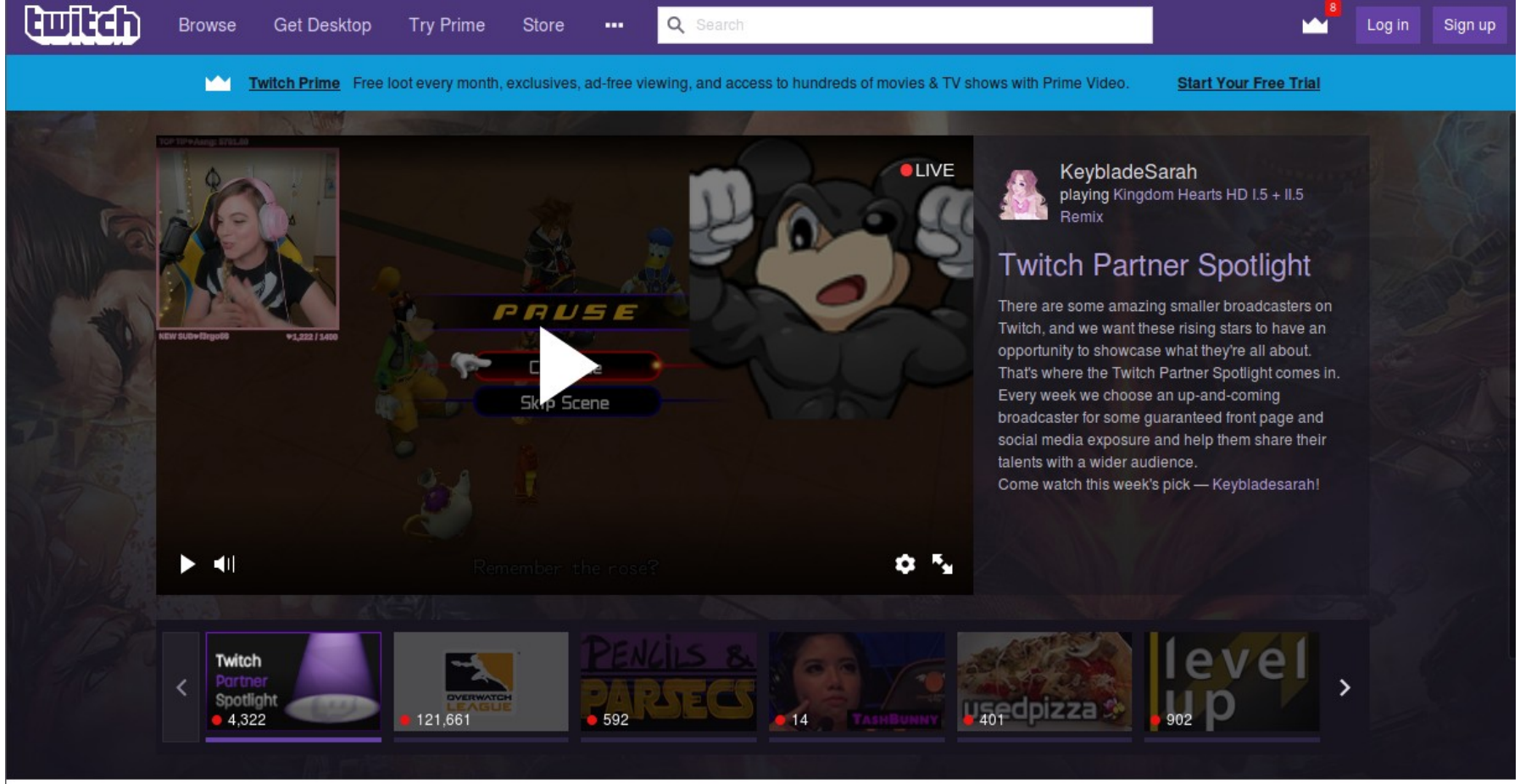
Kubernetes: A complete application deployment with encryption

Scale16x

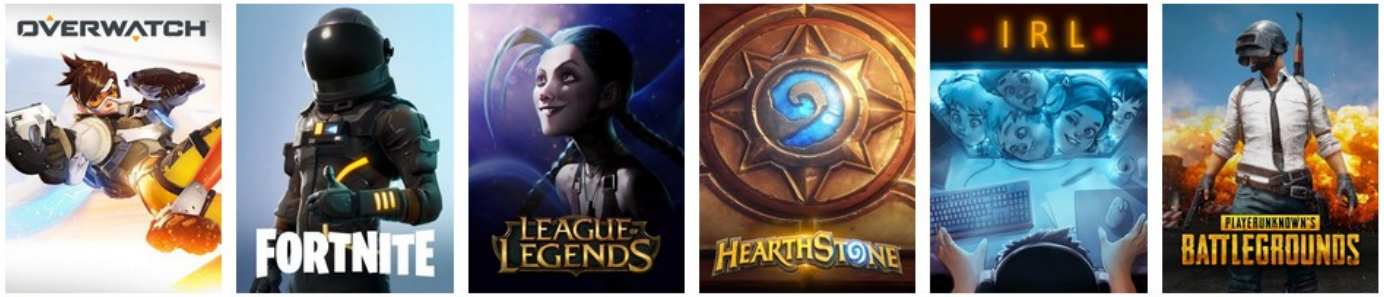
@mpetason

Who Am I?

