Kubernetes on AWS

Challenges

Viet Nguyen Chan
Senior DevOps Engineer
Fossil Group

What is Kubernetes?

Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure.

With Kubernetes, you are able to:

- Deploy your services quickly and predictably.
- Scale your applications on the fly.
- Seamlessly roll out new features.
- Optimize use of your hardware by using only the resources you need.

Kubernetes: Challenges?

It’s not that easy to run Kubernetes.

The top three concerns Kubernetes users have are :

- Managing multi-cloud or hybrid environments with Kubernetes
- Running stateful or data-intensive workloads. While Kubernetes is gaining capabilities, users are understandably concerned about the complexity of being able to run such workloads easily
- The complexity of operating Kubernetes in production in the enterprise

Kubernetes on AWS: Why?

Benefits:

- Persistent Volume : EBS, EFS
- Auto Scaling
- HA (regions, AZ)

Drawbacks:

- Managing a Kubernetes cluster is hard
- Managing AWS is hard

Deployment model

Legacy : write your own scrips

- Pros :
  - Flexible
- Cons :
  - Heavy maintenance

Deployment model

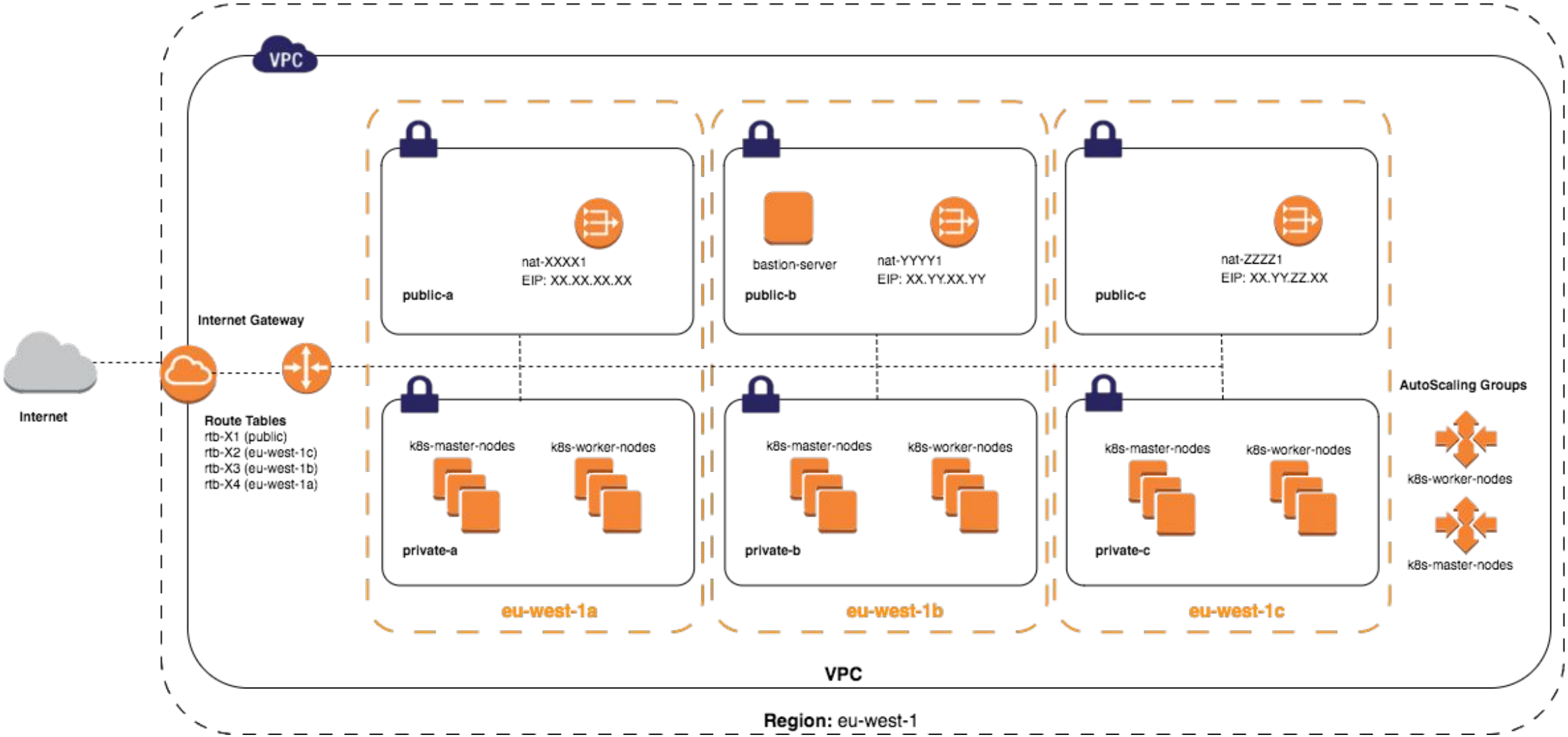
Current : Kops (Kubernetes Operations)

