
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

STP211-R1

Scaling on AWS for your first 10 million users

Chris Munns

Startup Tech Lead/Advisor
Amazon Web Services

Carole Suarez (she/her)

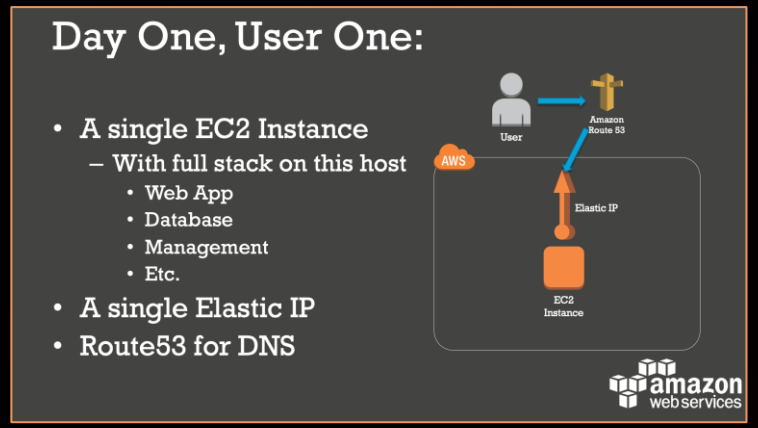
Startup Solution Architect
Amazon Web Services

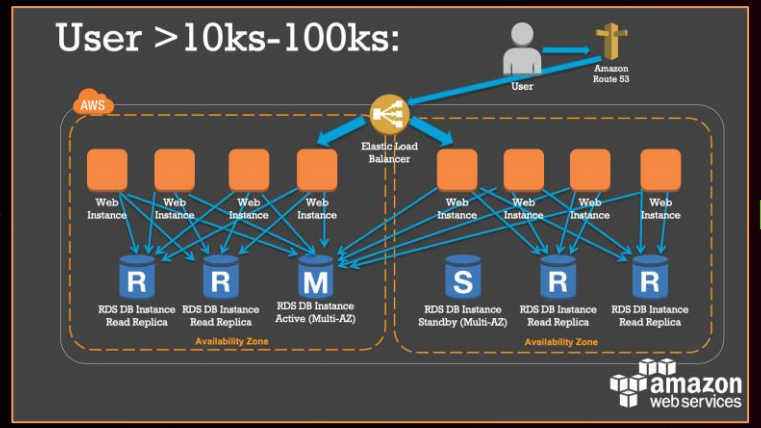
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

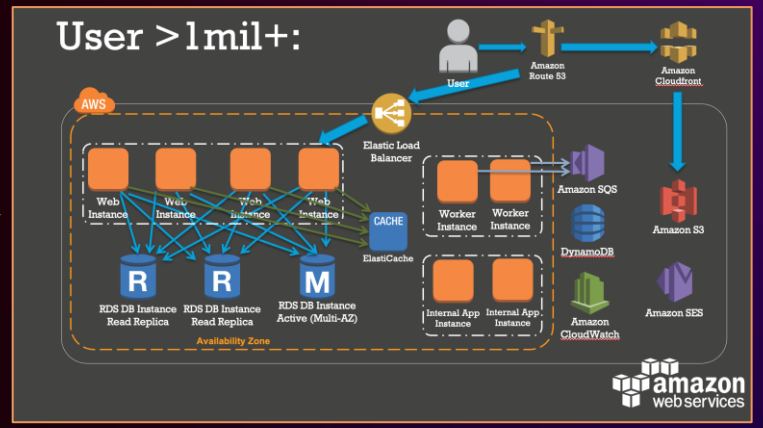
This is an old talk! But much has changed . . .

First delivered in 2013!

In the past this talk revolved around taking a single application and scaling it from a single host to multiple using load balancing, AWS Auto Scaling, and many other AWS services to shift parts of it to managed services.







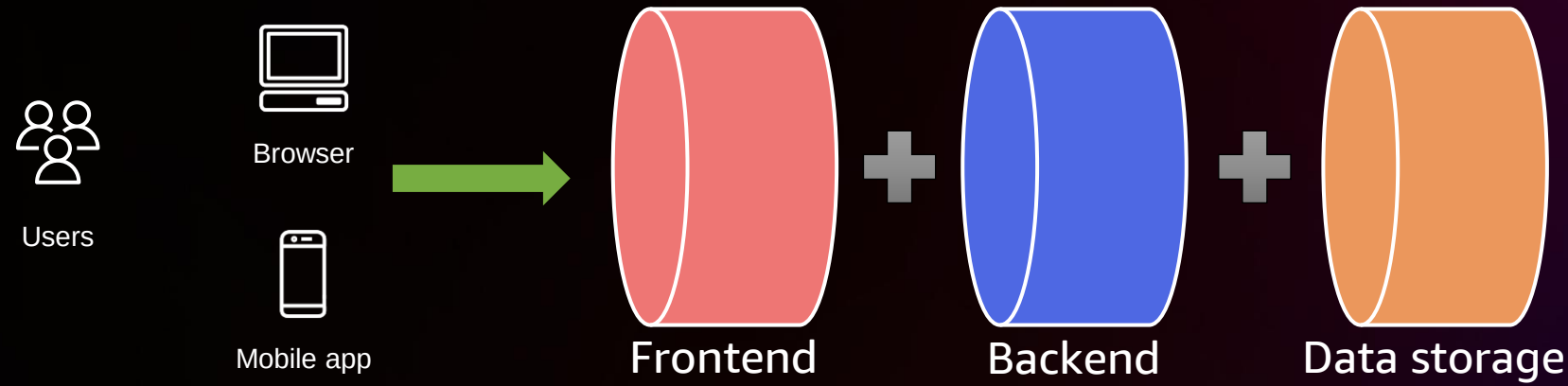
But today this doesn't really reflect the reality of where people start building an app.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What do we mean by an “app”

In the context of this session assume we mean the “full stack” necessary to deliver a business’s core technology product.

- Could be the entirety of a startup’s product
- Could be one of many products in a larger company
- For today: App == user-interfacing layer + business logic layer + data storage



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“No architecture is designed for high scalability on day 1. But we’ll try.”

Me

Today

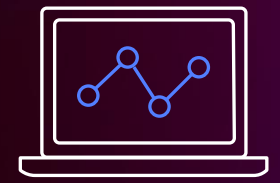
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Build

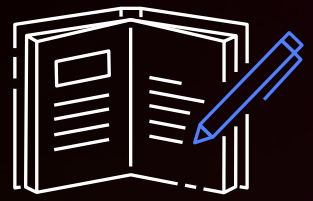


Measure





Learn

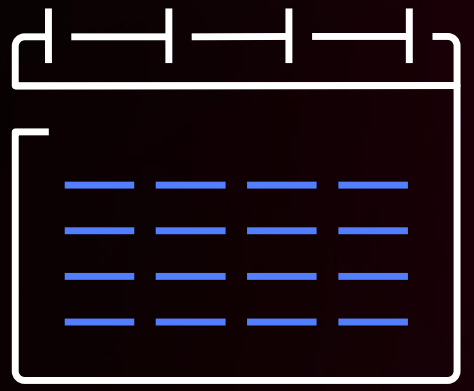


aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

So let’s start from

Day 1



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

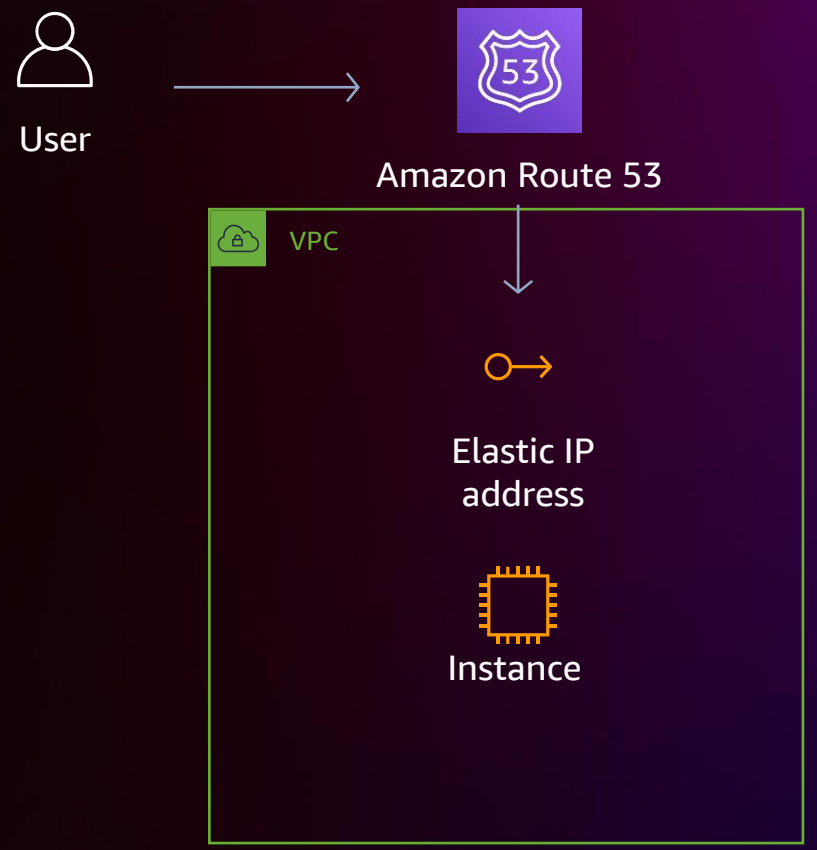
Users: >1

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Day 1 the old way – Single instance

Previously you would see a single-instance-based starting architecture that would host the 3 main layers of an application:

1. Presentation
2. Business
3. Data



In a sense this still happens, but now it's growing "out of style" for modern application development

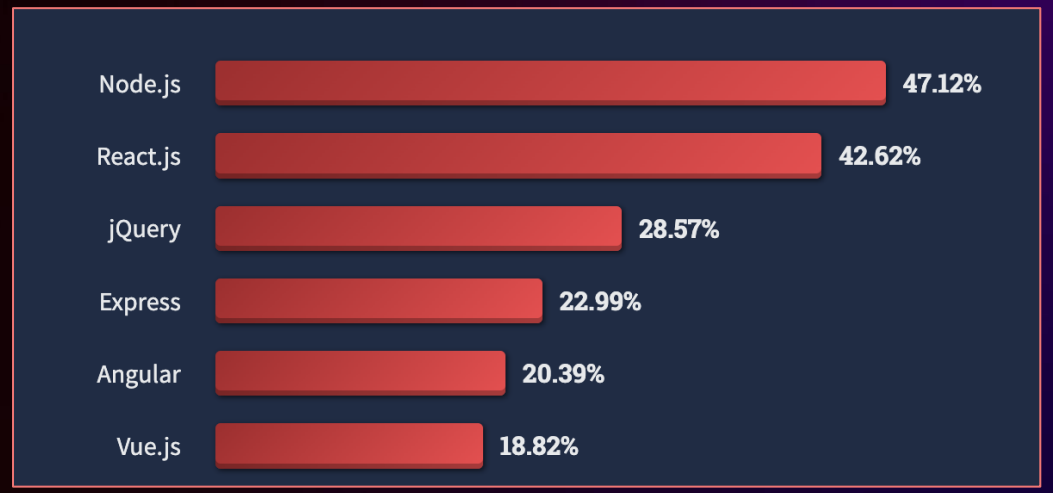
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What's changed?

This old way of starting host-based has grown out of favor due mostly to the popularity of frontend framework technology:

- Node.js React, jQuery, Angular, and Vue.js round out top web frameworks and technologies in the Stack Overflow Developer Survey 2022 (see chart/QR link)
  - Flutter and React Native are the two popular cross-platform frameworks
- Growing ecosystem on top of these frameworks
- Growing consulting populations

Static frameworks seemingly de facto standard now



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What’s the impact?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

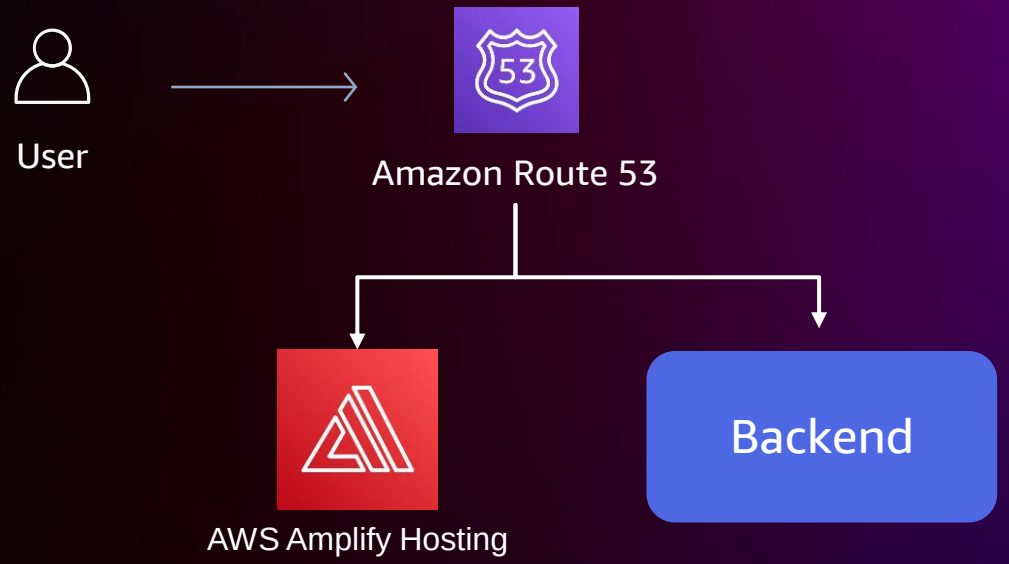
Users >1: With modern frontend

With a modern frontend, developers are choosing to deploy them to specialized hosting products.

