LEARNINGS FROM SCALING IDP

HOW TO HANDLE MILLIONS OF REQUESTS A DAY

miniOrange

Pratish Ray
2024

ABOUT ME

- IAM Expert

- 9 Years Of Experience in Cyber Security

- Scripting Aficionado

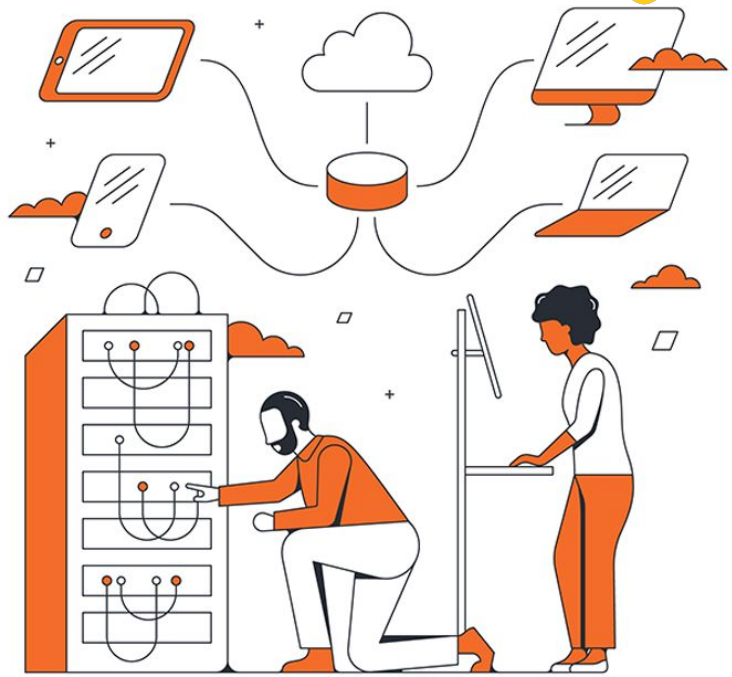
- Curious Cat

IdentityShield

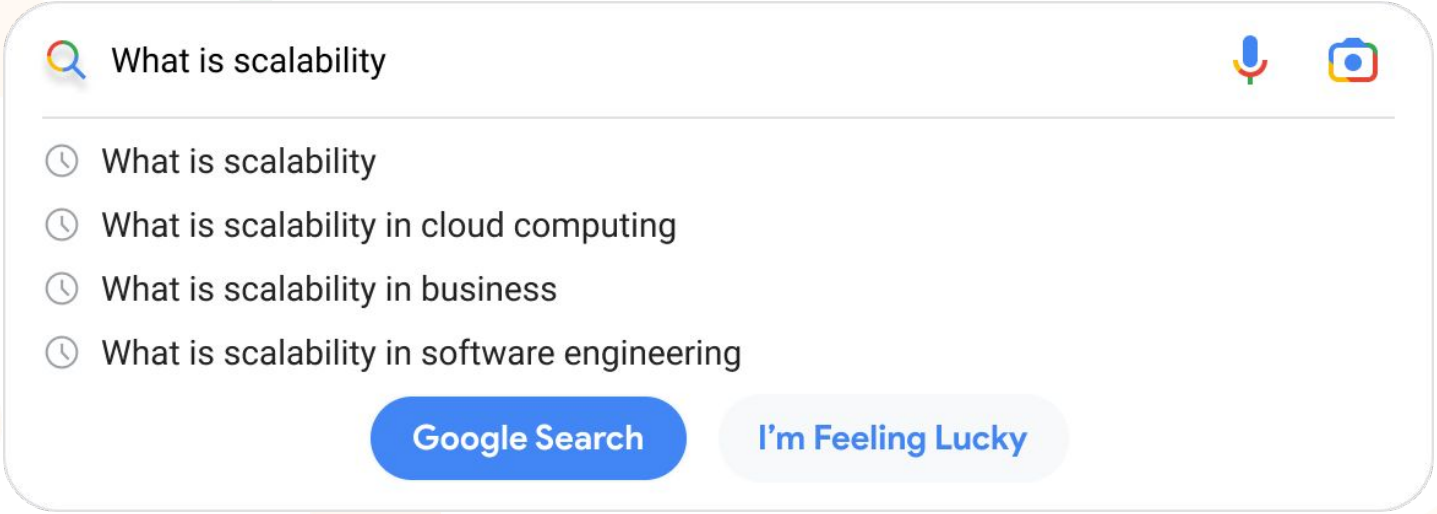
miniOrange

WHY DO YOU NEED TO SCALE?

- HANDLING INCREASED LOAD
- AVAILABILITY OF THE SERVICE
- RELIABILITY OF THE SERVICE
- GLOBAL EXPANSION



SO WHAT IS SCALABILITY?



IdentityShield

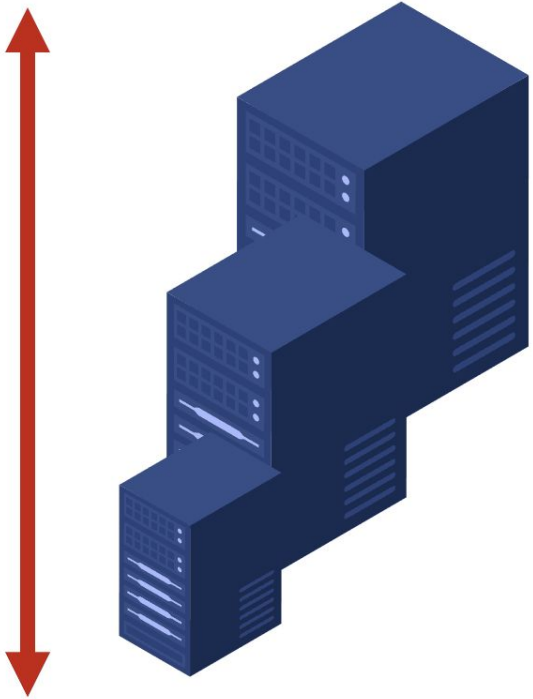
"Cloud scalability in cloud computing refers to increasing or decreasing IT resources as needed to meet changing demand."

miniOrange

TYPES OF SCALING

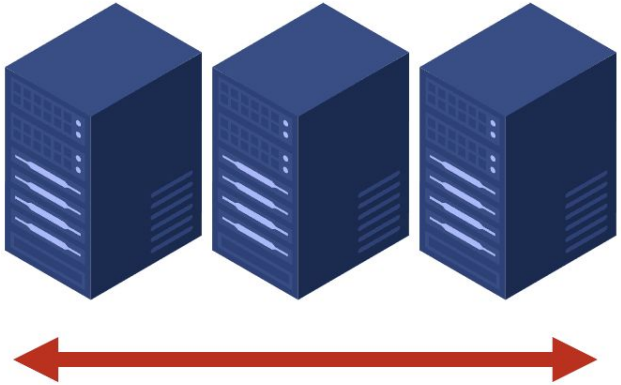
Vertical Scaling

Increase or decrease the capacity of existing services/instances.



Horizontal Scaling

Add more resources like virtual machines to your system to spread out the workload across them.