- Pros :
  - Production-grade
  - Deep integration with AWS
  - support Terraform (IaaC)
- Cons :
  - waiting for release cycle

Model :

- 3 masters (including etcd)
- 4x workers

Deployment model



Authentication

- OIDC
- You can try Heptio Authenticator for AWS!

What is Helm?

Helm is the Kubernetes package manager designed to manage the entire lifecycle of an application running in Kubernetes.

- Supports Application Bundles
- Improves CI/CD Workflow
- Advanced Templating Engine
- Eliminates numerical values in manifest
- Support Upgrade/Rollback of services

Security problems with Helm

Breaking Down The Problem :

- A **privileged API user**, such as a cluster-admin. We actually want these users to have access to the full power and convenience of helm charts.
- A **low-privilege API user**, such as a user who has been restricted to a single namespace using RBAC. We would like to allow these users to install charts into their namespace, but not affect other namespaces.
- An **in-cluster process**, such as a compromised webserver. There is no reason these processes should install helm charts, and we want to prevent them from doing so.
- A **hostile chart author** can create a chart containing unexpected resources. These can either escalate one of the other groups above, or run other malicious jobs.

https://engineering.bitnami.com/articles/helm-security.html

CI/CD

Jenkins

- Kubernetes plugin
- in-house tool for helm chart render and jenkins job automation

Why run Jenkins in Kubernetes

- You can scale the number of nodes in the cluster based on the load jenkins receives
- You can build your own custom version of Jenkins as a Helm chart and easily deploy it to the cluster
- You can pull your jenkins images from a public or private repository using Helm. Further instructions for that can be found in our repository

Cluster AutoScaler

The cluster autoscaler has a main loop that :

- Look for unschedulable pods (pods for which the scheduler has not been able to find a node to deploy them to)
- Calculate which of the node groups can ben expanded to accomodate these pods and expands one of them
- Check unneeded nodes, and remove them

Cluster AutoScaler

Problem ?

