AWS re:Invent

ARC211-R

Scaling up to your first 10 million users

Brian Farnhill
Solutions Architect
Amazon Web Services

Hong Pham
Solutions Architect
Amazon Web Services

AWS re:Invent

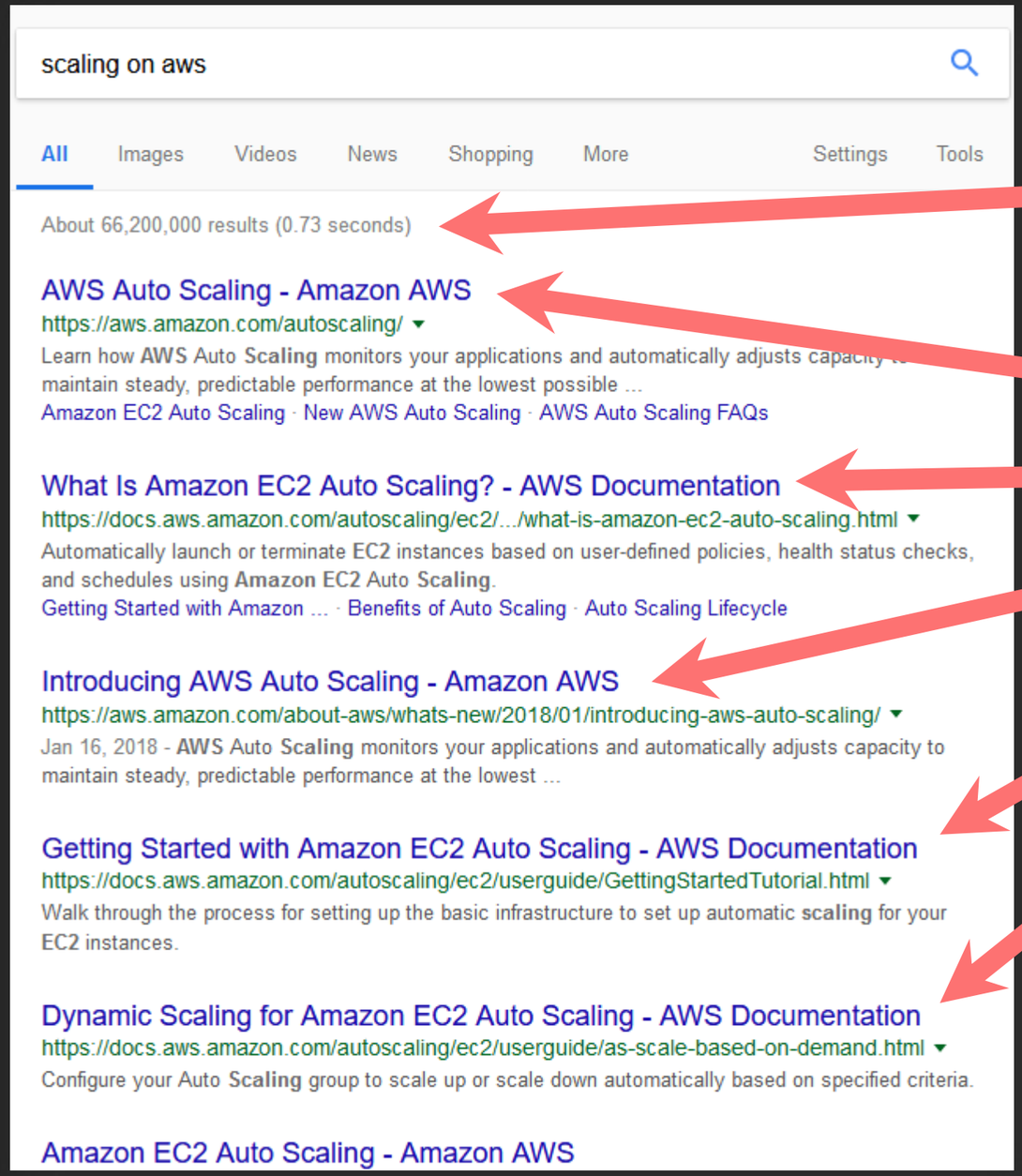
© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws





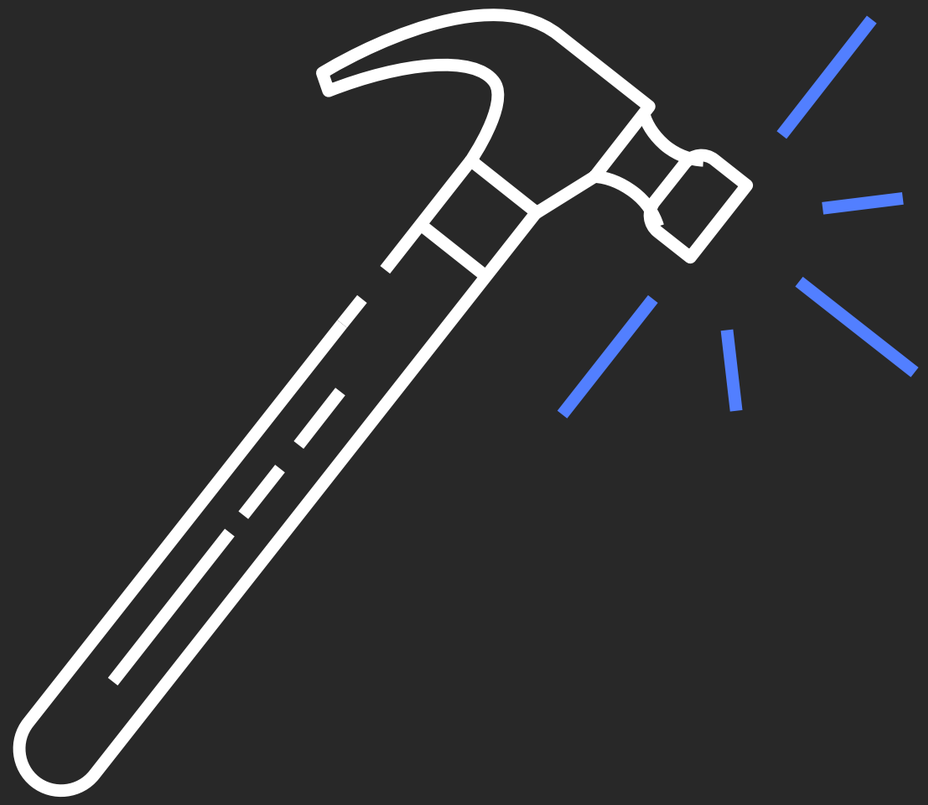




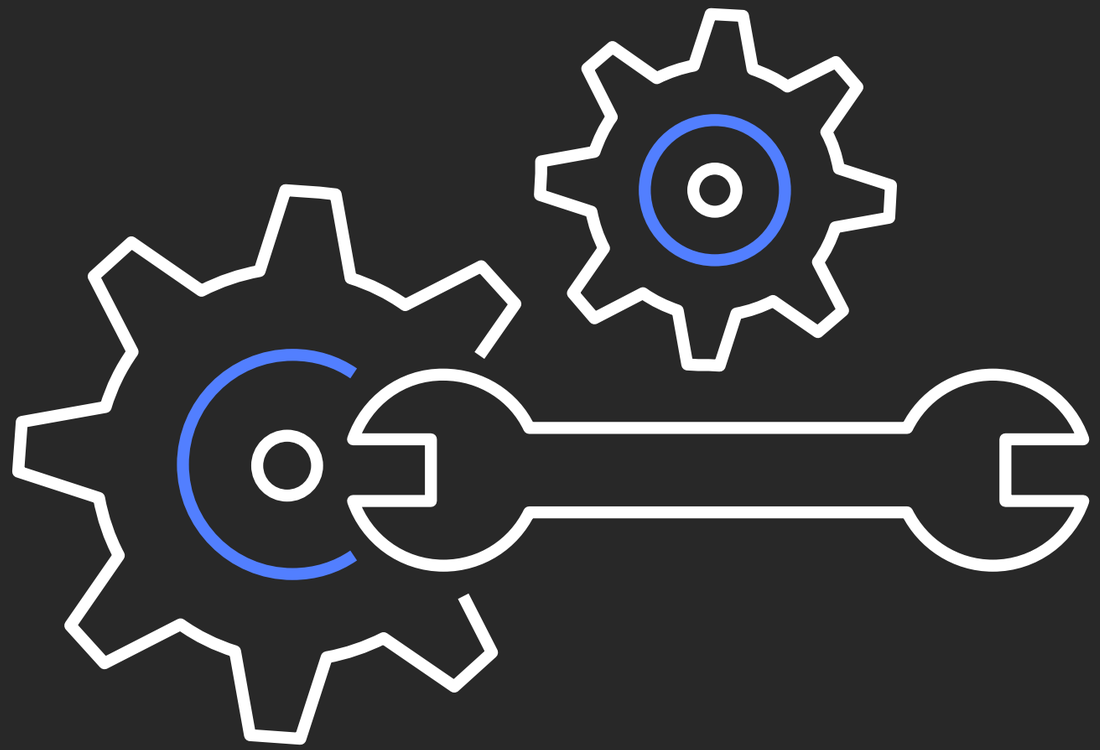
Now that's a lot of things to read!

This is *not* where we want to start!

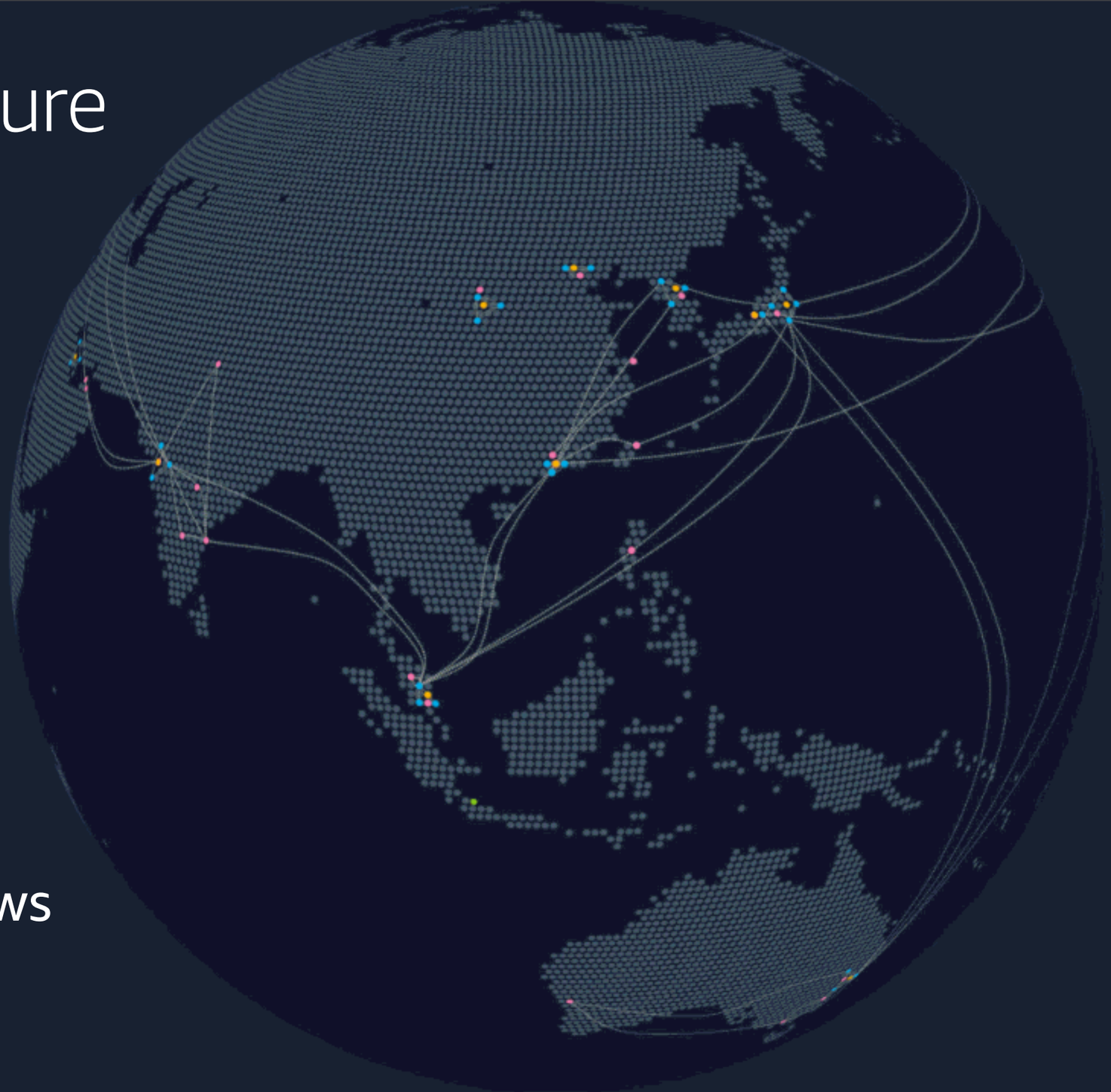
It’s not the single thing that fixes everything



What do we need first?



AWS Global Infrastructure



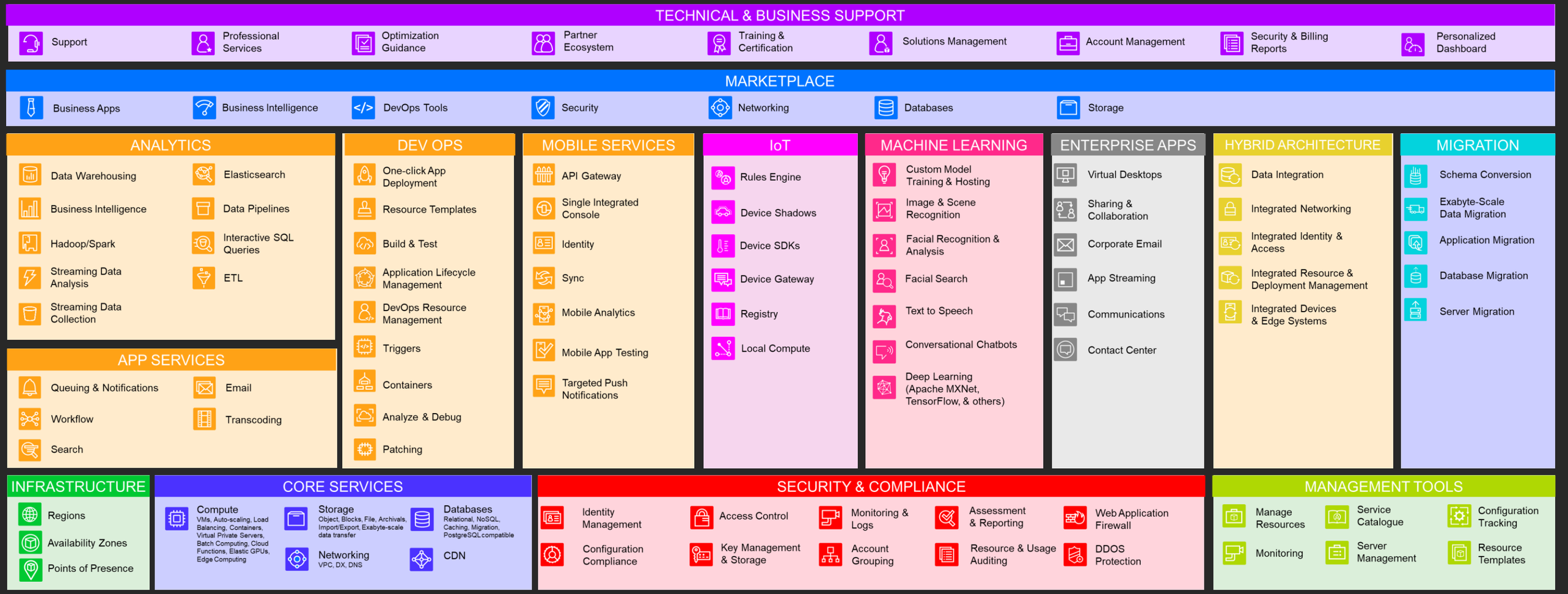
22 geographic Regions

69 Availability Zones

187 points of presence

See more at
https://www.infrastructure.aws

Robust, full-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)

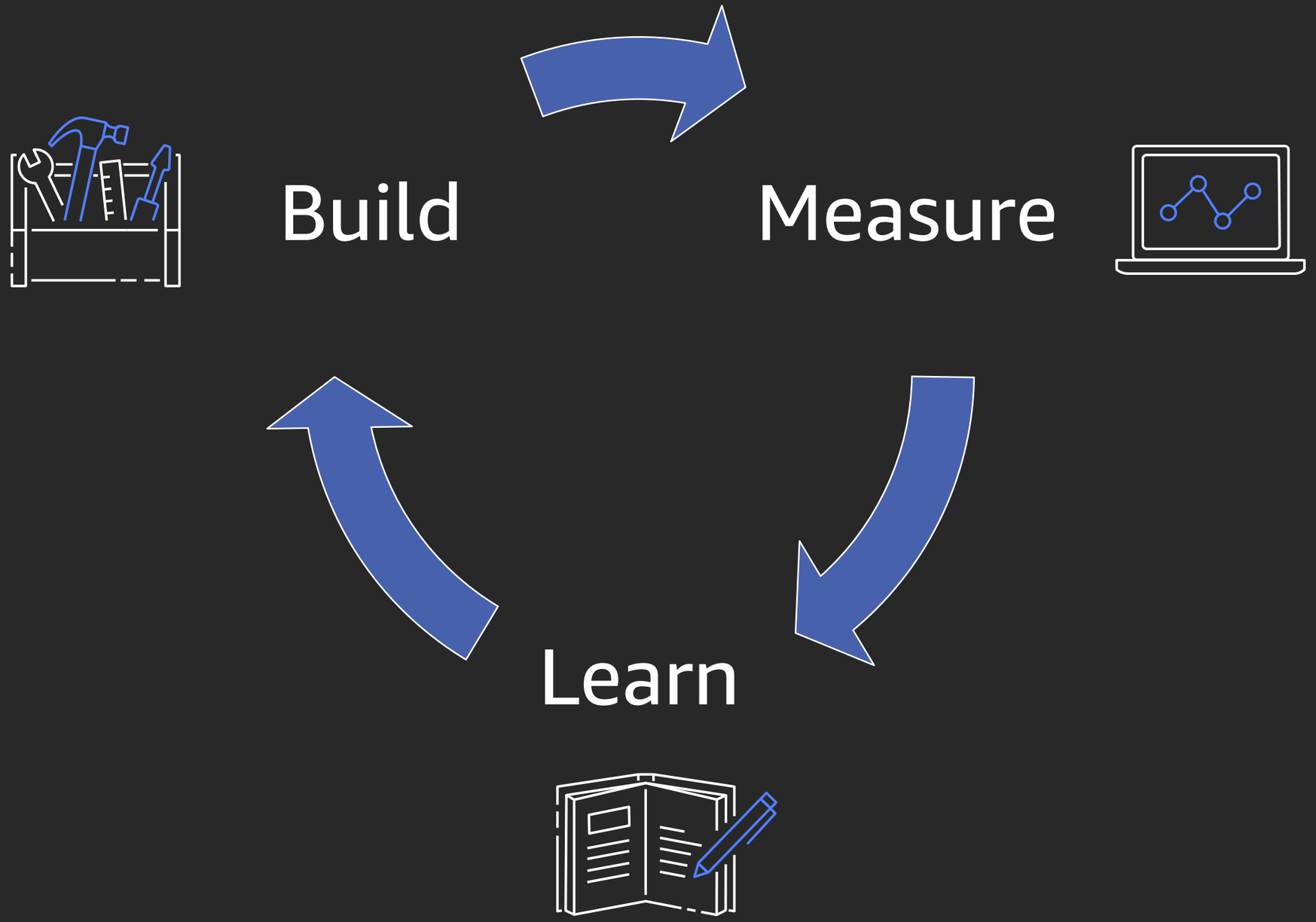
Considerations



"Many decisions are reversible, two-way doors."

Jeff Bezos

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.


