
AWS
re:Invent
NOV. 27 – DEC. 1, 2023 | LAS VEGAS, NV

ARC206

Scaling on AWS for the first 10 million users

Chris Munns
Startup Tech Lead/Advisor
Amazon Web Services

Skye Hart
Manager, Startup SA
Amazon Web Services

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

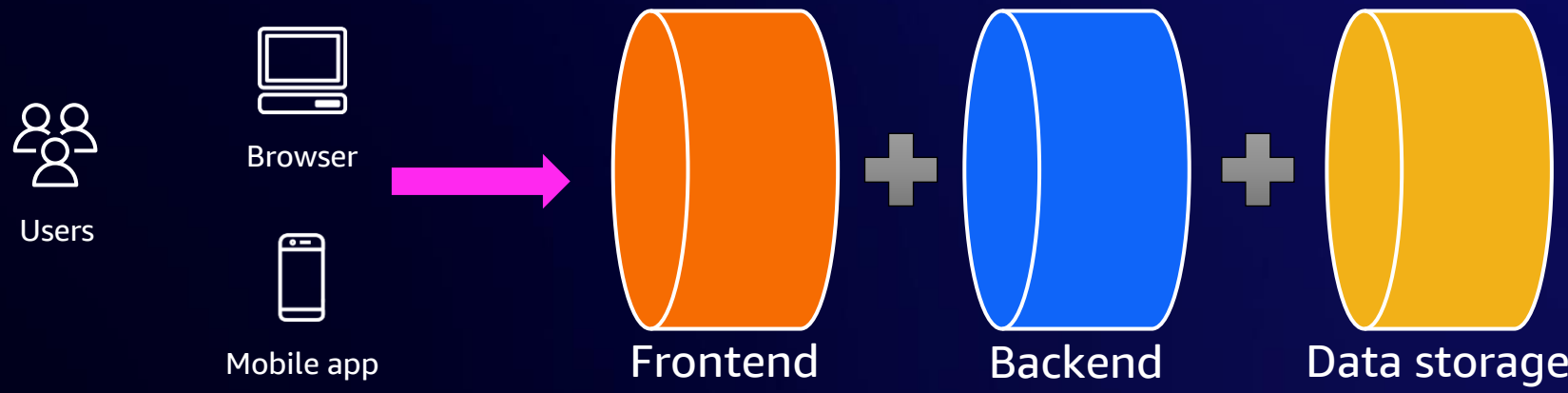
We start with the goal of launching a new app

© 2023, 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



aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Acknowledging current technology trends

- Modern frontend frameworks built using JavaScript or derivatives
- Full-stack frameworks that more closely integrate front and backend development
- Movement away from self-managed/DIY infrastructure to managed services
- Potential for rapid scale (measured in hours, not days)

aws

© 2023, 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Build

Measure

Learn

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

So let’s start from

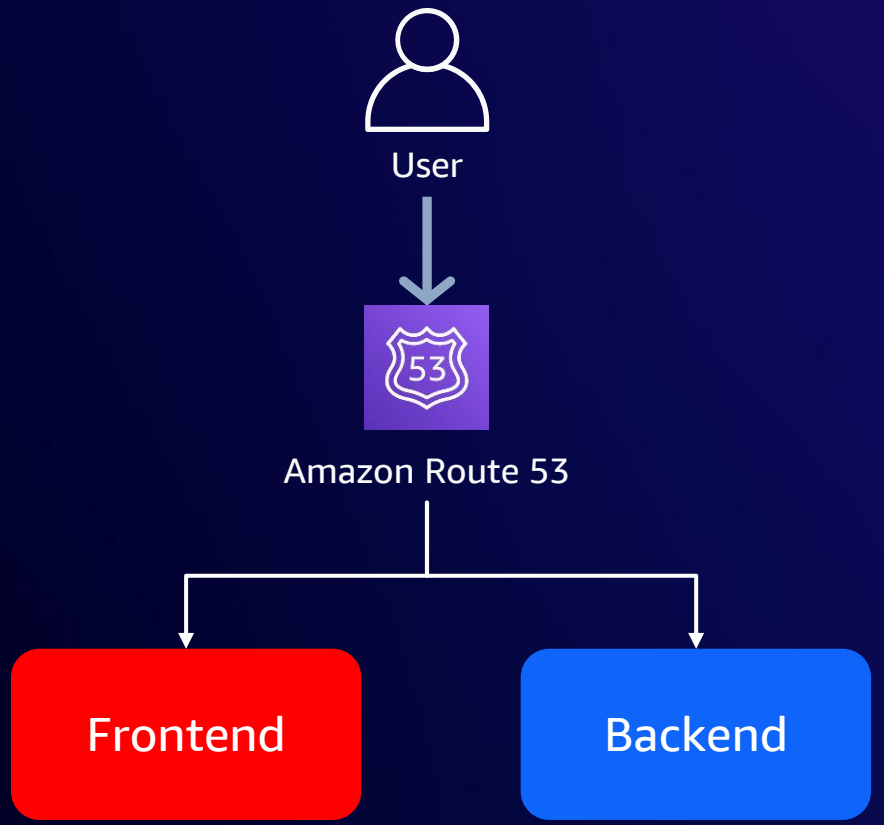
Day 1

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >1

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >1

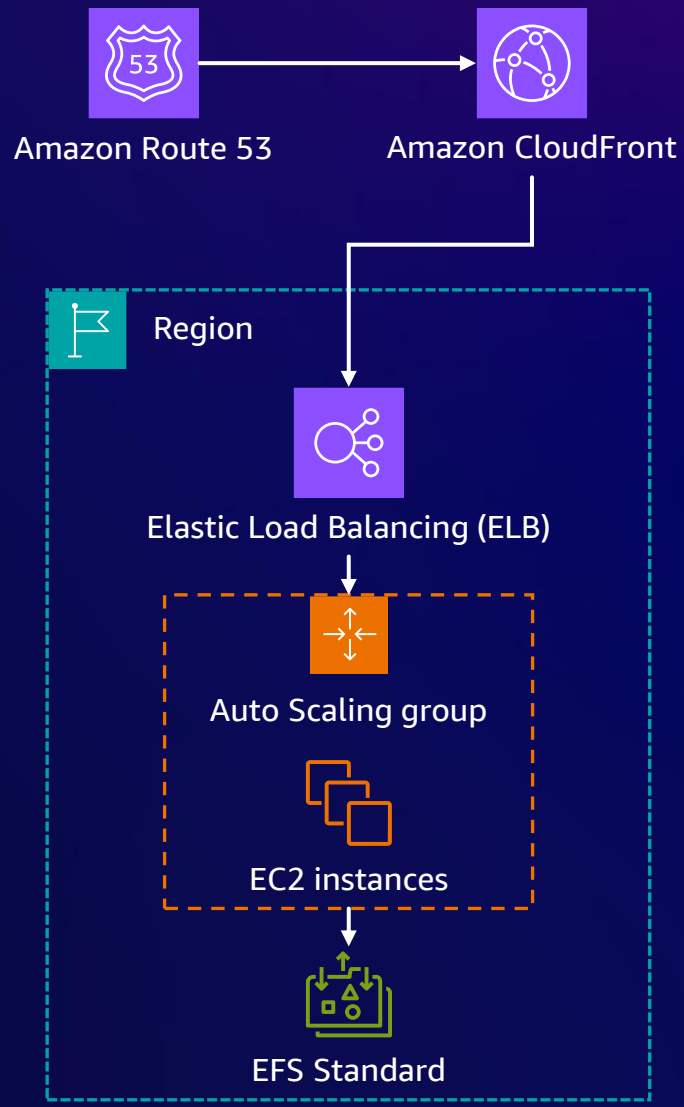


aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users >1: Traditional frontend hosting

Traditional frontend hosting would have you serve your frontend content (HTML, CSS, JavaScript, images, and so on) off of a simple web-serving stack. That stack would minimally be composed of:

- Hosting tier for the webserver app (Nginx, Apache, and so on)
  - Optionally, a shared storage layer
- A load balancer
- A CDN for edge caching



aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

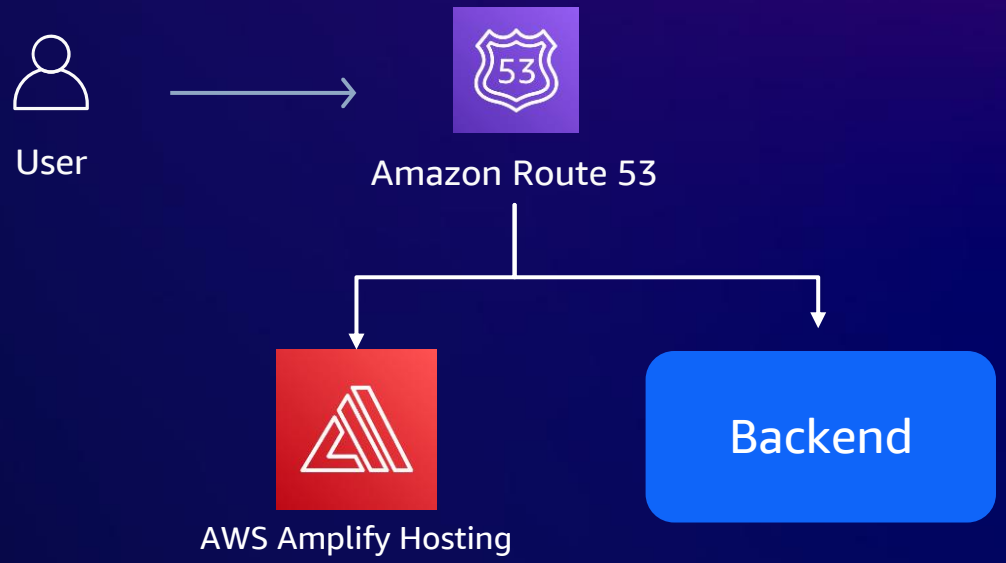
Users >1: With a modern frontend

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)



© 2023, 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

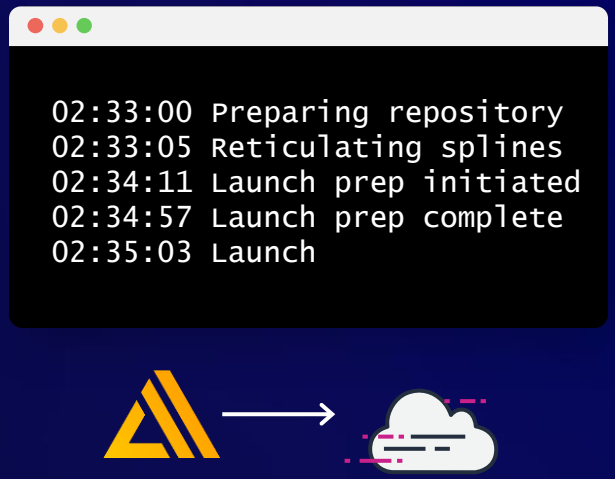
AWS CodeCommit

2

Configure build settings

3

Deploy your app

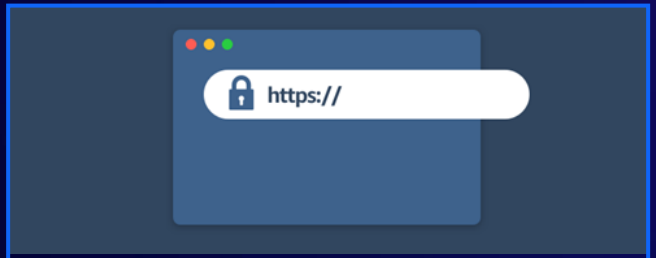


© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amplify Hosting features

FEATURES FOR HOSTING MODERN WEB APPLICATIONS

Globally available



Easy custom domain setup

Simplified continuous workflows

Feature branch deployments

Atomic deployments

Password protection

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amplify Hosting – Supporting modern frameworks

Client-side rendered (single-page application)

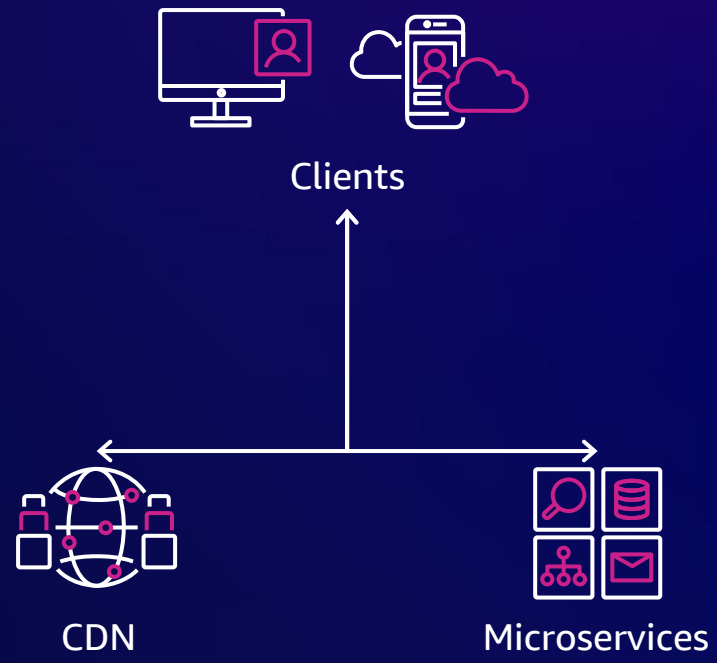
- Frontend application loaded as JavaScript and runs in the client browser
- JavaScript files containing application logic, UI, and communication with backend
- Popular frameworks such as React, Angular, and Vue

Server-side rendered (SSR)

- Rendering on the server before sending page to browser
- Data fetched from a database or CMS
- Ideal for applications that have personalized content for each user
- Popular frameworks include Next, Nuxt, and Gatsby

Static site generators (SSGs)

- Content generated at the build time
- Ideal for sites where content does not need to be highly personalized
- Typically used in concert with a headless CMS and CDN
- Popular solutions such as Gatsby, Eleventy, Hugo, VuePress, and Jekyll

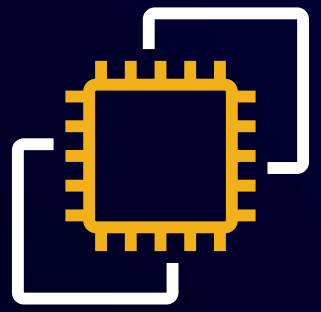


aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What about the backend?

