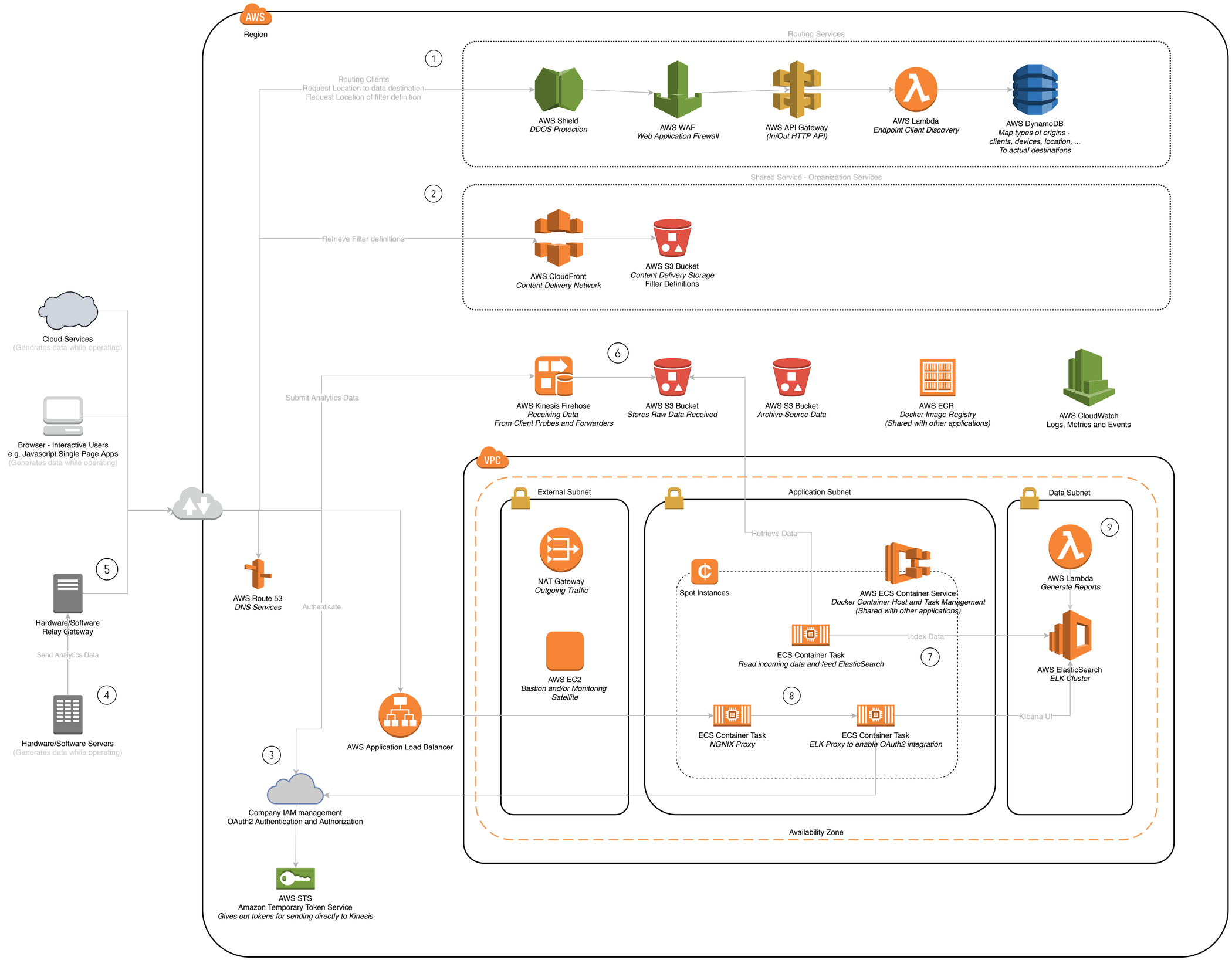
Flex4Apps Architecture for AWS

AWS Native Serverless Architecture

The convergence of cloud, communication and IoT infrastructure plus the trend towards virtual applications (e.g. migrating software to the cloud) create new challenges for application developers and infrastructure providers. The resulting systems are complex with dynamic resources hiding possible problems. This creates a requirement for flexible monitoring and optimisation methods. The Flex4Apps project addresses the challenges of monitoring and optimising large, distributed, cyber-physical systems. The goal of the project is to provide a solution to manage the high data volumes and complexity of system monitoring whilst disturbing the target system as little as possible.



1. Clients such as our servers installed at customer sites will call to the routing services endpoint to request the destination AWS Kinesis endpoint, allowing redirects to the appropriate global regions for data protection.
2. Clients will download a whitelist from the CDN explicitly listing which data should be collected and sent, allowing for on the fly changes if necessary. A cdn is well suited for many downloads of the same filter.
3. For integrated security an OAuth2 STS service will provide client code a valid temporary AWS token so they can start sending directly to Kinesis Firehose, this bypasses the need for serving endpoints which can potentially be hammered with data and keeps costs down.
4. Clients will buffer data and try and send the most optimal packets to kinesis to reduce bandwidth requirements and cost.
5. Aggregation Clients can also act as a gateway ensuring only a minimal number of servers need access to the internet and increase efficiency.
6. Firehose dumps the data into an S3 bucket as the volume and requests is hard to predict ensuring persistence.
7. An ECS docker container parses the incoming data and feeds this into ElasticSearch. This converts the hard to predict pull in a predictable push scenario. ElasticSearch can be easily overwhelmed and expensive to operate at scale, so to ensure you can run a minimal cluster, the docker container feeding it limits the items and insertion rate to keep running costs down.
8. Securing Kibana can be a challenge. One way to approach it would be through a specialised proxy integrating with the OAuth2 services. Kibana is exposed as a getting started UI.
9. To expose the data as analytical reports, a Lambda function could on a schedule produce reports and submit them to another system. They could be published to a website, an SQL database, an S3 bucket, a Data Warehouse, a big data analytics system or any BI tool. This further development is highly dependent on the use case and tools and is left as an exercise for the reader.

Flex4Apps Architecture Blueprint