TECHNICAL ARCHITECTURE

Example

VERSION: 0.8 REVISION DATE: 20.07.2018

Approval of the Technical Architecture indicates an understanding of the purpose and content described in this deliverable. By signing this deliverable, each individual agrees with the content contained in this deliverable.

Approver Name	Title	Signature	Date

Contents

Section 1 DOCUMENT SCOPE

This document describes the Technical Architecture of the Example System that implements the functionality and satisfies technical, operational and transitional requirements described in the Functional Specification, 18 Apr. 2018.

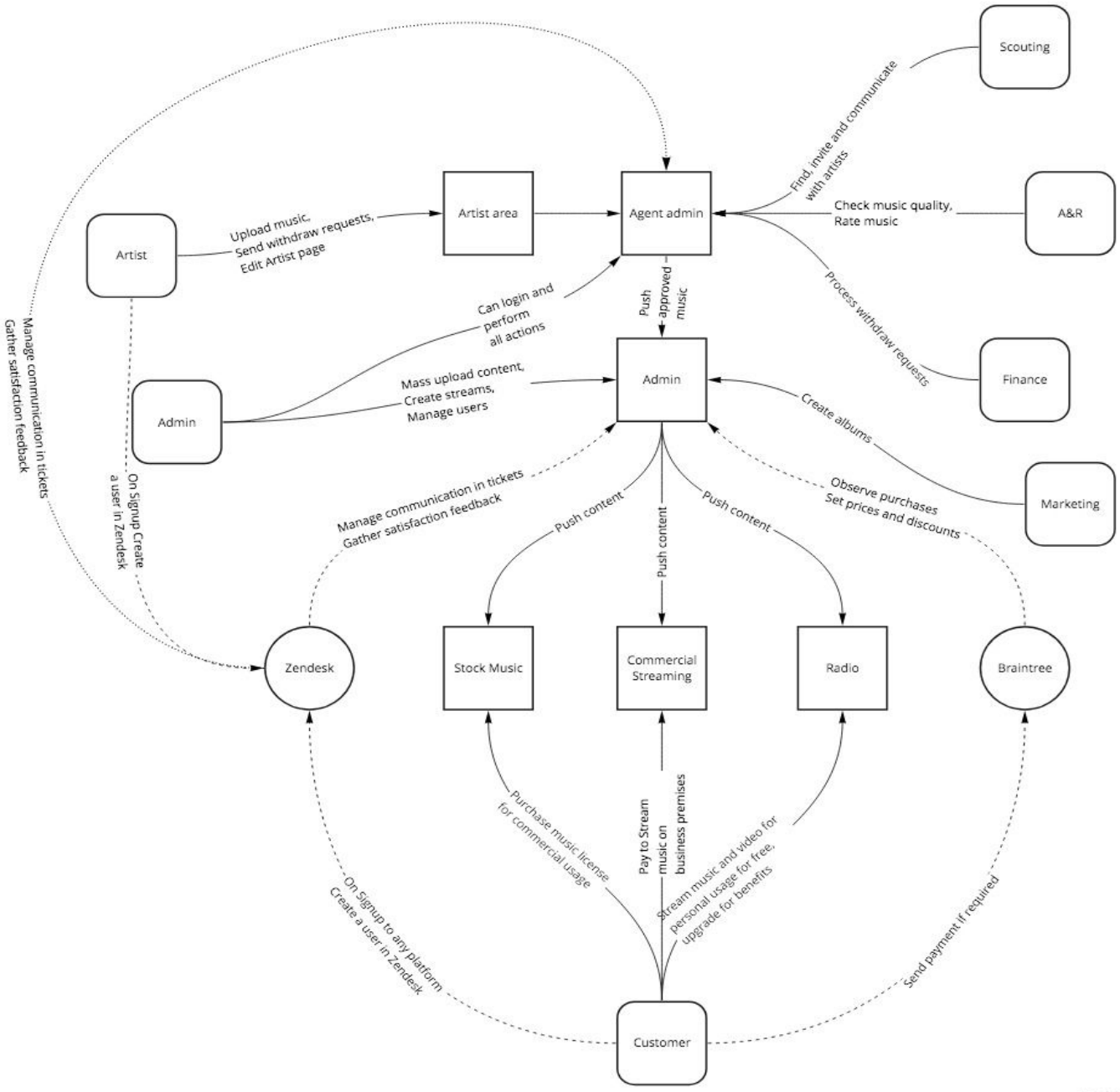
The goal of this Technical Architecture is to define the technologies, products, and techniques necessary to develop and support the system, and to ensure that the system components are compatible.

