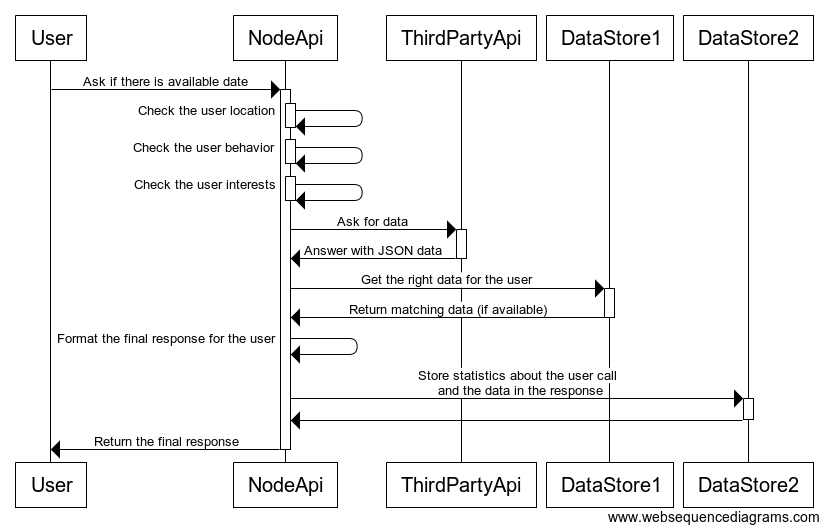
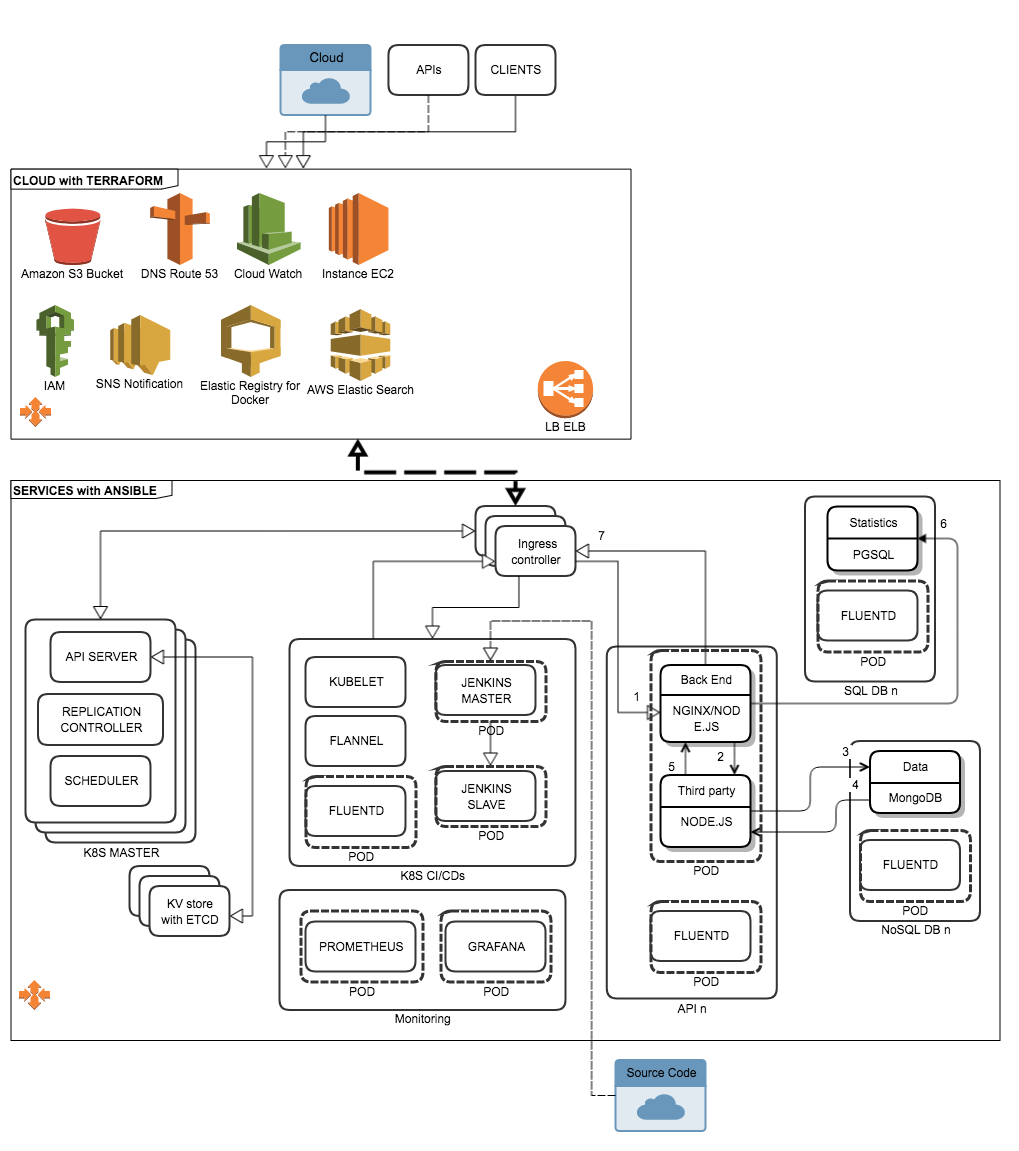
# Stack Communication Flow and Diagram