Build
Measure
Learn

Guiding tenets

Identify and avoid undifferentiated heavy lifting

Serverless versus managed versus run it yourself



“Yak (Bos grunniens) at Nam Tso Lake, Tibet Autonomous Region,” Creative Commons, license CC BY-SA 3.0, https://commons.wikimedia.org/wiki/Category:Yaks_of_Tibet#/media/File:Yak_at_Nam_Tso_Tibet.jpg

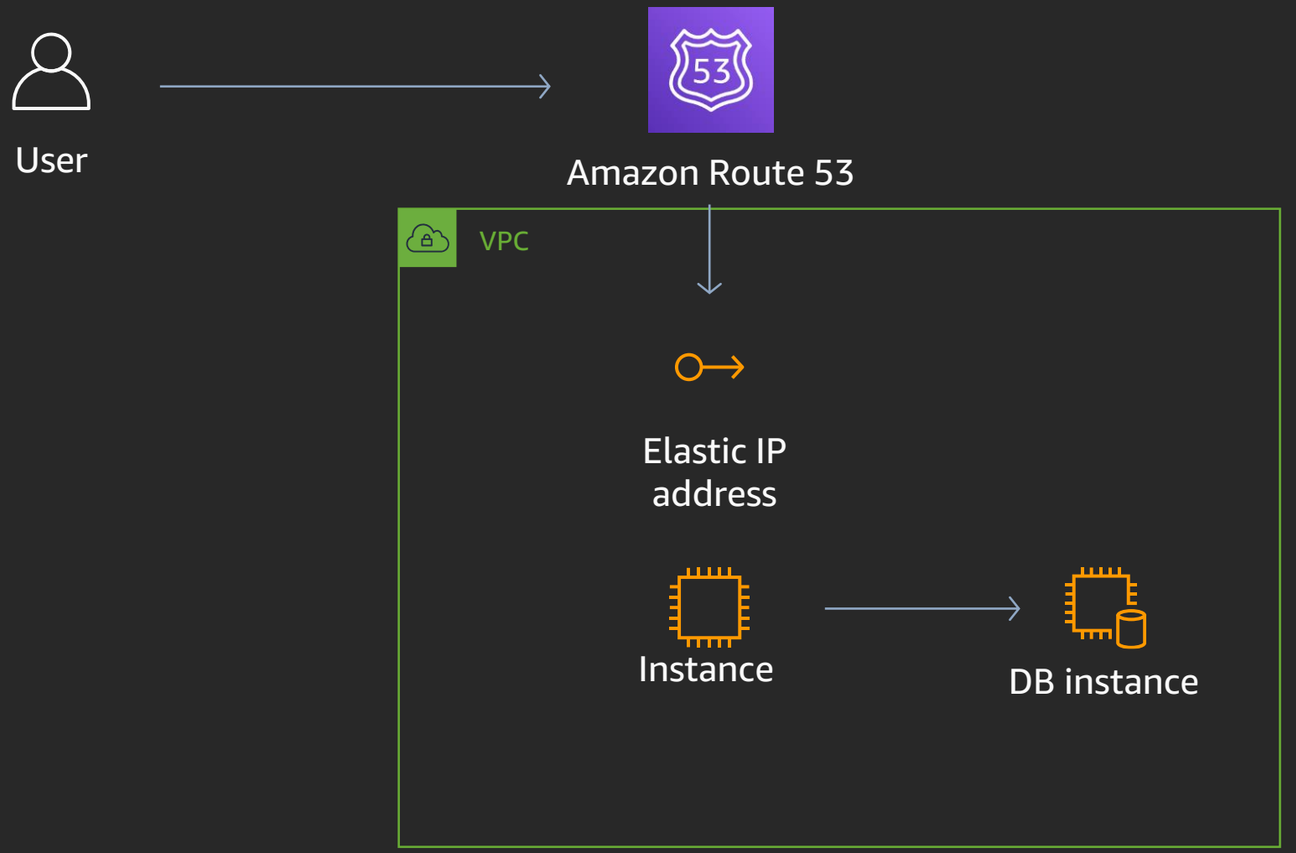
So let’s start from

Day 1



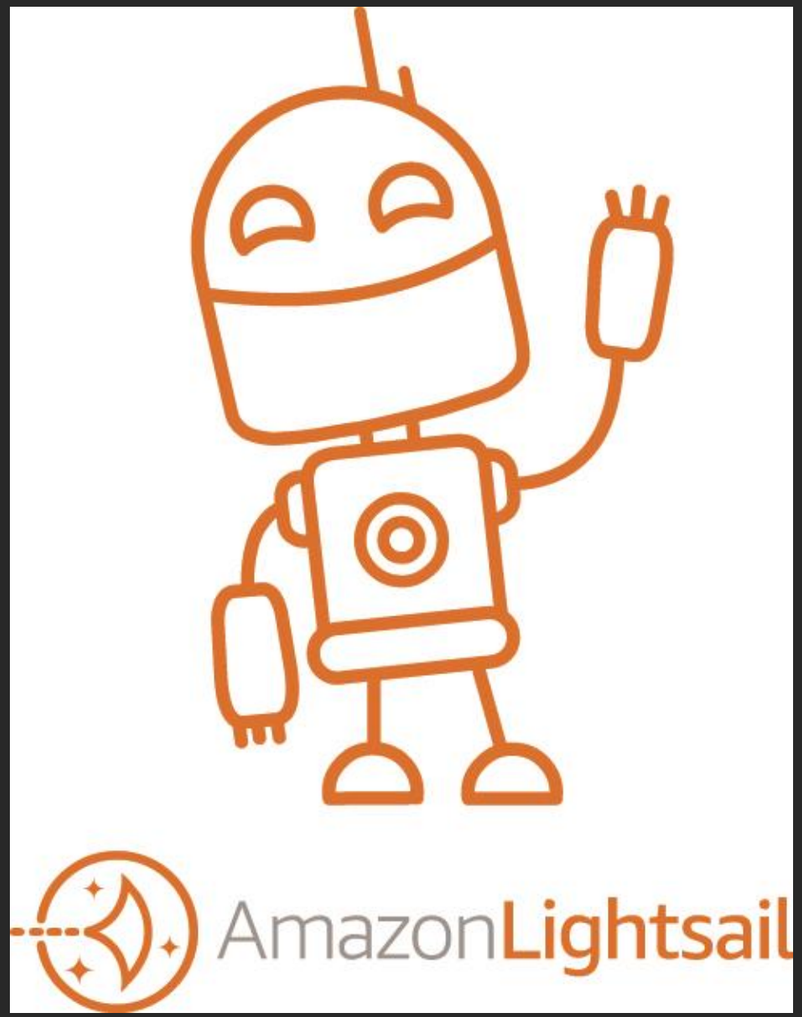
Users: >1

Users: >1


User
53
Amazon Route 53
VPC
Elastic IP address
Instance
DB instance

Amazon Lightsail: The easiest way to get started on AWS

- Choose from multiple 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)



Database options

Self-managed



Amazon EC2

Fully managed



Amazon RDS



Amazon DynamoDB



Amazon Neptune

Amazon Aurora



Amazon Aurora

- MySQL or PostgreSQL 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

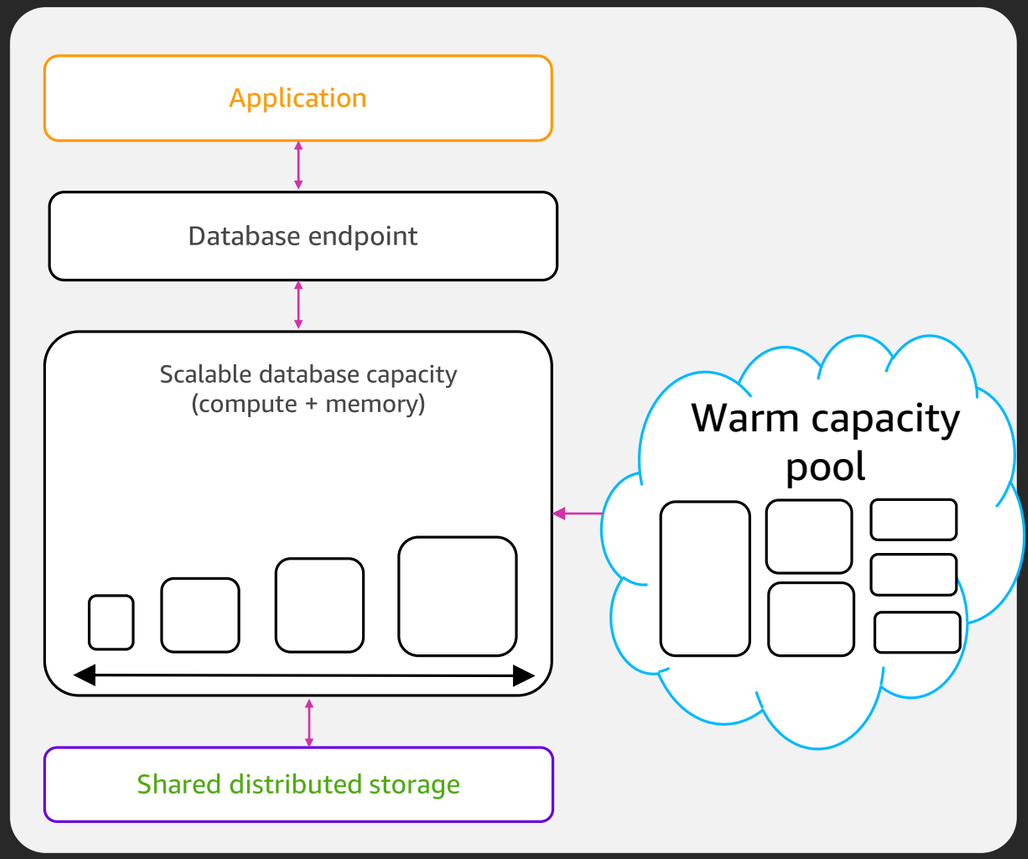


Amazon Aurora

- MySQL or PostgreSQL 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

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why start with SQL?

- Established and well-known technology
- Lots of existing code, communities, books, and tools
- You aren’t going to break SQL databases with 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.”

More than 5 TB in year 1?

Incredibly data-intensive workload?

OK!

You *might* need NoSQL

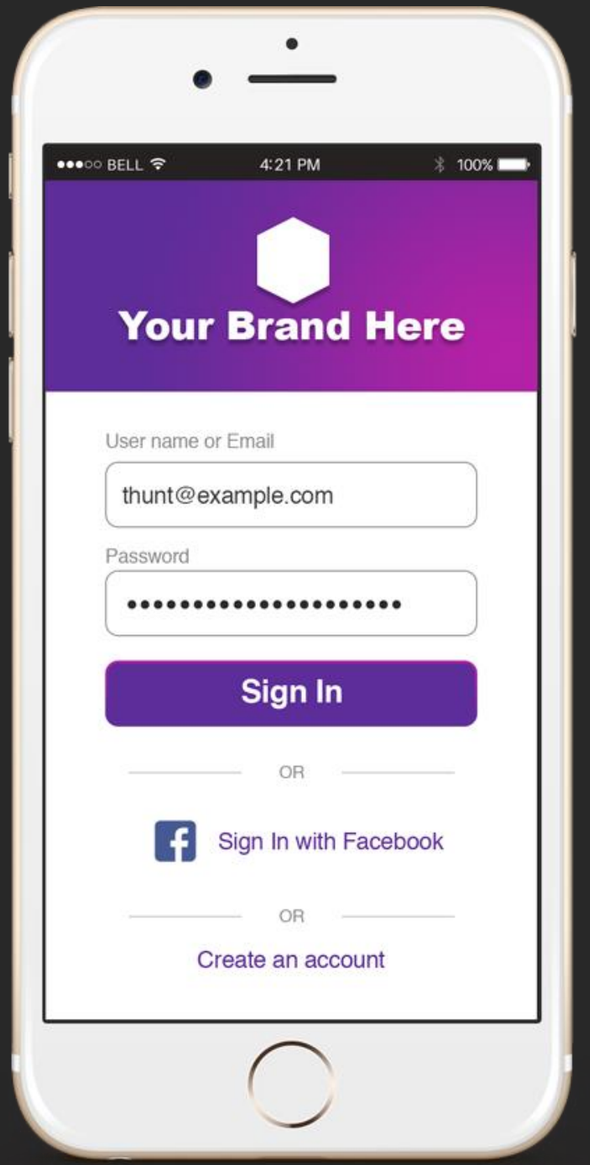
Why else might you need NoSQL?

- Super low-latency applications
- Metadata-driven data sets
- Highly nonrelational data
- Need schema-less data constructs*
- Rapid ingestion of data (thousands of records per second)
- Massive amounts of data (again, in the terabyte range)

*"Need" = "It's easier to do development without schemas"

Users: >1

Registration, sign in, and others



Amazon Cognito overview



Web and mobile apps

Developers focus on what is special about their app



Amazon Cognito

Amazon Cognito handles authorization and identity

Managed user directory



Hosted user interface



Federation

Google    SAML

AWS credentials



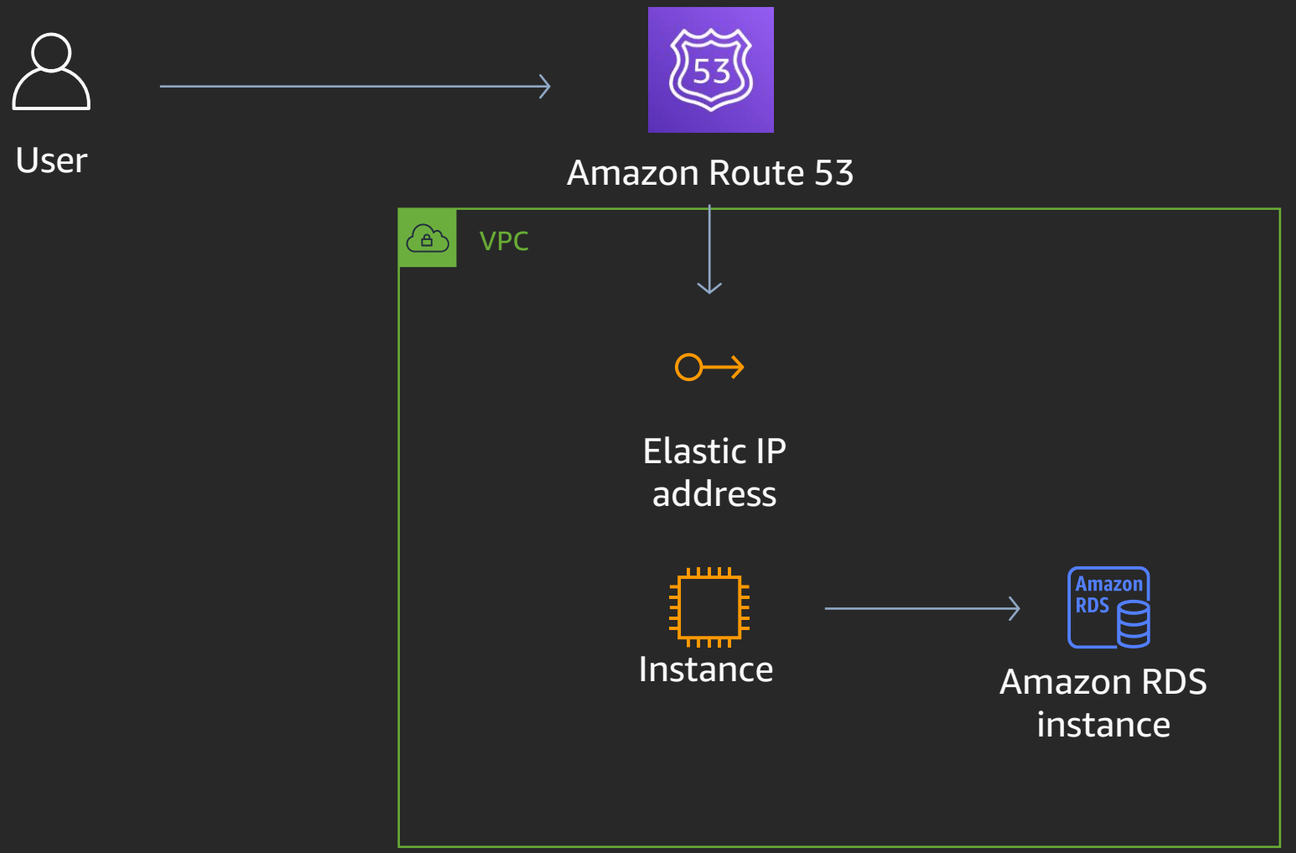
Standard tokens





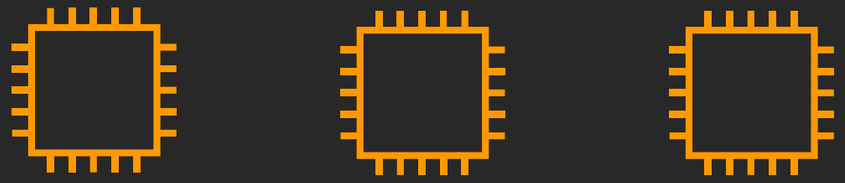
Users: >100

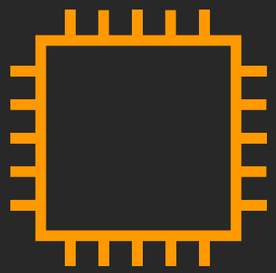
Users: >100

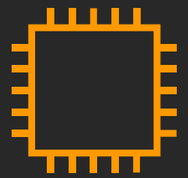

User
Amazon Route 53
VPC
Elastic IP address
Instance
Amazon RDS
Amazon RDS instance

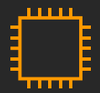
Users: >1000

Horizontally





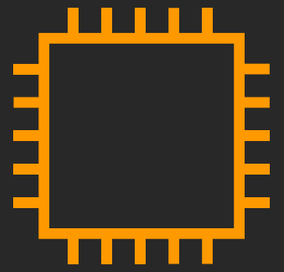




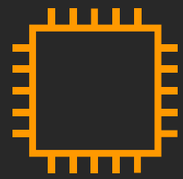
Vertically

“We’re gonna need a bigger box”

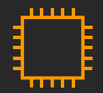
- Simplest approach
- Can now leverage PIOPS
- Many I/O instances
- Many memory instances
- Many CPU instances
- Many storage instances
- Easy to change instance sizes



c5.9xlarge



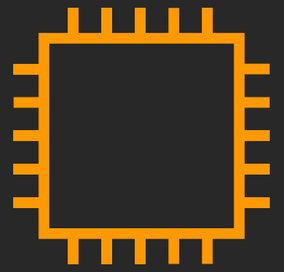
m5.2xlarge



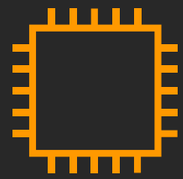
t3.nano

“We’re gonna need a bigger box”

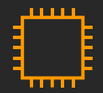
- Simplest approach
- Can now leverage PIOPS
- Many I/O instances
- Many memory instances
- Many CPU instances
- Many storage instances
- Easy to change instance sizes
- Will eventually hit an endpoint



c5.9xlarge

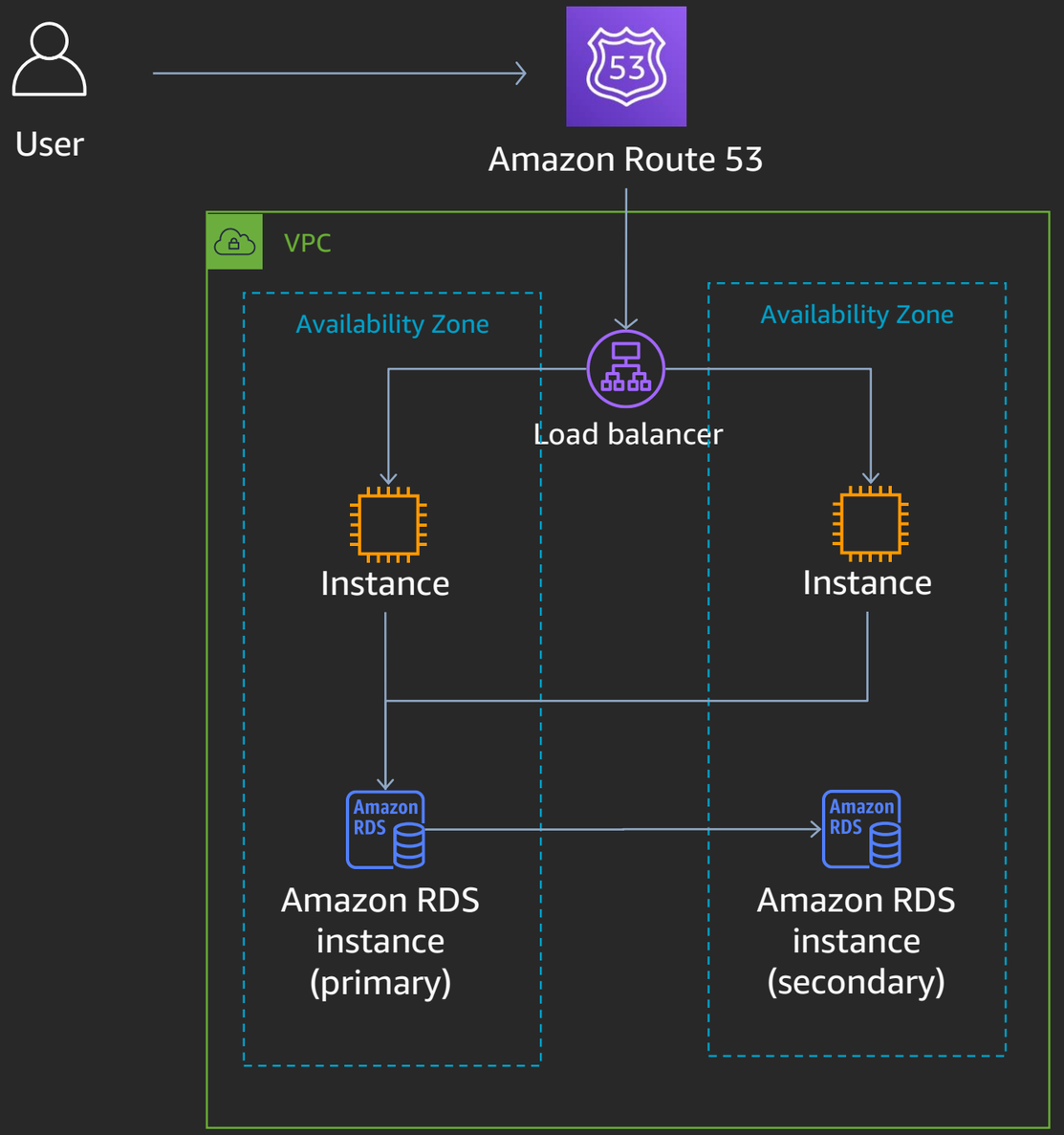


m5.2xlarge

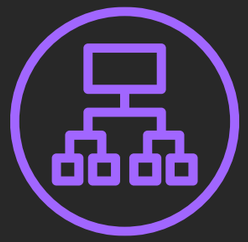


t3.nano

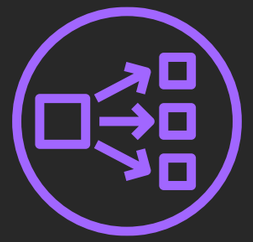
Users: >1000


User
Amazon Route 53
VPC
Availability Zone
Availability Zone
Load balancer
Instance
Instance
Amazon RDS
Amazon RDS
Amazon RDS instance (primary)
Amazon RDS instance (secondary)

