



kubernetes

Building Efficient Microservices Solutions With Kubernetes

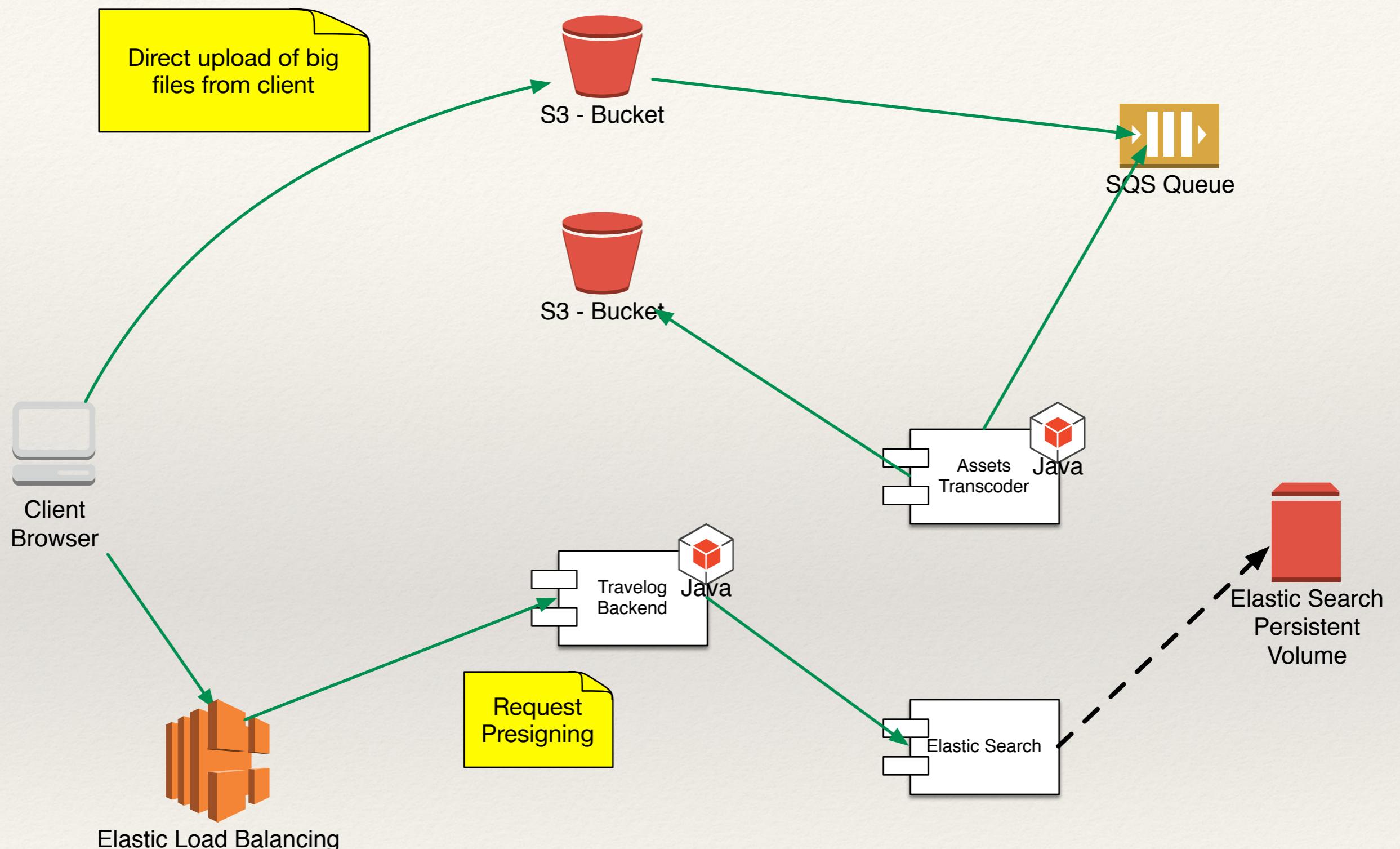
Konstantin Ignatyev
Solutions Architect



When we develop complex and important system we need to know how to:

- Ensure availability of a units at runtime;
- Configure units per env;
- Apply configuration changes;
- Have outage-less deployments;
- Troubleshoot;
- Protect secrets;
- Prevent unauthorized access;
- Manage storage;
- Scale;
- Use one unit from another (micro-services);
- Configure SAS per environment;
- Develop and debug locally;
- Manage CPU and Memory resources;
- Have preferred targets (high CPU, or GPU, etc.)
- Run batch jobs;
- Expose service(s) for external clients;
- Collect logs;
- Know where things are;

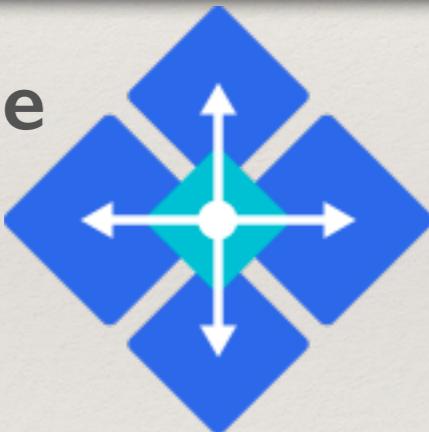
Sample Application: Travelog



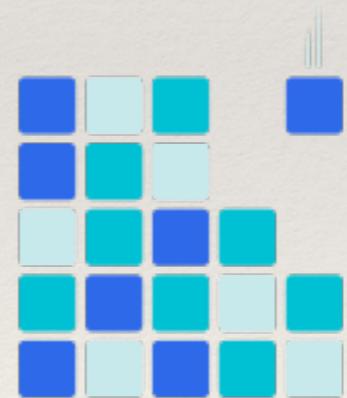
Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon **15 years of experience of running production workloads at Google**, combined with best-of-breed ideas and practices from the community.

