AWS re:Invent

ARC205

Scaling Up to Your First 10 Million Users

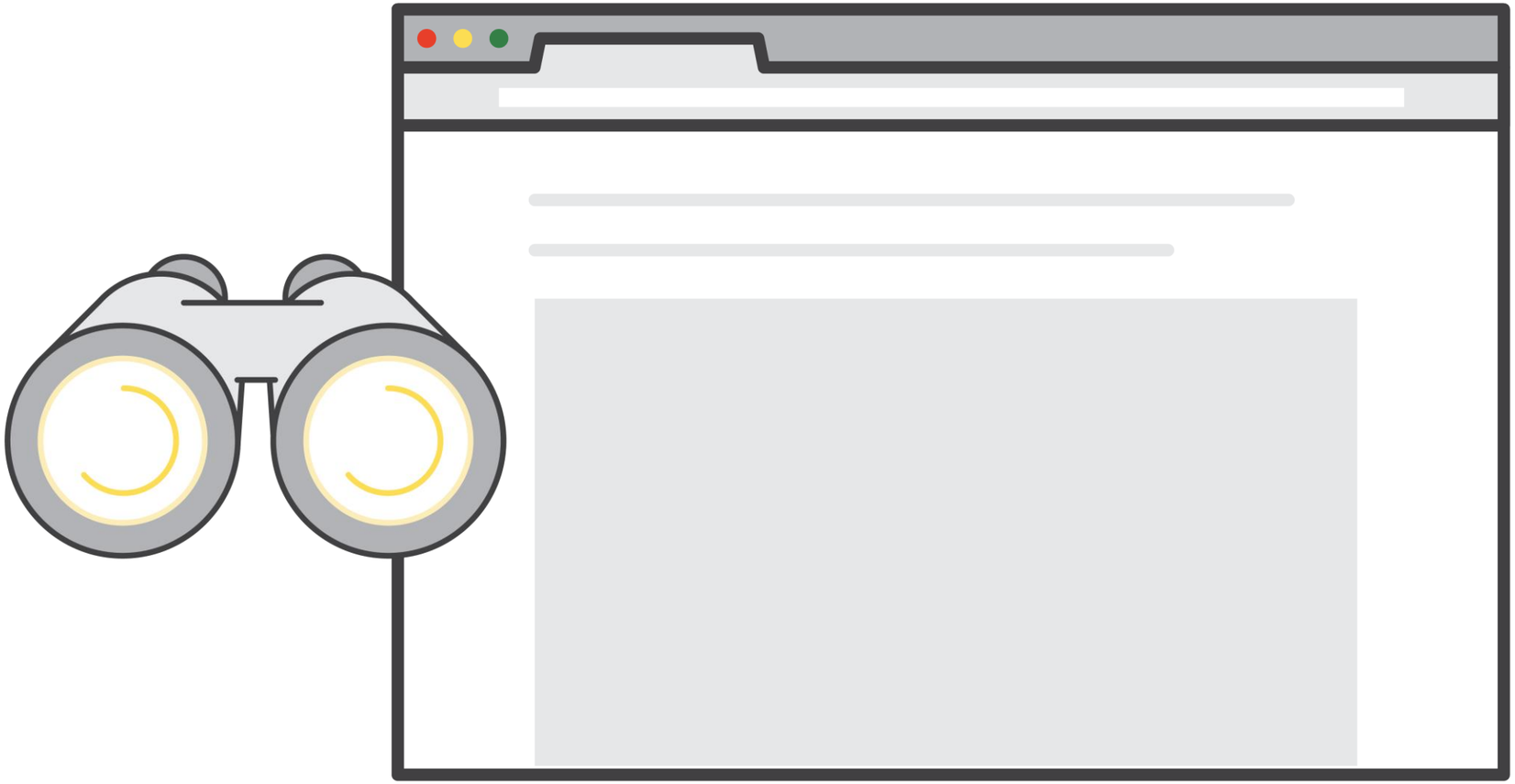
Ben Thurgood
Principal Solutions Architect
Amazon Web Services

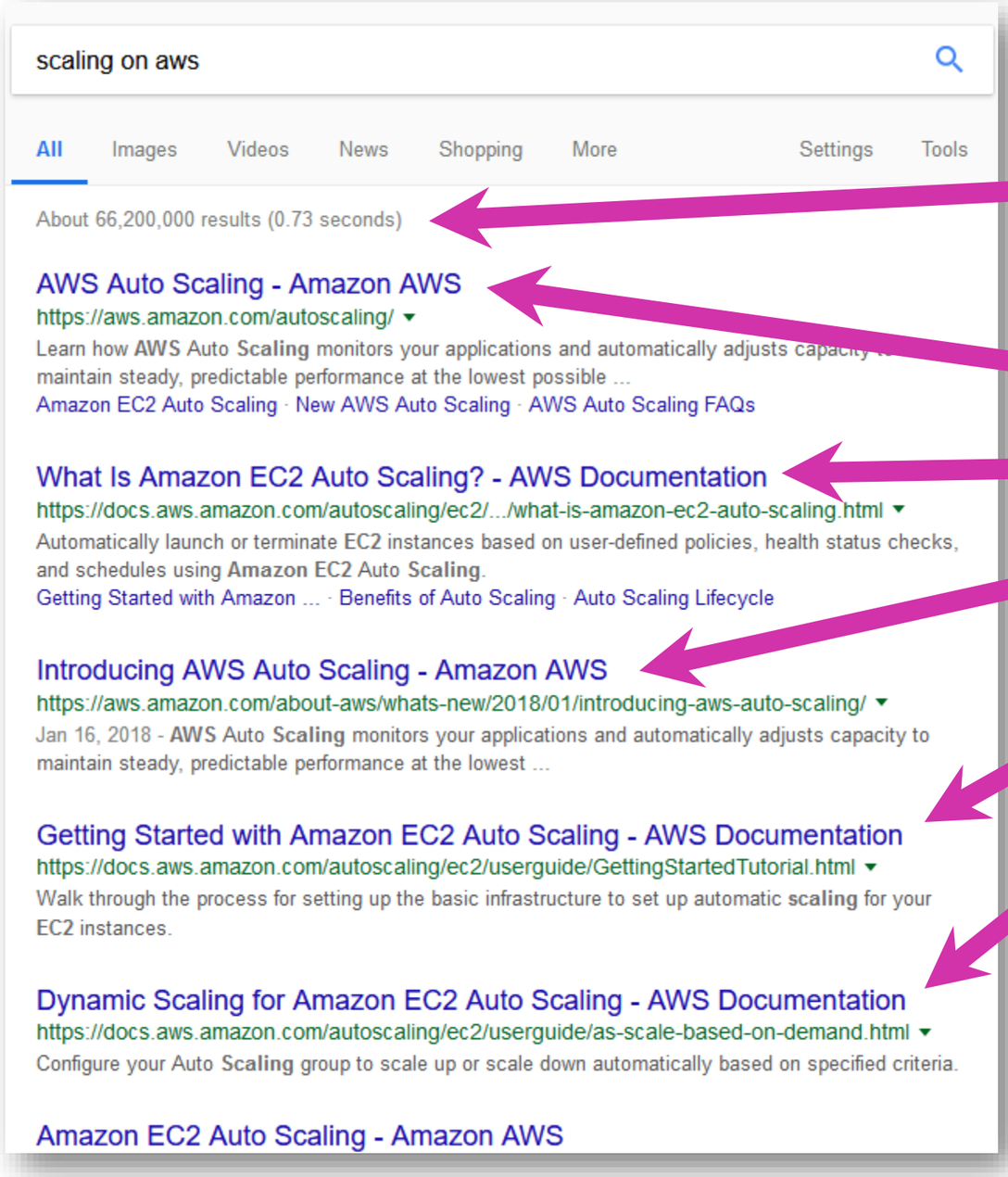
AWS re:Invent

aws


RIVERBANK STAND
CHAPPELL STAND
50

http://i.telegraph.co.uk/multimedia/archive/02674/CLIMBER_2674482b.jpg





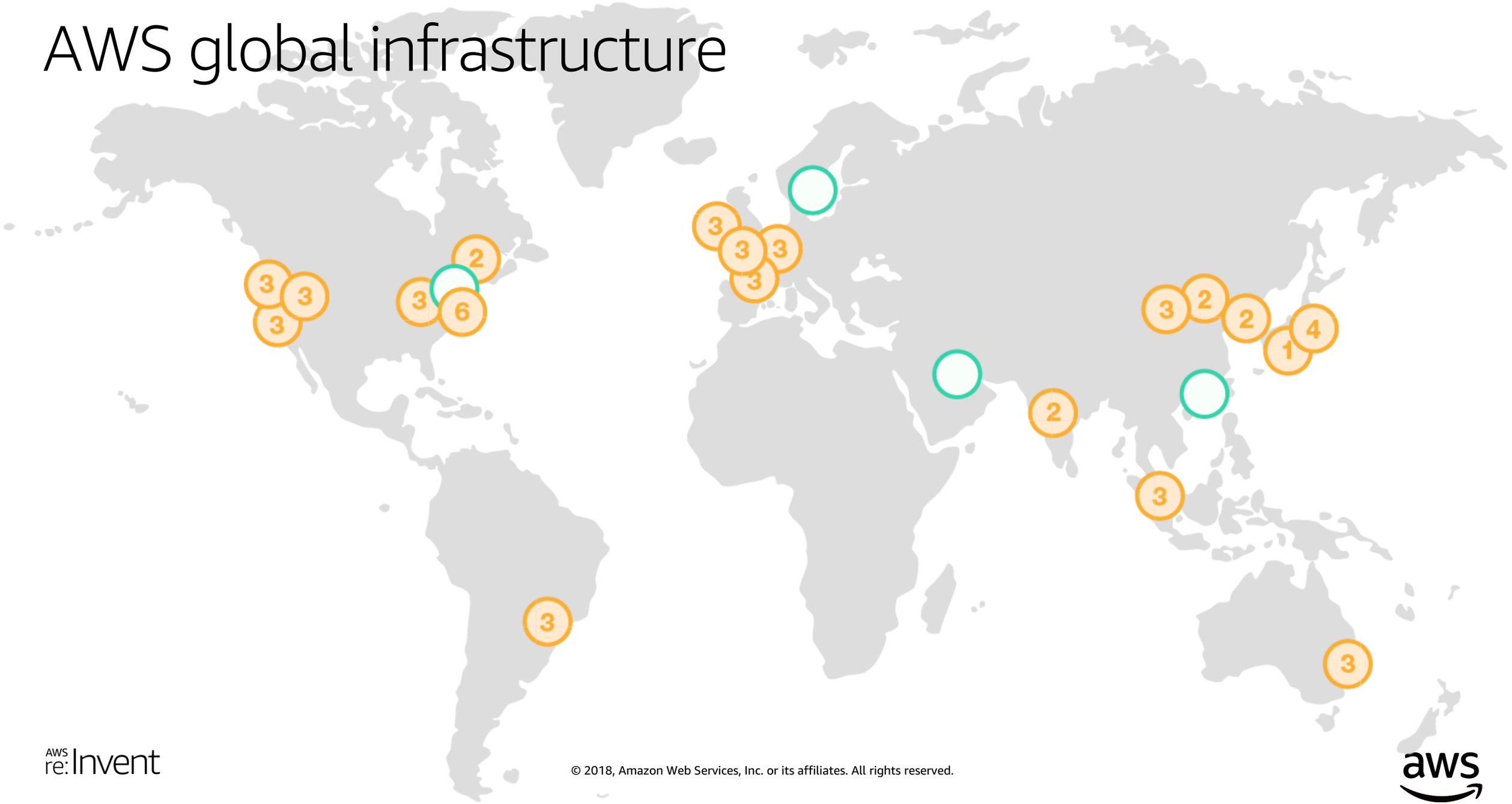
Now that's a lot of things to read!

This is NOT where we want to start!

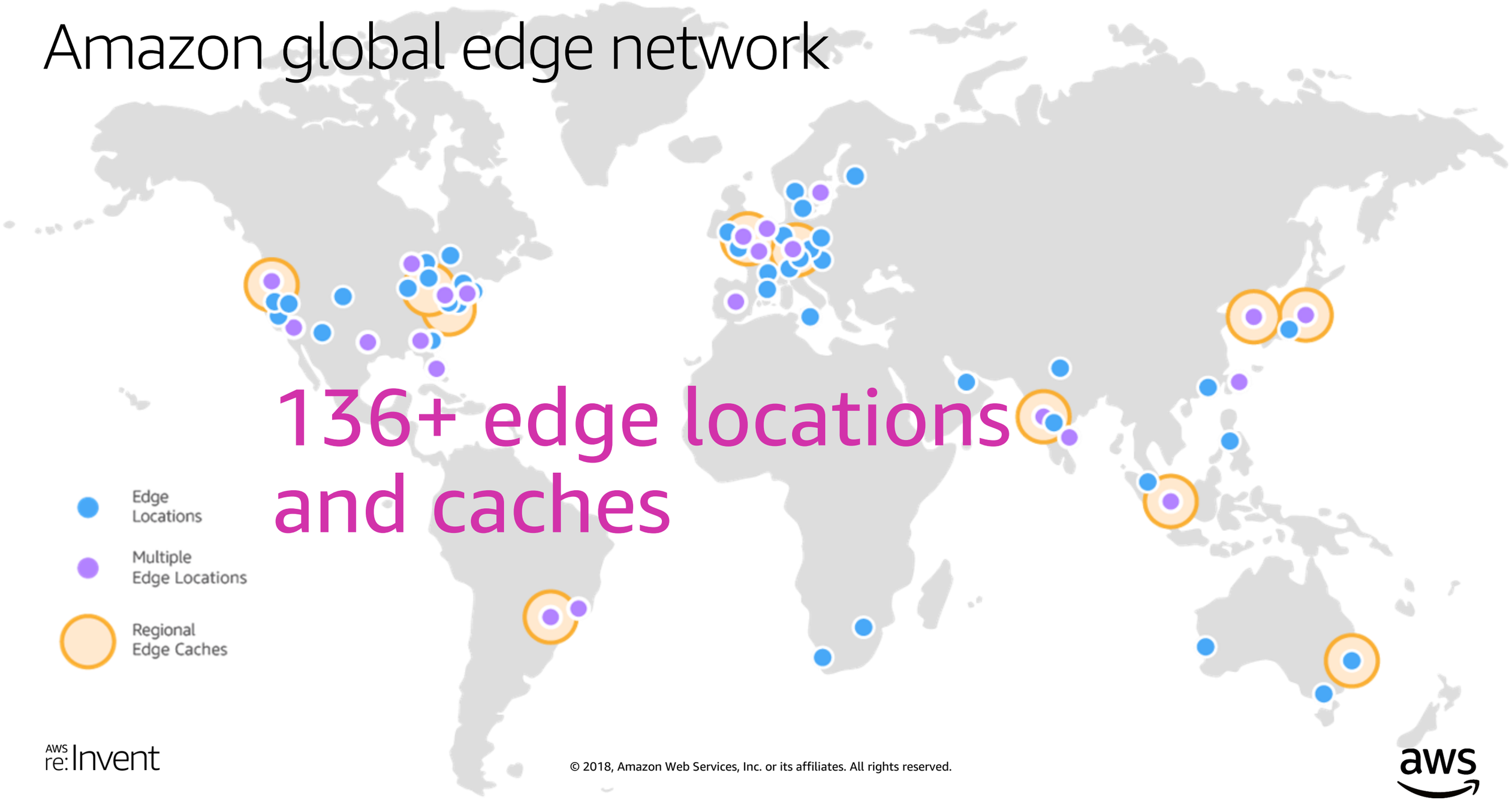
It’s not the single thing that fixes everything

What do we need first?

AWS global infrastructure

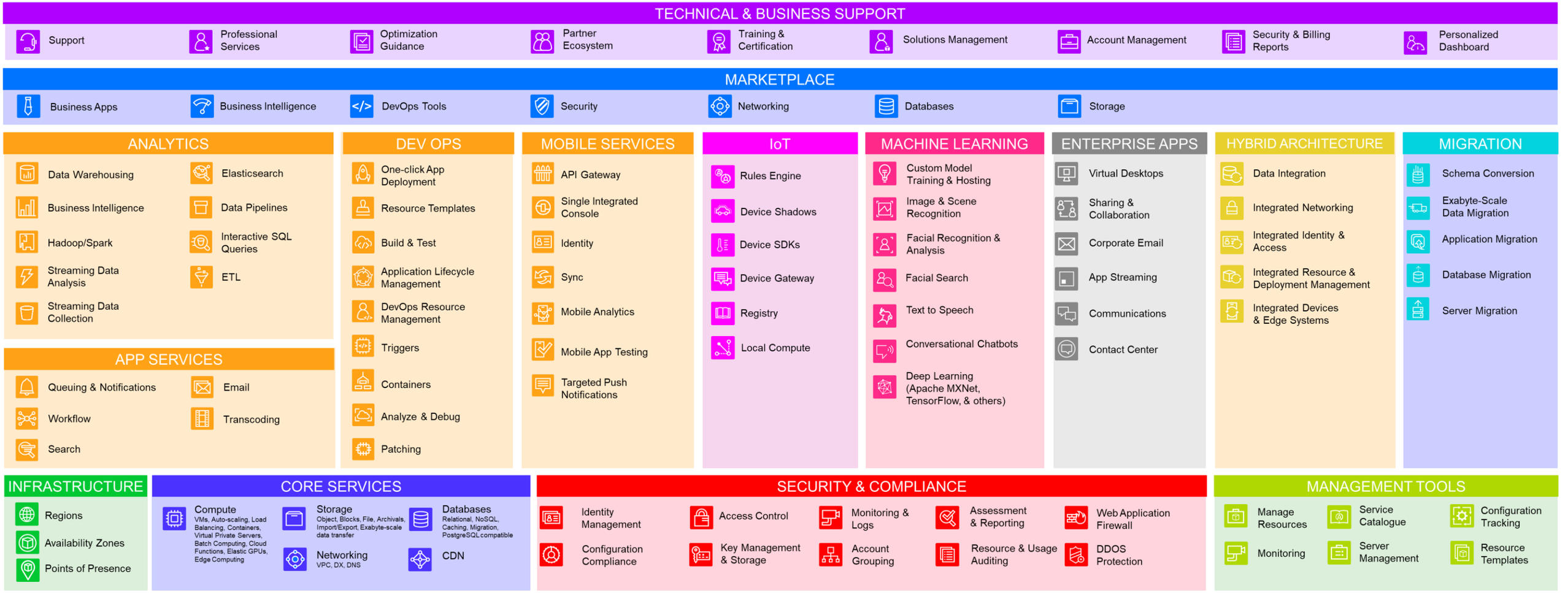


Amazon global edge network



136+ edge locations and caches

Robust, fully featured technology infrastructure



AWS building blocks

Inherently highly scalable, available, and fault-tolerant services

Amazon CloudFront
Amazon Route 53
Amazon Simple Storage Service (Amazon S3)
Amazon DynamoDB
Elastic Load Balancing
Amazon Elastic File System (Amazon EFS)
AWS Lambda
Amazon Simple Queue Service (Amazon SQS)
Amazon Simple Notification Service (Amazon SNS)
Amazon Simple Email Service (Amazon SES)
AWS Step Functions
. . .

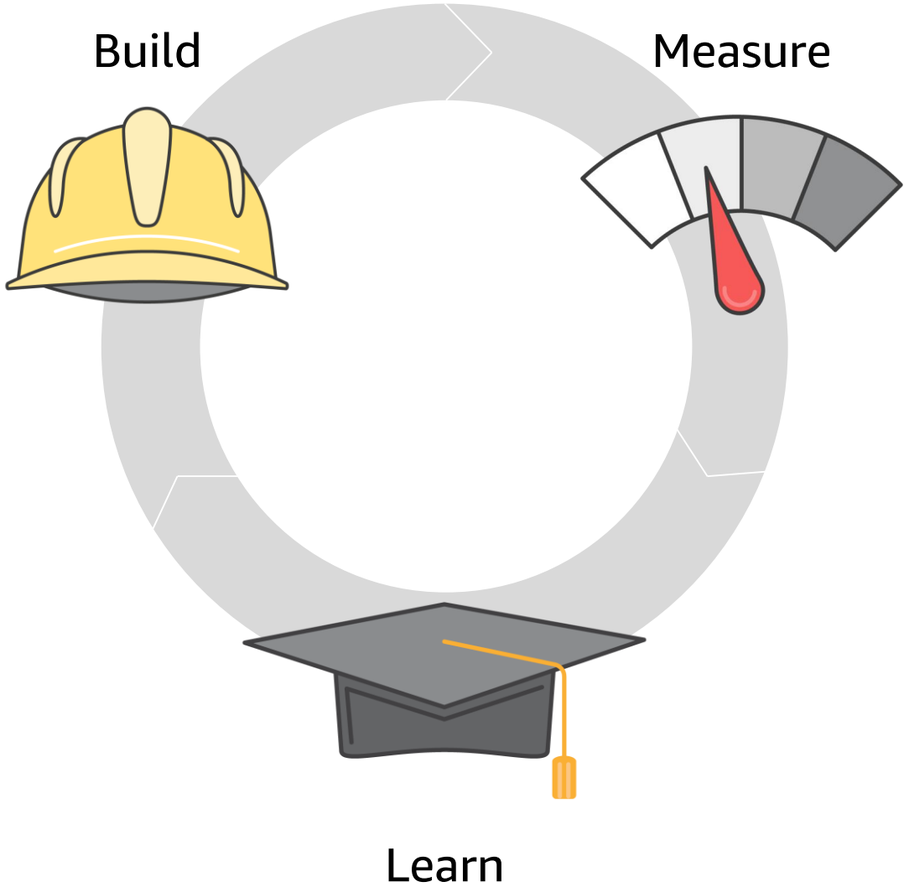
Highly scalable, available with the right architecture

Amazon Elastic Compute Cloud (Amazon EC2)
Amazon Elastic Block Store (Amazon EBS)
Amazon Relational Database Service (Amazon RDS)
Amazon Virtual Private Cloud (Amazon VPC)

Considerations

“Many decisions are reversible, two-way doors.”