Sharing the load



Application Load Balancer

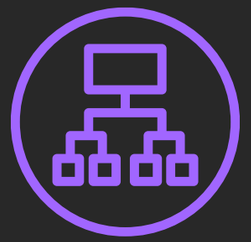


Network Load Balancer



Classic Load Balancer

Application Load Balancer



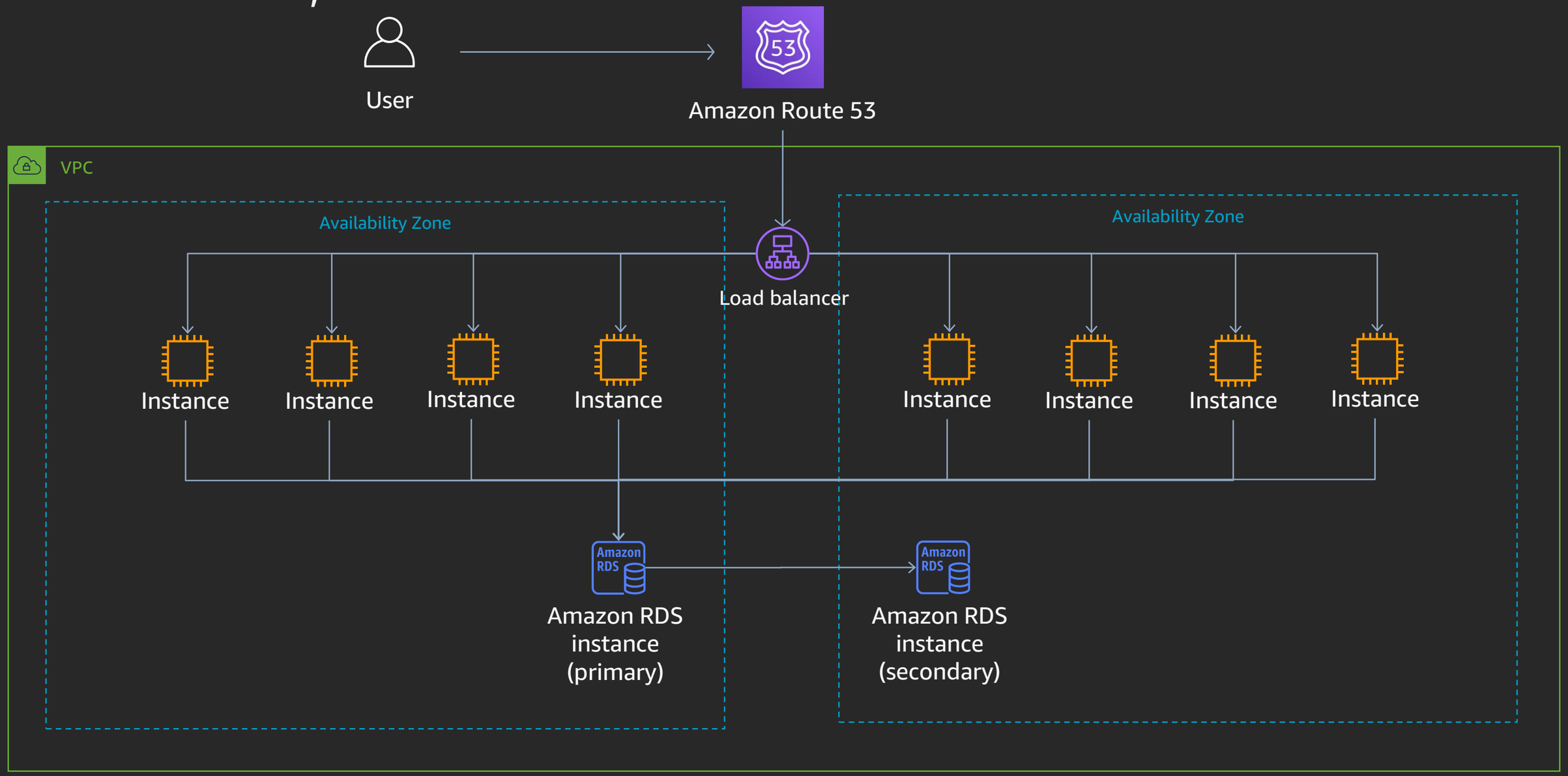
Application Load Balancer



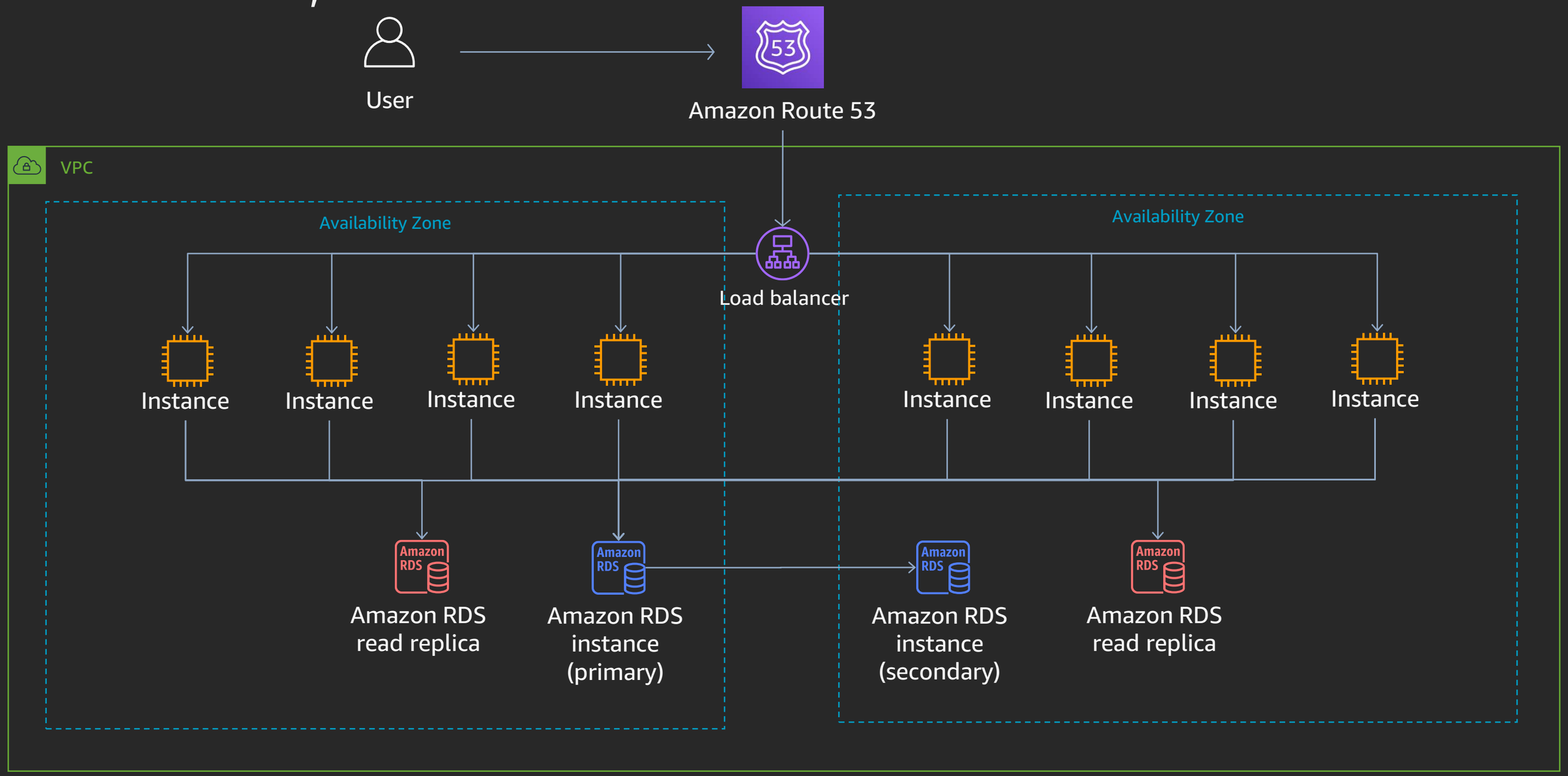
- Highly available
- 1–65,535
- Health checks
- Session stickiness
- Monitoring/logging
- Content-based routing
- Container-based apps
- WebSockets
- HTTP/2

Users: >10,000

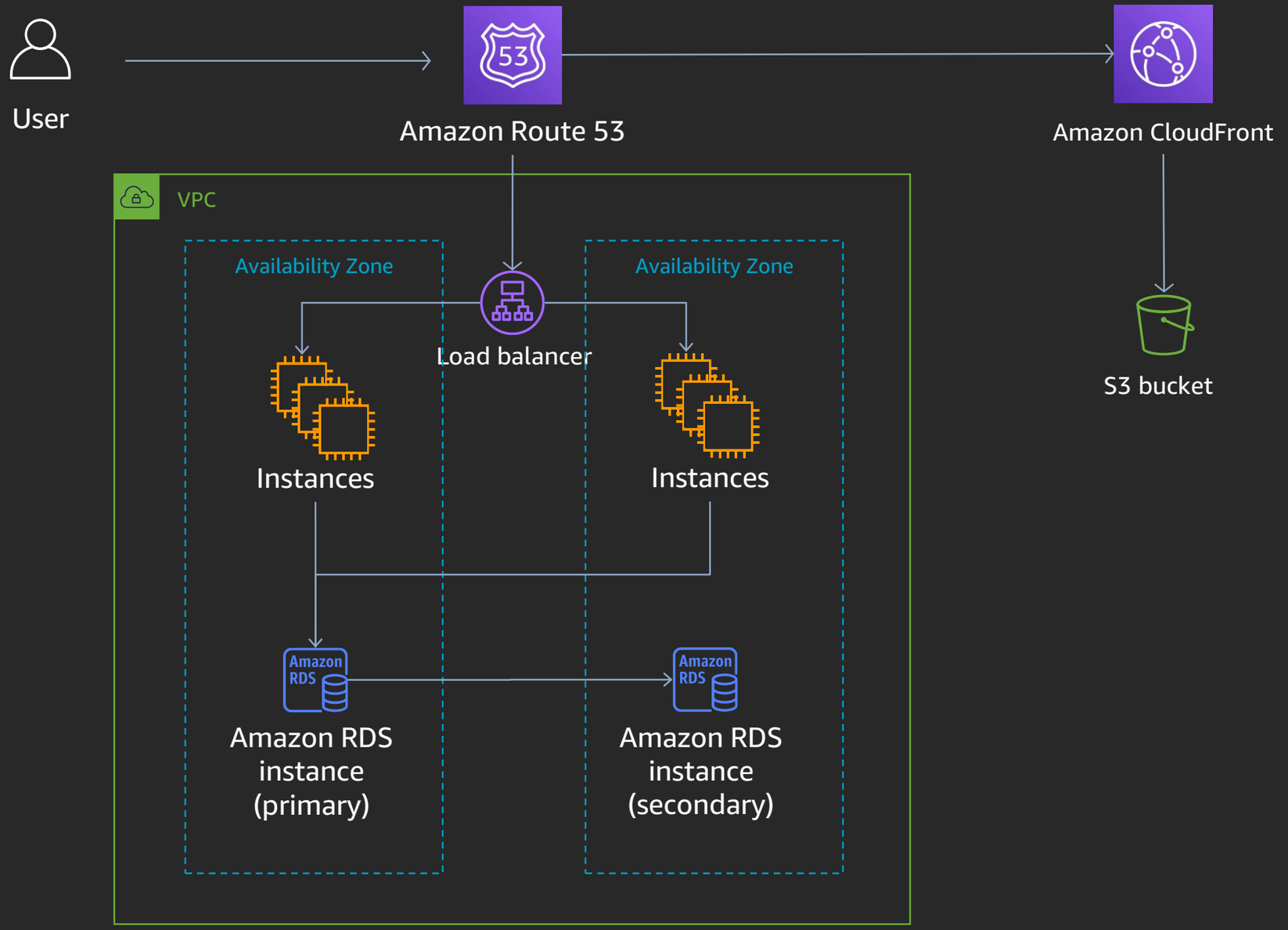
Users: >10,000


User
Amazon Route 53
VPC
Availability Zone
Availability Zone
Load balancer
Instance
Instance
Instance
Instance
Instance
Instance
Instance
Instance
Amazon RDS
Amazon RDS
Amazon RDS instance (primary)
Amazon RDS instance (secondary)

Users: >10,000


User
Amazon Route 53
VPC
Availability Zone
Availability Zone
Load balancer
Instance
Instance
Instance
Instance
Instance
Instance
Instance
Instance
Amazon RDS
Amazon RDS
Amazon RDS
Amazon RDS
Amazon RDS read replica
Amazon RDS instance (primary)
Amazon RDS instance (secondary)
Amazon RDS read replica

Shift some load around



Amazon S3



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



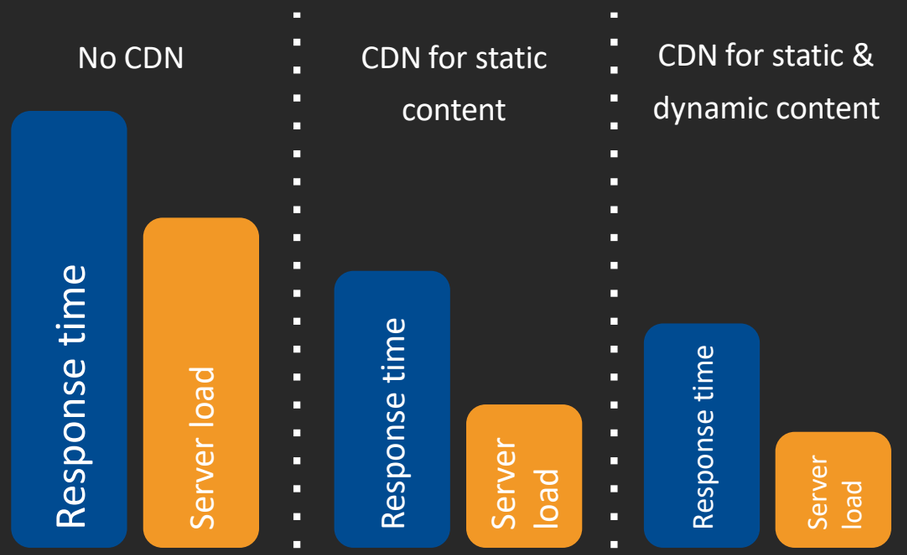
Amazon CloudFront

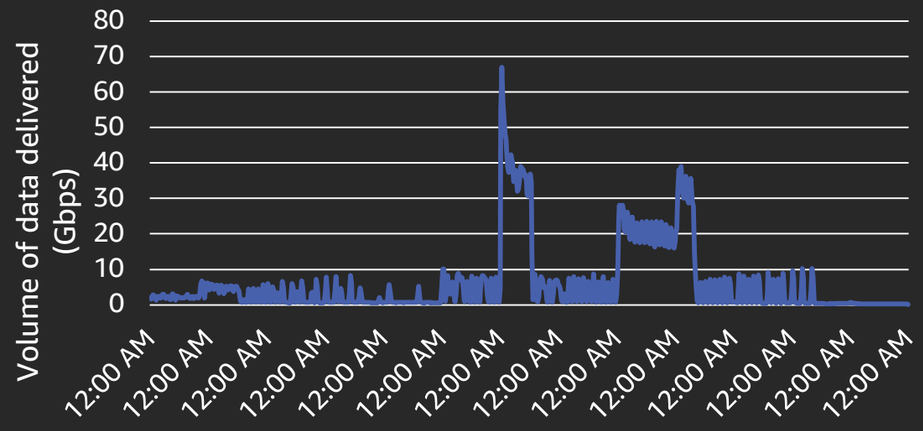
- Cache content for faster delivery
- Lower load upon origin
- Dynamic and static content
- Streaming video
- Custom SSL certificates
- Short time to live (TTL) (as little as 0 seconds)
- Optimized for AWS

