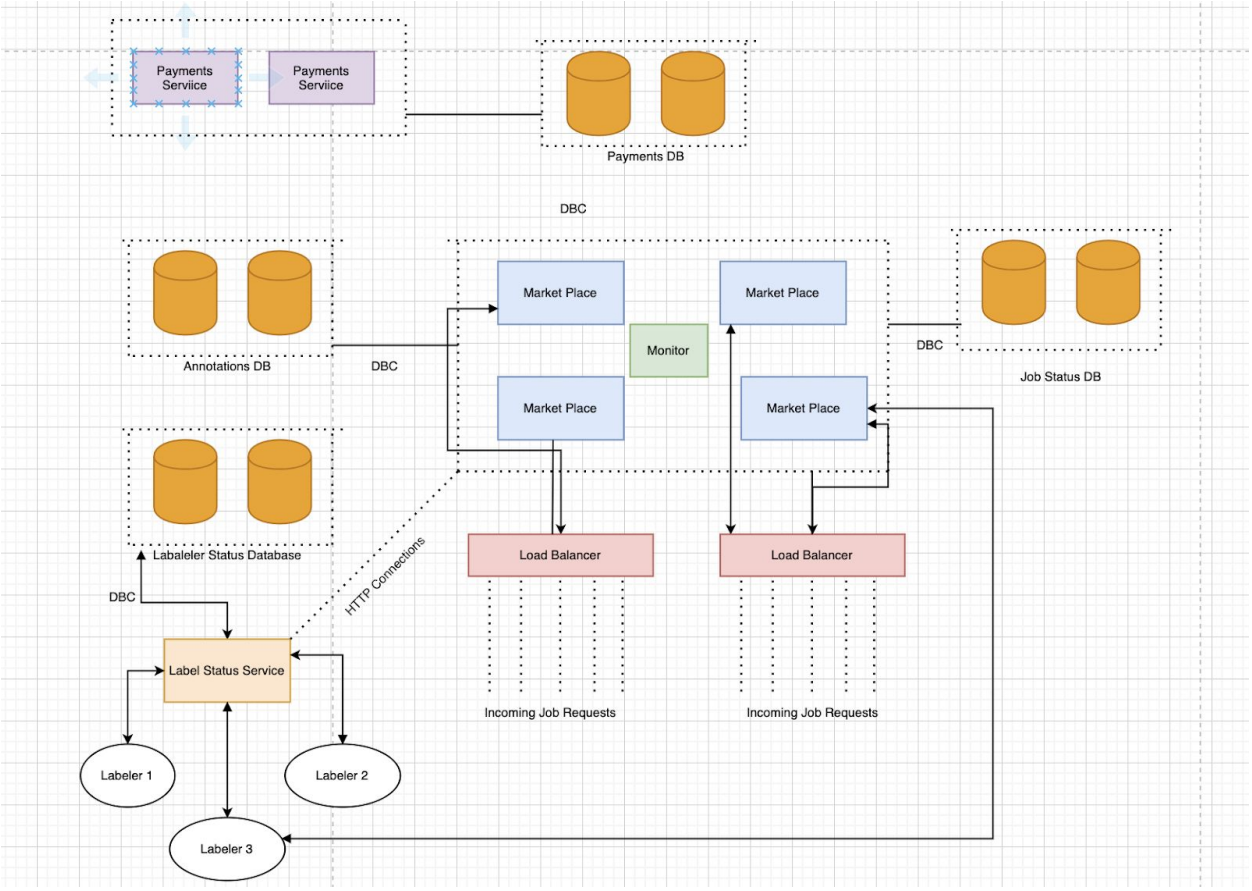
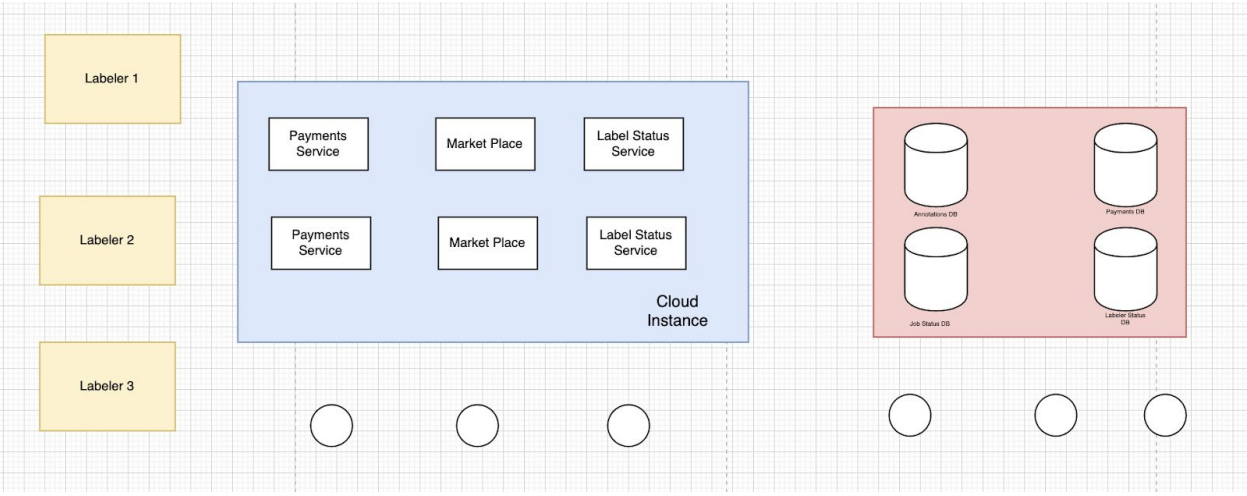
Main Architecture

