**SECURITY DOCUMENTATION FOR ON PREMISES DEPLOYMENT**

# Security pillars:

| Title | Description |
| --- | --- |
| Network Isolation | Will be handled by the client. |
| Docker Scanning | **Docker Notary** is used for the signing of Docker images, ensuring that all deployed images are both signed and verified. **AWS Inspector** is employed to perform vulnerability and malware scanning on Docker images stored in ECR before these images are pulled into client environments. |
| Access management and controls | User’s access management systems such as Active Directory, will be integrated through **Key Cloak** for managing access of the application interface. The access of the on-premises server environment will be handled by the client. This is to ensure only users who require it are allowed to access the data. All access to production data is logged for auditing purposes. We review which accounts can access our systems and the permissions they have regularly. Currently, auditing is carried out manually. However, we have the capability to introduce alerts or reporting functionalities in alignment with the specific requirements of the clients. |
| Application Layer threat prevention | The client will provide the on-premises data storage and will be responsible for managing the encryption of the stored data. This approach ensures that the client maintains full control over their data storage infrastructure and the security measures applied to protect their data. |
| File Storage | By default, the Application Server keeps the interim files, which can also be disabled if requested by the client. We apply PII data masking on the interim files for data protection. Model-Server and Graph-Server will not keep any persistent data (Interim files are configured to be deleted after the processing is completed). The NFS on client’s environment will be used for the input, output, and intermediate files with client’s encryption policy enabled. |
| Encryption | Our product offers the capability to configure SSL certificates for incoming traffic, enhancing the security of our application. Also, **TLS(1.2)** encryption is used to secure connections between MongoDB, and neo4j graph database servers and client applications. SSL is used to encrypt the data transmitted between RabbitMQ message brokers and client. applications. Mutual TLS (**mTLS**) is employed for securing internal service communication in the application. Our system employs a 256-bit encryption algorithm to ensure the secure encryption of data. |
| Backup and recovery | Backup and recovery procedures will be determined based on the specific requirements of the clients. We recommend the backup and recovery to be handled by the client. |
| High Availability and Fault-tolerance | To ensure high availability for MongoDB and Neo4j database, we suggest a three-node cluster. We recommend placing node of this cluster strategically in different high availability zones. The RabbitMQ will be deployed in three-node to ensure high availability and fault tolerance. |
| Logging and Automated compliance reporting | All commands run on the system, application component and application requests are all logged. We use technologies such as Elasticsearch/Kibana. Prometheus and Grafana for monitoring all the applications. |