Jeff Bezos

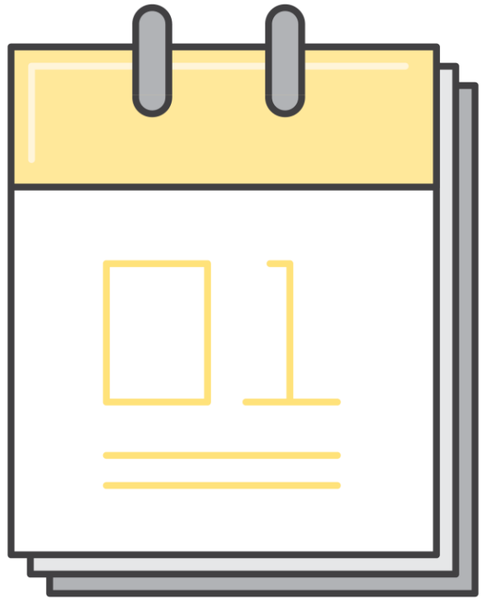


Guiding tenets

Identify and avoid undifferentiated heavy lifting (Wardley maps)

Serverless versus managed versus run it yourself

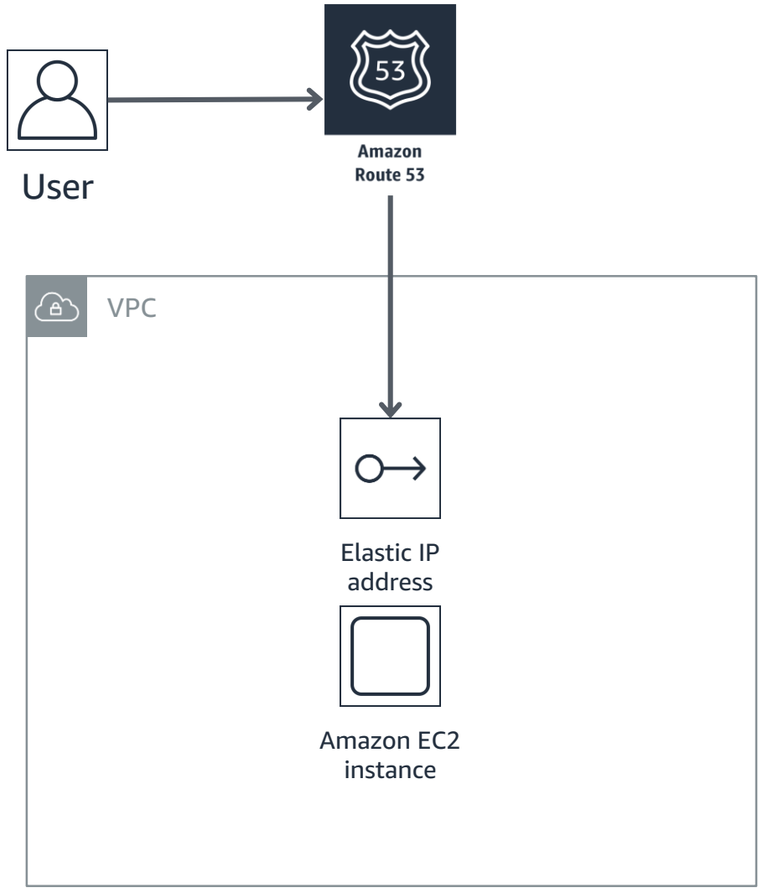
So let's start from day . . .



User

You

One user



Amazon Lightsail: The easiest way to get started on AWS

Choose from five plans that include bundled compute, storage, and networking

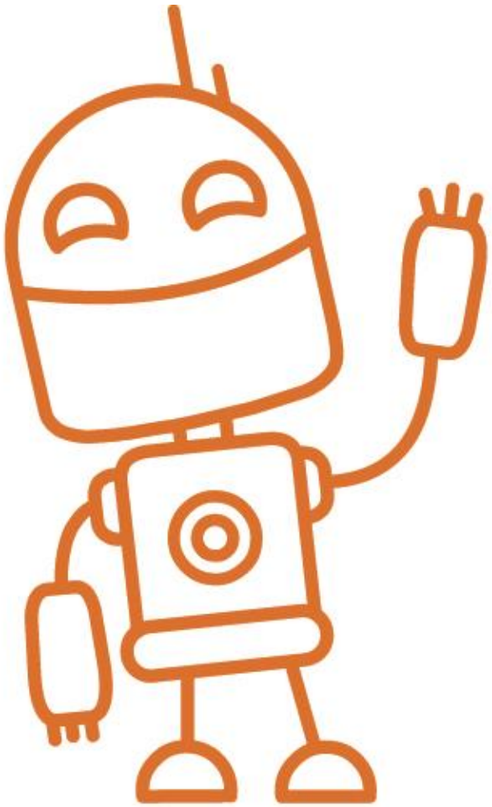
Benefit from a low, predictable price

Spin up a fully configured server in seconds

Manage from the intuitive Lightsail console

Scale with access to AWS services

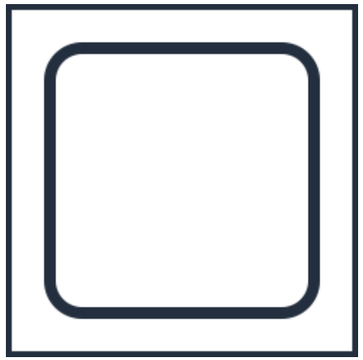
Automate with Lightsail API & AWS Command Line Interface (AWS CLI)



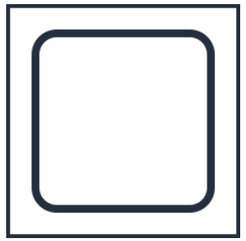
AmazonLightsail

“We’re gonna need a bigger box”

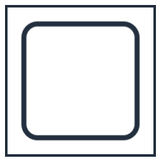
- Simplest approach
- Can now leverage PIOPS
- High I/O instances
- High memory instances
- High CPU instances
- High storage instances
- Easy to change instance sizes
- Will hit an endpoint eventually



c5.9xlarge



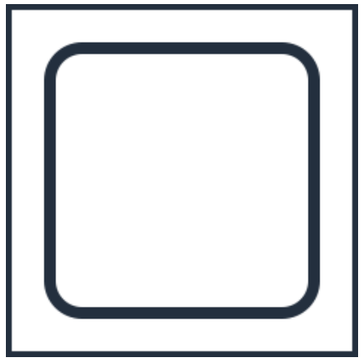
m5.2xlarge



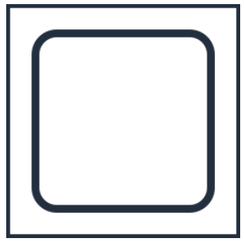
t3.nano

“We’re gonna need a bigger box”

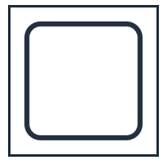
- Simplest approach
- Can now leverage PIOPS
- High I/O instances
- High memory instances
- High CPU instances
- High storage instances
- Easy to change instance sizes
- Will hit an endpoint eventually



c5.9xlarge



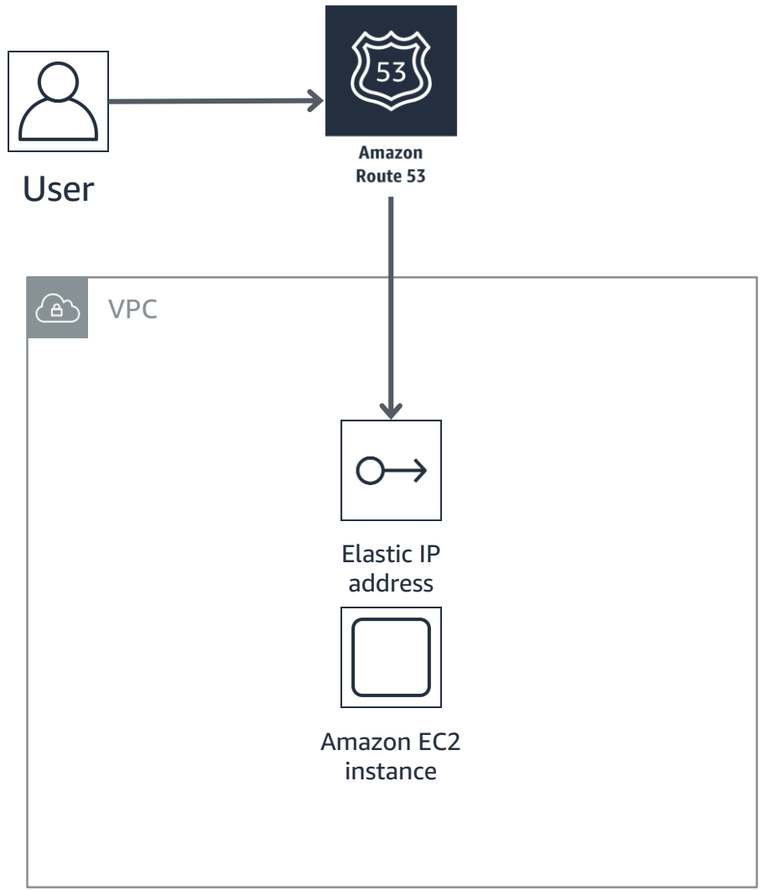
m5.2xlarge



t3.nano

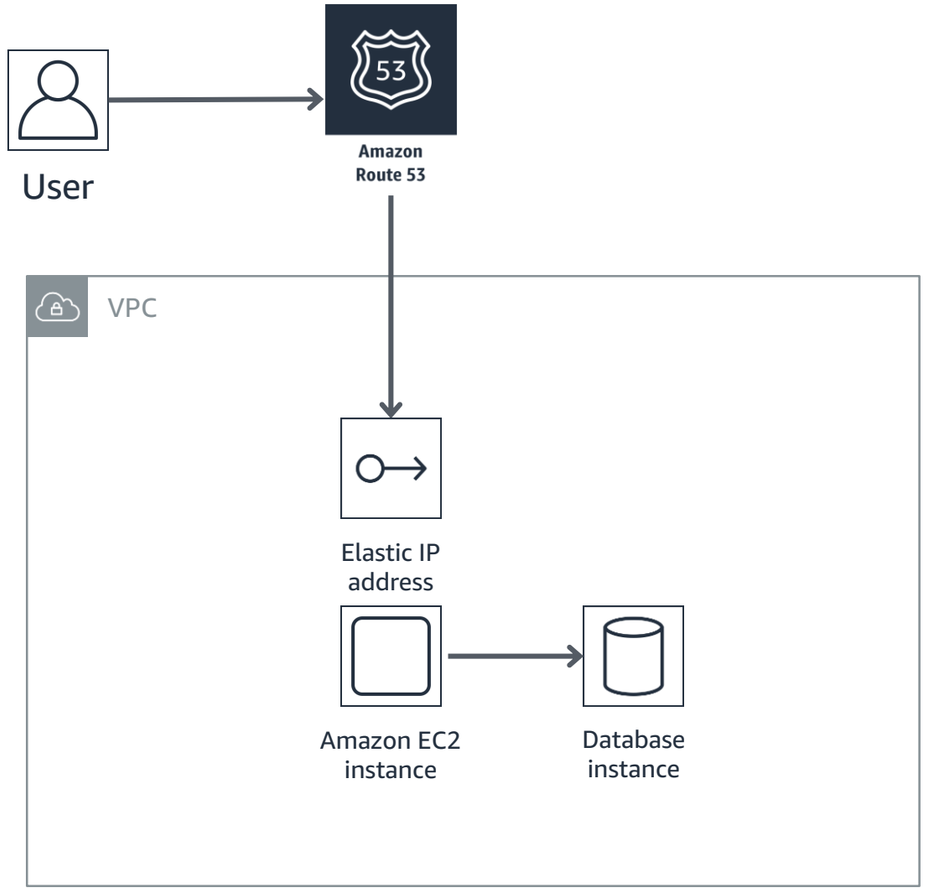
One user

- No failover
- No redundancy
- Too many eggs in one basket



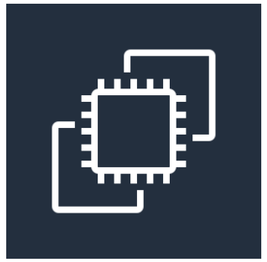
Users > 1

Users > 1



Database options

Self-managed



Amazon EC2

Fully managed

Amazon RDS

Amazon DynamoDB

Amazon Neptune

Amazon Aurora

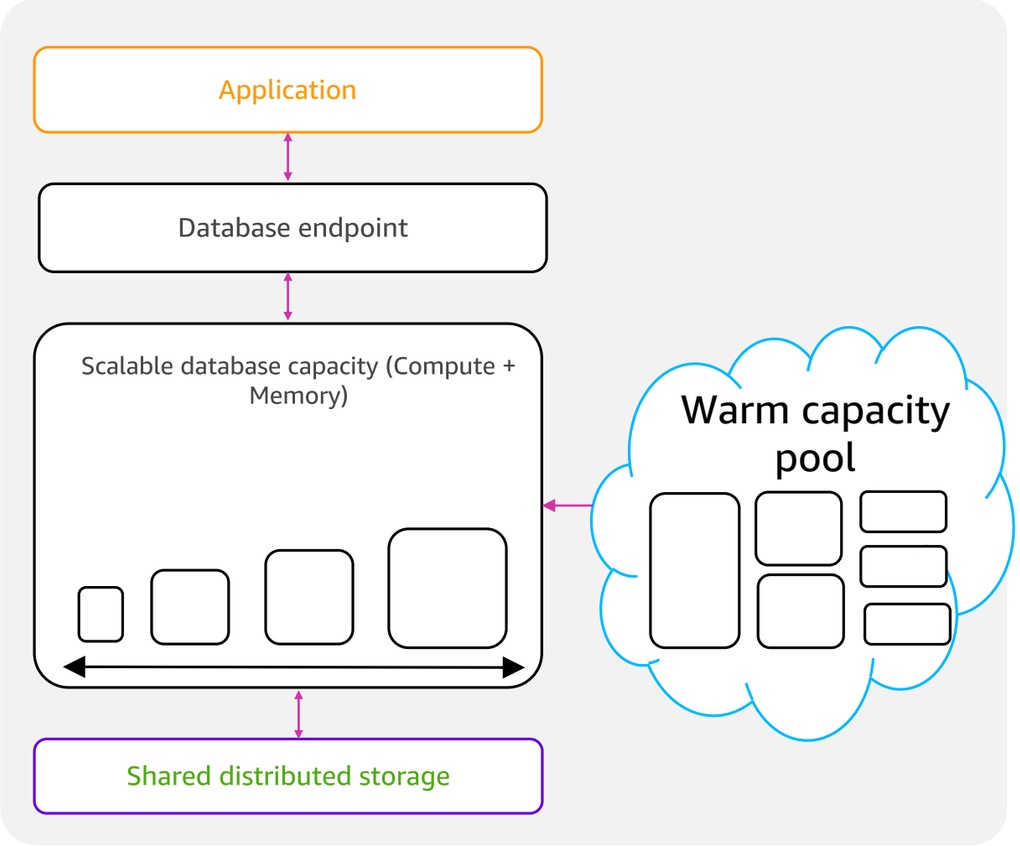
- MySQL or Postgres compatible
- Automatic storage scaling (up to 64 TB)
- Up to 15 read-replicas
- Continuous (incremental) backups to Amazon S3
- Six-way replication across three zones

Amazon Aurora

- MySQL or Postgres compatible
- Automatic storage scaling (up to 64 TB)
- Up to 15 read-replicas
- Continuous (incremental) backups to Amazon S3
- Six-way replication across three zones
- Serverless option

Aurora Serverless

On-demand, auto-scaling database for applications with variable workloads



Starts up on demand, shuts down when not in use

Automatically scales with no instances to manage

Pay per second for the database capacity you use

To NoSQL, or not to NoSQL?

Start with SQL databases

Why start with SQL?

- Established and well-known technology
- Lots of existing code, communities, books, and tools
- You aren’t going to break SQL DBs in your first millions of users. No, really, you won’t*
- Clear patterns to scalability

*Unless you are doing something SUPER peculiar with the data or you have MASSIVE amounts of it.
. . . but even then SQL will have a place in your stack

AH HA! You said,
“Massive amounts”

> 5 TB in year one?

Incredibly data intensive workload?

OK!

You *might* need NoSQL

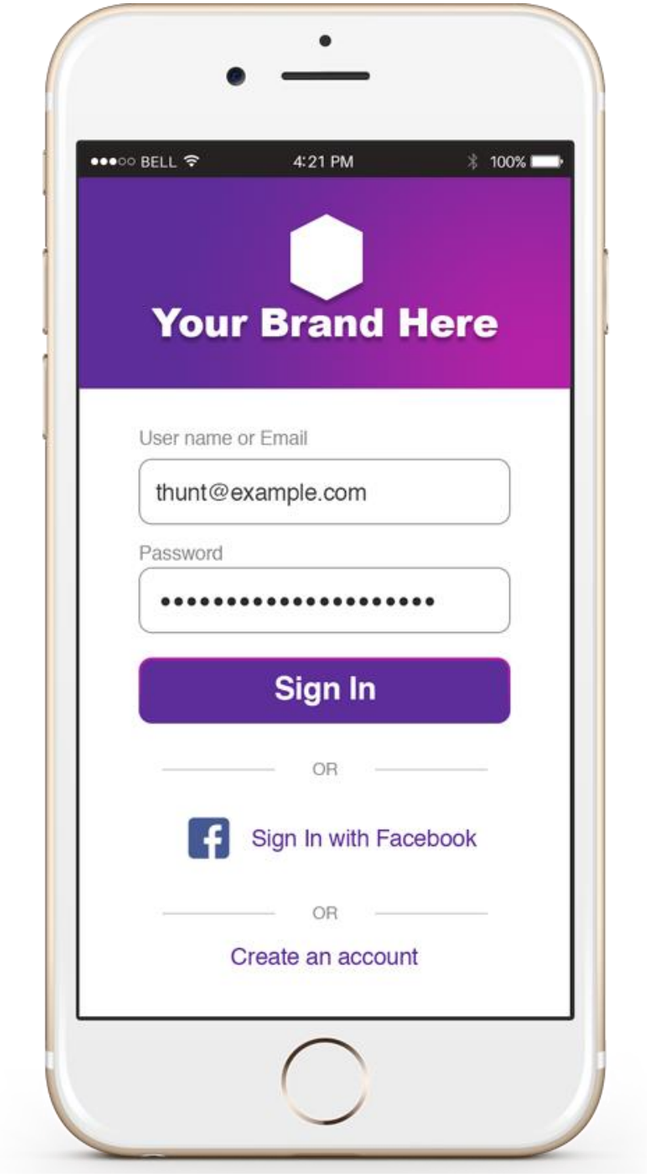
Why else might you need NoSQL?