- Mike Petersen
- Developer Advocate @ IBM
- K8s/Containers/esports
- https://github.com/mpetason/



Featured Games Games people are watching now

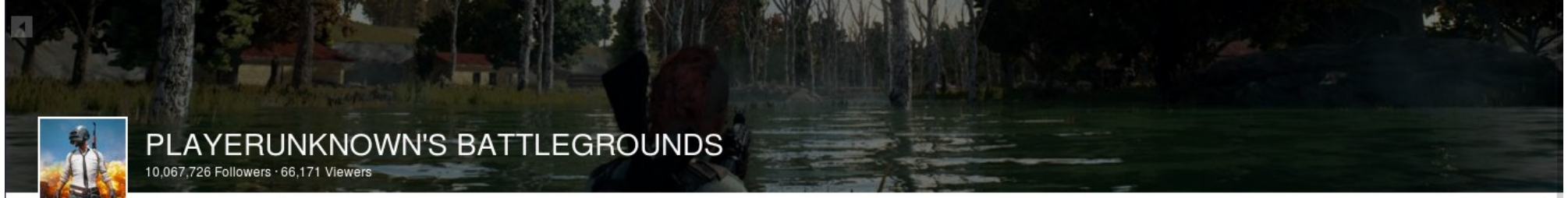


Overwatch 143,356 viewers
Fortnite 129,356 viewers
League of Legends 81,078 viewers
Hearthstone 45,520 viewers
IRL 29,812 viewers
PLAYERUNKNOWN'... 23,696 viewers

@mpetason

Twitch

- Live streams of video games and other interests.
- Most of the top streams focus on esports related games, however other games can spike to the top during release or heavy promotion.
- Streams can get 200k+ viewers during bigger events.



PLAYERUNKNOWN'S BATTLEGROUNDS

10,067,726 Followers · 66,171 Viewers

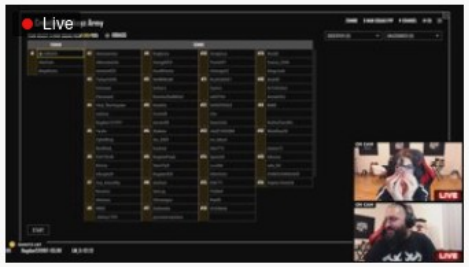
Live Channels | Videos | Clips

Follow

Language 1

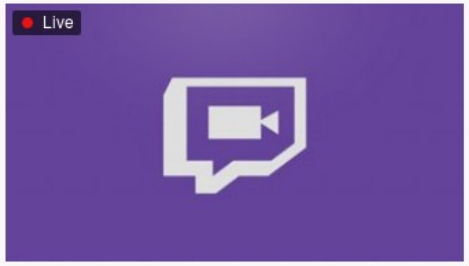
I'm the captain now
5,017 viewers on chocoTaco

GLL Season 1 - SUPER WEEK - Alpha Division EU…
4,555 viewers on GLL



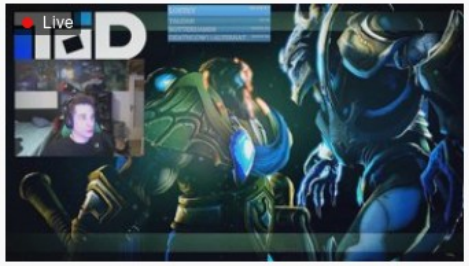
Duetul Salbatec
2,432 viewers on CreativeMonkeyz

PUBG and maybe Siege?
1,201 viewers on TSM_SmaK



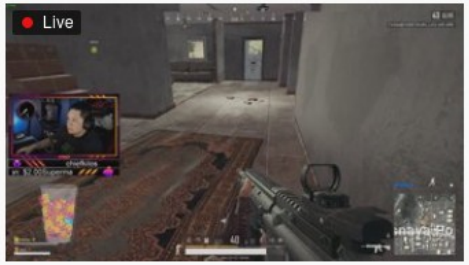
TSM_BreaK // Going for some high kill games
1,064 viewers on BreaK

FPP - Still trash but slightly improved trash.
845 viewers on AnneMunition



!giveaway [1080] PUBG
652 viewers on ToD

🍁 Crate Hunter Halifax! Show Me Your faxNade! 🍁
547 viewers on Halifax



Ump Day | vsnz
461 viewers on vsnz

GLL Season 1 - SUPER WEEK - Bravo Division E…
374 viewers on GLL2

WACKYJACKY101 - Solos and maybe some rand…
356 viewers on Wackyjacky101

TSM_rawryy // Back from IEM/SL, trying new mice…
333 viewers on TSM_rawryy

@mpetason

PlayerUnknown’s Battlegrounds

- Battle Royale – blends survival, exploration, and scavenging in a last man standing game.
- Each match starts with players parachuting from a plane onto a map area approximately 8 by 8 kilometres (5.0 mi × 5.0 mi) in size.
- Players start the game with nothing and are forced to search for weapons, armor, vehicles, and consumables.

