# High Level Design – We Own The Night

**Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Summary of Changes | Last Updated | Version |
| Alexis Lignereux | Initial Version | 11/02/2014 | 0.1 |
| Alexis Lignereux | Feedback changes | 12/02/2014 | 0.2 |
| Simon Brunton | Added missing information and fixed spelling and grammar | 13/02/2014 | 0.3 |
| Peter Abbott | Added real-time race results and based on WOTN 2013 | 20/02/2014 | 0.4 |
| Martin Nartey | Spelling and grammatical fixes | 20/02/2014 | 0.5 |
| Peter Abbott | Final amendments relating to real time race results | 21/02/2014 | 0.6 |
| Martin Nartey | Updated question and answer section based upon Nike DBT questions. | 26/02/2014 | 0.7 |
| Alexis Lignereux | Adding Addendum AWS template sections | 27/02/2014 | 0.8 |
| Alexis Lignereux | Updated Design refinements | 28/02/2014 | 0.9 |
| Alexis Lignereux | Updates High Level Diagram, Payload from TDL and AWS templates | 04/03/2014 | 0.10 |
| Alex Dergachev | Updated AWS templates. Removed Questions & Answers | 04/03/2014 | 0.11 |

**Agency Contact:** Simon Brunton

**Nike Business Sponsor:** Ashley Saunders

**Expected Launch Date:** 8th May 2014

**Expected Shelf Life:**

Evergreen – new race result application will be used for multiple races each year and race results will be available online until the start of next year’s WOTN campaign (~Jan 2015).

## Problem Statement

The existing AWS-hosted We Own The Night (WOTN) race results site solution needs to be updated for the events in 2014 and for future years. The site will be linked to from the Nike.com WOTN page.

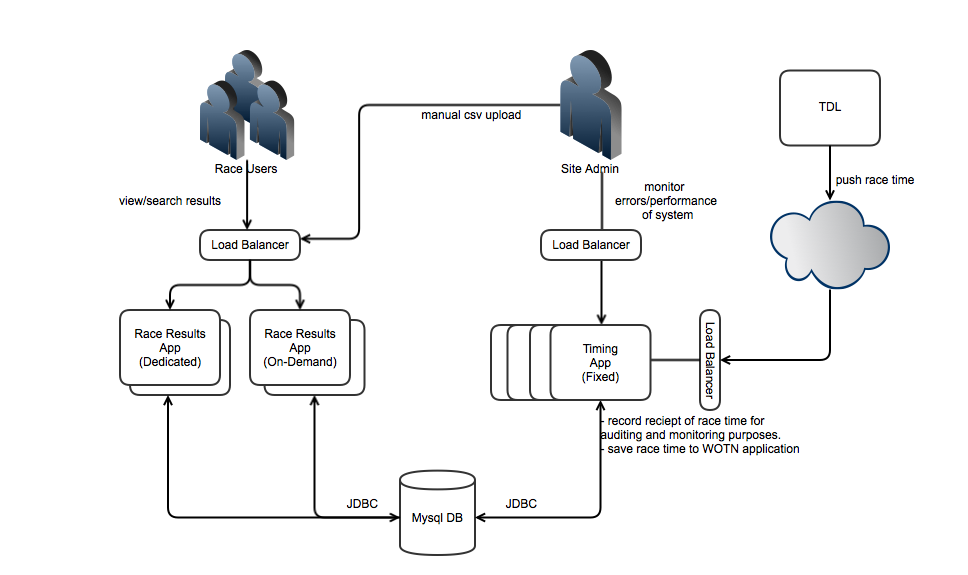
RFID timing data from the event will be supplied by the race vendor (assumed to be TDL) and displayed on the site.

## Proposed Solution

Re-use of We Own The Night 2013 code base, page layout, infrastructure and architecture for 5 markets. Enhance the solution to allow it to receive real-time race timing data. This will be supplied by TDL (to be confirmed).

The application will reuse the timing application that was developed as part of the Nike Women’s Marathon to process the race timing data.

## Logical High-Level Architecture Diagram



The race results site will be served by one or more Tomcat web servers hosted on Amazon Elastic Beanstalk, configured for auto-scaling. Race data will be stored in an Amazon RDS MySQL database in master/slave configuration.