- Super low-latency applications
- Metadata-driven datasets
- Highly nonrelational data
- Need schema-less data constructs*
- Rapid ingest of data (thousands of records/sec)
- Massive amounts of data (again, in the TB range)

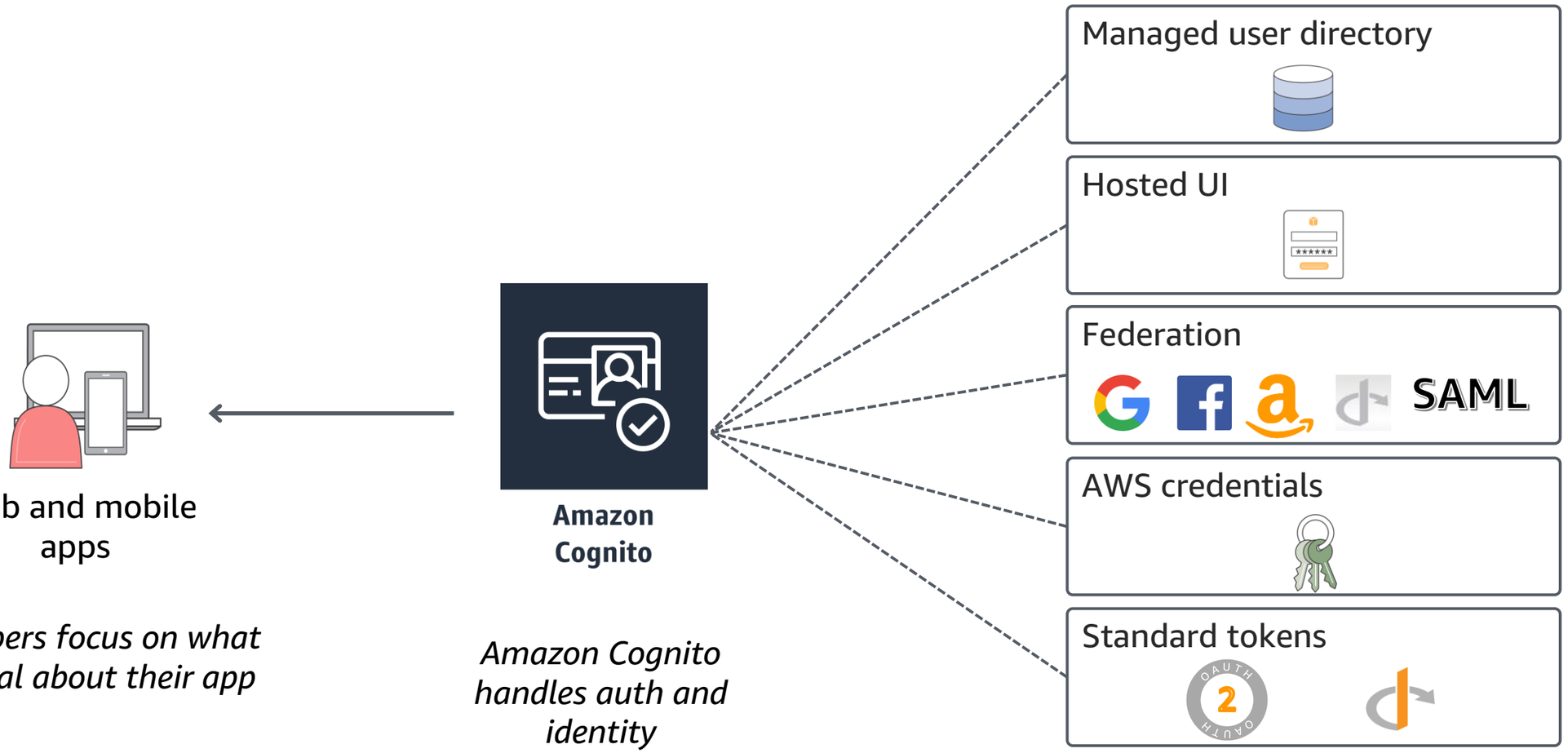
- *Need!= “It’s easier to do dev without schemas”

Users > 1

Registration, Sign In, and others



Amazon Cognito overview



Security checklist

AWS Identity and Access Management

Security checklist

AWS
CloudTrail

AWS Identity
and Access
Management

Amazon
GuardDuty

AWS Shield

Security checklist

AWS CloudTrail

AWS Identity and Access Management

Amazon GuardDuty

AWS Shield

AWS Secrets Manager

AWS WAF

AWS Certificate Manager

AWS Key Management Service

Security checklist

AWS CloudTrail

AWS Identity and Access Management

Amazon GuardDuty

AWS Shield

AWS Secrets Manager

AWS WAF

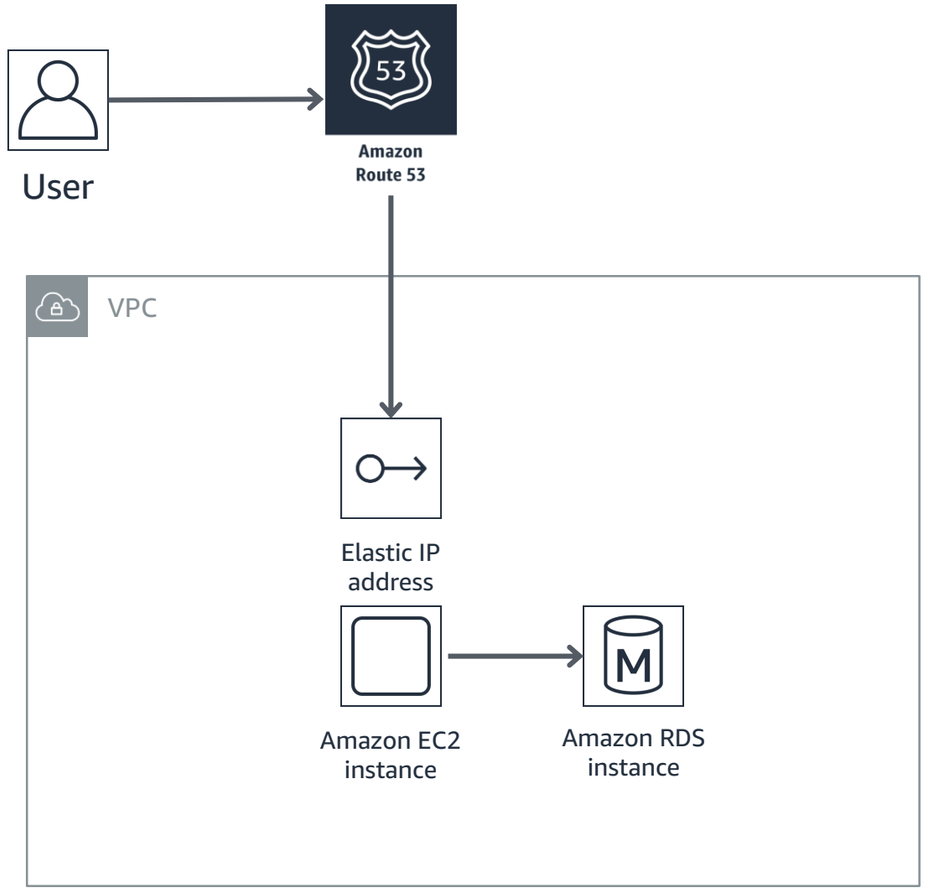
AWS Certificate Manager

AWS Key Management Service

AWS Config

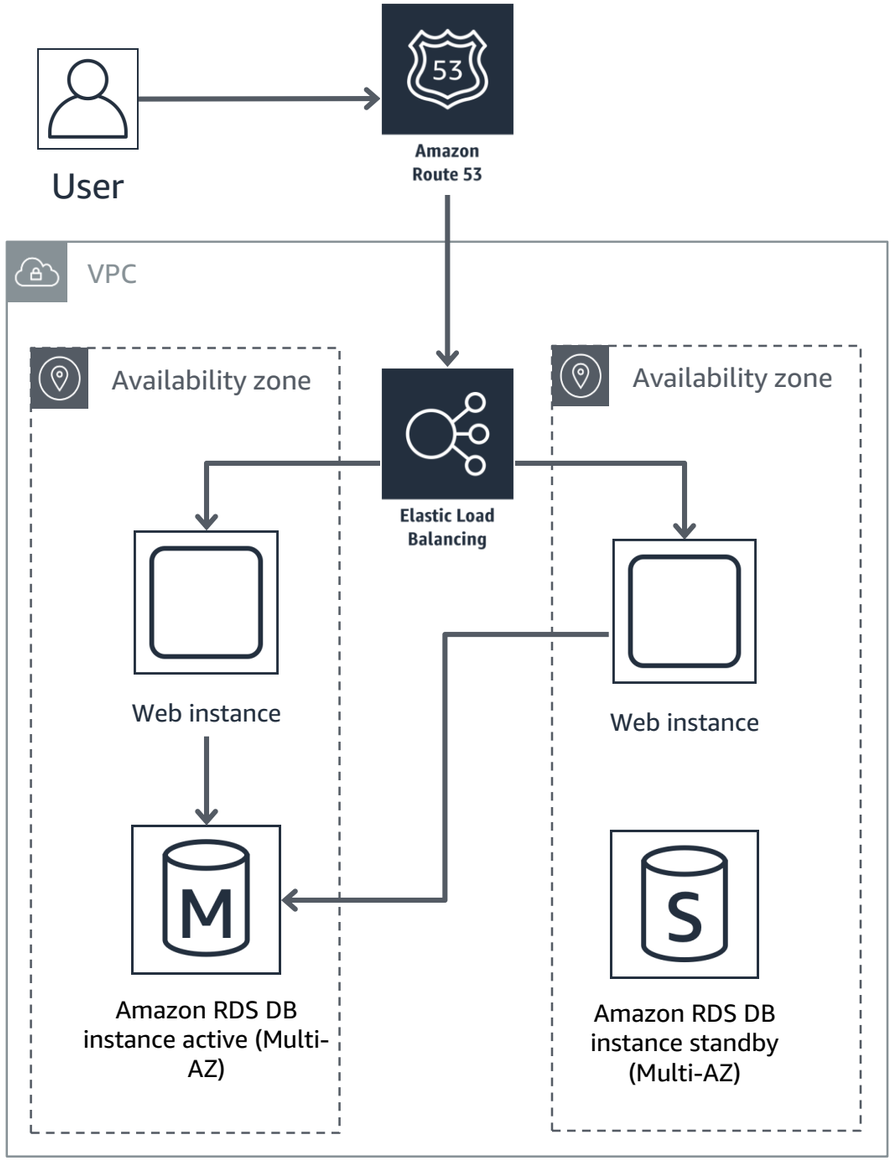
Users > 100

Users > 100



Users > 1000

Users > 1000



Sharing the load

Application Load Balancer

Network Load Balancer

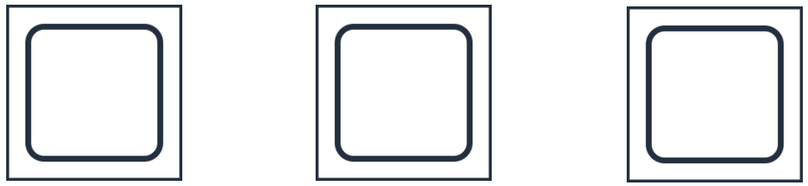
Classic Load Balancer

Application Load Balancer

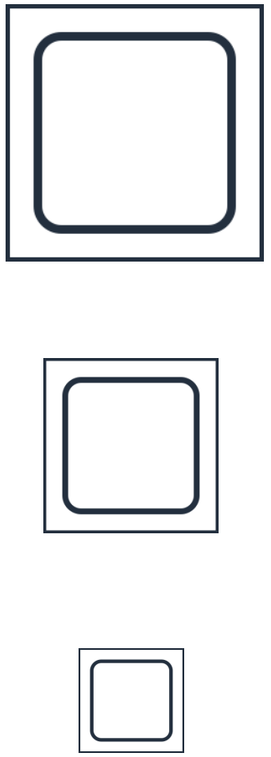
RECOMMENDED

- Highly available
- 1 - 65535
- Health checks
- Session stickiness
- Monitoring / logging
- Content-based routing
- Container-based apps
- WebSockets
- HTTP/2

horizontally

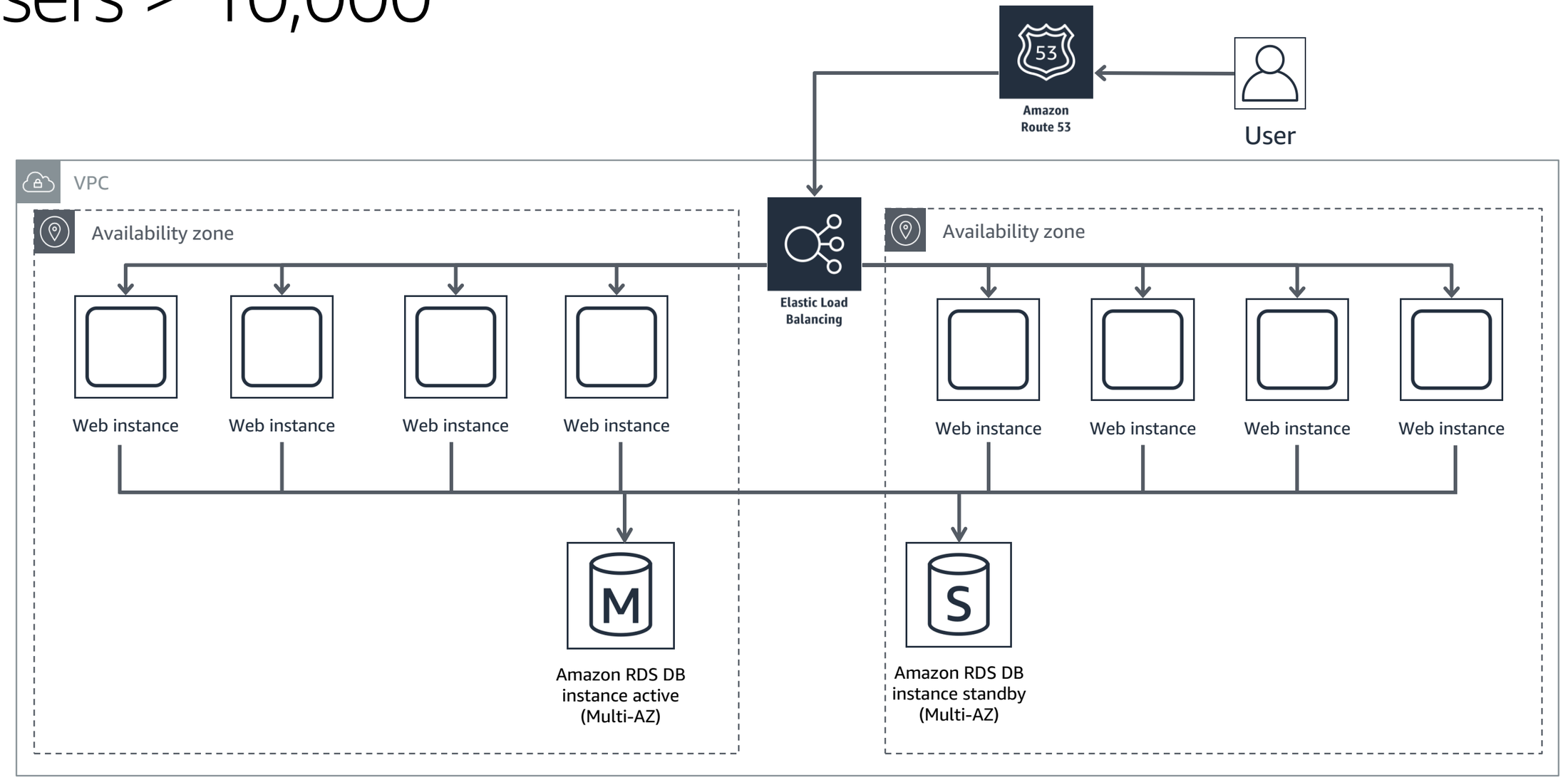


vertically

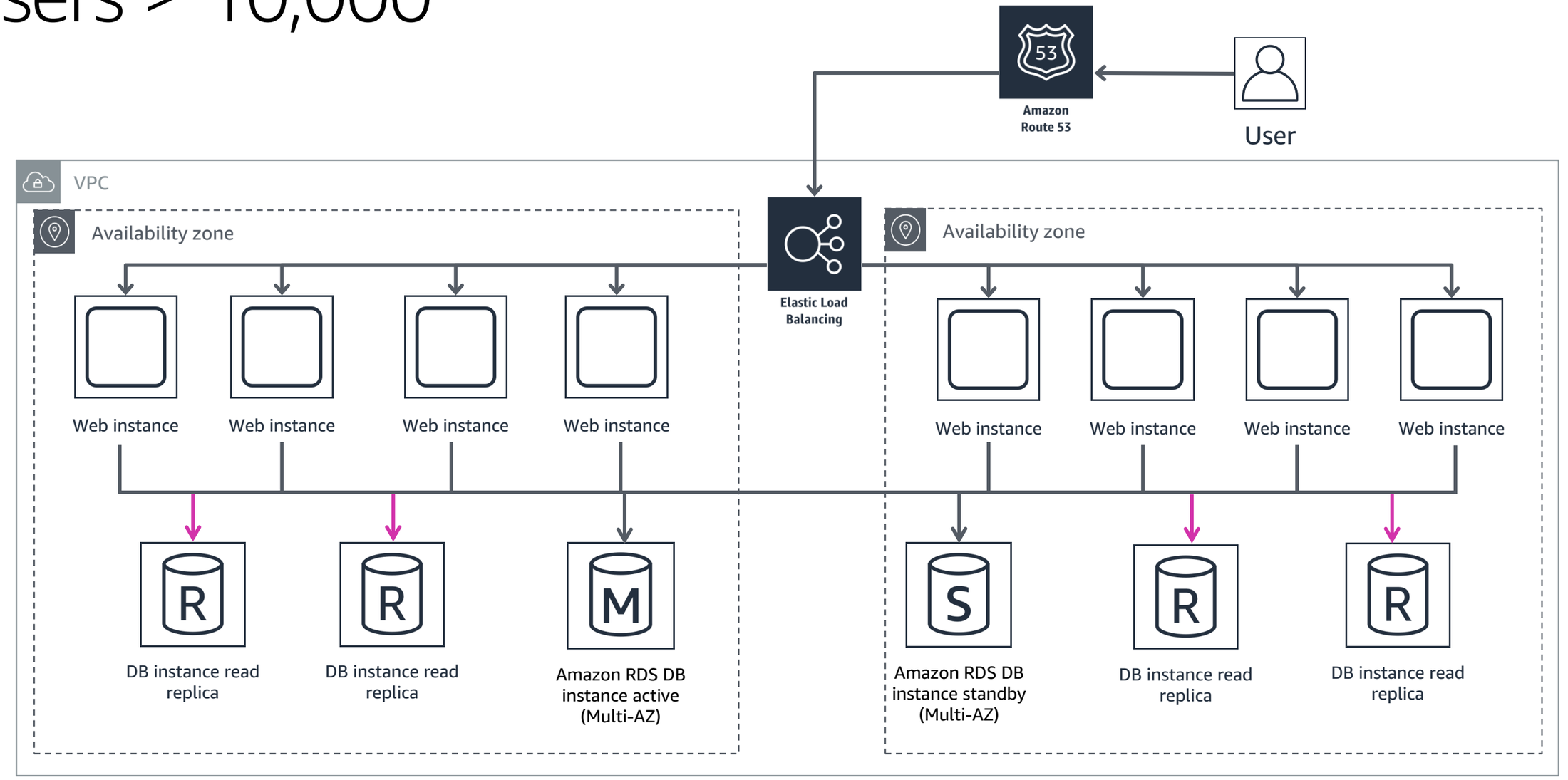


Users >10,000

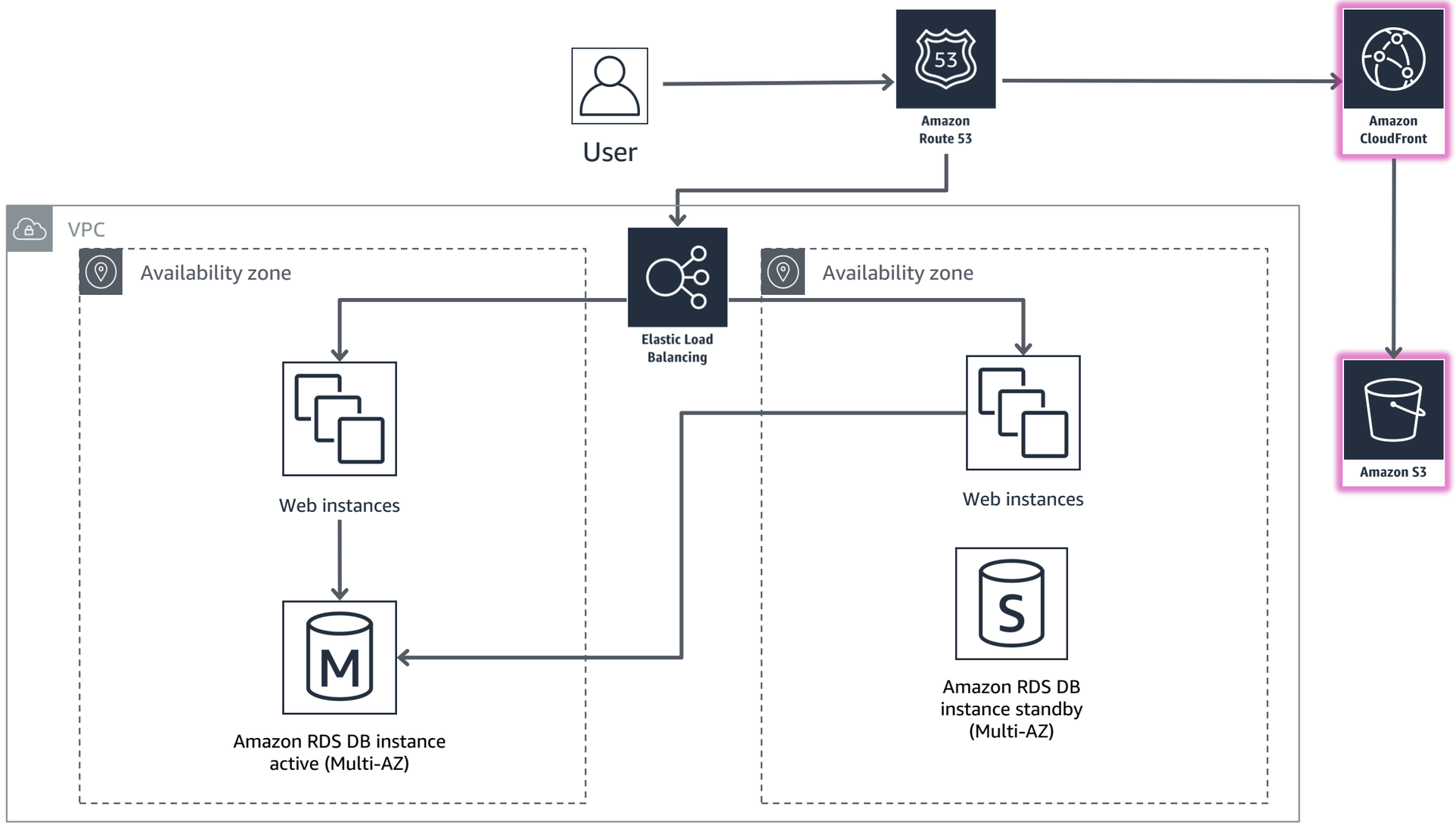
Users > 10,000



Users > 10,000



Shift some load around



Amazon Simple Storage Service (Amazon S3)

