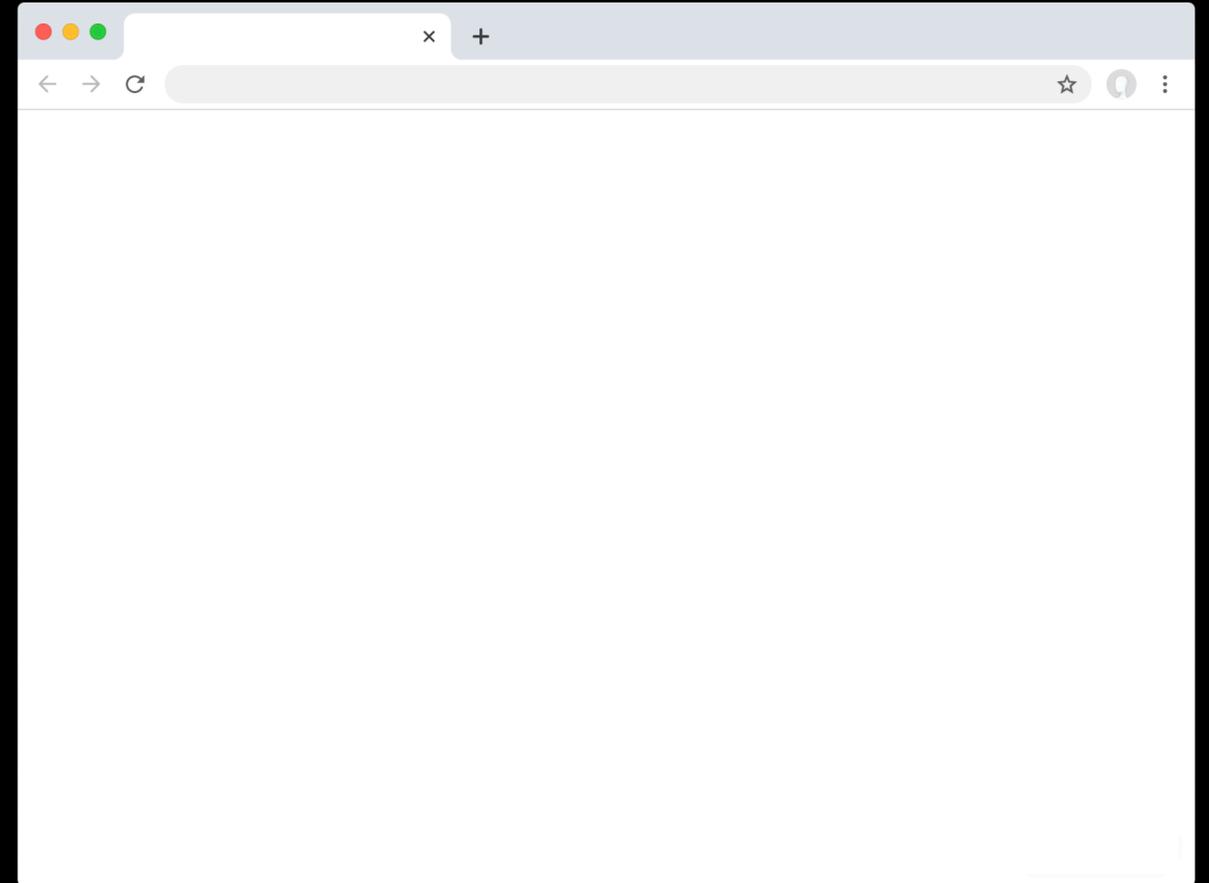


**JavaScript**

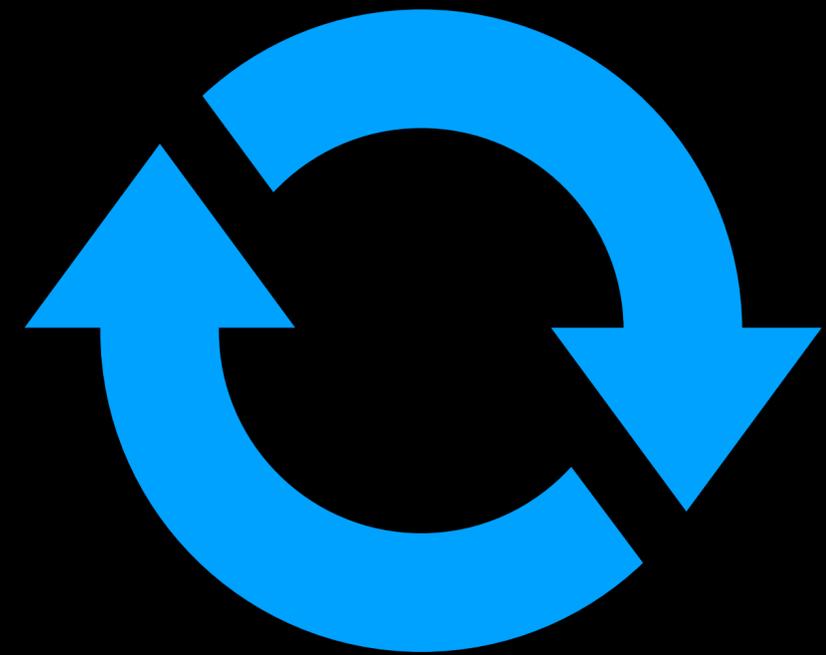
A yellow square containing the letters 'JS' in a bold, black, sans-serif font.