Why?

- Greatly reduced operations overhead
- Built in scale/performance
- Integrations with the modern frontend frameworks
- Aligned Developer experience capabilities

The backend then becomes a different component(s)



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amplify Hosting

DEPLOY AND HOST GLOBALLY USING AMAZON CLOUDFRONT

How it works

1

Connect your repository

GitHub

Bitbucket

GitLab

AWS CodeCommit

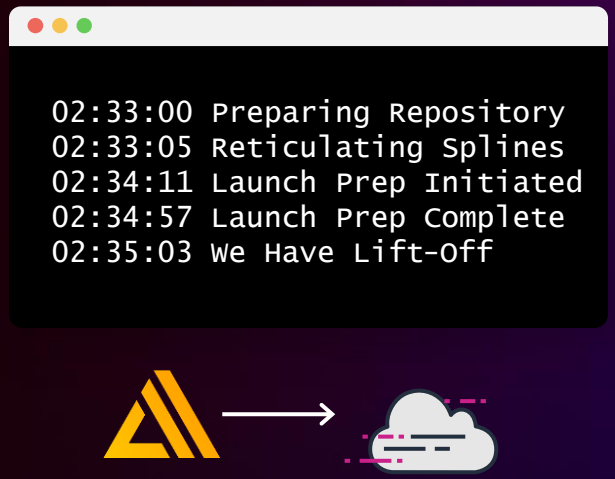
2

Configure build settings



3

Deploy your app



aws

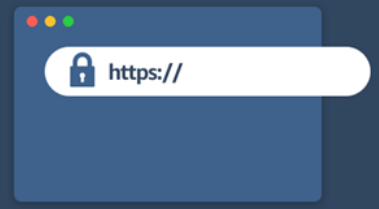
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amplify Hosting

FEATURES FOR HOSTING MODERN WEB APPLICATIONS



Globally available



Easy custom domain setup



Simplified continuous workflows



Feature branch deployments



Atomic deployments



Password protection

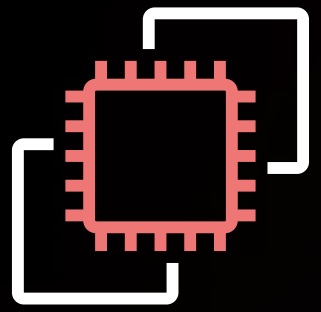
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What about the backend?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Options for compute



Amazon EC2

Virtual server instances in the cloud



Amazon ECS, Amazon EKS, and AWS Fargate

Container management service for running Docker on a managed cluster of EC2



AWS Lambda

Serverless compute for stateless code execution in response to triggers

aws

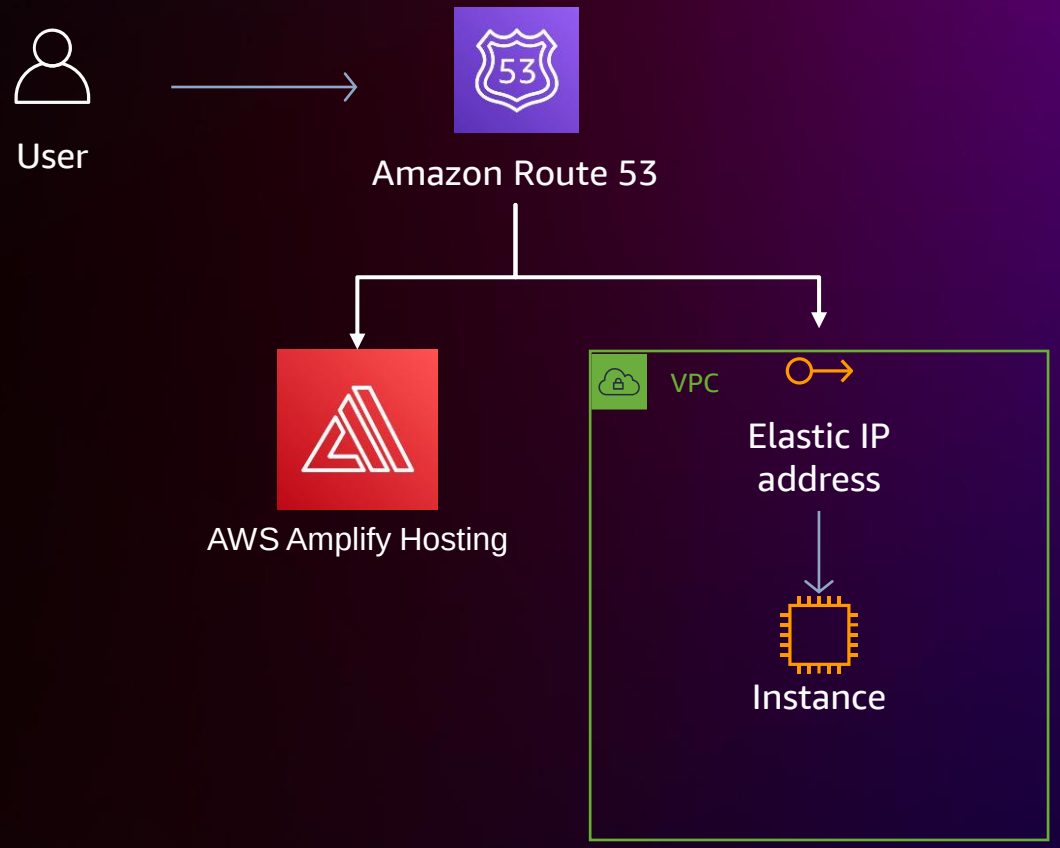
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Evaluating compute options

The instance-based model is still one possible model for hosting your backend business logic and data tiers, but with clear disadvantages:

- No **failover**
- No **redundancy**
- **Can't scale** individual components independently
- **Constrained on technology** choices for individual components

Too many eggs in one basket?



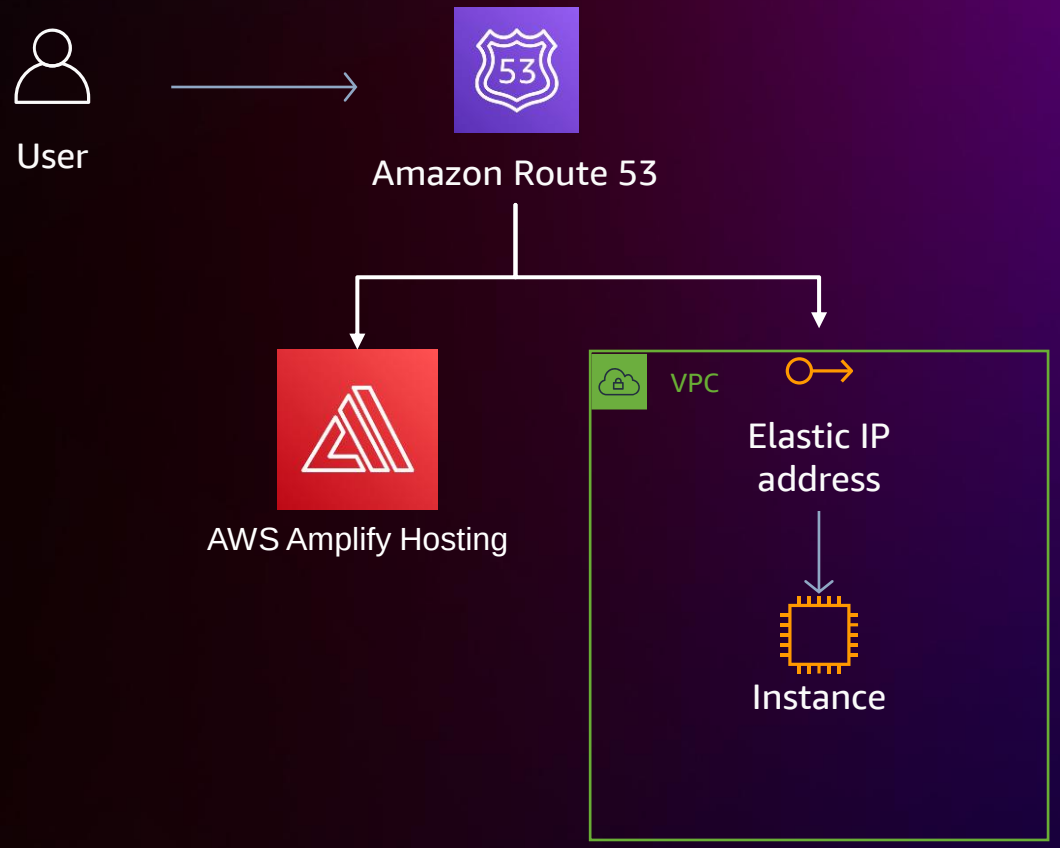
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We can go far with this, but . . .

The downsides of starting out this way become apparent quickly.

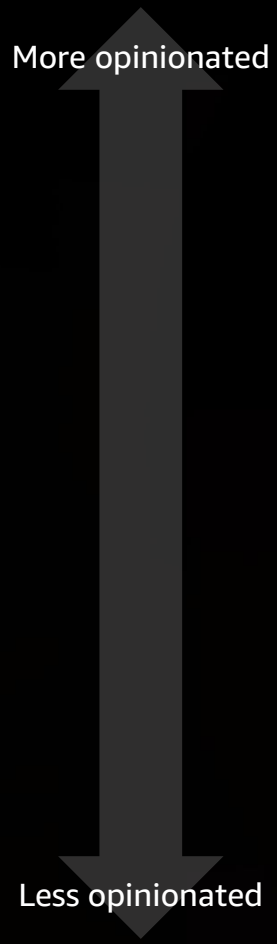
- Larger instance sizes address scale but not failover or redundancy challenges
- Management of the instance itself becomes a challenge of conflicting resource usage
  - even with containerization on a single instance
  - different scaling challenges for databases beyond just adding more compute/memory/storage

AWS's guidance: make use of managed compute for your backend and managed databases for your data tier.



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Evaluating managed compute on AWS

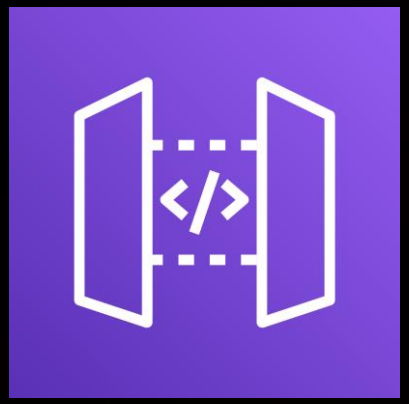


	AWS manages	Customer manages
AWS Lambda Serverless functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate Serverless containers	• Container orchestration, provisioning • Cluster scaling • Physical hardware, host OS/kernel, networking, and facilities	• Application code • Data source integrations • Security config and updates, network config, management tasks
ECS/EKS Container-management as a service	• Container orchestration control plane • Physical hardware software, networking, and facilities	• Application code • Data source integrations • Work clusters • Security config and updates, network config, firewall, management tasks
EC2 Infrastructure-as-a-Service	• Physical hardware software, networking, and facilities	• Application code • Data source integrations • Scaling • Security config and updates, network config, management tasks • Provisioning, managing scaling, and patching of servers

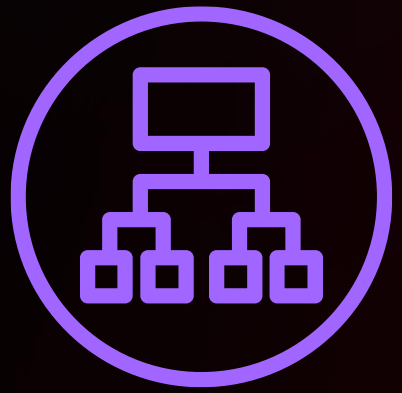
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Exposing business logic to the frontend