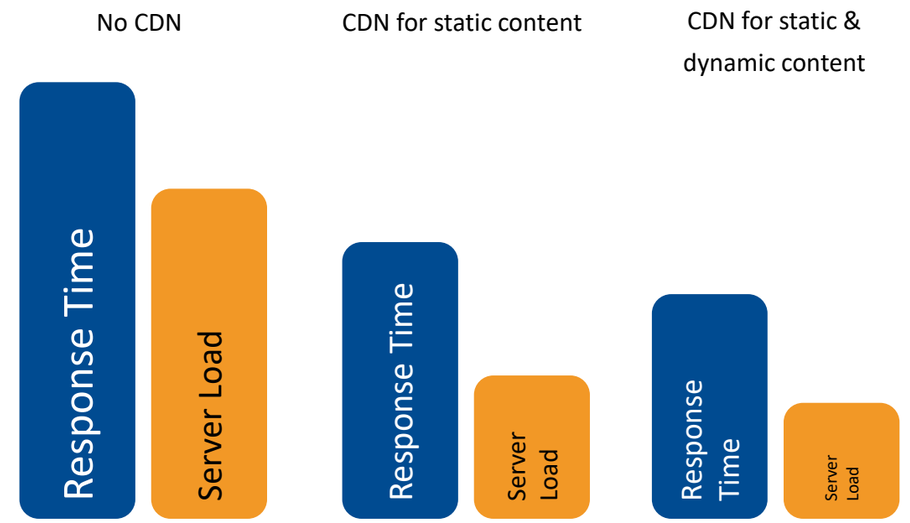
- Object-based storage
- Highly durable
- Great for static assets
- “Infinitely scalable”
- Objects up to 5 TB in size
- Encryption at rest and in transit

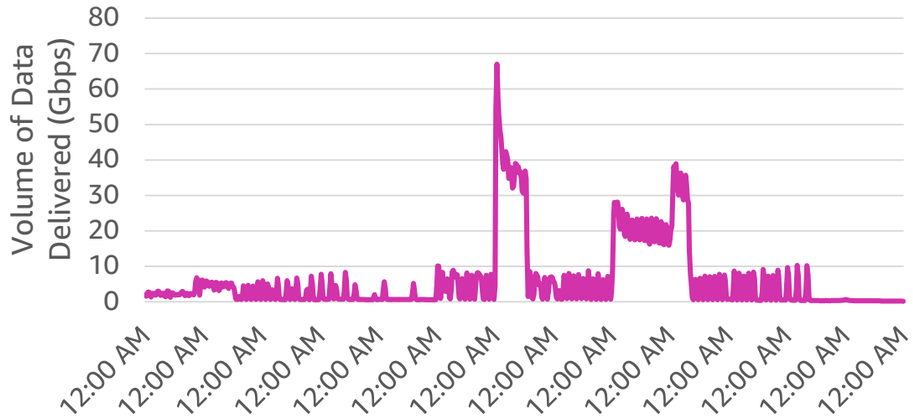
Amazon CloudFront

- Cache content for faster delivery
- Lower load on origin
- Dynamic and static content
- Streaming video
- Custom SSL certificates
- Low TTLs (as short as 0 seconds)
- Optimized for AWS

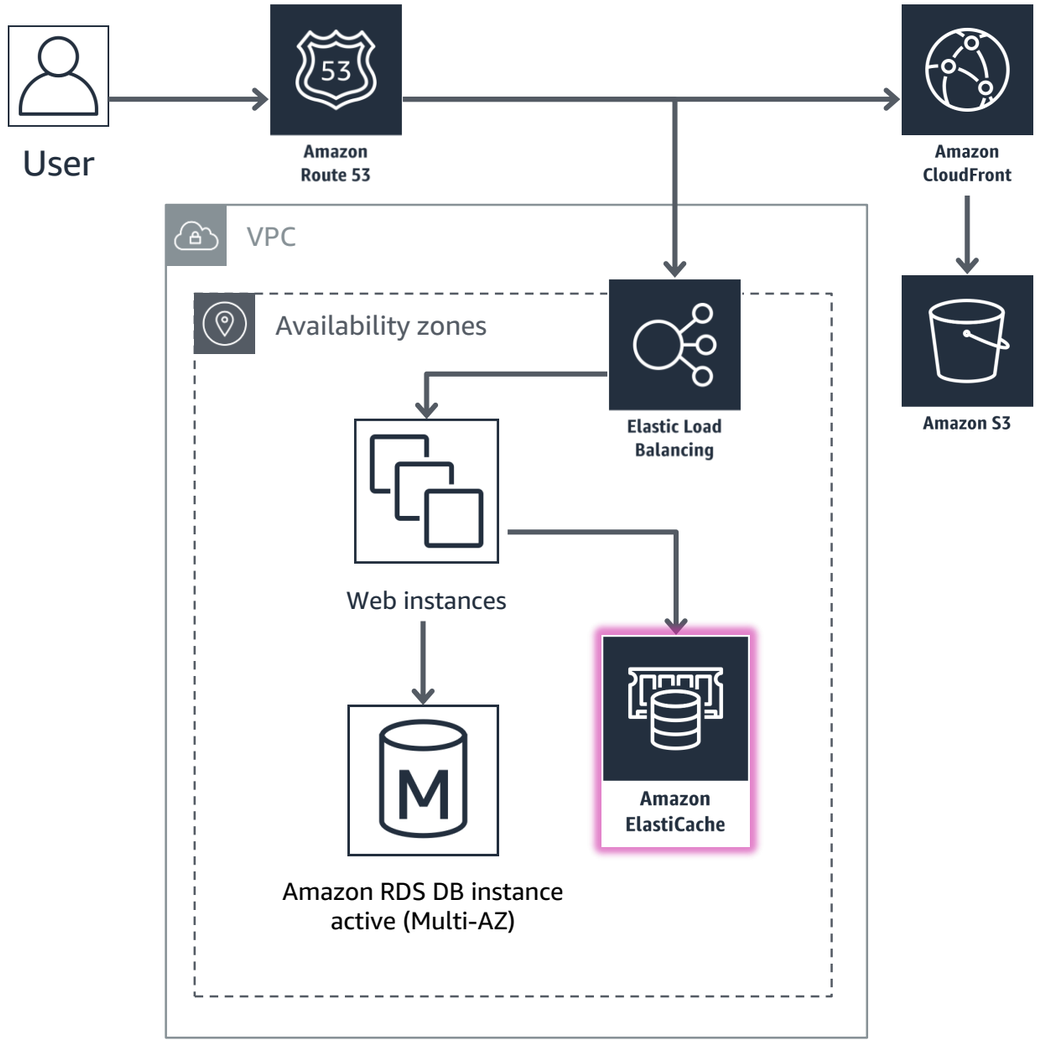
Amazon CloudFront

Amazon CloudFront





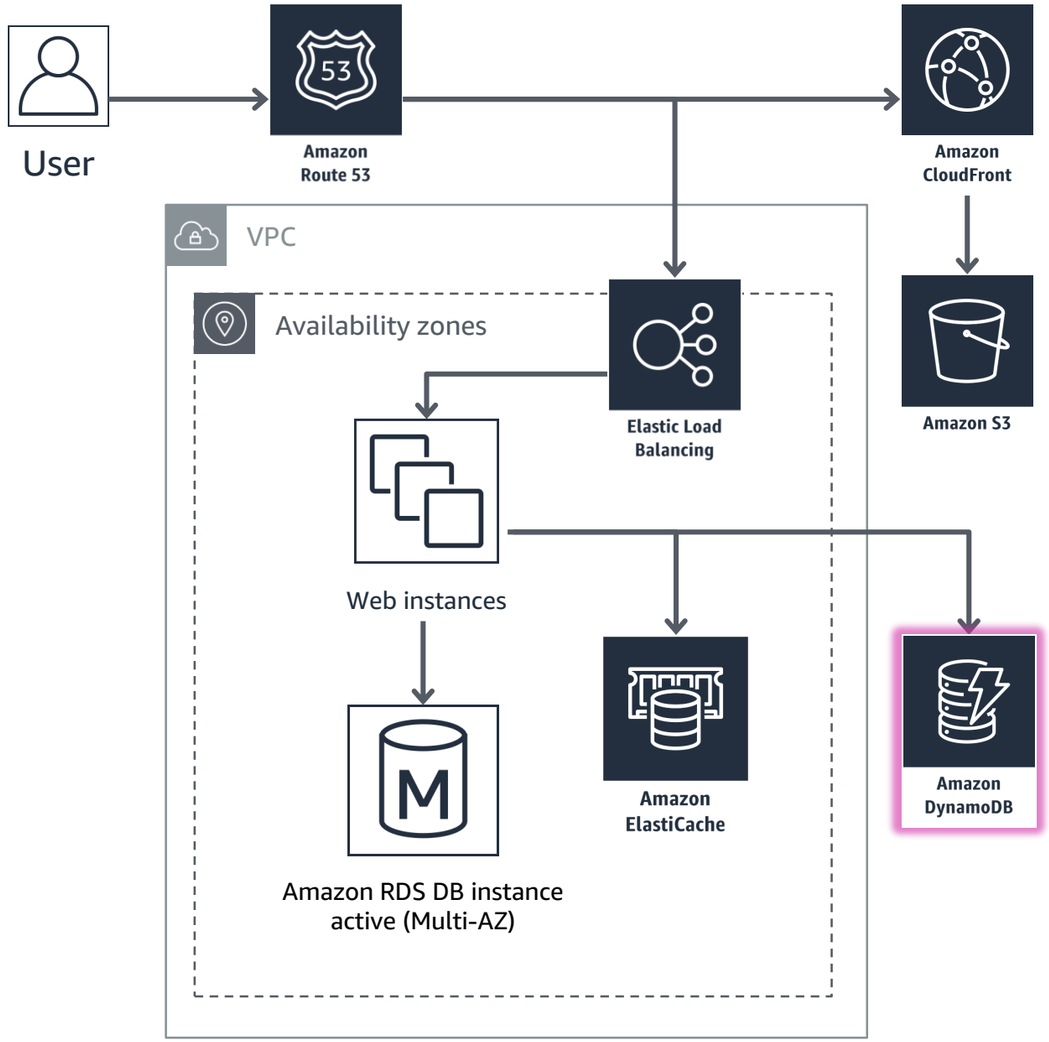
Shift some **more** load around



Amazon ElastiCache

- Managed Memcached or Redis
- Scale from one to many nodes
- Self-healing (replaces dead instance)
- Single-digit ms speeds (usually)
- Local to a single AZ for Memcached
- Multi-AZ possible with Redis

Shift **even more** load around



Amazon DynamoDB

- Managed NoSQL database
- Provisioned throughput
- Fast, predictable performance
- Fully distributed, fault tolerant
- JSON support
- Items up to 400 KB
- Time-to-live (TTL)
- Streams and triggers
- AWS Application Auto Scaling
- Global tables

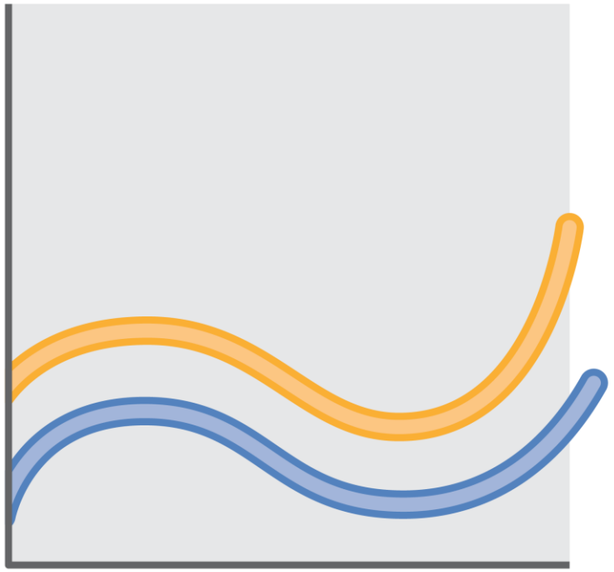
Amazon DynamoDB

AWS Database Migration Service (AWS DMS)!

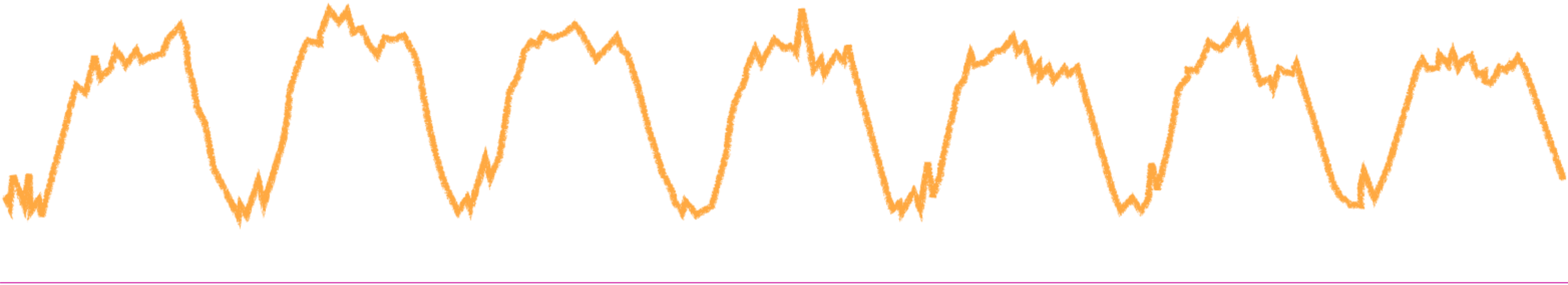
- Managed NoSQL database
- Provisioned throughput
- Fast, predictable performance
- Fully distributed, fault tolerant
- JSON support
- Items up to 400 KB
- Time-to-live (TTL)
- Streams and triggers
- AWS Application Auto Scaling
- Global tables

Now that our web tier is much more lightweight, we can revisit the beginning of our talk . . .

Auto Scaling!