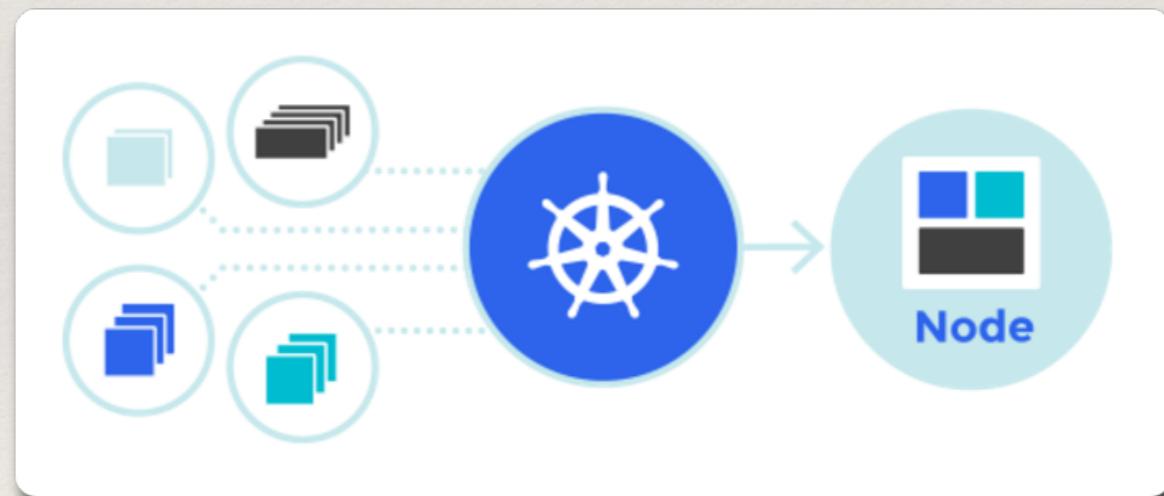
Planet Scale

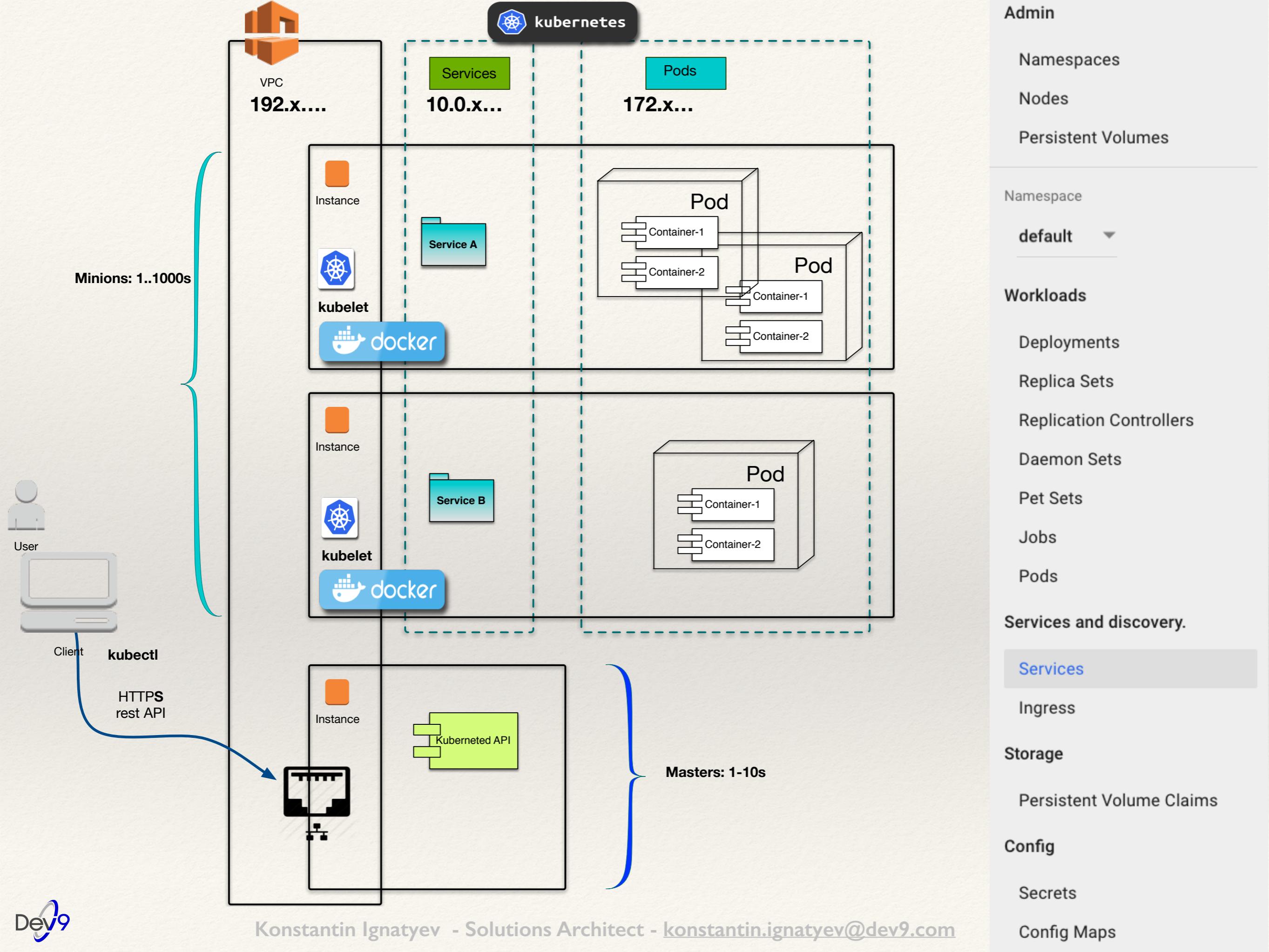


Never Outgrow



Run Anywhere





How to keep application working when something might fail:



ReplicationController

Ensures that we have requested number of replicas running at all times, and takes care of rolling updates when necessary.