THREE OPTIONS FOR EXPOSING AN API



Amazon API Gateway



Application Load Balancer

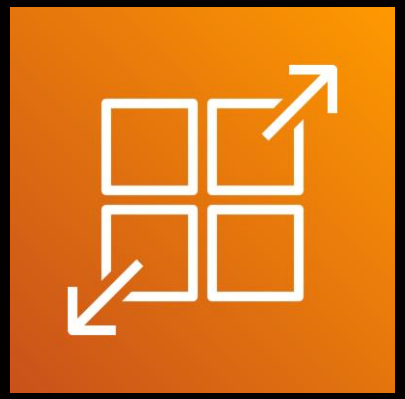


AWS AppSync

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS App Runner



AWS App Runner

- Build, deploy, and run containerized web applications and API services
- Simplified management reduces overall operational overhead and need for deep experience running containers
- Built on ECS with Fargate, Auto Scaling, Elastic Load Balancing (ELB), and Amazon Elastic Container Repository (Amazon ECR)
- Supports popular language runtimes such as Node.js, Python, php, Go, Java, .Net, and Rails
- Both public and private applications

aws

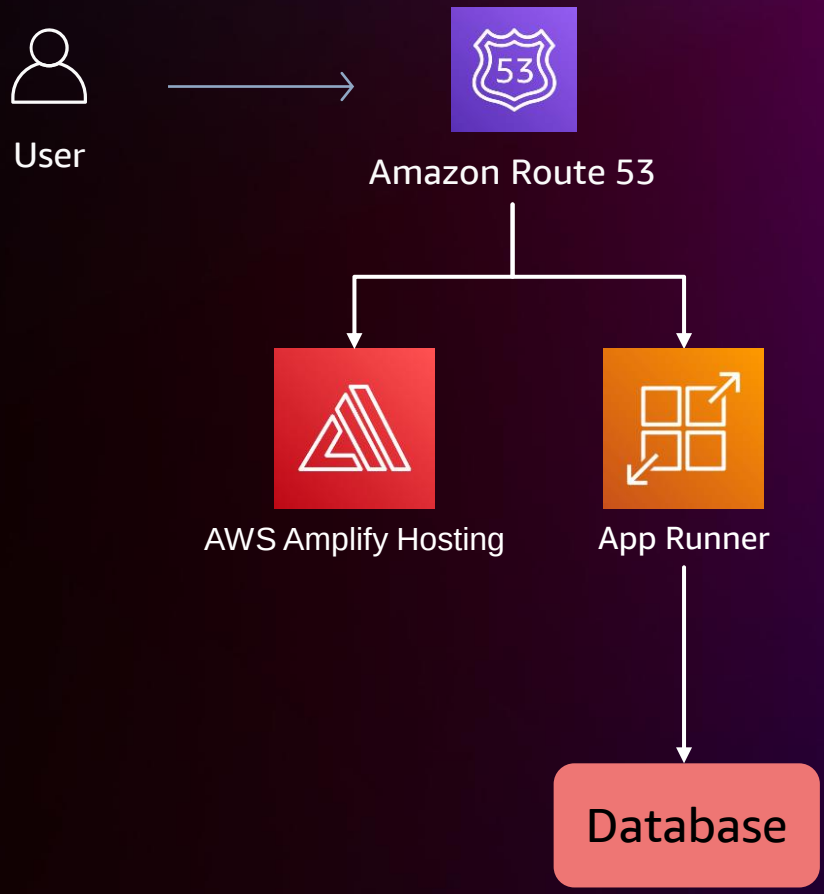
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users >1: with modern frontend and backend

Developers are looking to leverage managed compute to rapidly start building and deploying their backend applications.

Why?

- Greatly reduced operations overhead
- Built in scale/performance
- Integrations with the modern backend frameworks
- Aligned Developer experience capabilities



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

To NoSQL, or not to NoSQL?



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Start with SQL databases

© 2022, 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

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ah ha!

You said, “massive amounts of data.”

That’s me.

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multiple terabytes of data in year 1?

Incredibly data-intensive workload?

OK!

You *might* need NoSQL

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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 multiple terabyte range)

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

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

But this isn’t most of you.
So . . .

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Start with SQL databases

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Aurora

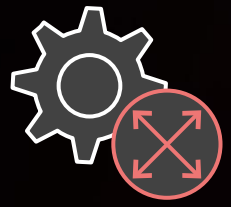
Relational databases built for the cloud – performance and availability of commercial databases at 1/10th the cost



Performance and scalability

Several times faster than standard MySQL and PostgreSQL

15 read replicas



Availability and durability

Fault-tolerant self-healing storage

6 copies of data across 3 AZs

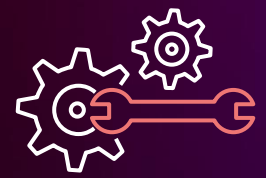
Single global database with cross-Region replication



Highly secure

Network isolation

Encryption at rest/transit



Fully managed

Managed by Amazon RDS: no hardware provisioning, software patching, setup, configuration, or backups

The fastest growing service in the history of AWS

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Aurora Serverless v2



On-demand and auto scaling configuration

Automatically scales capacity based on application needs

Simple **pay-per-use** pricing per second

Next version scales instantly to support demanding applications

Worry-free database capacity management

aws

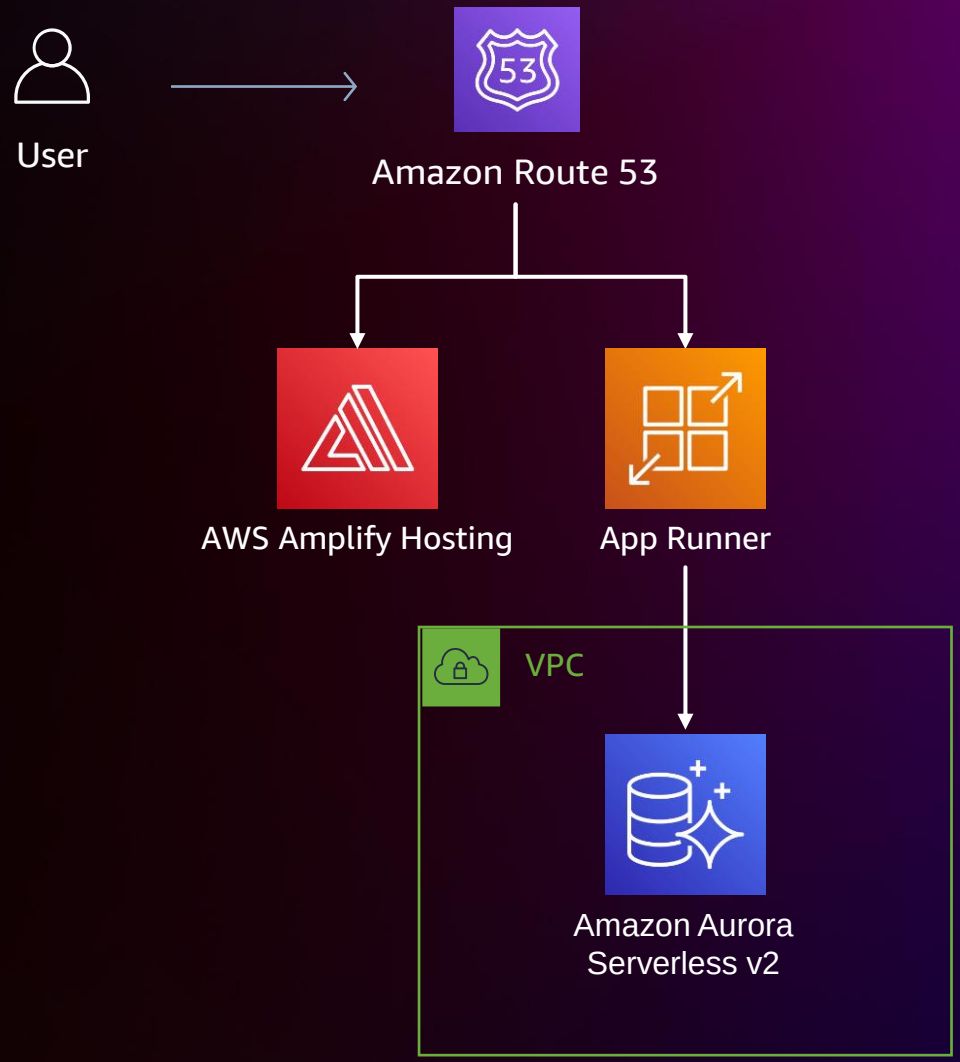
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users >1:

By leveraging a managed services for frontend, backend, and database we can start off day 1, user 1, with a great foundation and little overhead.

- No self-managed infrastructure
- Built in scalability OR easy knobs to turn to increase capacity as needed
- Built in high availability (multi-AZ) in a single Region
- Layers of security and access controls from the start to establish good practices
- Aligned costs to value

From here we can go pretty far!



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >100

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >1000

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000

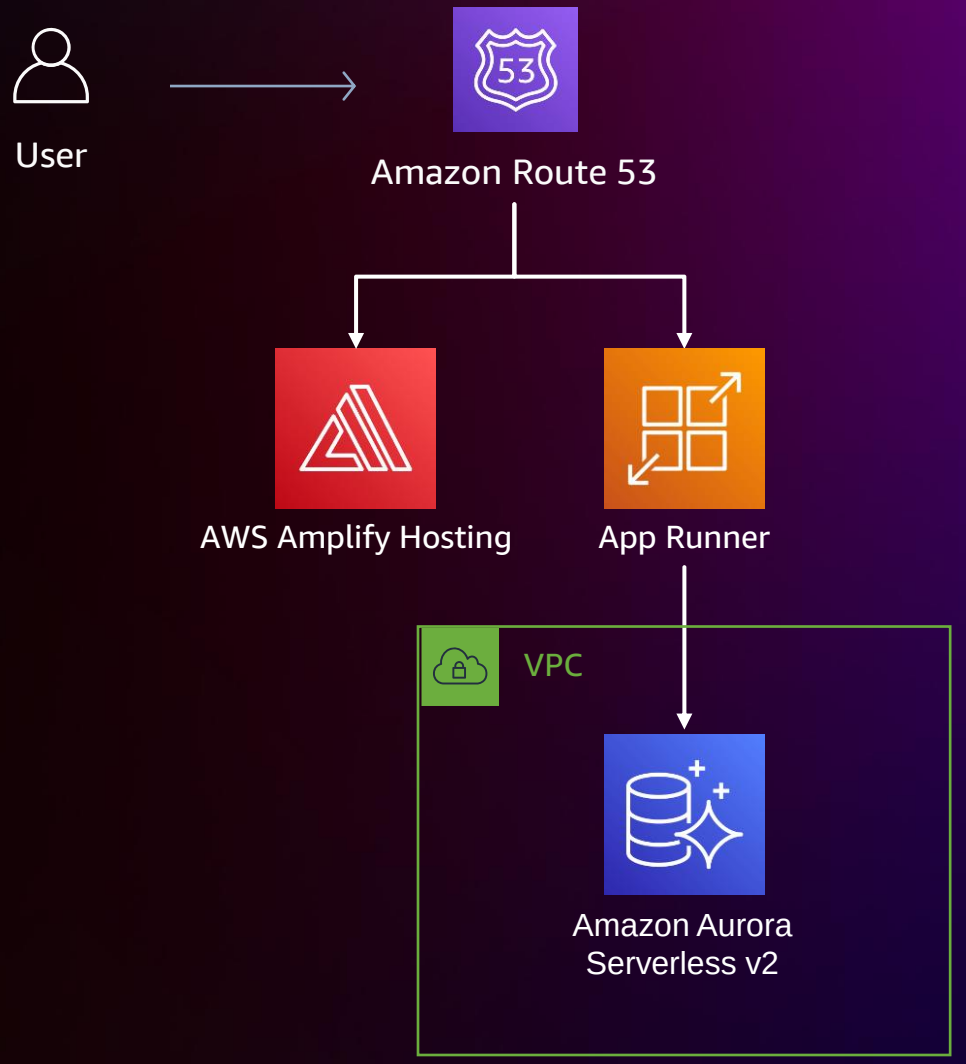
maybe?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000. What starts to go wrong?

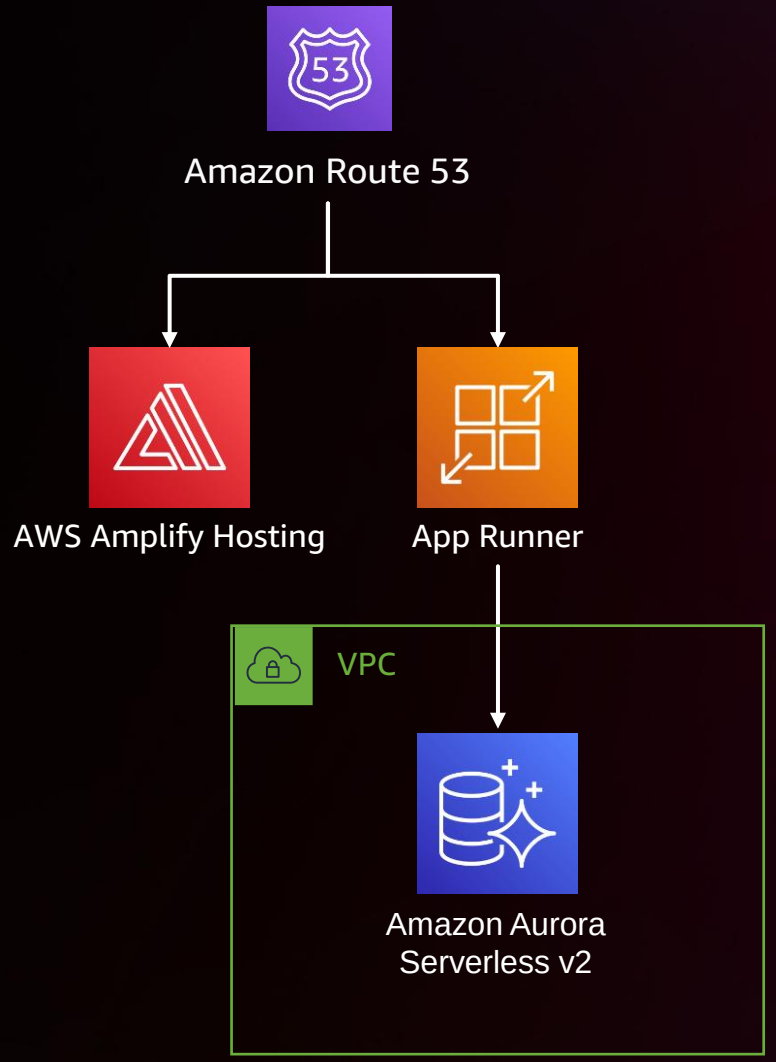
The current stack will scale incredibly far, but the scaling of single tier/monolithic applications can sometimes only go so far. You'll eventually run into issues common in most architectures:

- Varied needs of the product complicating others
- Poor performance in one part impacting other parts
- Slowing queries in the database due to large table sizes/index growth



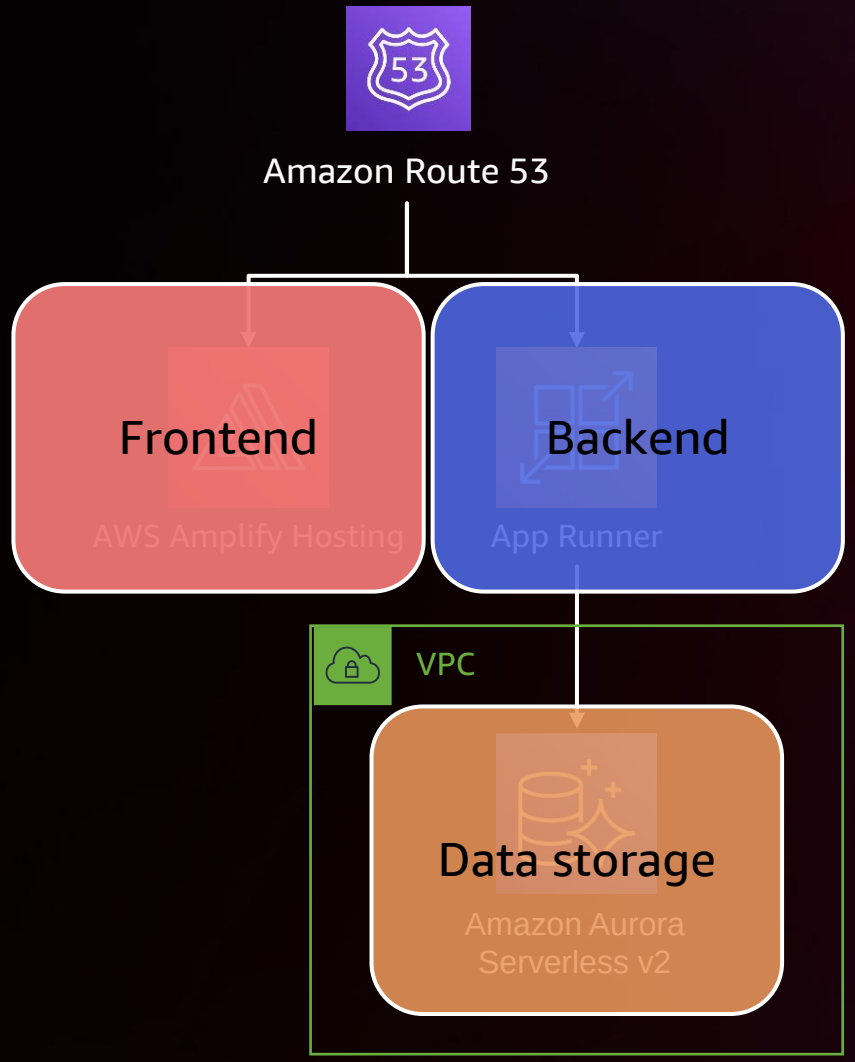
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s dive in



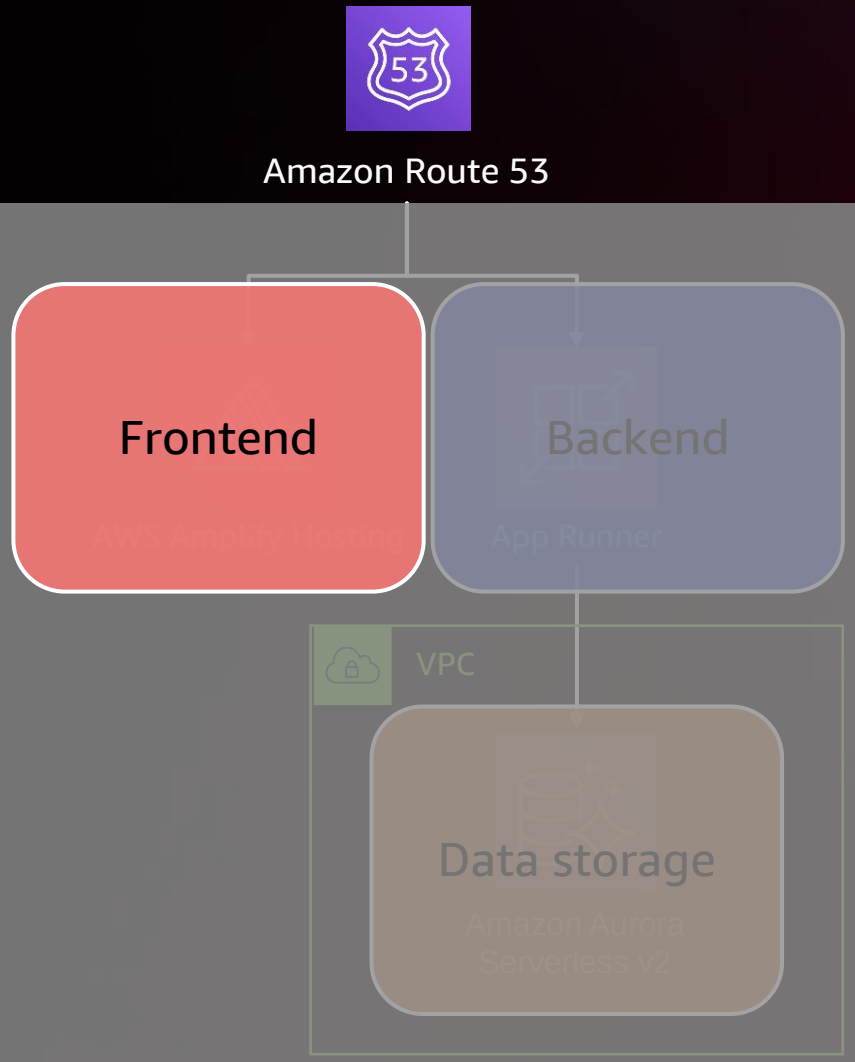
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s dive in



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s dive into the frontend tier



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling the frontend



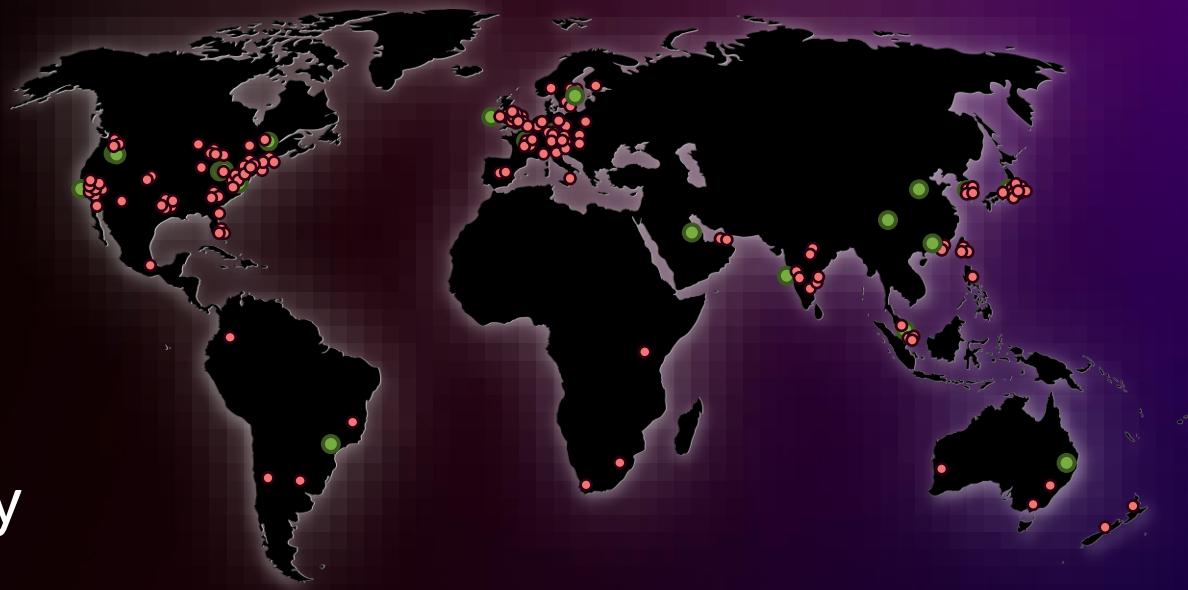
Amazon CloudFront

Generally speaking, Amplify Hosting can scale to meet customer needs.

Built on top of the 410+ Amazon CloudFront PoPs globally

Performance typically comes from

- Tuning frontend code
- Reducing the number of backend calls
- Caching images/JavaScript/CSS effectively



Amazon CloudFront Points of Presence (PoP)

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling the frontend



Amazon CloudFront

Generally speaking, Amplify Hosting can scale to meet customer needs.

Built on top of the 410+ Amazon CloudFront PoPs globally

Performance typically comes from

- Tuning frontend code
- Reducing the number of backend calls
- Caching images/JavaScript/CSS effectively

```
customHeaders:
  - pattern: '/img/*'
    headers:
      - key: 'Cache-Control'
        value: 's-maxage=3600'
```

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Before we go too much further!

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



We can’t tune what we aren’t measuring

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS services for observability



Amazon CloudWatch

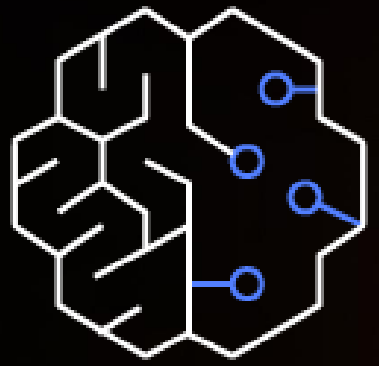


AWS X-Ray

Dashboards
Logs
Metrics
Alarms
Events
RUM

Traces
Analytics
Service map

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Leverage machine learning (ML) to assist you

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS services for ML-assisted DevOps



Amazon
DevOps Guru

Detect unusual behavior, analyze performance, and drive correction of issues



Amazon
CodeGuru

Analyze application code for common issues, performance, and cost improvements

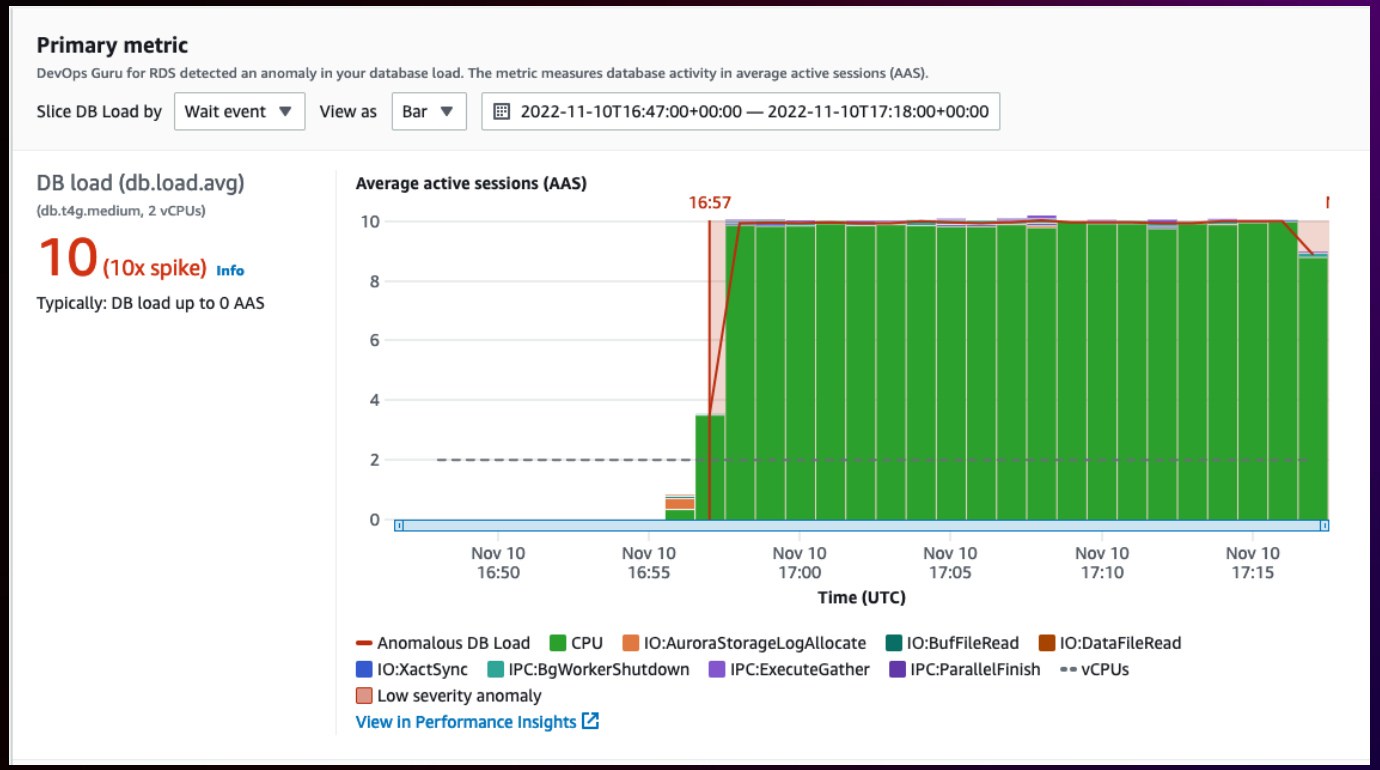
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tuning for scale