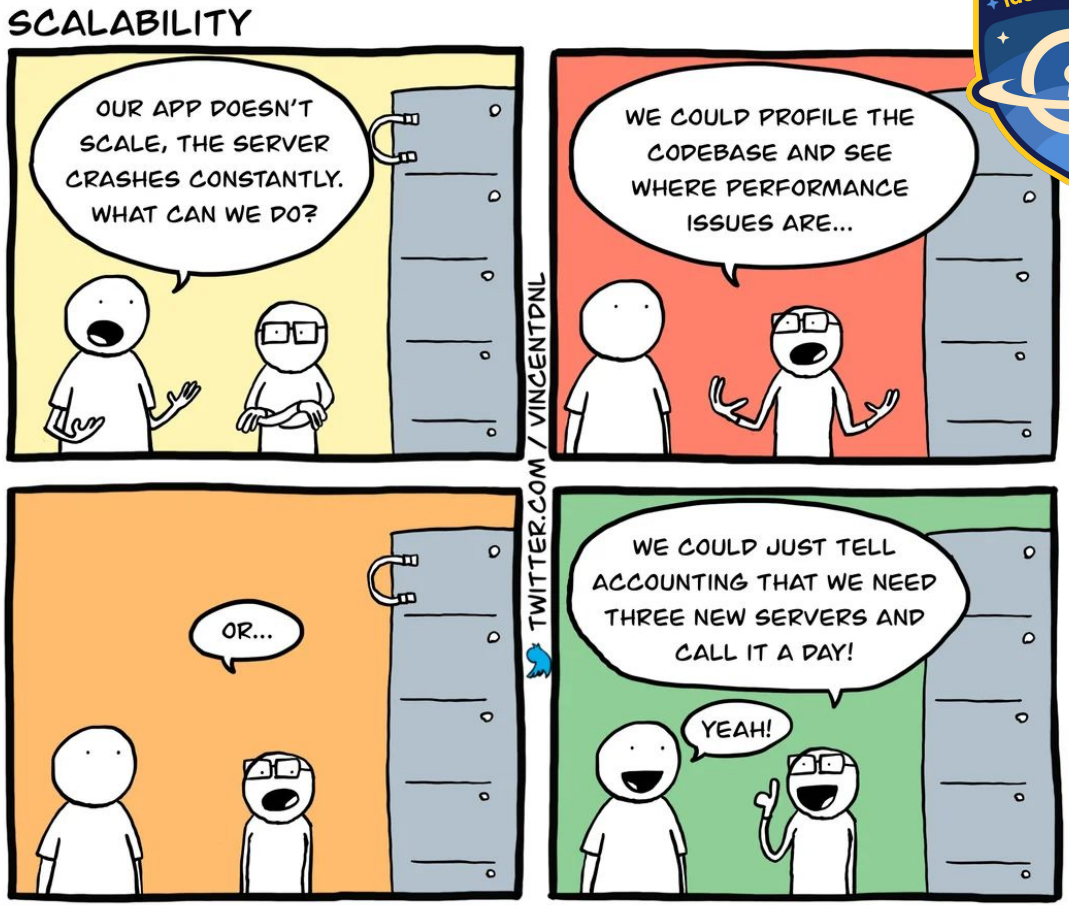


IdentityShield

SO LET'S JUST ADD MORE SERVERS AND IT WILL SOLVE EVERYTHING

miniOrange

- IS THAT SIMPLE?
- WHAT ABOUT COST?
- IS MORE BETTER?



WHAT SCALING LOOKS LIKE THROUGH THE LENS OF miniOrange

OKAY LET'S JUMP DOWN THE RABBIT HOLE

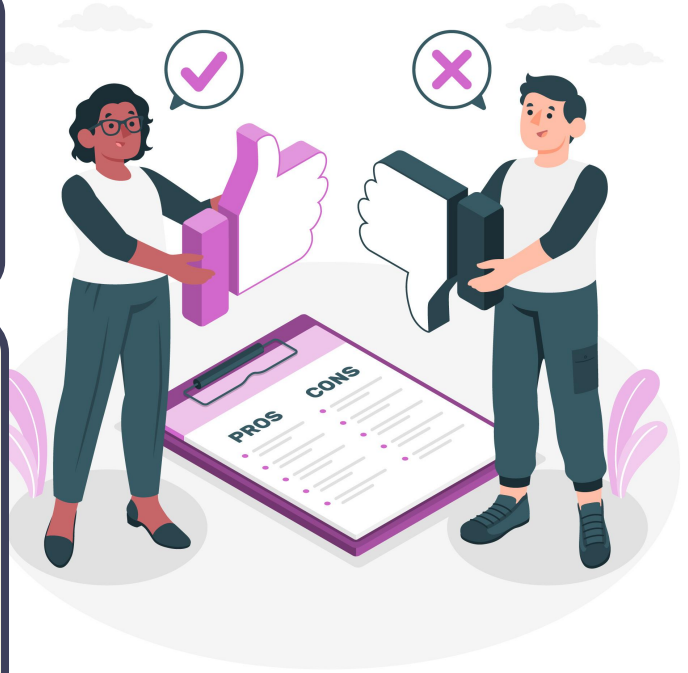
HORIZONTAL & VERTICAL SCALING

THE UPSIDE

- Quick fix and provides immediate relief.

THE PITFALL

- Temporary Relief
- Increased Costs
- Can lead to downtime during migration



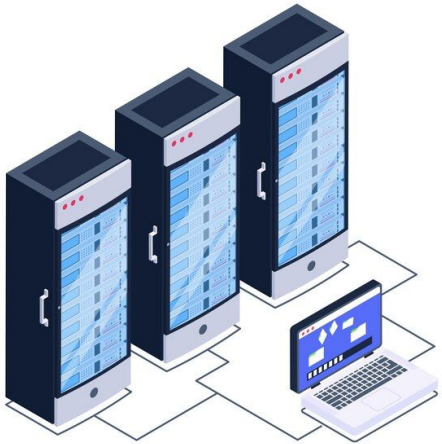
WHAT ARE THEY NOT TALKING ABOUT

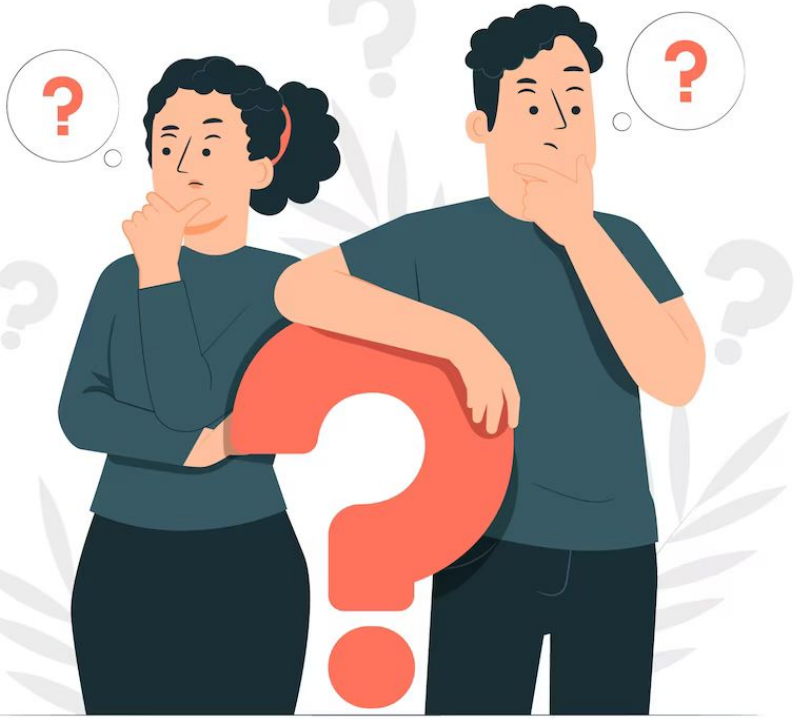
- WHICH SERVER TO SCALE?
- HOW TO DISTRIBUTE LOAD?
- APPLICATION SUPPORT FOR HORIZONTAL SCALING
- MONITORING & MAINTENANCE
- HOW DO YOU EVEN SCALE?