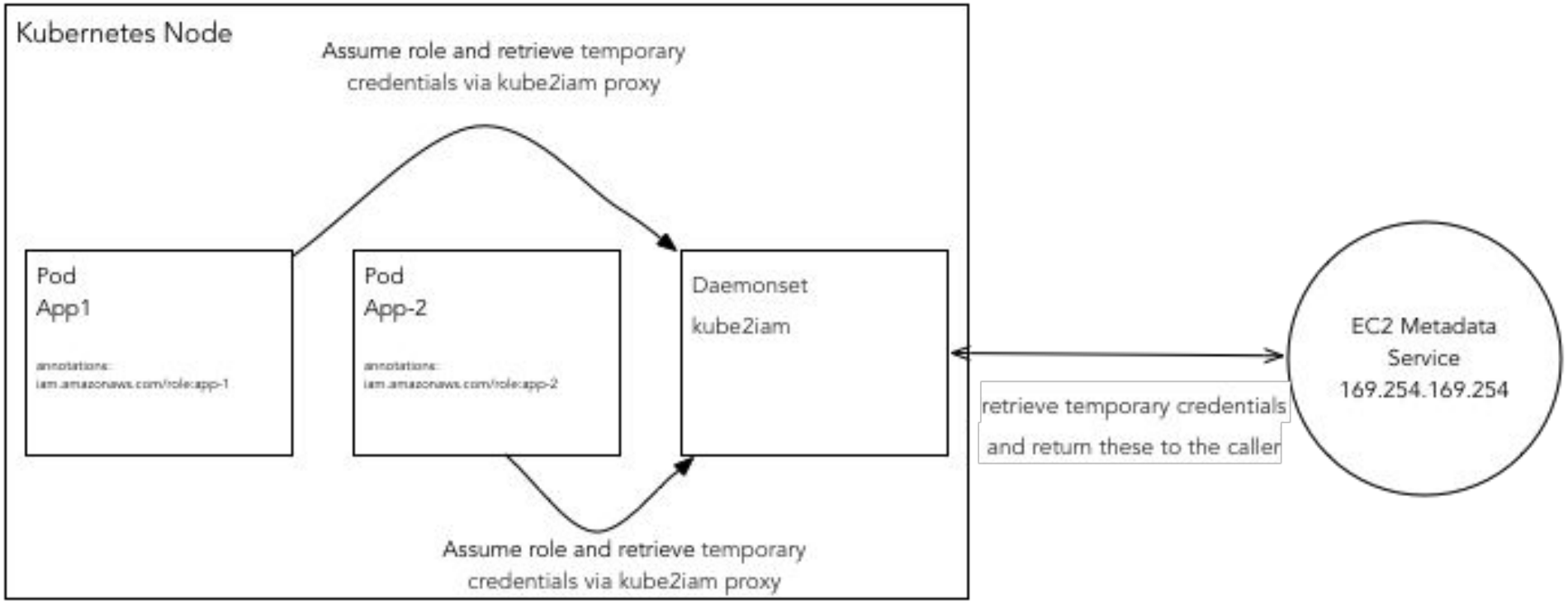
- Work with custome scheduler

Ingress Controller

- Support ELB and ALB
- K8s v1.9 support NLB

IAM Permissions

- Kubernetese is not native the AWS IAM roles and permissions.
- Solution : Kube2IAM



KubeDNS

- Many problems
- Tuning

ExternalDNS

- Route53 DNS integration via External DNS

Problem with scheduling

Java problem with container (Docker) :

- As of Java SE 8u131, and in JDK 9, the JVM is Docker-aware with respect to Docker CPU limits transparently. That means if -XX:ParalllelGCThreads, or -XX:CICompilerCount are not specified as command line options, the JVM will apply the Docker CPU limit as the number of CPUs the JVM sees on the system

- For Docker memory limits, there is a little more work for the transparent setting of a maximum Java heap. To tell the JVM to be aware of Docker memory limits in the absence of setting a maximum Java heap via -Xmx, there are two JVM command line options required, -XX:+UnlockExperimentalVMOptions -XX:+UseCGroupMemoryLimitForHeap.

Problem with scheduling

Solution :

- Short-term : write own custom scheduler
- Long-term : There’s an experimental support in the JVM that has been included in JDK10 to support cgroup memory limits in container (i.e. Docker) environments.

Observability

- Metrics
- Logging
- Tracing

Metrics

- Prometheus
- Grafana

Metrics

- Good :
  - Pull model
  - Powerful query language
  - Service discovery

- Problem :
  - scaling becomes an issue in large deployments.

Metrics

- Solution ?
  - Use influxdb as backend
  - Prometheus advises a push-based approach for collecting metrics for short-lived jobs.
  - Multi-prometheus cluster using prometheus-operator