This document will also:
Define baseline sizing and performance requirements;
Identify the hardware and software specifications for the Development, Testing, QA and Production environments;

Section 2 OVERALL TECHNICAL ARCHITECTURE

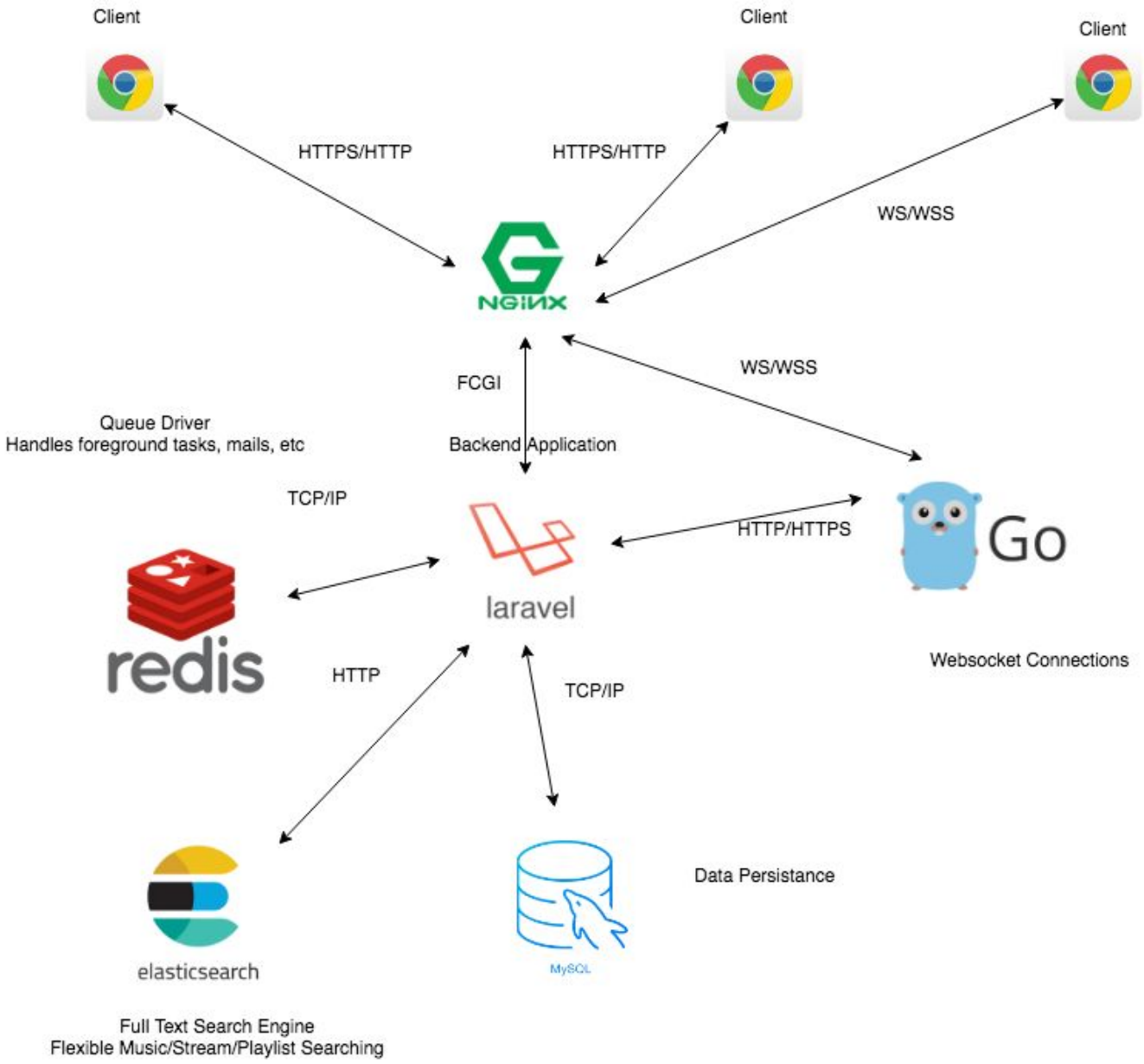
2.1 System Architecture Context Diagram

The **System Architecture Context Diagram** show a system, as a whole and its inputs and outputs from/to external factors.



2.2 System Architecture Model

The **System Architecture Model** represents the various architecture components that comprise the system, and shows their interrelationships.



2.2.1 Overall Architectural Considerations

Technical Platform decisions:

- Example applications will be deployed as docker images in kubernetes cluster in Google Cloud Platform.
- Kubernetes workers will be configured in auto-scaling mode. It provides flexible scaling to met current load.
- High performance message system (Redis) will be used in cluster mode.
- Will be used MySQL relational database in high performance and high availability configuration.
- Will be used ElasticSearch engine to collect application logs and to provide high speed searching.
- All infrastructure will be configured and described with Infrastructure as Code principle. For this purpose will be used Terraform and Ansible.
- Building and delivering applications will be proceed with Jenkins and Ansible.

Security

Basic security behaviors:

- Authentication: Login using at least an email and a password
- 2F-Authentication: User login verifying with additional step via third party services like Google, etc.
- All connection will be encrypted with TLS.
- Authentication: Login using at least an email and a password
- Authorization: according to their profile, online user must be granted or not allowed to receive some specific services (possibility to add advertisements to the existing streams, etc.)
- Confidentiality: sensitive data must be encrypted if any (credit card payments) .
- Safety: Credit card data must not be kept at a local database.
- Data integrity : Data sent across the network cannot be modified by a tier
- Auditing: Some sensitive actions can be logged

Monitoring

- Kubernetes cluster will be monitored with built-in Google monitoring services and with Prometheus+Grafana services.

Persistence

Data persistence will be addressed using a relational database MySQL and Eloquent Object Relational Mapping capability will be used.

Reliability/Availability

High availability is required since there are monetary issues related to the systems availability. All users should not be disappointed. The system’s high availability will also ensure customer satisfaction and loyalty.

Targeted availability: 24 hours a day, 7 days a week

Performance
Search queries should return 90% of the time below 5 sec.
Credit card payment transaction should finish in 10 sec.

Internationalization
Initially the system will support Polish, English and Italian. It should be easily modifiable to extend language support.

