Architecture Overview

March 2022

LOCUSVIEW

Agenda

01 Mobile & Web

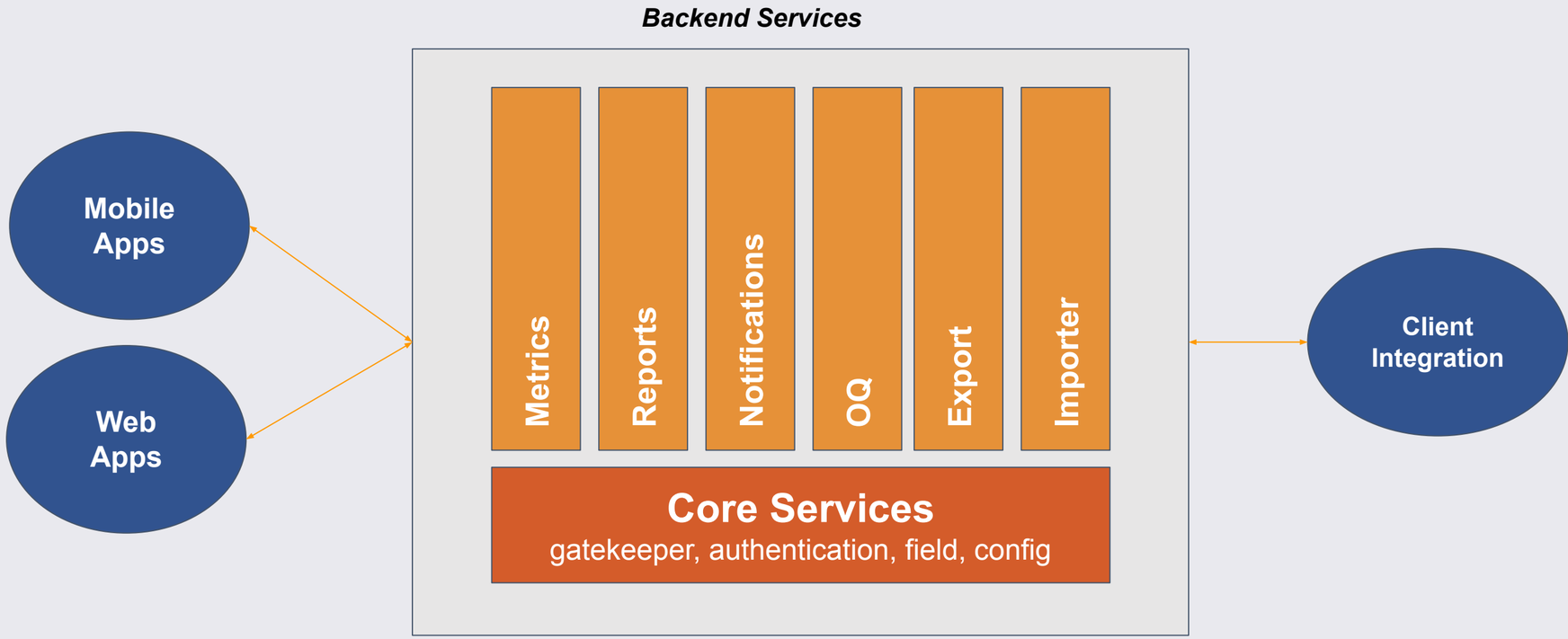
02 Backend Architecture

03 API Structure

04 A bit about Cyber Security & Environments

Overview

- Locusview SaaS platform is hosted on AWS, and serving our customers from a unified cloud infrastructure.
- Our mobile and web applications, as well as our clients, interacts with the backend services over Rest APIs.



Locusview Mobile

- **Cross-platform mobile application**
  - 95% of the code is shared between the different platforms (iOS\Android\Windows 10)
  - Custom plugins for OS related operations (public & proprietary ones)
  - Based on Apache Cordova\Ionic Capacitor.
- **Full offline support**
  - Internal relational DB based on SQLite
  - Data sync process
- **Configurable**
  - Built to support ongoing customers needs

Locusview Mobile (Cont.)

- **SQLite**
  - Internal relational database
  - Stores data for offline access
- **External Devices**
  - Different connection protocols (fusion, barcode, gps, RTK, LRF)
  - Manage multiple BT\BLE connections

Locusview Mobile (Cont.)