ROTISSERIE

About Contact Related

Current: Wackyjacky101 - 2

Next Closest: ItsMeHawaii - 2 →

Pin Stream | View the Code | Fork on Github

@mpetason

The Application: Rotisserie.tv

- Rotisserie displays PlayerUnknown's Battlegrounds streams with the least amount of players left standing.
- Streams are pulled from Twitch using Livestreamer
- The number alive is cropped out and is sent through Tesseract-OCR.
- https://github.com/IBM/rotisserie

https://rotisserie.tv/all

JSON Raw Data Headers

Save Copy


```
0:
    stream_name:    "cheese05"
    alive:          3
    stream_url:     "https://player.twitch.tv/?channel=cheese05"
    updated:        "2018-03-06T17:29:50.713Z"
1:
    stream_name:    "yeahunter"
    alive:          58
    stream_url:     "https://player.twitch.tv/?channel=yeahunter"
    updated:        "2018-03-06T17:29:50.713Z"
2:
    stream_name:    "TSM_Viss"
    alive:          100
    stream_url:     "https://player.twitch.tv/?channel=TSM_Viss"
    updated:        "2018-03-06T17:29:50.713Z"
```


https://rotisserie.tv/current

JSON Raw Data Headers

Save Copy


```
stream_name:    "CanSungur"
alive:          25
stream_url:     "https://player.twitch.tv/?channel=CanSungur"
updated:        "2018-03-06T17:50:20.577Z"
```


@mpetason

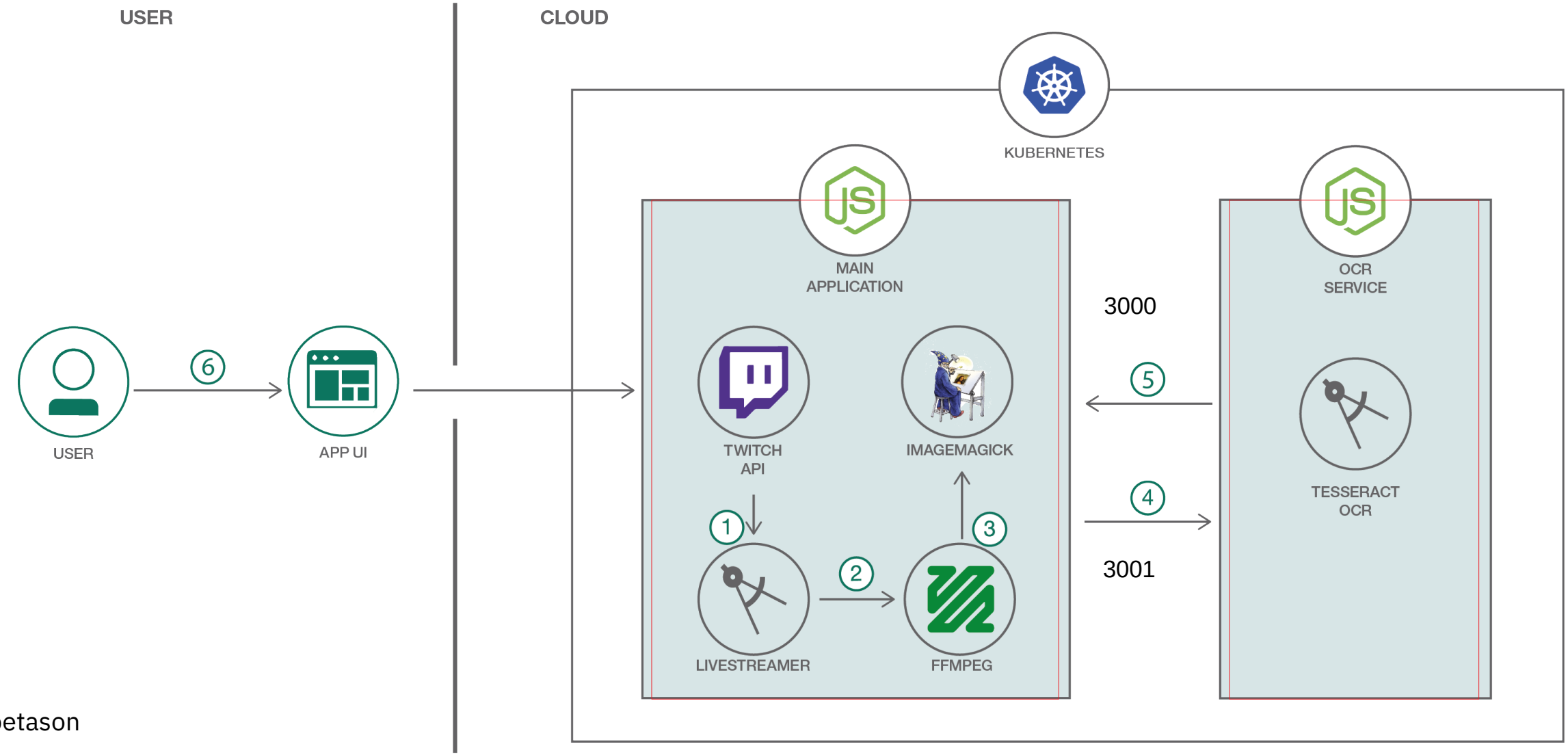
Tesseract OCR

- Tesseract OCR is an optical character recognition engine for various operating systems.
- Open Sourced in 2005.
- OCR – conversion of images into text.
- https://github.com/tesseract-ocr/tesseract