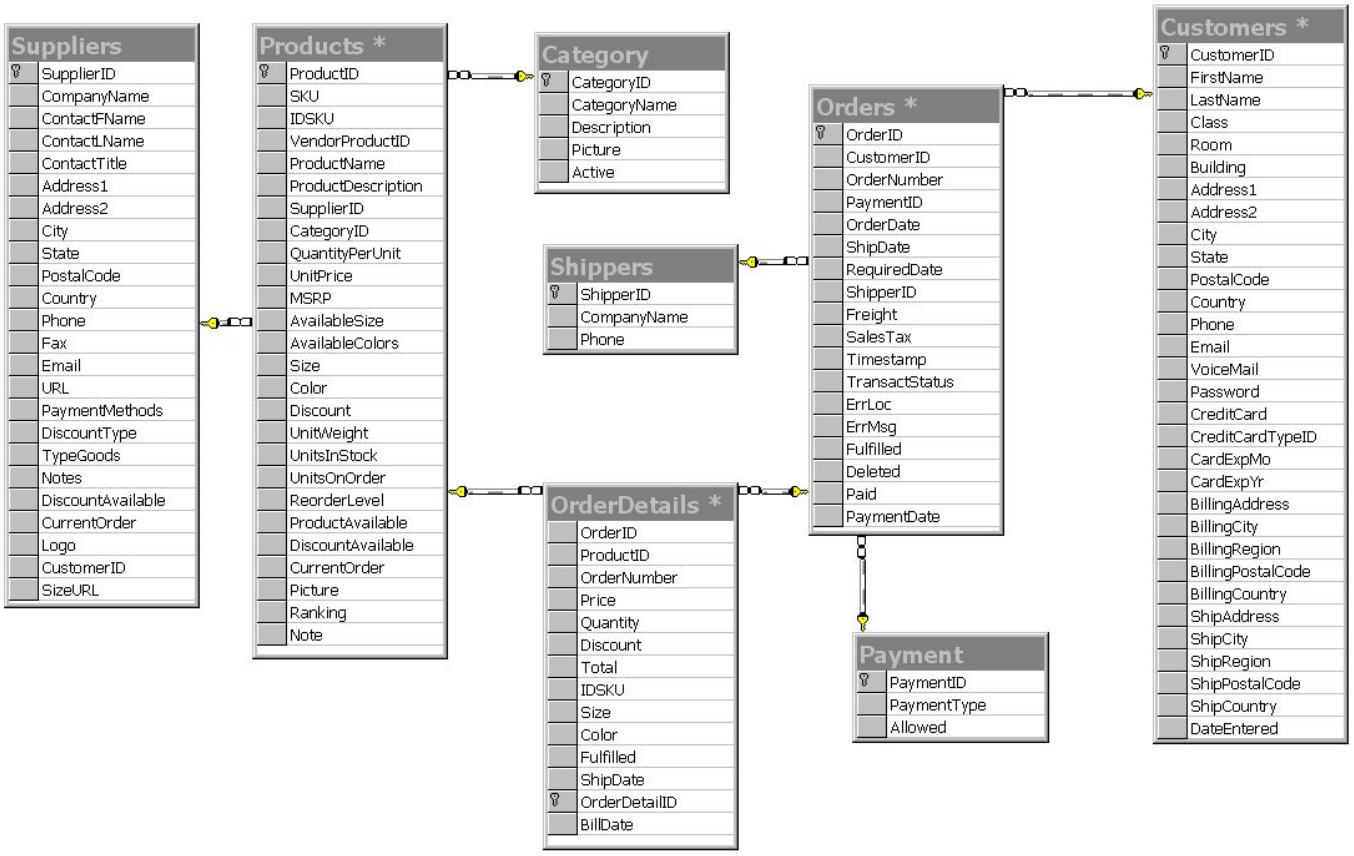
2.3 System Architecture Component Definitions

2.3.1 System Architecture Component

The **Architecture Component Definitions** section provides narrative describing and explaining each architecture component in the System Architecture Model, and identifies specific elements that comprise that component in this system.

Architecture Component	**Component Elements**
Client Application	Framework **Angular 6**
Backend Application	Programming Language **PHP 7.2.2** Framework **Laravel 5.6** Unit Tests **PHPUnit 7.2.6** Package Manager **Composer 1.5.0** Development Tools: **Swagger UI**, **Mysql Workbench Model**, **PHPUnit Code Coverage Reports**
Websockets	Programming Language **Golang 1.1** Third Party Libraries: **GoRequest, Gorilla Websocket** Websocket connections handler
Elasticsearch	Elasticsearch **6.1** Full Text Search Engine to perform and combine many types of searches - structured, unstructured, geo, metrics.
Database	Relational database Mysql **5.7**
Redis	In-memory data structure store, used as a database, cache and message broker.
Web server	**Nginx 1.15.0**