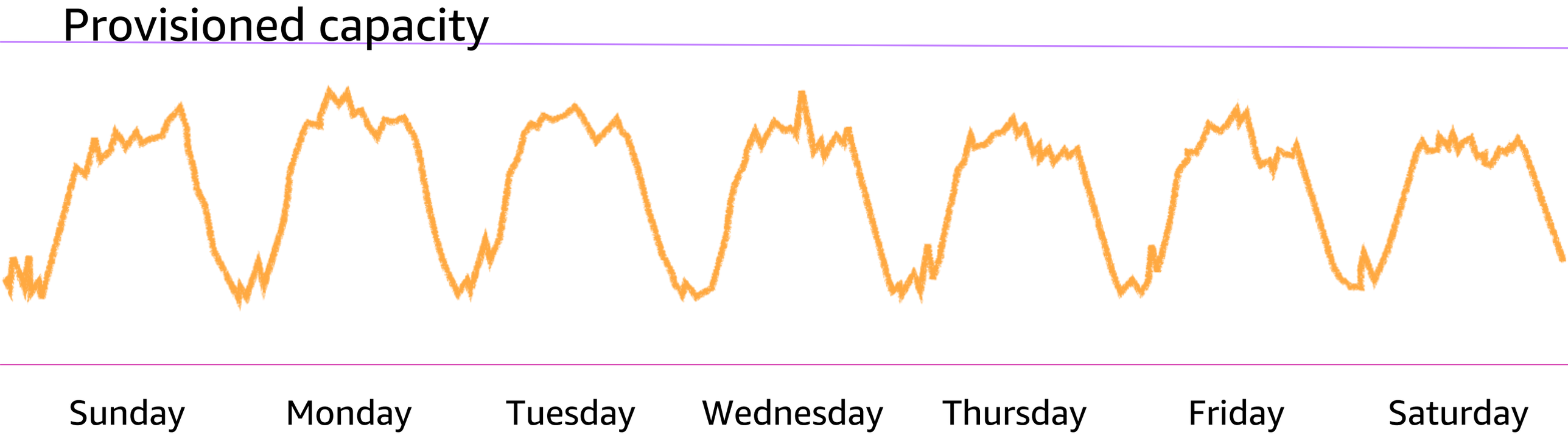


Typical weekly traffic to Amazon.com



Sunday Monday Tuesday Wednesday Thursday Friday Saturday

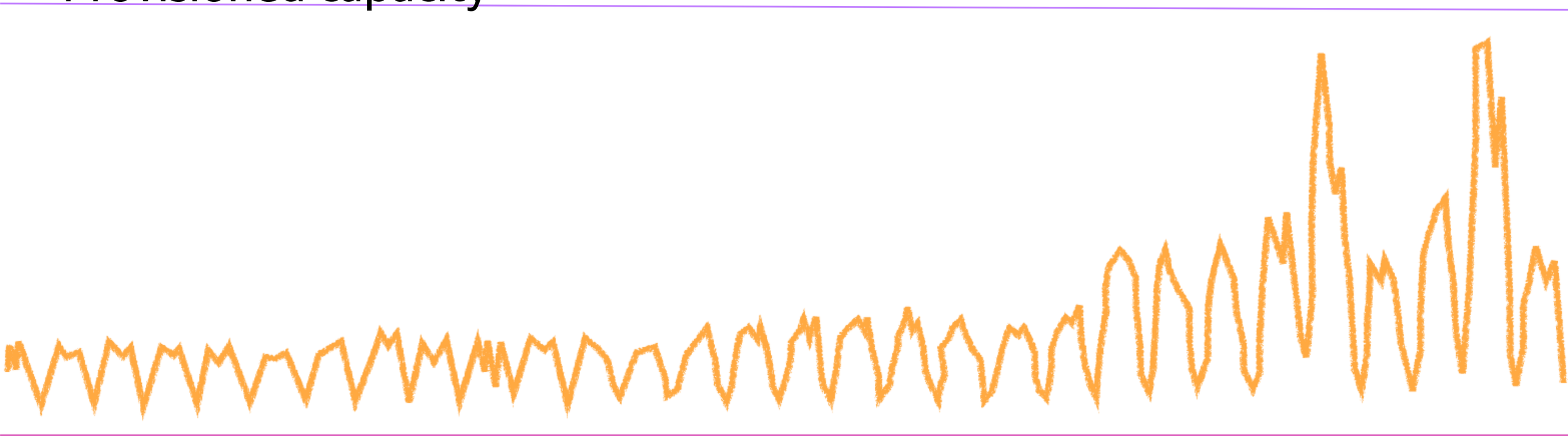
Typical weekly traffic to Amazon.com



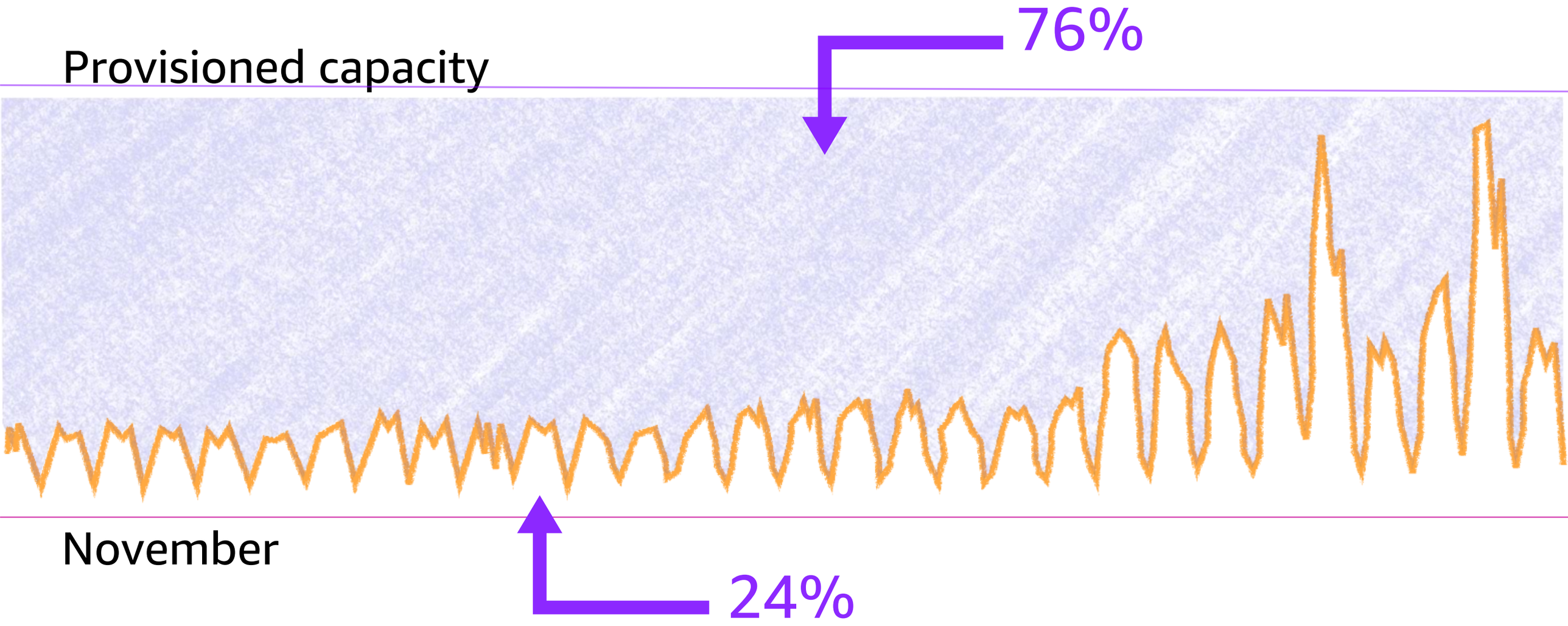
November traffic to Amazon.com

November

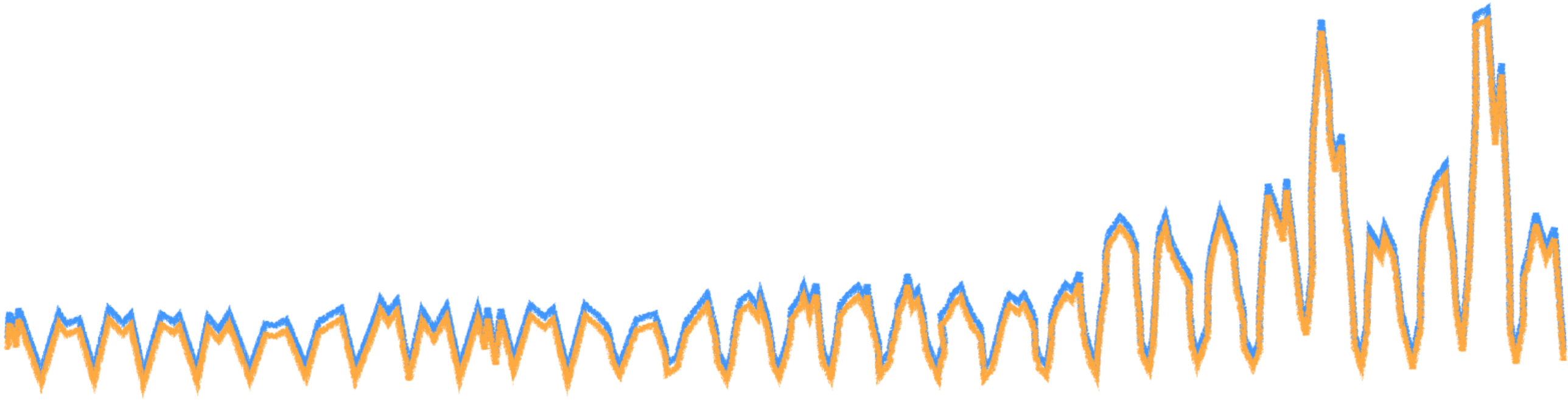
November traffic to Amazon.com



November traffic to Amazon.com

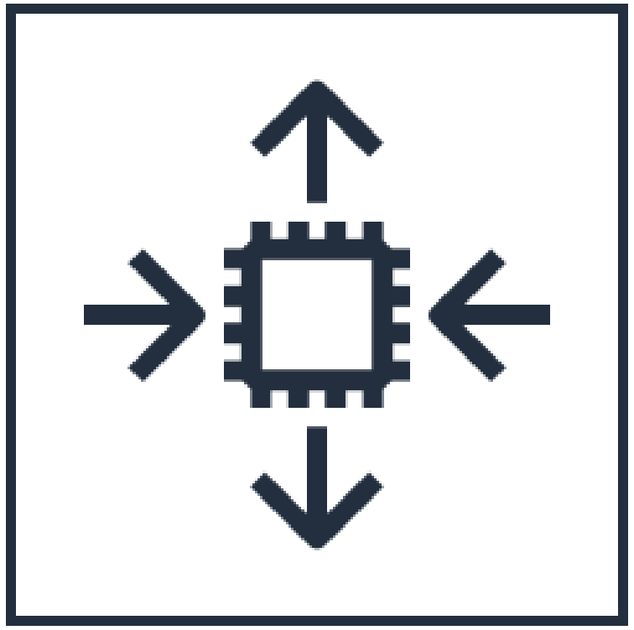


November traffic to Amazon.com



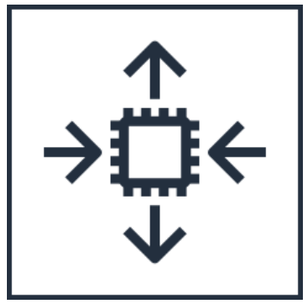
November

Auto Scaling lets you do this!



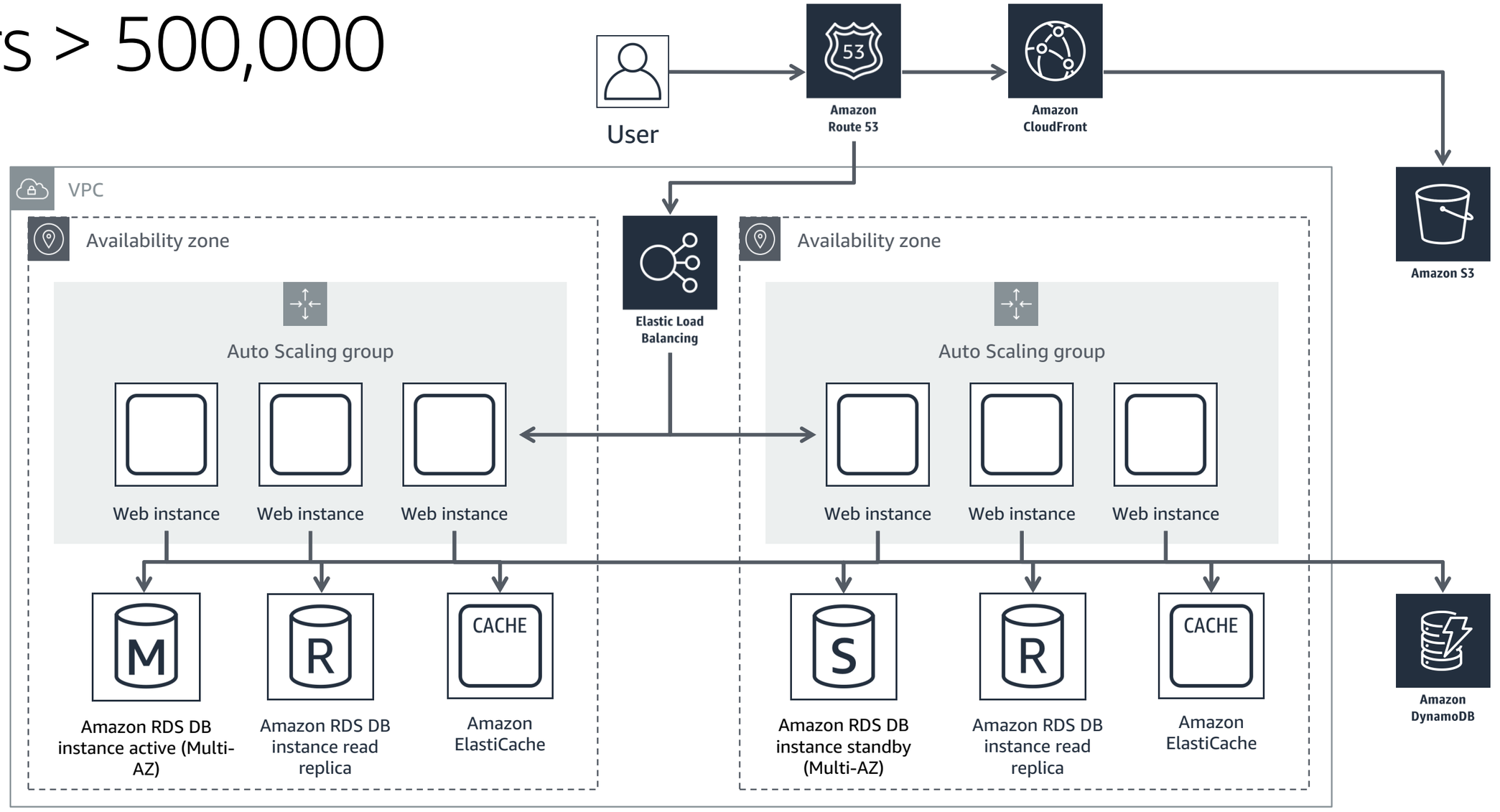
Auto Scaling

- Automatic resizing of compute clusters
- Define min/max pool sizes
- Amazon CloudWatch metrics drive scaling
- On-Demand or Spot Instances

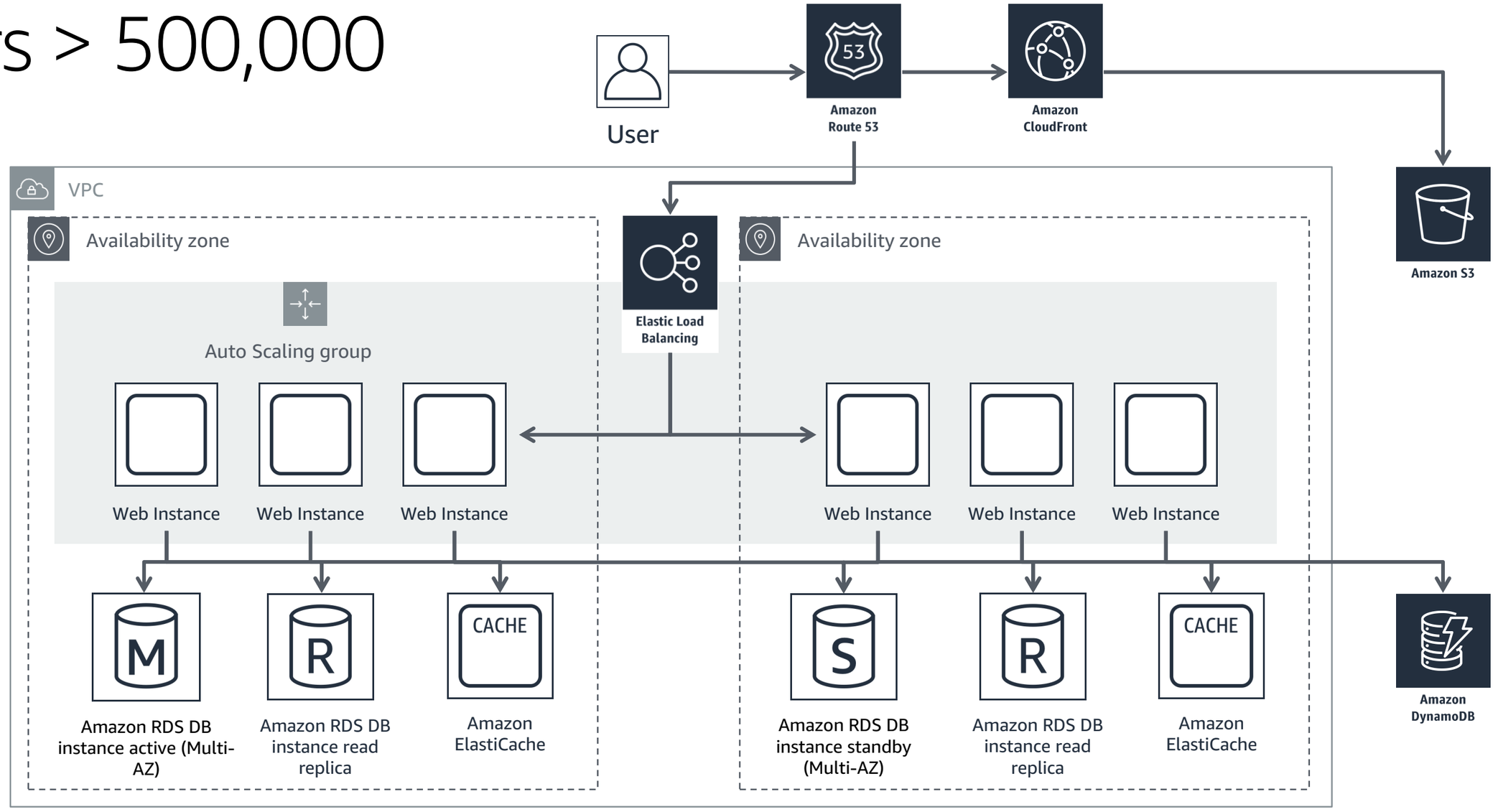


```
aws autoscaling create-auto-scaling-group
--auto-scaling-group-name MyGroup
--launch-configuration-name MyConfig
--min-size 4
--max-size 200
--availability-zones us-west-2c, us-west-2b
```

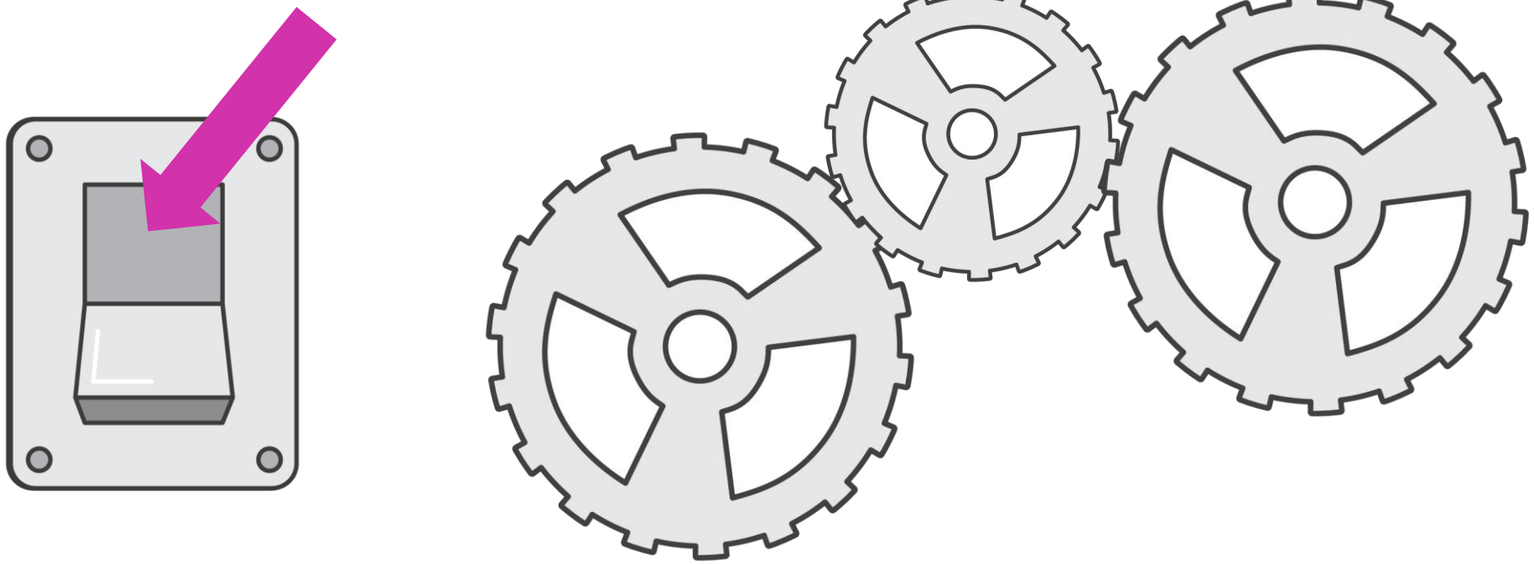
Users > 500,000



Users > 500,000



Use automation



AWS Systems Manager

- In the cloud and on-premises
- Automate admin tasks
- Shell access (no bastions)
- Reasonably priced

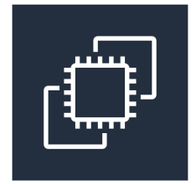
AWS infrastructure automation

Convenience

Control

AWS infrastructure automation

Do it yourself



Convenience

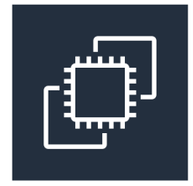
Control

AWS infrastructure automation

Higher-level services

Do it yourself

AWS
OpsWorks



Amazon EC2

Convenience

Control

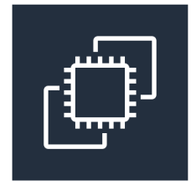
AWS infrastructure automation

Higher-level services

Do it yourself

AWS Elastic Beanstalk

AWS OpsWorks



Amazon EC2

Convenience

Control

AWS infrastructure automation

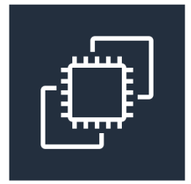
Higher-level services

Do it yourself

AWS Elastic Beanstalk

AWS OpsWorks

Amazon ECS



Amazon EC2

Amazon EKS

Convenience

Control

AWS infrastructure automation

Higher-level services

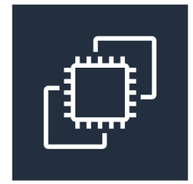
Do it yourself

AWS Fargate

AWS Elastic Beanstalk

AWS OpsWorks

Amazon ECS



Amazon EC2

Amazon EKS

Convenience

Control

AWS infrastructure automation

Higher-level services

Do it yourself

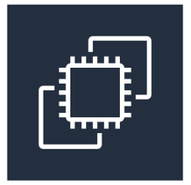
AWS Fargate

AWS Elastic Beanstalk

AWS OpsWorks

AWS CloudFormation

Amazon ECS



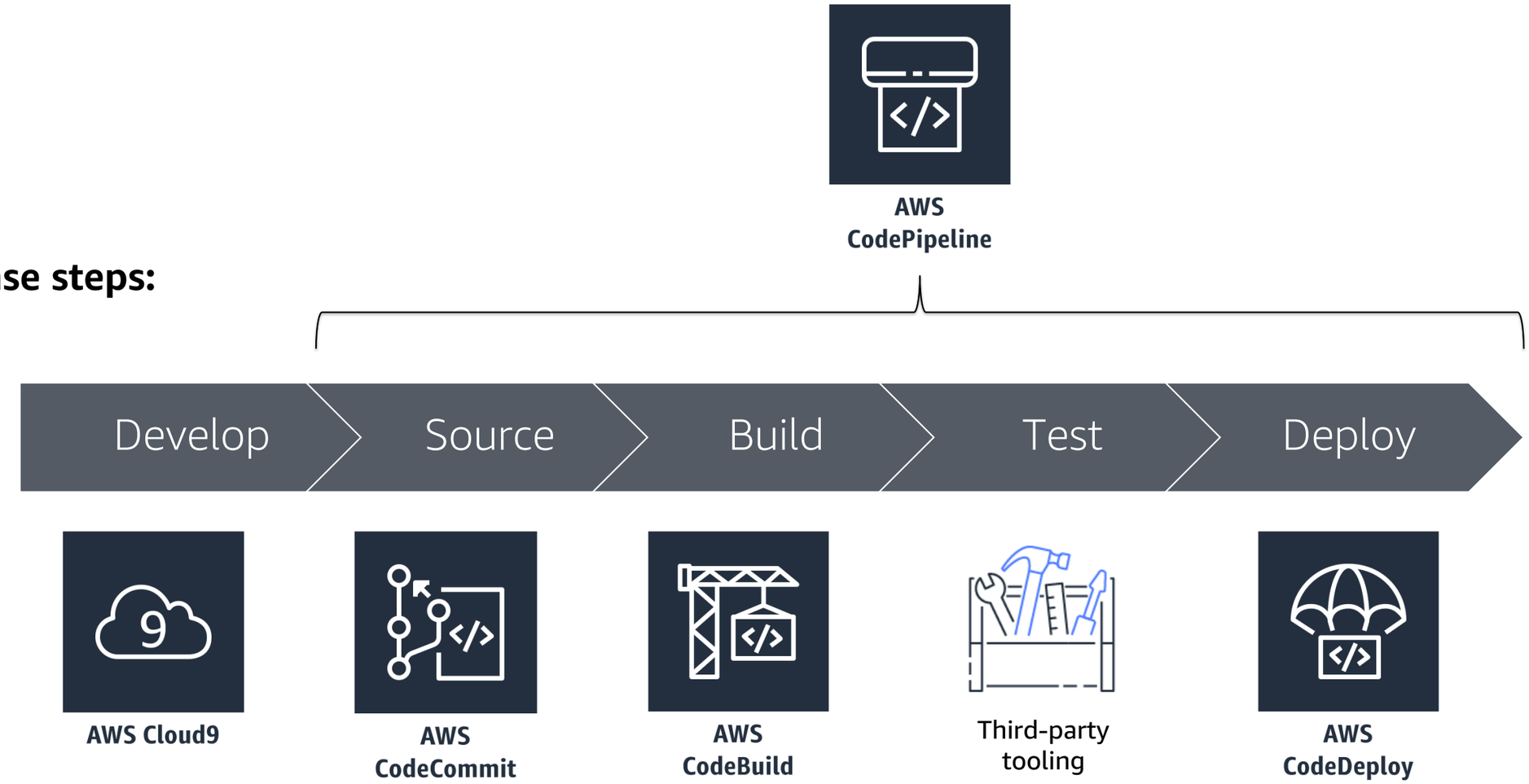
Amazon EC2

Amazon EKS

Convenience

Control

AWS Code Services

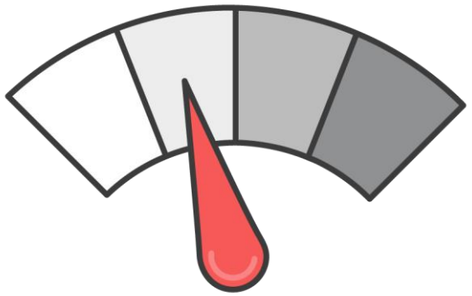


AWS CodeStar

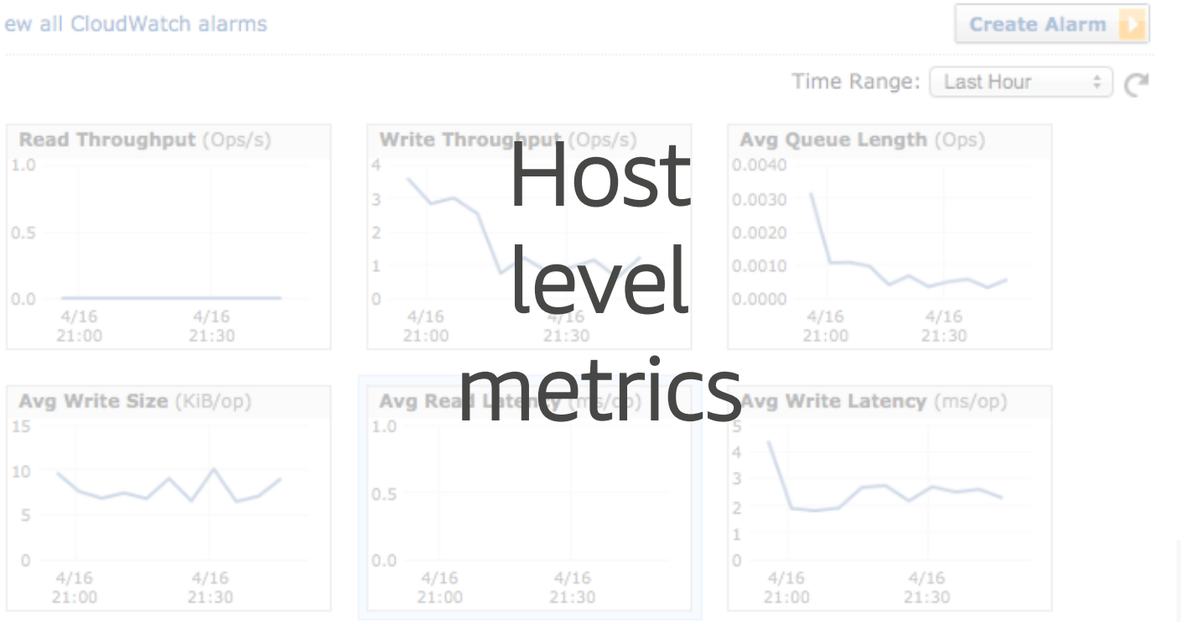
Quickly develop, build, and deploy applications on AWS