© 2023, 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 instances

AWS Lambda

Serverless compute for stateless code execution in response to triggers

aws

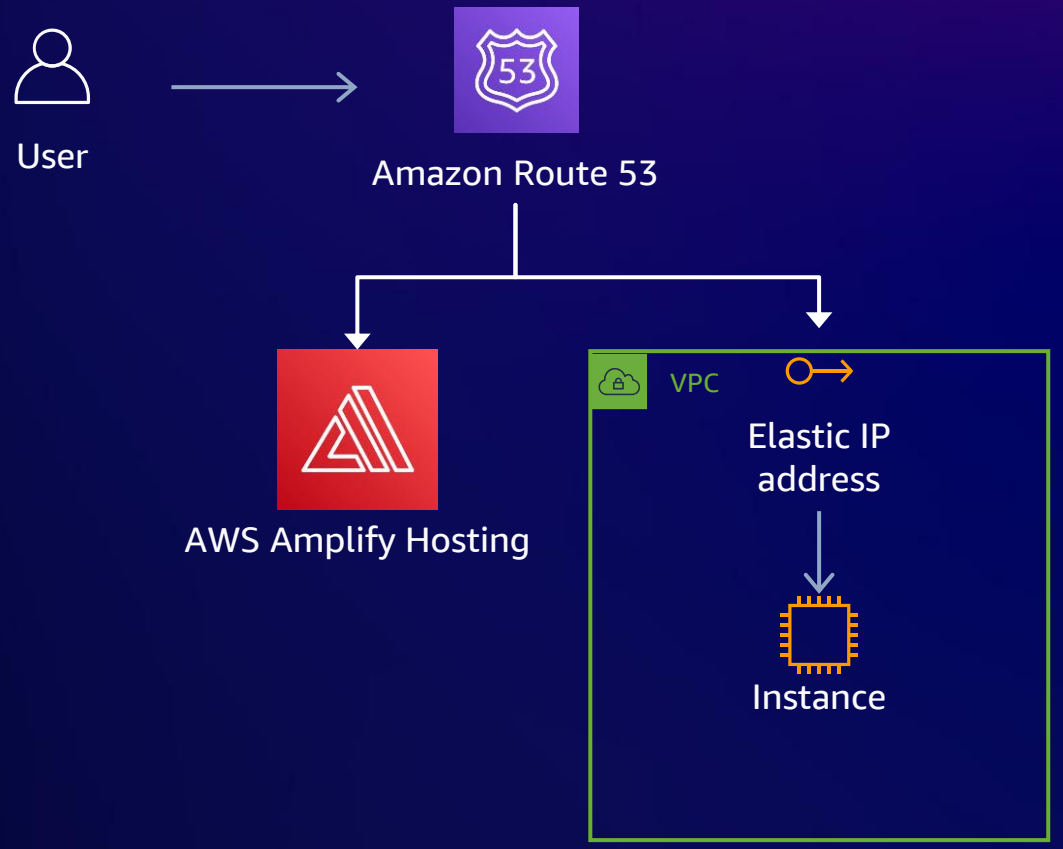
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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?



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We can go far with this, but . . .

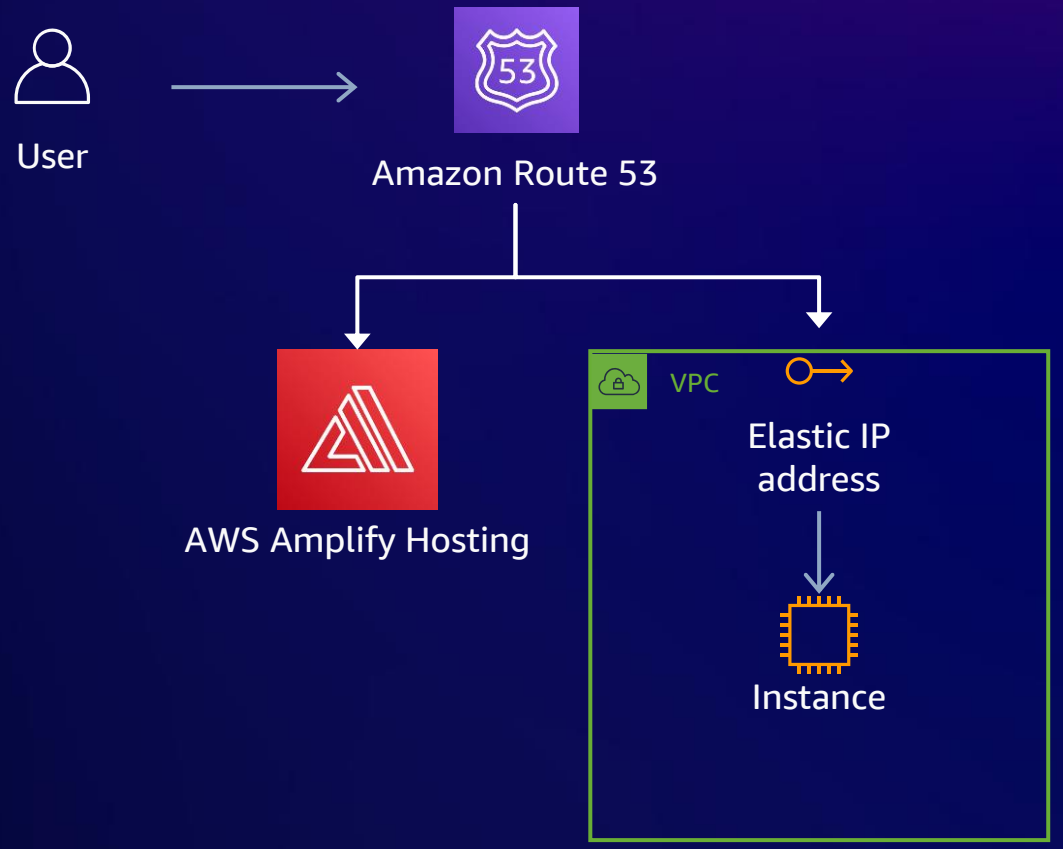
The downsides of starting out this way become apparent quickly

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



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

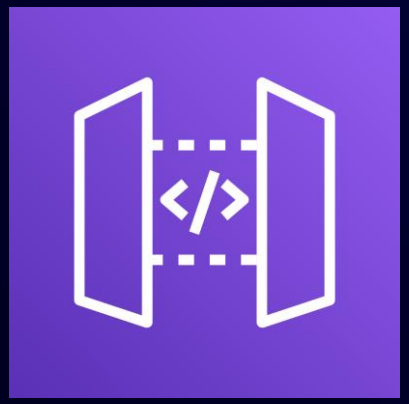
Evaluating managed compute on AWS

	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
Amazon ECS/ Amazon 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
Amazon EC2 Instance-based compute 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

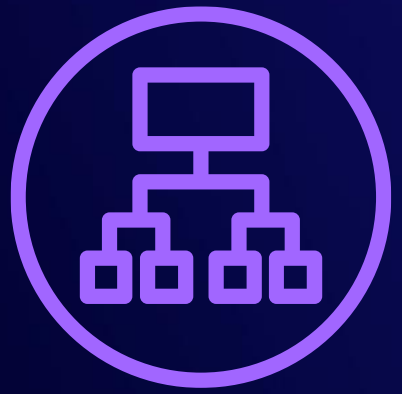
aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Exposing business logic to the frontend

THREE OPTIONS FOR EXPOSING AN API



Amazon API Gateway



Application Load Balancer

AWS AppSync

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Picking an API fronting service cheat sheet

Complex API with multiple data sources or very unique queries against data?

- AWS AppSync

WebSockets?

- Amazon API Gateway

Need transforms, throttling, usage tiers, flexible auth?

- Amazon API Gateway

Single API action/method, billions+ of requests per day?

- Application Load Balancer

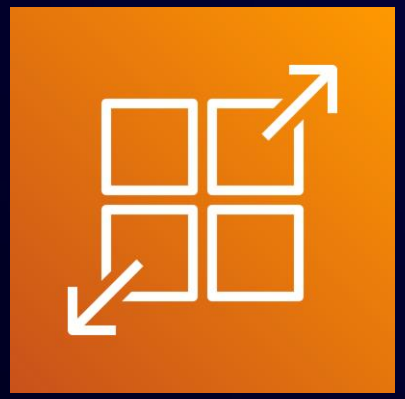
Typical API, with millions of requests per month?

- Amazon API Gateway

aws

© 2023, 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

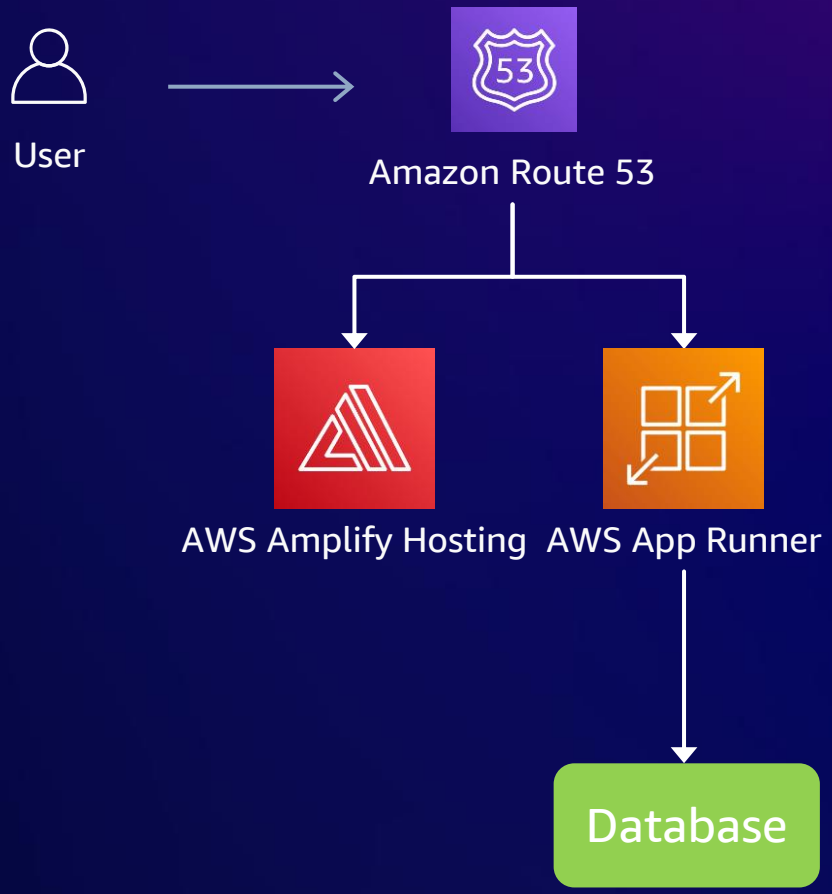
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

To NoSQL or not to NoSQL?

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Start with SQL databases

© 2023, 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aha!

You said, “massive amounts of data.”

That’s me.

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Multiple terabytes of data in year 1?

Incredibly data-intensive workload?

Okay!

You might need NoSQL

© 2023, 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

But this isn’t most of you. So . . .