IdentityShield

miniOrange

01 WHAT TO SCALE?

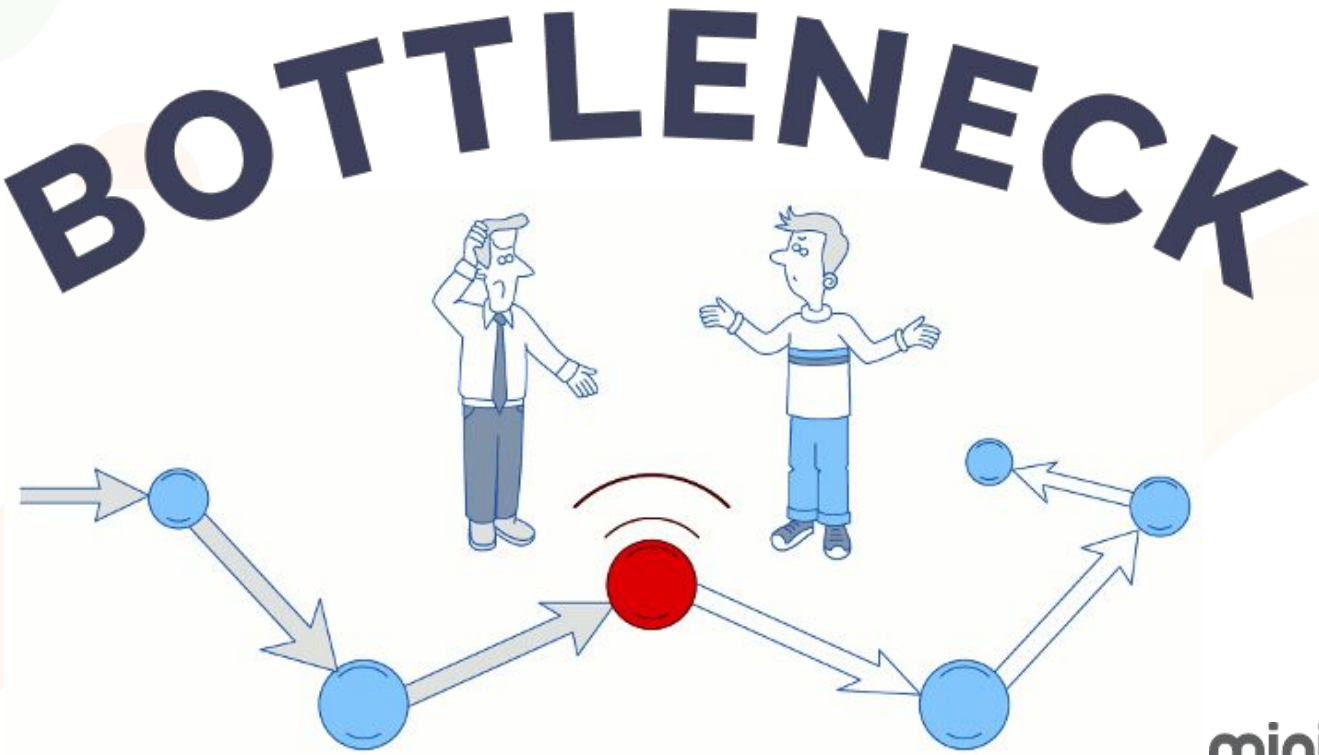




APPLICATION SERVER

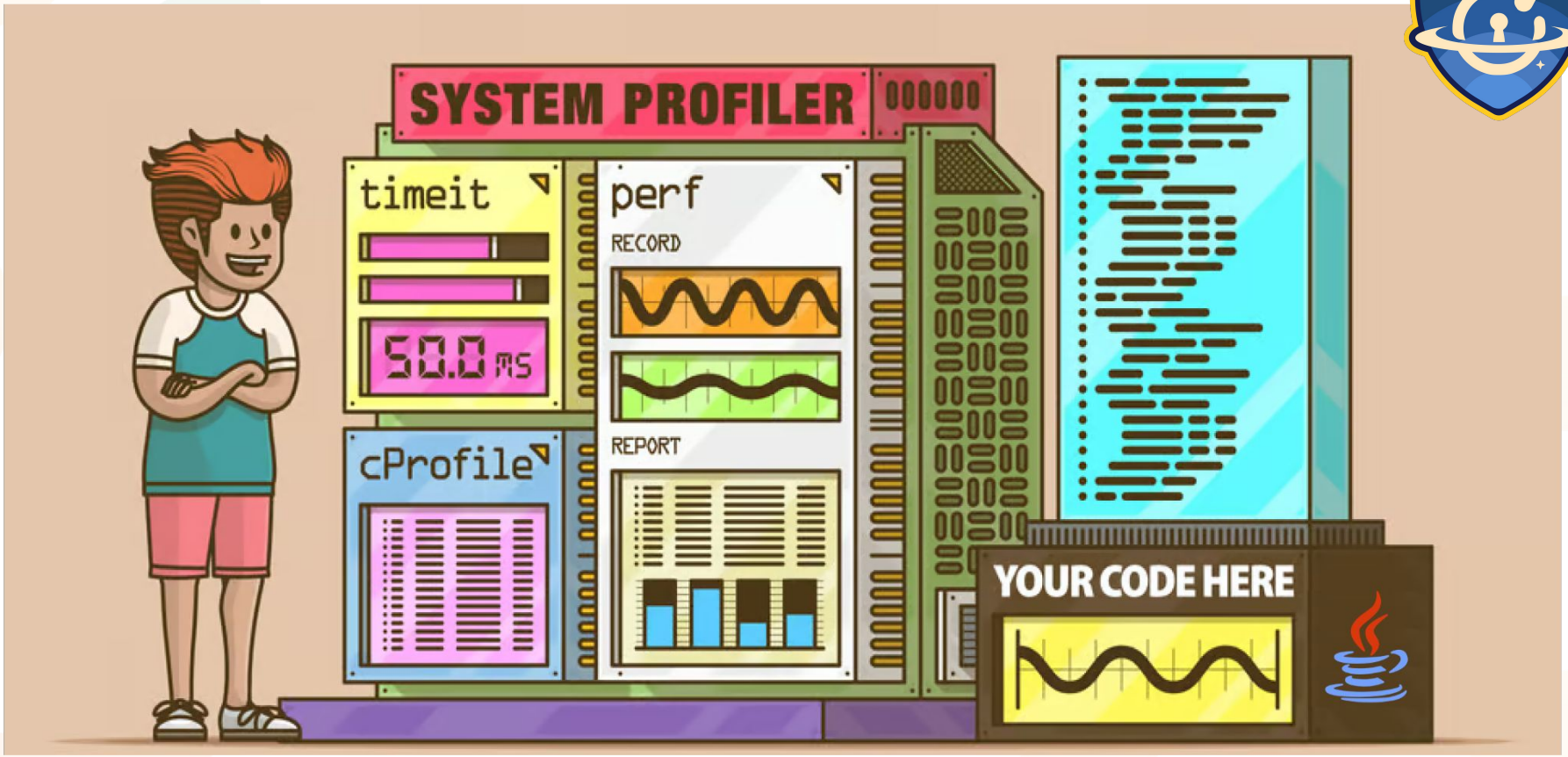
DATABASE SERVER

KEY WORD IS ....



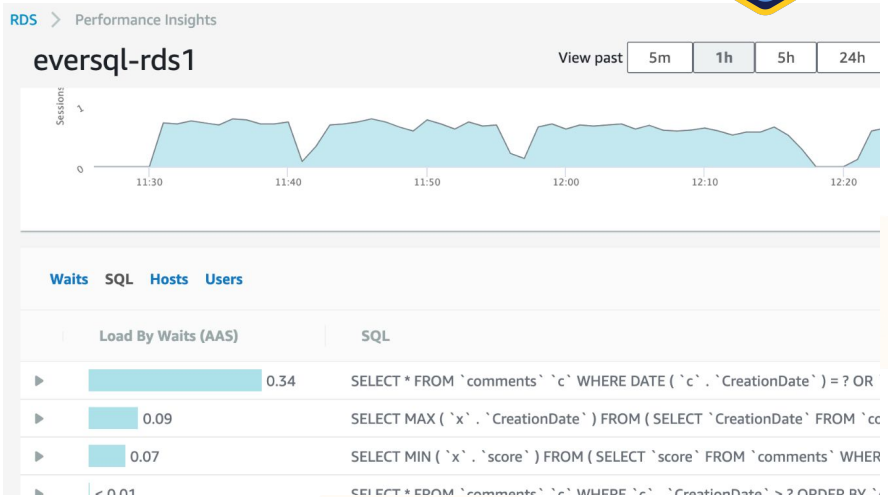
IdentityShield

miniOrange



QUERY OPTIMIZATION & INDEXING

- Application Monitoring Tool
- EXPLAIN ANALYZE
- Denormalization
- Indexing




```
bigdb=# explain analyze select * from traffic where serial_id = 1;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..389406.77 rows=1 width=1908) (actual time=13022.771..13024.689 rows=1 loops=1)
   Workers Planned: 2
   Workers Launched: 2
   ->  Parallel Seq Scan on traffic  (cost=0.00..388406.67 rows=1 width=1908) (actual time=11691.284..13012.262 rows=0 loops=3)
         Filter: (serial_id = 1)
         Rows Removed by Filter: 507306
 Planning Time: 0.291 ms
 Execution Time: 13024.774 ms = 13 seconds
```

TIPS AND TRICKS

- align your where clauses based on your indexes otherwise it will result in a full table scan and cause performance bottlenecks
- do not use %% in like query. It gives up checking indexes. Use % at the end of your string
- avoid OR and IN queries where you can. Better to use UNION or BETWEEN instead
- ORDER BY and WHERE should align so that indexes can be used.

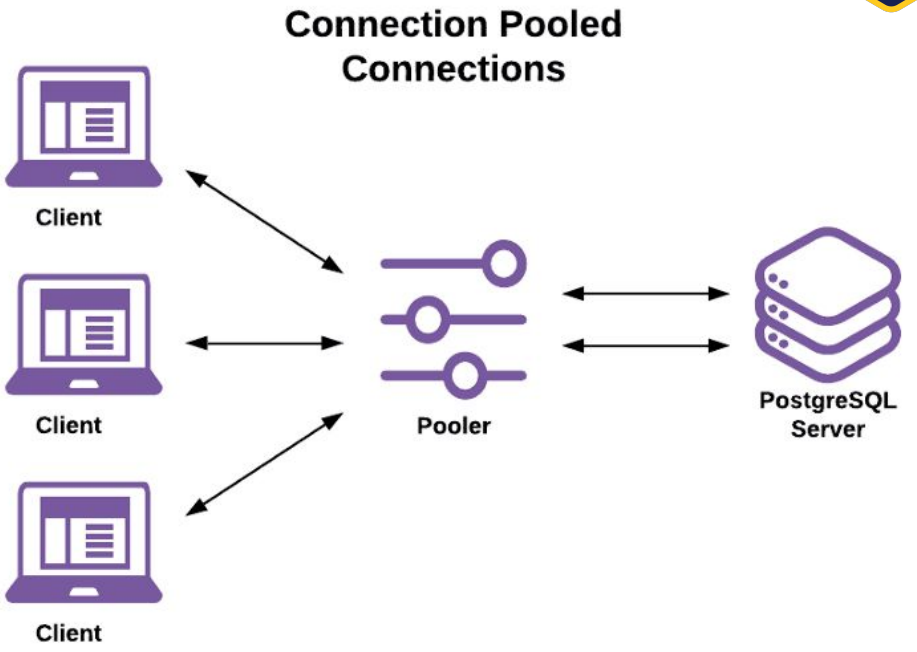
TIPS AND TRICKS

- UNION ALL vs UNION. UNION ALL is faster as it doesn't remove duplicates whereas UNION does.
- Your INDEXES should be determined based on your where clause.
- FOR JOIN put the larger tables first. For MYSQL put smaller tables first.

