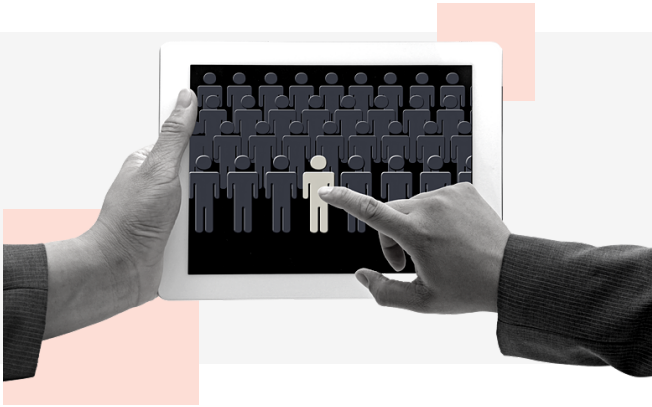
Virgil.

About Virgil

Virgil specializes in developing technology-enabled solutions that help individuals and businesses streamline the recruiting process. Hcareers is one of their most popular recruitment platforms that connects employers with hospitality talent.



Challenges

- The platform needed to be modernized in order to improve code performance.
- Bulk job posting was a time-consuming process.
- Quickly parsing resumes and building a match-making engine using Machine Learning.
- For further visibility into the ROI, improved analytics and a customized solution were necessary.
- Integration with existing ATS (Applicant tracking systems) for better recruitment management.

Solution

Understanding the architecture

- During the first few weeks, our managed engineering team spent time learning about the current code, platform architecture, and business logic. Following that, we re-architected the architecture to be built on microservices, allowing the services to be decoupled and the application to be fail-safe.

Built scalable platform

- Increasing the platform's scalability was one of the major concern. Hence, the system is containerized using the AWS Fargate to delegate the scalability.

Faster report generation

- The data was centralized using AWS Athena for faster analytics and report generation.

Improved wait time

- We devised a system to keep track of messages in the queue while implementing a queuing mechanism for submitting job postings. It will check for job posting notifications on a regular basis and act on them instantly. Employers were having to wait longer earlier when AirSchedular was used to execute this task.

- This also increased debugging visibility for developers, allowing them to know if there were any errors in uploading files.

Code optimization

- For faster resume processing and better application performance, we improved the existing code.

Caching mechanism implementation

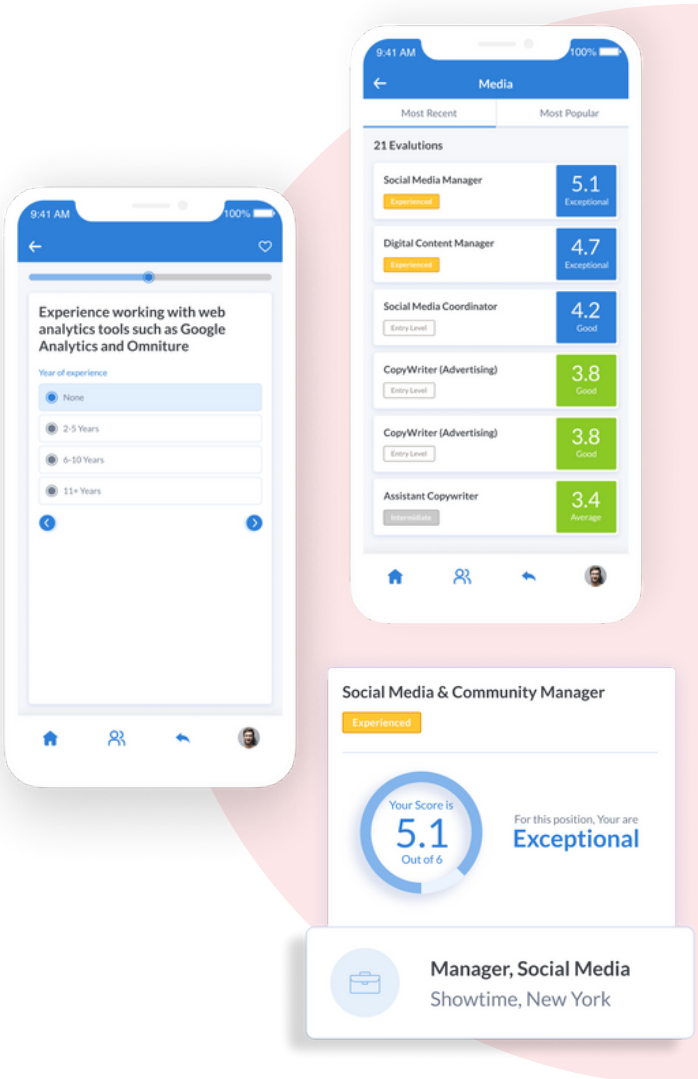
- The caching approach was put in place to reduce the number of times the Google location search API billed for frequently searched job locations. We used ElastiCache for two months and preserved the caching limit, which allowed us to cache locations several times while only being paid once.