Livestreamer

- Livestreamer is a command-line utility that pipes video streams from various services into a video player.
- The main purpose of Livestreamer is to allow the user to avoid buggy and CPU heavy flash plugins but still be able to enjoy various streamed content.
- https://github.com/chrippa/livestreamer

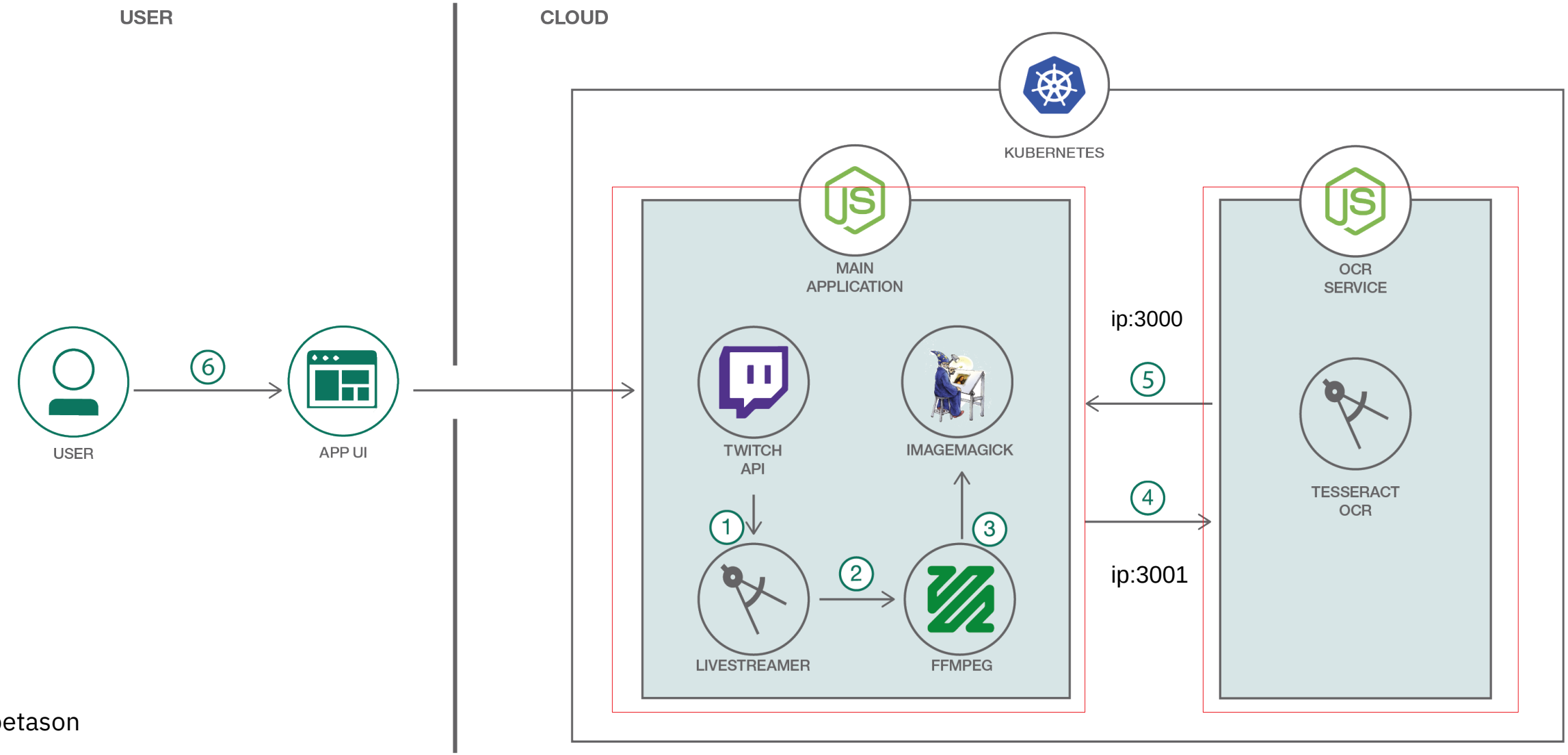
Node.js Application Architecture



Node.js

- It was easy to setup locally for development.
- The team could get up to speed since it was easy to use.
- Downside: dependencies based on versions. We ran into a few issues with dependencies across different development environments.

Docker Architecture



Docker

- Made development across OS/Environments easier.
- Standardized on a single version of Node
- Perfect for not having to install requirements on the local system.
- Easier to split services up into distinct containers – Microservices.