CONNECTION POOLING

LIFECYCLE

1. Opening a connection to the database using the database driver
2. Opening a TCP socket for reading/writing data
3. Reading / writing data over the socket
4. Closing the connection
5. Closing the socket



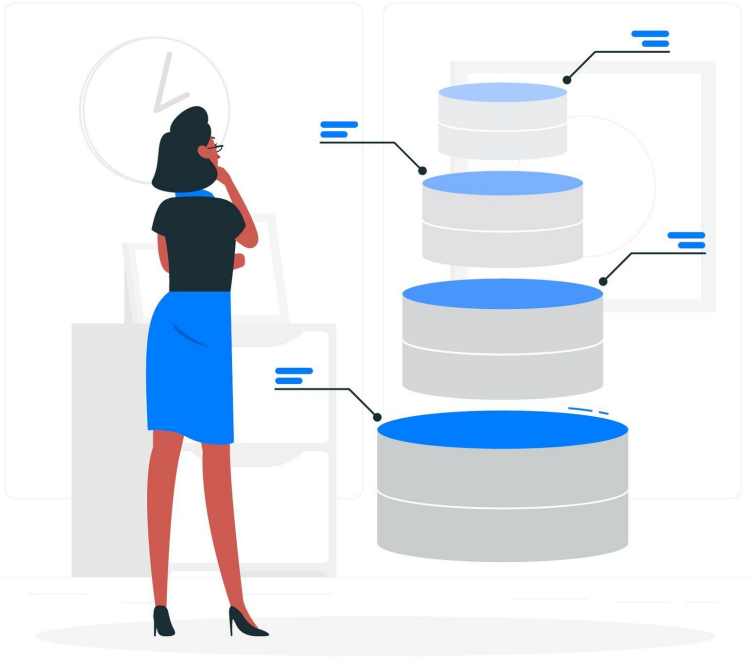
POPULAR POOLING LIBRARIES

Library	Language	Features
HikariCP	Java	High performance, lightweight, reliability
c3p0	Java	Highly configurable, robust recovery features
SQLAlchemy Pool	Python	Supports multiple strategies, integrated with ORM
node-postgres Pool (pg)	Node	Simple client pooling, automatic management

HOW TO CALCULATE MAXIMUM NUMBER OF CONNECTIONS?

$$\text{Max No of Connections} = \frac{\text{Total Available RAM} - \text{Base Memory Usage}}{\text{Estimated Memory per Connection}}$$

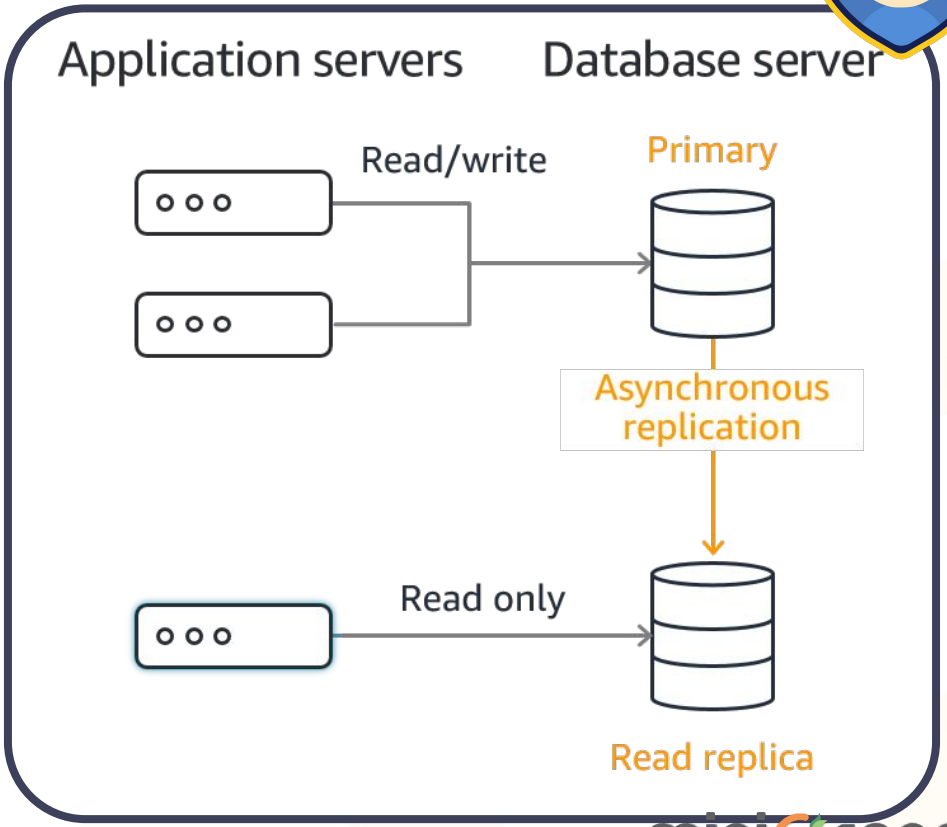
VERTICAL SCALING



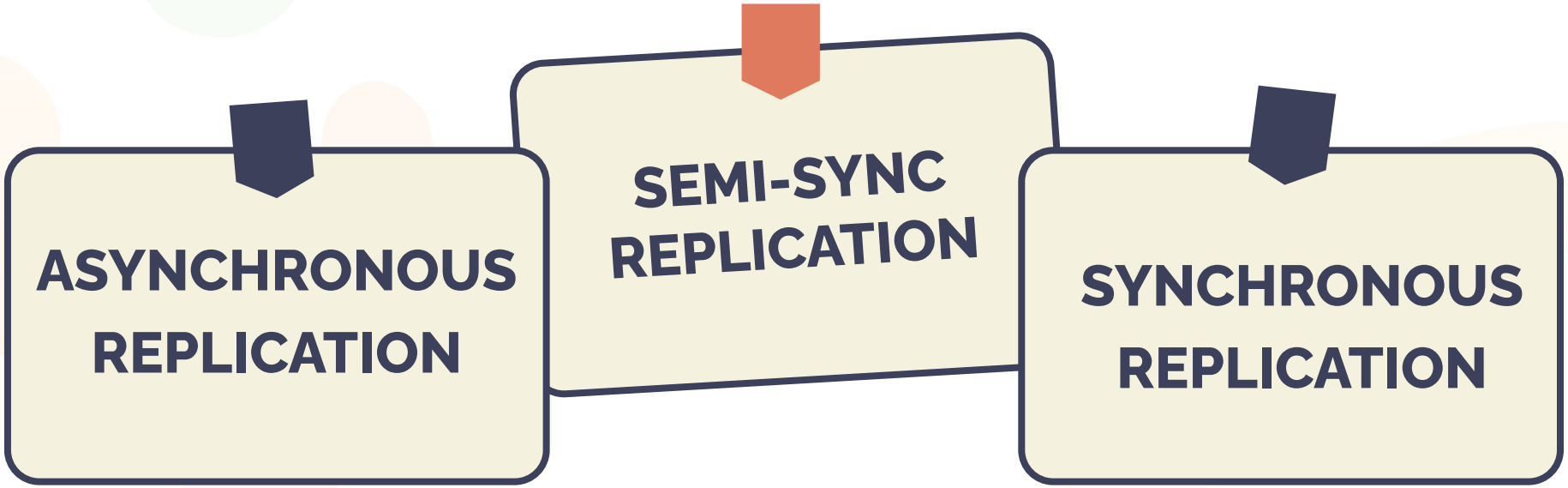
CREATE READ-REPLICA TO AVOID DATA MIGRATION