- **Main Languages & Tools**
  - Apache Cordova
  - Ionic Capacitor
  - Angular 12
  - Ionic 4
  - HTML5
  - SCSS
  - SQLite
- **Supported OS**
  - iOS (Deployment over VPP)
  - Android
  - Windows 10

Download on the App Store

GET IT ON Google Play

Download from Windows Store

Locusview Web

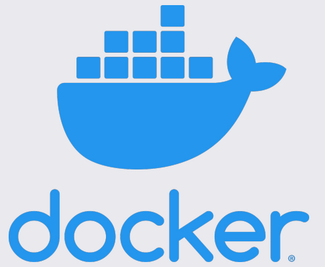
- Used for configuring the entire system, plan work, review & approve the data that was collected by the field users.
- Cloud based web application
- Shared JS code with the Mobile app
- Hosted on **AWS CloudFront**
- Main Tech:
  - Angular 13
  - TypeScript
  - HTML5

Backend Architecture

- **Microservices** Architecture
- Our backend is built of small services, each one **responsible for a specific task**
- Each service manages its' own storage.
- Services are built & deployed **independently**.
  - Using the same design principles, 3rd party company and building blocks.
- Communication between services is done using APIs.

Backend Architecture

- Each service can scale independently as needed.
- Deployed using **Docker** & AWS **ECS**.
  - Scalability
  - High Availability

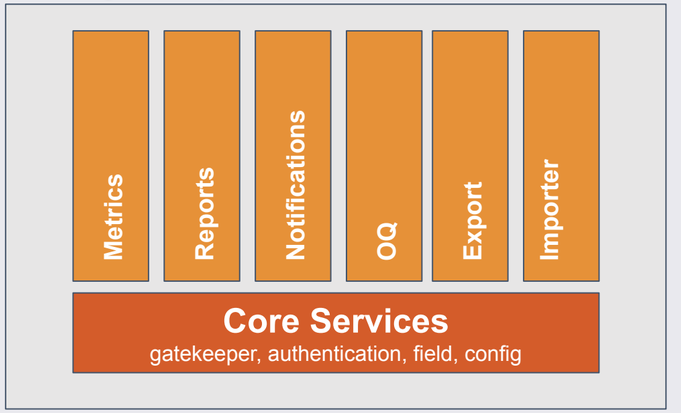


Backend Architecture

- **Common platform artifacts**
  - Central location for shared code
  - Used across all our services
- Main Languages & Tools
  - Java & Spring
  - Dropwizard
  - jOOQ & Liquibase
  - Docker

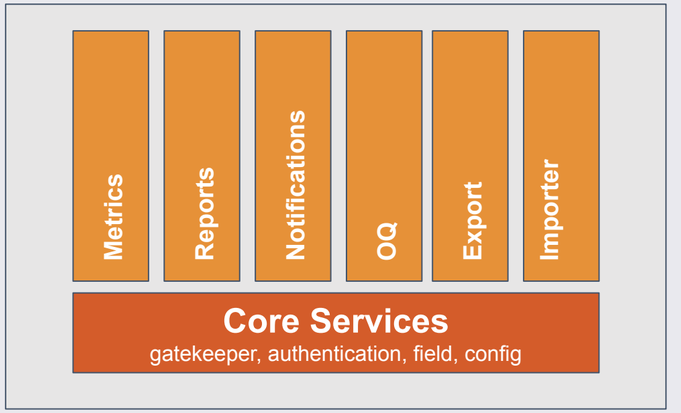

Core Backend Services

- Our core backend services are the fundamental building blocks of the platform. Other services relay on these core services in order to operate.
- Following is the list our our core services.
  - **Gatekeeper** - The gateway to our backend. The gatekeeper is responsible for verifying that an incoming request is authenticated, and to manage multi-tenancy and auditing related aspects
  - **Authentication** - The service that is responsible for users and groups management, authentication and authorization.



