# Solution Document

# for

# *Not Just Any Startup*

*Not Just Any Startup*'s mobile application platform enables real-time consumer - service-provider interaction. We propose migrating to AWS to help *Not Just Any Startup*:

* Scale without need to guess capacity: provision EC2 instances (virtual servers) on-demand using Amazon**Elastic Compute Cloud** (**EC2**) & Auto Scaling

Elastic

*High Availability*

*Scalability*

*Secure*

Manageability

High Performance

Fault-tolerant & Recoverable

* Distribute traffic across healthy EC2 instances in multiple Availability Zones with Cross -zone load balancing option of AWS **Elastic Load Balancing (ELB)**
* Plan for Disaster Recovery
  + redundant instances across AZs
  + highly durable S3 storage (& protection through bucket policies)
  + Elastic Block Store (EBS) volume snapshots
  + RDS - multi-region - snapshots & read replicas
  + Amazon CloudFormation - templatize environment & resource stack
* Manage identities & sync user data across devices using **Cognito**
  + temp credentials avoid need for individual AWS or IAM accounts using Identity Pools & an associated IAM Role (default access Cognito Sync)
  + local cache & service deltas sync'd over HTTPS when client is online
* Push notifications to clients using **Simple Notification Service (SNS)** topics
* Collect, visualize, understand app usage data with Amazon Mobile Analytics:
  + KPIs - *Engagement* (DAU, MAU, new users, Sticky Factor (DAU/MAU)), *Retention* (Day 1/3/7), *Sessions*, *Revenue* (ARPPDAU) etc.
* Configure a performant database layer: with Relational Database Service **(RDS)**
  + Scale Up: Instance Class up to 32 VCPUs & 244 GB memory
  + Scale out: Read replicas to increase performance
* Leverage self-healing infrastructure: Amazon **CloudWatch** dynamically scales (out/in) instances. **CloudWatch Logs** monitors application logs and sends error notification to Amazon Simple Notification Service (SNS) when over threshold. Upon launch, new instances execute a UserData script (**bootstrap**), and auto-configure (apply IAM Roles, etc.) and install applications.
* Secure data at rest and in transit
  + at rest: S3 bucket policies, IAM policies, SSE or Client Side Encryption
  + in flight: SSL API endpoints for S3
* Secure environment in face of expansion: **Identity & Access Management (IAM)** for authentication/authorization to AWS infrastructure, with Managed Policies applied to Groups and transitively to the users in a group
* Archive/delete data: using Lifecycle rules to auto transition from S3 buckets to Amazon Glacier
* Manage & replicate environments: AWS CloudFormation templates describe resources (eg. Auto Scaling group, load balancer, database), managed as a single unit. A stack can be created in a different region for DR.

**Flow**

* the public **ELB** has a HTTPS listener for encrypted connections (uses SSL certificate from AWS Certificate Manager), and distributes traffic across healthy instances in both AZs.
* Instances are launched within VPC **subnets** (identified by CIDR blocks).
* **CloudWatch** will execute an Auto Scaling Policy (max, min, desired) when a parameter threshold is crossed. **Auto Scaling** service will scale out/in EC2 instances in the Auto Scaling group.
* an internal load balancer routes traffic to the private subnet
* **Security groups** (stateful, virtual firewalls with in/out rules) are set up at the instance-level to control traffic based on port, protocol, source/destination. For example, { port: 80, protocol: http, source: all incoming}, {port: 22, protocol: SSH, source: <your IP>}
* **Endpoints** (route to service url) will connect instances to S3 without requiring NAT or IGW
* **Roles/Temporary Security Tokens** enable the application to authenticate against AWS APIs. Actors assume a role and receive a temp token associated with the policies of the role. For eg:
  + IAM role granting access to S3 bucket assigned to EC2 instance
  + app running on instance assumes instance role when invoking S3 API, obtains temporary token from **AWS Security Token Service (STS)**, allowing it to authenticate with S3
* authorization to AWS resources happens through Policies, eg:
  + **Effect**: 'Allow', **Action**: ['s3:GetObject', 's3:ListBucket'],   
    **Condition**: { 'IpAddress': {'aws:SourceIp': '192.168.0.2'}},   
    **Resource**: ['arn:aws:s3:::mybucket/\*']
  + Policies will be assigned to Groups to simplify managing permissions
    - eg: create 'IAM\_Administrators' group, assign 'IAMFullAccess' managed policy, create 'Administrator' user and add it to the 'IAM\_Administrators' group
* **Multi-AZ RDS DB cluster** uses synchronous replication between Primary & Secondary across AZs
  + provides fast-failover, minimizing RTO to minutes (Recovery Time Objective -- max downtime to resume in a failure)
  + daily Automated Backups (during maintenance window) in conjunction with transaction logs minimize Recovery Point Objective (RPO) to as little as last five minutes
* **Cross-Region Read Replicas** (async copy) offload read queries from Primary for performance
* a **push notification service** such as Amazon Device Messaging (ADM) or Google Cloud Messaging for Android (GCM) serves as an Amazon **SNS** endpoint
* user session state for web layer maintained in **Dynamo DB** for highly scalable architecture
* ElastiCache (Redis) used to cache frequently queried user data for performance at DB layer

**ASSUMPTIONS**

The IT organization will define the following parameters:

* standards (production, dev) for instance types (vCPU, memory, storage, network) & AMIs
* the presentation tier subnet be made private by removing the Routing Table rule ('Destination CIDR ->Target') that directs traffic to Amazon VPCs IGW. The IGW does NAT for Internet traffic.
* configure the ELB for sticky-session binding to simplify application development
* archival frequency & time-window -- for taking EBS delta snapshots on EC2 & RDS instances
  + an EBS volume is auto-replicated within an AZ. But volume snapshots (saved in S3) are replicated across AZs in a region and this is deemed sufficient for DR
  + Snapshots will be used to create recovery / expansion volumes only in the region where they are created -- otherwise will need to copy snapshot to required region.
* encrypt EBS volumes using AWS KMS for key mgmt (AES-256) where data at rest to be secured
* consider balance of reserved (save up to 75% over hourly rates, but commit up to 3 years, and pay upfront), on-demand (for peak loads) and spot **instances** (non-critical workloads)
* cost-effective **Instance stores** (ephemeral, block-level) used for run/scratch-data areas
* **General Purpose SSD EBS volumes** providing upto 10k IOPS sufficient for small databases, dev and test environments. **Provisioned IOPS SSD** required for IO-intensive loads in production.
* **Network Access Control Lists (ACLs)** or stateless firewalls at subnet level for added security.
* Application and Database tier in private subnets, with a NAT Gateway for Internet access. NAT Gateway is a high-availability instance for NAT and forwarding traffic to Amazon VPC's IGW.
* Amazon **CloudWatch Logs agent** installed on server instances
* Route 53 DNS failover routing policy provides higher availability in instance or AZ failures
* for managing operations, a set of principals with IAM administrative privileges will be tasked with managing access to AWS resources and roles.
* The Amazon RDS DB Instance class (compute & memory) will change as the need for vertical scaling arises (upto 32 vCPUs and 244 GB memory)
* For higher throughput, we can place an Amazon Simple Queue Service (SQS) messaging cluster between the presentation and application layers coupled with Long Polling (WaitTimeSeconds) for querying from the queue.