2.3.2 Example Project database schema



Section 3 System Construction Environment

3.1 Development Environment

3.1.1 Developer Workstation Configuration

Hardware:
core i5 6600K, 8-16GB RAM, 240 SSD disk. 100 mbit/s internet.
OS: Ubuntu 16
Software:
mailhog/mailhog:latest
mysql:5.7
redis:5.0-rc-alpine
elasticsearch:2.4.6-alpine
nginx 1.15.0
Backend software modules:

```
"php": "7.1.3",
"barryvdh/laravel-dompdf": "^0.8.2",
"drawmyattention/xerolaravel": "1.0.*",
"elasticquent/elasticquent": "1.0.6",
"fico7489/laravel-pivot": "*",
"fideloper/proxy": "^4.0",
"guzzlehttp/guzzle": "^6.3",
"huddledigital/zendesk-laravel": "^2.3",
"intervention/image": "^2.4",
"james-heinrich/getid3": "^1.9",
"laravel/cashier-braintree": "~2.0",
"laravel/framework": "5.6.*",
"laravel/socialite": "^3.0",
"laravel/tinker": "^1.0",
"league/csv": "^9.1",
"league/flysystem-aws-s3-v3": "~1.0",
"orchestra/parser": "~3.0",
"pda/pheanstalk": "~3.0",
"socialiteproviders/instagram": "^3.0",
"spatie/laravel-newsletter": "^4.2",
"superbalist/laravel-google-cloud-storage": "^2.0",
"textalk/websocket": "1.0.*",
"tjphippen/docusign": "0.3.*@dev",
"true/punycode": "~2.0",
"tymon/jwt-auth": "0.5.*",
"zendesk/zendesk_api_client_php": "^2.2",
"predis/predis": "~1.0"
```

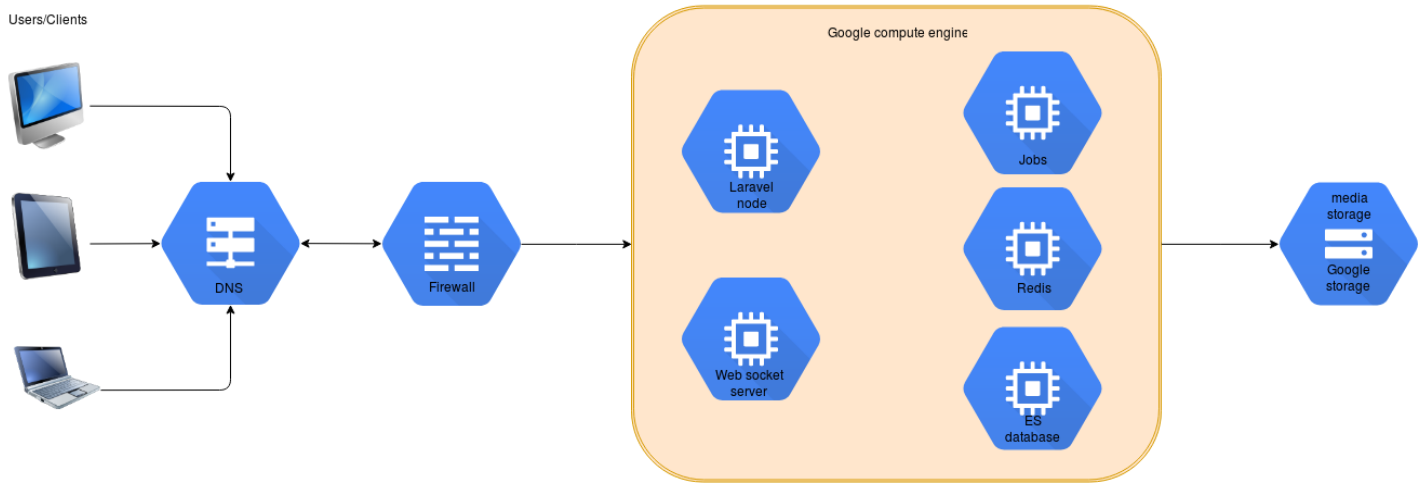
Frontend software modules:

Angular CLI: 6.0.8
Node: 9.9.0
rxjs: 6.2.1
typescript: 2.7.2
webpack: 4.8.3
Angular: 6.0.6
Angular libraries: animations, common, compiler, compiler-cli, core, forms, http, language-service, platform-browser, platform-browser-dynamic, router
Angular packages:

@angular-devkit/architect	0.6.8
@angular-devkit/build-angular	0.6.8
@angular-devkit/build-optimizer	0.6.8
@angular-devkit/core	0.6.8
@angular-devkit/schematics	0.6.8
@angular/cdk	6.3.0
@angular/cli	6.0.8
@angular/material	6.3.0
@ngtools/webpack	6.0.8
@schematics/angular	0.6.8
@schematics/update	0.6.8