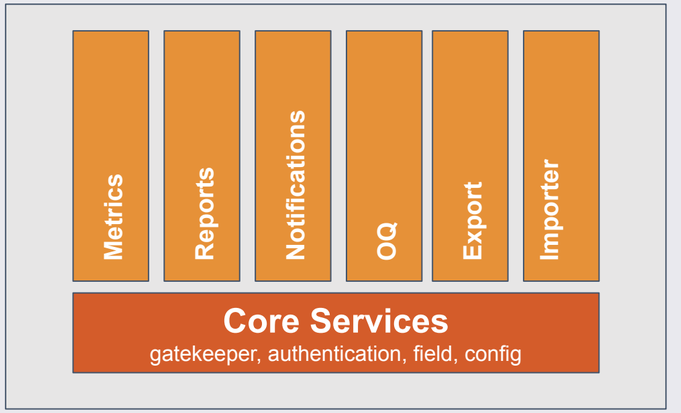
Core Backend Services

- **Configuration** - This service is responsible for managing our core platform configuration - main objects structures and relationships.
- **Field** - The service that manages the data that is collected in the field (projects, forms, assets). This service is the source of truth for this data, and many of our other services consumes this data.

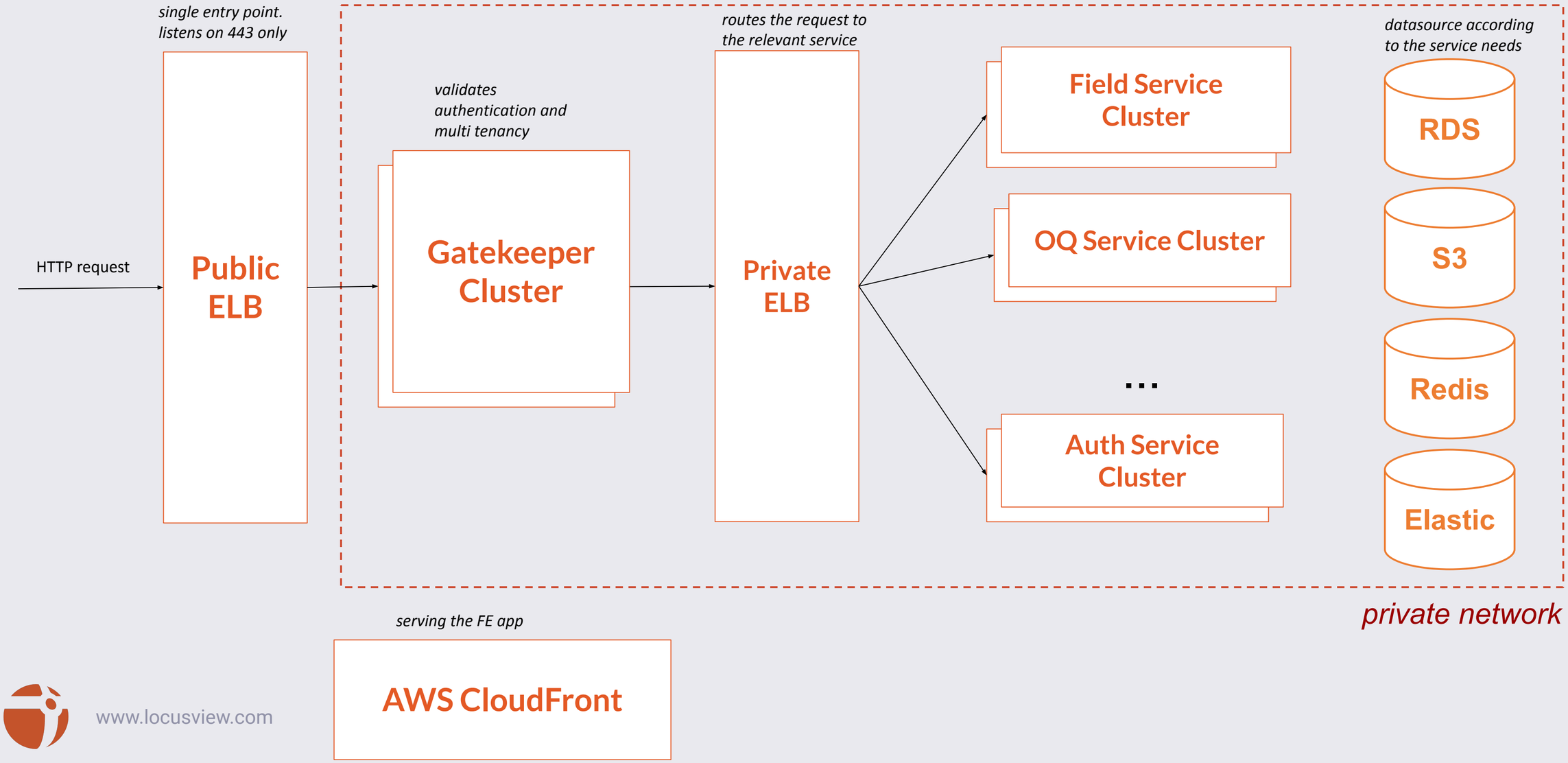


Additional Backend Services

- **Notifications** - Responsible for sending email notifications to our end users
- **Metrics** - Perform data aggregation for field data