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Start with SQL databases

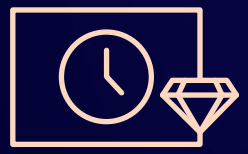
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Aurora

UNPARALLELED HIGH PERFORMANCE AND AVAILABILITY AT GLOBAL SCALE WITH FULL MYSQL AND POSTGRESQL COMPATIBILITY AT 1/10TH THE COST OF COMMERCIAL DATABASES

Performance & scalability

- 5x throughput of standard MySQL and 3x of standard PostgreSQL
- Scale out up to 15 read replicas
- Decoupled storage and compute-enabling cost optimization
- Fast database cloning
- Distributed, dynamically scaling storage subsystem

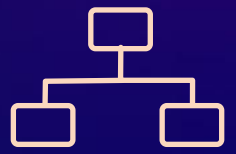


Availability & durability

- 99.99% availability with Multi-AZ
- Data is durable across 3 AZs within a Region (*Customers only pay for 1 copy)*
- Automatic, continuous, incremental backups with point-in-time recovery (PITR)
- Failovers in < 10 seconds
- Fault-tolerant, self-healing, auto-scaling storage
- Global database for disaster recovery

Highly secure

- Network isolation
- Encryption at rest/in transit
- Supports multiple secure authentication mechanisms and audit controls



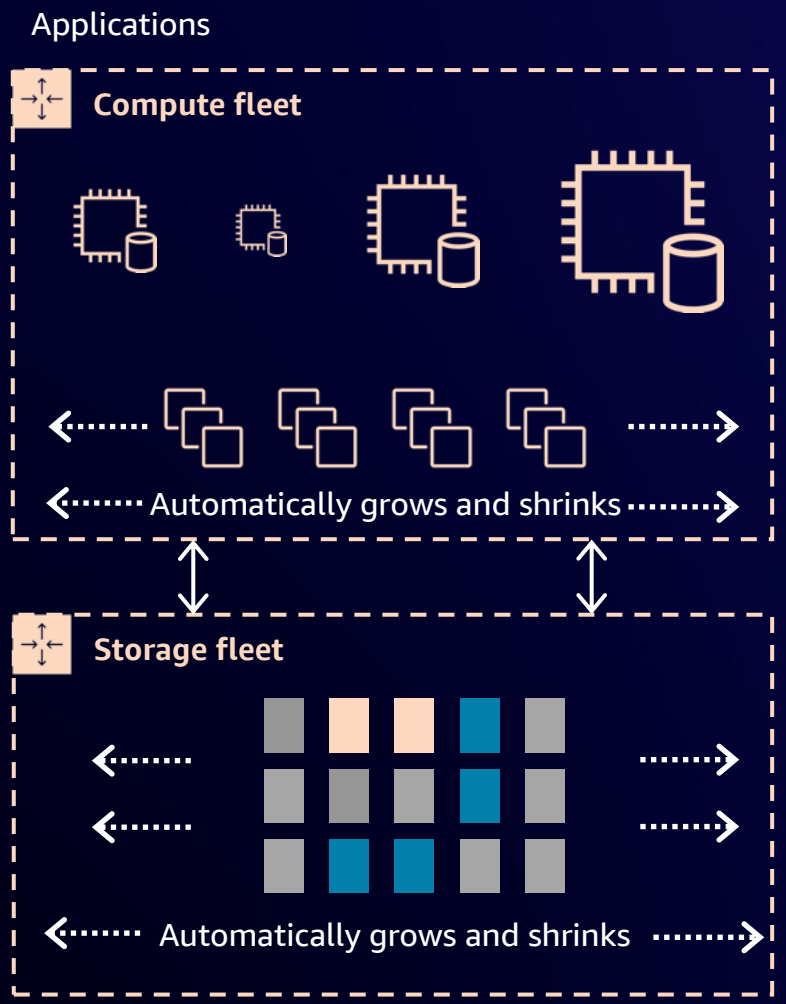
Fully managed

- Automates time-consuming management of administration tasks like hardware provisioning, database setup, patching, and backups
- Serverless configuration options

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Aurora Serverless v2



- Scales in fine-grained increments to provide just the right amount of database capacity in response to the demands of your application's events
- Scales instantly in a fraction of a second even for the most demanding applications
- Up to 90% cost savings when compared to provisioning for peak load
- Full breadth of Aurora capabilities, including parallel query, global database, read replicas, and multi-AZ support

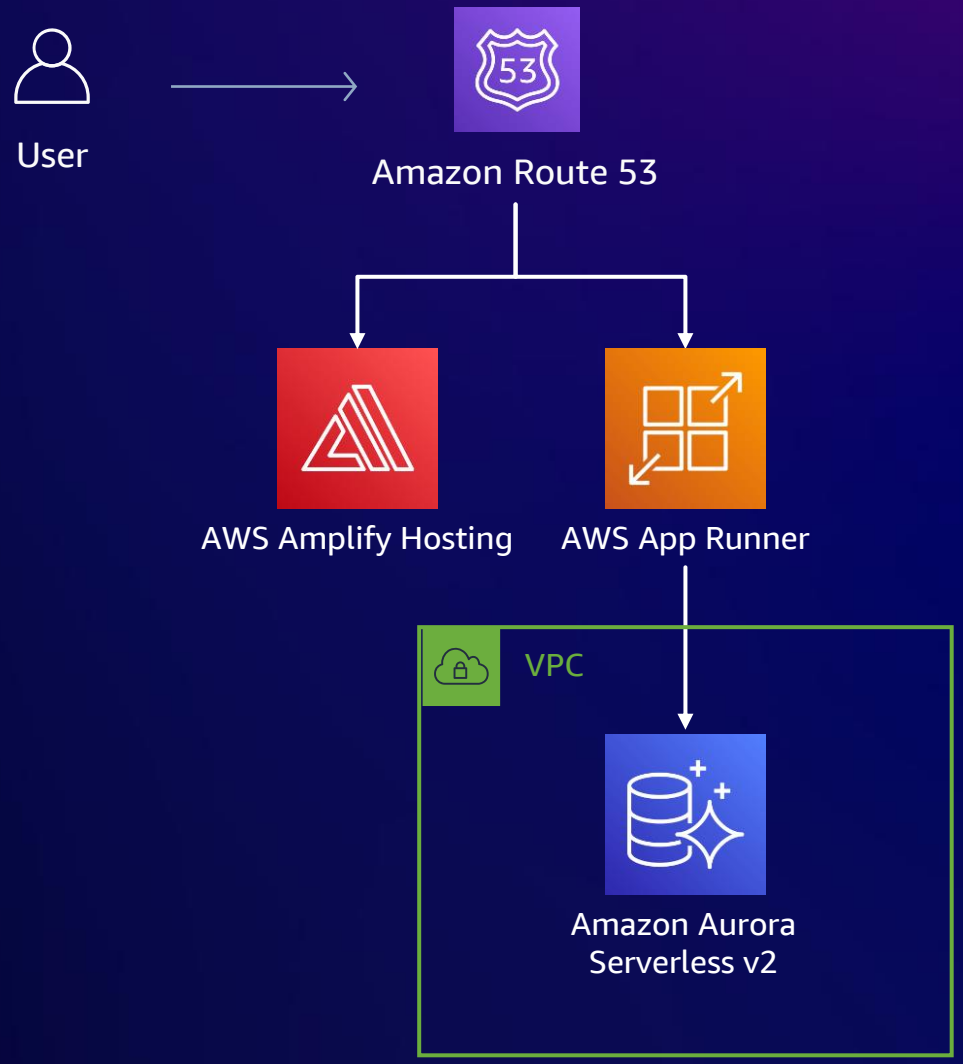
aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users >1:

By leveraging 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!



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >100

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >1000

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000

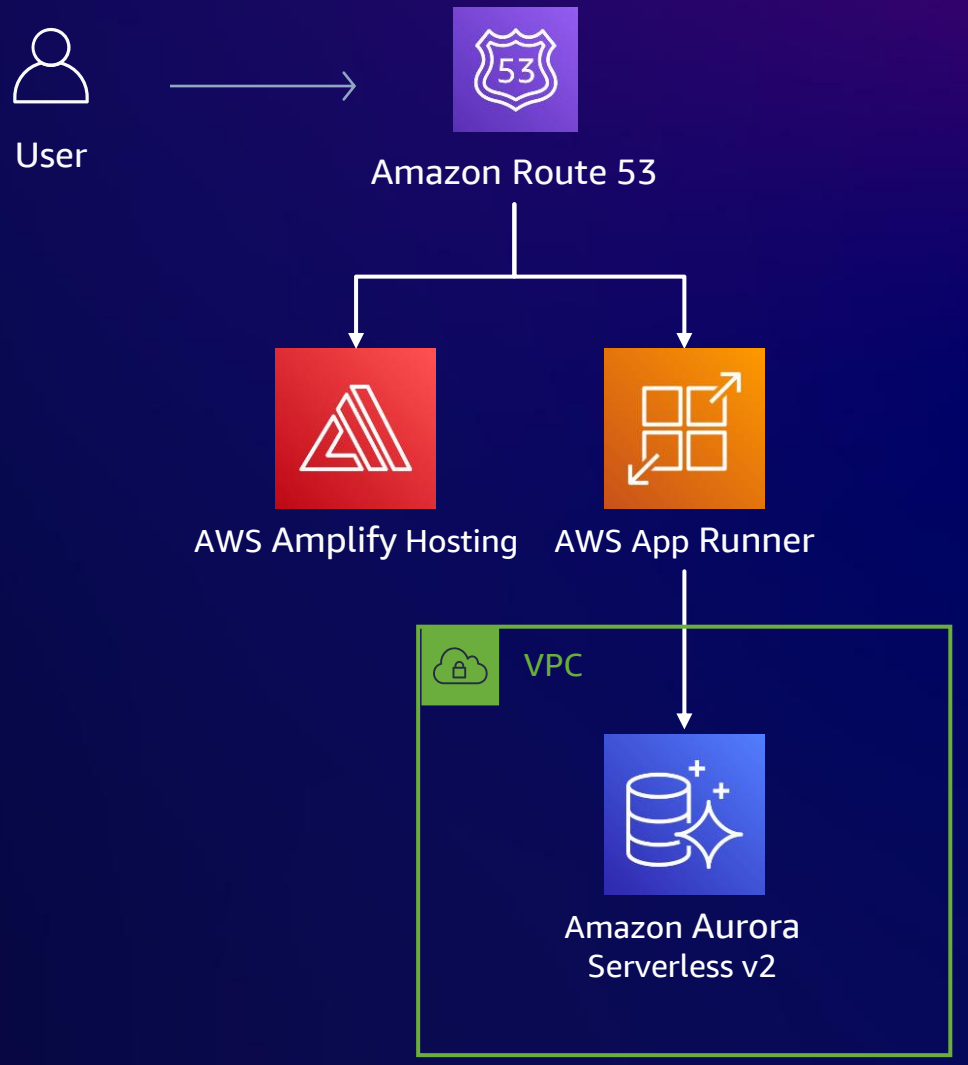
maybe?

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

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

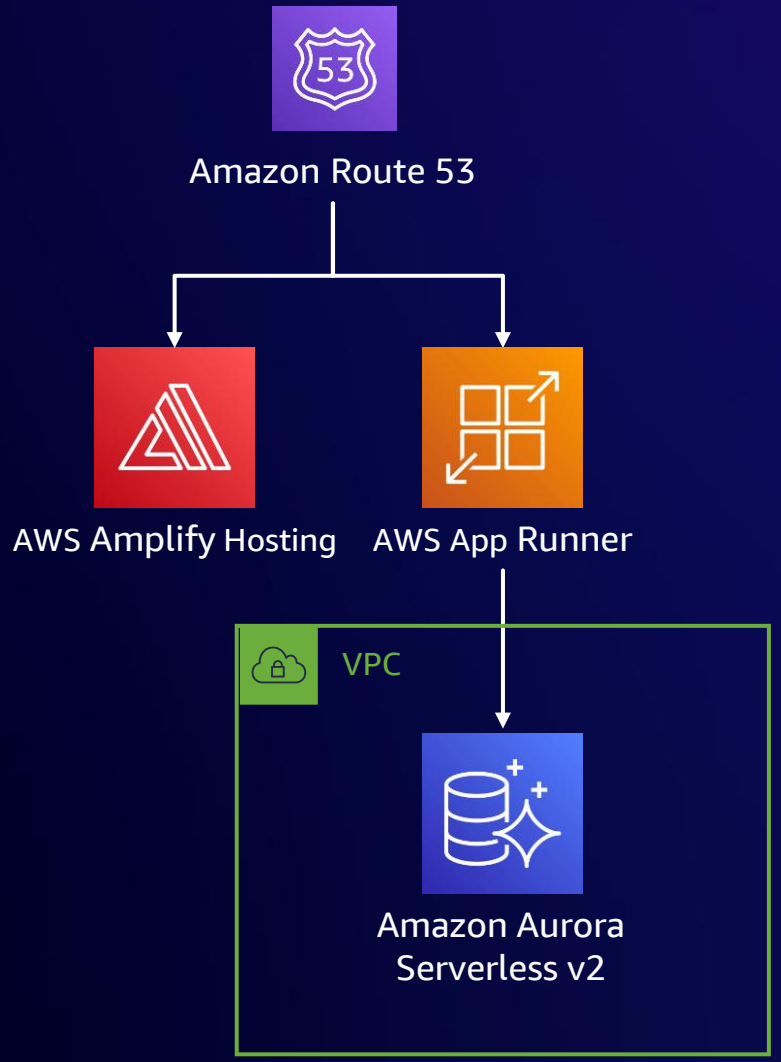
The current stack will scale incredibly far, but the scaling of single tier/monolithic applications can sometimes only go so far. You'll eventually run into issues common in most architectures:

- 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



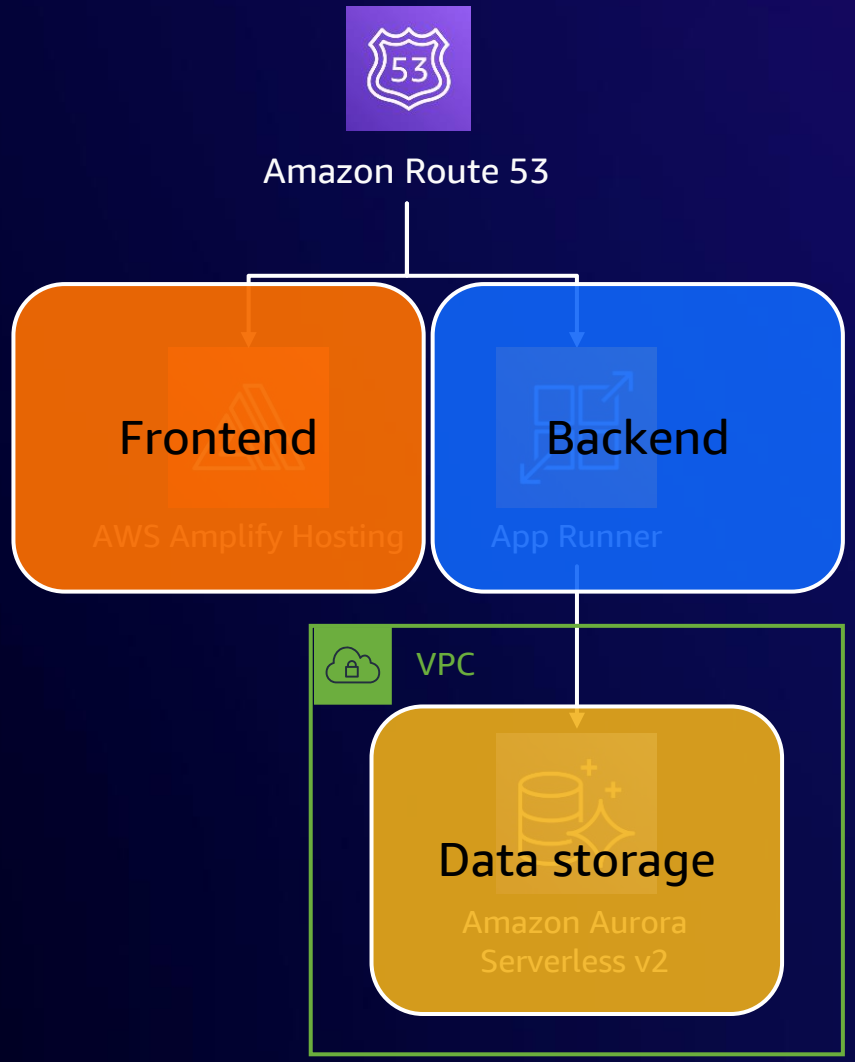
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s learn more