Implementation of effective matchmaking process

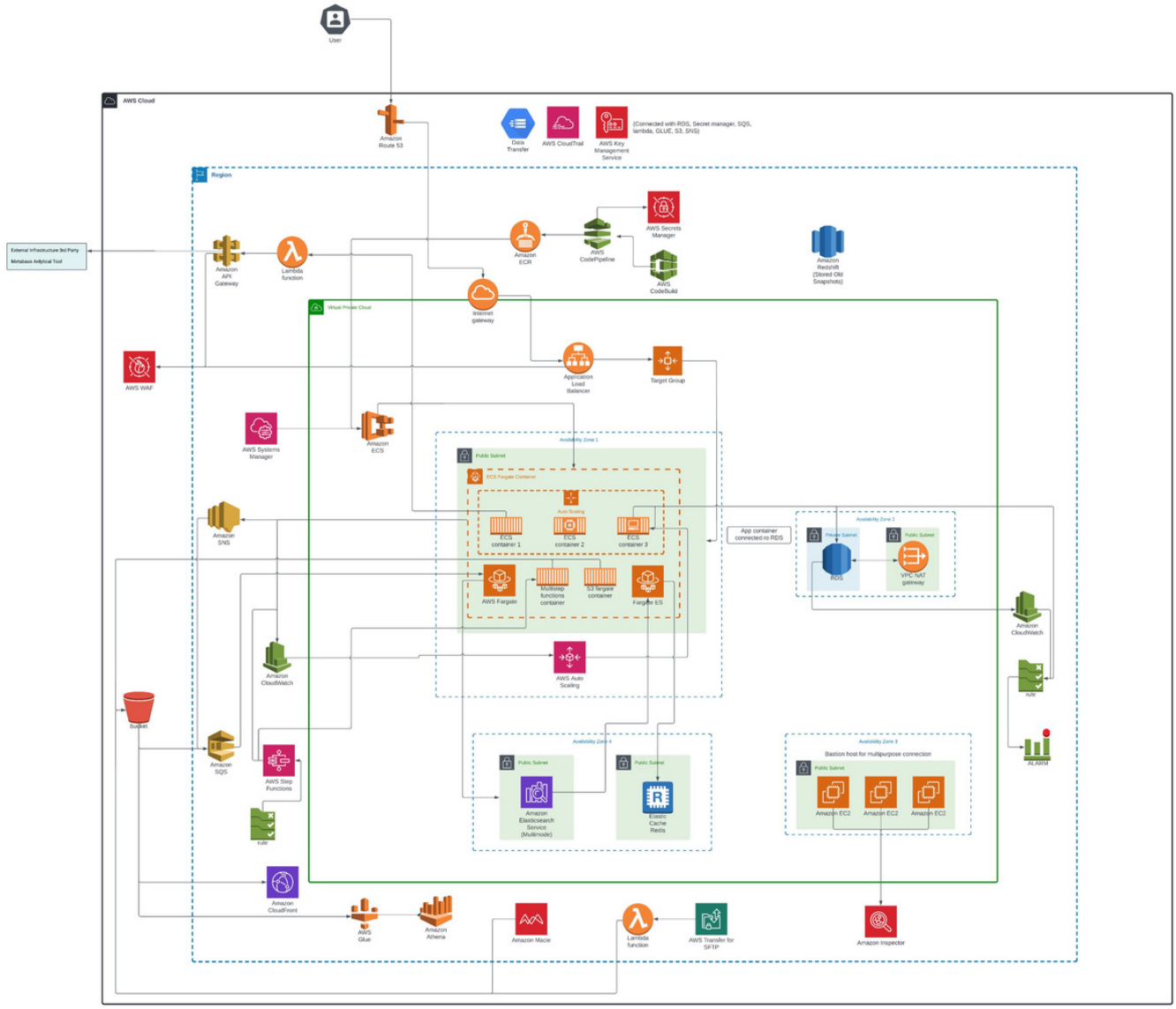
- We enabled a mass job posting mechanism in the system for large-scale recruiting and candidate screening based on the preferences of thousands of relevant profiles.

Key Results

- Significantly improved scalability of the platform.
- Significantly enhanced application performance, resulting in an increased number of users.
- Increased visibility into credit usage by clients using the newly developed credit-ledger system.
- Using AWS managed services, the platform can handle more than 600,000 users per day, which is 10 times the previous figure, without compromising the user experience.



Architecture Overview



Amazon Lambda:

- AWS Lambda transfers files from one location of Amazon S3 to another. For example, if one applicant from India is applying for jobs based in the US, it transfers the resume from an S3 storage location in India to S3 in the US.
- Also, AWS Lambda updates any changes made in the applicant's records to the underlying database. For example, applicants' records have some limits set for deletion from the database, or whenever new users register with the platform, it updates the database.
- Once applicants apply for a job, a confirmation email is sent. AWS Lambda triggers the SQS function and executes micro functions to send an email.