Microservices

- Structures an application as a collection of loosely coupled services.
- Breaks up the application into smaller parts, that can be deployed independently.
- Makes it easier to drop in new pieces as replacements.
- Best example in our application : Tesseract-OCR

App Dockerfile

```
FROM node:8-alpine

COPY app.js /
COPY package.json /
COPY package-lock.json /
COPY public /public
RUN echo http://nl.alpinelinux.org/alpine/edge/testing >> /etc/apk/repositories && \
    apk add --no-cache livestreamer ffmpeg imagemagick git py2-singledispatch && \
    npm install

ARG OCR_HOST
ENV OCR_HOST=$OCR_HOST

EXPOSE 3000

CMD ["node", "app.js"]
```

OCR Dockerfile

```
FROM node:8-alpine

COPY ocr.js /
COPY package.json /
COPY package-lock.json /
RUN apk --update --no-cache --virtual wget-deps add ca-certificates openssl && \
    apk --no-cache add tesseract-ocr git && \
    wget -q -P /usr/share/tessdata/ https://github.com/tesseract-ocr/tessdata/raw/master/eng.traineddata && \
    apk del wget-deps && \
    npm install

EXPOSE 3001

CMD ["node", "ocr.js"]
```

Static Dockerfile

```
FROM nginx:1.13-alpine

RUN apk add --no-cache bash

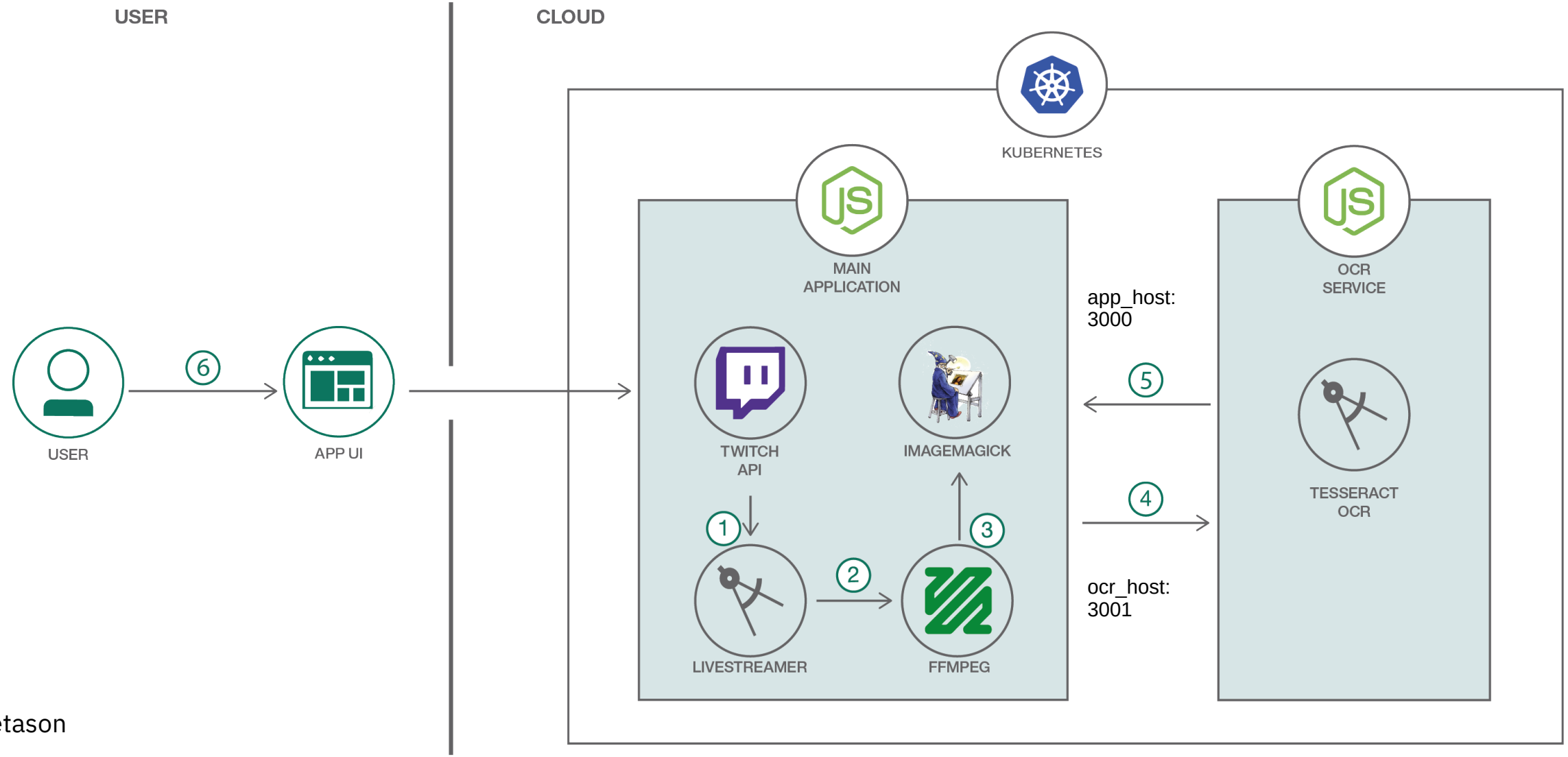
COPY ./conf/nginx.conf /etc/nginx/nginx.conf
USER nginx
COPY ./conf/static-nginx.conf /nginx-sites/rotisserie-static.template
COPY ./public /rotisserie-static
USER root
RUN mkdir /var/run/nginx
RUN chown nginx:root /nginx-sites
RUN chown nginx:nginx /var/run/nginx

USER nginx
CMD ["/bin/bash", "-c", "envsubst < /nginx-sites/rotisserie-static.template > /nginx-sites/rotisserie-static.conf &&
nginx"]
```