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s learn more



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Before we go too much further

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

We can’t tune what we aren’t measuring

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS services for observability

Amazon CloudWatch

- Dashboards
- Logs
- Metrics
- Alarms
- Events
- Synthetic Canaries
- Real User Monitoring (RUM)

AWS X-Ray

- Traces
- Analytics
- Service map

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Leverage machine learning (ML) to assist you

© 2023, 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

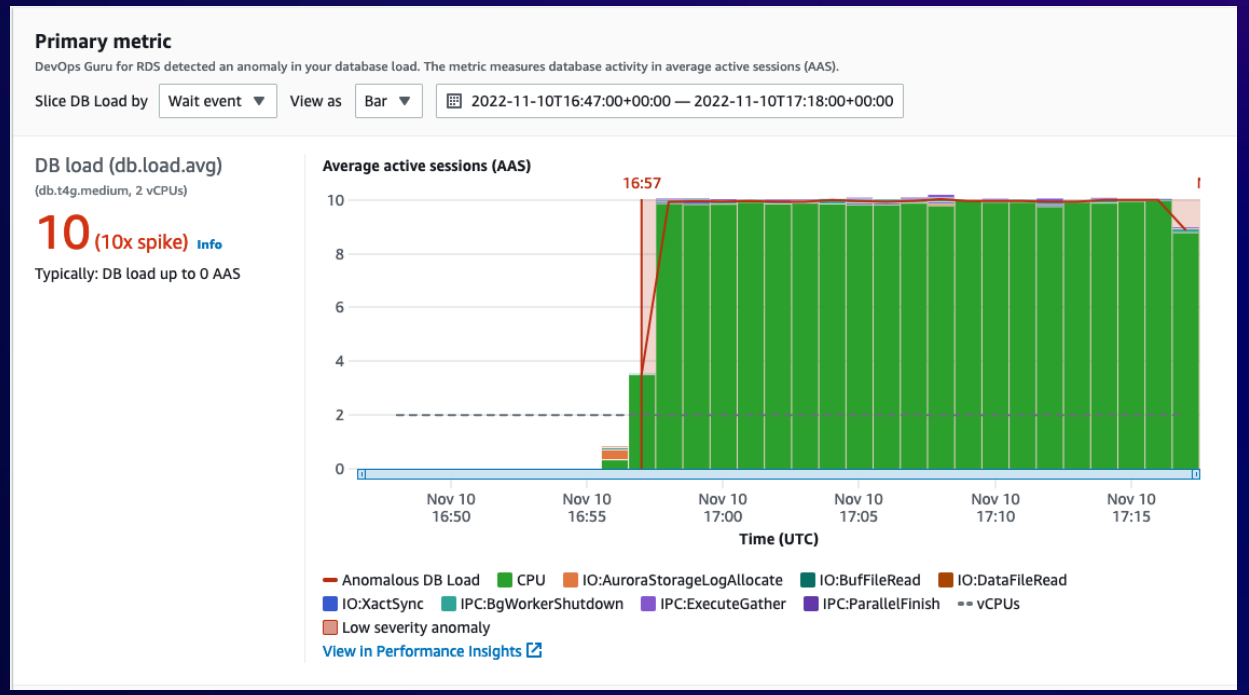
aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tuning for scale

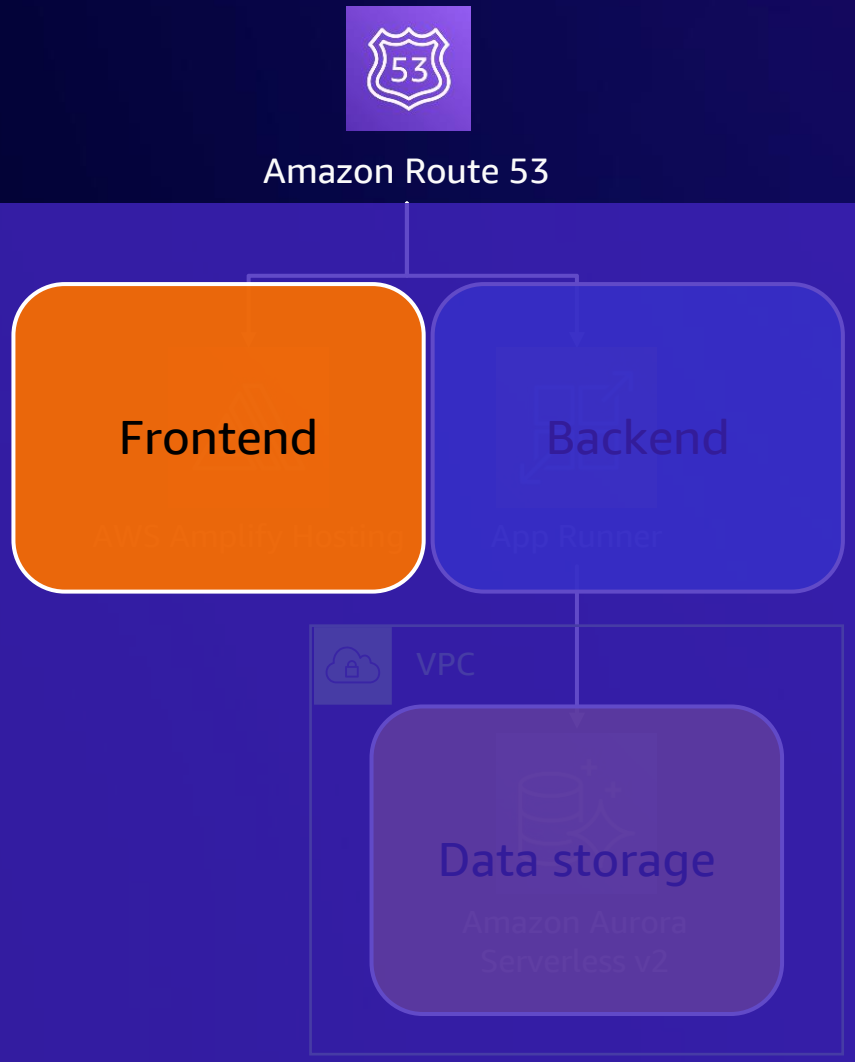
With data in hand, you can now begin to tackle some of the most common pain points in scaling your application:

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



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s learn more about the frontend tier



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling the frontend

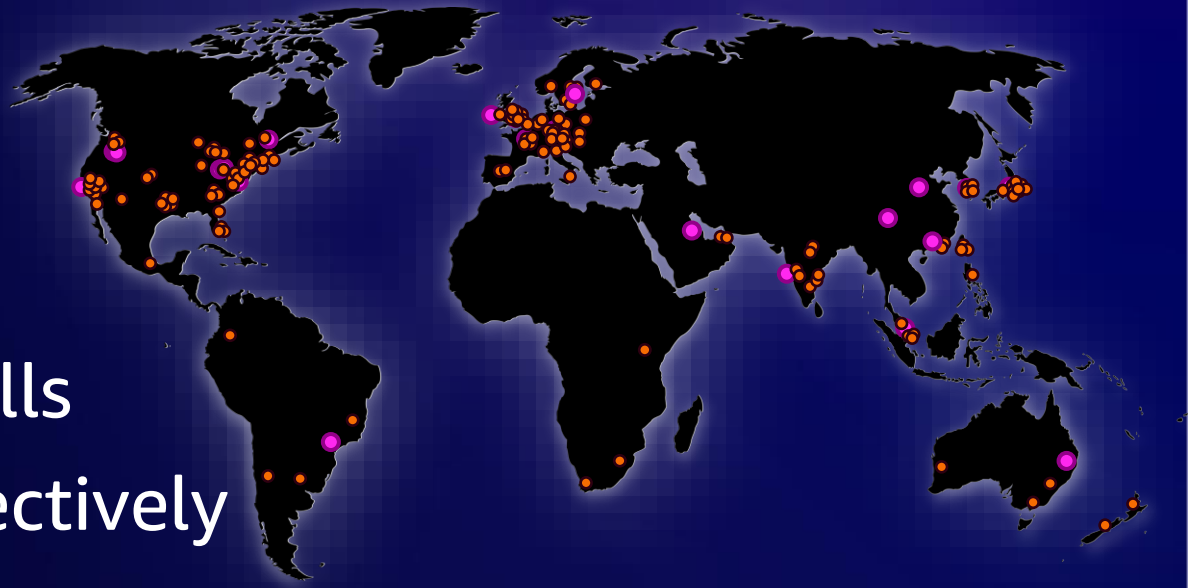
Amazon CloudFront

Generally speaking, Amplify Hosting can scale to meet customer needs

Built on top of the 550+ 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

© 2023, 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 550+ Amazon CloudFront PoPs globally

Performance typically comes from

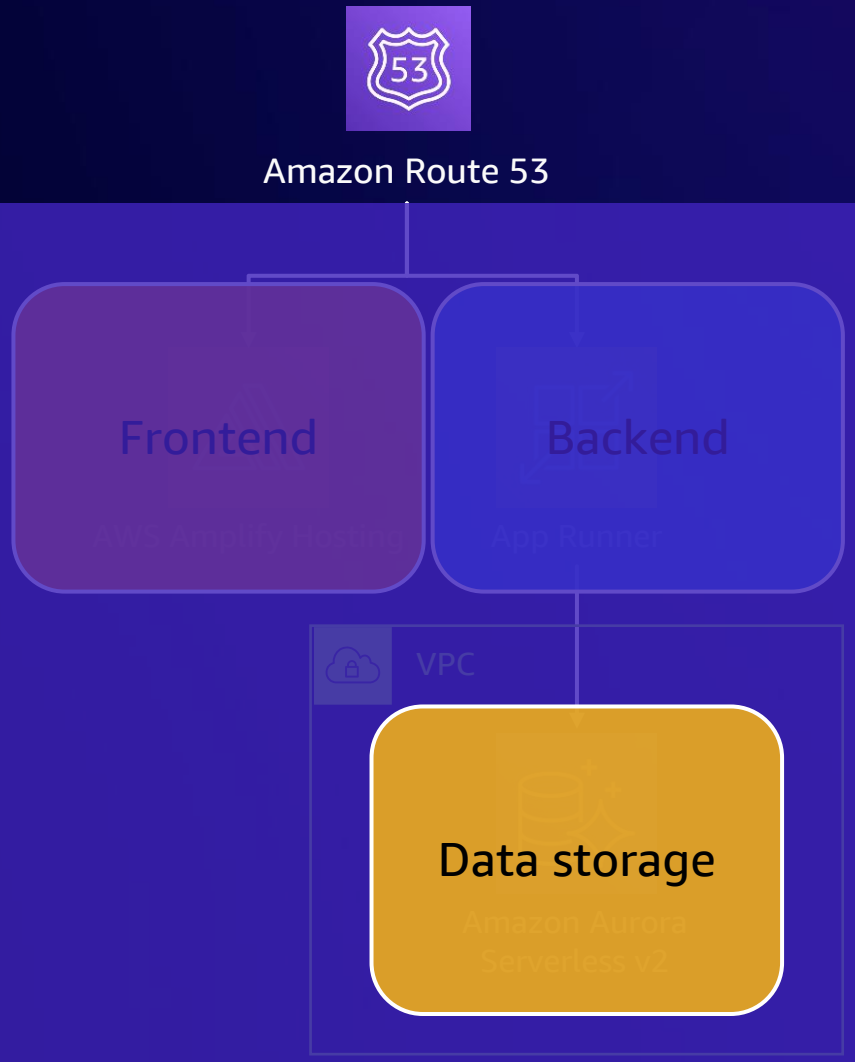
- Tuning frontend code
- Reducing the number of backend calls
- Caching images/JavaScript/CSS effect