CQRS / READ-WRITE REPLICAS

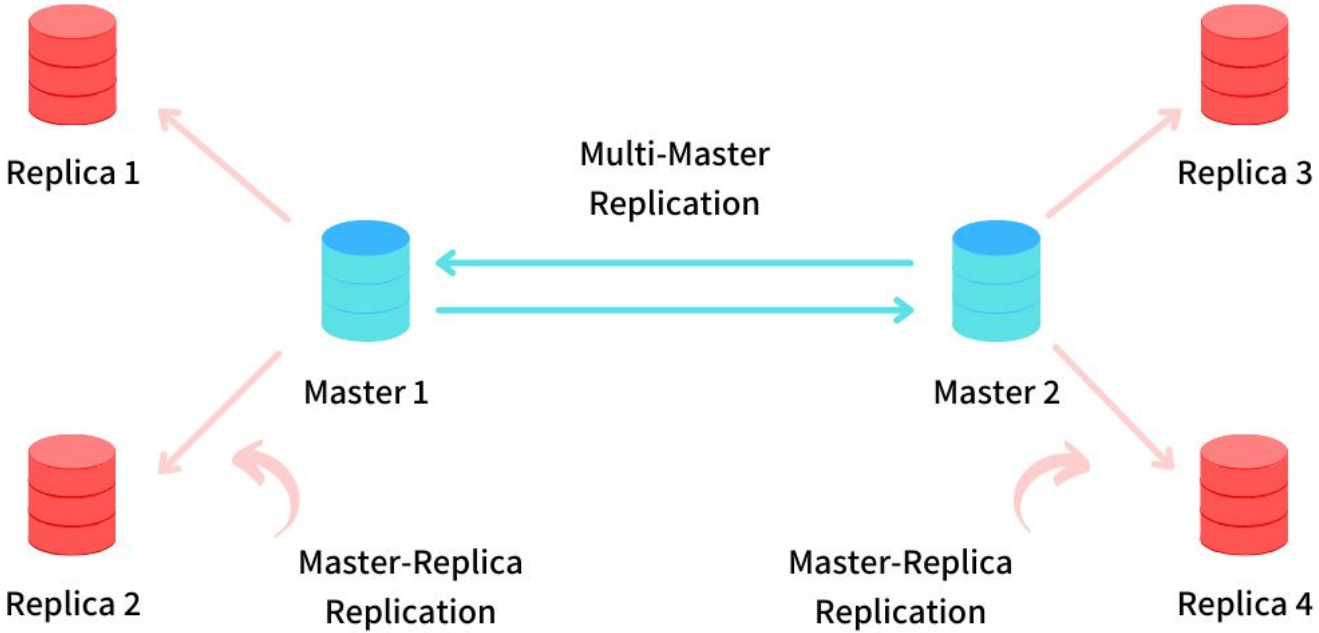
SEGREGATE READ AND WRITE FOR BETTER SCALING



TYPES OF REPLICATION



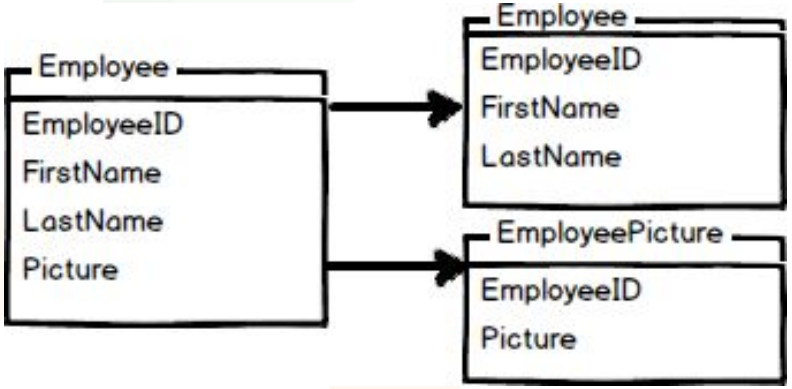
MULTI PRIMARY REPLICATION



CONCERNS WITH MULTI-MASTER

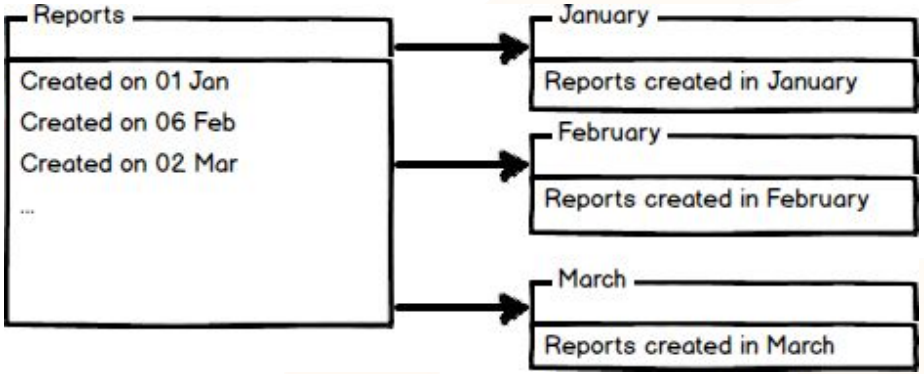
- Eventual Consistency
- Sluggish Performance
- Conflict Resolution

PARTITIONING & SHARDING



VERTICAL PARTITIONING

HORIZONTAL PARTITIONING



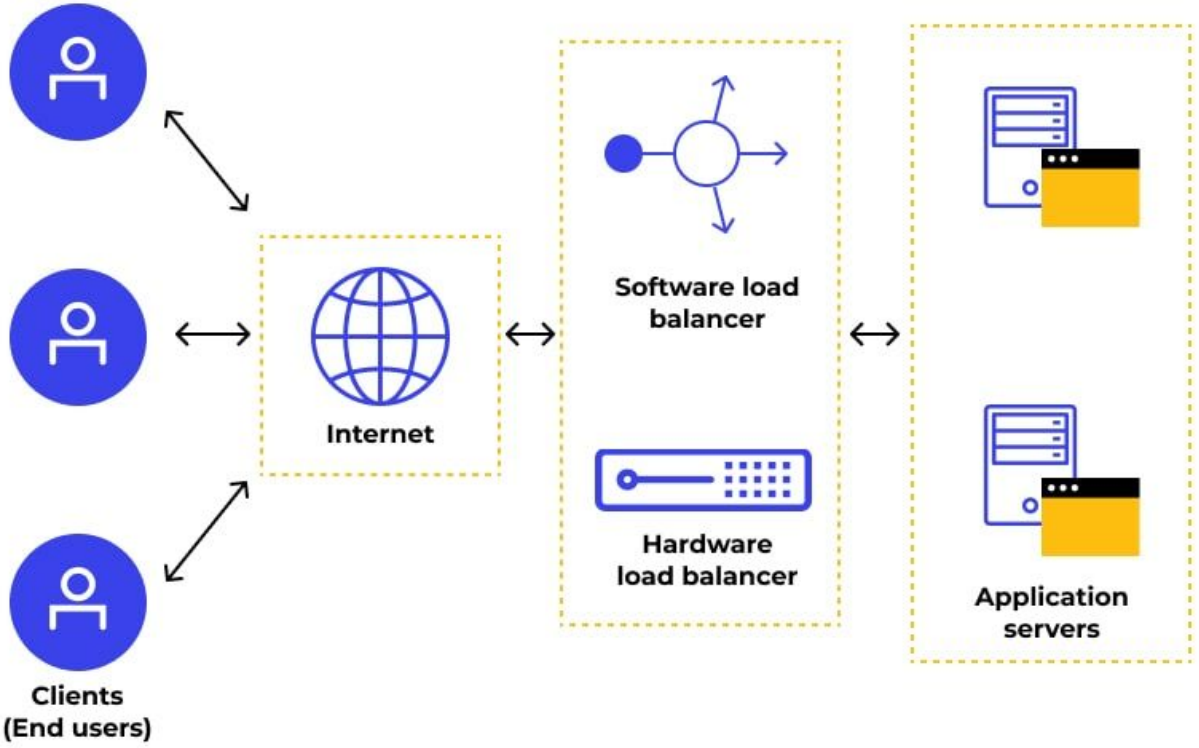
CACHING

- Faster Access
- Improve performance
- Less load on the Database
- Increased Resilience

TIPS AND TRICKS

- Always cache data that is read frequently modified less often
- Try and avoid caching large objects to speed up performance
- Choose the write caching strategy and TTL for each entry
- When to invalidate the cache is important to avoid stale data
- Make sure to test under load
- Make sure to secure sensitive information
- Use Redis and use frameworks like Spring Cache