Logging

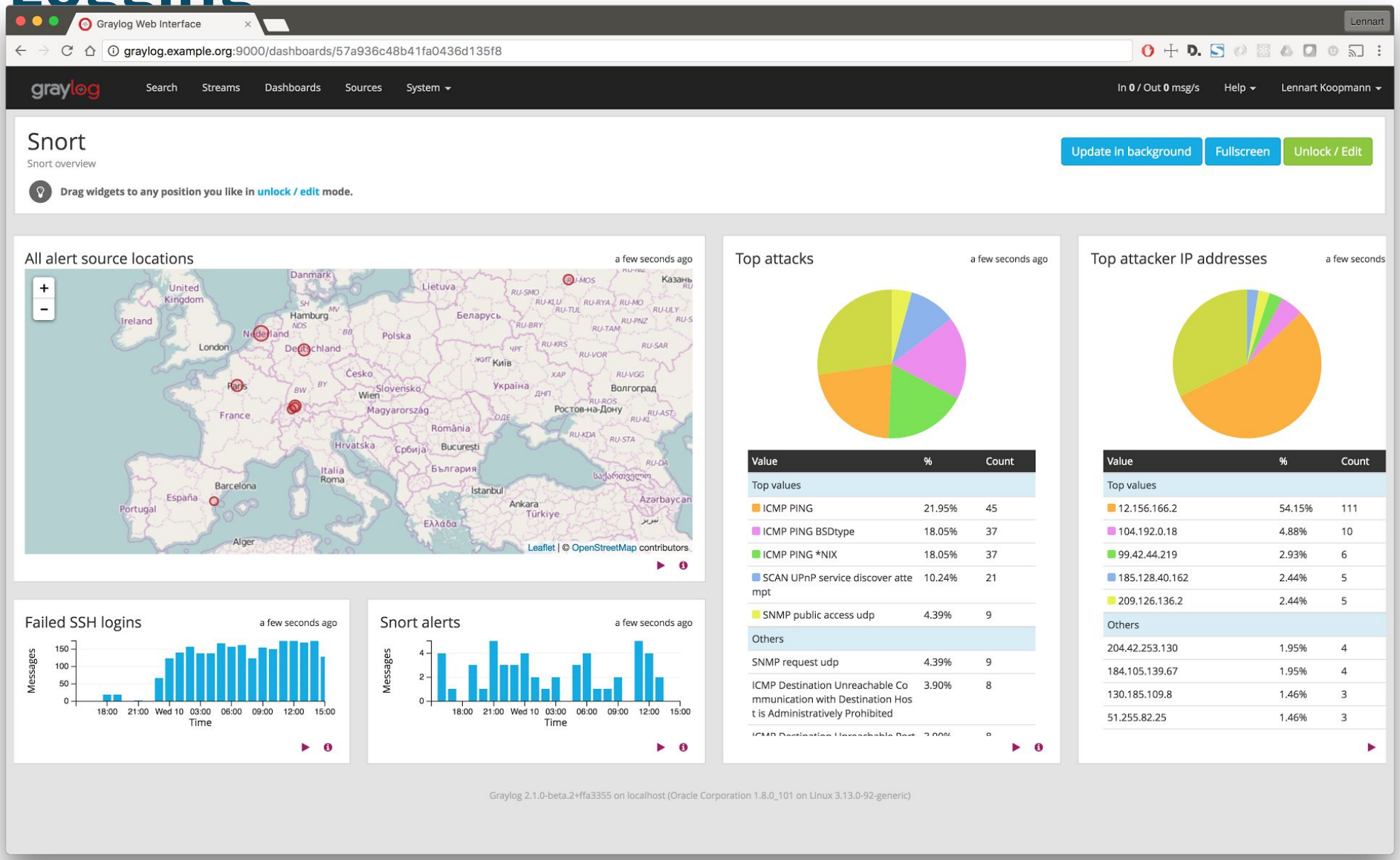
- Migrate from ELK to Graylog

Logging

Why Graylog?

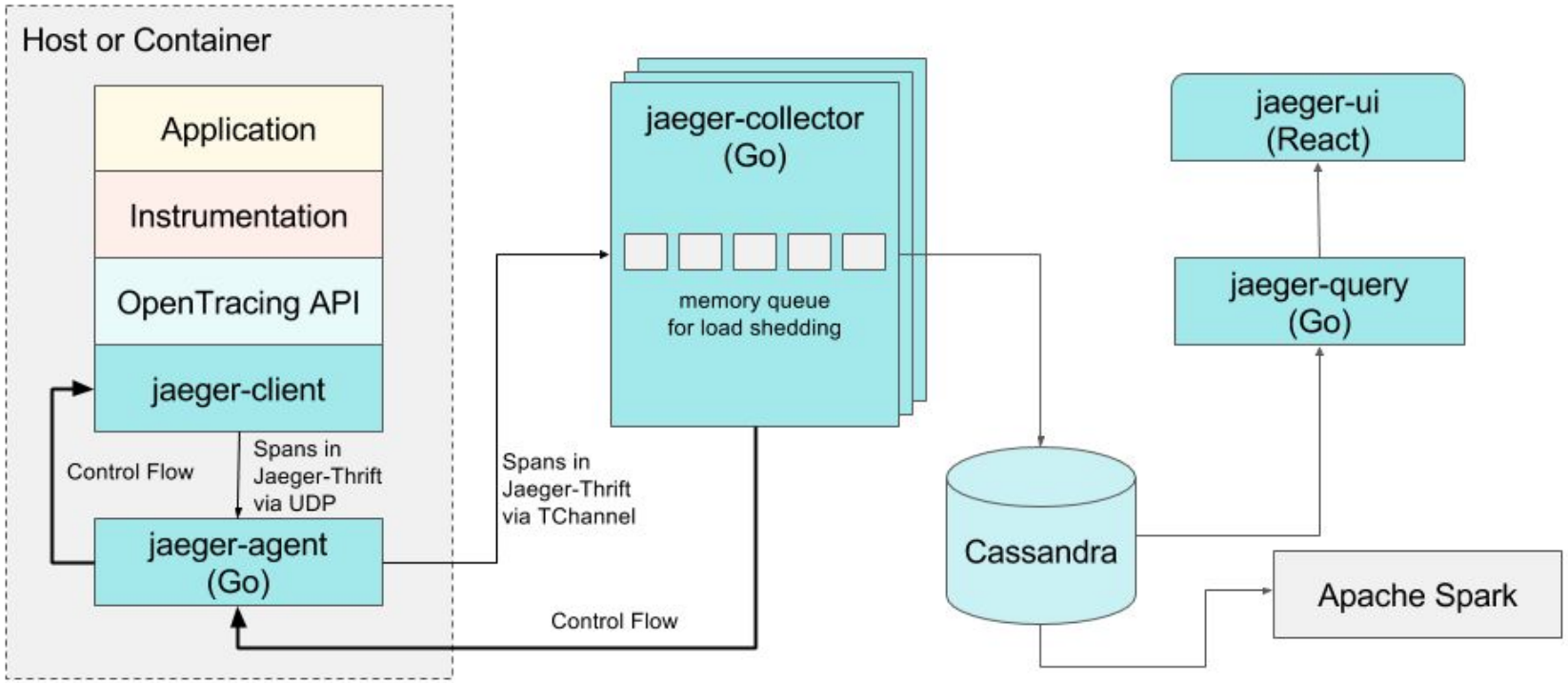
- Pros :
  - based on ElasticSearch
  - HA & Scaling
  - Authentication/Authorization
  - GELF
  - support streams, dashboards, triggers, alerts
- Cons :
  - ElasticSearch .... and MongoDB
  - maintenance ?