- **Reports** - Generate report for projects data
- **OQ** - Validates crew personnel for their qualifications
- **Export** - Service that handles data ETL to our client systems
- **Importer** - Brings existing GIS layer from our clients and makes it available to Locusview users.

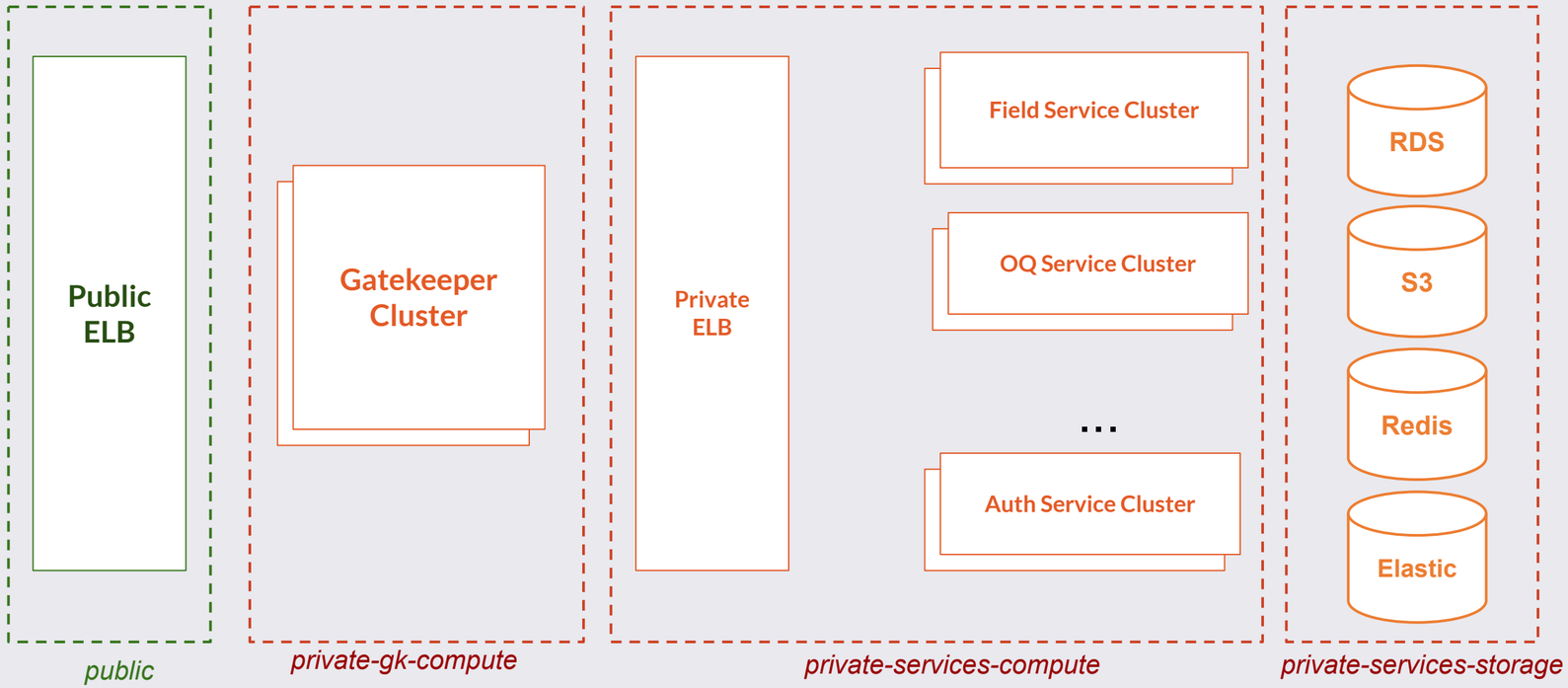


High Level Backend Diagram



High Level Network Structure

- Services and storage are located on a private network without any public interface.
- Accessing these resources is only possible thru a VPN.
- Our gatekeeper has a dedicated subnet since it handles unauthenticated requests.
- Each subnet is “splitted” between AZs