Alpine Base + Node:Alpine

- Alpine Linux is a Linux distribution built around musl libc and BusyBox. The image is only 5 MB in size and has access to a package repository that is much more complete than other BusyBox based images.
- Small container images that can be used as a base.
- Node:alpine allows us to have node installed by default for our App + OCR containers.
- Great when uploading images constantly on a slow connection.
- We started with Ubuntu images which were a big bigger to start with.
- https://hub.docker.com/_/alpine/
- https://hub.docker.com/_/node/

Kubernetes Architecture



Kubernetes

- We can take advantage of IBM Cloud and expose it externally.
- Easier to scale up and down.
- Prioritizes splitting applications up into microservices.

KubeDNS and Environment Variables

- Kubernetes DNS schedules a DNS Pod and Service on the cluster, and configures the kubelets to tell individual containers to use the DNS Service's IP to resolve DNS names.

- ```
  $ k exec rotisserie-app-5dfdcd96bc-vrbzp env|grep -i host
  HOSTNAME=rotisserie-app-5dfdcd96bc-vrbzp
  ROTISSERIE_APP_SERVICE_HOST=172.21.82.245
  ROTISSERIE_STATIC_SERVICE_HOST=172.21.109.199
  ROTISSERIE_OCR_SERVICE_HOST=172.21.36.196
  DEFAULT_HTTP_BACKEND_SERVICE_HOST=172.21.203.197
  NGINX_SERVICE_HOST=172.21.38.164
  KUBE_LEGO_NGINX_SERVICE_HOST=172.21.134.15
  KUBERNETES_SERVICE_HOST=172.21.0.1
  ```



  ```

Kubernetes Deployment

- Currently managing with bash scripts.
- Future – Looking into Helm, or for alternatives to using a make file as we expand.

```
IMAGE_TAG=$$(cat $(REV_FILE)) envsubst < deploy/rotisserie.yaml | kubectl apply -f -
```


```yaml
apiVersion: v1
kind: Service
metadata:
  name: rotisserie-static
spec:
  ports:
  - port: 8082
    protocol: TCP
    name: rotisserie-static
  selector:
    app: rotisserie-static
---
apiVersion: v1
kind: Service
metadata:
  name: rotisserie-ocr
spec:
  ports:
  - port: 3001
    protocol: TCP
    name: rotisserie-ocr
  selector:
    app: rotisserie-ocr
---
apiVersion: v1
kind: Service
metadata:
  name: rotisserie-app
spec:
  ports:
  - port: 3000
    protocol: TCP
    name: rotisserie-app
  selector:
    app: rotisserie-app
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: rotisserie-static
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: rotisserie-static
    spec:
      containers:
      - name: rotisserie-static
        image: $docker_username/rotisserie-static:$IMAGE_TAG
        imagePullPolicy: Always
        env:
        - name: NGINX_LISTEN
          value: "*:8082"
        ports:
        - containerPort: 8082
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: rotisserie-app
spec:
  template:
    metadata:
      labels:
        app: rotisserie-app
    spec:
      containers:
        - name: rotisserie-app
          image: $docker_username/rotisserie-app:$IMAGE_TAG
          env:
          - name: token
            valueFrom:
              secretKeyRef:
                name: rotisserie-secrets
                key: token
          - name: clientID
            valueFrom:
              secretKeyRef:
                name: rotisserie-secrets
                key: clientID
          ports:
            - containerPort: 3000
```

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: rotisserie-ocr
spec:
  template:
    metadata:
      labels:
        app: rotisserie-ocr
    spec:
      containers:
        - name: rotisserie-ocr
          image: $docker_username/rotisserie-ocr:$IMAGE_TAG
          ports:
            - containerPort: 3001
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: rotisserie-ingress
  annotations:
    kubernetes.io/tls-acme: "true"
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
spec:
  tls:
  - secretName: rotisserie-tls
    hosts:
    - $APP_HOSTNAME
  rules:
  - host: $APP_HOSTNAME
    http:
      paths:
      - path: /current
        backend:
          serviceName: rotisserie-app
          servicePort: 3000
      - path: /
        backend:
          serviceName: rotisserie-static
          servicePort: 8082
      - path: /all
        backend:
          serviceName: rotisserie-app
          servicePort: 3000
```


```
$ k get svc,deploy,ingress,pod
NAME                    TYPE          CLUSTER-IP     EXTERNAL-IP      PORT(S)                    AGE
svc/kube-lego-nginx     ClusterIP     10.10.10.93    <none>           8080/TCP                   3d
svc/kubernetes          ClusterIP     10.10.10.1     <none>           443/TCP                    125d
svc/nginx               LoadBalancer  10.10.10.132   169.60.141.165   80:30230/TCP,443:30417/TCP 3d
svc/rotisserie-app      ClusterIP     10.10.10.243   <none>           3000/TCP                   3d
svc/rotisserie-ocr      ClusterIP     10.10.10.196   <none>           3001/TCP                   3d
svc/rotisserie-static   ClusterIP     10.10.10.9     <none>           8082/TCP                   3d

NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deploy/kube-lego        1         1         1            1           3d
deploy/nginx            1         1         1            1           3d
deploy/rotisserie-app   1         1         1            1           3d
deploy/rotisserie-ocr   1         1         1            1           3d
deploy/rotisserie-static 1        1         1            1           3d

NAME                    HOSTS                       ADDRESS          PORTS     AGE
ing/kube-lego-nginx     rotisserie.tv,www.rotisserie.tv  10.186.59.103...  80        3d
ing/rotisserie-ingress  rotisserie.tv,www.rotisserie.tv  10.186.59.103...  80, 443   3d

NAME                                READY     STATUS    RESTARTS   AGE
po/kube-lego-337898312-mhq8c        1/1       Running   0          3d
po/nginx-364277614-4k0t3            1/1       Running   9          3d
po/rotisserie-app-1965680393-d67lb  1/1       Running   7          3d
po/rotisserie-ocr-1590928554-w7ndb  1/1       Running   0          3d
po/rotisserie-static-1929736494-04z7g 1/1     Running   0          3d
```