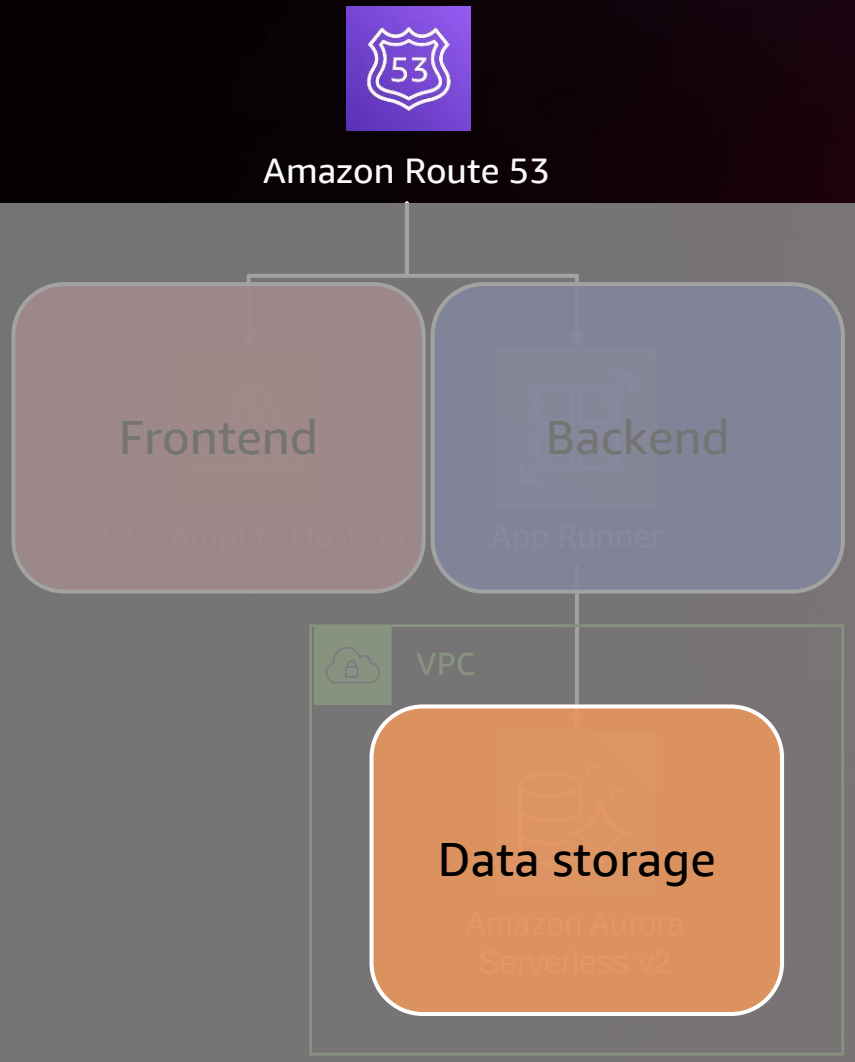
With data in hand you can now begin to tackle some of the most common pain points in scaling your application:

- Slow database queries
- Slow API requests
- Failures due to increased traffic
- Service-to-service communication



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

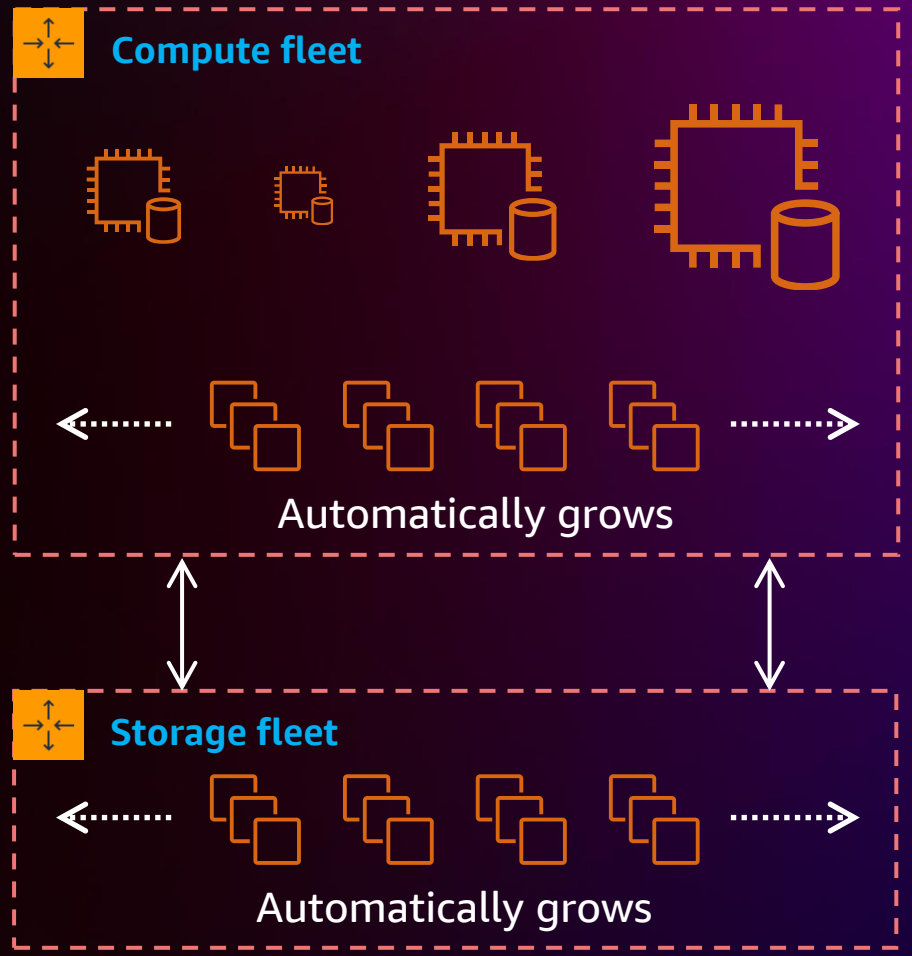
Let’s dive into the data tier



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: scaling

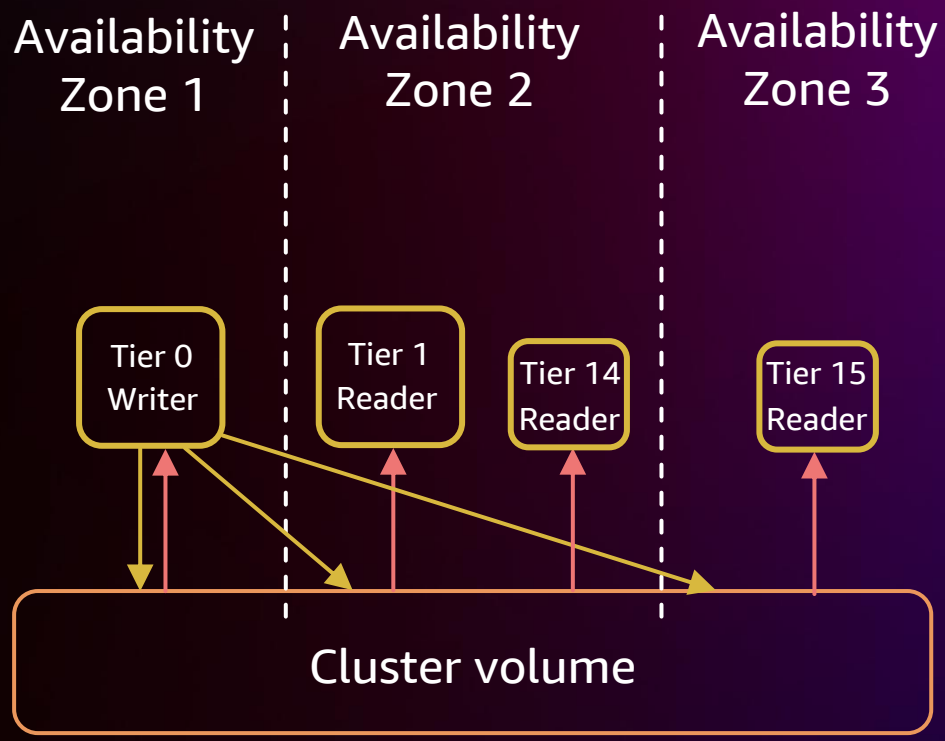
- Scales in place in under a second by adding more CPU and memory resources
- No impact due to scaling even when running hundreds of thousands of transactions
- Compute fleet continuously monitored and scaled horizontally for heat management
- Background movement of idling instances while preserving state (e.g., buffer pool, connections)
- Up to 15x faster scale downs



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: scaling

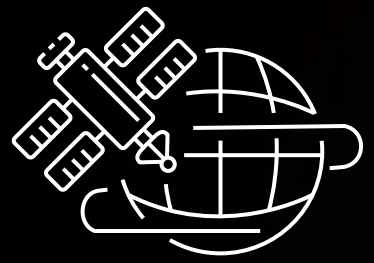
- Up to 15 read replicas act as failover targets
- All instances inherit capacity configuration from the cluster
- Tier 0 and 1 read replicas match the size of the primary instance
- Deploy across separate AZs
- Multi-AZ Aurora clusters supported by 99.99% uptime SLA



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon RDS Proxy

A FULLY MANAGED, HIGHLY AVAILABLE DATABASE PROXY FOR AMAZON RDS AND AMAZON AURORA



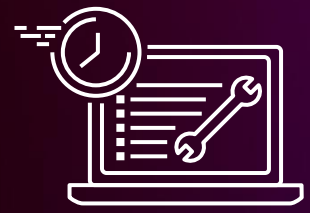
Pool and share DB connections for improved app scaling



Increase app availability and reduce DB failover times



Manage app data security with DB access controls



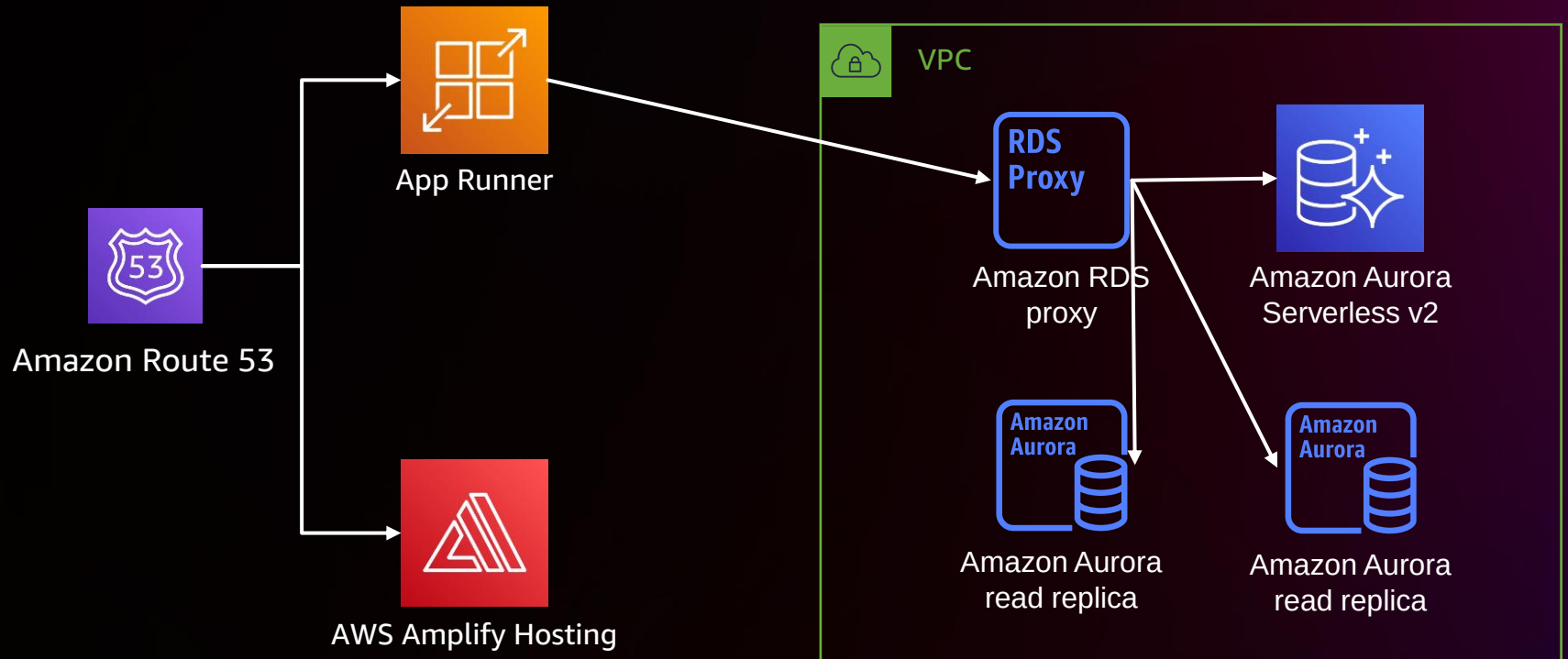
Fully managed DB proxy, compatible with your database