02 HOW TO DISTRIBUTE LOAD?



LOAD BALANCER

LOAD BALANCING ALGORITHMS

STATIC LOAD BALANCING

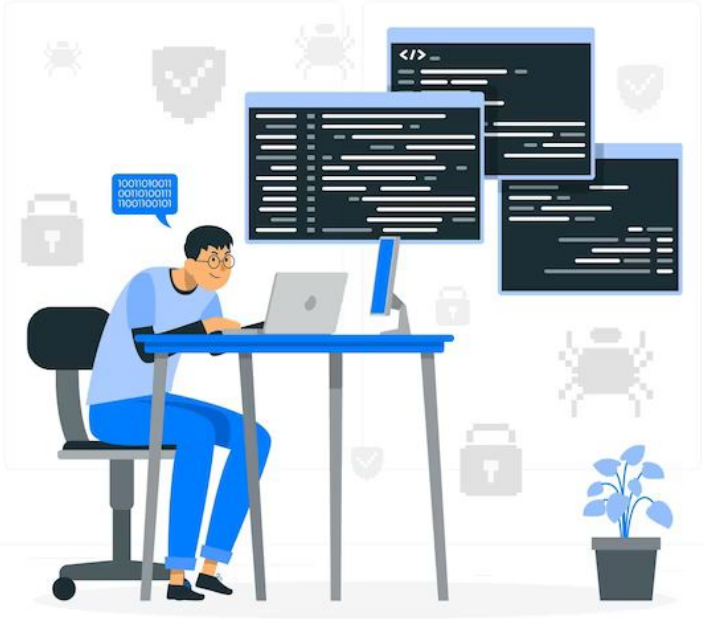
- ROUND - ROBIN
- WEIGHTED ROUND - ROBIN
- SOURCE IP HASH

DYNAMIC LOAD BALANCING

- LEAST CONNECTION
- WEIGHTED LEAST CONNECTION
- LEAST RESPONSE TIME
- RESOURCE BASED

03 APPLICATION SUPPORT FOR HORIZONTAL SCALING

- Statelessness
- Session Management
- Avoid using auto-incremented IDs
- Support Read and Write Replicas