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

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

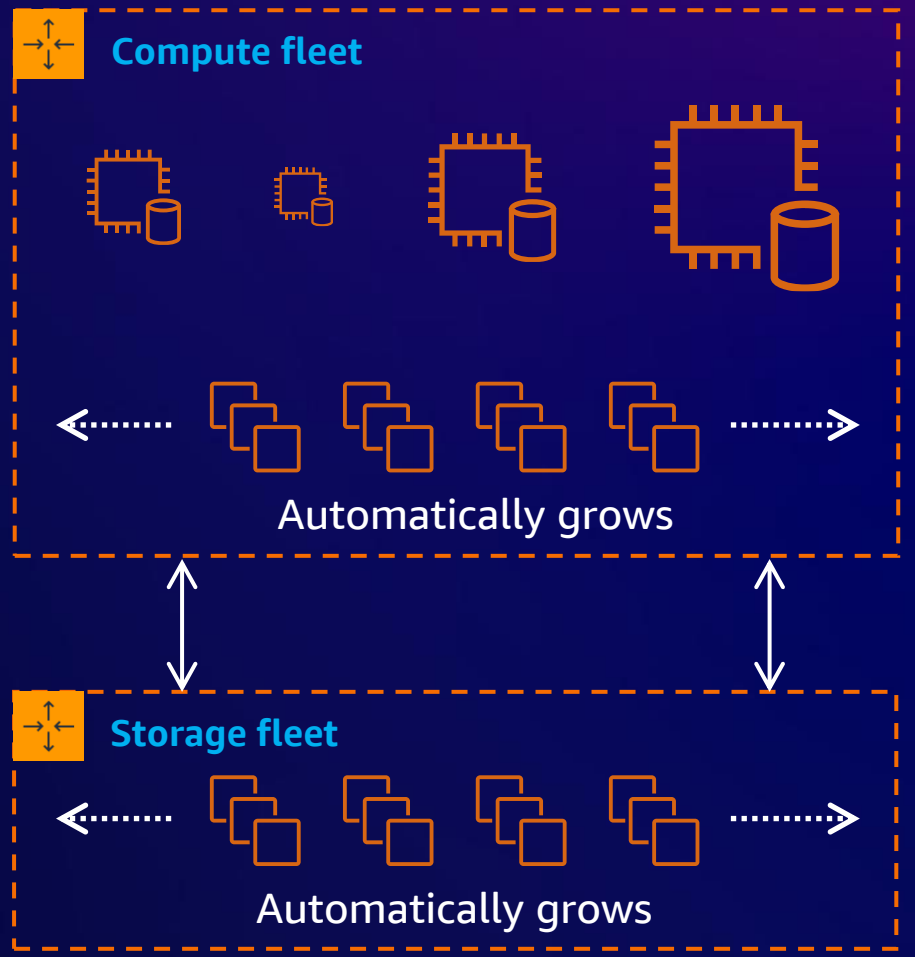
Let’s learn more about the data tier



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: Scaling

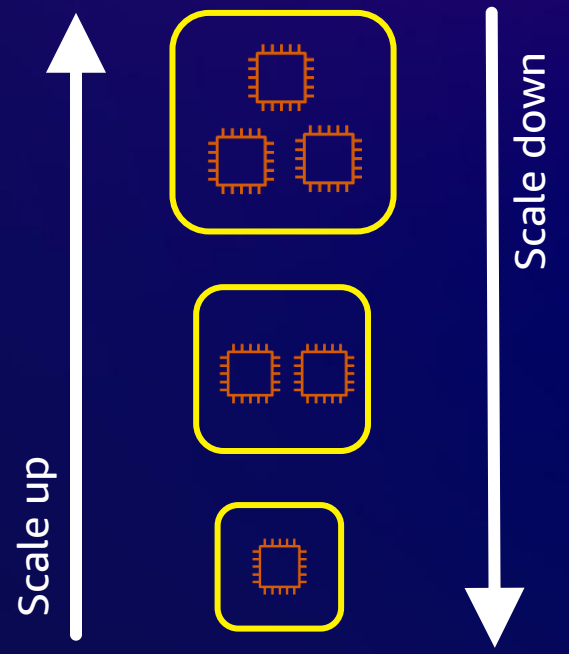
- Scales in place in under a second by adding more CPU and memory resources
- No impact due to scaling even when running hundreds of thousands of transactions
- Compute fleet continuously monitored and scaled horizontally for heat management
- Background movement of idling instances while preserving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



aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: Fine-grained capacity adjustments

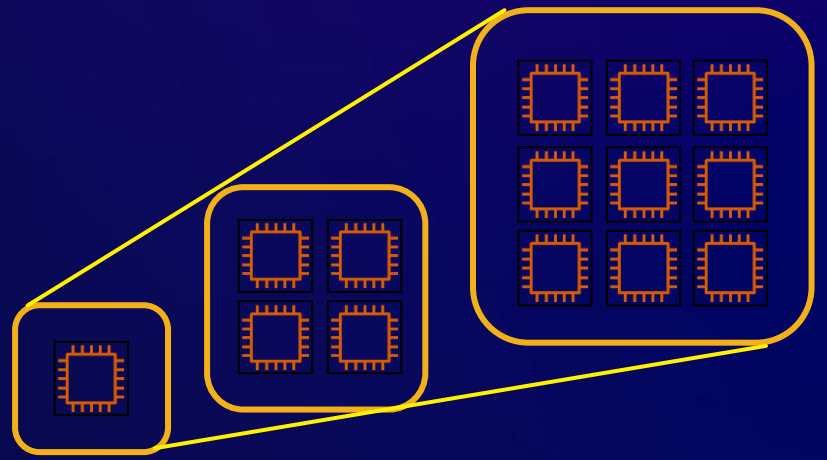
- Aurora Serverless capacity is measured in Aurora capacity units (ACU)
- 1 ACU comes with 2 GiB of memory
- Starting capacity as small as 0.5 ACU (1 GiB)
- Maximum capacity is 128 ACU (256 GiB)
- Fine-grained scaling with 0.5 ACU increments



aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: Scaling factors

- Scale-up rate is predictable and proportional to current capacity – larger instances scale up faster
- CPU utilization of both foreground and background processes (e.g. purge or vacuum)
- Memory utilization of internal data structures (e.g. buffer pool)
- Network throughput is proportional to capacity; capacity is scaled to match network throughput needs

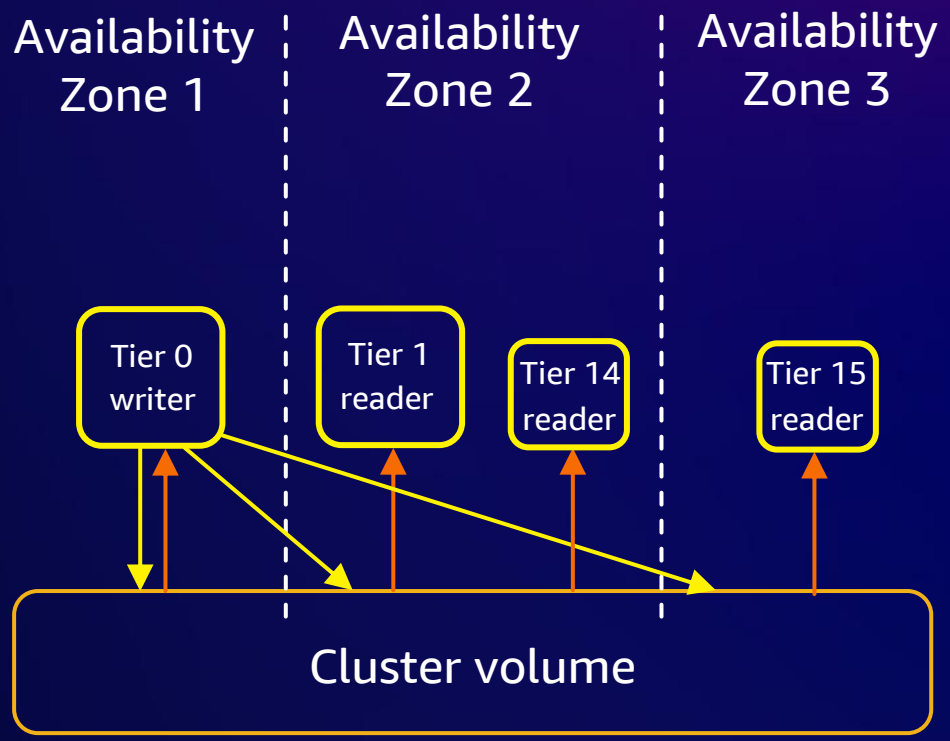


aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Aurora Serverless v2: Scaling with replicas

- 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
© 2023, 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

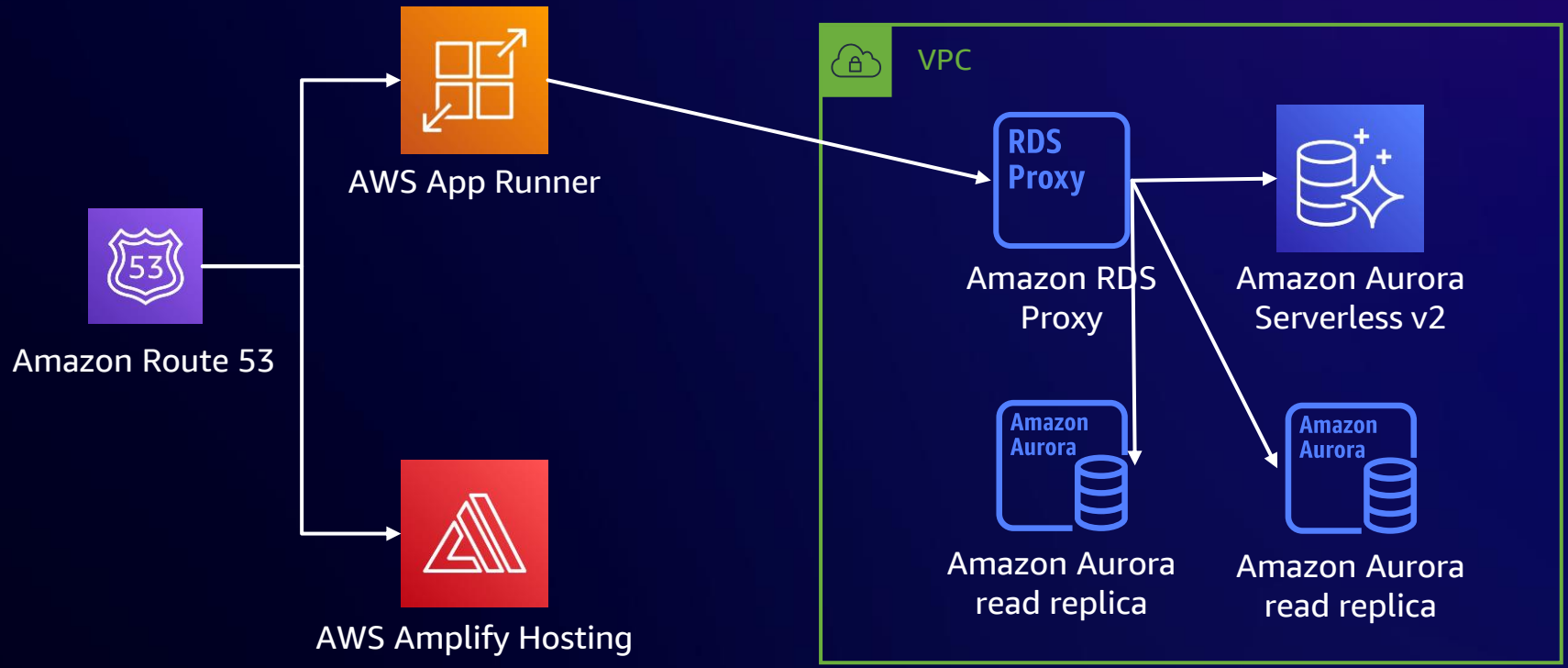
Fully managed DB proxy, compatible with your database