Amazon RDS Proxy supports Aurora Serverless v2, including mixed configurations with Aurora provisioned and serverless instances within a cluster

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

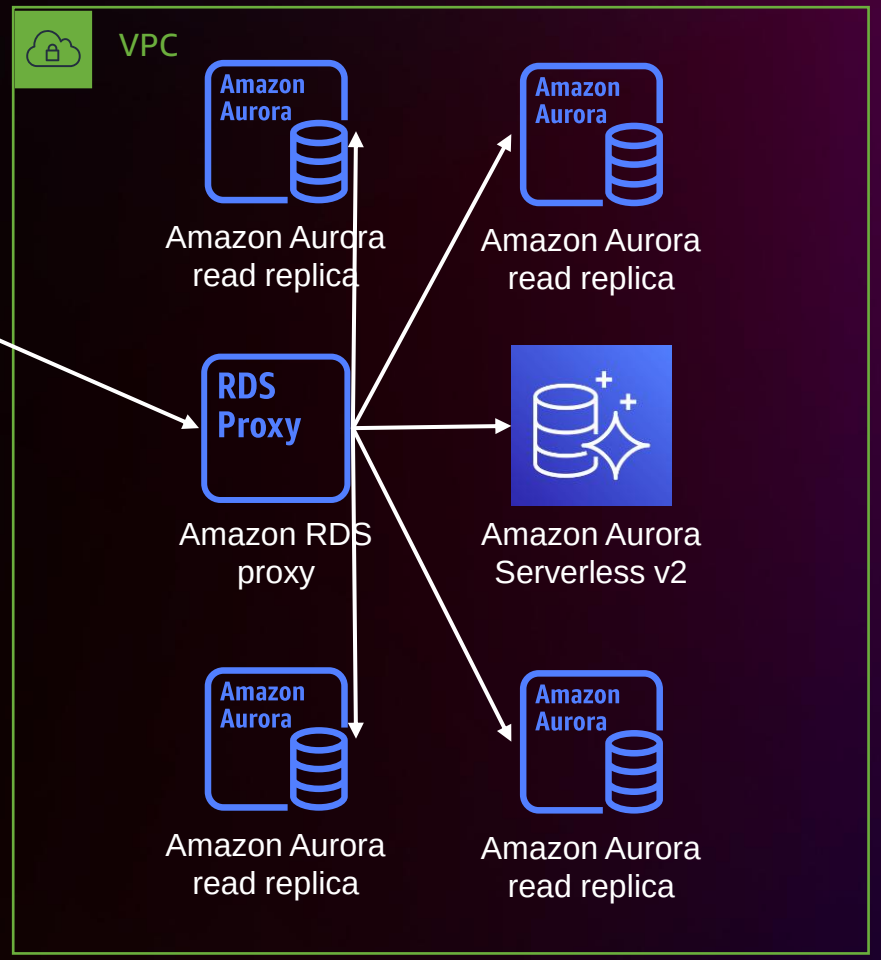
Scaling Aurora Serverless v2



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling Aurora Serverless v2

We can continue to scale by adding more read replicas and increasing the instance sizes of the nodes in the cluster



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“The best database queries are the ones you never need to make (often).”

Me

Today

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon ElastiCache



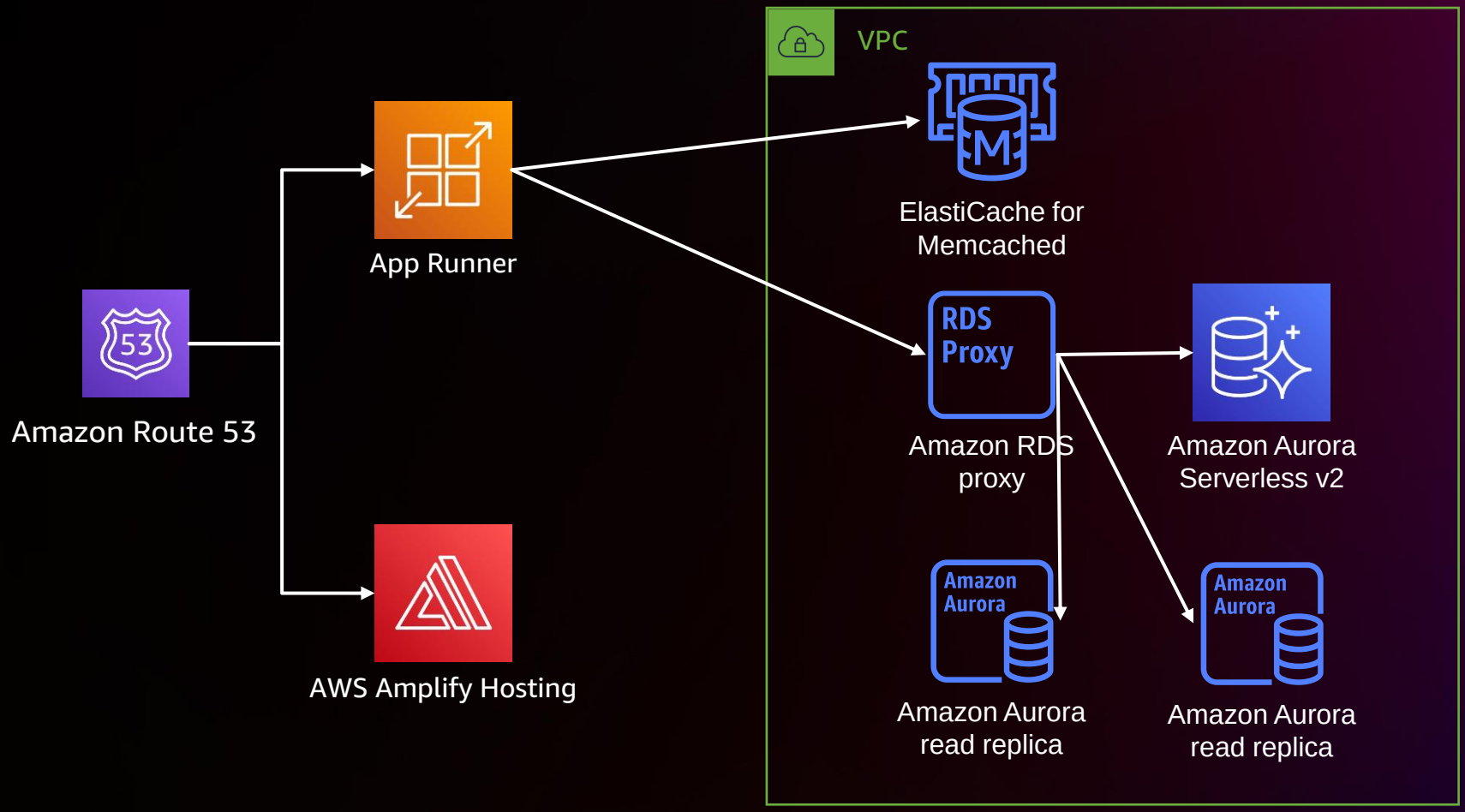
Amazon ElastiCache

- 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

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Add database caching with ElastiCache

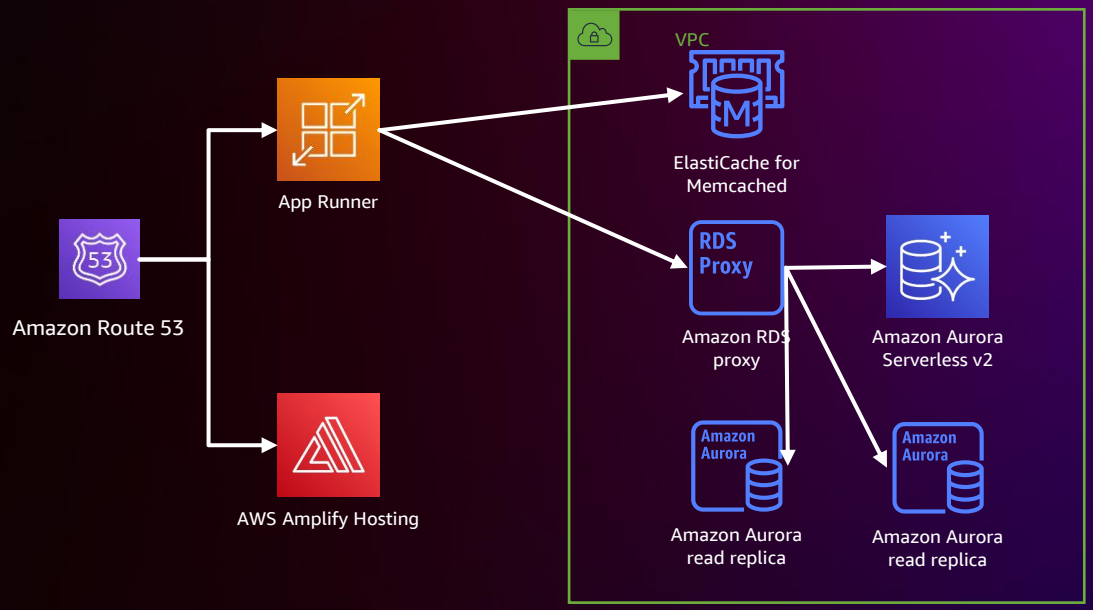


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling the data tier

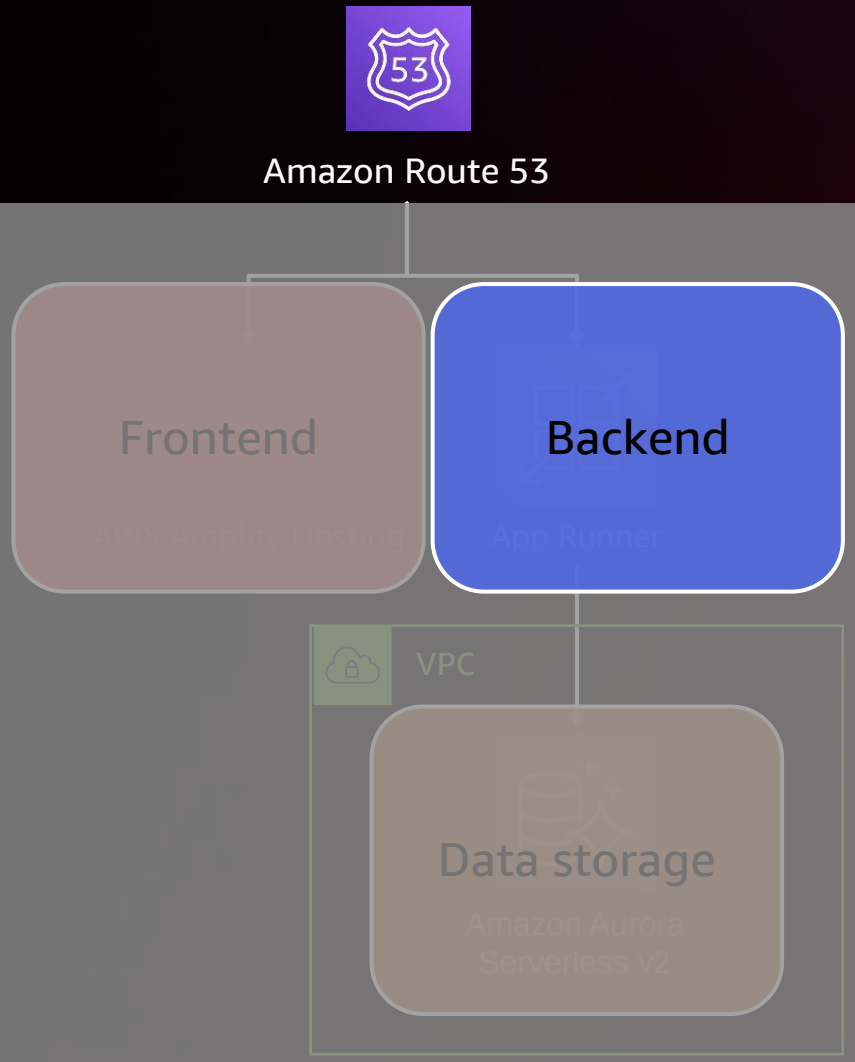
Three main methods for scaling our data tier:

- Increasing the size of the instance(s) used
- Adding read replicas and a proxy to help scale read queries
  - Typically minor application changes
- Using caches to remove queries from even needing to be made
  - requires more significant application changes and new logic to handle