- Start developing on AWS in minutes
- Work across your team, securely
- Manage software delivery easily
- Choose from a variety of project templates

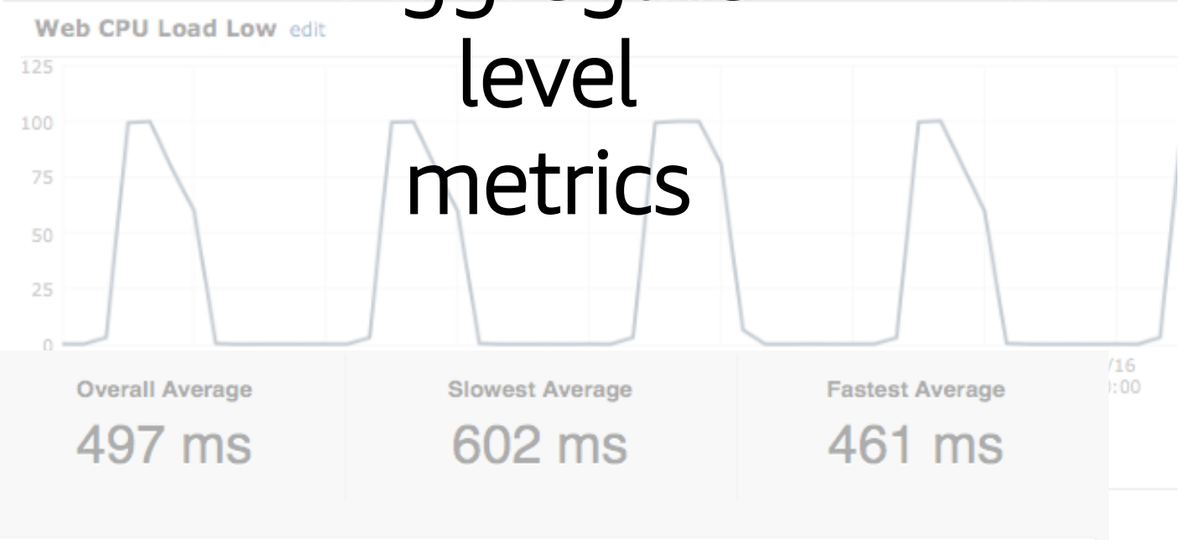
Users >500,000



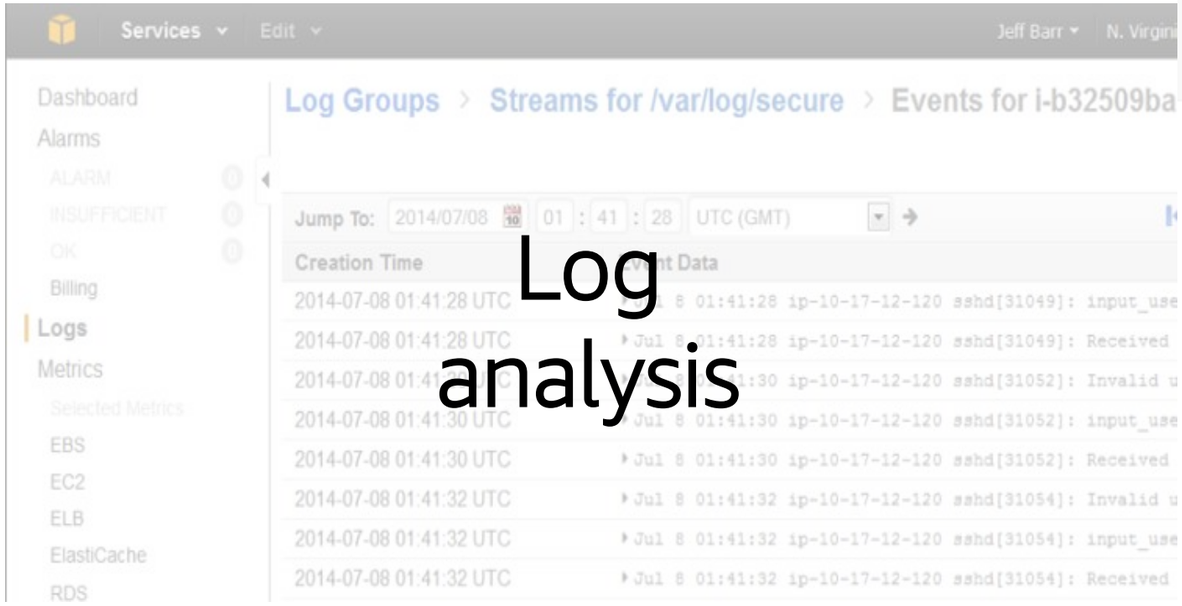
- Monitoring, metrics, and logging
  - If you can’t build it internally, outsource it! (third-party SaaS)
- What are customers saying?
- Try to squeeze as much performance out of each service/component



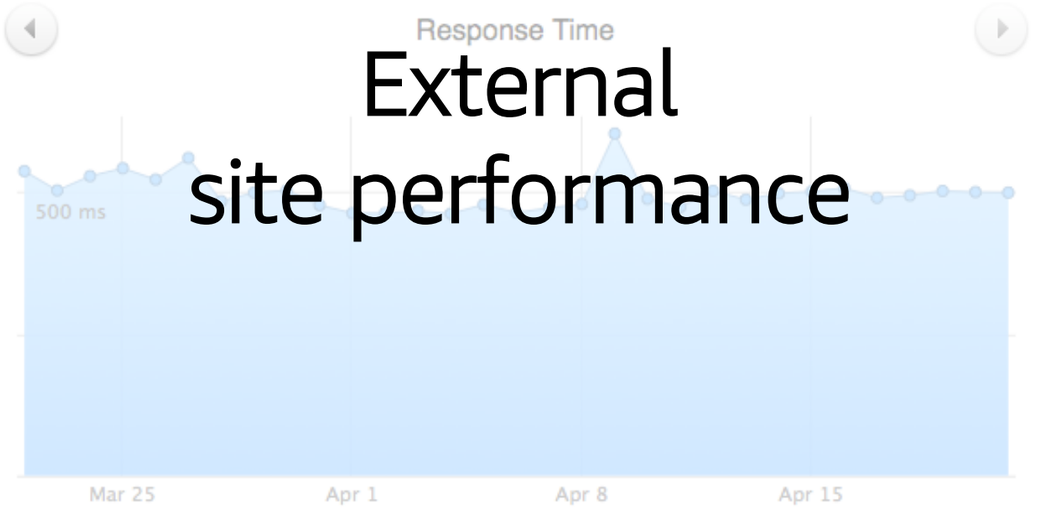
Host level metrics



Aggregate level metrics



Log analysis



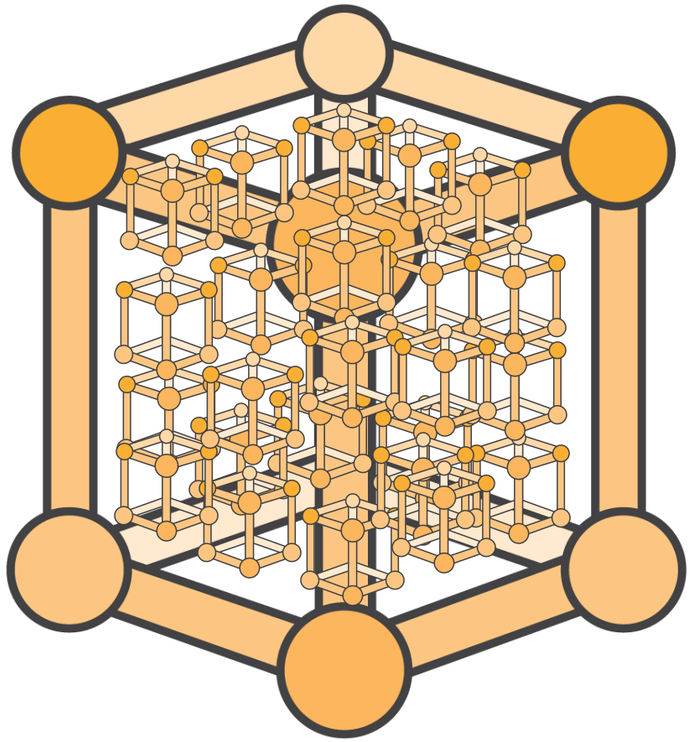
External site performance

Amazon CloudWatch

- Collect: Metrics and logs
- Monitor: Alarms and dashboards
- Act: Autoscaling and events
- Analyze: Trends and metric math
- Compliance and security

There are further improvements to be made in breaking apart our web/app layer

The monolithic architecture



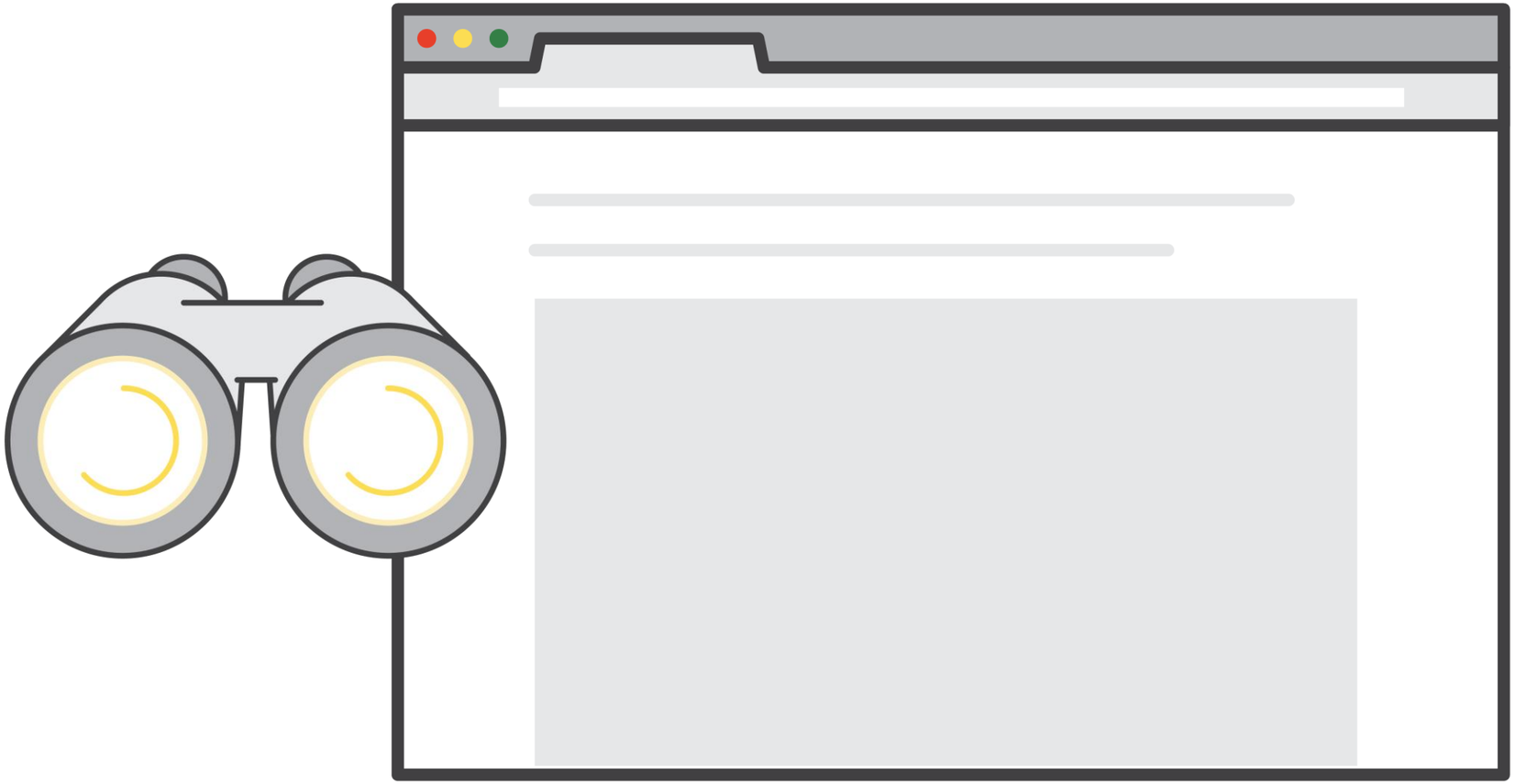
User interface

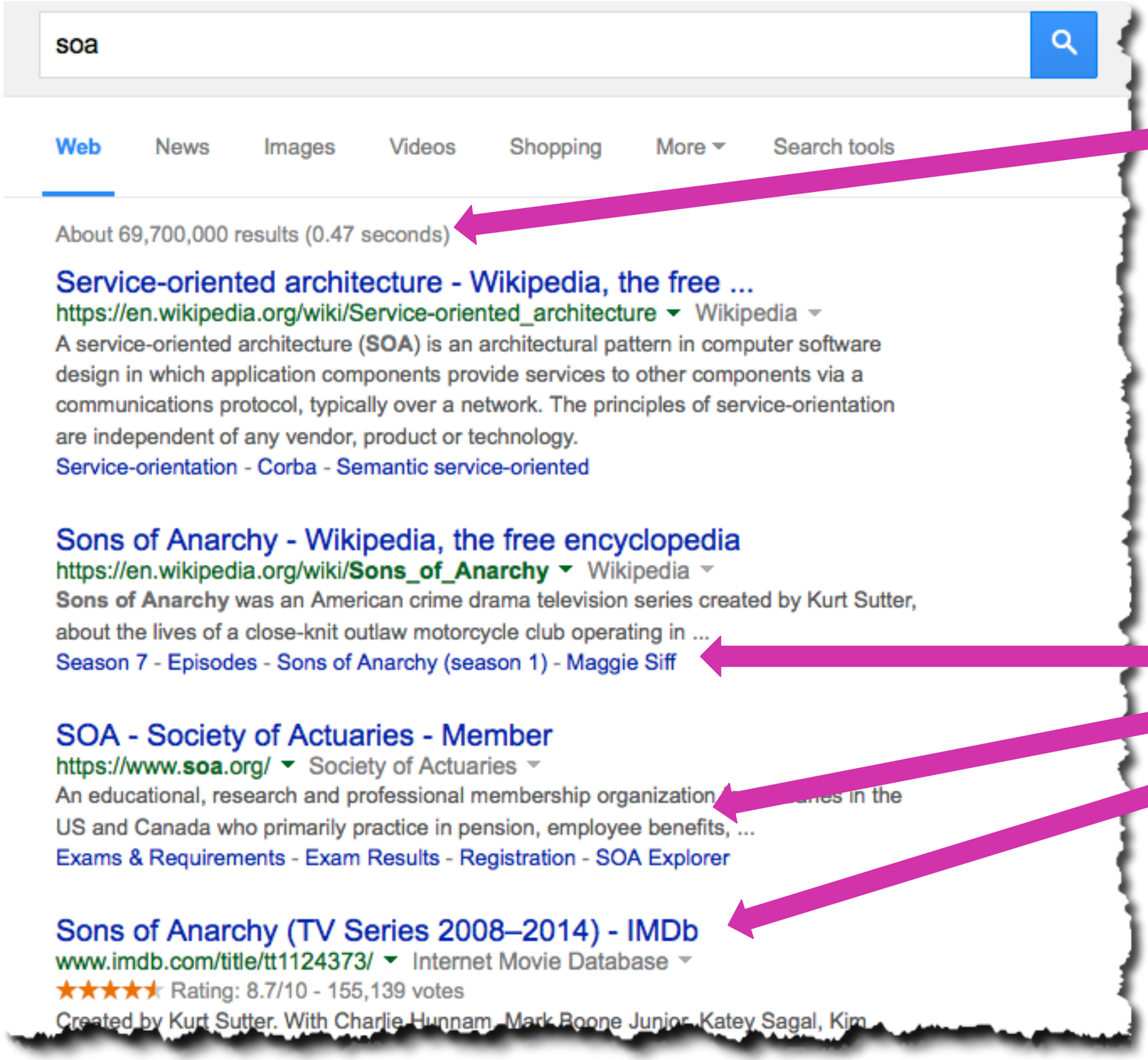
Business logic

Data access

SOA

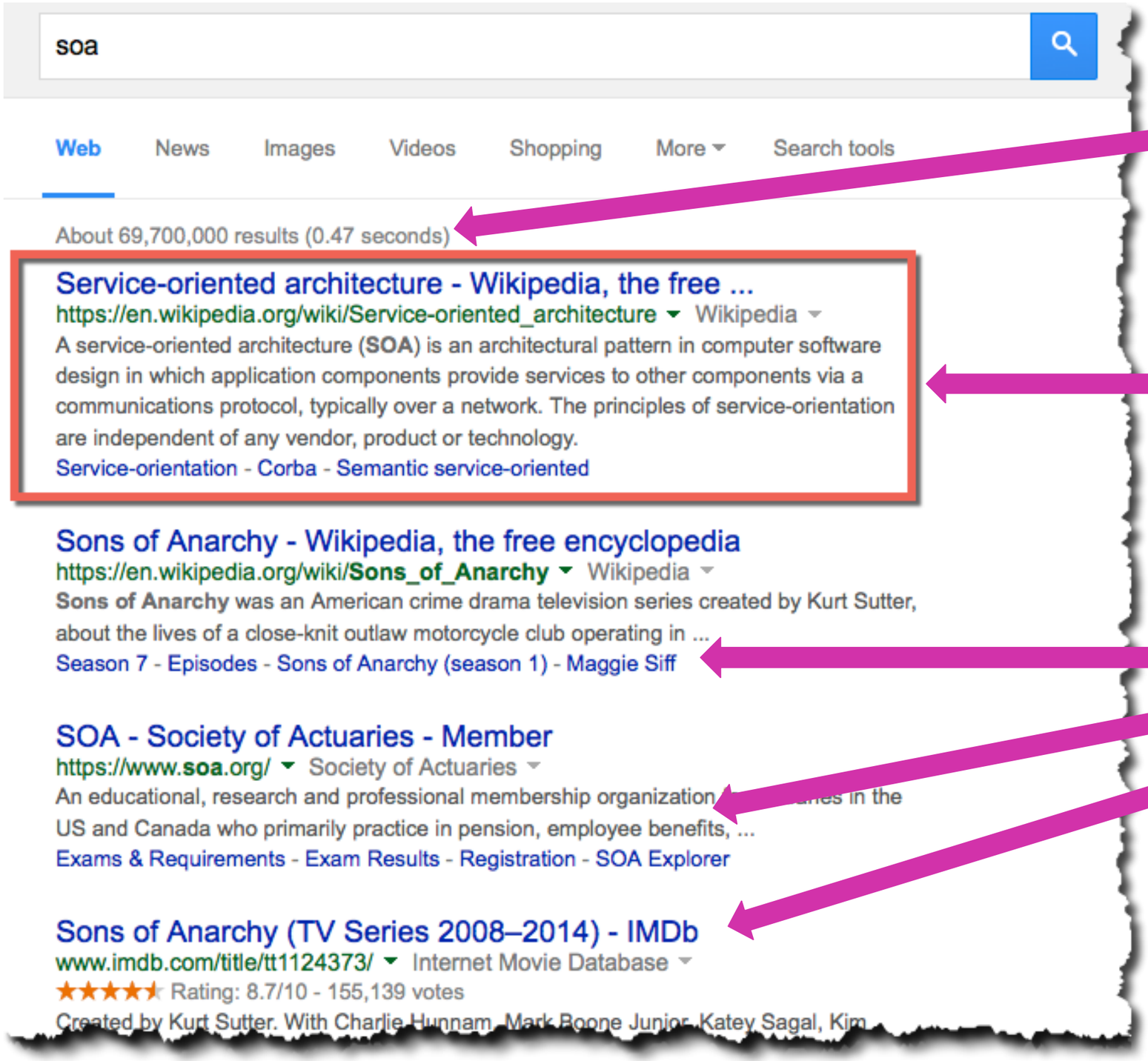
What does this mean?





