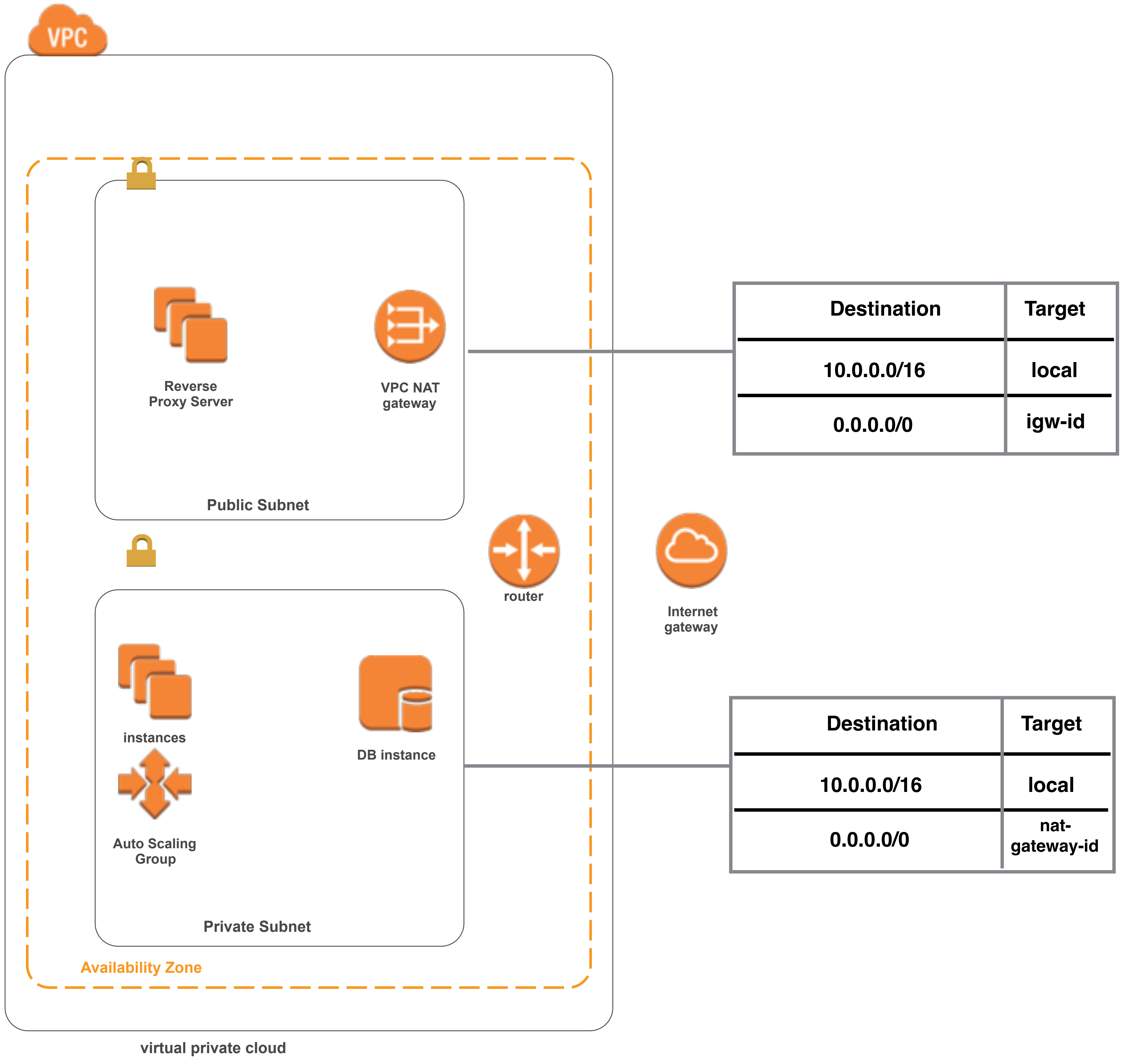