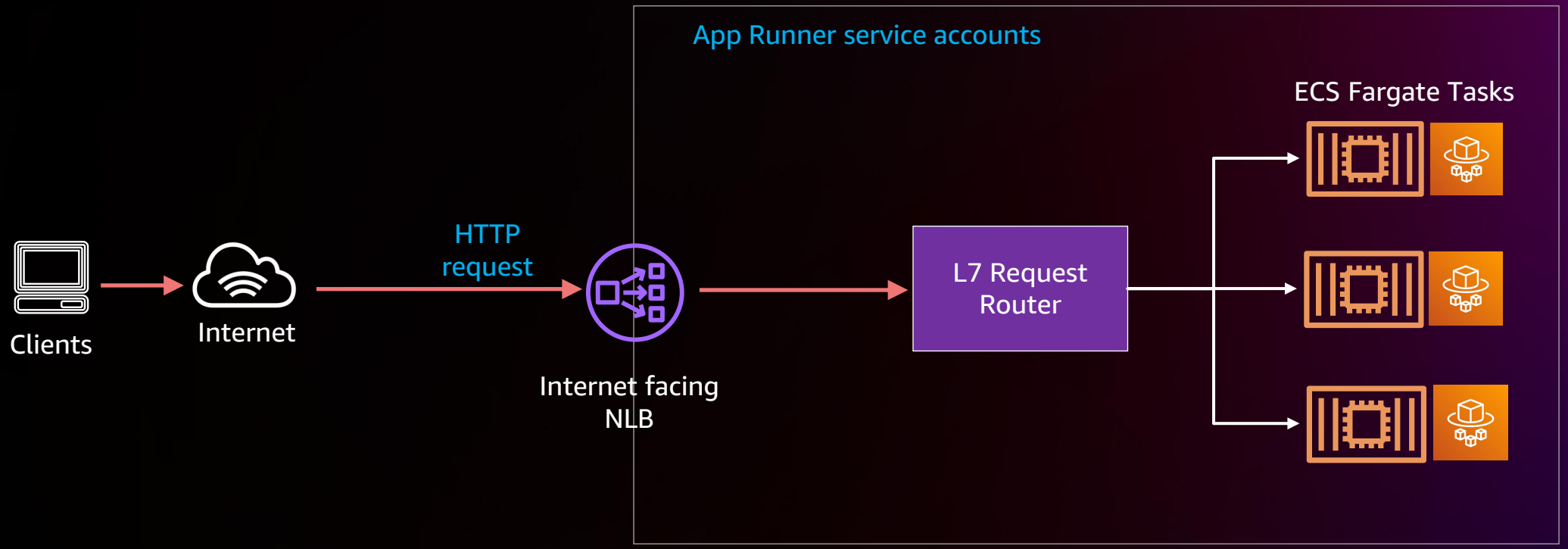
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s dive into the backend tier



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

App Runner: Under the hood



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

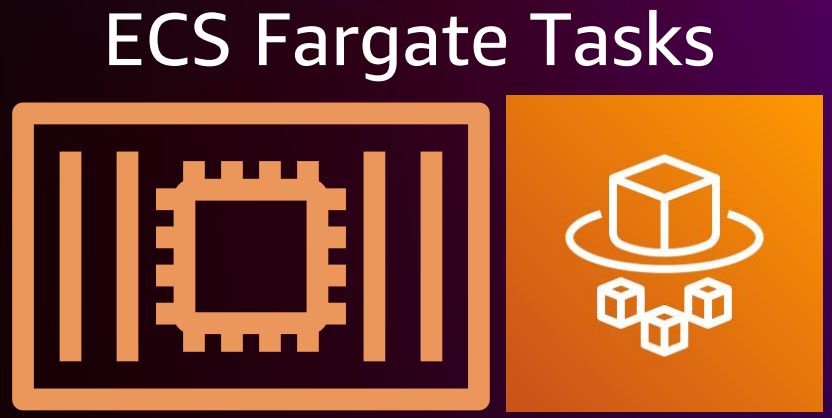
App Runner: scaling instance sizes

Individual instances are configured for a mix of CPU and Memory (see table)

Maximum number of concurrent requests per instance: 200

Maximum number of instances per service: 25

Current limit of 5000 concurrent requests per App Runner service (your deployed application)



CPU	Memory
1 vCPU	2GB, 3GB, 4GB
2 vCPU	4GB

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

App Runner: scaling number of instances

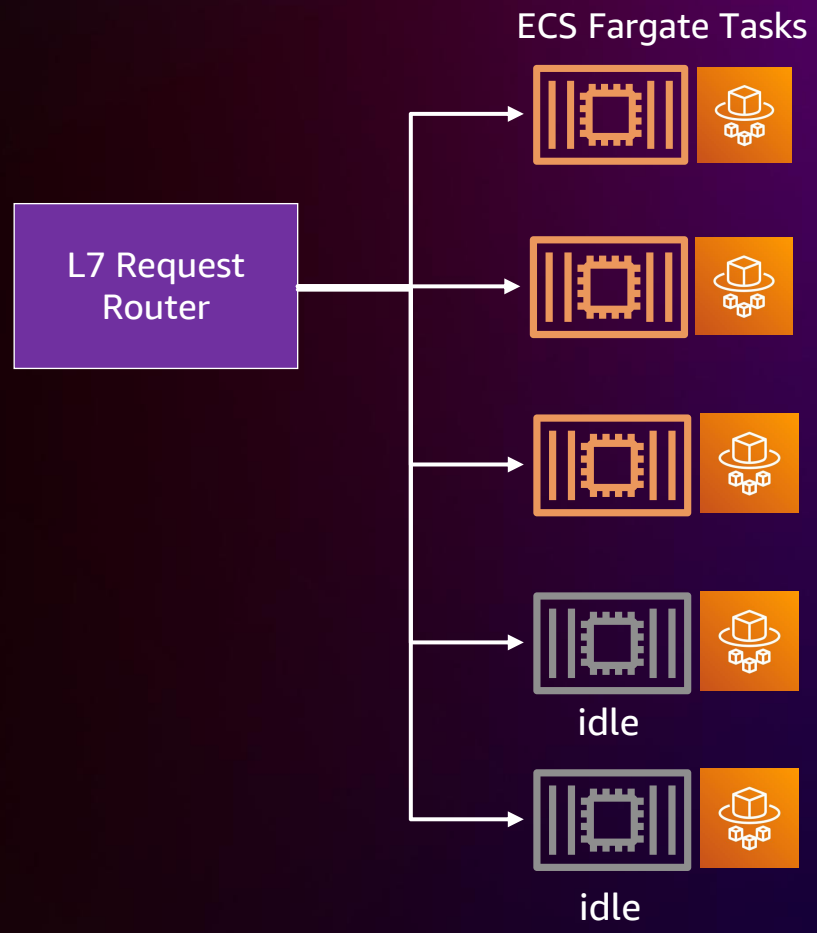
Scales by concurrent requests sent to an “instance”

Instances automatically added to the request router

Instances not serving requests can go idle to save costs

Can specify a max for cost control e.g., dev environments

When there are no requests, App Runner scales down to 1 (default), and keeps memory provisioned to minimize cold start latency



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

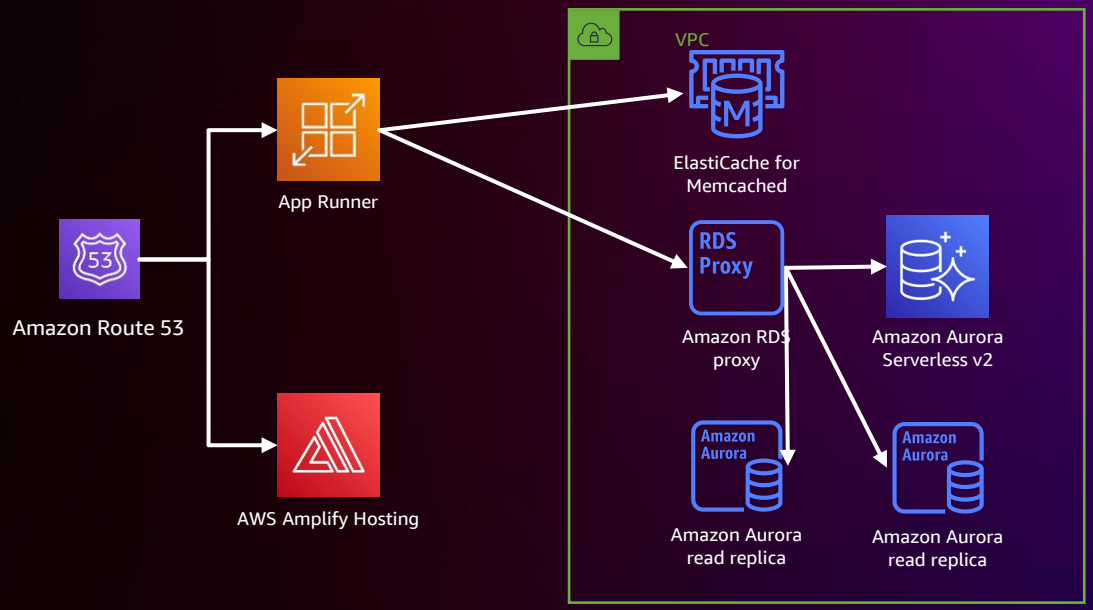
Scaling the backend tier

With basic configuration you can hit 5000 concurrent requests.

- For context, at 2 seconds per request you could perform ~150K requests per minute

Application performance tuning remains key

- Reducing slow database queries
- Profiling code with CodeGuru and similar tools for costly/slow logic
- Caching in the edge/application client where possible



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >100,000

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

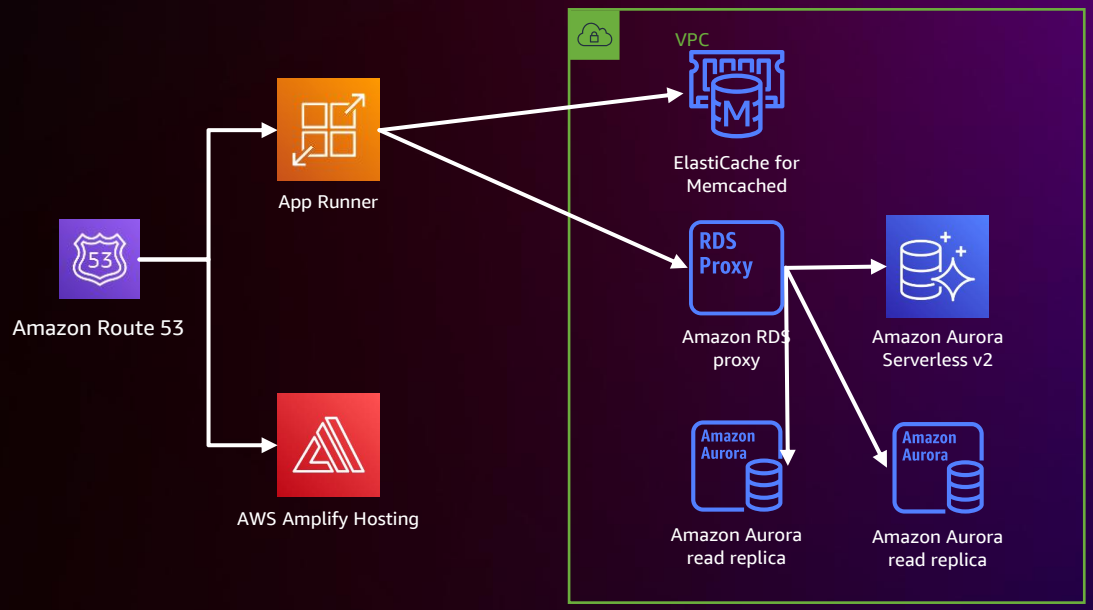
Users: >1,000,000

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >1,000,000

At some point you will outgrow the patterns we've discussed so far:

- Need more scale than App Runner can support today in single service
- Database writes begin to become a bottleneck
- Application complexity/feature growth begins to necessitate new infrastructure needs



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Going the microservices route

Moving to a service-oriented or microservices based architecture is a refactor that requires deep planning across all layers.

- Start with with the easiest to cut away features/capabilities that don’t involve too many cross-function ties
  - Data domain mapping
  - Business function mapping
- Good time to evaluate other compute technologies for specific needs
- Will need to think about how to “glue” everything together

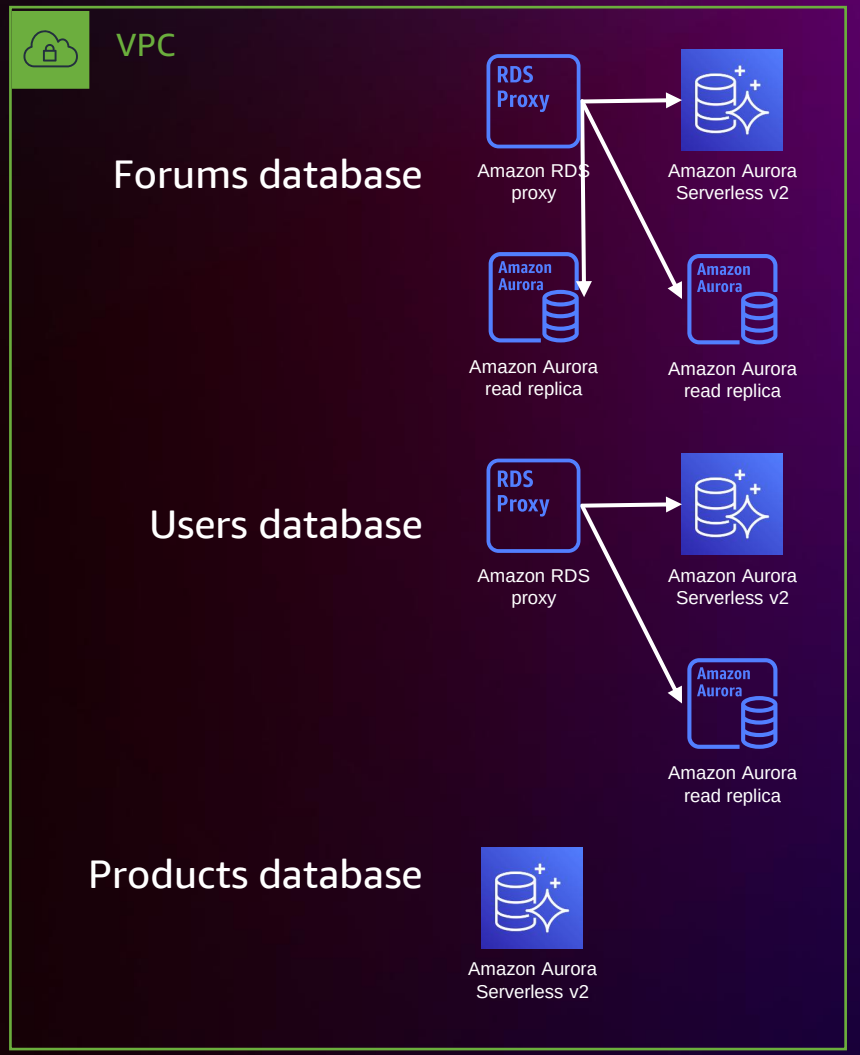


aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Database federation

- Split up databases by function/purpose
  - Repeat scaling patterns discussed previously for each
  - Can mix and match to align with specific business needs
- Harder to do cross-function queries
- Essentially delays sharding/NoSQL
- Won't help with single huge queries or tables with incredible amounts of data
  - Often the result of misaligned workloads to the technology. E.g., Data warehouse workloads on RDBMSes