Now that's a lot of things to read!

This is NOT where we want to start!

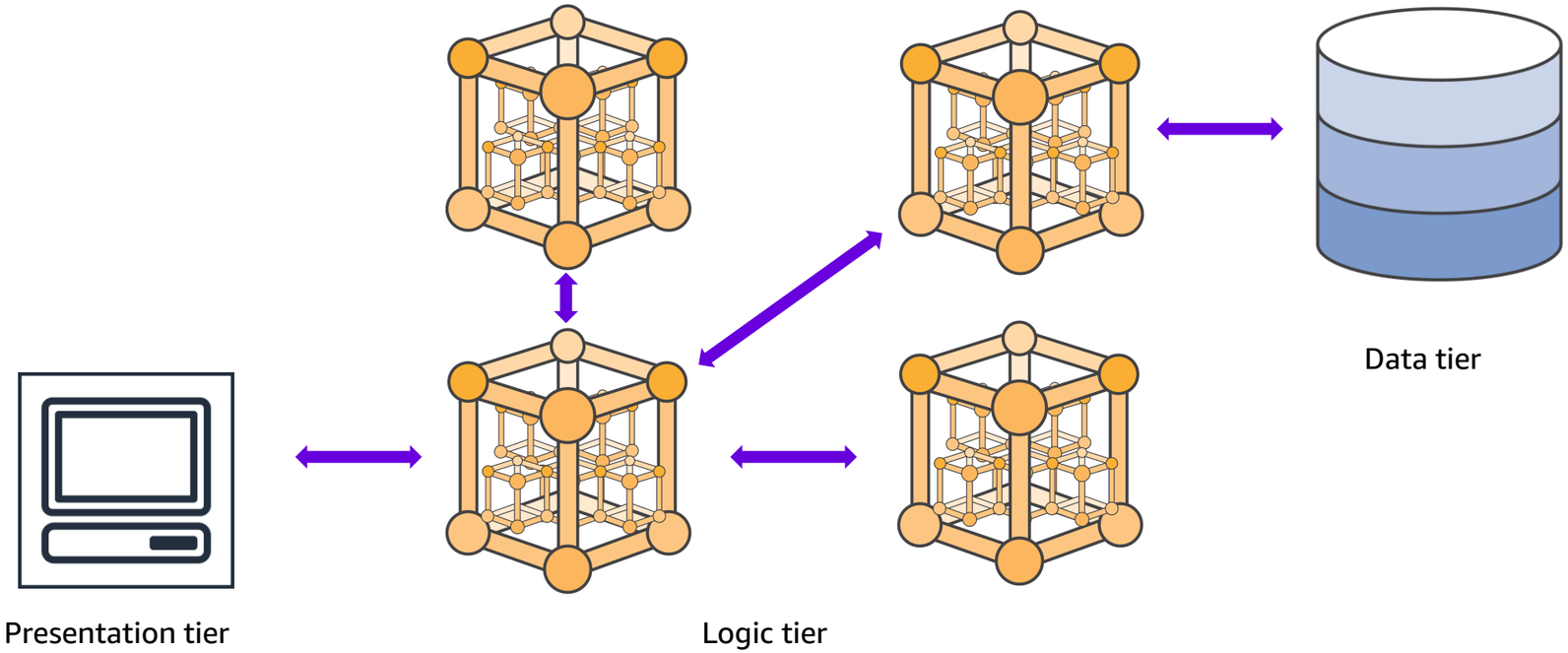


Now that's a lot of things to read!

This IS where we want to start!

This is NOT where we want to start!

The service-oriented architecture

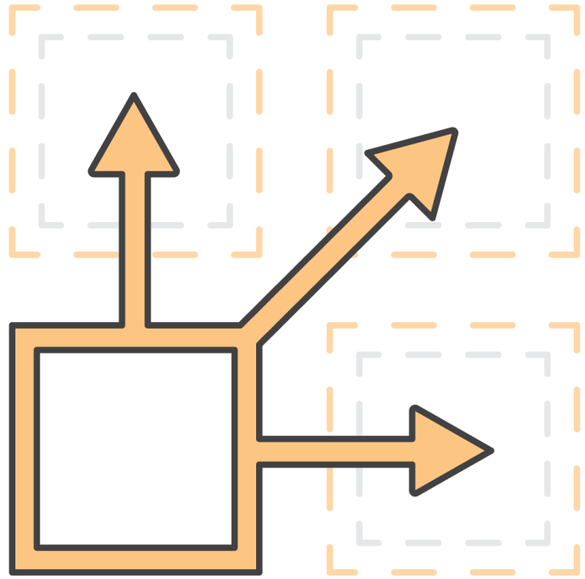


SOAing

Move services into their own tiers

- Treat them separately
- Scale them independently

It offers flexibility and greater understanding of each component



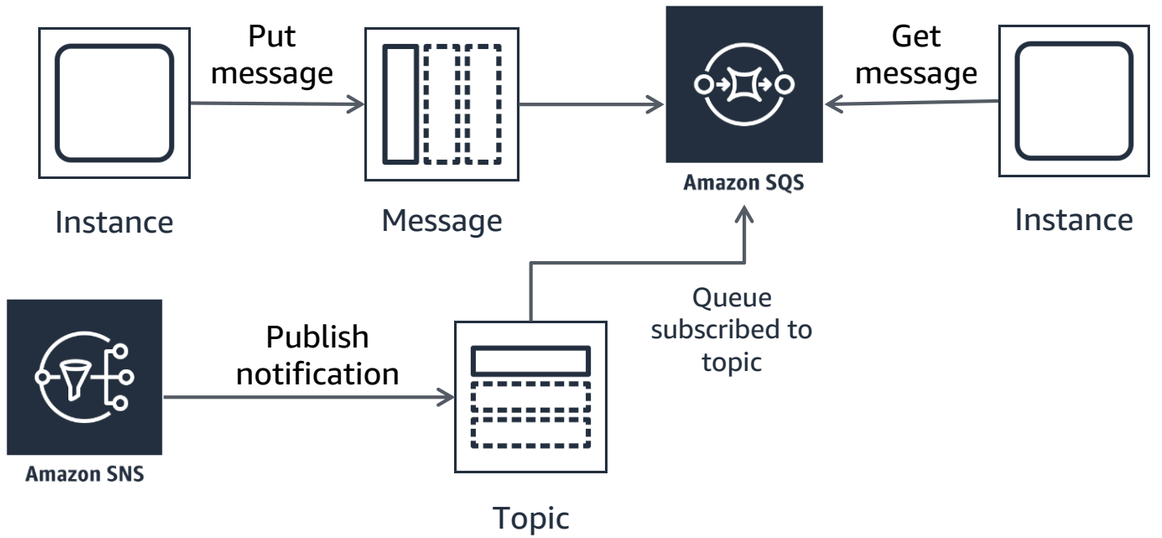
Serverless = winning

Don’t reinvent the wheel

- API
- Queuing
- Transcoding
- Search
- Databases
- Monitoring
- Logging
- Compute
- Machine learning

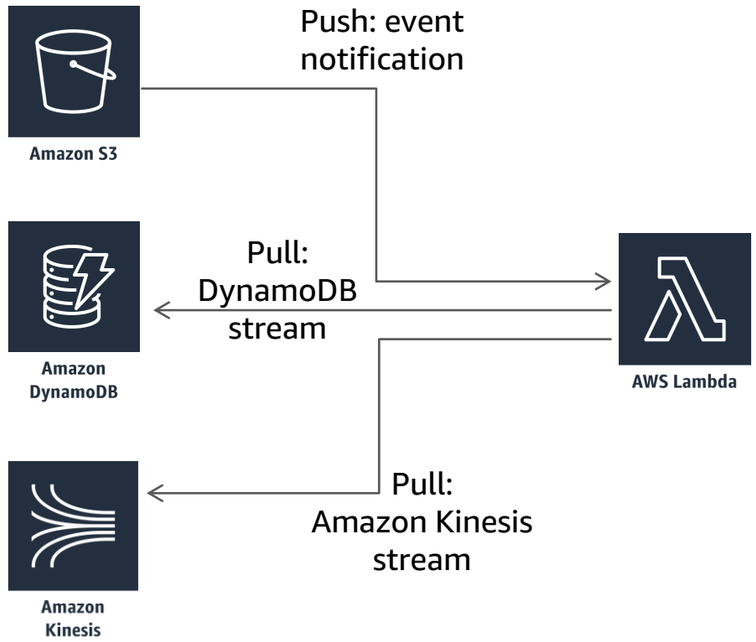
Loose coupling – Amazon SQS and Amazon SNS

- Reliable (multi-AZ)
- Scalable (unlimited messages)
- Secure (queue authentication)
- Simple (simple APIs)
- FIFO now supported



Event-driven compute – AWS Lambda

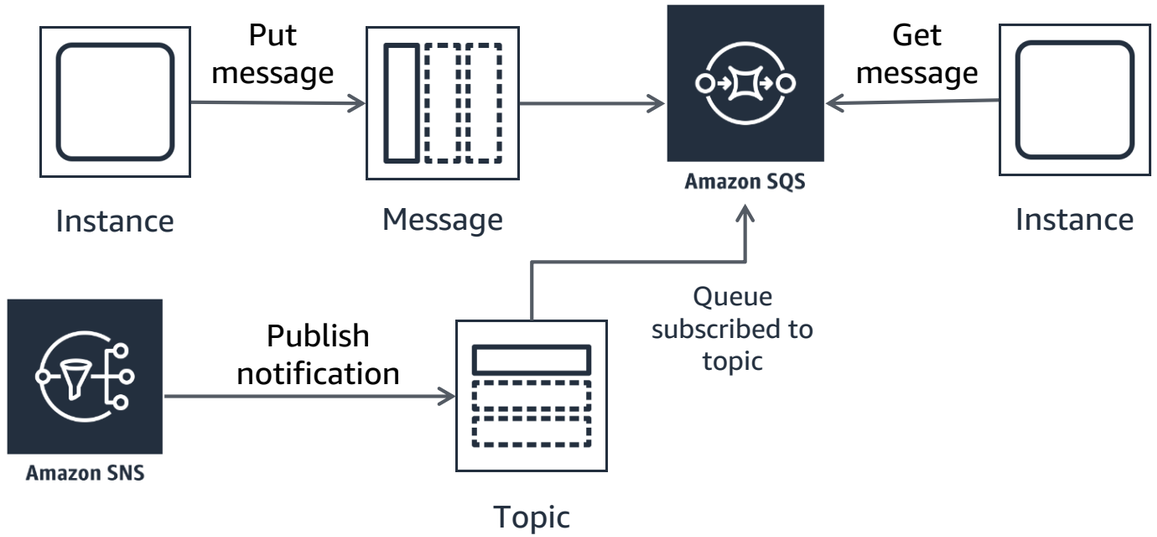
- Functions triggered by events
- Node.js (JavaScript), Java, Python, and C#
- Serverless
- Implicit scaling

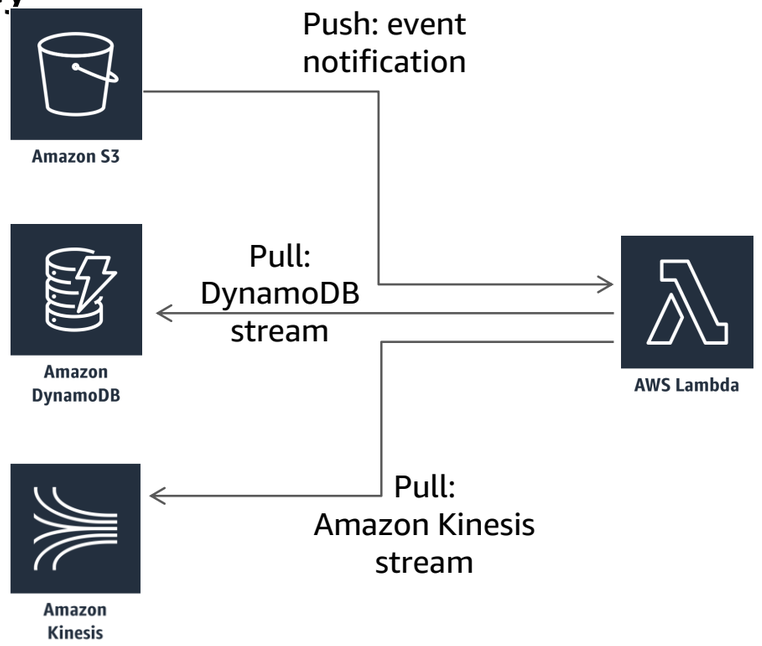


Loose coupling sets you free!

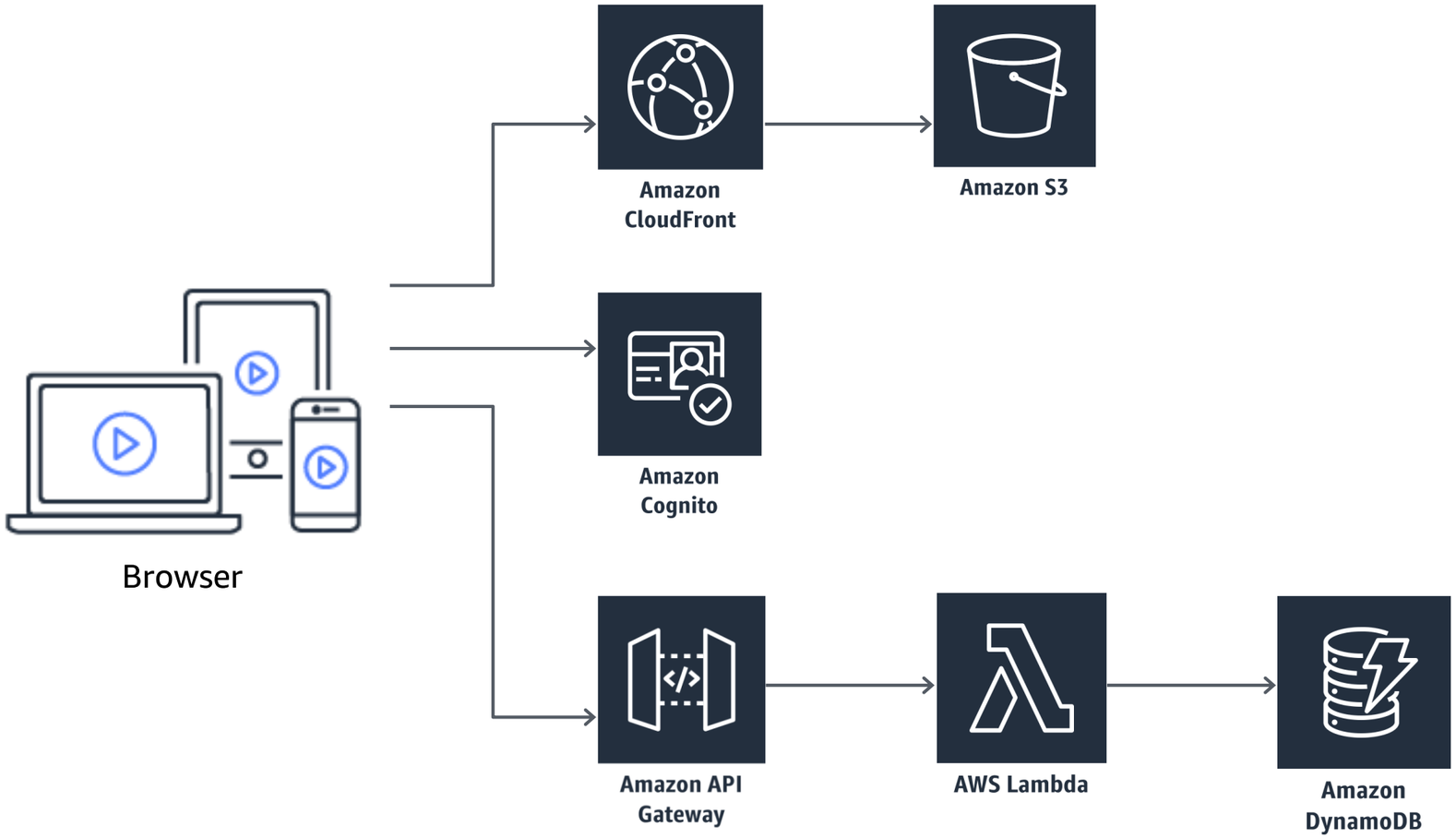
The looser they're coupled, the bigger they scale

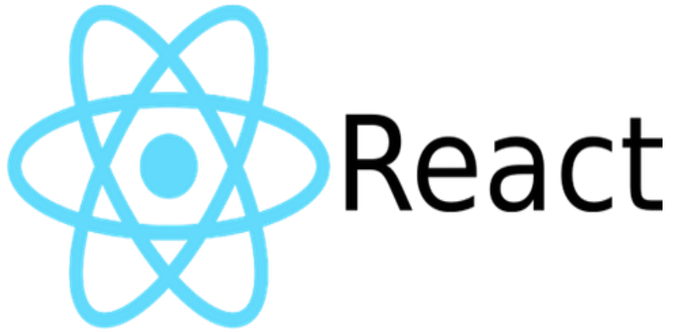
- Independent components
- Design everything as a black box
- Decouple interactions
- Favor services with built-in redundancy and scalability
  - Don’t build your own!



