Part C: Scalable System Design

At this final stage, the aim was to make our already robust system (from Part B) more scalable. We had to make sure that the system could autonomously manage it without slowing down or crashing should the load unexpectedly increase, for example from a spike in internet of things device activity or user activity.

Part B already featured Auto Scaling Groups; however, we now included more intelligent scaling rules depending on real-time CPU use or incoming traffic. This lets the web and backend layers adjust their sizes according to demand.

For the web layer, if necessary, we can now go from 2 to even 10 EC2s.
For the backend, the group adapts according to request load or processing time.

This guarantees users never run into delays, no matter how hectic life becomes.

To manage erratic loads on the database:

Aurora Serverless v2 automatically scales database capacity depending on requests and connections.
Alternately, choose RDS MySQL Multi-AZ with more IOPS and auto-scaling storage.

This enables the DB side to keep up with increasing application demands.

We could also add: if required.

Amazon Cloud Front, a CDN, offers quicker worldwide content delivery.
For file/data storage offloading, Amazon S3.
For event-based processing from IoT devices, Amazon Kinesis or Lambda.

With all these enhancements, the system becomes not just resilient but also really elastic. It scales depending on actual use, so keeping costs under control since we only pay for what we use. Performance remains high.

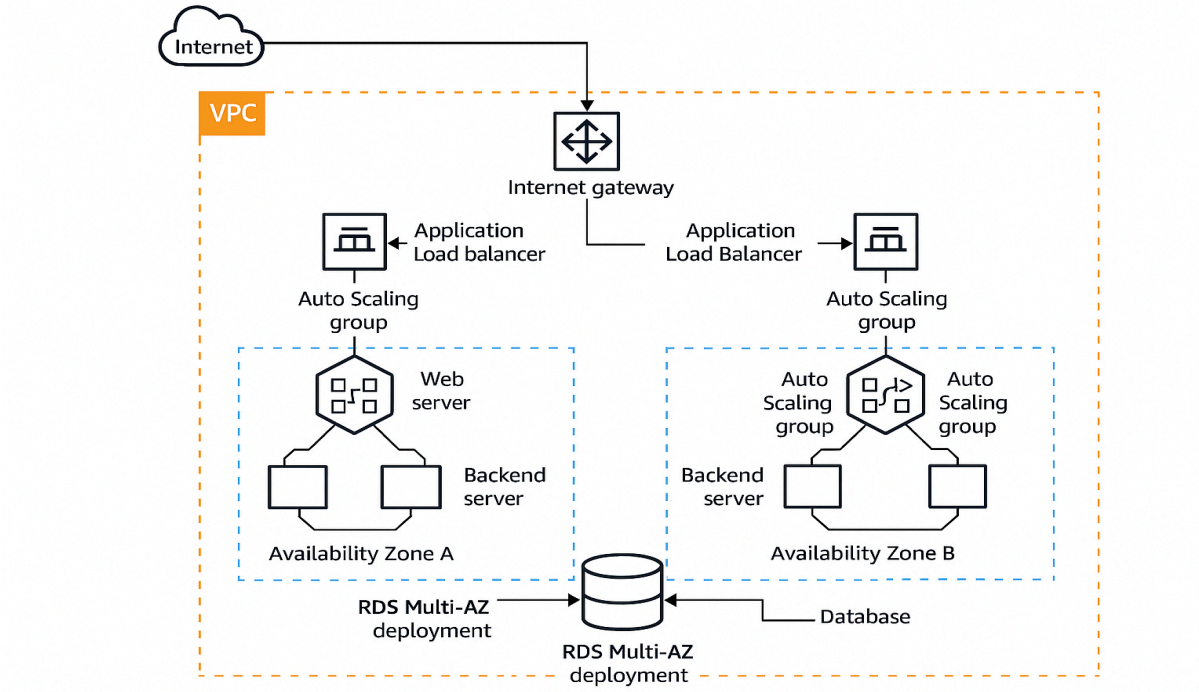

Figure: Scalable Architecture with Auto Scaling and RDS Multi-AZ