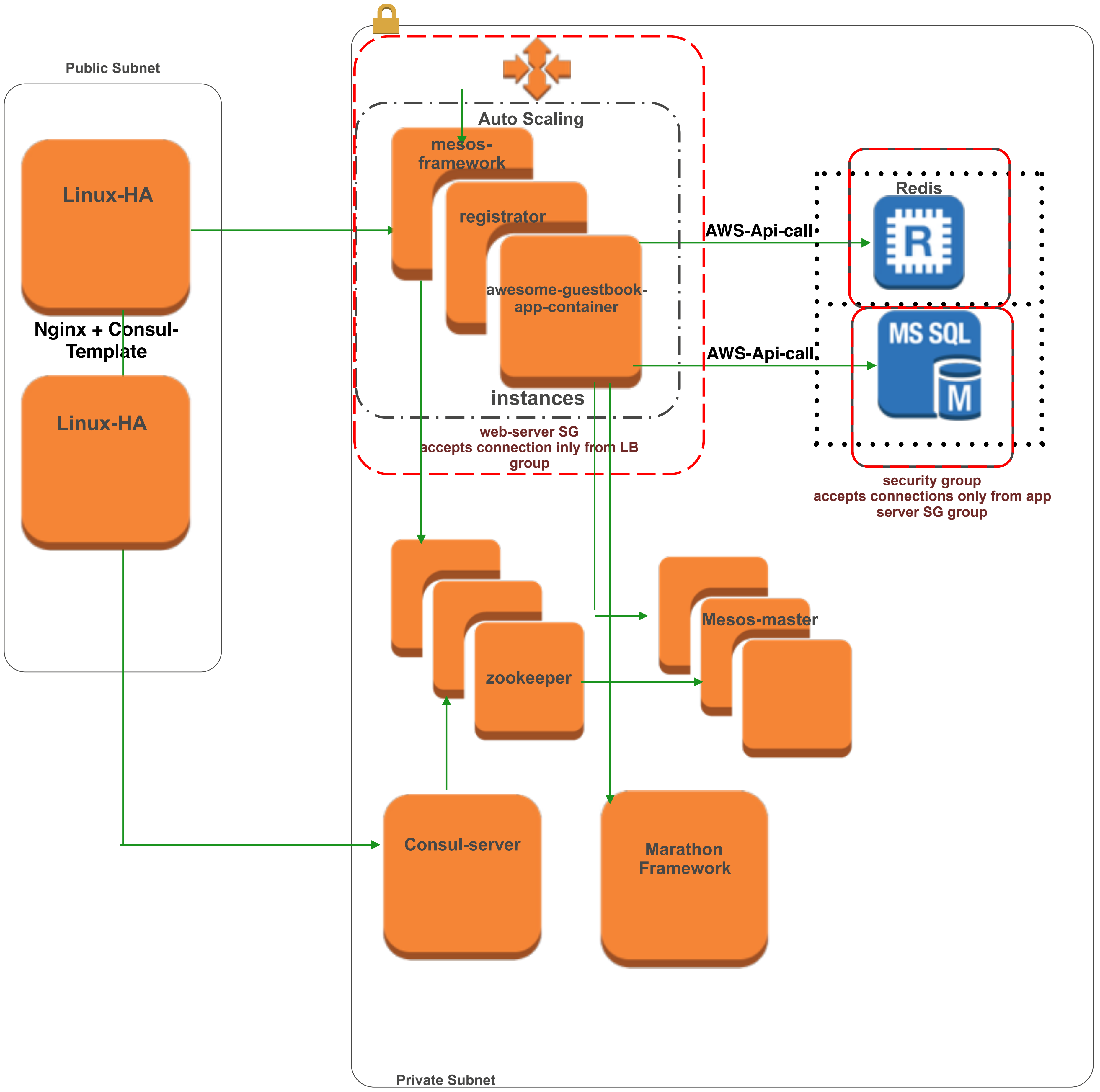


Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	nat-gateway-id


Public Subnet
Linux-HA
Nginx + Consul-Template
Linux-HA
Auto Scaling
mesos-framework
registrator
awesome-guestbook-app-container
instances
web-server SG
accepts connection inly from LB group
AWS-Api-call
AWS-Api-call
Redis
MS SQL
security group
accepts connections only from app server SG group
zookeeper
Mesos-master
Consul-server
Marathon Framework
Private Subnet

1. **Assumption**:
    1. Awesome-Guestbook application is Nginx.
    2. Workstation system should have Virtualbox installed. Make sure the compatibility with Vagrant version.
    3. All apps are running as docker container
    4. Chef as a configuration managent tool.

Diagram: See above slides for complete architecture diagram on AWS cloud.

Explanation: Since we are deploying application as a docker container, we need to introduce new set of tools in the infrastructure to accommodate container technologies. Below is the list of tools that is needed to deploy any container on AWS cloud.
a) Mesos
b) Marathon
c) Consul
d) Zookeeper
e) Consul-Template
f) Registrator

Other than this we will use usual set of tools for running a web application.
a) Nginx
b) Redis
c) MysqlDB

Repositories.
a) Github
b) DockerHub or on premise docker registory

Docker-machine and VirtualBox is good to have for local development environment

Working of Mesos:
Mesos works in a master slave architecture. On mesos master node, master daemon runs which talk to the agent daemons running on mesos-slave nodes. In addition mesos runs a framework that runs tasks on these agents. Mesos framework consist two components, a scheduler and executor.
A scheduler tells mesos master what resources in terms of CPU, Mem and Disk are available on agent nodes & executor is responsible for launching tasks on agent nodes.
Mesos keeps all its Master configuration in zookeeper for fault tolerance

Why Marathon:
By **Marathon** Framework all the tasks which is launching the containers or apps is done via calling Marathon api. Marathon reads all config options from a Json file and tells the executor to run the tasks on available node resources

Setting up mesos:
There are EC2 scripts, deploy scripts & chef cookbooks are available for deploying mesos cluster. In our case we will use mesos cookbook available in chef supermarket for deploying mesos and marathon.

Mesos cluster:
We need set of EC2 instances which will work as a cluster for mesos. Best way is to create a ASG of on-demand instances for mesos master & agent. In the startup script of ASG we need to define the installation of chef client with appropriate client.rb file, Authentication file and any other necessary configs like local DNS name server etc.
It is best a practice to have mesos master cluster on a 3 separate EC2 instances and mesos agent cluster on a 5 EC2 instances which belongs to ASG in addition 3 instances are necessary for zookeeper quorum.
At a particular time only one instance serve as a mesos master agent and rest work as a standby.

When does Consul comes in picture?:
In a infrastructure where multiple containers serves as a application or some micro-services, the most common problem arises is service discovery. Consul solves the problem of service discovery in a large infrastructure. In addition consul also provides other features like Health checking, Key/Value store.
In short all consul agents talk to consul server to discover the provider of a service.

Consul-Template:
Consul-Template is an application which queries the Consul server and updates any template on the file system. In our case consul-template runs on front-end LB's and updates upstream files for each server block of nginx.

Configuration Management:
Chef is one of the most popular config management tool. It works on a pull based mechanism and with Ruby DSL support, writing a system state is quite easy. Also with large support of open source cookbooks for doing almost anything on the infrastructure. It my first choice for managing infra as a code. In the above architecture from mesos slaves to Frontend Nginx server, everything is configured by custom wrapper cookbooks. Chef roles is the best way to define a server state with multiple cookbooks. For e.g. Front-end role is responsible for setting up the Nginx, Heartbeat and consul-template on a particular server. For cost saving you may use a single server for Chef-server, Marathon and Sensu.

Why Chef?:
Some might argue that a push based mechanism like Ansible is more fast and reliable way for config management. In my opinion with features like cookbook versioning, syntax checking before uploading code to chef server and daemon mode makes chef a more reliable and less error prone software. Again you can find cookbooks for almost anything.

Reverse Proxy:
Nginx as a reverse proxy server is my first choice for tcp/ip load balancer. Easy to install, unlimited plugins support and its eally fast. With features like limiting number of connections and request from client gives the ability to fight against Dos types of attacks. All these things are fairly simple and straight forward.

High Availability:
Linux-HA is a daemon that provides resource sharing and membership in a cluster infrastructure.
With AWS's elastic-ip feature, setting up HA for load balancer on AWS is really simple.

Monitoring:
Sensu monitoring system is simple and fast. With Sensu in your infrastructure, adding up and deleting a node becomes quite easy as compares to Nagios, where one has to manually do the job of registering with nrpe daemon. Again the support of plugins in Sensu enables the administrator to monitor everything on the infrastructure. Simple json format and api is one the coolest features of Sensu.

Continues Integration and Deployment:
GO-CD is a CI/CD tool by thought works. Its web based configuration make it fairly simple to implement any deployment technique. Similarly capistrano is a command line based deployment tool comes in from of a gem(cap).

On the first look the above architecture explanation may looks bulky and hell of job. But in reality, most of the things are one time job, like setting up chef-server and Sensu, or ASG configurations with bootstrapping script or Go-Cd setup with capistrano. Once things are in place deployment becomes so easy that even a new joined front-end can learn deployment in a day and can perform the day 1 deployment.