Monitoring (Prometheus, Grafana) ensures 99.99% availability, with multi-region disaster recovery for failover."

This **FASTER framework** ensures you always provide **structured, detailed, and impactful** answers. 🚀 Let me know if you want **mock questions to practice!** 😊

Not like this step by step guide from calculating requests and load on servers, how many servers or pods, what are cpus and rams etc whichever required from start to end pls explain

To estimate **infrastructure requirements** for a **highly available, zero-downtime payment system** like Google Pay, we will calculate:

1. **Total Transactions Per Second (TPS)**
2. **Latency per Transaction**
3. **CPU Requirements**
4. **Memory (RAM) Requirements**
5. **Storage & Database Estimations**
6. **Network Bandwidth**
7. **Final Deployment Plan (Pods, Nodes, Load Balancers)**

Step 1: Define Workload & Transactions Per Second (TPS)

Assumptions:

- **1 billion users worldwide**.
- **10 million daily active users (DAU)**.
- **Each active user does 5 transactions/day**.
- **Peak hours = 4 hours/day**, so traffic spikes within this window.

Calculate TPS:

$$\text{Total transactions per day} = 10M \times 5 = 50M$$

$$\text{Average TPS} = \frac{50M}{24 \times 3600} = \frac{50M}{86400} \approx 578 \text{ TPS}$$

$$\text{Peak TPS} = \frac{50M}{4 \times 3600} = \frac{50M}{14400} \approx 3472 \text{ TPS}$$

> ✅ **We need to design the system for peak TPS = 3500 TPS.**

Step 2: Estimate Latency Per Transaction

Each transaction involves:

1. **API Call → Gateway (5ms)**
2. **Authentication Check (10ms)**
3. **Balance Check (Redis, 2ms)**
4. **Transaction Processing (Database Commit, 40ms)**
5. **Fraud Detection (Async Call, 30ms)**
6. **Notification Trigger (5ms)**

Total Transaction Time (Latency)

$$5 + 10 + 2 + 40 + 30 + 5 = 92ms$$

> ✅ **Total time per transaction: ~100ms**

Step 3: Estimate CPU Requirements

Each CPU can handle **25 requests/sec** (avg based on benchmarking).

$$\text{Total CPUs required} = \frac{3500 \text{ TPS}}{25} = 140 \text{ vCPUs}$$

Each server has **8 vCPUs**, so:

$$\text{Total servers required} = \frac{140}{8} = 18 \text{ servers} (roundedup)$$

> ✅ **We need at least 18 application servers (pods).**

Step 4: Estimate Memory (RAM) Requirements

Each transaction requires:

- **100KB request processing**
- **150KB Redis lookup**
- **250KB database write**

Total memory usage per transaction:

$$100 + 150 + 250 = 500KB$$

Total memory per second:

$$3500 \times 500KB = 1.75GB/sec$$

Each server has **32GB RAM** and handles **300 TPS**.

$$\text{Total RAM required} = \frac{1.75 \times 3600}{32} = 197 \text{ GB} \approx 200 \text{ GB}$$

> ✅ **We need at least 200GB RAM across all servers.**

Step 5: Estimate Database Requirements

- **PostgreSQL for transactions**
- **Sharded into 10 database instances**
- **Redis for caching (low-latency reads)**

Each transaction stores **500 bytes**.

$$\text{Total data per day} = 50M \times 500B = 25GB$$

Total storage required for **1-year retention**:

$$25GB \times 365 = 9.1TB$$

> ✅ **We need 10 PostgreSQL shards, each with 1TB storage.**

Step 6: Estimate Network Bandwidth

Each request is **1MB (including API calls, DB, logs, response)**.

$$\text{Total bandwidth per second} = 3500 \times 1MB = 3.5GB/sec$$

For **4 hours of peak traffic**:

$$3.5GB \times 14400 = 50.4TB$$

> ✅ **We need a network capacity of ~50TB per day.**

Final Deployment Plan

Application Servers (Microservices)

Component	Servers/Pods	vCPUs	RAM (GB)
API Gateway	4 Pods	8	32
Authentication	6 Pods	24	48
Payment Processing	18 Pods	140	200
Fraud Detection	6 Pods	24	48
Notification	4 Pods	8	32
Total	**38 Pods**	**200**	**360GB**

Database & Caching

Component	Shards	Storage (TB)	Memory (GB)
PostgreSQL	10	10TB	320
Redis Cache	4	2TB	128

High Availability & Scaling Plan

- **Auto-scaling (Kubernetes HPA)**: Scale pods up/down based on load.
- **Load Balancer (AWS ALB/NGINX)**: Distributes traffic to services.
- **Multi-Region Disaster Recovery**: Active-active deployment across data centers.
- **Blue-Green Deployment**: Ensures zero downtime upgrades.

Final Infrastructure Summary

- **Compute**: 38 microservices pods (200 vCPUs, 360GB RAM).
- **Database**: 10 PostgreSQL shards (10TB storage).
- **Cache**: 4 Redis nodes (2TB storage, 128GB RAM).
- **Network**: 50TB/day traffic capacity.
- **Storage**: 9.1TB/year data growth.

This **architecture ensures zero downtime, auto-scaling, and 99.99% availability** for millions of transactions daily.

🚀 Key Takeaways

1. **Calculate TPS (Transactions Per Second).**
2. **Measure time per transaction.**
3. **Compute CPU, RAM, storage, and network bandwidth.**

4. **Use microservices + Kubernetes for auto-scaling.**
5. **Ensure high availability with load balancers & multi-region databases.**

This structured **step-by-step method** ensures you **can answer any system design question confidently**! 💡🎯

I like the answer, pls make it as pdf downloadable

It looks like there was an encoding issue with special characters in the PDF. I'll fix it and generate a new downloadable version.

You can download the PDF with detailed infrastructure estimation here:
Download PDF