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: travelog-ws-rc
spec:
  replicas: 1
  template:
    metadata:
      name: travelog-ws-pod
      labels:
        app: travelog-ws-pod
    spec:
      containers:
        - name: travelog-ws-container
          image: "kgignatyev/travelog-ws:2017-01-25-22-21-23"
          ports:
            - containerPort: 9100
          volumeMounts:
            - name: travel-properties
              mountPath: "/etc/travelog"
              readOnly: true
          volumes:
            - name: travel-properties
              secret:
                secretName: "travel-properties"
```

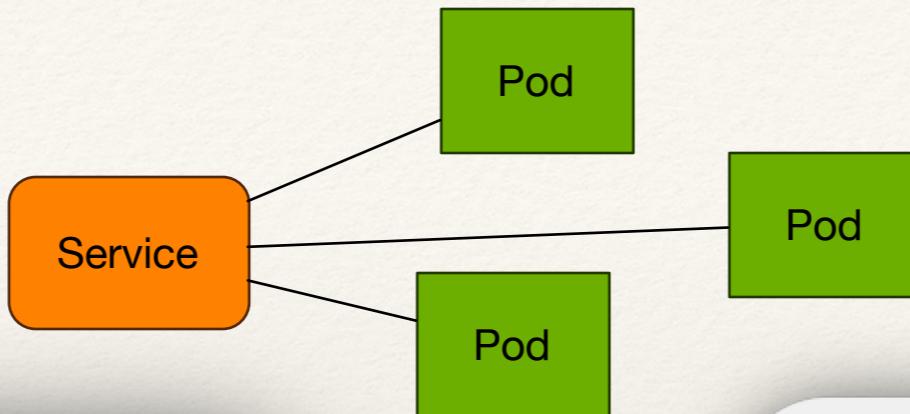
How to make our functionality available for client when pod-s come and go:



Service

Service ≈ Internal Load Balancer

Ensures that clients have a permanent endpoint to call



```
apiVersion: v1
kind: Service
metadata:
  name: travelog-ws-service
spec:
  selector:
    app: travelog-ws-pod
  ports:
    - port: 9100
      targetPort: 9100
```

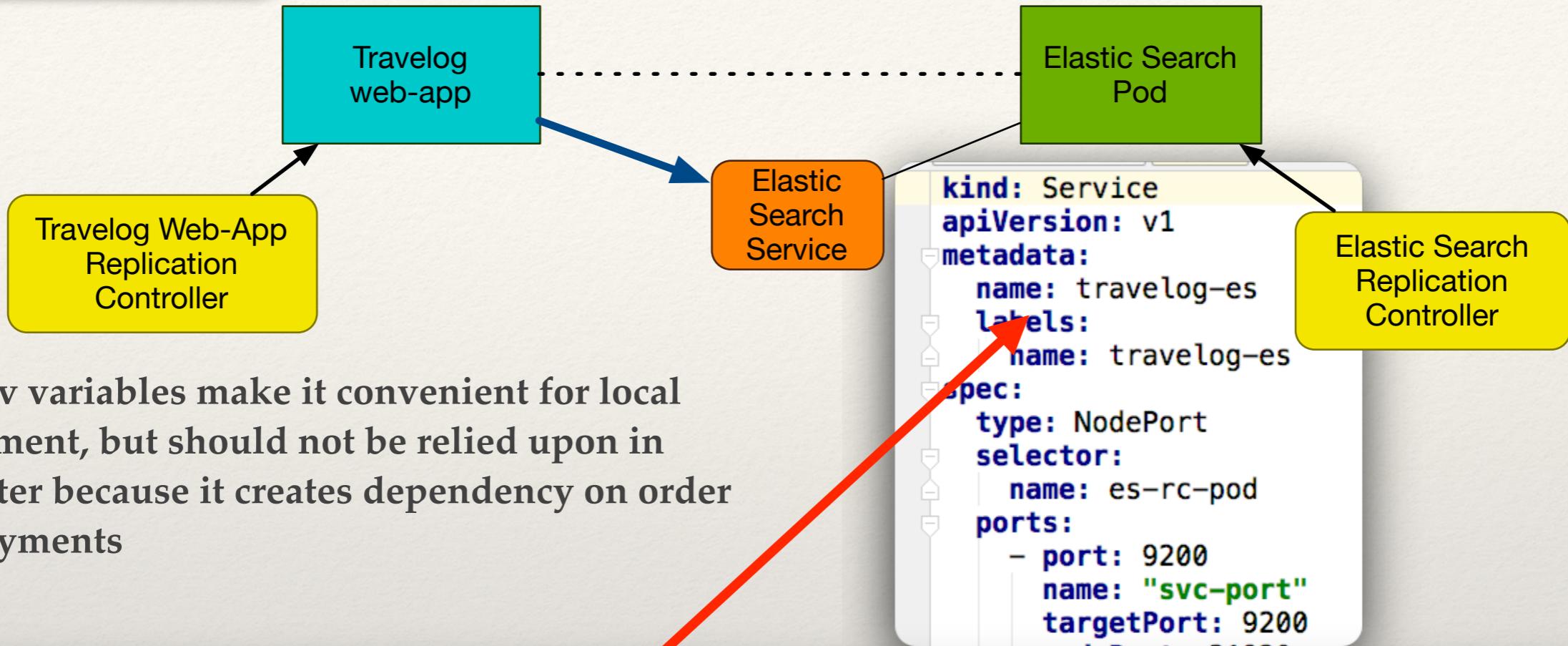
```
apiVersion: v1
kind: ReplicationController
metadata:
  name: travelog-ws-rc
spec:
  replicas: 1
  template:
    metadata:
      name: travelog-ws-pod
      labels:
        app: travelog-ws-pod
    spec:
      containers:
        - name: travelog-ws-container
          image: "kgignatyev/travelog-ws"
          ports:
            - containerPort: 9100
```

A Kubernetes `Service` is an abstraction which defines a logical set of `Pods` and a policy by which to access them - sometimes called a micro-service. The set of `Pods` targeted by a `Service` is (usually) determined by a [Label Selector](#) (see below for why you might want a `Service` without a selector).

How to make our functionality available for client when pod-s come and go:



Service



Note: env variables make it convenient for local development, but should not be relied upon in live cluster because it creates dependency on order of deployments