Data Storage

- We have multiple options for storing data. Each services can choose its own method based on the service needs.
- All sources should support multi tenancy and high availability.
- **RDS *(MySQL)*** - Our main data storage for persistent relational database.
- **Elastic Cache *(Redis)*** - Our distributed cache. Used for storing frequently accessed data.
- **S3** - Our file storage. Stores all binary data (photos, pdf files etc.)
- **OpenSearch** - Our data aggregation service

Inter-Service Communication

- Our services communicate with each other using HTTP requests. These requests always sent to the internal LB that handles the routing.
  - Async messaging option will be added to the platform on H2 2022
- Services never share or access each other’s data sources. For example, a service can’t access other services database.

API Structure

- Our API design follows RESTful architectural style. In Short -
  - We’re using HTTP verbs to describe the operation we’d like to make on an object (`GET`, `POST`, `PUT`, `PATCH`, `DELETE`).
  - The API is stateless.
  - The API path is built based on the resources structure (for example `/api/v1/projects/{id}/materials`)
  - Filtering and sorting based on Query Params (for example - `/api/v1/projects?sortBy=lastUpdated&order=desc&stepTypeId=3,7`)

API Structure

- Standard error handling - `401`, `403`, `422` HTTP Responses

```
{
   "field": "name"
   "errorCode": "TOO_LONG",
   "description": "Project name is too long",
   "parameters": [64],
}
```

- Authentication & Rate limit

Multi Tenancy

- Compute resources are shared between clients
  - Load Balancers, Machines, Clusters etc.
- Data Storage is segregated between clients depending on the storage limitations.
  - Platform components enforces data segregation. Engineers attention is required.
- Each client is associated with a sub-domain (e.g https://client1.locusview.io)
  - The sub domain is associated with an `organizationId` that uniquely identifies a client.
  - Platform components verify data segregation based on the `organizationId`.
- More on multi tenancy here

A bit about Cyber Security

- Quarterly Penetration Testing by a third party
- A single point of entry to our backend services
- IDS / IPS
- WAF rollout in Q2 2022
- Hardened instances
- Vulnerability scans of external library
- Separate network and accounts
- Minimal access to production resources

Development environments

- For development purposes, we have multiple environments:
  - **Labs** - An environment that holds our next major release during the development phase (master branch)
    - Each developer is assigned with an organization on our labs (e.g https://matan.nortecview.com)
  - **Staging** - An environment that holds the next major release during regression testing
    - We use clients' configuration for testing purposes
  - **Custom X** - Environments that are used for testing sensitive \ large features when needed
- Full environments list can be found here.

Production Environments

- For development purposes, we have the following environments:
  - **Production** - The main env of a client. Used to collect **real production data**. (e.g https://client.locusview.io)
  - **Test** - An environment that is used for testing purposes (e.g https://clienttest.locusview.io)
  - **Dev** - An environment that is used for configuration updated and clients' integration testing. (e.g https://clientdev.locusview.io)
  - **RC** - Used by our clients to test our next major releases (e.g https://clientrc.locusview.io)
- All the above environments are running on our **production infrastructure** and should be treated with caution.
- **Quick cheat** - Labs environments end with **nortecview.com**, production ends with **locusview.io**

Versioning

- **Major Release** - A release that contains a significant amount of features.
- **Minor Release** - A release that contains a small amount of features or an accumulation of bug fixes.
- **Hotfix** - A release that contains a critical fix that must be deployed immediately.
- Each release is assigned with a version using the following format:

  <major>.<minor>.<hotfix>.<build number>

  For example: **15.1.0.220**
- Full details can be found here.

Questions?

LOCUSVIEW