Kubernetes Dashboard

- Quick overview of the health of the infrastructure.
- kubectl proxy
- http://127.0.0.1:8001/ui

Cluster

- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace

default

Overview

Workloads

- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Discovery and Load Balancing

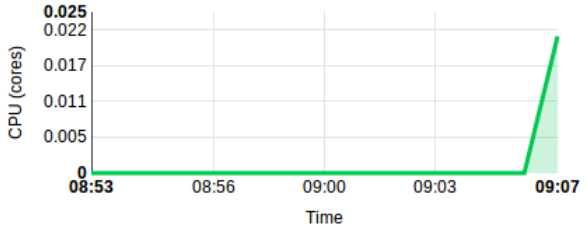
- Ingresses
- Services

Config and Storage

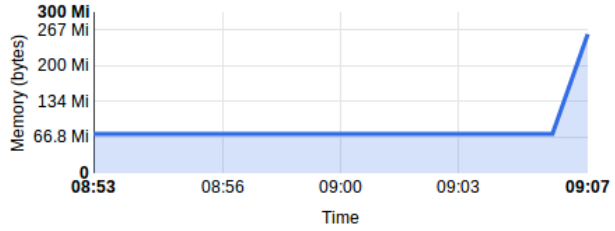
- Config Maps
- Persistent Volume Claims
- Secrets

About

CPU usage



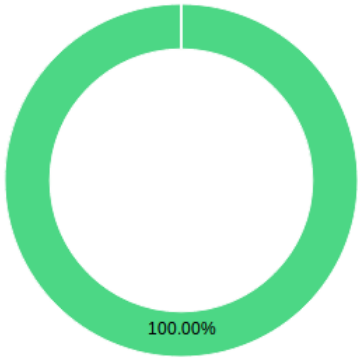
Memory usage



Resource Status

Pods

Running	5
Pending	0
Failed	0



Pods

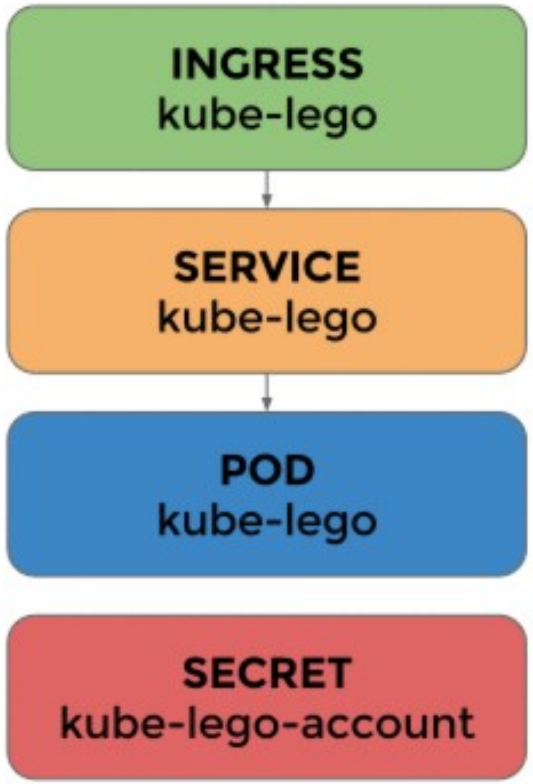
Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
rotisserie-ocr-1590	10.186.59.105	Running	0	57 seconds	-	65.164 Mi
rotisserie-app-196	10.186.59.103	Running	0	a minute	-	53.160 Mi
kube-lego-337898	10.186.59.98	Running	0	16 hours	0	9.539 Mi
nginx-364277614-	10.186.59.103	Running	9	16 hours	0.021	130.285 Mi
rotisserie-static-19	10.186.59.98	Running	0	16 hours	0	1.313 Mi