The real-time race timing service will be hosted on a separate AWS Tomcat cluster to allow the separation of traffic and the ability to scale the servers independently.

TDL will push race times to a race timing service. The race timing service will record receipt of the timing data and then feed the result into the database for the results site to display to users.

TDL will push the following fields

* Bib number
* First Name
* Last Name
* Crossing Point (start, [1km, 2km, .., 9km], finish)
* Crossing Time
* Total Time
* Race Id
* Average Pace
* Name search opt-in

Additionally an administrator will be able to upload CSV race data file into the application to override results time received from TDL if necessary. This will be used to make post-race amendments not sent through the timing application interface.

## Design Refinements

* Clarify format of the results received from TDL.
* Clarify integration work required to receive results from TDL.
* If required, confirm format of manual CSV upload.
* Establish how performance testing of the TDL interface will be carried out.
* Establish the level of application caching required.
* Determine the level of work to adapt the existing Nike Women’s Marathon race timing service.
* Timing application could be extended to send via ExactTarget the race result emails.
* Clarify that RacePartner provide user opt-in status with regard to whether their result is searchable on the site by first/last name or not.

## Hosting Requirements

* What services are being called?
  + We will not be calling into any Nike services from this site.
* Is SSL required
  + The interface between TDL and the race timing service will be over SSL.
* Is this a Facebook app?
  + No
* What will the target URL be named (e.g. gameonworld.nike.com)
  + TBC
* Is Akamai required?
  + No but We Own the Night used CloudFront so it is anticipated that this will be utilized
* Performance Goals and Elasticity considerations
  + See Data Persistence section
* Expected Volumes (total and concurrent users)
  + WOTN will be 5 separate events, although 2 of these are on the same day. Based on previous similar campaigns, we can expect a traffic spike in which 70% of the attendees visit the site over the period of an hour.
  + As an upper limit, we do not expect more than 25,000 visitors over any given one-hour period. Based on the above hourly request count, we expect that we will need to serve around seven requests per second on average. Assuming a multiplier of three to derive a peak request rate, we should aim to support 21 requests per second. It should be noted that this multiplier has not been based on previous measurements of standard deviation or peak rates, so should be validated via performance tests and future observation.
* Availability (uptime/failover requirements)
  + Availability will be not less than that of the Amazon Web Services platform
* Outside dependencies (non-Nike services/technologies)
  + Race results to be provided by TDL. They will push race times into the application.

**Tech stack**

* Spring Framework
* Java
* HTML
* CSS3
* JQuery
* Ajax
* JDBC

## Data Persistence

**Race**

* Race ID
* Race Name
* Race Contact Details
* Number Of Runners
* Locale
* Race Gun Time (addition to schema from previous year)

**Race Result**

* Result Id
* Bib Number
* First Name
* Last Name (for security purposes may be truncated to just initial).
* Split Times (if available)
* Race Position
* Final Race Time
* Average Pace

If TDL support it, additional attributes may be added to the interface. For example, an opt in/out flag may be captured during registration which would indicate that the user does not want their name searchable.

Caching will be introduced to cache aggregated results to reduce load on the Database. Testing will indicate where caching will be needed.

**Data Schema**

TBC

## Security of PII data

We will be storing the first and last names of runners, both of which are considered PII data. This data will have been captured and provided by a third party, and we will not be using UPM IDs.

The first name and last name of runners will be encrypted when stored in the database. There will be no auto-completion facility when searching by runner name.

The RDS database and Elastic Beanstalk instances will be secured using Amazon’s usual mechanisms of access control restriction, and will only be open on port 80 for HTTP traffic.

## Risks & Dependencies

* Delays in setting up integration with TDL and being able to perform an end-to-end test across the whole system.
* The ability to adapt the Nike Women’s Marathon race timing service.
* If the race results data is delayed, its appearance on the site will also be delayed.
* AWS environments must be provisioned within the specified timeframe, allowing us to perform QA and performance testing.

## Summary

Since we are modifying an existing codebase, we have a medium to high degree of confidence in the project. The existing race timing system will allow us to monitor and diagnose the performance of the race timing system on race day.