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

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

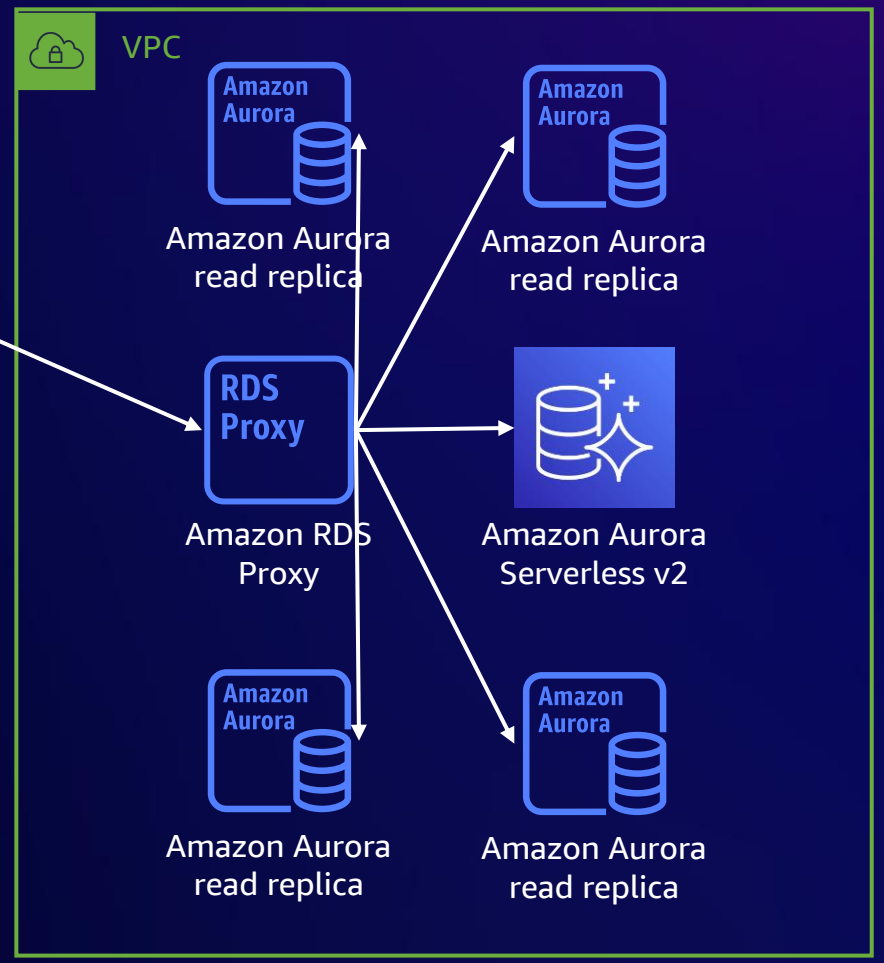
Scaling Aurora Serverless v2



© 2023, 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



© 2023, 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon ElastiCache

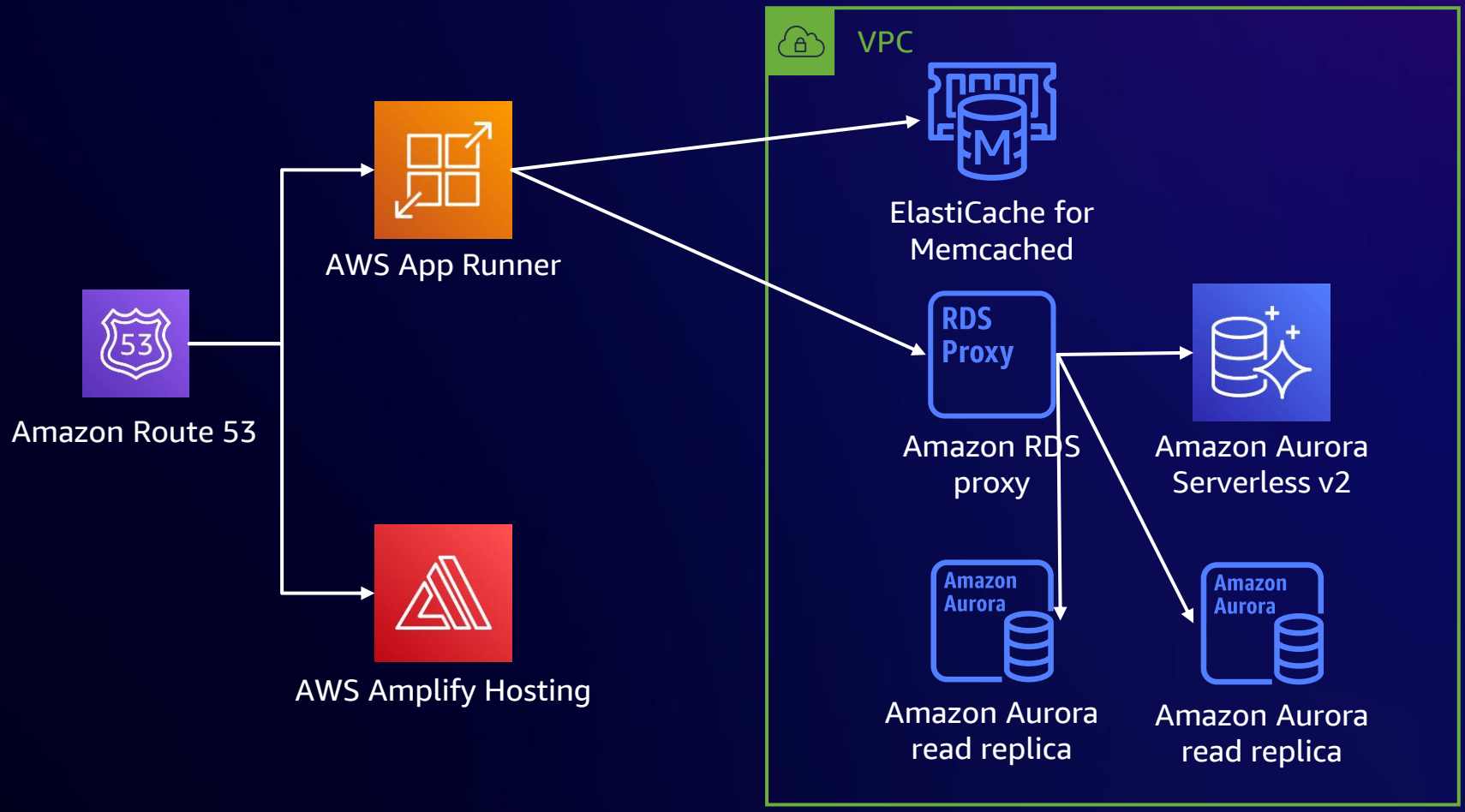
Amazon ElastiCache

- 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Add database caching with ElastiCache

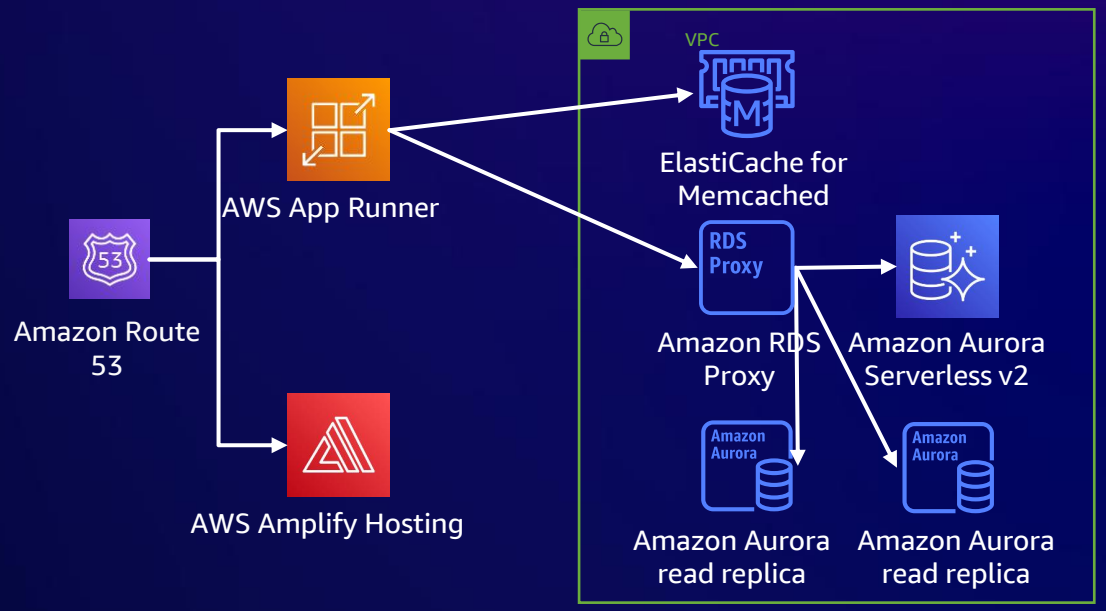


© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Scaling the data tier

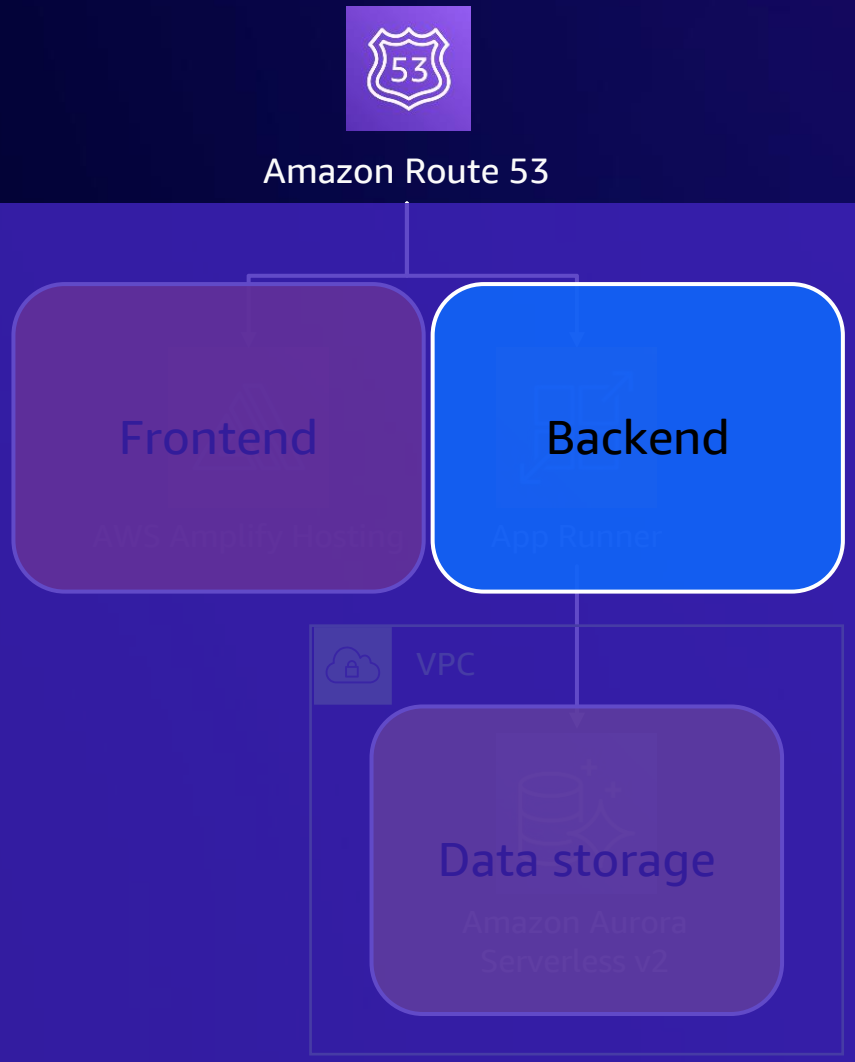
Three main methods for scaling the data tier:

- 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



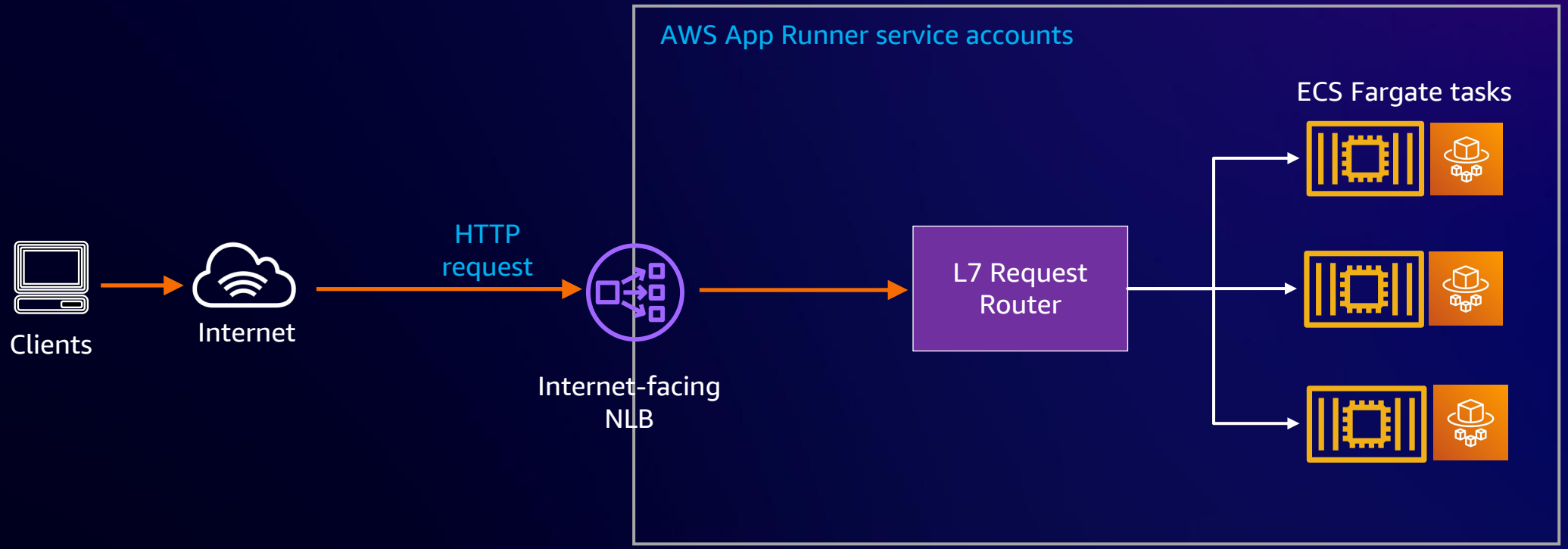
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s learn more about the backend tier