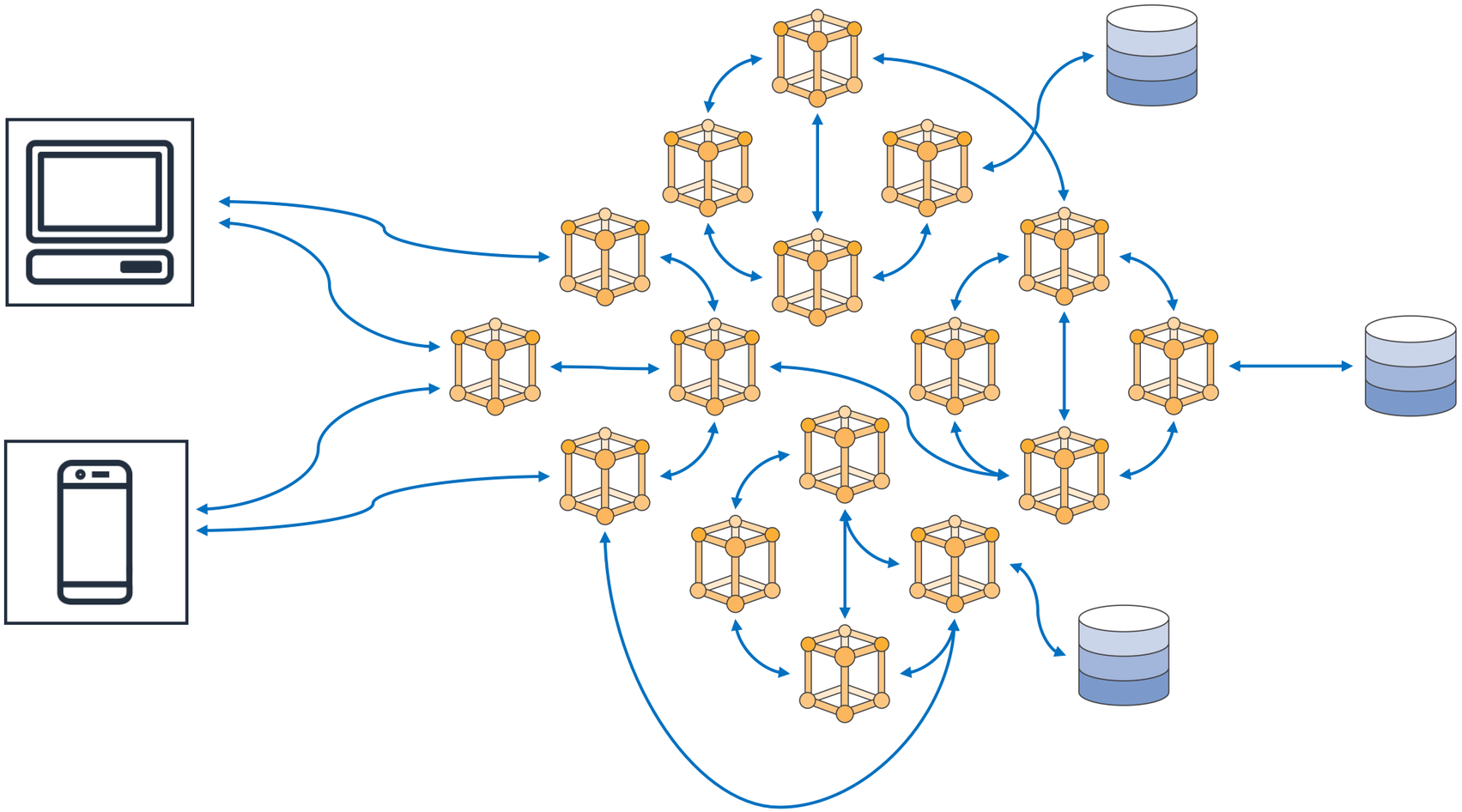


Serverless Web application

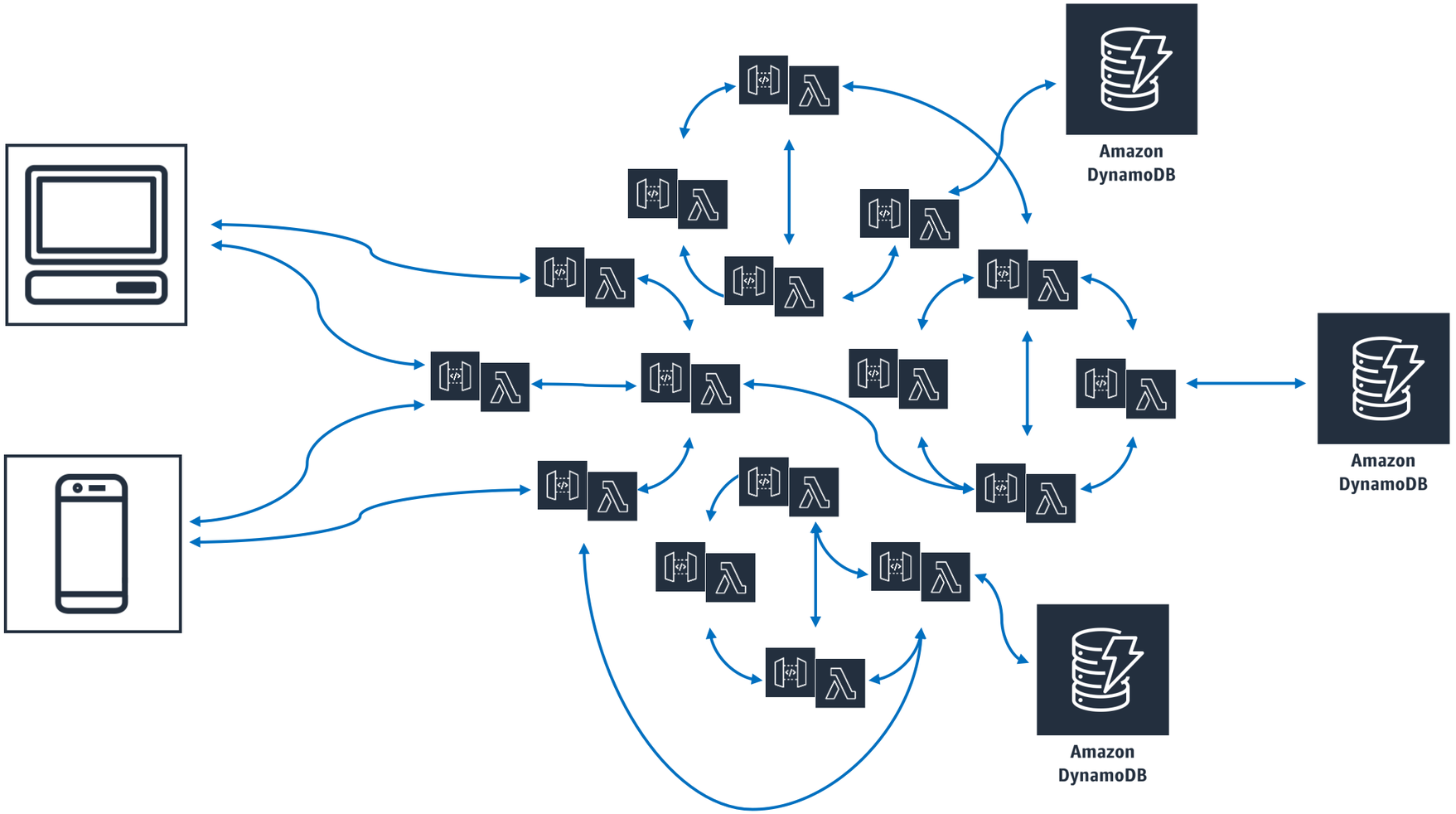




The microservices architecture



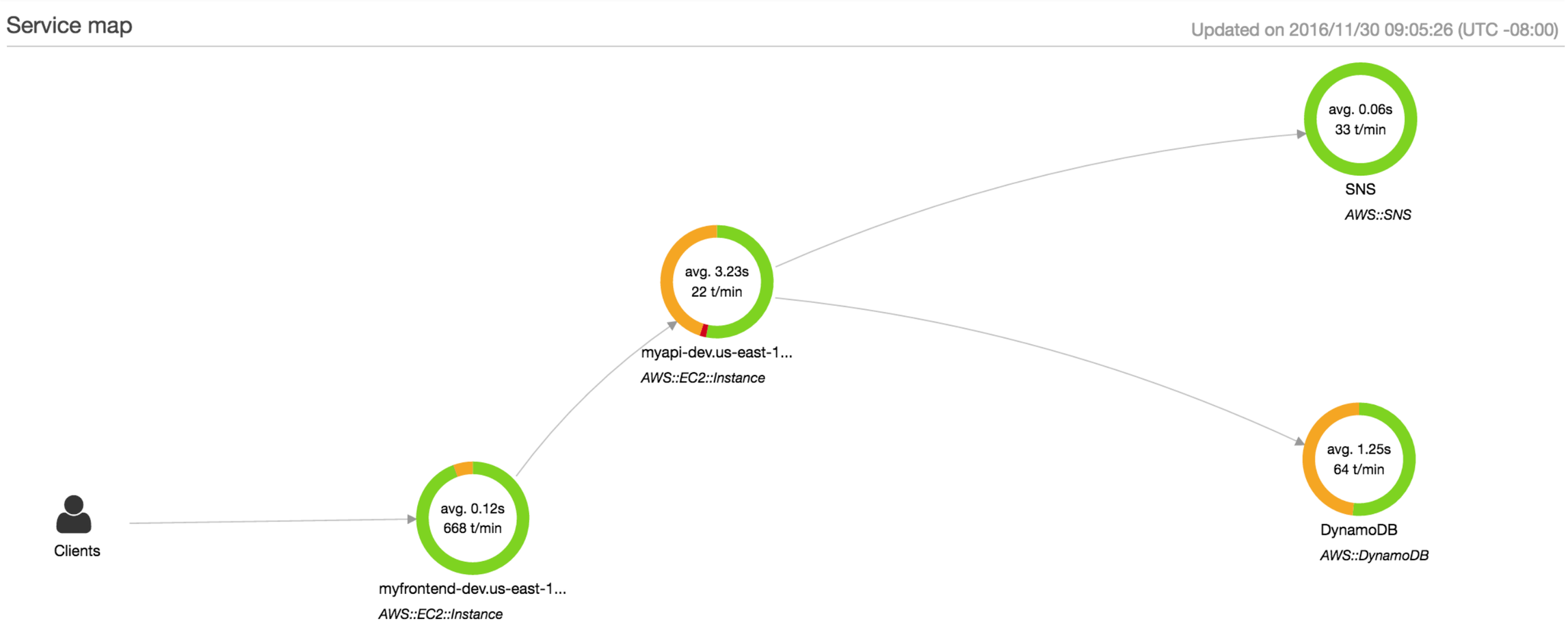
The microservices architecture



AWS X-Ray

- Identify performance bottlenecks and errors
- Pinpoint issues to specific service(s) in your application
- Identify impact of issues on users of the application
- Visualize the service call graph of your application

Visualize service call graph



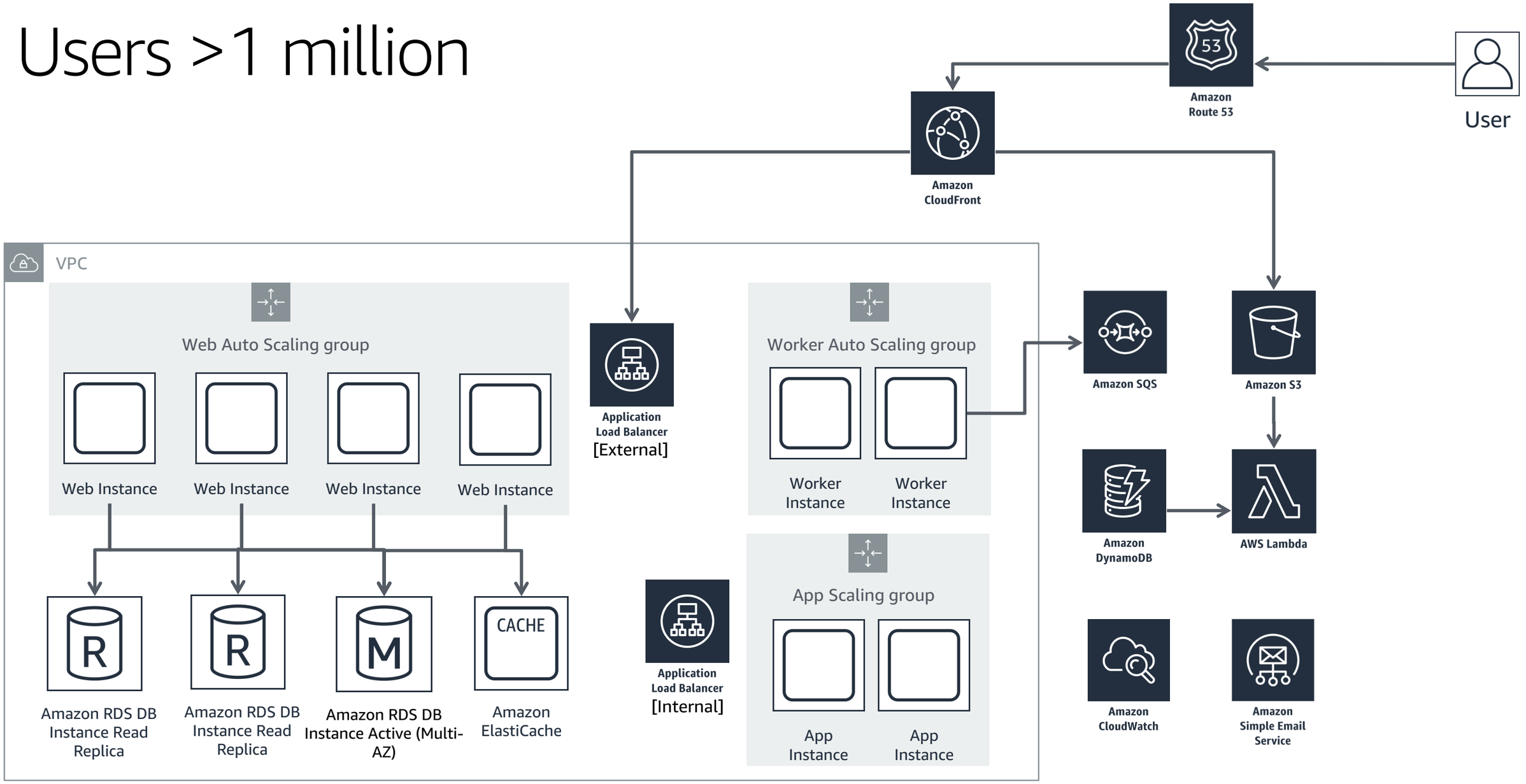
Users >1,000,000

Users >1 million

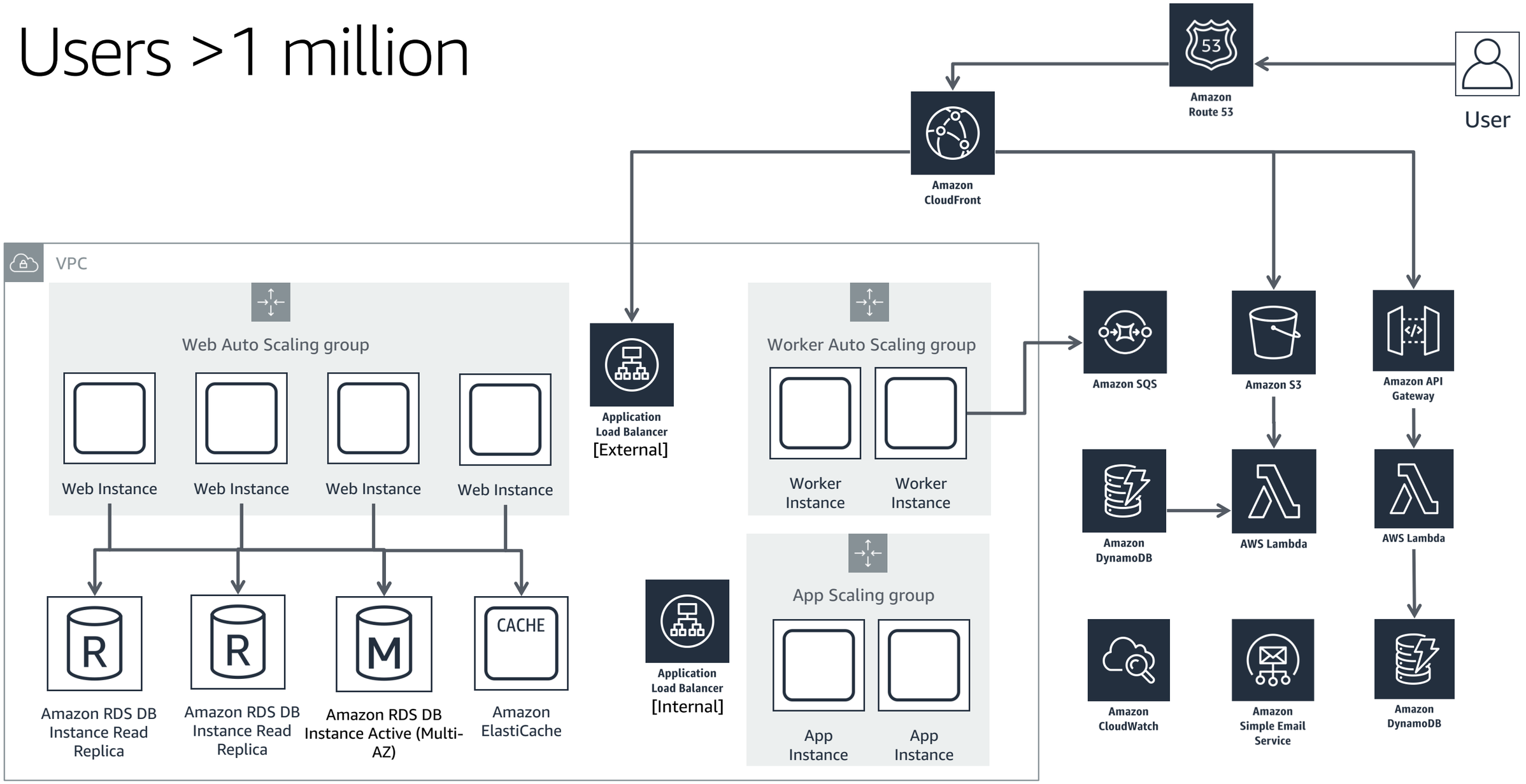
Reaching a million and above is going to require some bit of all the previous things

- Multi-AZ
- Elastic Load Balancing between tiers
- Auto Scaling
- Service oriented architecture (SOA)
- Serving content smartly (Amazon S3/CloudFront)
- Caching off DB
- Moving state off tiers that auto scale

Users >1 million



Users >1 million



The next big steps

Users >5 million–10 million

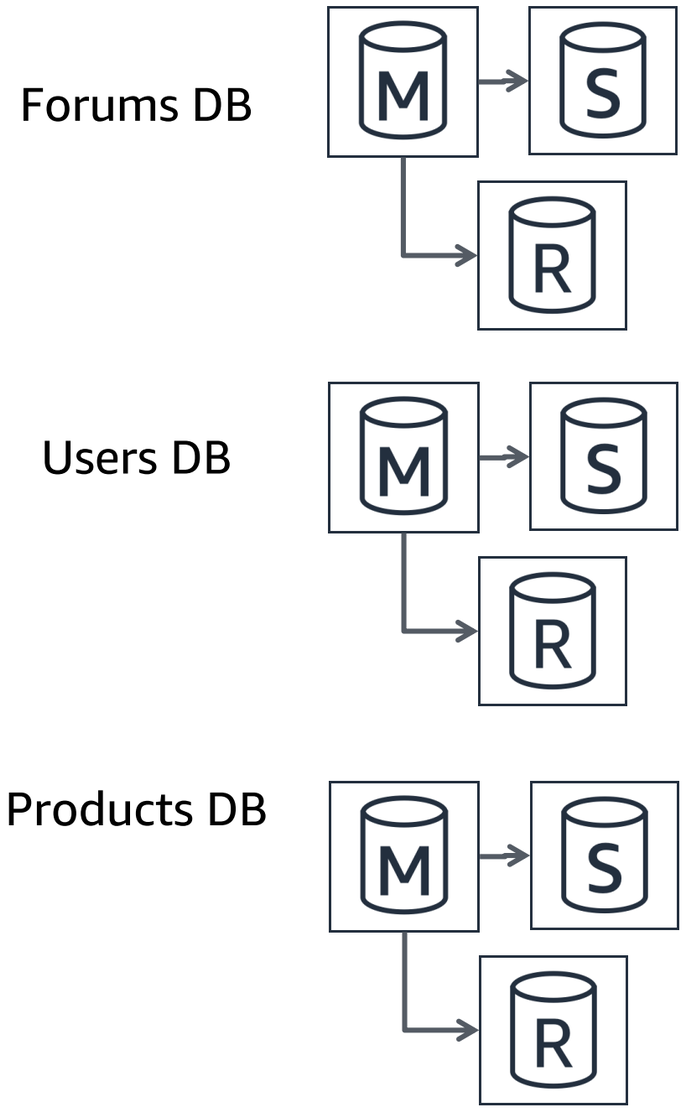
Database issues?

Solutions

- Federation—Splitting into multiple DBs based on function
- Sharding—Splitting one dataset up across multiple hosts
- Moving some functionality to other types of DBs (NoSQL, Graph)

Database federation

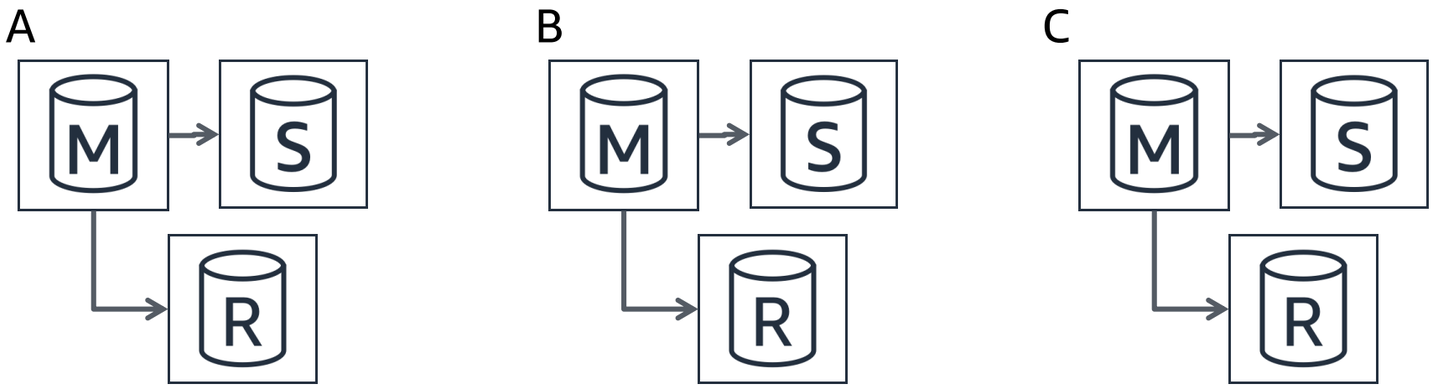
- Split up databases by function/purpose
- Harder to do cross-function queries
- Essentially delays sharding/NoSQL
- Won’t help with single huge functions/tables



Sharded horizontal scaling

- More complex at the application layer
- No practical limit on scalability
- Operation complexity/sophistication
- Shard by function or key space
- RDBMS or NoSQL

User	ShardID
002345	A
002346	B
002347	C
002348	B
002349	A



Shifting functionality to NoSQL

- Similar in a sense to federation
- NoSQL versus SQL
- Leverage managed services like DynamoDB

Some use cases

- Leaderboards/scoring
- Rapid ingest of clickstream/log data
- Temporary data needs (cart data)
- “Hot” tables
- Metadata/lookup tables

A quick review

A quick review

- Multi-AZ your infrastructure
- Make use of self-scaling services—Application Load Balancer, Amazon S3, AWS Lambda, Amazon SNS, Amazon SQS, AWS Step Functions, and others
- Build in redundancy at every level
- Start with SQL. Seriously
- Cache data both inside and outside your infrastructure
- Use automation tools in your infrastructure

A quick review continued

- Make sure you have good metrics/monitoring/logging
- Split tiers into individual services (SOA)
- Use Auto Scaling once you're ready for it
- Don't reinvent the wheel
- Move to NoSQL if and when it makes sense

10+ million users!

To infinity . . .

User >10 million

- More fine-tuning of your application
- More SOA of features/functionality
- Going from multi-AZ to multi-region
- Possibly start to build custom solutions
- Deep analysis of your entire stack
- Build serverless whenever possible

Related builder sessions

Wednesday, November 28

Scale up a Web Application

5:30 – 6:30 p.m. | Aria West, Level 3, Starvine 3, Table 4

Thursday, November 29

Scale up a Web Application

4:00 – 5:00 p.m. | Mirage, Grand Ballroom D, Table 6

Next steps?

Read!

aws.amazon.com/documentation

aws.amazon.com/architecture

aws.amazon.com/well-architected

aws.amazon.com/solutions

aws.amazon.com/quickstart

Start using AWS

aws.amazon.com/free

Next steps?

forums.aws.amazon.com

aws.amazon.com/answers

aws.amazon.com/premiumsupport

A solutions architect

Thank you!

Ben Thurgood
btgood@amazon.com

AWS re:Invent

aws

Please complete the session survey in the mobile app.

AWS re:Invent

aws