04 HOW TO SCALE SERVERS?

MONITOR PERFORMANCE

SETUP ALERTS

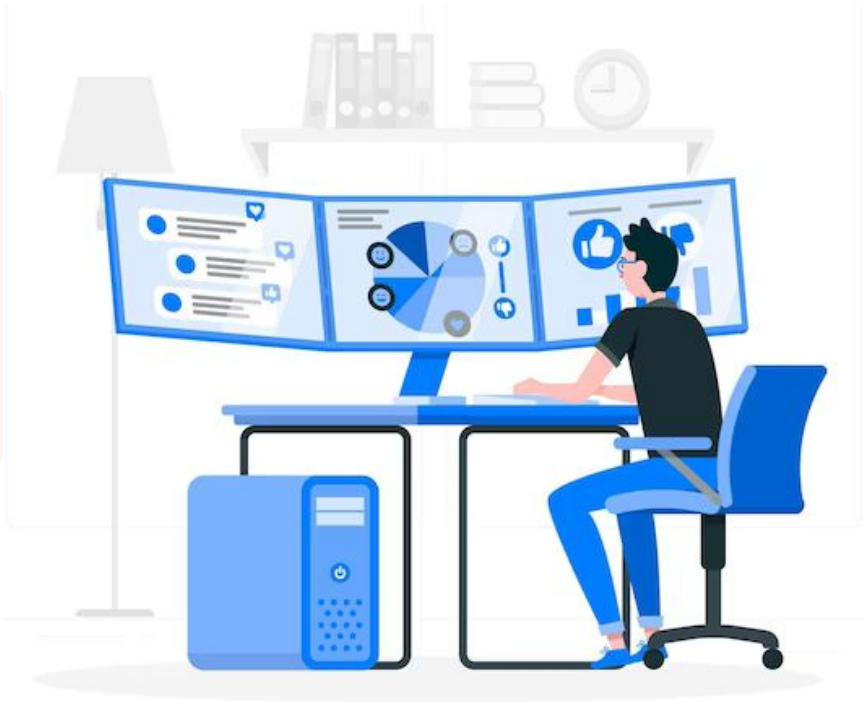
AND ADD SERVERS MANUALLY

splunk>

Prometheus

IdentityShield

DATADOG



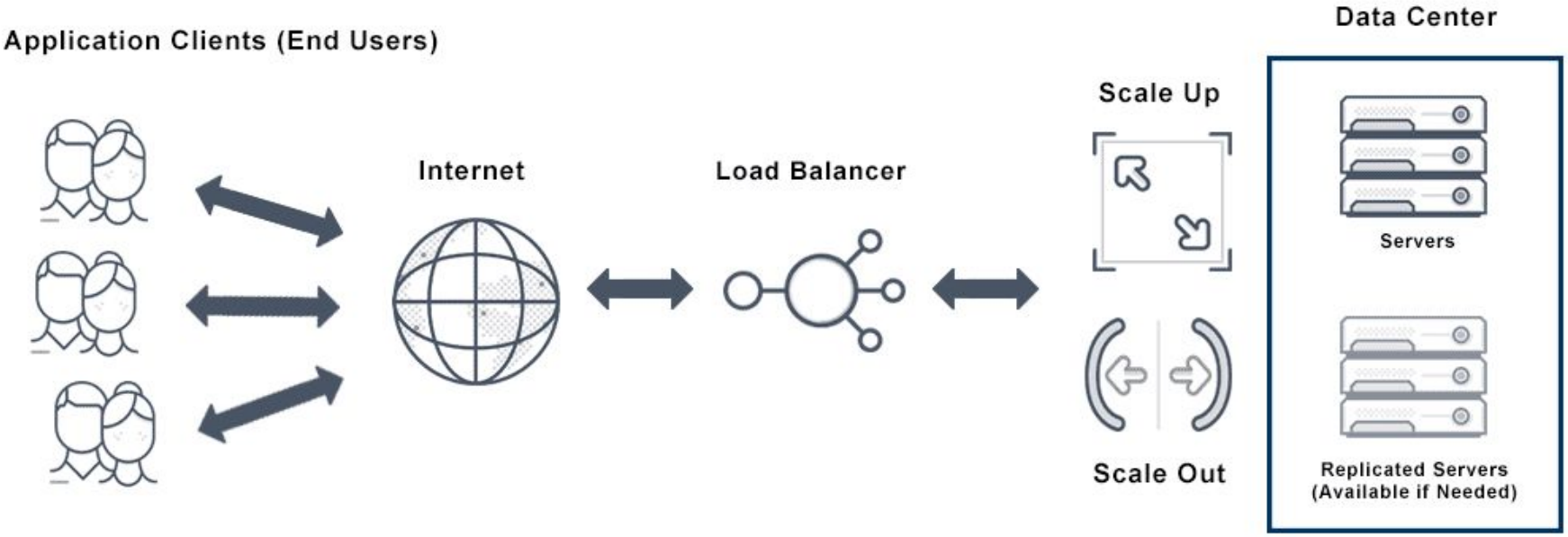
Grafana

New Relic

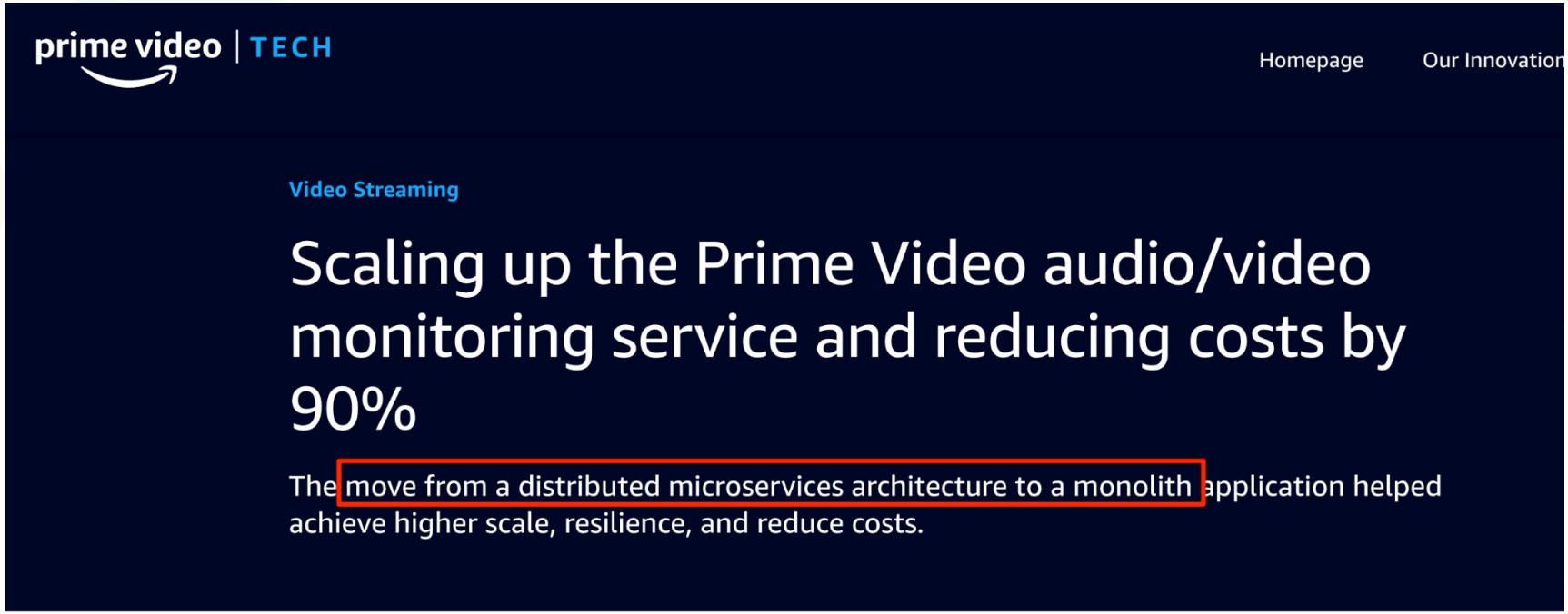
Nagios®

miniOrange

AUTOSCALING

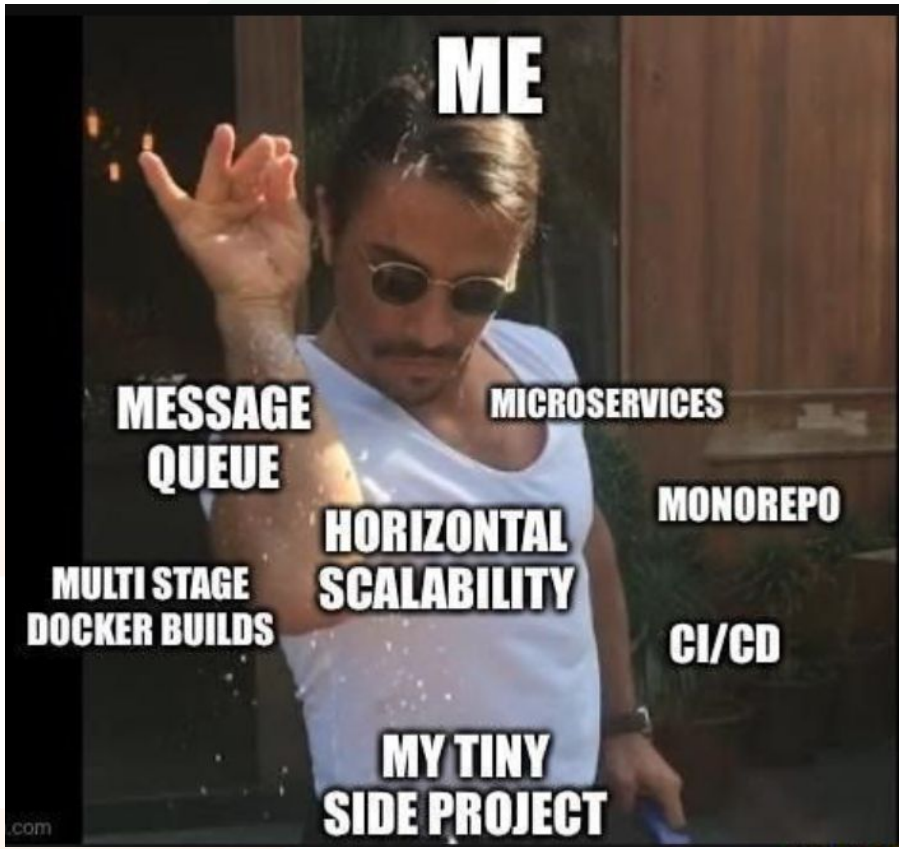


MICROSERVICES



https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

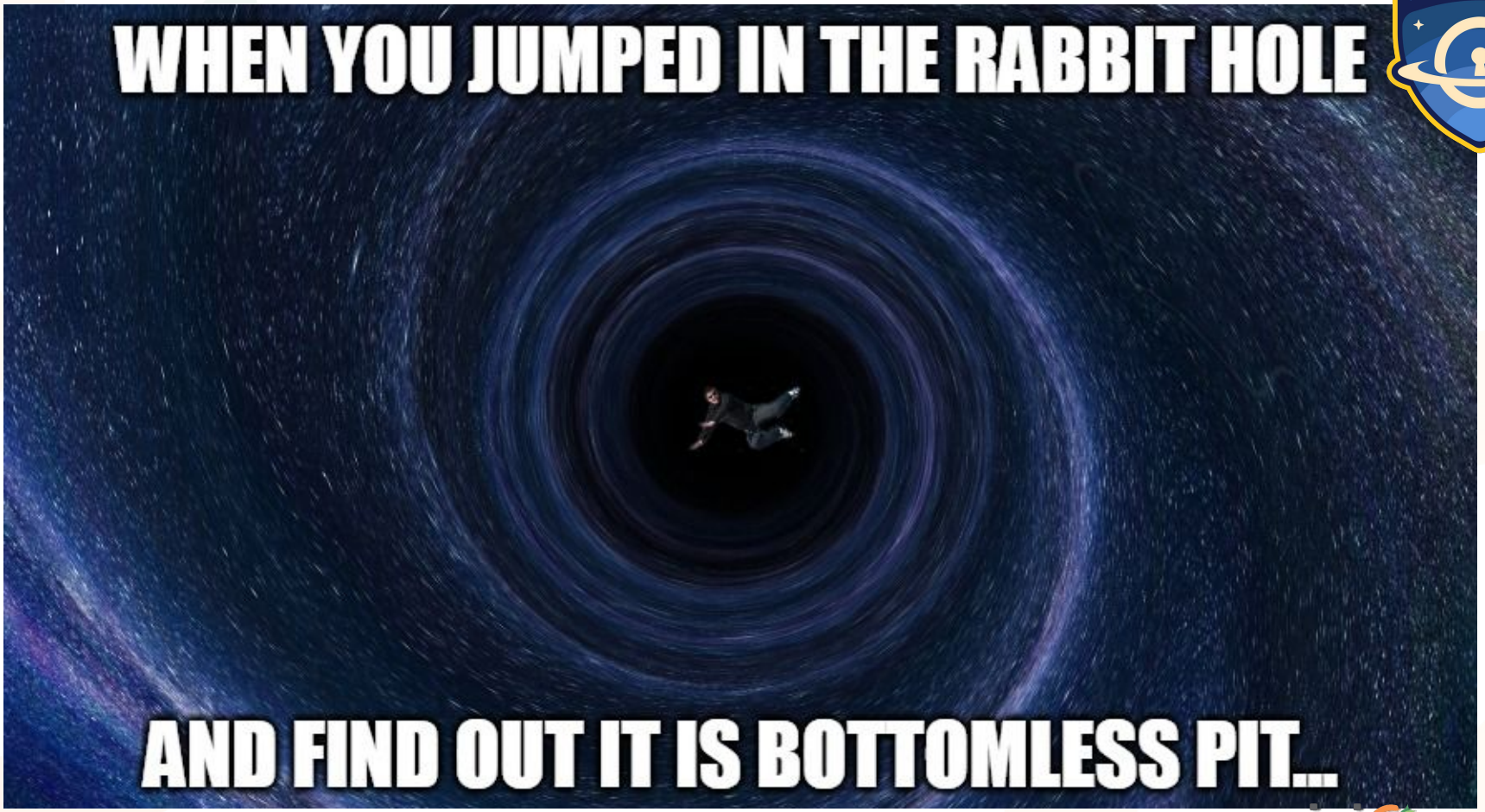
WHAT ABOUT MICROSERVICES?


ME
MESSAGE QUEUE
MICROSERVICES
MONOREPO
HORIZONTAL SCALABILITY
MULTI STAGE DOCKER BUILDS
CI/CD
MY TINY SIDE PROJECT


MONOLITH
MICROSERVICES
imgflip.com
ifunny.co

JOKES APART

- SCALABILITY
- FLEXIBILITY IN TECHNOLOGY
- RESILIENCE
- DEPLOYMENT SPEED
- TEAM AUTONOMY


WHEN YOU JUMPED IN THE RABBIT HOLE
AND FIND OUT IT IS BOTTOMLESS PIT...