Amazon Fargate:

- AWS Fargate manages containerized code of our application platform.
- It has been chosen for better security and for eliminating the operational overhead of scaling, patching, and managing servers.

Amazon RDS:

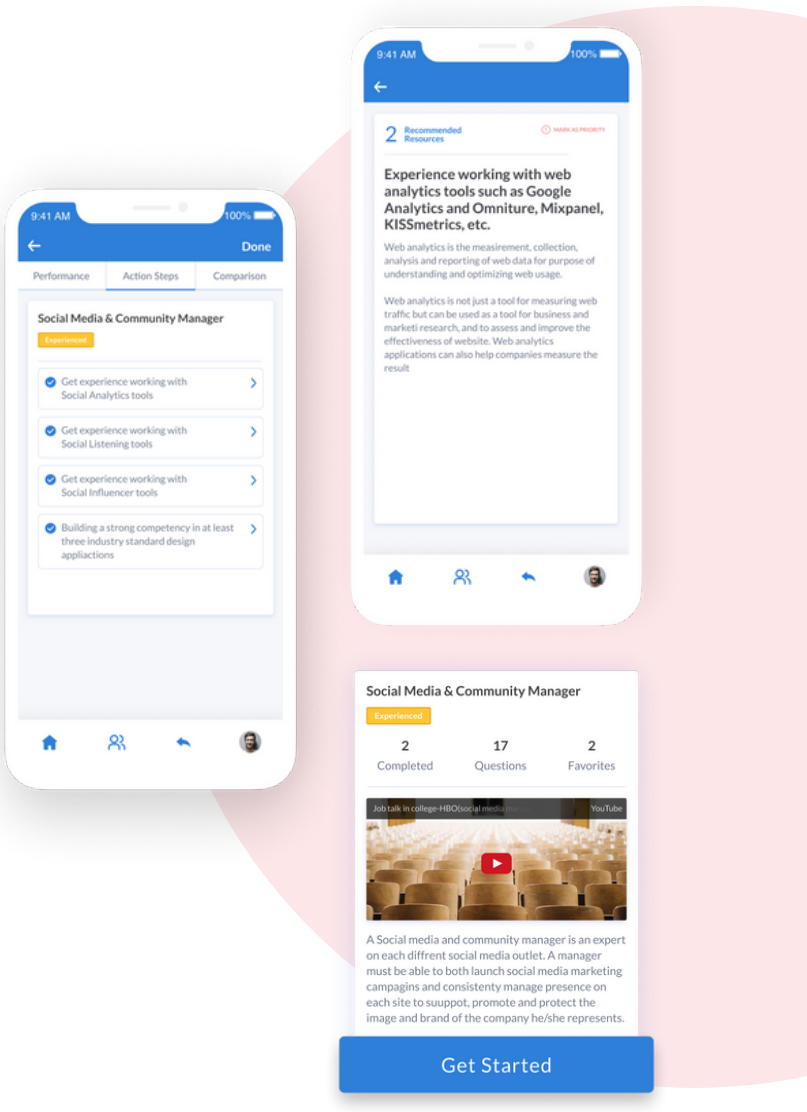
- All the app data for multiple microservices has been stored in Amazon RDS for scalability and ease of operability.
- It performs complex queries and various operations on relational data. For example, fetching job data when users hit the search with their preferences.
- In case of application data loss, automated snapshots have been used for better availability and durability.
- RDS management console is employed to view operational metrics, monitor memory, and I/O activities, and connections.

Amazon EC2:

- EC2 connects external resources to private resources, creates virtual private networks, and hosts tools, among other things.
- A special-purpose web API for limited user accounts has also been hosted on EC2. In our case, these are Premium accounts of employers, admin users, and so on.

AWS Athena:

- AWS Athena was used to query massive, raw application data files in the whole application database, such as job postings. It acts as a medium between S3-stored Parquet files and third-party analytics service.



AWS Glue:

- AWS Glue integrated with Athena was utilized to get data from Parquet files hosted on S3.

Elasticsearch:

- Since our job application platform contains enormous amounts of data from a variety of sources and places, candidates must use a filter to narrow down their job search results based on their interests.
- Elasticsearch did the job of quickly getting data based on user-defined filters such as employment location, job role, and so on.

Amazon ECR:

- We used Amazon Elastic Container Registry (ECR) to store our docker images for easy deployment and download.

Monitoring & troubleshooting:

- AWS CloudWatch and CloudTrail are two AWS services we've utilized for monitoring and troubleshooting.
- CloudWatch Container Insights, composite alarms, and CloudWatch Logs Insights are used to collect and evaluate various metrics, logs, and real-time metrics.
- The use of CloudTrail made security analysis and troubleshooting more easier. It also allowed us to have tighter AWS account security governance and risk auditing.