```
public class TravelogService {  
    public String esBase() {  
        String hostFromEnv = System.getenv( name: "TRAVELOG_ES_SERVICE_HOST" );  
        String res = "http://travelog-es:9200";  
        if (hostFromEnv != null) {  
            res = "http://" + hostFromEnv + ":" + System.getenv( name: "TRAVELOG_ES_PORT_9200_TCP_PORT" );  
        }  
        return res;  
    }  
}
```

How to configure and apply configuration changes (slide 1)



ConfigMap, Secret

```
Document 1/1 apiVersion:  
apiVersion: v1  
kind: Secret  
metadata:  
  name: travel-properties  
type: Opaque  
data:  
  travel.properties: I0dlbmVyYXR
```

```
13 Properties context = readProperties( '../secrets/travel', env)  
14  
15 StringWriter sw = new StringWriter()  
16  
17 context.store(sw, "Generated for env:" + env)  
18  
19 String confFileContent64 = Base64.getEncoder().encodeToString(sw)  
20  
21 def out = "target/k8s/travel-secret." + env + ".yml"  
22 processTemplate("src/secret tmpl.yml",  
23                 [secretName: 'travel-properties',  
24                  secretKey: 'travel.properties',  
25                  secretContent: confFileContent64], out)
```

```
kubectl create -f infrastructure/target/k8s/travelog-secret.dev.yml
```

```
Konstantins-Mac-Pro:~ kgignatyev$ kubectl get secrets  
NAME                                     TYPE           DATA   AGE  
default-token-m26lu                      kubernetes.io/service-account-token  3       88d  
fluentd-conf                             Opaque          1       7d  
kgignatyev-registrykey                   kubernetes.io/dockercfg            1       56d  
mcytravel-properties                     Opaque          1       88d  
mcytravel-ssl                           Opaque          2       54d  
travel-properties                        Opaque          1       10d  
Konstantins-Mac-Pro:~ kgignatyev$
```

How to configure and apply configuration changes (slide 2)



kubernetes

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: assets-transcoder-rc
spec:
  replicas: 1
  template:
    metadata:
      name: assets-transcoder-pod
      labels:
        app: assets-transcoder-pod
    spec:
      containers:
        - name: assets-transcoder-container
          image: "kgignatyev/assets-transcoder"
          ports:
            - containerPort: 7200
      volumeMounts:
        - name: travel-properties
          mountPath: "/etc/travel"
          readOnly: true
      volumes:
        - name: travel-properties
          secret:
            secretName: "travel-properties"
```

```
Document 1/1 apiversion:
apiVersion: v1
kind: Secret
metadata:
  name: travel-properties
type: Opaque
data:
  travel.properties: I0dLbmVyYXRlc3Q=
```

ConfigMap, Secret

```
version: v1
kind: ReplicationController
metadata:
  name: travelog-ws-rc
spec:
  replicas: 1
  template:
    metadata:
      name: travelog-ws-pod
      labels:
        app: travelog-ws-pod
    spec:
      containers:
        - name: travelog-ws-container
          image: "kgignatyev/travelog-ws:2017-05-10"
          ports:
            - containerPort: 7100
      volumeMounts:
        - name: travel-properties
          mountPath: "/etc/travelog"
          readOnly: true
      volumes:
        - name: travel-properties
          secret:
            secretName: "travel-properties"
```

```
TravelogConfig
package kg.i.presentations.k8s.travelog;
import ...
@Configuration
@PropertySource("file:/etc/travelog/travel.properties")
public class TravelogConfig {
  @Bean
  TravelConfigProperties configProperties() { return new TravelConfigProperties(); }
}
```

How to know where things are and what is going on:



Dashboard

kubernetes Workloads + CREATE

Admin
Namespaces
Nodes
Persistent Volumes

Namespace
default

Workloads

Deployments
Replica Sets
Replication Controllers
Daemon Sets
Stateful Sets
Jobs
Pods

Services and discovery
Services
Ingresses

Storage
Persistent Volume Claims

Config
Secrets
Config Maps

CPU usage
CPU (cores)
Time
0.004
0.003
0.002
0.001
0
20:31 20:33 20:36 20:39 20:42 20:45

Memory usage ⓘ
Memory (bytes)
Time
1.05 Gi
715 Mi
477 Mi
238 Mi
0
20:31 20:33 20:36 20:39 20:42 20:45

Replication Controllers

Name	Labels	Pods	Age	Images
es-rc	name: es-rc-pod	1 / 1	7 days	kgignatyev/es:2017-01-29-20-24-21
travelog-ws-rc	app: travelog-ws-pod	1 / 1	10 days	kgignatyev/travelog-ws:2017-01-25-22-21-23

Daemon Sets

Name	Labels	Pods	Age	Images
fluentd-agent	app: fluentd-agent tier: logging version: v1	1 / 1	7 days	kgignatyev/fluentd:2017-01-29-17-3...