**JS**

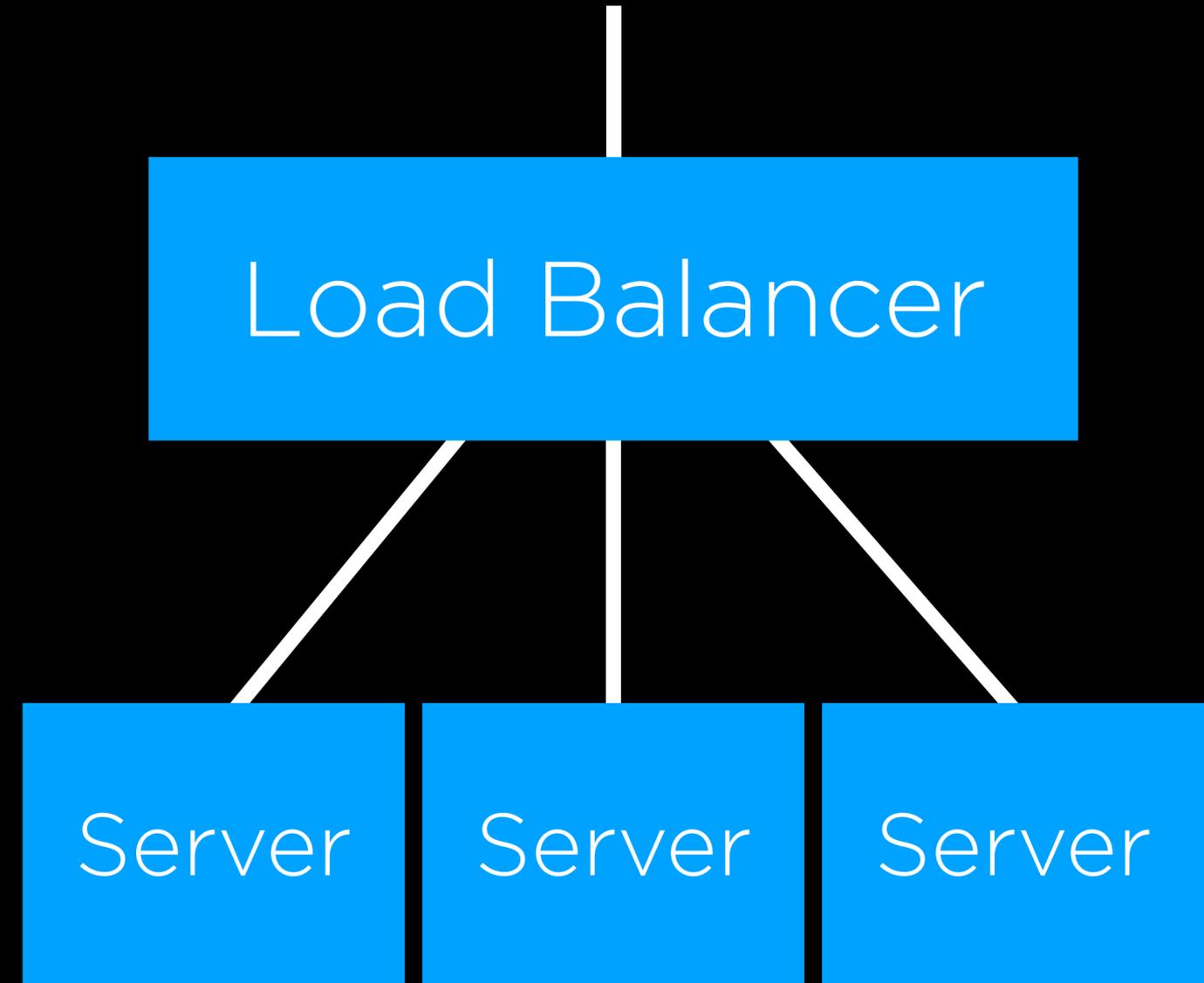
# User Interfaces



Testing and CI/CD



# Scalability and Security



# Web Programming

with Python and JavaScript