Amazon CloudFront

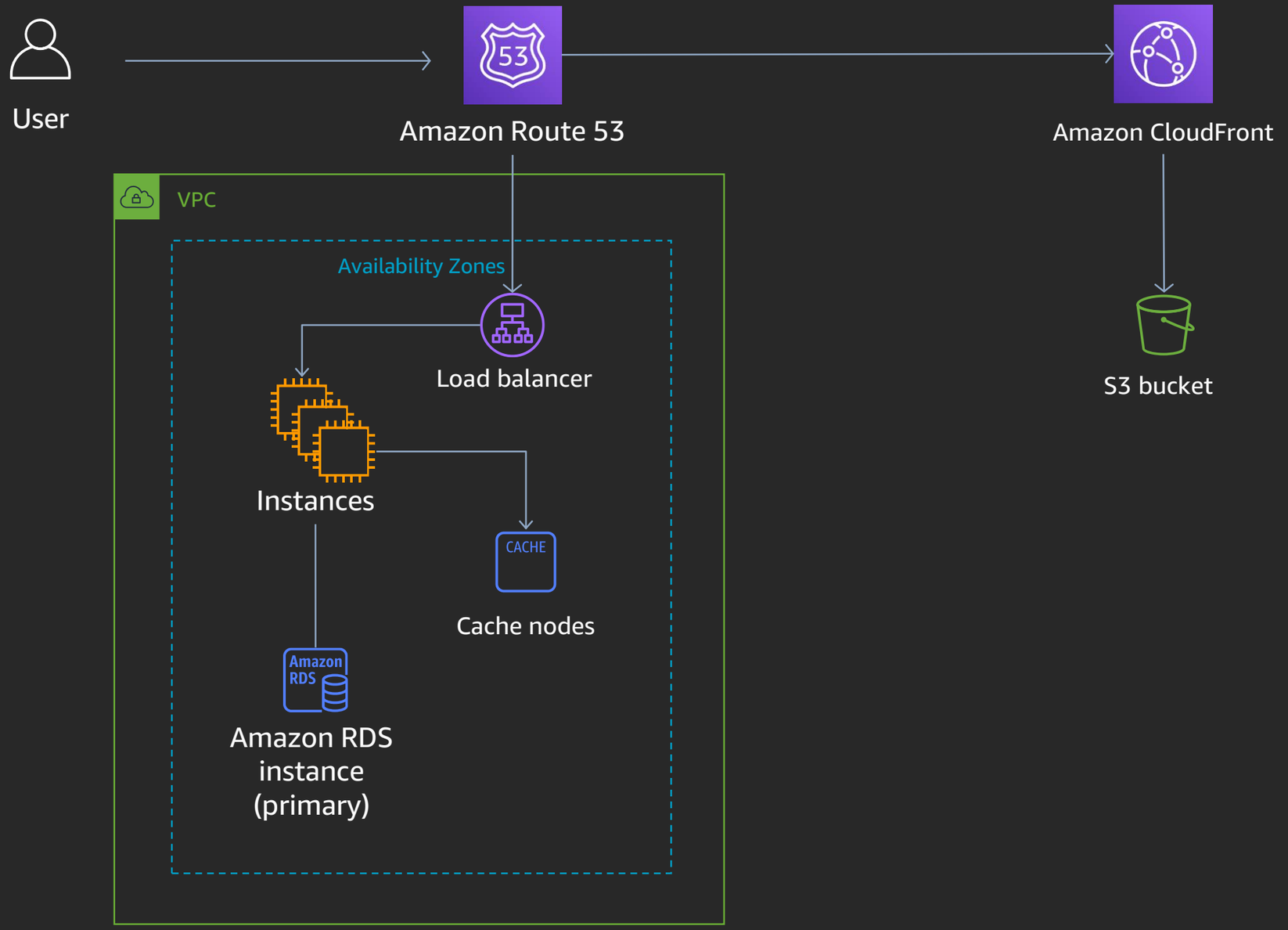


Amazon CloudFront





Shift some *more* load around



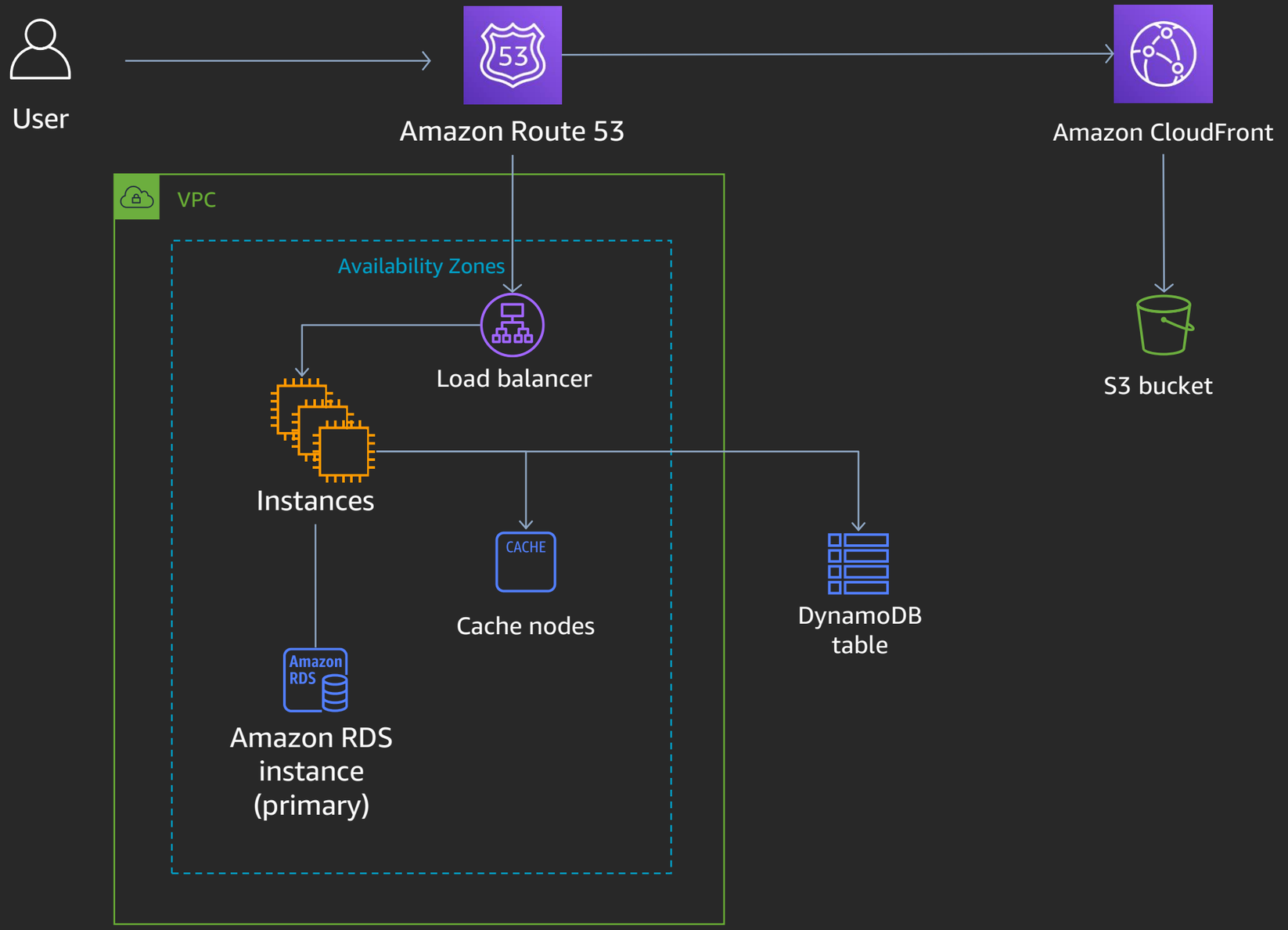
Amazon ElastiCache



Amazon ElastiCache

- Managed Memcached or Redis
- Scale from one to many nodes
- Self-healing (replaces dead instance)
- Single-digit millisecond speeds (usually)
- Multi-AZ deployments for availability

Shift *even more* load around



Amazon DynamoDB

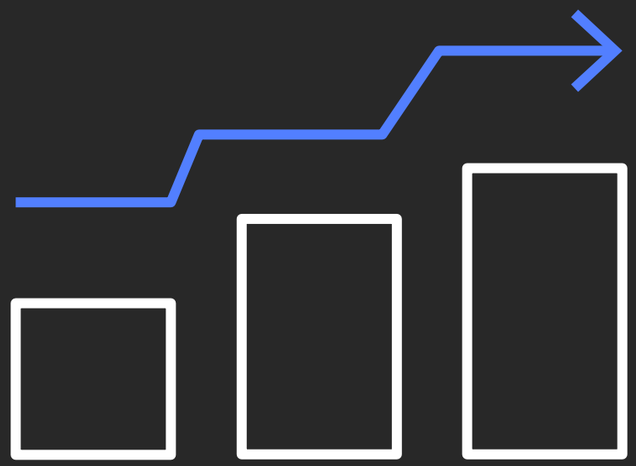


Amazon DynamoDB

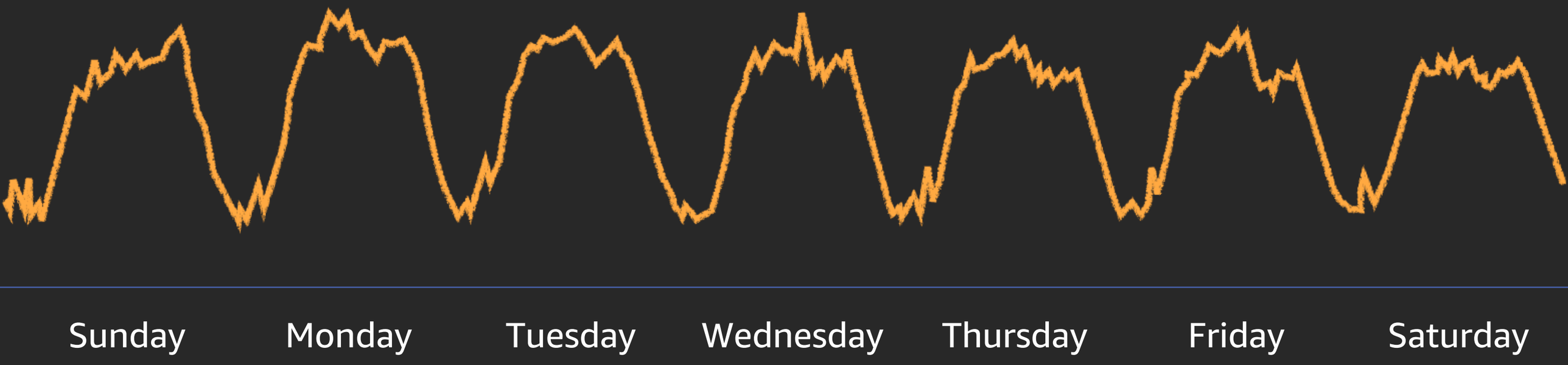
- Managed NoSQL database
- Provisioned throughput
- Fast, predictable performance
- Fully distributed, fault tolerant
- JSON support
- Items up to 400 KB
- TTL
- Streams and triggers
- AWS Application Auto Scaling
- Global tables