Pods

Name	Status	Restarts	Age	CPU (cores)	Memory (bytes)	⋮
es-rc-eb08fb5e84e82d7947a2651151bc6b...	Running	3	7 days	0.002	339.379 Mi	⋮
fluentd-agent-4fz72	Running	3	7 days	0.002	68.348 Mi	⋮
travelog-ws-rc-w20fr	Running	6	10 days	0	527.289 Mi	⋮

How to know where things are and what is going on:



CLI: kubectl

```
Konstantins-Mac-Pro:~ kgignatyev$ kubectl get secrets
NAME                           TYPE              DATA   AGE
default-token-m26lu            kubernetes.io/service-account-token  3      88d
fluentd-conf                   Opaque             1      7d
kgignatyev-registrykey        kubernetes.io/dockercfg       1      56d
mcytravel-properties           Opaque             1      88d
mcytravel-ssl                  Opaque             2      54d
travel-properties               Opaque             1      10d

Konstantins-Mac-Pro:~ kgignatyev$ kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
es-rc-eb08fb5e84e82d7947a2651151bc6b3-sj366   1/1     Running   3          7d
fluentd-agent-4fz72                1/1     Running   3          7d
travelog-ws-rc-w20fr              1/1     Running   6          10d

Konstantins-Mac-Pro:~ kgignatyev$ kubectl get services
NAME            CLUSTER-IP   EXTERNAL-IP  PORT(S)         AGE
kubernetes      10.0.0.1    <none>        443/TCP        89d
travelog-es     10.0.0.92   <nodes>       9200:31920/TCP,9300:31930/TCP   7d
travelog-ws-service 10.0.0.145 <none>        9100/TCP       10d

Konstantins-Mac-Pro:~ kgignatyev$ kubectl get rc
NAME        DESIRED   CURRENT   READY   AGE
es-rc        1          1          1      7d
travelog-ws-rc 1          1          1      10d

Konstantins-Mac-Pro:~ kgignatyev$
```

How to know where things are and what is going on:



SSH to a running pod and execute desired commands

`kubectl exec -ti travelog-ws-rc-w20fr sh`

```
Konstantins-Mac-Pro:~ kgignatyev$ kubectl exec -ti travelog-ws-rc-w20fr sh
/ # env
TRAVELOG_ES_PORT_9200_TCP_PORT=9200
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.0.0.1:443
TRAVELOG_ES_PORT_9300_TCP_PORT=9300
TRAVELOG_ES_PORT_9200_TCP_PROTO=tcp
TRAVELOG_ES_PORT_9300_TCP_PROTO=tcp
TRAVELOG_ES_SERVICE_HOST=10.0.0.92
HOSTNAME=traveleg-ws-rc-w20fr
SHLVL=1
HOME=/root
JAVA_VERSION_BUILD=13
TRAVELOG_ES_PORT_9200_TCP=tcp://10.0.0.92:9200
TRAVELOG_ES_PORT=tcp://10.0.0.92:9200
TRAVELOG_ES_SERVICE_PORT=9200
TRAVELOG_ES_PORT_9300_TCP=tcp://10.0.0.92:9300
TRAVELOG_WS_SERVICE_PORT_9100_TCP_ADDR=10.0.0.145
JAVA_VERSION_MAJOR=8
TRAVELOG_WS_SERVICE_PORT_9100_TCP_PORT=9100
KUBERNETES_PORT_443_TCP_ADDR=10.0.0.1
TRAVELOG_WS_SERVICE_PORT_9100_TCP_PROTO=tcp
TRAVELOG_ES_SERVICE_PORT_SVC_PORT=9200
TRAVELOG_WS_SERVICE_SERVICE_HOST=10.0.0.145
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/jdk/bin
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
```

UNIX

Where there is a shell, there is a way.





How to scale :

Manually: `kubectl --scale=x -f ...`
Auto: Horizontal Pod Autoscale

Support for Horizontal Pod Autoscaler in kubectl

Horizontal Pod Autoscaler, like every API resource, is supported in a standard way by `kubectl`. We can create a new autoscaler using `kubectl create` command. We can list autoscalers by `kubectl get hpa` and get detailed description by `kubectl describe hpa`. Finally, we can delete an autoscaler using `kubectl delete hpa`.

In addition, there is a special `kubectl autoscale` command for easy creation of a Horizontal Pod Autoscaler. For instance, executing `kubectl autoscale rc foo --min=2 --max=5 --cpu-percent=80` will create an autoscaler for replication controller foo, with target CPU utilization set to 80% and the number of replicas between 2 and 5. The detailed documentation of `kubectl autoscale` can be found [here](#).

Support for custom metrics

Kubernetes 1.2 adds alpha support for scaling based on application-specific metrics like QPS (queries per second) or average request latency.

- `KUBE_ENABLE_CLUSTER_AUTOSCALER` - it enables cluster autoscaler if set to true.
- `KUBE_AUTOSCALER_MIN_NODES` - minimum number of nodes in the cluster.
- `KUBE_AUTOSCALER_MAX_NODES` - maximum number of nodes in the cluster.