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS App Runner: A closer look



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

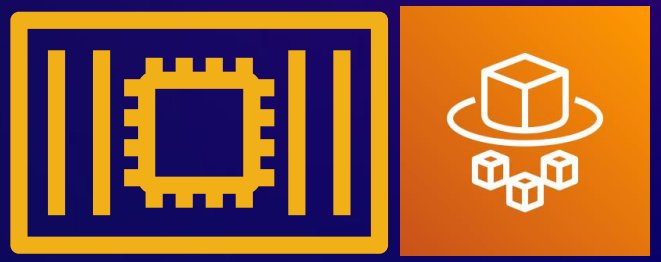
App Runner: Scaling instance sizes

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

*Default/soft limit which can be increased

ECS Fargate tasks



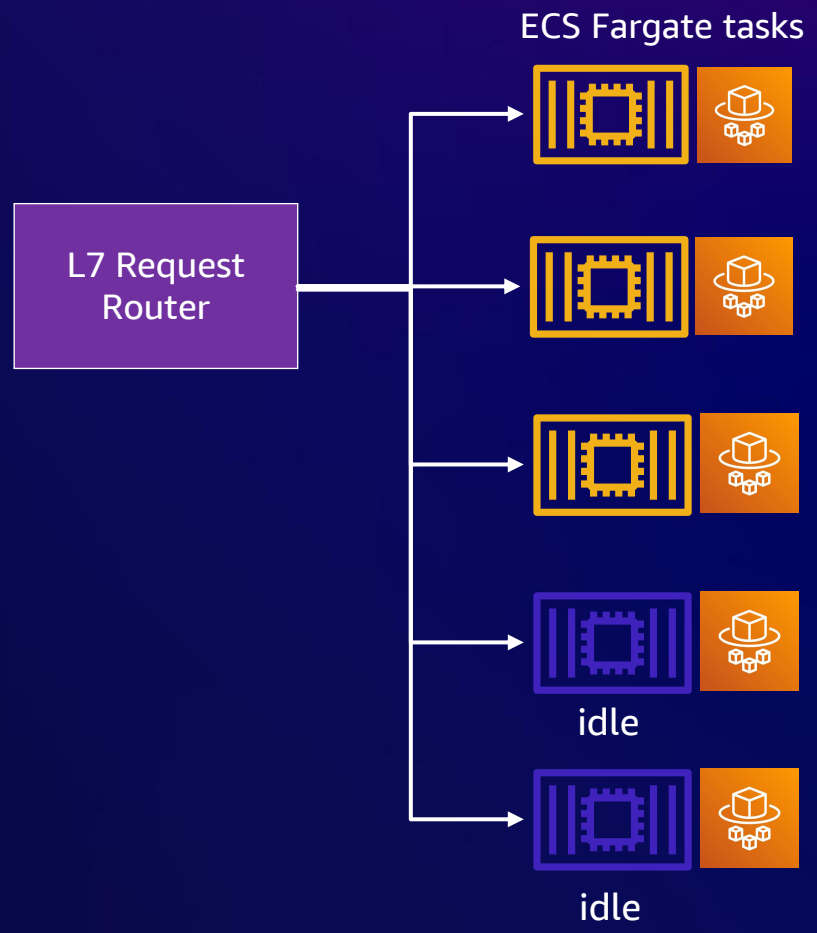
CPU	Memory
0.25 vCPU	0.5 GB
0.25 vCPU	1 GB
0.5 vCPU	1 GB
1 vCPU	2 GB
1 vCPU	3 GB
1 vCPU	4 GB
2 vCPU	4 GB
2 vCPU	6 GB
4 vCPU	8 GB
4 vCPU	10 GB
4 vCPU	12 GB

aws

© 2023, 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



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

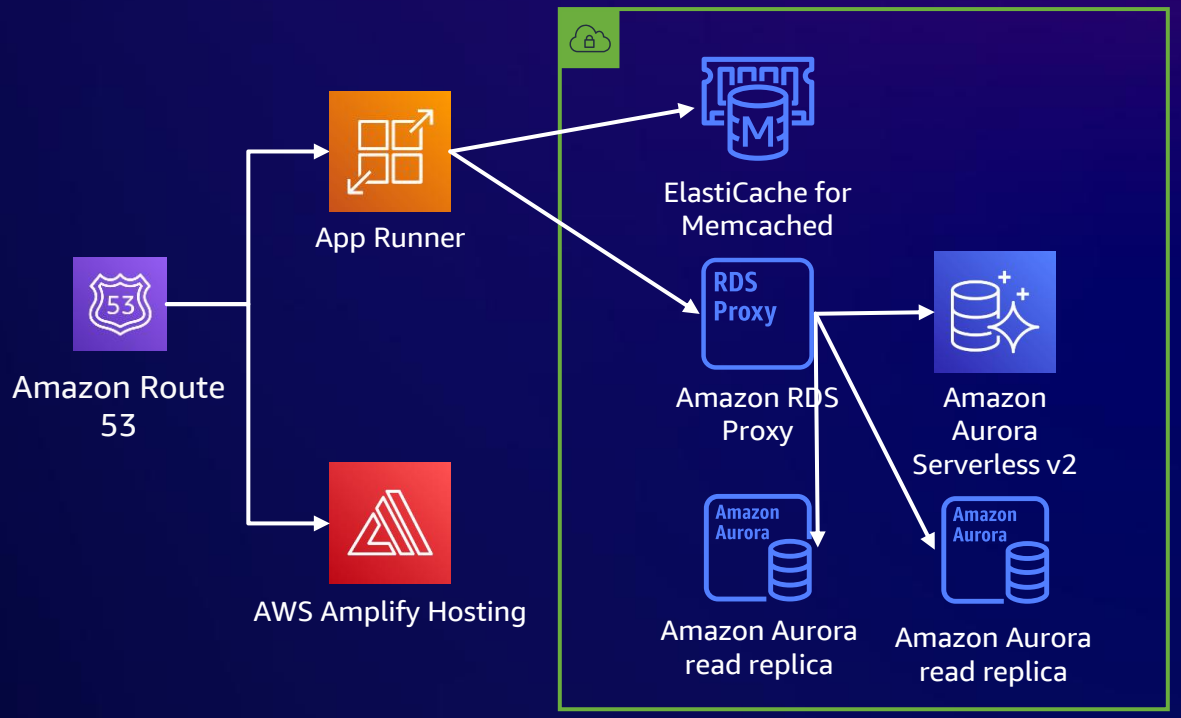
Scaling the backend tier

With basic default 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



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >100,000

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

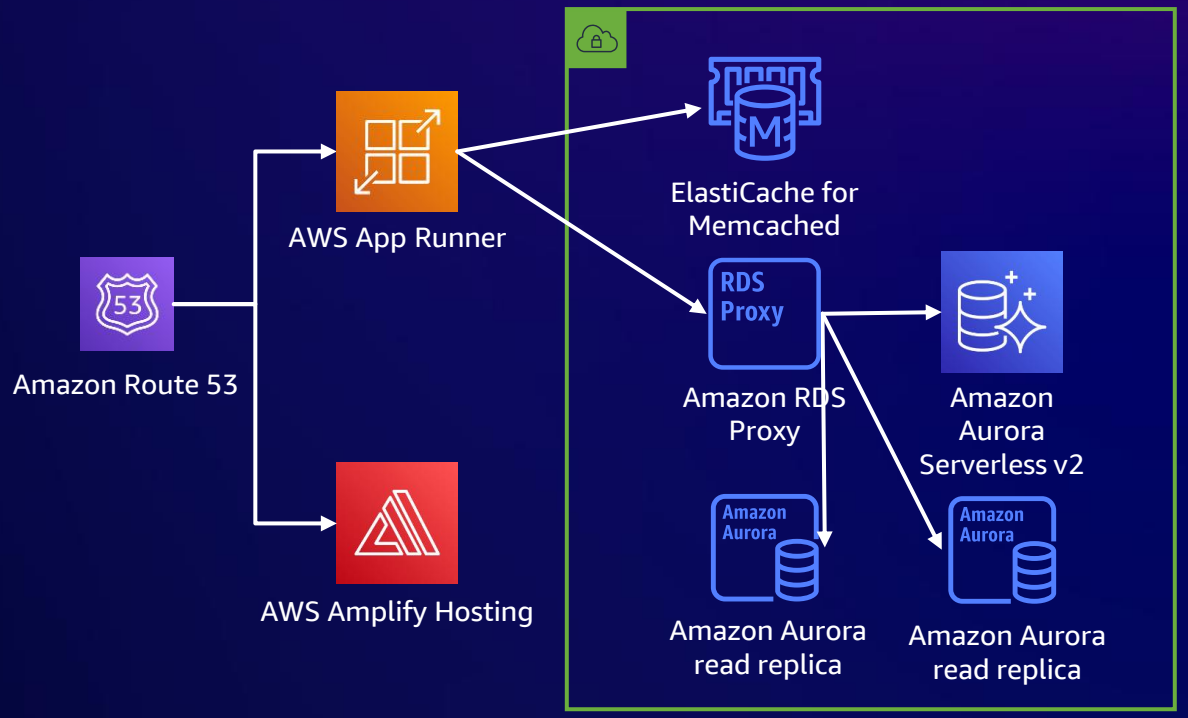
Users: >1,000,000

© 2023, 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:

- Application complexity/feature growth begins to necessitate new infrastructure needs
- Database writes begin to become a bottleneck
- Development and operational tasks weighed down by the monolithic app structure



© 2023, 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

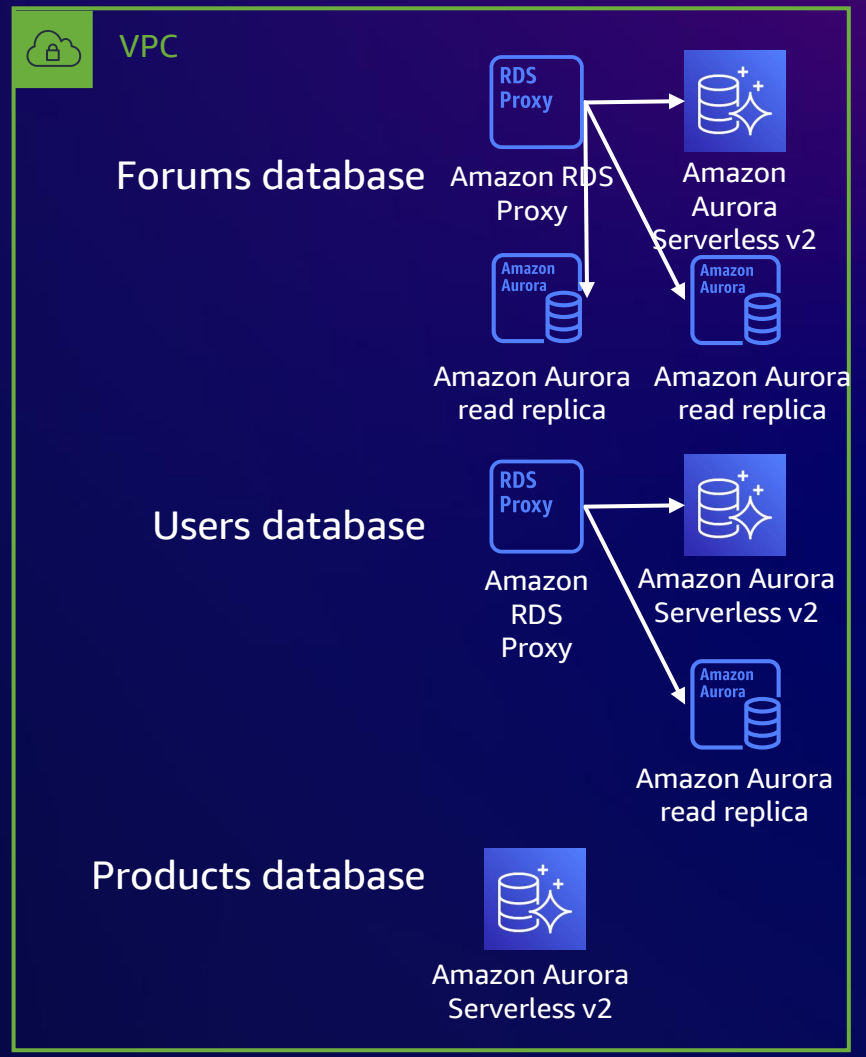
aws

© 2023, 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

© 2023, 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Breaking up the backend tier

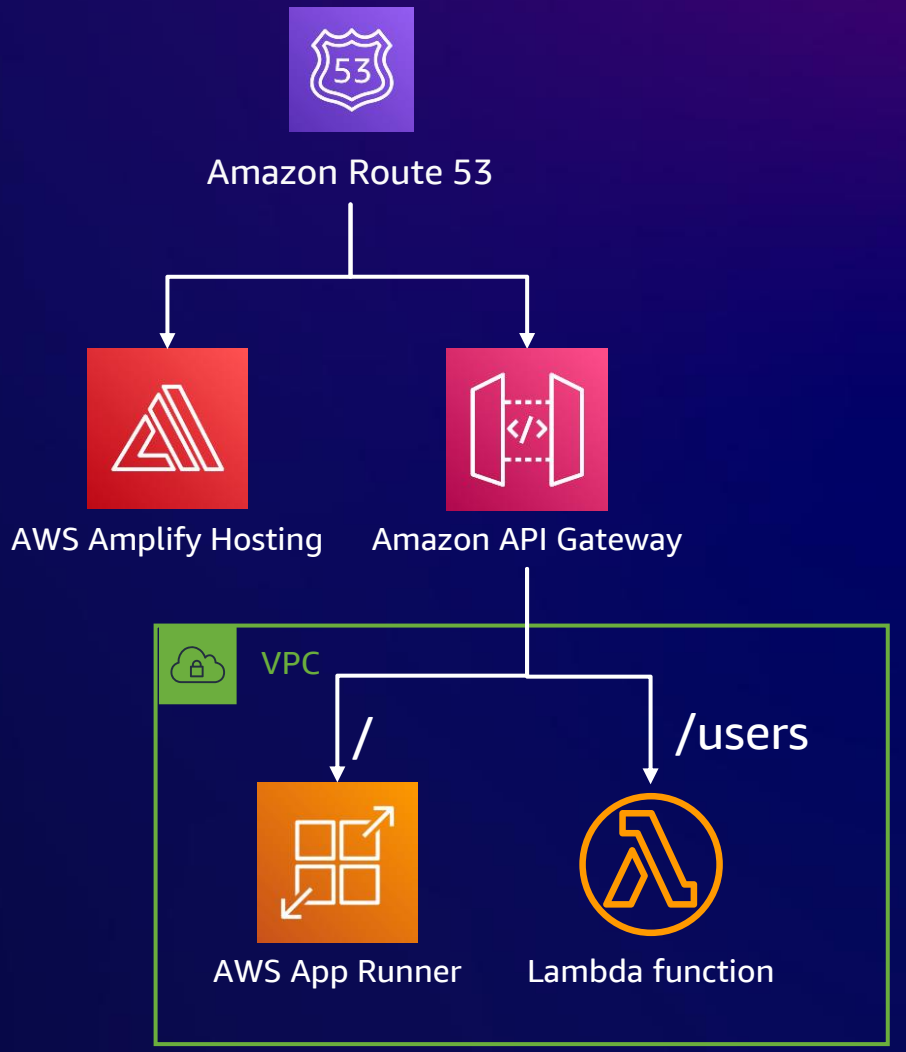
Breaking up the backend can mirror the data tier