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

Auto Scaling

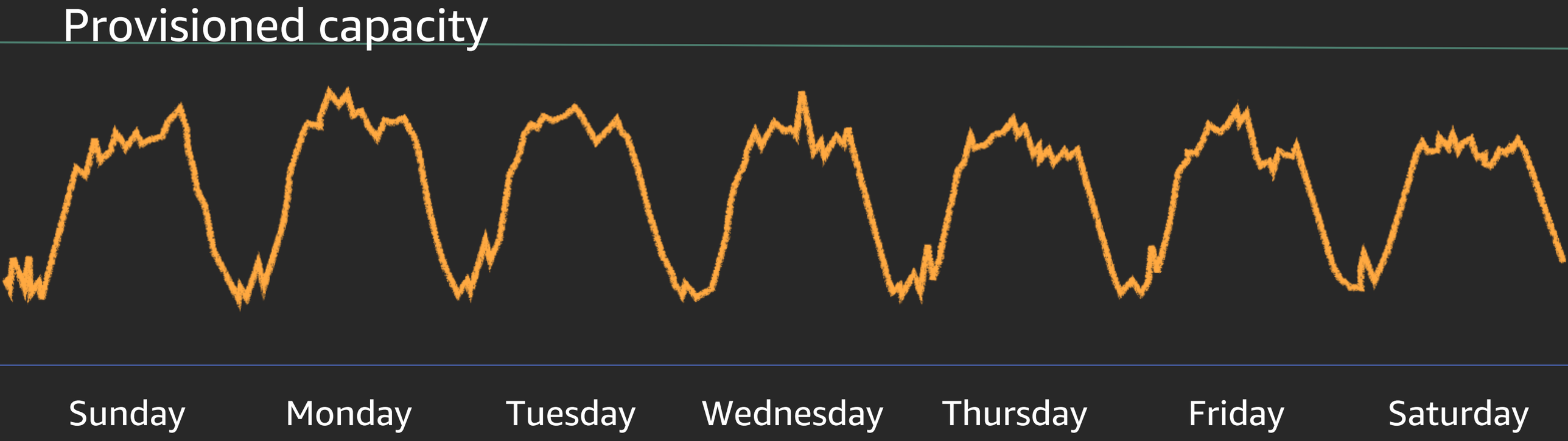


Typical weekly traffic to Amazon.com

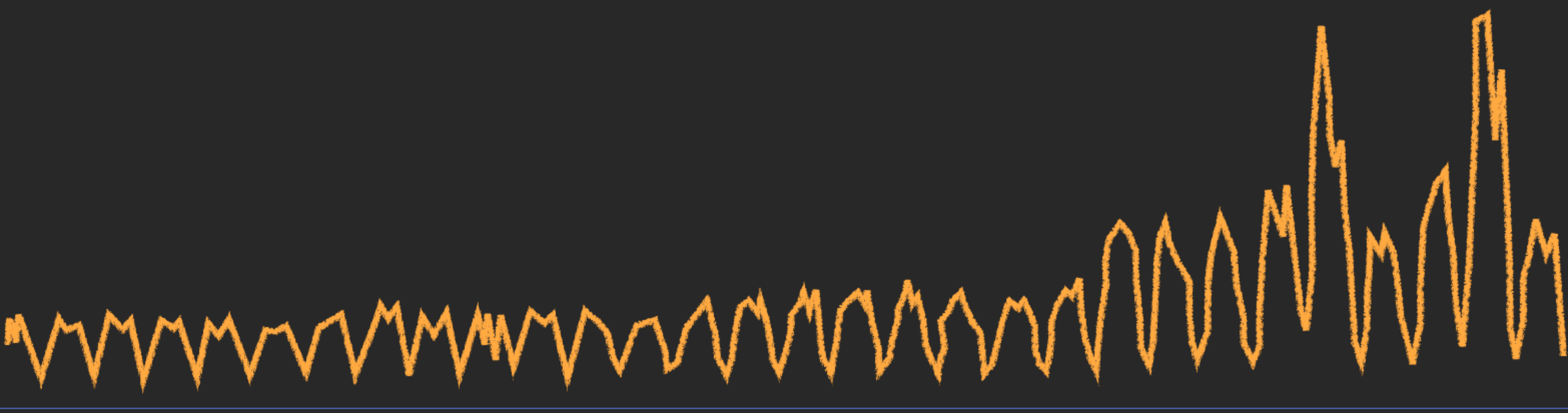


Sunday Monday Tuesday Wednesday Thursday Friday Saturday

Typical weekly traffic to Amazon.com

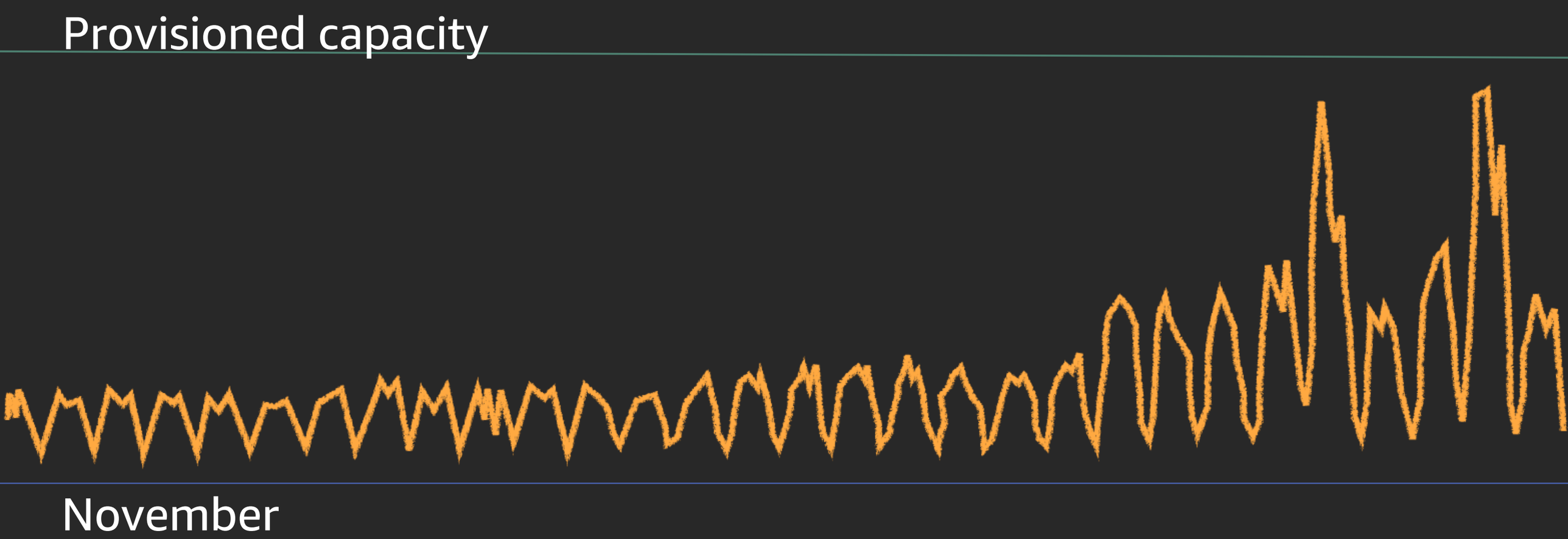


November traffic to Amazon.com

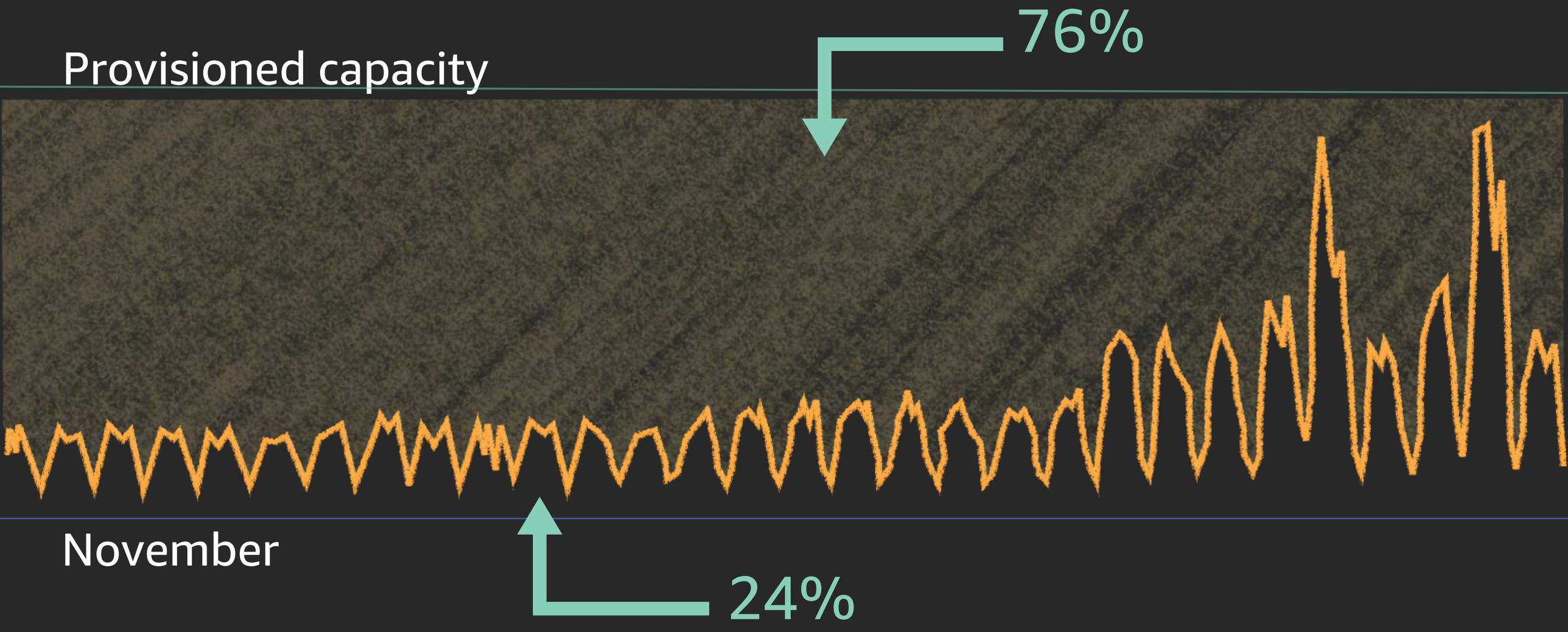


November

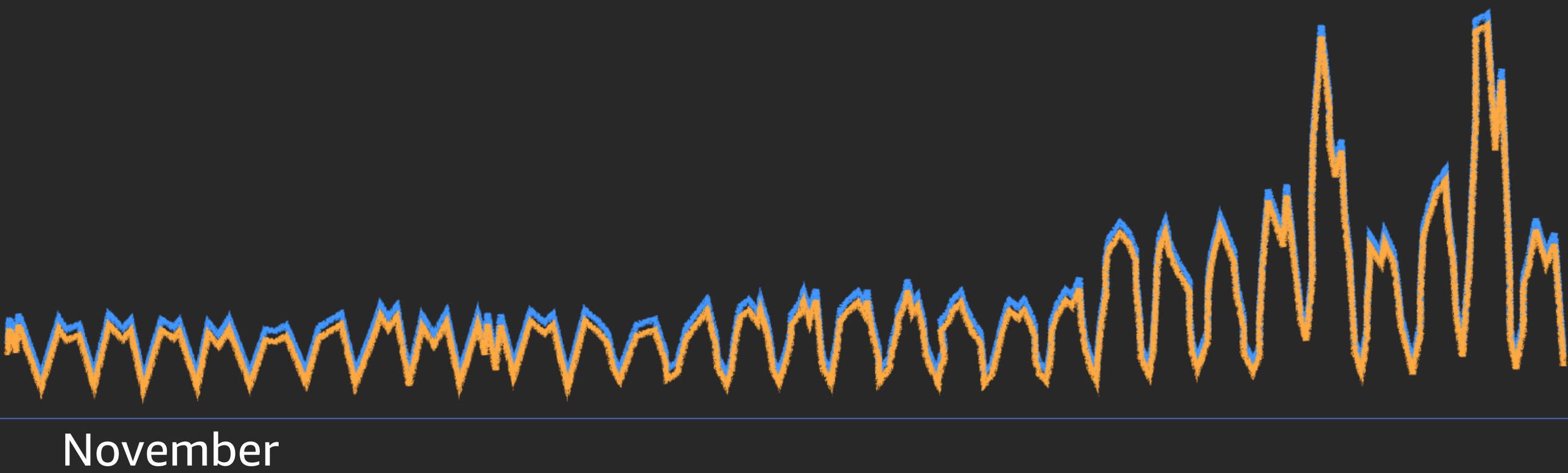
November traffic to Amazon.com



November traffic to Amazon.com

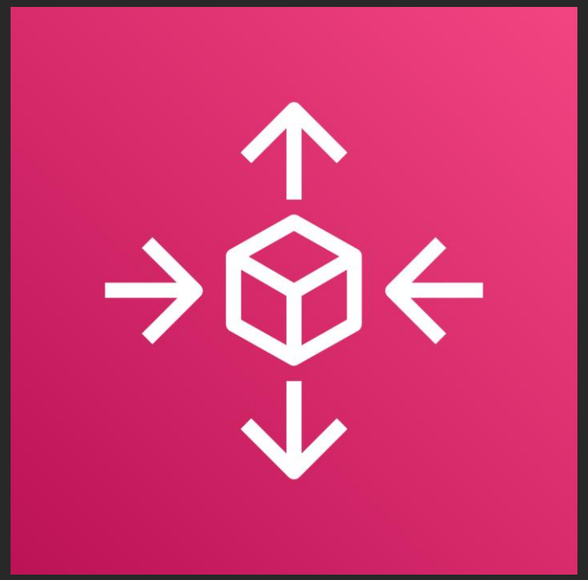


November traffic to Amazon.com



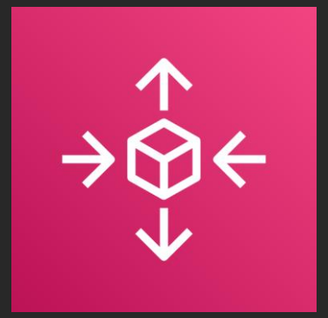
November

Auto Scaling lets you do this



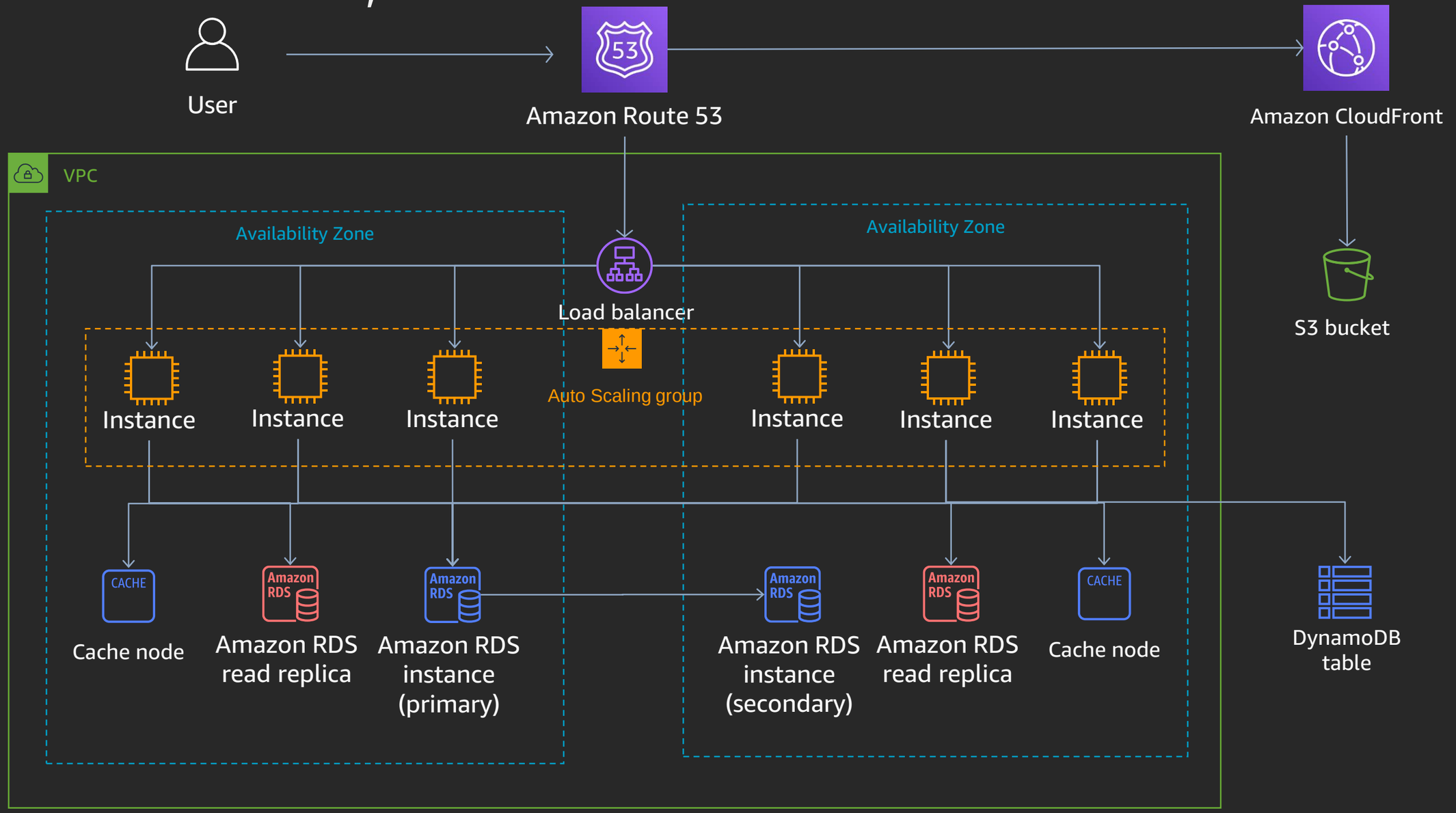
Auto Scaling

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

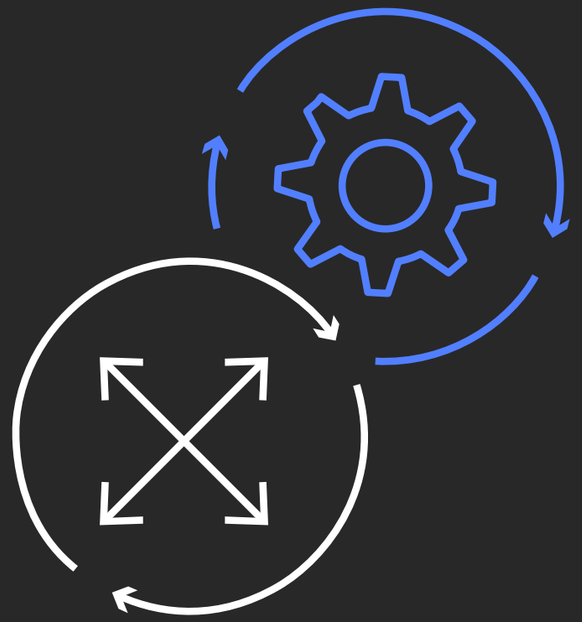


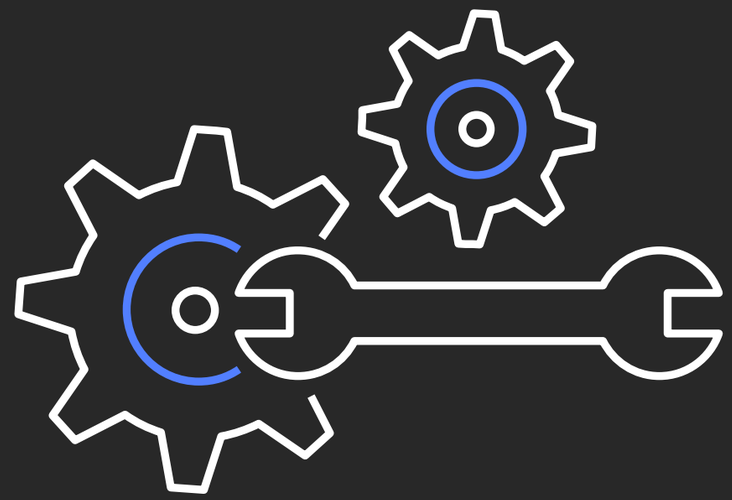
```
aws autoscaling create-auto-scaling-group
--auto-scaling-group-name MyGroup
--launch-configuration-name MyConfig
--min-size 4
--max-size 200
--availability-zones us-west-2c, us-west-2b
```

Users: >500,000



Use automation





AWS Systems Manager: Automate operational tasks



AWS Systems Manager

In the cloud and on premises

Managed remote access (no bastions)

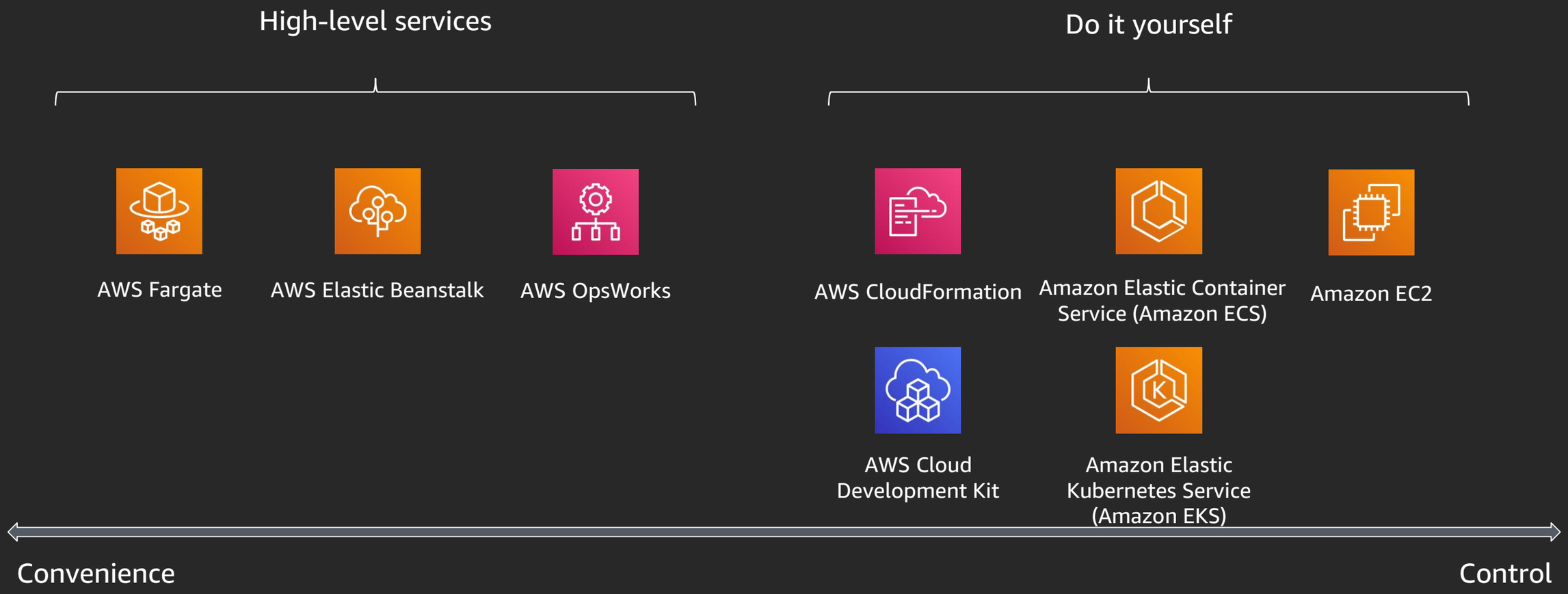
Automate common tasks

Compliance management and reporting

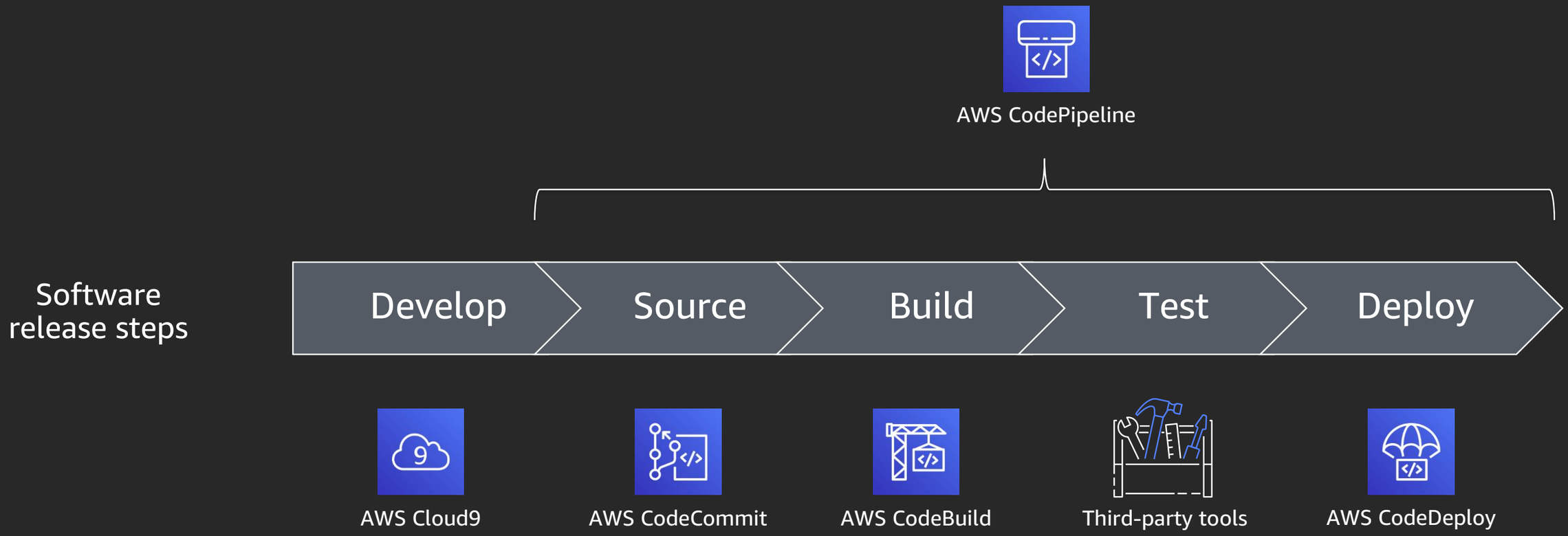
Basic and advanced parameter store

Reasonable price

AWS infrastructure automation



AWS code services



AWS CodeStar

Quickly develop, build, and deploy applications on AWS



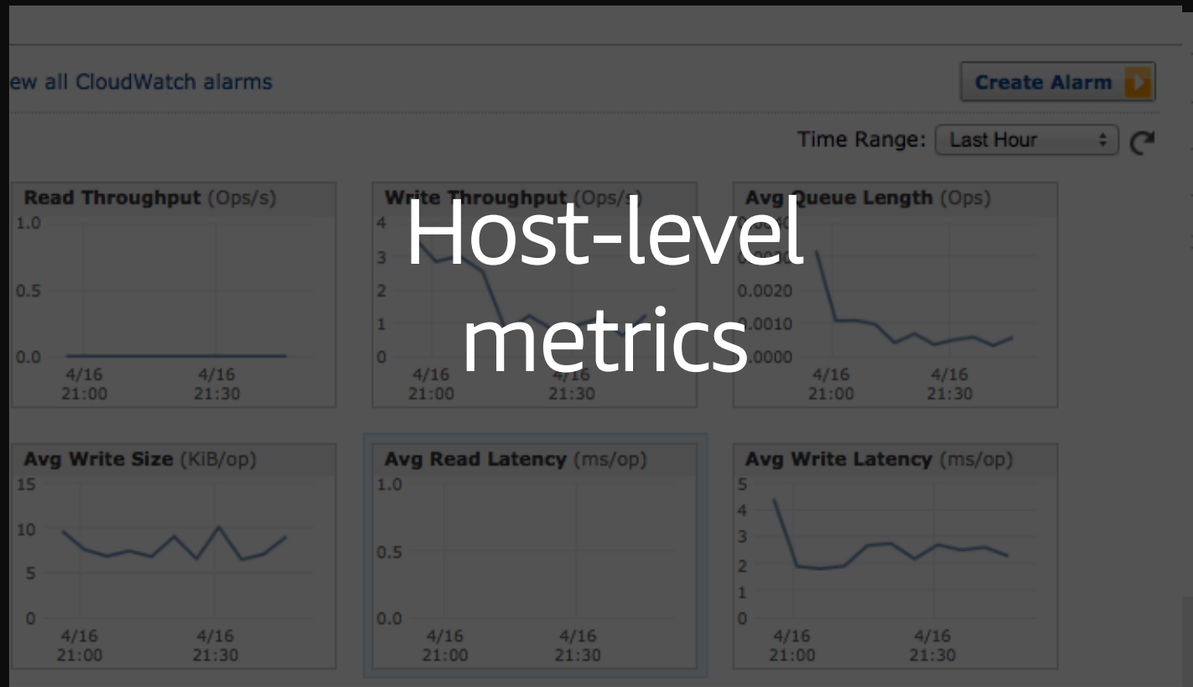
AWS CodeStar

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

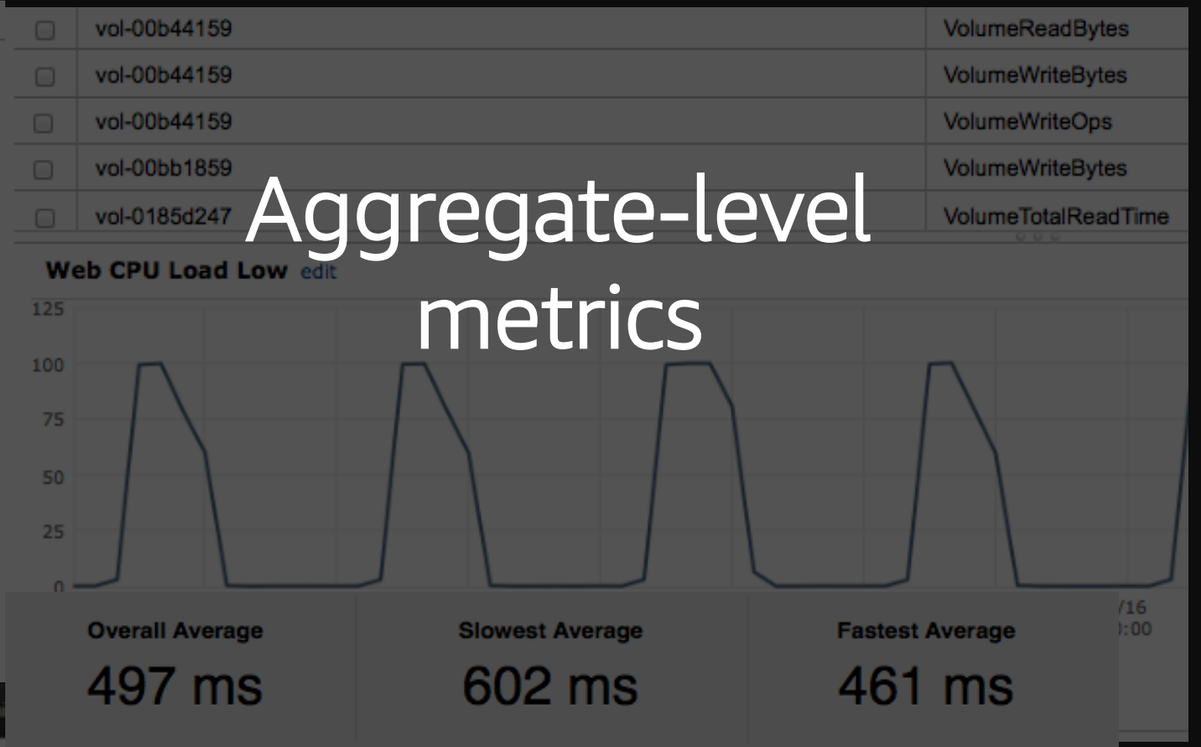
Users: >500,000



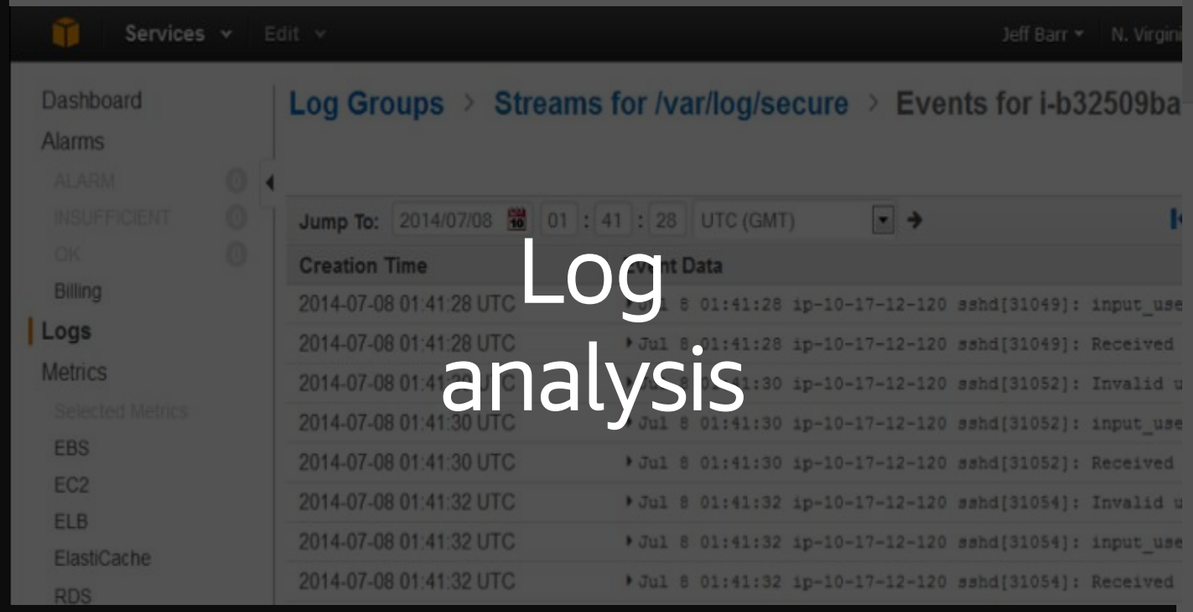
- Monitoring, metrics, and logging
  If you can’t build it internally, outsource it (third-party SaaS)
- What are customers saying?
- Try to squeeze as much performance as possible out of each service/component