How to have outage-less updates



Rolling updates

```
kubectl create -f travelog-webapp/target/k8s/travelog-ws-rc.yml --image=<new image>
```

k8s start new pods based on new image, reroutes traffic to them and then kills old pods.

Passing a configuration file

To initiate a rolling update using a configuration file, pass the new file to `kubectl rolling-update`:

```
$ kubectl rolling-update NAME -f FILE
```



PersistentVolume types are implemented by storage plugins:

- GCEPersistentDisk
- AWSElasticBlockStore
- AzureFile
- AzureDisk
- FC (Fibre Channel)
- NFS
- iSCSI
- RBD (Ceph Block Device)
- CephFS
- Cinder (OpenStack block storage)
- Glusterfs
- VsphereVolume
- HostPath (single node testing only – local storage is not recommended)

How to have manage persistent volumes:

Persistent Volumes, Persistent Volume Claims

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: es-rc
spec:
  replicas: 1
  template:
    metadata:
      name: es-rc-pod
      labels:
        name: es-rc-pod
    spec:
      containers:
        - name: es-rc-pod
          image: kgignatyev/es:2017-01-29-20-24-21
          volumeMounts:
            - mountPath: "/usr/share/elasticsearch/data"
              name: elastic-search-data
          ports:
            - containerPort: 9200
            - containerPort: 9300
        volumes:
          - name: elastic-search-data
            persistentVolumeClaim:
              claimName: "pvelastic.claim"
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvelastic.claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  selector:
    matchLabels:
      name: "elastic-pv"
```

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: elastic-pv
  labels:
    name: elastic-pv
spec:
  capacity:
    storage: 40Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: '/var/data/elastic'
```

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: elastic-pv
  labels:
    name: elastic-pv
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  awsElasticBlockStore:
    volumeID: vol-ae8b5326
    fsType: ext4
```

Local dev

Cloud



kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
    - name: db
      image: mysql
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
    - name: wp
      image: wordpress
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

How to have manage resource limits:

pod resource requests;

node selectors;

namespace limits

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
    - name: nginx
      image: nginx
      imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

```
$ kubectl create namespace myspace
```

```
$ cat <<EOF > compute-resources.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    pods: "4"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
EOF
```

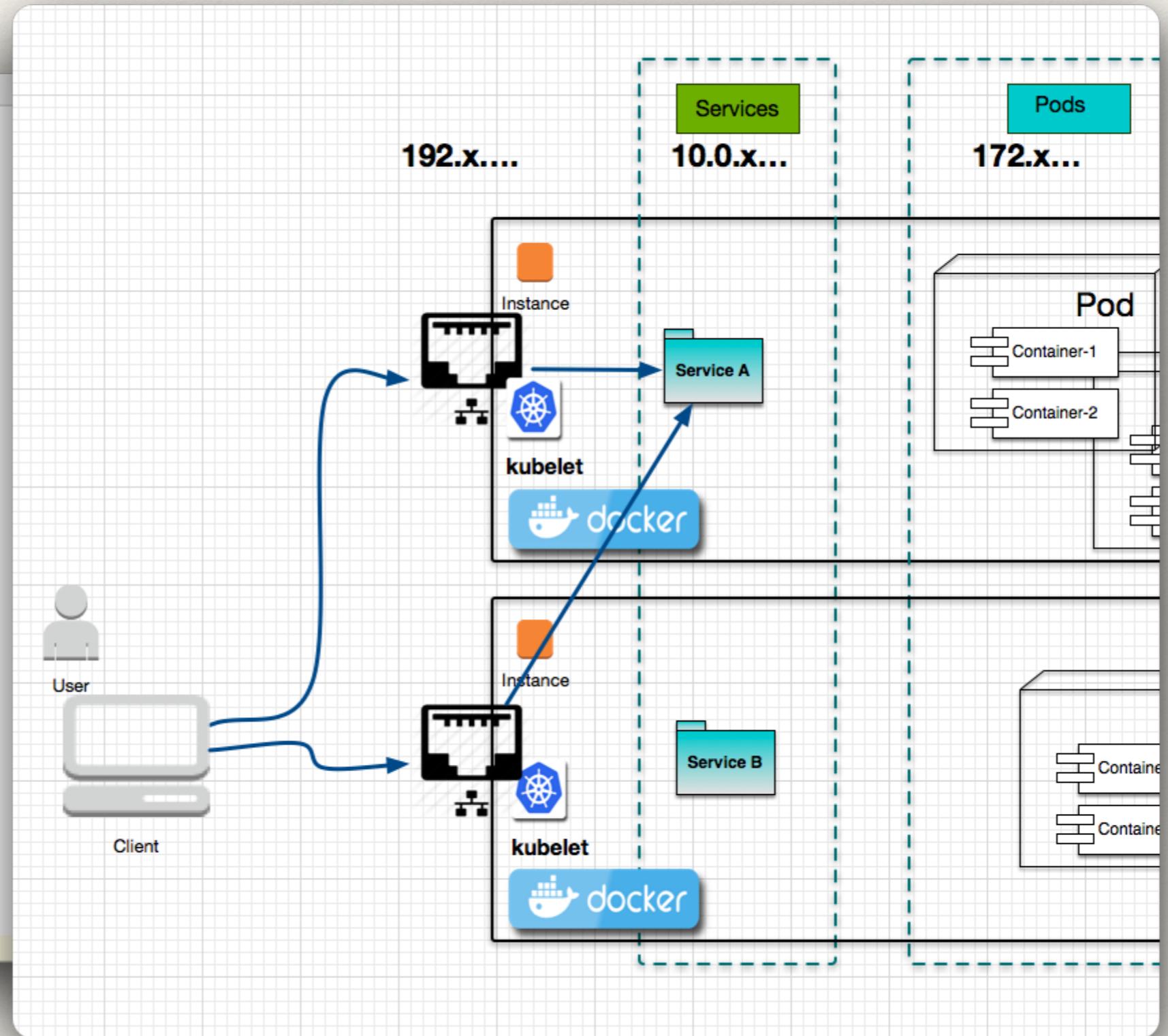
```
$ kubectl create -f ./compute-resources.yaml --namespace=myspace
```



How to expose services for external clients:

NodePort, LoadBalancer, Ingress