## Addendum – AWS template

**QA & DEV environments** (one each for **wotn-results** and **wotn-timing**; small cluster [2 instances] for QA if possible to simulate prod testing)

AWS Hosting for **wotn-results** and **wotn-timing** (clustered)

Key/Value Pairs:

instance.type => m3.medium

elb.http.port => 80

elb.https.port => 443

jvm.max.perm => 512

jvm.mem.max => 2048

jvm.mem.min => 2048

autoscale.unit => Percent

autoscale.stat => Maximum

autoscale.measure => CPUUtilization

autoscale.lowerbreach => -1

autoscale.lowerthreshold => 40

autoscale.upperbreach => 1

autoscale.upperthreshold => 60

autoscale.az => 2

autoscale.breachduration => 5

autoscale.period => 1

autoscale.min.size => 2

autoscale.max.size => 2

healthcheck.page => /status.jsp

deploy.module => beanstalk::code

logging.enabled => true

email => alexis.lignereux@akqa.com, peter.abbott@akqa.com, alexander.dergachev@akqa.com

ami.id => ami-be07dad7

**MySQL Setup:**

Static ip address (used to connect from to administer MySQL instance)

Database size: **Single AZ RDS instance (db.m3.medium)**

Instance size: **16GB**

How large will the database be when mature? >**500,000 records**

What are the queries you will be running? **CRUD operations**

What storage engine will you be using? **InnoDB**

*key\_buffer\_size* **=> default**

*innodb\_buffer\_pool\_size* **=> default**

*tmp\_table\_size* **=> default**

*query\_cache\_size* **=> default**

*thread\_cache\_size* **=> default**

**Perf & Prod Environments**

AWS Hosting for **wotn-results** (clustered)

Key/Value Pairs:

instance.type => m3.large

elb.http.port => 80

elb.https.port => 443

jvm.max.perm => 1024

jvm.mem.max => 5632

jvm.mem.min => 5632

autoscale.unit => Percent

autoscale.measure => CPUUtilization

autoscale.lowerbreach => -1

autoscale.lowerthreshold => 40

autoscale.upperbreach => 1

autoscale.upperthreshold => 60

autoscale.az => 2

autoscale.breachduration => 5

autoscale.period => 1

autoscale.min.size => 2

autoscale.max.size => 4

healthcheck.page => /status.jsp

deploy.module => beanstalk::code

logging.enabled => true

email => alexis.lignereux@akqa.com, peter.abbott@akqa.com, alexander.dergachev@akqa.com

ami.id => ami-be07dad7

AWS Hosting for **wotn-timing** (clustered)

Same as above plus changes:

<…>

autoscale.stat => Maximum

autoscale.min.size => 4

**MySQL Setup:**

Static ip address (used to connect from to administer MySQL instance)

Database size: **Multi-AZ RDS Instance (db.m3.xlarge) with ability to setup Read Replica**

Instance size: **16GB**

How large will the database be when mature? >**500,000 records**

What are the queries you will be running? **CRUD operations**

What storage engine will you be using? **InnoDB**

*key\_buffer\_size* **=> default**

*innodb\_buffer\_pool\_size* **=> default**

*tmp\_table\_size* **=> default**

*query\_cache\_size* **=> default**

*thread\_cache\_size* **=> default**

**Sign-off:**

***Security*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_**

***Brand Architecture*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_**

***Infrastructure Architecture*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_**

***Data Privacy*: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_**

**Appendix**

*Personal Identifying Information (PII)*

• Full Name (last name, with first name or first initial, and/or middle name)

• Address (street address lines. City, State, Country ok if not connected to street address)

• Email Addresses

• Phone numbers

• Birth Date (full day, month, year combination)

• User PINs, Login, Passwords, Tokens containing personal information

• User Relationships/ Friends

• Photographic images of users (not avatars or other image substitutes)

• Non-aggregated latitude/longitude of activities or other fine-grained location data, e.g. GPS, routes. (not Locale, Country, State/Province)

*Personal Identifiers*

• UPMID

• CMDBID

• Acxiom ID

• HHID (Household Id)

• IP Addresses

• Order Numbers

• Analytics/Browsing data, if UPMID or other identifier included

• Device Access Tokens

*Partner Information*

• None