How would you build a system to receive thousands of tasks per day, coordinate between labeling companies on the status of a task, and ensure jobs are complete. In other words, build a system where labeling partner's will interface with Alectio, and store the annotations.



Runtime View



Deployment View

The entire system is split into microservices so that the system can be scaled to handle thousands of requests. There are a number of market place instances that can perform the same tasks. The market place instances are stateless, and they can be horizontally scaled whenever there is an increase in load.

There are multiple load balancers that are present in the L4 and L7 layers. The L7 load balancers are present across regions balance requests to different instances of the market place system. There are also a number of L4 load balancer which are present in the application layers that balance the tasks needed to be done by the market place from the ML optimization engine.

There is a job status service that constantly connects to various labelers and checks if they are available or not. The job status service updates the availability of each labeler in the labeler's status database.

There are multiple databases that the market place systems have to interact with, there is a database system for annotations that is maintained, these databases are used by the market place system to be store the various annotations that are provided by the label makers. Each instance of a marketplace gets the annotations from the labeler and stores it in the annotations database.

There is a database that stores the status of jobs that are sent to the label makers. The marketplace instances check these databases and keep track of all the jobs given to various labelers and ensure that have been successfully completed. The market place

instances periodically check the job status DB and check up on label requests that are still pending.

There is a database that stores payment information of all the payments that have been made with each labeler. The payment service manages the payments with the labelers

All the databases have a hot standby in case any of them go down, the replica instance can instantly get running with no downtime or data loss.

Sequence of events

1. Load Balancer receives job requests and forwards the requests to market place instance that has the least amount of load
2. Label Status Service keeps connecting to labelers and updates the labeler status DB with the labelers that are currently available.
3. Market place instance receives the request and checks the labeler status database to see which labeler is available
4. Market place instance send a job request along with the instructions to the labeler and updates the job status DB
5. When the market place instance receives a request that the labeling has been completed, the market place instance transfers the annotations from the client to the annotations DB.
6. The market place instance updates the job status DB
7. The payments service updates the payments database after payment has been made with the labeling company.

Fault Tolerances - How would you keep track of the system to not proxy the job to the labeling partner, and what instrumentation tools would you use to make sure one of our servers are not down due to memory usage, CPU usage, etc

We can make use of server health check tools to determine spikes in memory or CPU usage. There are various health check tools that are available in the market to check for server health such as Appdynamics, Splunk, Data Dog etc

If the applications are run in containers such as Docker, it is easily possible to monitor the health of the containers and increase resources if necessary.

Follow up: How would Alectio ensure a labeling partner is “alive”, what steps need to happen so that Alectio re-routes the job to choose another labeling partner?

- There is a monitor application that is running to keep checking whether the marketplace applications have been up and running or not. The monitor application runs a heartbeat with each of the marketplace instances and keeps checking whether they are alive or not. If there is a break in the heartbeat, it instantly means that the instance is down, and it can either restart the instance or bring up another instance to handle the load.

Follow up: What structure and formats will the logs be stored if any?

- The logs would be stored in log files. The logs from all the instances could be aggregated using a data aggregator such as Splunk. The log files of each date could be compressed and a new log file is created. All the statements in all the log files contain timestamps for better traceability.

Security

What tools and methodologies would you use to ensure the communication between Alectio and a Labeling Partner is secure and encrypted, as shown in Figure 2 ?

- HTTPS protocol can be used to ensure that the communications between Alectio and the Labeling partner are encrypted and secure.
- Using Public Key Infrastructure to ensure that the data between the two systems is encrypted.
- Checksums can be added to ensure the integrity of the data.