```
kind: Service
apiVersion: v1
metadata:
  name: travelog-es
  labels:
    name: travelog-es
spec:
  type: NodePort
  selector:
    name: es-rc-pod
  ports:
    - port: 9200
      name: "svc-port"
      targetPort: 9200
      nodePort: 31920
    - port: 9300
      name: "status-port"
      targetPort: 9300
      nodePort: 31930
```



How to have collect logs:



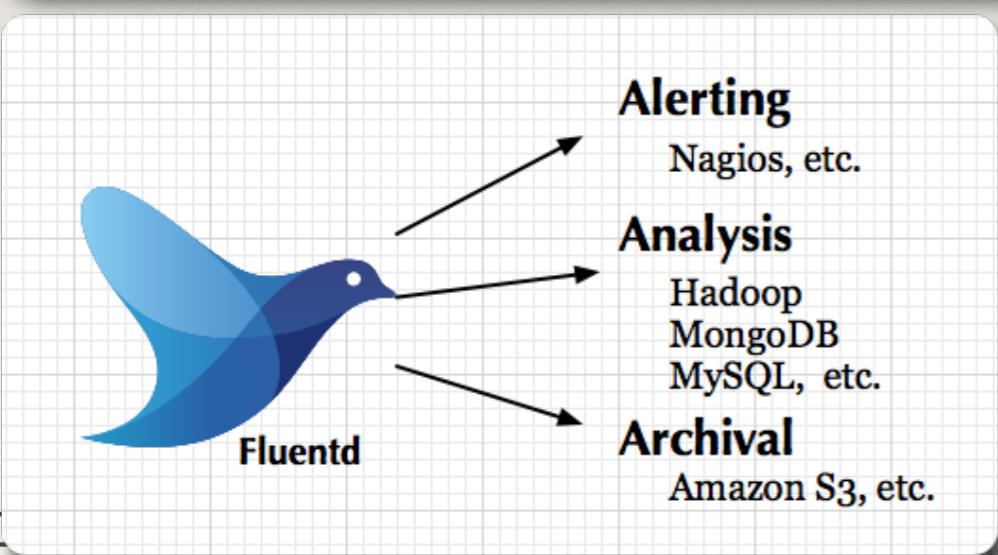
DaemonSet: guarantees that every (or selected) node run specified container, for example log collection daemon with k8s plugin to enhance container logs.

```
<source>
  type tail path
  /var/log/containers/*.log
  pos_file Fluentd-docker.pos
  time_format %Y-%m-%dT%H:%M:%S
  tag kubernetes.*
  format json
  read_from_head true
</source>

<filter kubernetes.var.lib.docker.containers.*.*.log>
  type kubernetes_metadata
</filter>

