Server Architecture Design Diagrams

Document Status: DRAFT

Document Date: 03/27/2015

Owners:

Yiran Mao

Design Principle

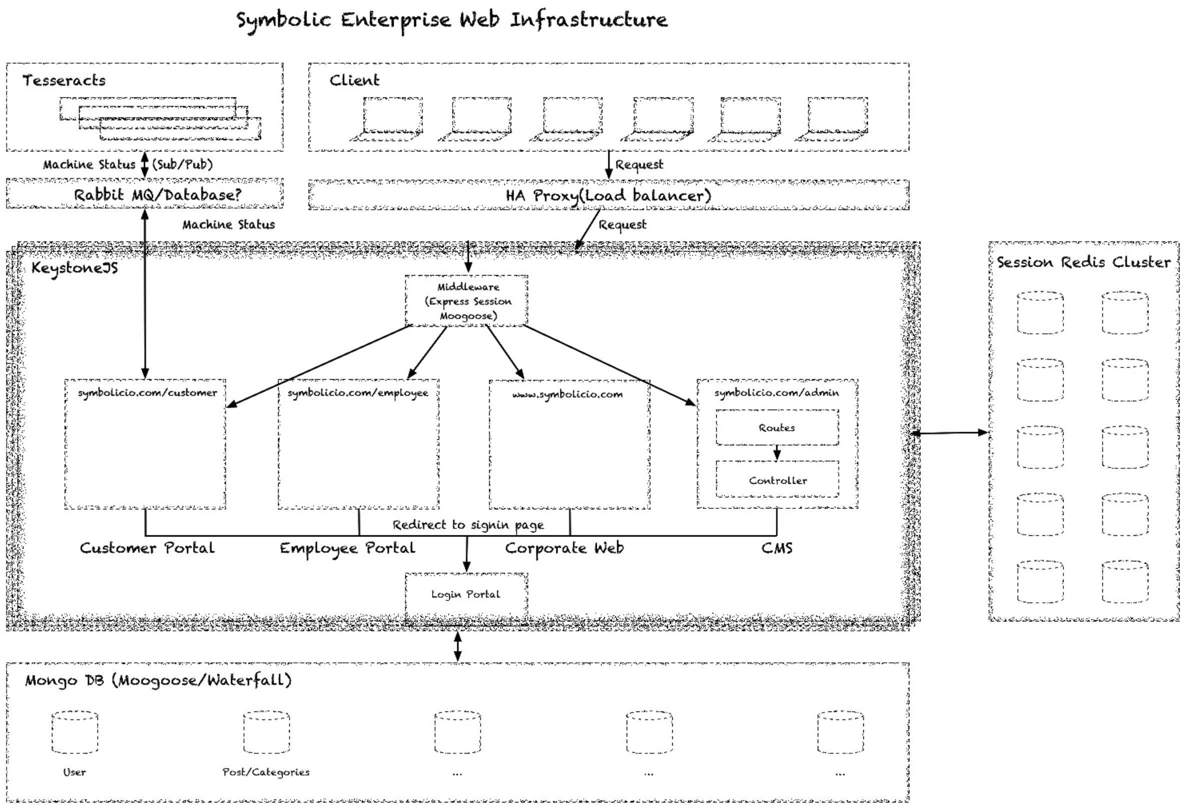
The server architecture design based on follow 6 principles, which are listed in priority order:

- Scalability
- Security
- Flexibility of development
- Easiness of deployment
- Easiness of development
- Flexibility of deployment

Server Architecture Diagram

Overview

The following Diagram describes the server architecture layout for symbolic corporation web infrastructure.



The entire application will be built upon KeystoneJS framework. The KeystoneJS framework already provides us the user authentication, view rendering, content management capability, which to a large extent simplifies our development process. We will develop the modules defined in <Application Module Design> upon the current code base in phase II.

The application communicates with Redis database for the session persistence, which preserves the session ID and user ID relationship for per logged-in user per device. It also handles the automatic expiration and clear up for the session storage.

The application relies on mongo database for long-term data persistence, which is suitable for less structured information like corporate website content. However, there could be possibility we rely on other relational database later on.

In-system Entities

Application

The application will contain 4 parts: corporate website, customer portal, employee portal and content management system.

The corporate website will serve as the main communication protocol for potential customers, investors and general public that are willing to know the products and services provided by Symbolic IO Corporation. It is also the protocol for white paper release and customer support.

Content management system is the internal management tool for the corporate website. The system allows technology writers, editors to work on the information release by the corporation and the administrator will have the privilege to review the content before it is released on the corporate website.

The customer portal will give Symbolic IO Corporation customers capability to see the running status for their devices, purchase licenses for extra device capability or consult directly with our customer service team. It may be wired up to retrieve information from Tesseract through SQL database or rabbitMQ directly depends on the information we want to provide to customers through customer portal.

The employee portal will be provided to Symbolic IO Corporation employees that allow them to see corporate-wide news, benefits, and job openings. The outline for the portal is not yet specified and will be further developed in a separate develop phase.

HAProxy

HAProxy works as the load balancer in the system that can help the system scale in the future. We probably won't need the proxy server to handle the request balance yet but will definitely get it in the state ready to deploy.

Redis Cluster (Database)

The redis cluster will store the session information in key value pair. The session ID will be stored in user's cookie for later retrieval of the user identification in the following request once they logged in to the system.

Mongo Cluster (Database)

The mongo cluster will store all the information needed by the corporate website, including categories, posts, users. We manipulate the database by Mongoose ORM framework that provides us a concise way to define type and do data validation before data is actually stored in the database.

Peripheral Entities

Rabbit MQ

Rabbit MQ is current communication of the Tesseract and the node server for management portal within the box. We may create a new channel for out-world communicate on the box and the potential structure might be Tesseract communicate with a relational database like MySQL and the customer portal ret

SQL(databse)

The SQL database is the alternative way for device information retrieval for the customer portal.

Tesseract

Tesseract will be the first and eventually one of the products that we can retrieve the information and purchase license from customer portal. The system has also planned to develop a call home protocol for Tesseract once the device raises traps that worth Symbolic IO Corporation service team or customer's notice.

Long-term Scaling Plan

Application

The application can be replicated over paralleled servers and use HAProxy as the load balancer that redirects the requests to one of the servers that eventually processes the incoming requests.

We may potentially rely on Docker to make the deployment process smoother.

Session Management

The solely redis database can essentially be replaced by a redis cluster. The redis cluster can store the session information distributively according to the session ID hash value to handle enormous session requests. One possibility to implement the cluster is using the session key and hashes it to the corresponding machine within the cluster for user authentication.

Long-term Data Storage

The mongo database and potential relational database all have well-established ways to scale. It can use index to scale-up the performance of information retrieval on each database or scale-out by different content it stores.