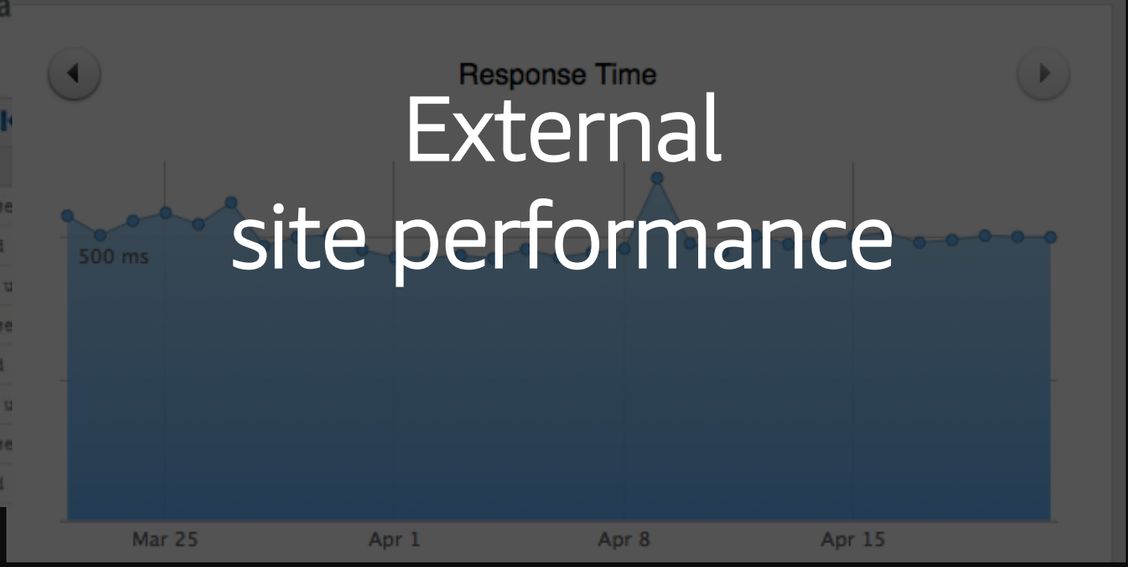
Host-level metrics



Aggregate-level metrics



Log analysis



External site performance

Amazon CloudWatch

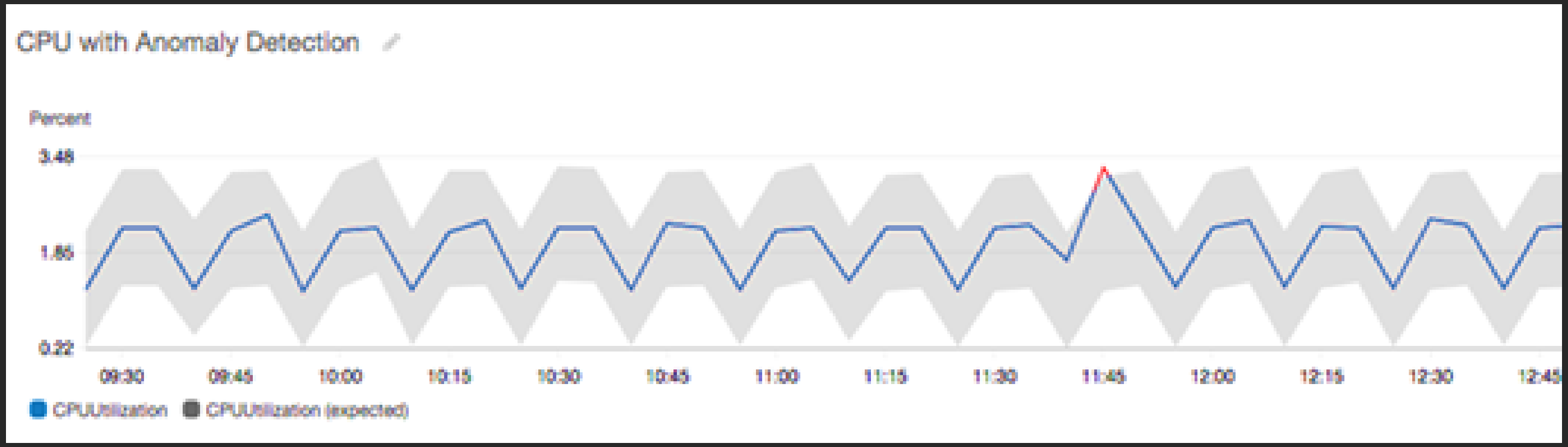


Amazon CloudWatch

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

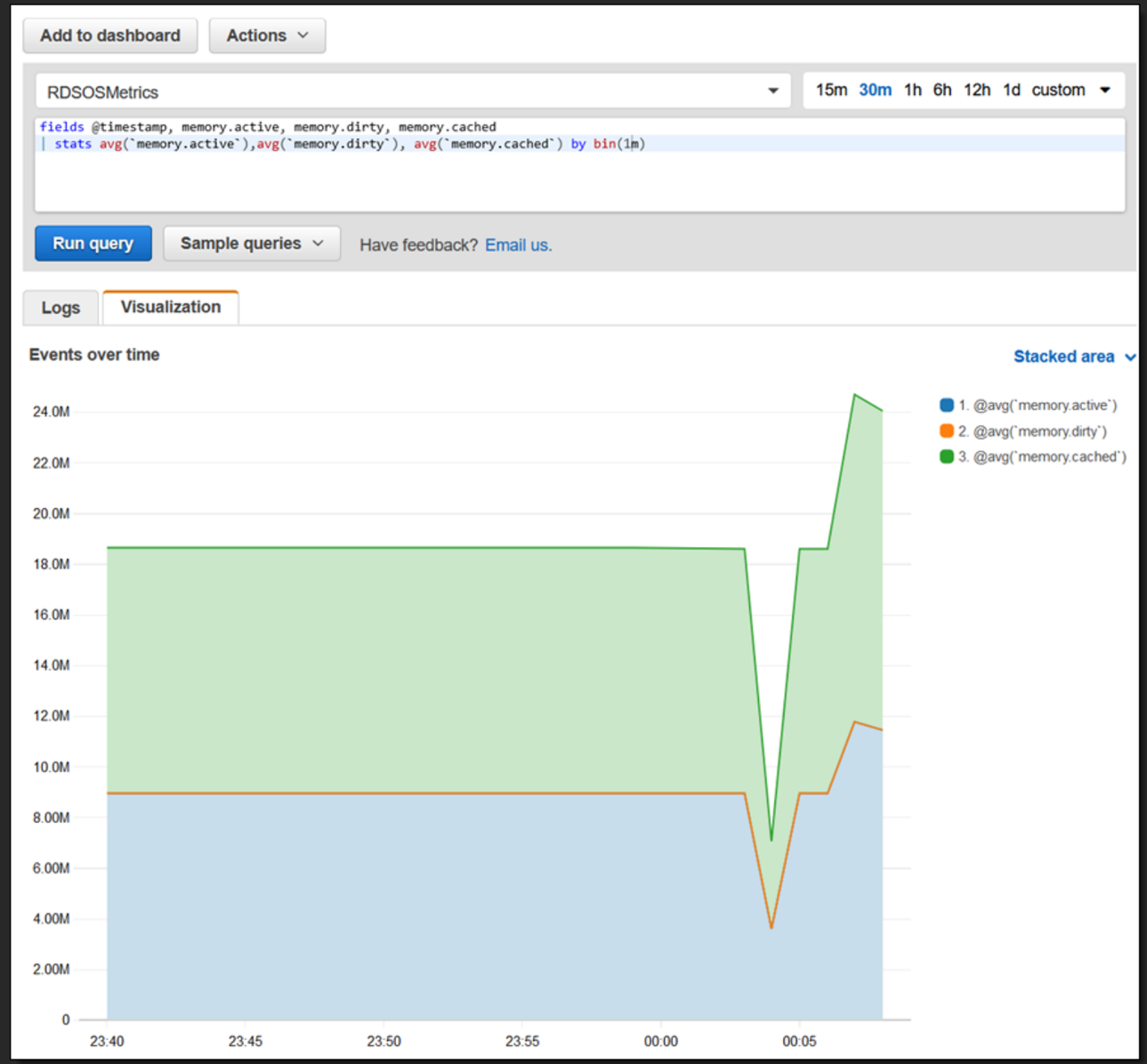
Amazon CloudWatch anomaly detection

Know what looks different without defining static thresholds



Amazon CloudWatch Logs insights

Query logs and create visualizations in order to better understand your data



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

Monolithic architecture



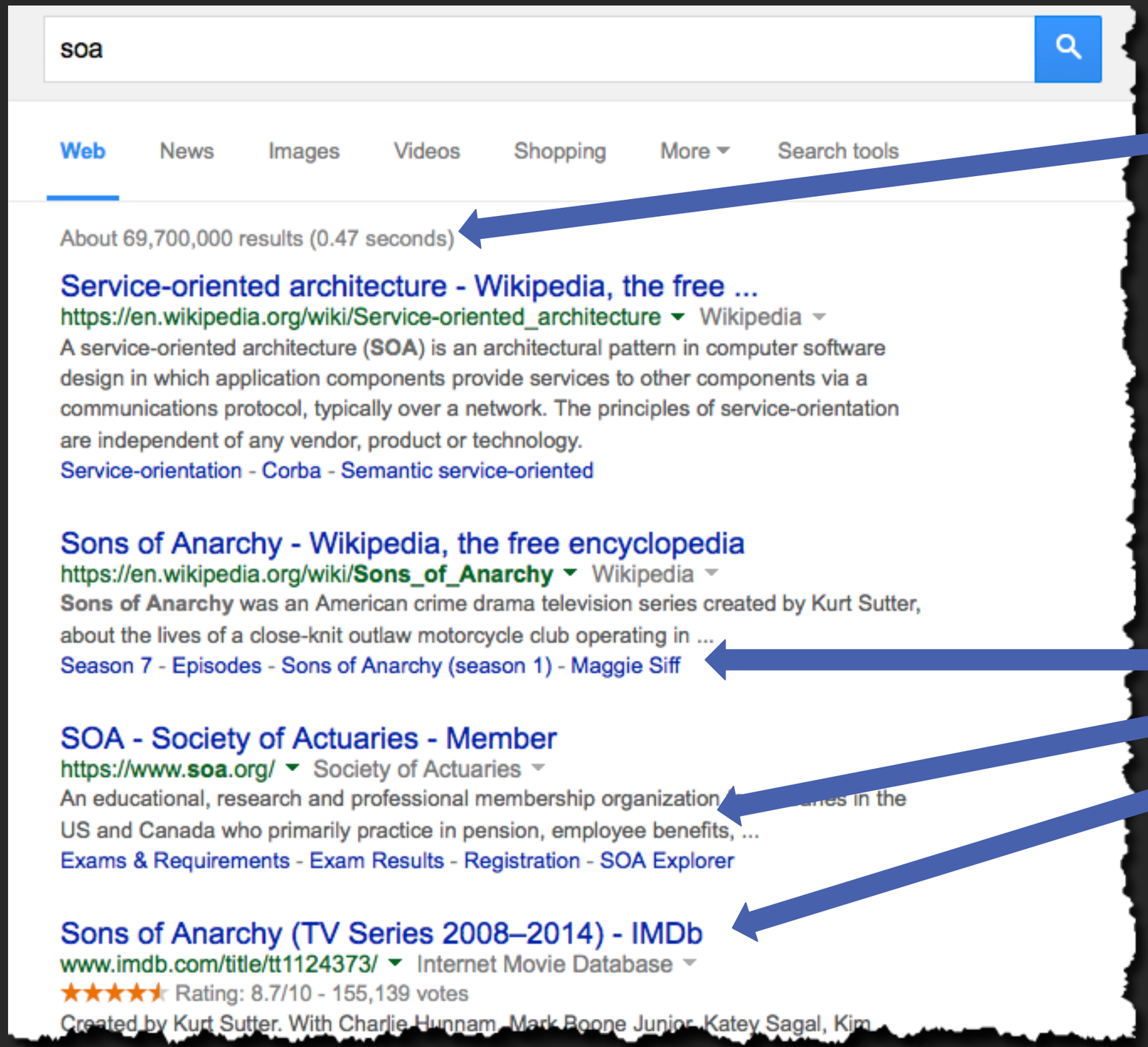
- User interface
- Business logic
- Data access

No separation

SOA

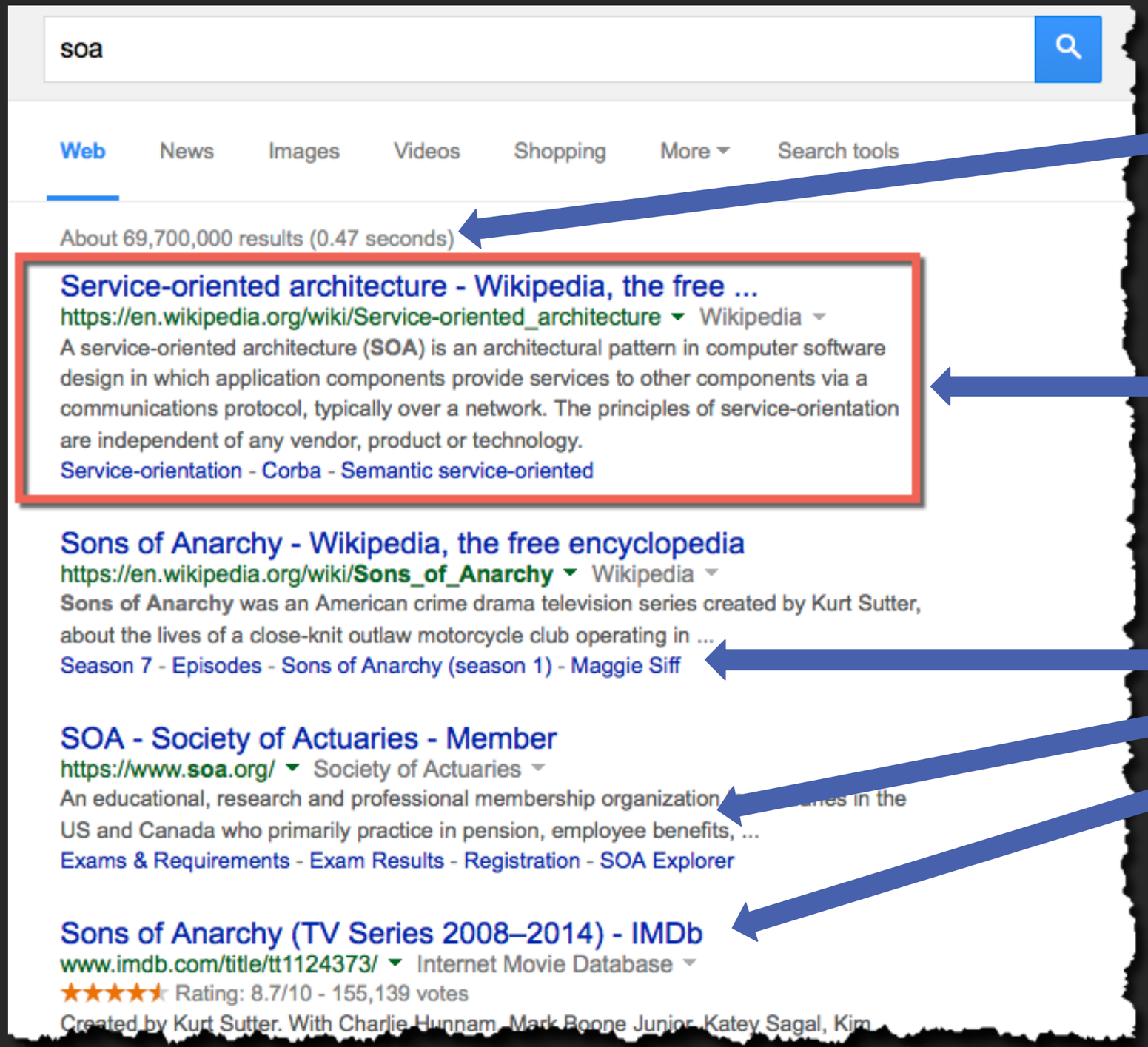
What does this mean?





Now that's a lot of things to read!

This is *not* where we want to start!

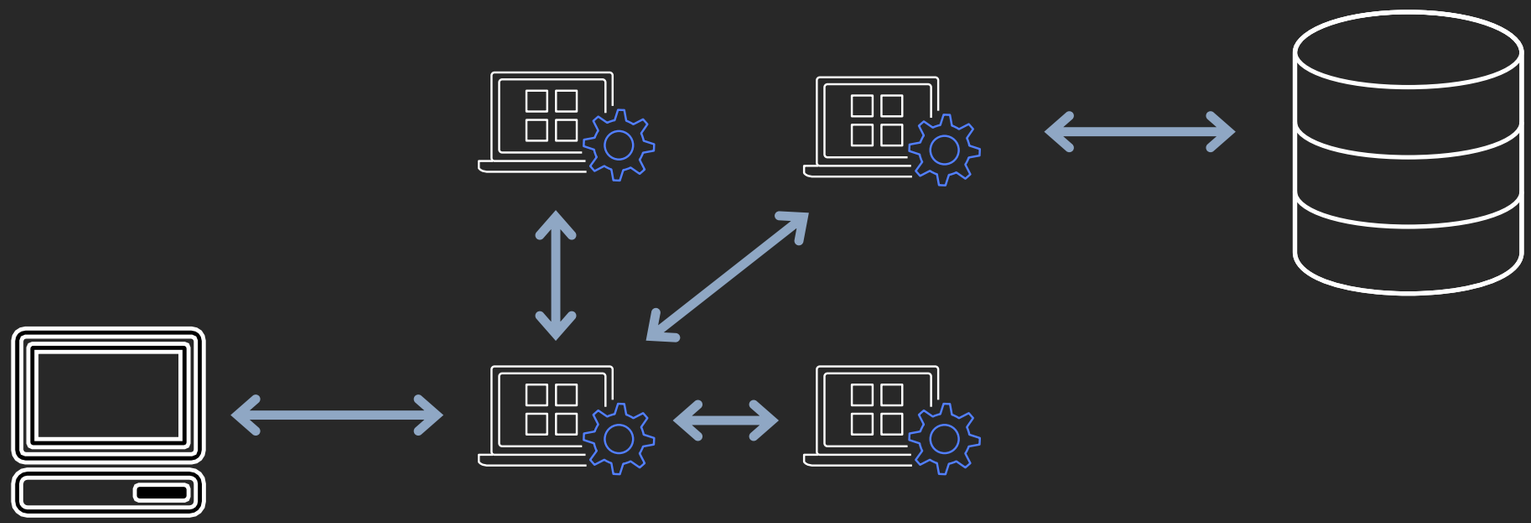


Now that's a lot of things to read!

This *is* where we want to start!

This is *not* where we want to start!