<match **>
  type stdout
</match>
```

```
metadata:
  name: fluentd-agent
  labels:
    tier: logging
    app: fluentd-agent
    version: v1
spec:
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      hostPID: true
      hostIPC: true
      hostNetwork: true
      containers:
        - resources:
            requests:
              cpu: 0.15
          securityContext:
            privileged: true
          image: kgignatyev/fluentd:2017-01-29-1
          name: fluentd
          volumeMounts:
            - name: logs
              mountPath: /var/log/containers/
              readOnly: true
            - name: mnt
              mountPath: /mnt/
            - name: config
              mountPath: /fluentd/etc/
        volumes:
          - name: config
            secret:
              secretName: "fluentd-conf"
          - name: logs
            hostPath:
              path: /var/log/containers/
          - name: mnt
            hostPath:
              path: /mnt/
```



How to have collect logs:



Log collectors have k8s aware plugins to enhance log messages with necessary details

	Table	JSON	+/-	1 sec	10 sec	30 sec	60 sec	Link to /logz-oxzrppltcfcihxkuwbxhspjqckgbugz-170206_v2/ht
⌚	@timestamp							⌚ ⓘ ⓘ ⓘ February 5th 2017, 23:30:51.000
t	docker.container_id							⌚ ⓘ ⓘ ⓘ 8d29a3aca25dbd06da56ee2259c8ee3e9348679ccab8f532becdd4dcee735b09
t	fluentd_tags							⌚ ⓘ ⓘ ⓘ kubernetes.var.log.containers.kube-addon-manager-minikube_kube-system_kube-addon-manag e2259c8ee3e9348679ccab8f532becdd4dcee735b09.log
t	kubernetes.container_name							⌚ ⓘ ⓘ ⓘ kube-addon-manager
t	kubernetes.host							⌚ ⓘ ⓘ ⓘ minikube
t	kubernetes.labels.component							⌚ ⓘ ⓘ ⓘ kube-addon-manager
?	kubernetes.labels.kubernetes_io/minikube-addons			⌚ ⓘ ⓘ ⓘ	⚠ ⚡			addon-manager
t	kubernetes.labels.version							⌚ ⓘ ⓘ ⓘ v6.1
t	kubernetes.namespace_name							⌚ ⓘ ⓘ ⓘ kube-system
t	kubernetes.pod_id							⌚ ⓘ ⓘ ⓘ 87ce0691-e5d2-11e6-a8a2-1af65948c45e
t	kubernetes.pod_name							⌚ ⓘ ⓘ ⓘ kube-addon-manager-minikube
t	log							⌚ ⓘ ⓘ ⓘ INFO: == Kubernetes addon update completed successfully at 2017-02-06T07:30:51+0000 ==
t	stream							⌚ ⓘ ⓘ ⓘ stdout
t	tags							⌚ ⓘ ⓘ ⓘ _logz_http_bulk_json_8070
t	type							⌚ ⓘ ⓘ ⓘ http-bulk

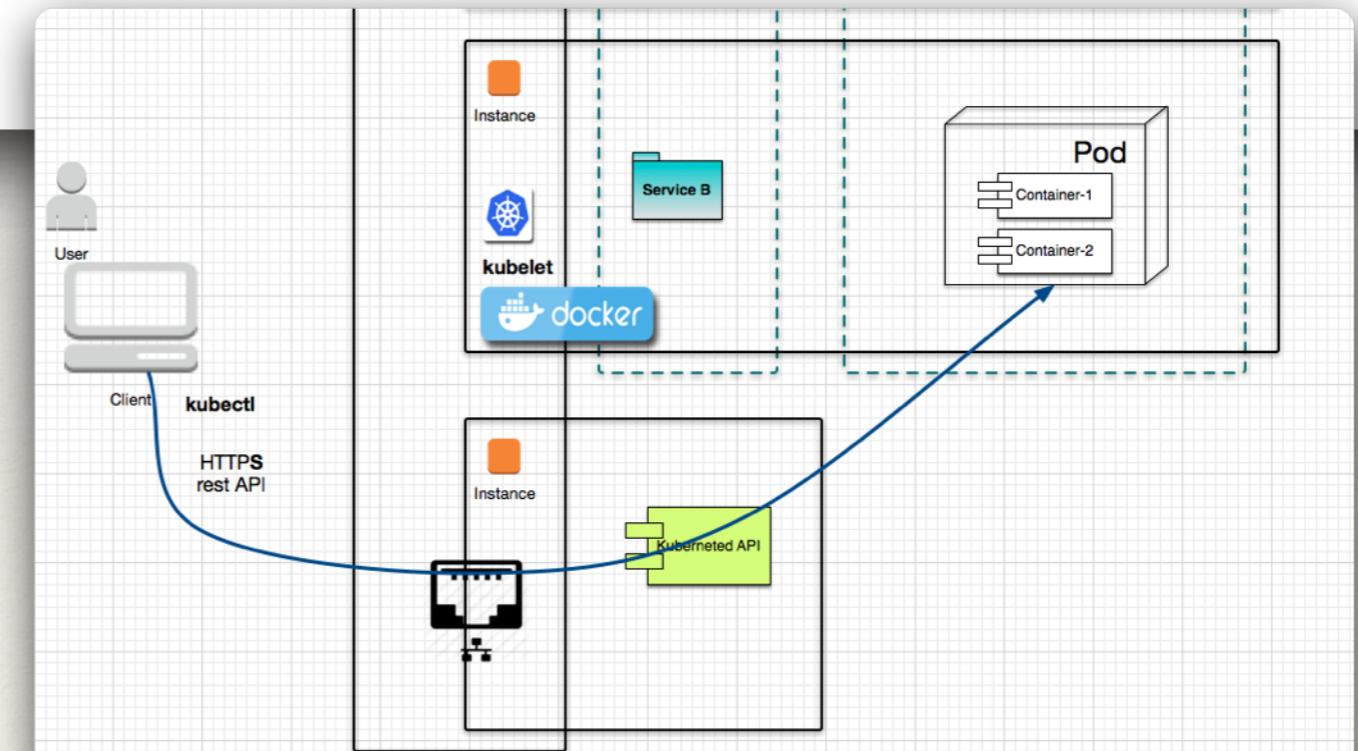
How to develop locally:



Port-forward

```
#!/usr/bin/env bash
echo "starting port forward to Elastic Search:"
podName=`kubectl get pods -l name=es-rc-pod -o=jsonpath='{.items[0].metadata.name}'`  
kubectl port-forward $podName 9200:9200
```

```
public class TravelogService {
    public String esBase() {
        String hostFromEnv = System.getenv( name: "TRAVELOG_ES_SERVICE_HOST");
        String res = "http://travelog-es:9200";
        if (hostFromEnv != null) {
            res = "http://" + hostFromEnv + ":" + System.getenv( name: "TRAVELOG_ES_PORT_9200_TCP_PORT");
        }
        return res;
    }
}
```



Minikube



[build](#) passing [codecov](#) 33%



What is Minikube?

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

Features

- Minikube packages and configures a Linux VM, the container runtime, and all Kubernetes components, optimized for local development.
- Minikube supports Kubernetes features such as:
 - DNS
 - NodePorts
 - ConfigMaps and Secrets
 - Dashboards
 - Container Runtime: Docker, and [rkt](#)
 - Enabling CNI (Container Network Interface)

How to run batch and cron jobs:



BatchJob, CronJob, ‘naked’ pod

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      name: pi
    spec:
      containers:
        - name: pi
          image: perl
          command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
  restartPolicy: Never
```

```
apiVersion: batch/v2alpha1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/* * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: hello
              image: busybox
              args:
                - /bin/sh
                - -c
                - date; echo Hello from the Kubernetes cluster
  restartPolicy: OnFailure
```

Questions ?