## Source Diagram



## Diagram of the platform



## Technologies used by functions

|  |  |  |
| --- | --- | --- |
| OS | Linux | AMI for Ubuntu 16.04 |
| COMPUTE | EC2 instance | AWS choice |
| DATABASE | Amazon RDS with PGMongoDB | MongoDB for data with Json supportAmazon RDS can be PSQL for SQL storage |
| BACKUP | S3 Buckets | AWS S3 is an object storage provided by AWS. Objects themselves can be anything and its used for all storage |
| APPS ORCHESTRATION | Kubernetes | Kubernetes is an open source system for managing [containerized applications](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/) across multiple hosts; providing basic mechanisms for deployment, maintenance, and scaling of applications. **Controllers**: Kubernetes HA master will run Kubernetes API Server, Controller Manager and Scheduler services  **Workers**: Kubernetes Nodes or Minions will run Docker, Kubernetes Proxy and Kubelet services, have also CNI installed for networking between containers |
| KV STORAGE | ETCD | Key/Value to maintain the K8S state and store cluster data |
| CONTINUOUS INTEGRATION | Jenkins | Create an artifact of your application (deployable one or more artifacts, manifesto (README, INSTALL, VERSION, CHANGELOG), logs, debug artifacts, runtime/build time dependencies and their versions**)** |
| INFRA DEPLOYMENT | Terraform | Terraform (json format) enables you to safely and predictably create, change, and improve production infrastructure. |
| SERVICE DEPLOYMENT | Ansible | Ansible is a tool that streamlines installing and managing services. |
| LOGGING | FluentD/AWS Elasticsearch | FluentD like CloudTrail will capture all compatible events and pushed to Elasticsearch service for log management |
| MONITORING | CloudWatch | CloudWatch Metric Alarm  Can be backuped with Prometheus with K8S |
| ALERTING | SNS | Send SMS from CloudWatch |
| DASHBOARDING | Grafana | Graph data and write your script to fetch and process data |
| REGISTRY | Amazon Elastic Container Registry (ECR) | Registry allow to store and share Docker images |
| LOAD BALANCING | Elastic LB | Amazon services for load balancing |
| DNS | Route 53 | DNS routing and name |
| NETWORK | Ingress Controller | This resource could be used to give services externally reachable URLs and load balance traffic into our cluster |
| CDN | CloudFront | Content delivery network distribute service spatially relative to end-users to provide high availability and high performance. |

## Platform explanation

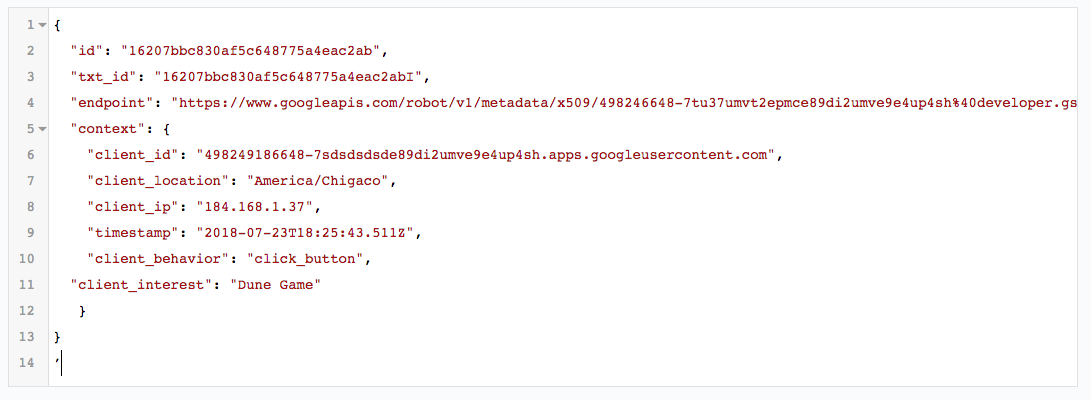
Our target includes:

* VPC from Amazon for the VPN network
* EC2 instances for Kubernetes Control Plane: Controller Manager, Scheduler, API Server.
* EC2 instances for the HA etcd cluster.
* EC2 instances as Kubernetes workers (aka Minions)
* Container networking using Kubenet plugin relying on CNI
* EC2 instances for Grafana dashboarding
* HTTPS communication between all components
* All Kubernetes and etcd components run as services directly in the EC2 (not on containers)
* EC2 instance and/or local computer for terraform (manage cloud creation like EC2 instances, route53, certificates or network layer, LB, Firewall, apply security groups, etc)
* EC2 instance and/or local computer for ansible (manage services like K8S, docker, security packages, SSH keys and ETCD)
* A github private account to store all the codeline
* AWS RDS with PostgreSQL
* NoSQL instance with MongoDB
* Route53 to DNS routing
* CloudWatch for monitoring
* CloudTrail for the logging applications and services
* Cognito/Vault to secrets
* SNS for alerting by SMS
* CloudFront for the CDN to provide the best quick way
* S3 bucket for the storage and cold backup
* IAM for users and groups access
* Registry to host docker image

The following diagram describes how interaction between the communication flow between the whole stacks:

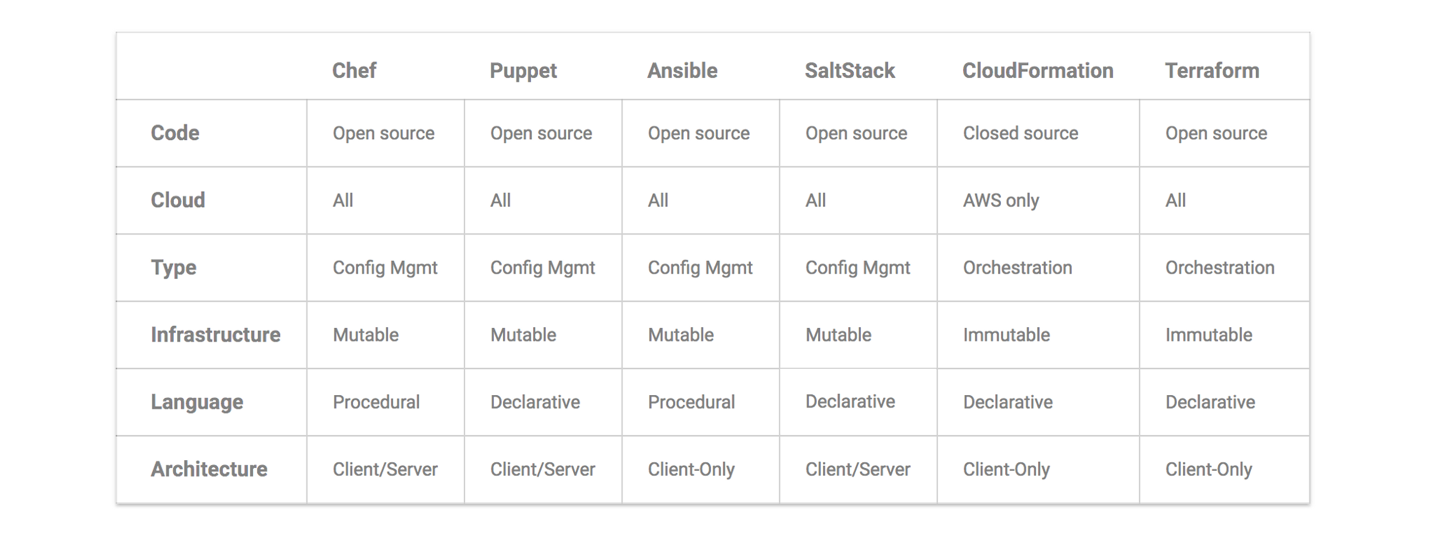
1. A client (human or not human) makes a request to the API server, on the uri path handled by the api  
   example: https://<apiserver\_url>/apis/hello
2. Client request CDN first then client access to the ELB who makes routing decisions at the application layer and route requests to ports on each container instance in the cluster
3. The application deployment is managed by Kubernetes. Client access to NodeAPI nginx+node.js backend **(1)** who ask if a data is available
4. NodeAPI picked up location, behavior and interests and request ThirdPartyAPI to provide a response **(2)**
5. ThirdPartyAPI answer with a JSON data and give the response **(3)** if user matching or not **(4)** provided by DynamoDB database
6. The final response is formatted by the NodeAPI **(5)** and pushed to store statistics on the SQL database RDS **(6)**
7. The NodeAPI server proxies the response to the initial client **(7)**.

Example of JSON job:



## Considered Alternatives

Puppet vs. Chef vs. Ansible vs. SaltStack vs Terraform



I had considered the platform with full AWS services like:

* Elastic for the load balancing
* API Gateway to manage the API
* Lambda to deploy the code
* SNS for notification and alerting
* DynamoDB for the JSON storage in NoSQL
* RDS for the SQL storage with MariaDB
* S3 bucket for storage
* CloudWatch for the monitoring and SNS for alerting

On the proposed platform, I had also considered the Service Discovery with Consul (unlikely in this case due to fixed services/application) and Vault to enable secrets on K8S (Kubernetes authentication and PostgreSQL database backends by example).

In the same way, I could be considered a messaging system like RabbitMQ or Kafka (queue usage).