Service-oriented architecture



Presentation tier

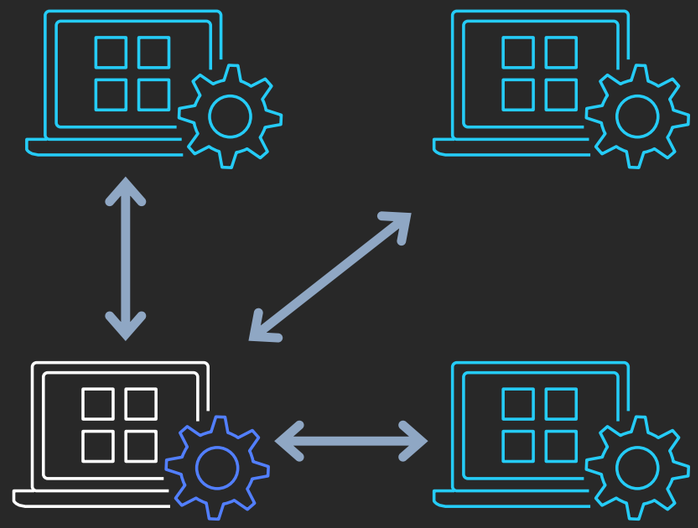
Logic tier

Data tier

SOA'ing

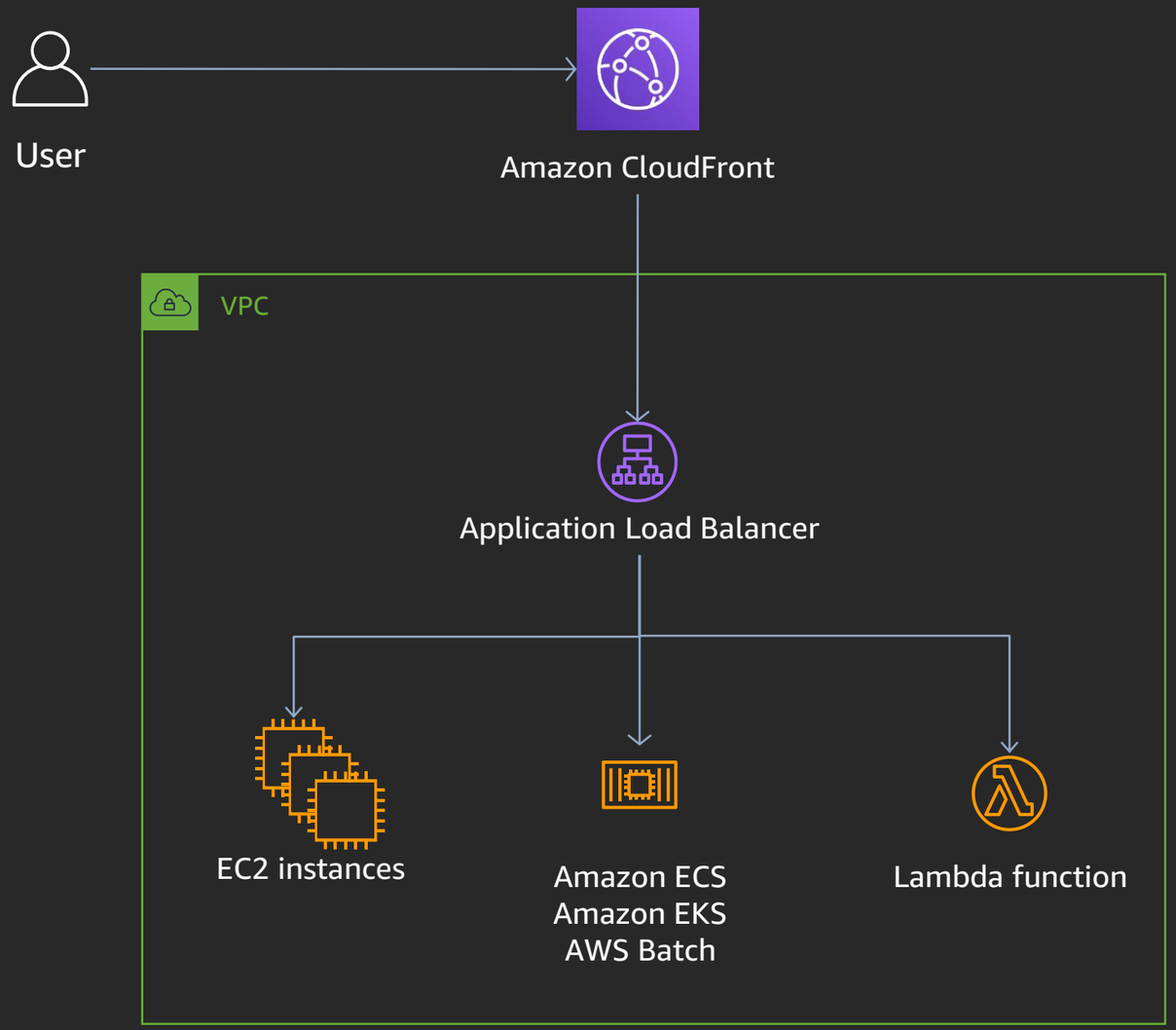
Move services into their own tiers

- Treat them separately
- Scale them independently



It offers flexibility and greater understanding of each component

Moving toward modern applications with containers

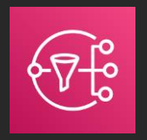


Serverless = winning

Don’t reinvent the wheel

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


Amazon API Gateway

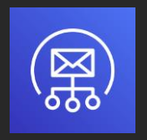

Amazon SNS


Amazon Elasticsearch Service


Amazon SQS

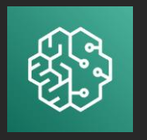

AWS Fargate


AWS Lambda


Amazon SES

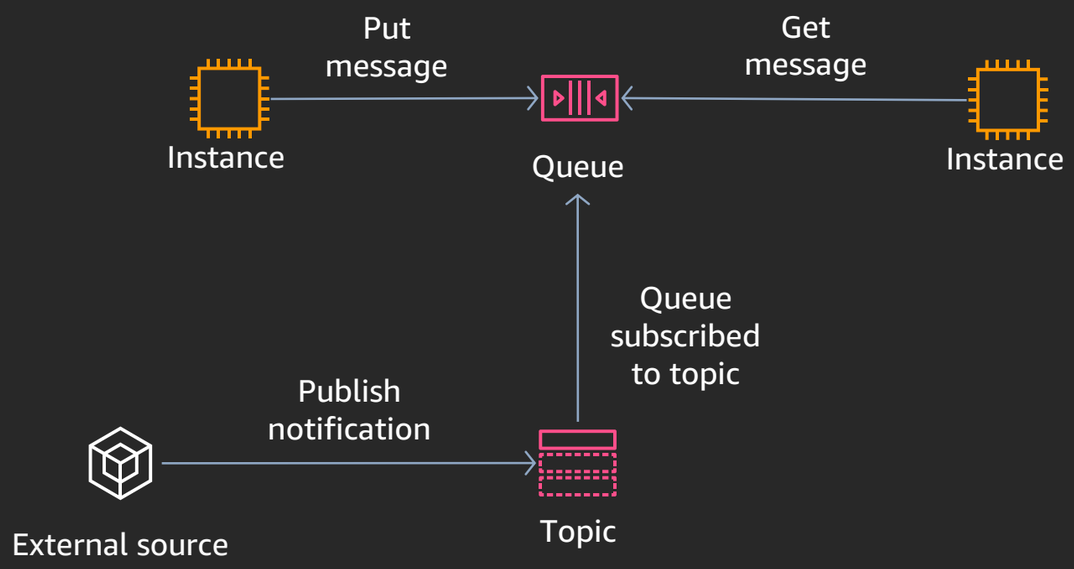

AWS Step Functions


AWS Elemental MediaConvert


Amazon SageMaker

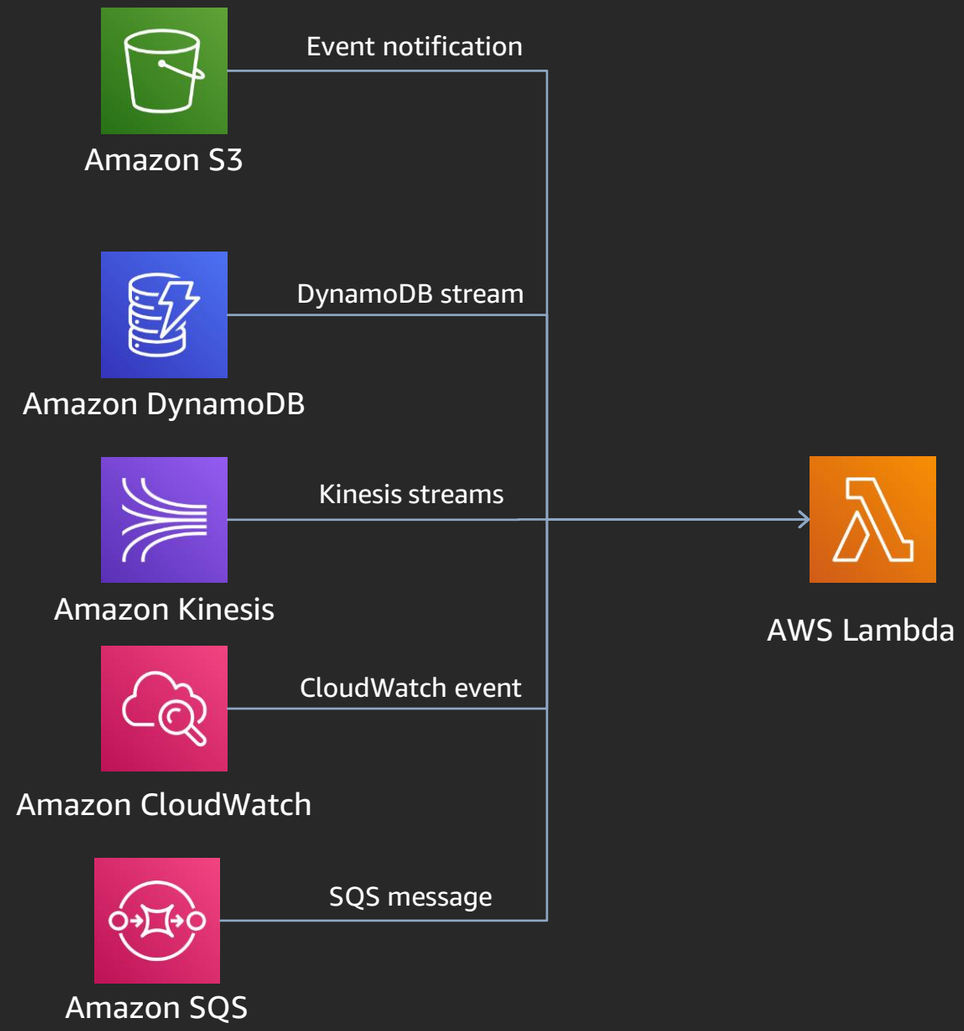
Loose coupling: Amazon SQS and Amazon SNS

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



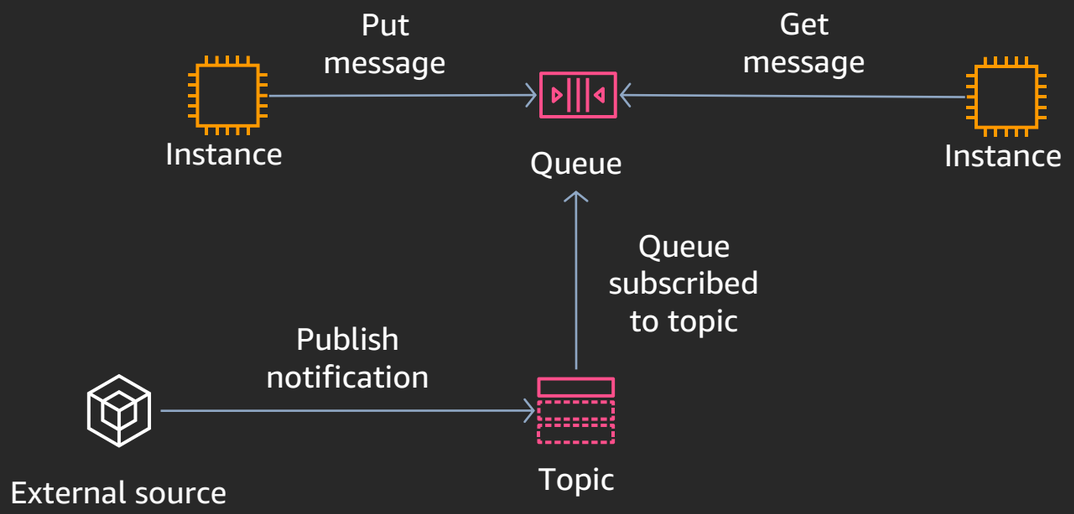
Event-driven compute: AWS Lambda

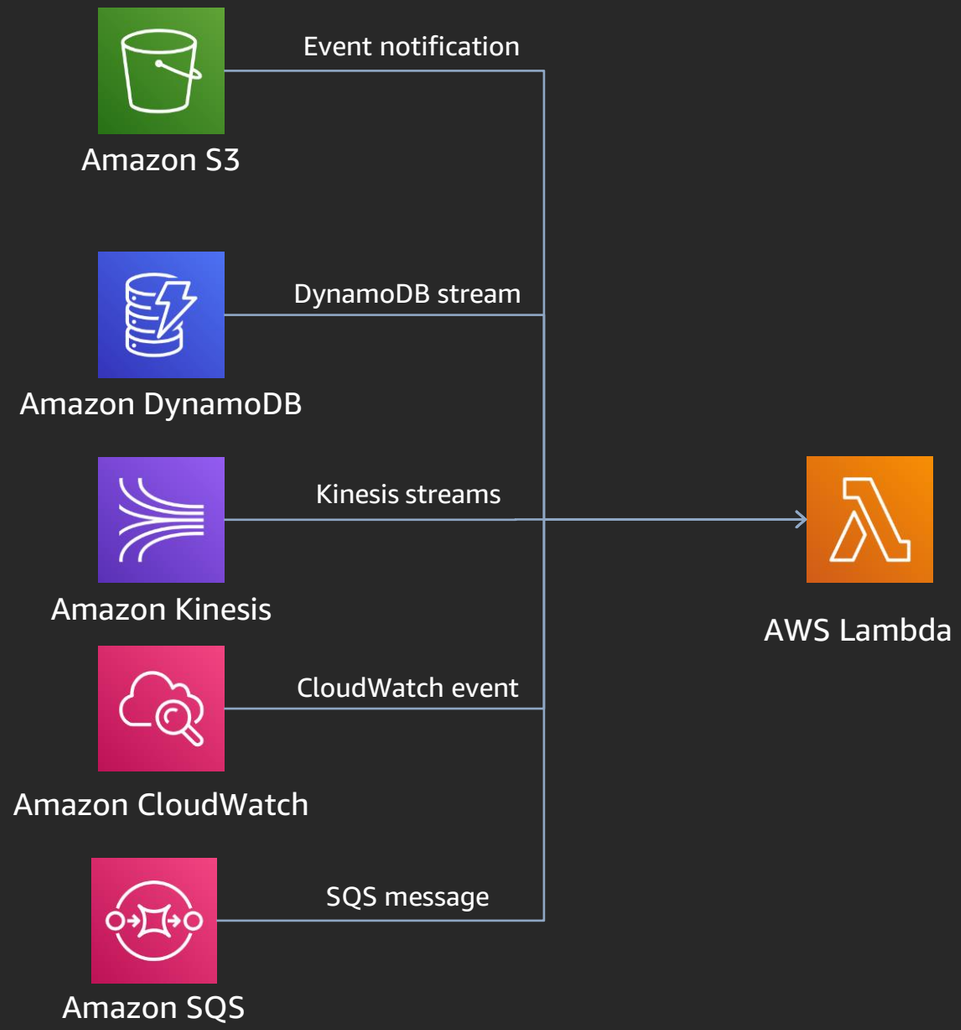
- Functions triggered by events
- Java, Go, PowerShell, Node.js, C#, Ruby, Python
- Serverless
- Implicit scaling