Follow up: How would Alectio prevent DDoS attacks, and what other malicious attacks will Alectio be able to prevent?

Alectio can prevent DDoS attacks by adopting the following methods

- Firewall

  There could be a firewall which can be set up that can filter the incoming requests based on various parameters

- Whitelisting

  Alectio could whitelist only the labelers and the other vendors with whom they would like to communicate so that only authorized connections are received from other systems.

- Monitoring Traffic

  Traffic should be monitored on a regular basis so that it is easy to identify normal and abnormal traffic

Alectio should prevent the following kinds of attacks as they are possible.

- **Direct Access Attacks:** Could be avoided by using firewalls
- **Eavesdropping:** Could be avoided by using encryption

Follow up: How would Alectio ensure authentication and authorization with a Labeling Partner. If Alectio uses a Public Key Infrastructure model, what are the advantages and disadvantages over a session token

Advantages of PKI over session token model

- Security: Public key infrastructure model is way more safe and secure than the session token.
- Ease of Maintain: Once the initial setup is done, PKI is very easy to use and maintain, because the algorithm is constant, and only the keys are changed

Disadvantages of PKI over session token model

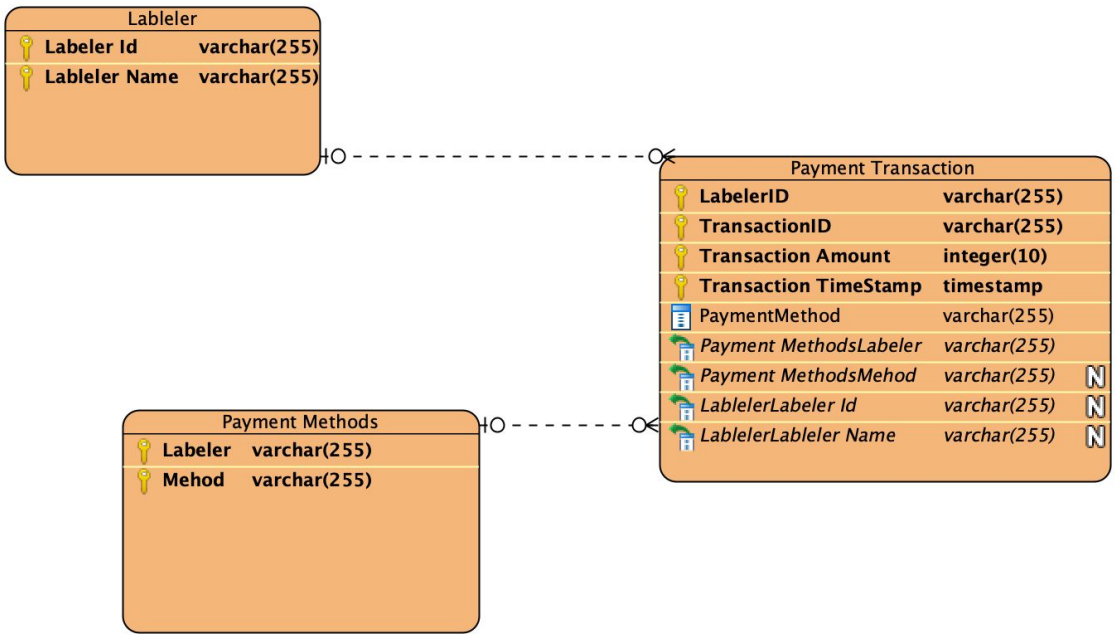
- High Overhead: Public Key Infrastructure model has a heavy overhead when compared to a session token model. It takes a heavy amount of resources to set it up,
- Speed: Public Key Infrastructure model is much slower when compared to the session token model.

Follow Up: Where would Alectio store the job transactions?

There is a relational database that stores all the payment transactions between Alectio and the labeling partners. Each payment made to a partner, the transactions are recorded in the payments database.

Follow Up: How would you design a table to show the user the aggregated cost for each labeling partner

Alectio has chosen for them to complete a task?



The above diagram represents the table for storing the payment information. The labeler table contains the label company information. The payment transaction table contains all the information of a payment transaction between Alectio and a partner.

We can calculate for each labeling partner by doing a simple join with the Labeler table and Payment Transaction Table,

Follow Up: How would Alectio prevent multiple of the same transactions from occurring?

We can prevent multiples of the same transactions by using a relational structured database to check whether the same transaction has already been present in the database or not. The primary key of the payment table is the labeler, transaction id, the

amount, and the time stamp. So, if two transactions occur for the same details, there would be an error that is thrown, and we can immediately identify multiple transactions of the same type.

The payments service should keep checking and monitoring transactions, and should be able to prevent duplicate transactions from happening.