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Shifting functionality to NoSQL



Amazon DynamoDB

Leverage managed services such as DynamoDB

- Supports massive scale with consistent low latency

Example use cases

- “Hot” tables
- Metadata/lookup tables
- Leaderboards/scoring
- Temporary data needs (cart data)

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

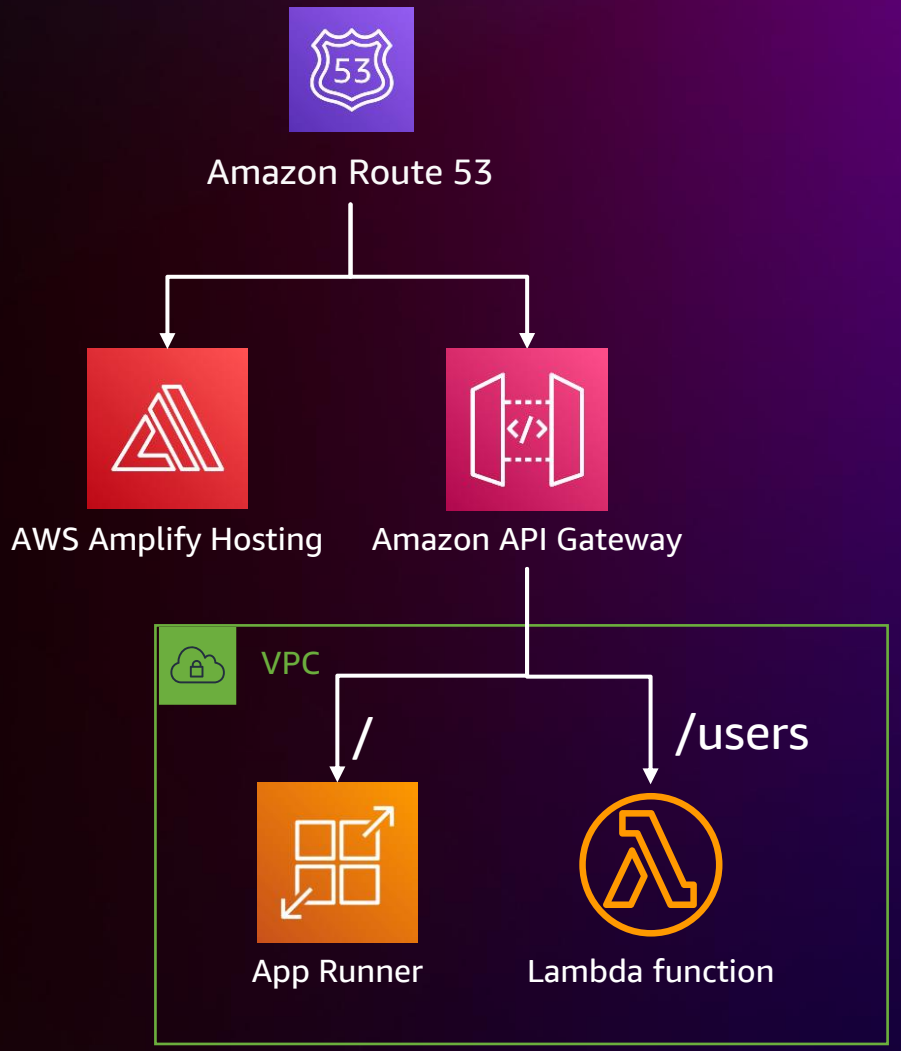
Breaking up the backend tier

Breaking up the backend can mirror the data tier:

- Split the application into new federated services aligned to data patterns
- Revisit which managed compute best aligns

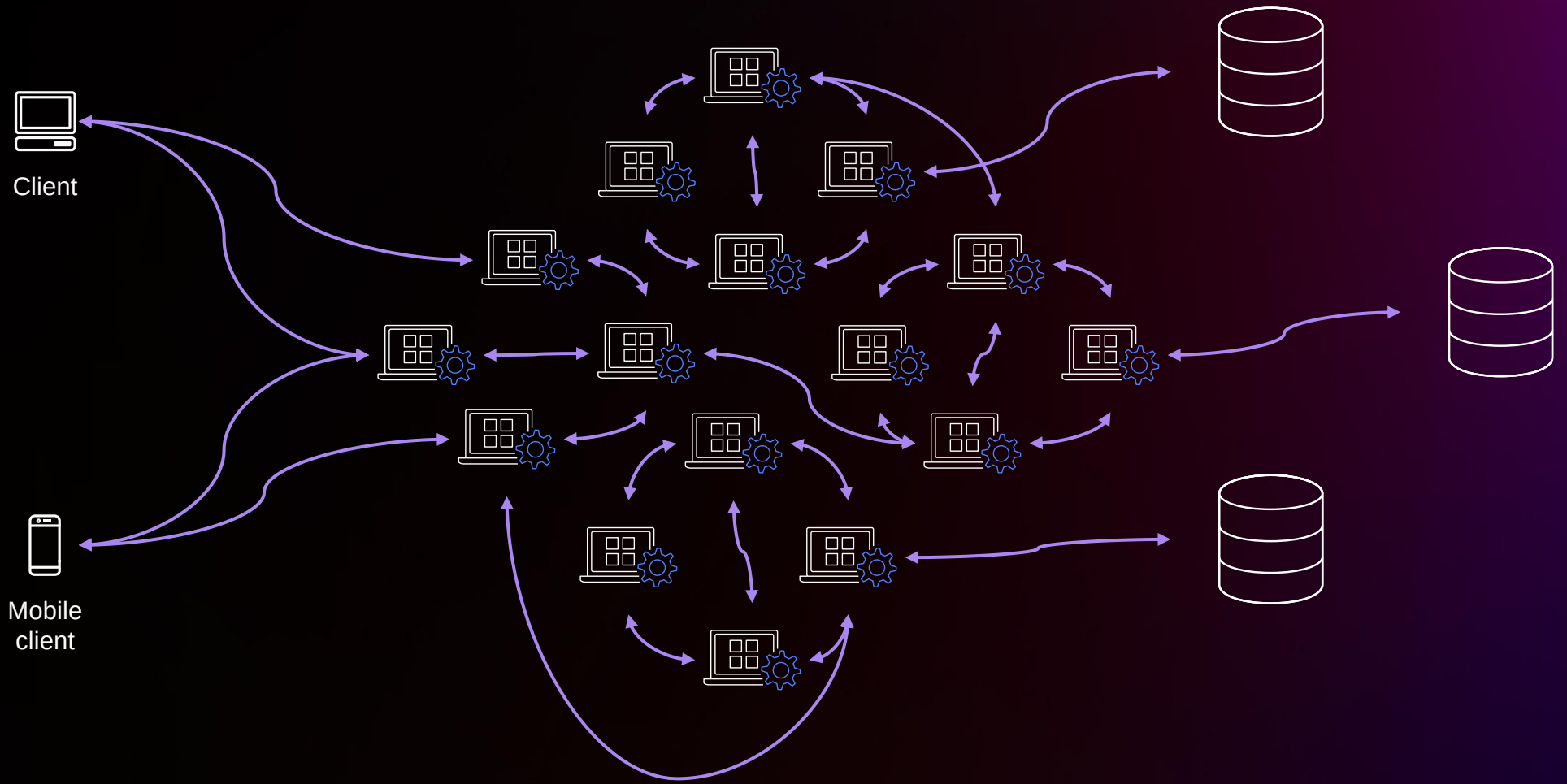
Explore what business logic can move to internal services

- Moving from sync to async
- Leveraging queues, topics, buses, and streams to build event-driven architectures



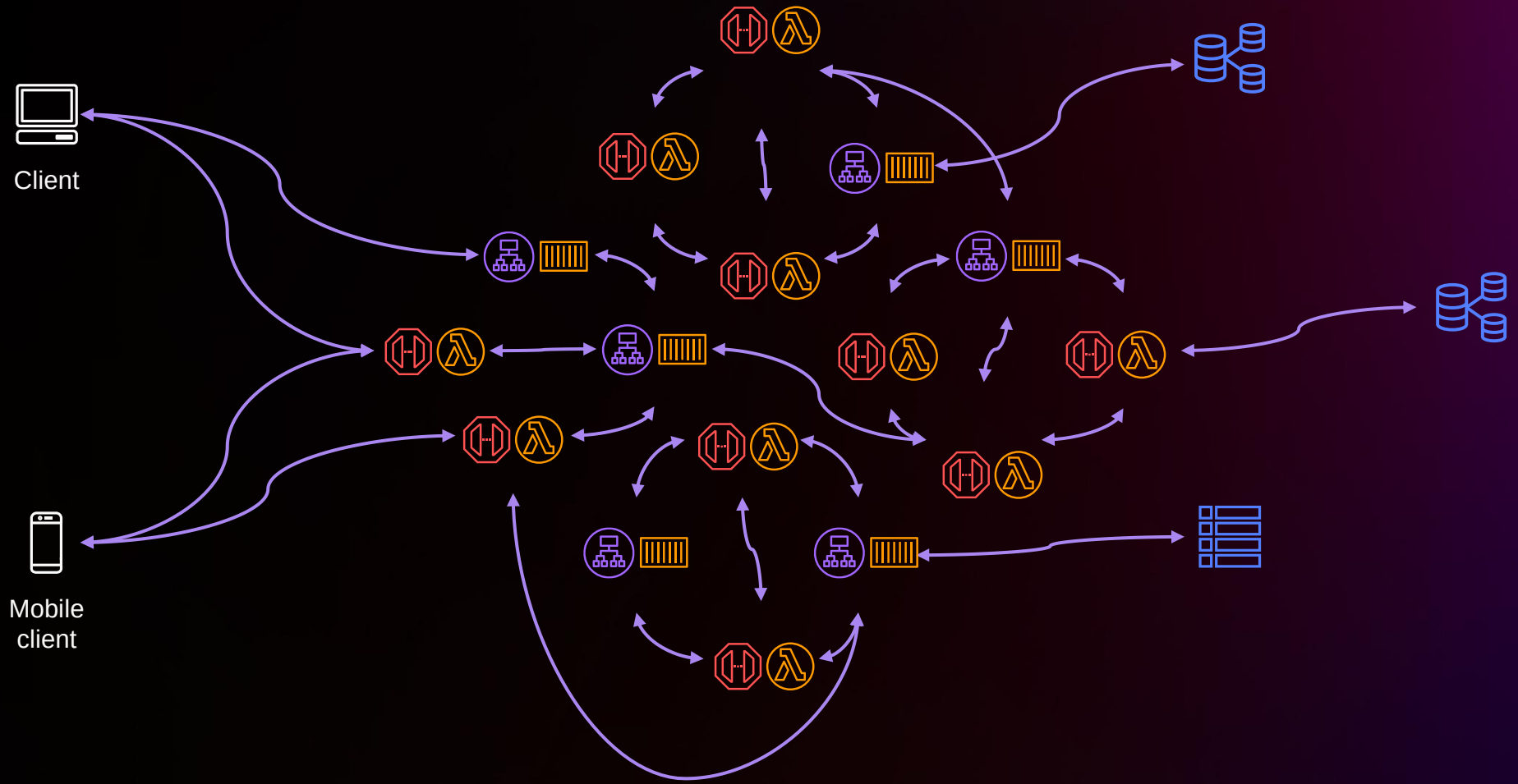
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The microservices architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The microservices architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

10 million+ users

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10 million

- More distribution of features/functionality across internal microservices
- Deeply analyze your entire stack's performance and continue to find areas to improve
- Possibly start to build on self-managed compute
- Evaluating how to improve caching at all tiers



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

To infinity . . .

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

FIN/ACK – (in closing)

We’re in such a better place in being able to scale out of the box than we were even 9 years ago

- Built-in scalability for many tiers that goes far toward meeting *most* needs
- There are more resources available throughout the stack: bandwidth, CPU, memory, storage

The bulk of scaling wins come from doing less

- Caching at both the edge and origin
- Reducing scope of database queries and data processed

Evaluate refactoring cautiously

- Federating data can still be an easy way to win
- Look for “best fit” technologies based on need

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.