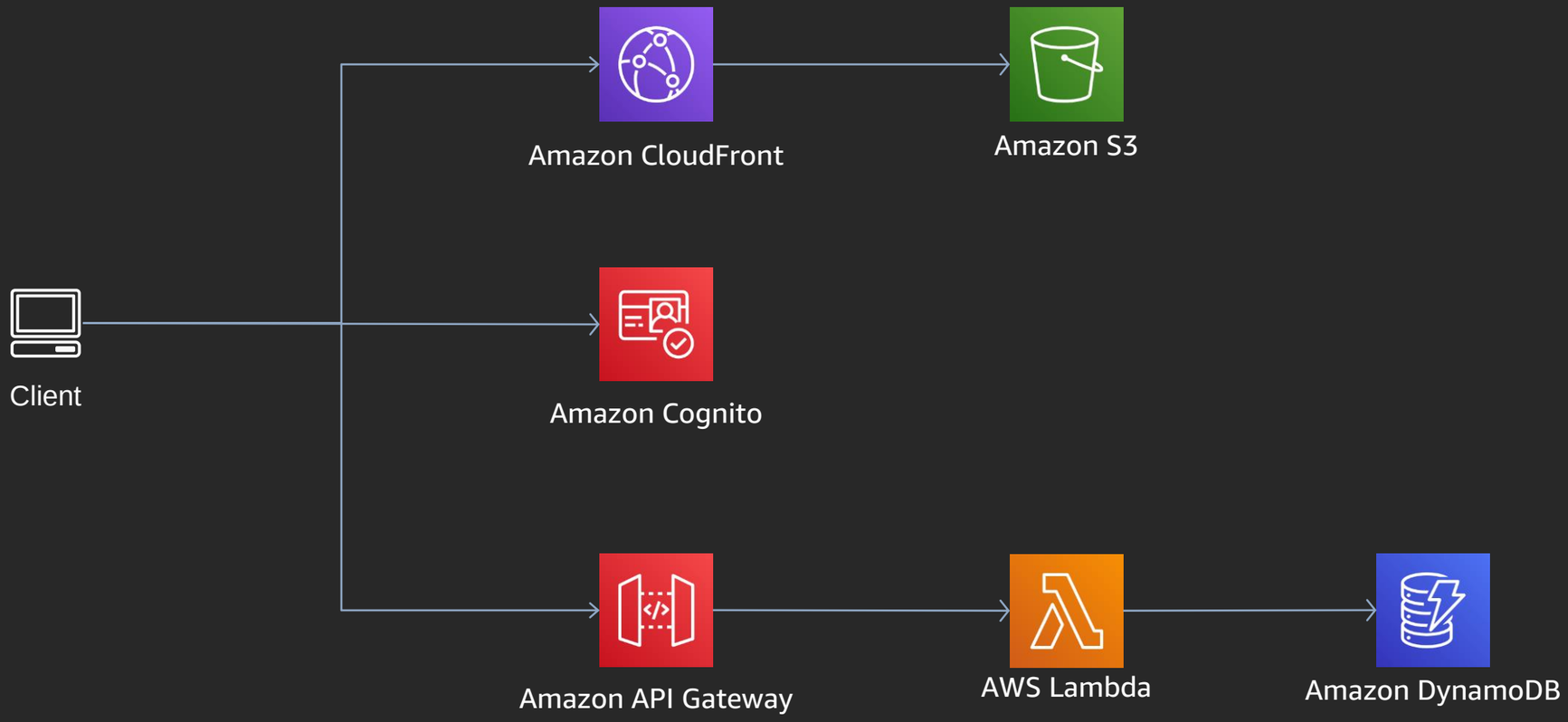
Loose coupling sets you free

- 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*



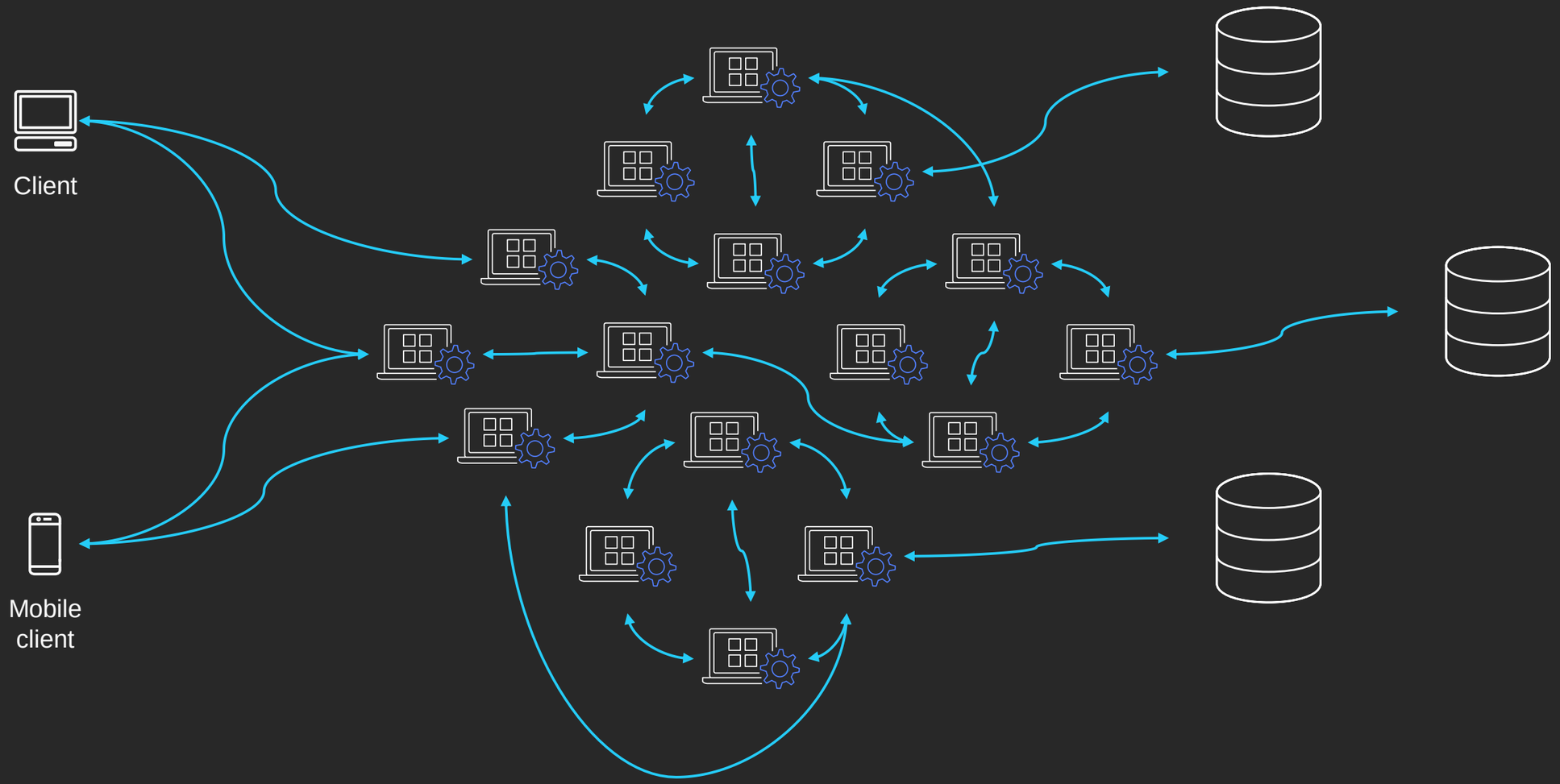


Serverless web application

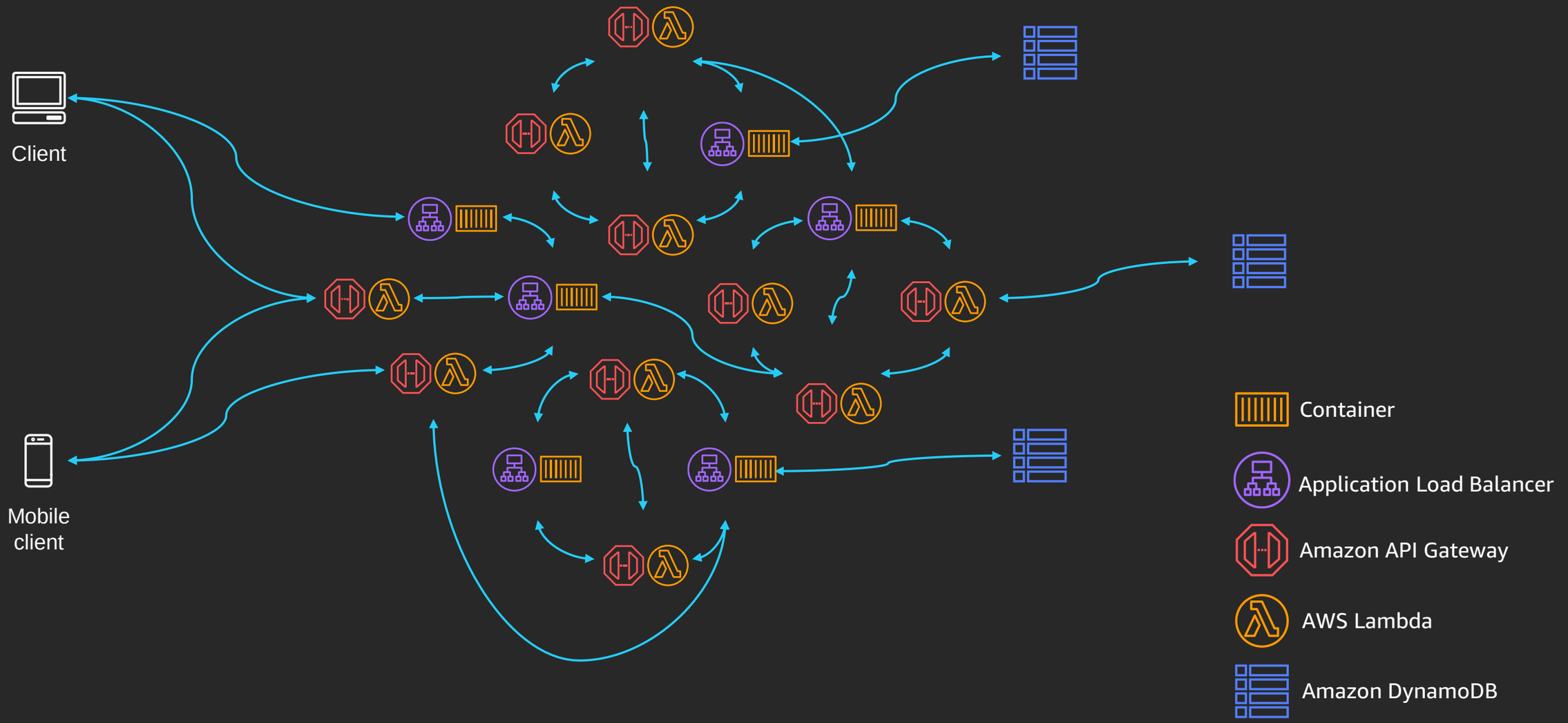



ember
React
Vue.js
METEOR
AWS Amplify

The microservices architecture



The microservices architecture



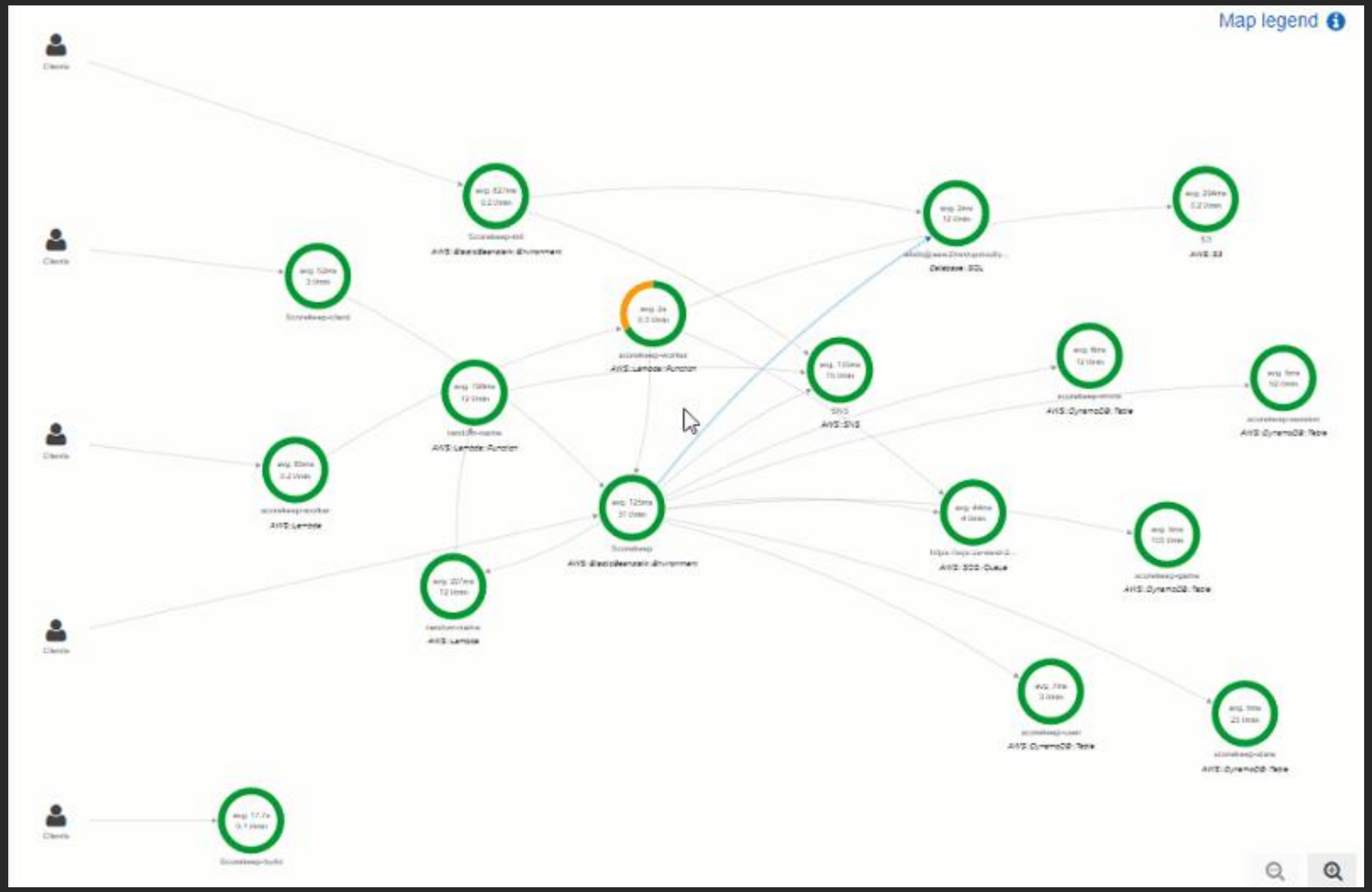
AWS X-Ray



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



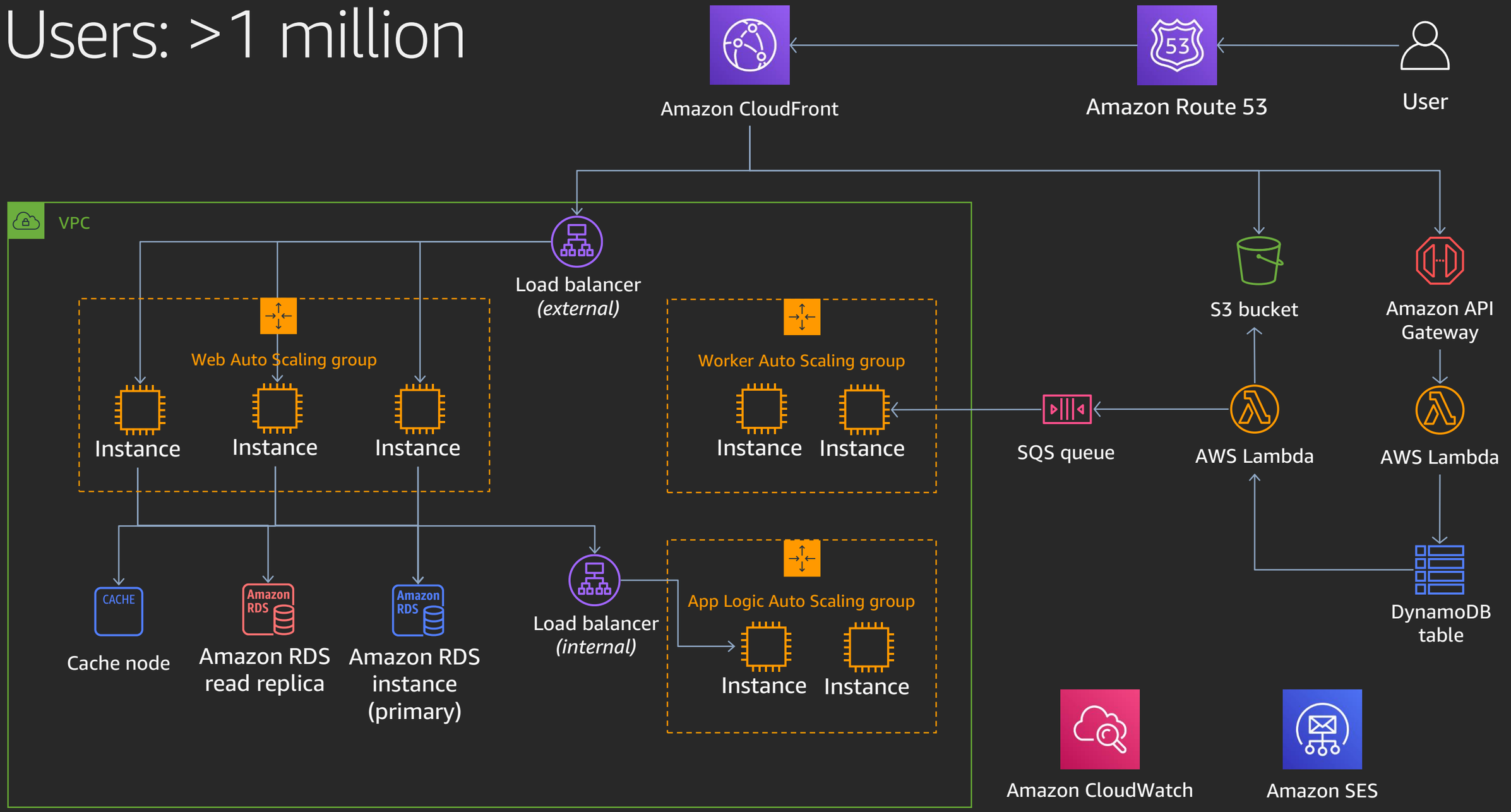
Users: >1 million

Users: >1 million

Reaching and exceeding a million requires 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 databases
- Moving state-off tiers that auto scale

Users: >1 million



The next big steps

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >5 million to 10 million

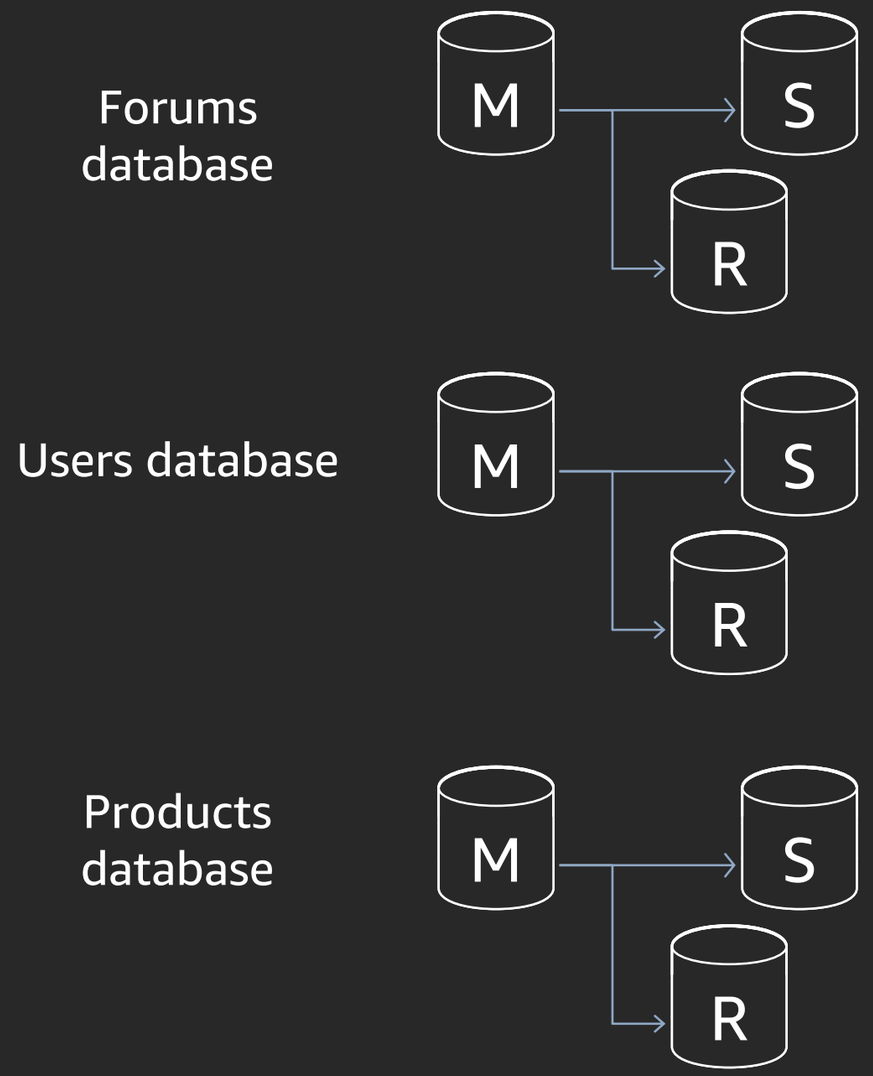
Database issues?

Potential solutions

- Federation: Splitting into multiple databases based on function
- Sharding: Splitting one data set across multiple hosts
- Moving some functionality to other types of databases (NoSQL, Graph)

Database federation

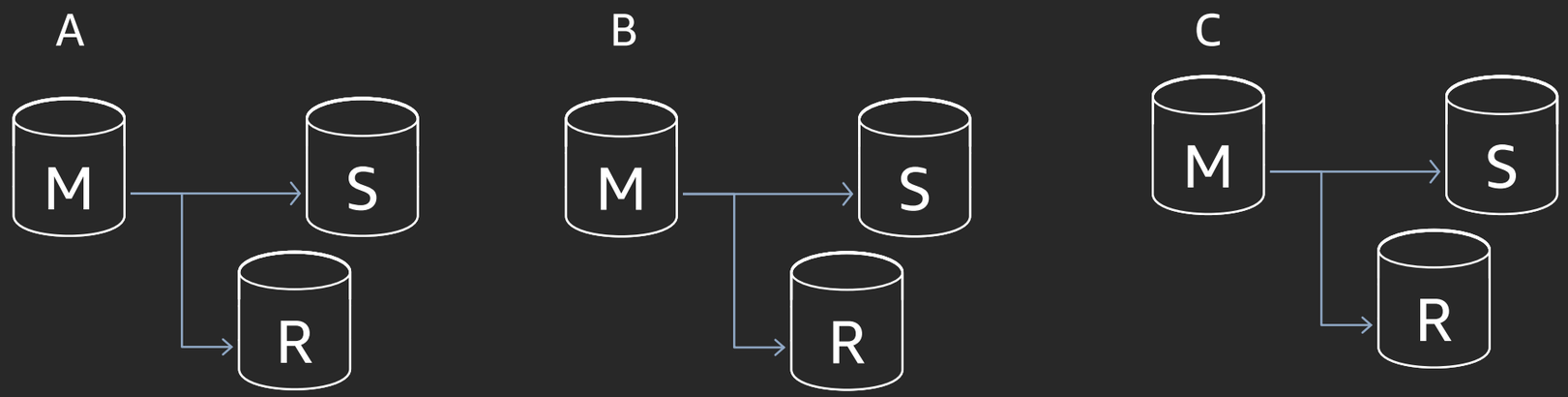
- Split up databases by function/purpose
- Harder to do cross-function queries
- Essentially delays sharding/NoSQL
- Won’t help with single huge functions or tables



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



Amazon DynamoDB

Similar in a sense to federation

NoSQL versus SQL

Leverage managed services such as 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

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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 . . .

Users: >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
- Deeply analyze your entire stack
- Build serverless whenever possible

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

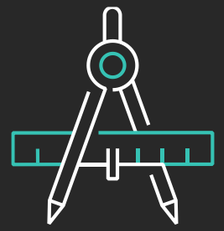
aws.amazon.com/answers

aws.amazon.com/premiumsupport

A solutions architect

Learn to architect with AWS Training and Certification

Resources created by the experts at AWS to propel your organization and career forward



Free foundational to advanced digital courses cover AWS services and teach architecting best practices



Classroom offerings, including Architecting on AWS, feature AWS expert instructors and hands-on labs



Validate expertise with the **AWS Certified Solutions Architect - Associate** or **AWS Certification Solutions Architect - Professional** exams

Visit aws.amazon.com/training/path-architecting/

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws training and certification

Thank you!

Brian Farnhill
brfarn@amazon.com

Hong Pham
phamh@amazon.com

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.