Logging


Graylog Web Interface
Lennart
graylog.example.org:9000/dashboards/57a936c48b41fa0436d135f8
graylog
Search
Streams
Dashboards
Sources
System
In 0 / Out 0 msg/s
Help
Lennart Koopmann
Snort
Snort overview
Update in background
Fullscreen
Unlock / Edit
Drag widgets to any position you like in unlock / edit mode.
All alert source locations
a few seconds ago
Leaflet | © OpenStreetMap contributors
Top attacks
a few seconds ago
Value % Count
Top values
ICMP PING 21.95% 45
ICMP PING BSDtype 18.05% 37
ICMP PING *NIX 18.05% 37
SCAN UPnP service discover attempt 10.24% 21
SNMP public access udp 4.39% 9
Others
SNMP request udp 4.39% 9
ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited 3.90% 8
Top attacker IP addresses
a few seconds
Value % Count
Top values
12.156.166.2 54.15% 111
104.192.0.18 4.88% 10
99.42.44.219 2.93% 6
185.128.40.162 2.44% 5
209.126.136.2 2.44% 5
Others
204.42.253.130 1.95% 4
184.105.139.67 1.95% 4
130.185.109.8 1.46% 3
51.255.82.25 1.46% 3
Failed SSH logins
a few seconds ago
Messages
Time
Snort alerts
a few seconds ago
Messages
Time
Graylog 2.1.0-beta.2+ffa3355 on localhost (Oracle Corporation 1.8.0_101 on Linux 3.13.0-92-generic)

Tracing

- Jaeger



Database

- Storage in K8s v1.9 :
  - StatefulSets is stable
  - Consumption of statically provisioned raw block persistent volumes for Fibre Channel
  - CSI alpha
- EBS issues :
  - https://dzone.com/articles/fixing-kubernetes-failedattachvolume-and-failed-mo
  - https://portworx.com/ebs-stuck-attaching-state-docker-containers/
  - https://jobs.zalando.com/tech/blog/reattaching-kafka-ebs-in-aws/
- Altenative EBS :
  - gluster-kubernetes
  - OpenEBS
  - Rook
- Currently, still rely on AWS RDS

SSL

- Write own CRD to maintain SSL for ingress-controller
- You can try Kube-lego!

EKS & Fargate

- AWS-managed Kubernetes
- clusterless/serverless containers

Other tips & tricks

- Ensure that your nodes have capacity to handle at least one node failure
- Delete unused resources
- Tag all resources
- Seperate IGs
- Run Pods with corret node-selectors
- Notice AWS Limits

Next steps

- Database on K8s
- DR
- NetworkPolicy
- ServiceMesh

Resources

- https://github.com/aws-samples/aws-workshop-for-kubernetes

Thank you