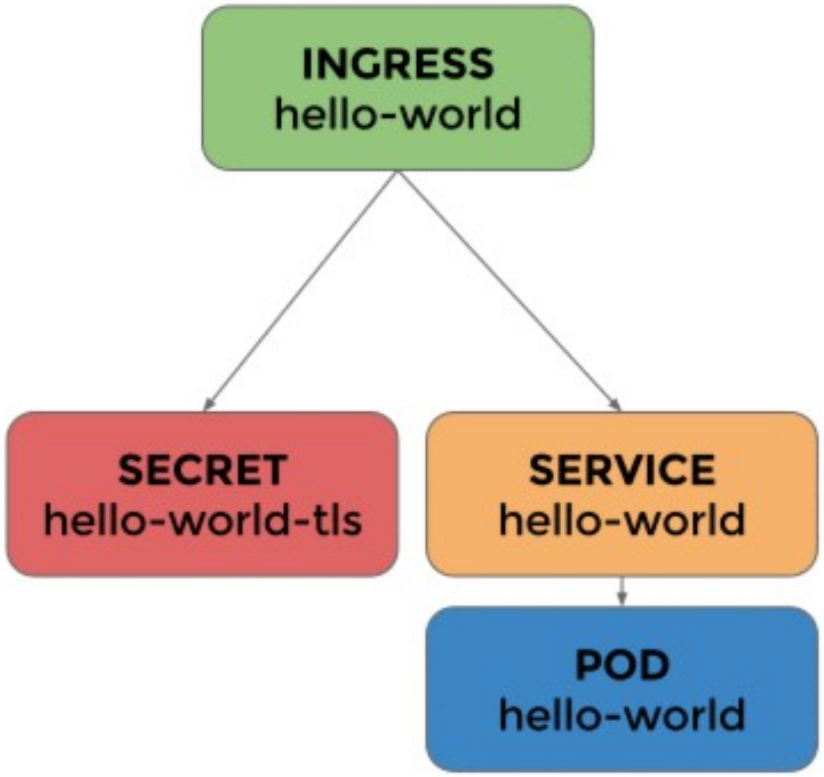
@mpetason

Kube-Lego

- Automated management of certificates.
- Configures certificates based on ingress resources, which makes it easier to add/manage multiple hostnames.
- Auto redirects and forces encryption based on annotations.
- Kube-Lego has been discontinued, with new development on: https://github.com/jetstack/cert-manager/

Kube-Lego Architecture





JETSTACK©

What’s Ingress?

- A collection of rules that allow inbound connections to reach the cluster services.
- In Kubernetes it is managed by an Ingress Controller, which watches for changes in Ingress Resources.
- In our case the Ingress Controller is Nginx.
- The Ingress Controller watches for changes to Ingress Resources, then uses the new information to reload Nginx.

Kubernetes Ingress

Rotisserie-Ingress yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: rotisserie-ingress
  annotations:
    kubernetes.io/tls-acme: "true"
    ingress.kubernetes.io/ssl-redirect: "true"
    kubernetes.io/ingress.class: nginx
spec:
  tls:
  - secretName: rotisserie-tls
    hosts:
    - $APP_HOSTNAME
  rules:
  - host: $APP_HOSTNAME
    http:
      paths:
      - path: /current
        backend:
          serviceName: rotisserie-app
          servicePort: 3000
      - path: /
        backend:
          serviceName: rotisserie-static
          servicePort: 8082
      - path: /all
        backend:
          serviceName: rotisserie-app
          servicePort: 3000
```

Nginx Deployment yaml

```
spec:
    containers:
    - args:
      - /nginx-ingress-controller
      - --default-backend-service=default/rotisserie-static
      - --nginx-configmap=default/nginx
      env:
      - name: POD_NAME
        valueFrom:
          fieldRef:
            apiVersion: v1
            fieldPath: metadata.name
      - name: POD_NAMESPACE
        valueFrom:
          fieldRef:
            apiVersion: v1
            fieldPath: metadata.namespace
      image: gcr.io/google_containers/nginx-ingress-controller:0.8.3
      imagePullPolicy: Always
      name: nginx
      ports:
      - containerPort: 80
        protocol: TCP
      - containerPort: 443
        protocol: TCP
```


Issues we ran into

- RBAC was added to Kubernetes 1.8. After upgrading the cluster we had to create user permissions for Kube-Lego to work.
- OCR doesn’t always give accurate results. Anything overlapping the number of players alive can mess up character recognition.
- Not all ingress controllers work with Kube-Lego, even if they are based on Nginx. We spent a while troubleshooting why two ingress resources with the same hostname (required for kube-lego) weren’t being read into the Nginx configuration.

Future Projects

- Porting to Fortnite (soonish).
- Use Istio for mobile optimization.
- Image and object tracking with paper Magic: The Gathering

Keeping up with Kube

- Community: https://kubernetes.io/community/
- Slack: http://slack.k8s.io/
- https://github.com/kubernetes/kubernetes

Questions?