SUMMARY

1. SCALABILITY IS MULTIFACETED
2. IT'S NOT ALWAYS HARDWARE
3. IMPORTANCE OF THE RIGHT TOOLS
4. ADDING NEW STUFF ADDS COMPLEXITY
5. CONTINUOUS EVOLUTION

FOR ALL WHO LIKE SHINY NEW STUFF

NANOSERVICES

DO YOU HAVE ANY QUESTIONS?

pratish@xecurify.com | www.miniorange.com

https://github.com/**frittlechasm**

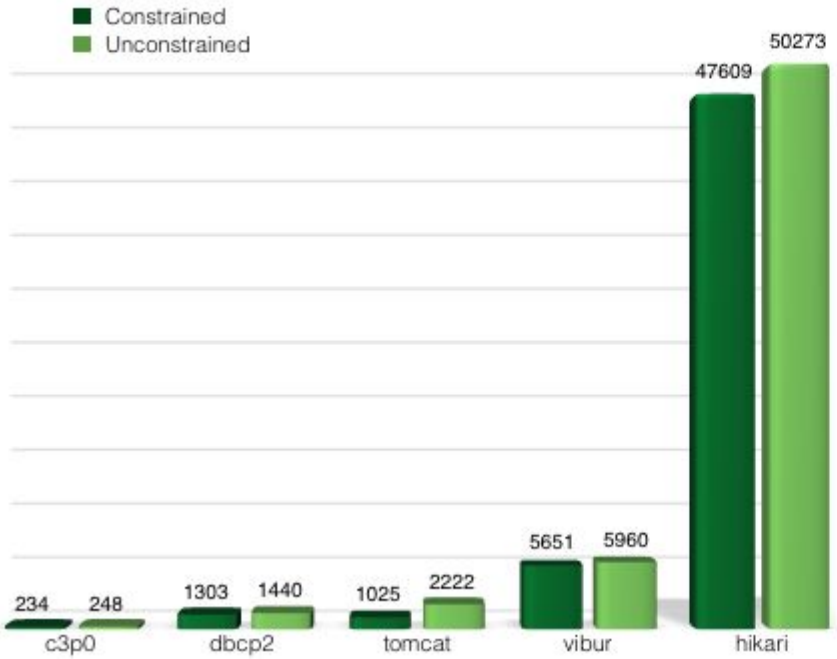
https://www.linkedin.com/in/**pratish-ray/**

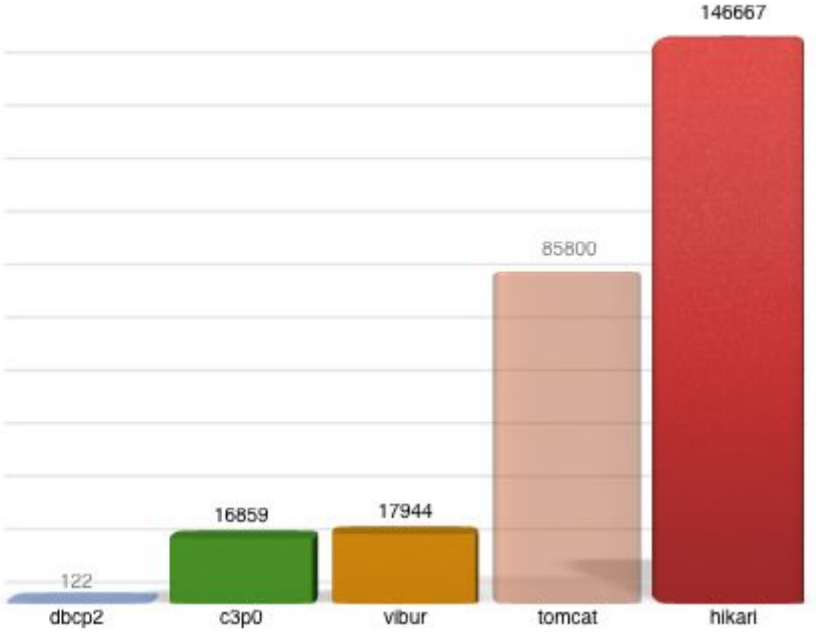
THANK YOU FOR BEING A LOVELY AUDIENCE

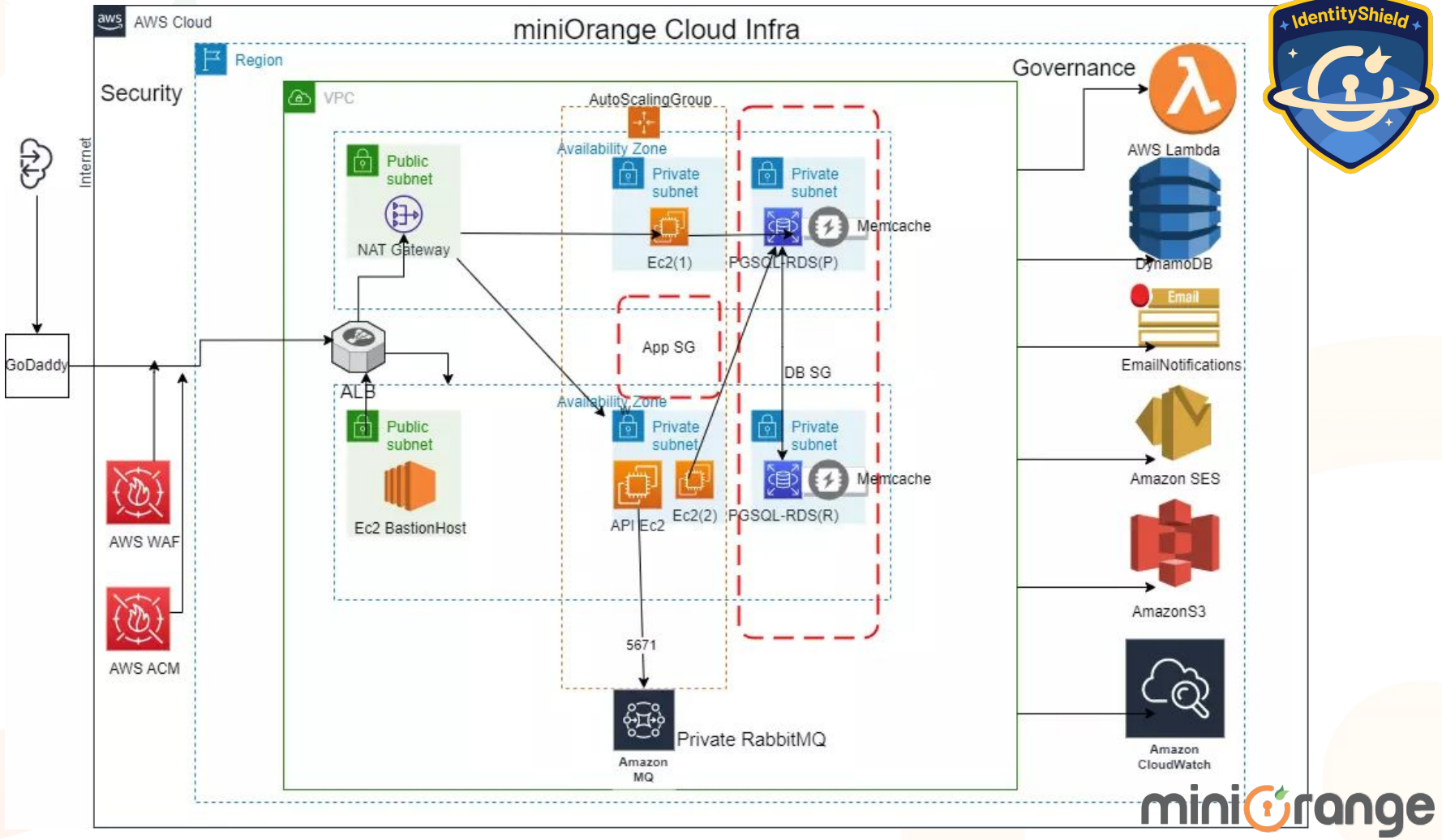
miniOrange

FEEDBACK

miniOrange


Connection Cycle ops/ms
Constrained
Unconstrained
234
248
1303
1440
1025
2222
5651
5960
47609
50273
c3p0
dbcp2
tomcat
vibur
hikari


Statement Cycle ops/ms
122
16859
17944
85800
146667
dbcp2
c3p0
vibur
tomcat
hikari


AWS Cloud
miniOrange Cloud Infra
Region
Security
VPC
AutoScalingGroup
Availability Zone
Public subnet
NAT Gateway
Private subnet
Ec2(1)
Private subnet
PGSQL-RDS(P)
Memcache
Internet
GoDaddy
ALB
App SG
DB SG
Availability Zone
Public subnet
Ec2 BastionHost
Private subnet
API Ec2
Ec2(2)
Private subnet
PGSQL-RDS(R)
Memcache
AWS WAF
AWS ACM
5671
Amazon MQ
Private RabbitMQ
Governance
AWS Lambda
DynamoDB
Email
EmailNotifications
Amazon SES
AmazonS3
Amazon CloudWatch