3.1.2 Development infrastructure configuration

single node configuration:

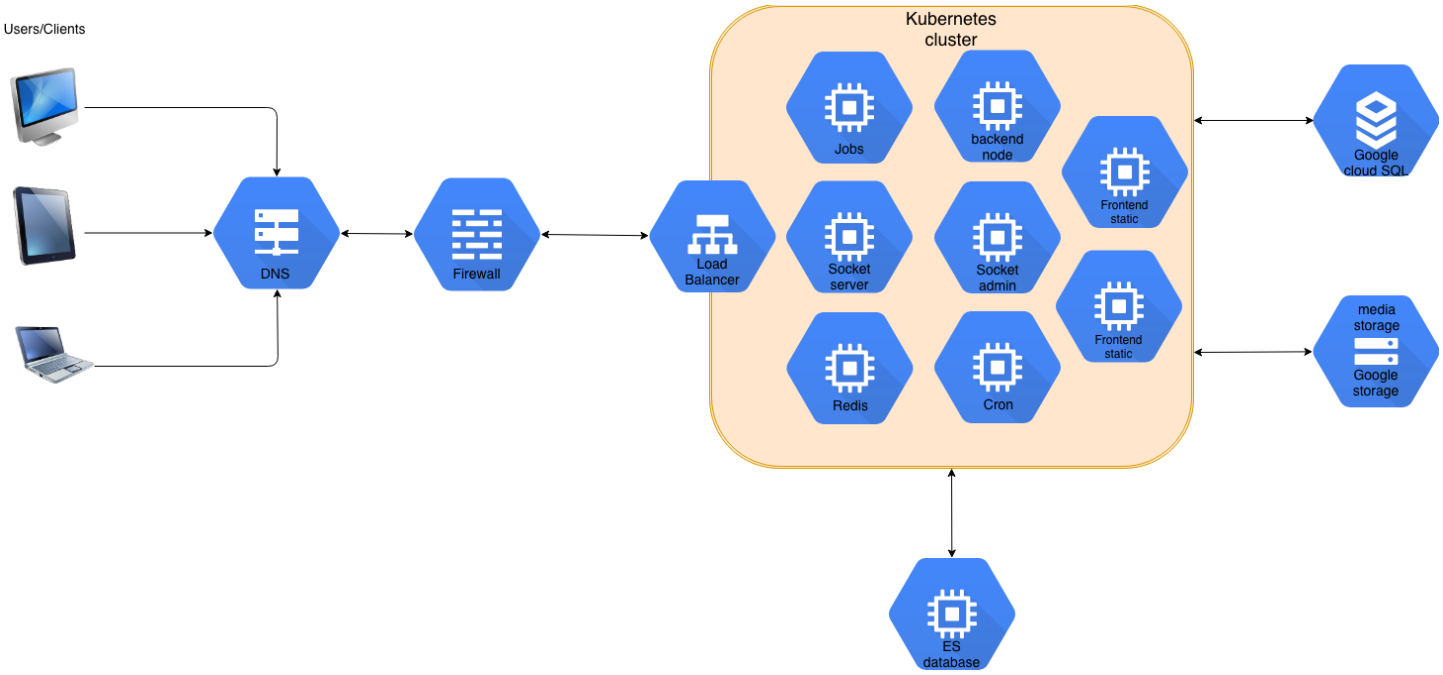


3.2 QA Environment

3.2.1 QA Workstation Configuration

Hardware:
core i5 4690, 8GB RAM, 240 SSD disk. 100 mbit/s internet.
OS: Ubuntu 16
Browsers:
Chrome 67; Firefox 61.0.1
Browserstack
Postman v6.1.4
Screenshots: Shutter 0.93.1;
Video recording: SimpleScreenRecorder 0.3.11

3.2.2 QA infrastructure configuration



3.3 Acceptance Environment

3.3.1 Production infrastructure configuration