- 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

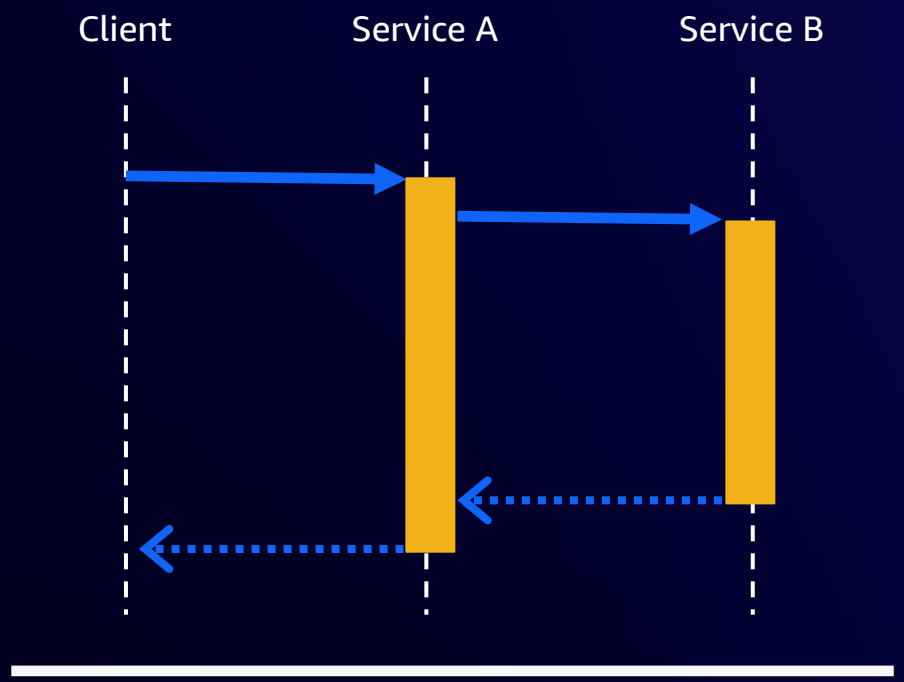
- Internal facing API-based services
- Moving from sync to async
- Leveraging queues, topics, buses, and streams to build event-driven architectures

Microservices complexity grows at a factor of scale

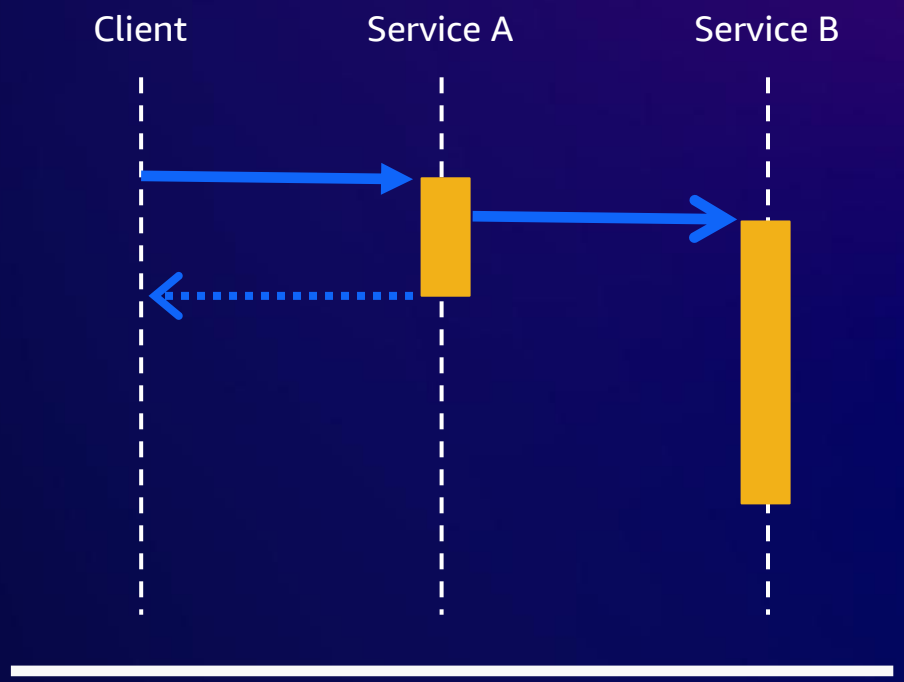


aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thinking asynchronously



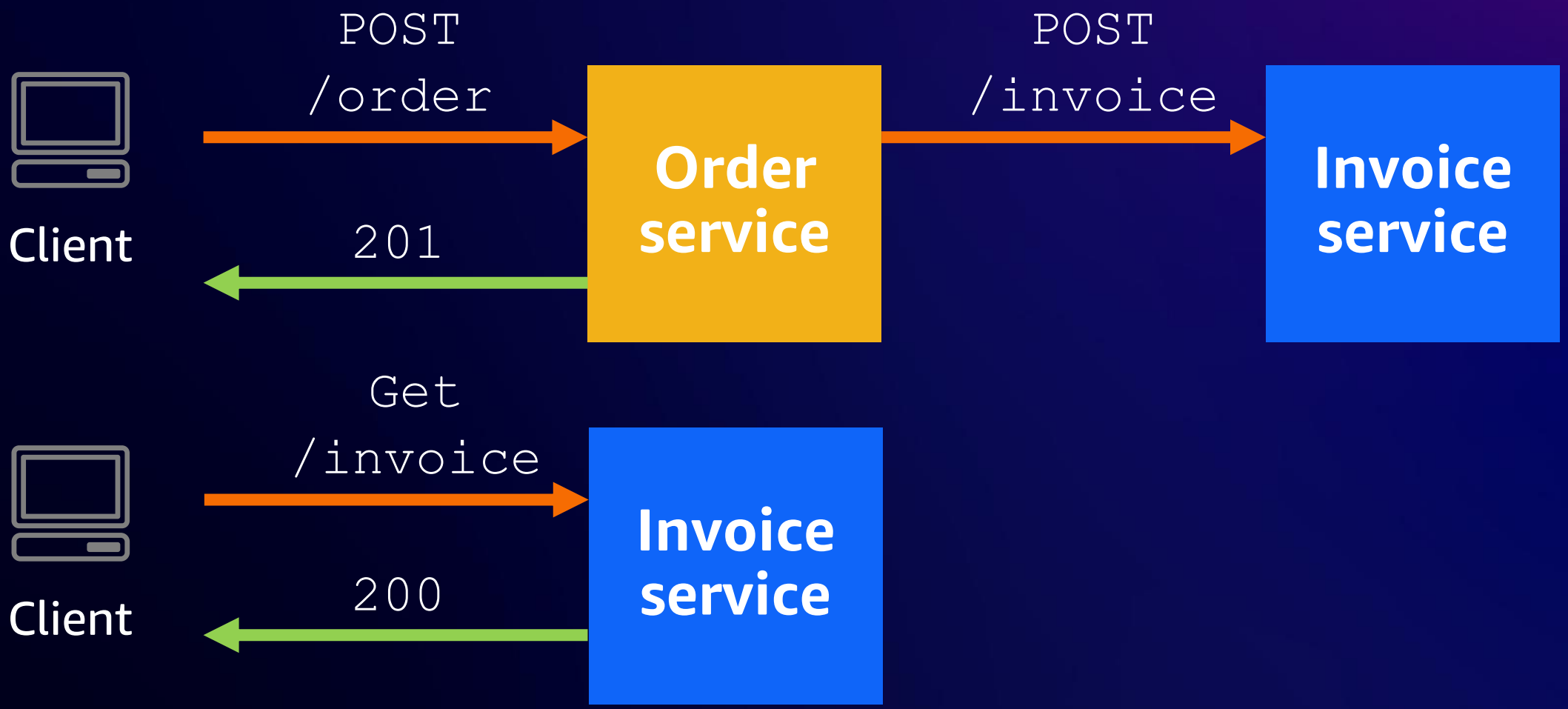
Synchronous commands



Asynchronous events

aws
© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Asynchronous APIs



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“The time spent to try making a process async will pay for itself in you gaining deeper understanding of what is really happening with your data.”

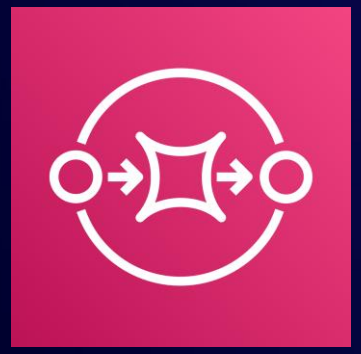
Me

Right now

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Topics, streams, queues, and buses

Amazon Simple Notification Service (Amazon SNS)



Amazon Simple Queue Service (Amazon SQS)

Amazon EventBridge

Amazon Kinesis Data Streams

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Async service decider cheat sheet

Massive throughput/ordering/multiple consumers/replay?

- Amazon Kinesis Data Streams

One to mostly one or minimal fanout, direct to Lambda/HTTP target?

- Amazon SNS

Buffer requests until they can be consumed, whether ordered or not?

- Amazon SQS

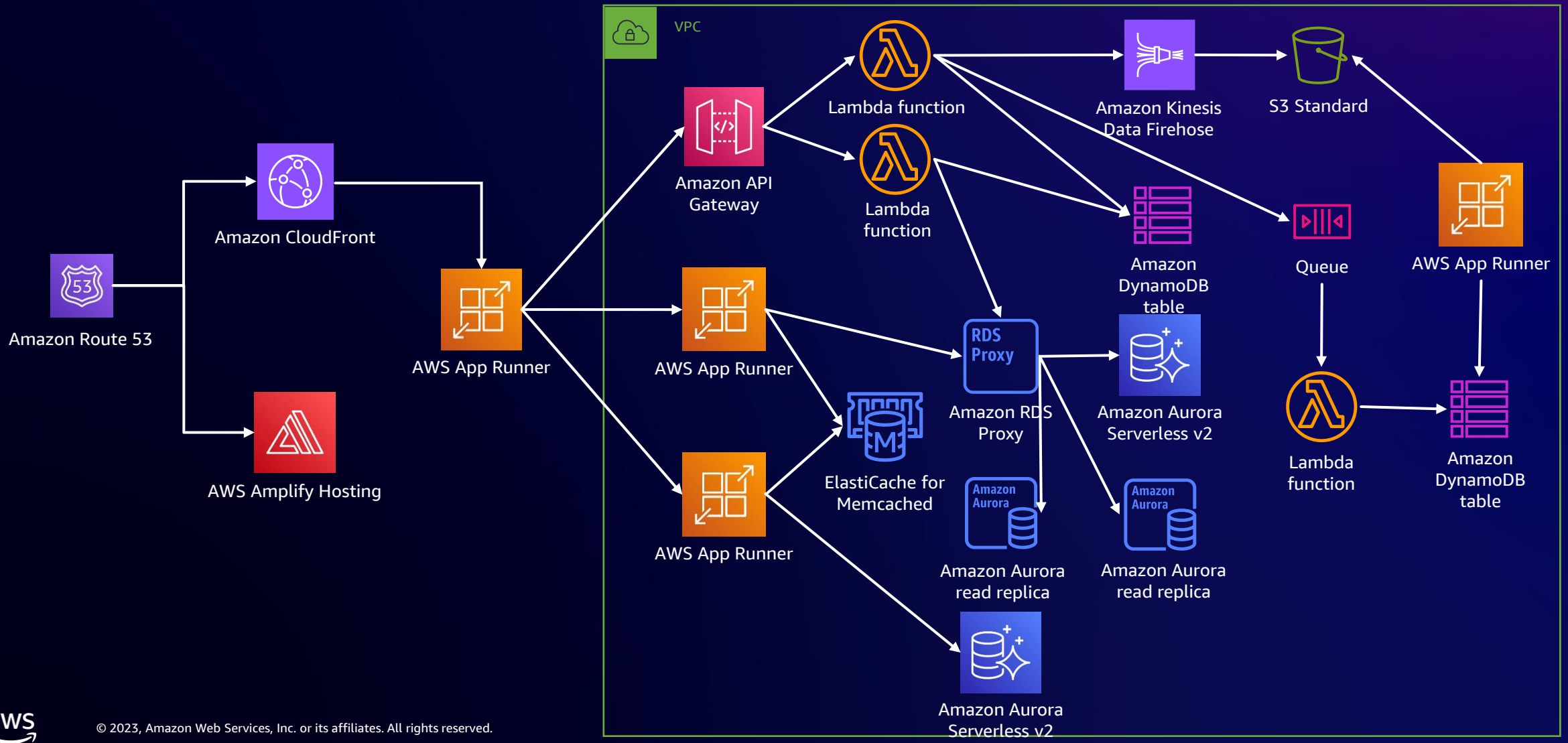
One to many fanout, lots of different consumer targets, schema matching, granular target rules?

- Amazon EventBridge

aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The microservices architecture



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Users: >10,000,000

© 2023, 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
- Start to build on self-managed compute
- Evaluating how to improve caching at all tiers

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

To infinity . . .

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In closing

We’re in such a better place in being able to scale out of the box than we were even 5 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

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Please complete the session survey in the mobile app

Chris Munns

Skye Hart

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.