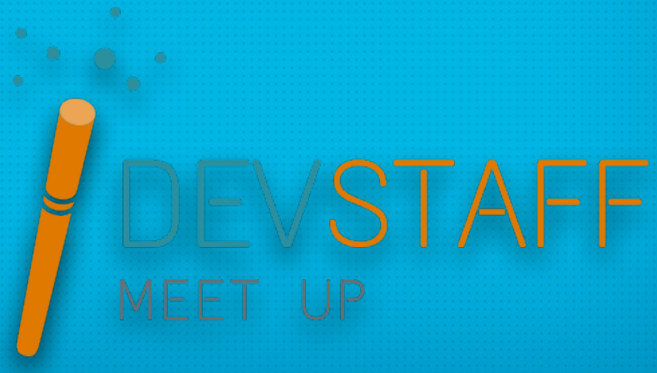
A HOME FOR YOUR MICROSERVICES:

K8S



BORING

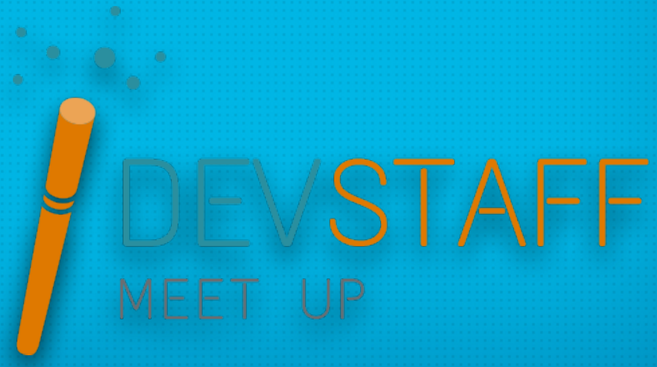
kubernetes

A HOME FOR YOUR MICROSERVICES:

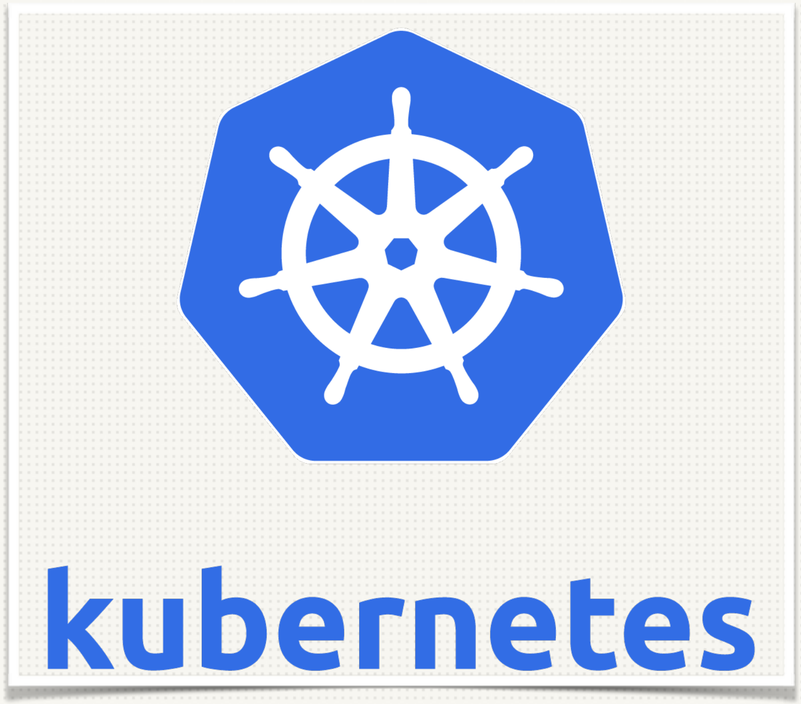
~~K8S~~

KATE'S

woohoo!! click-bait title!!



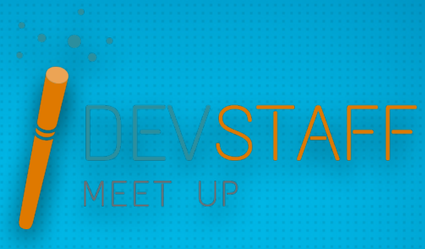
OUR GOAL FOR TODAY


kubernetes

+


WordPress

kubernetes

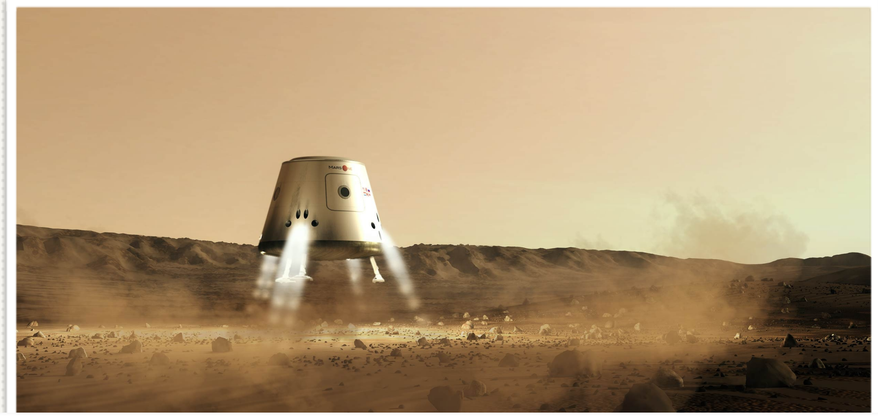


A HOME FOR YOUR CODE

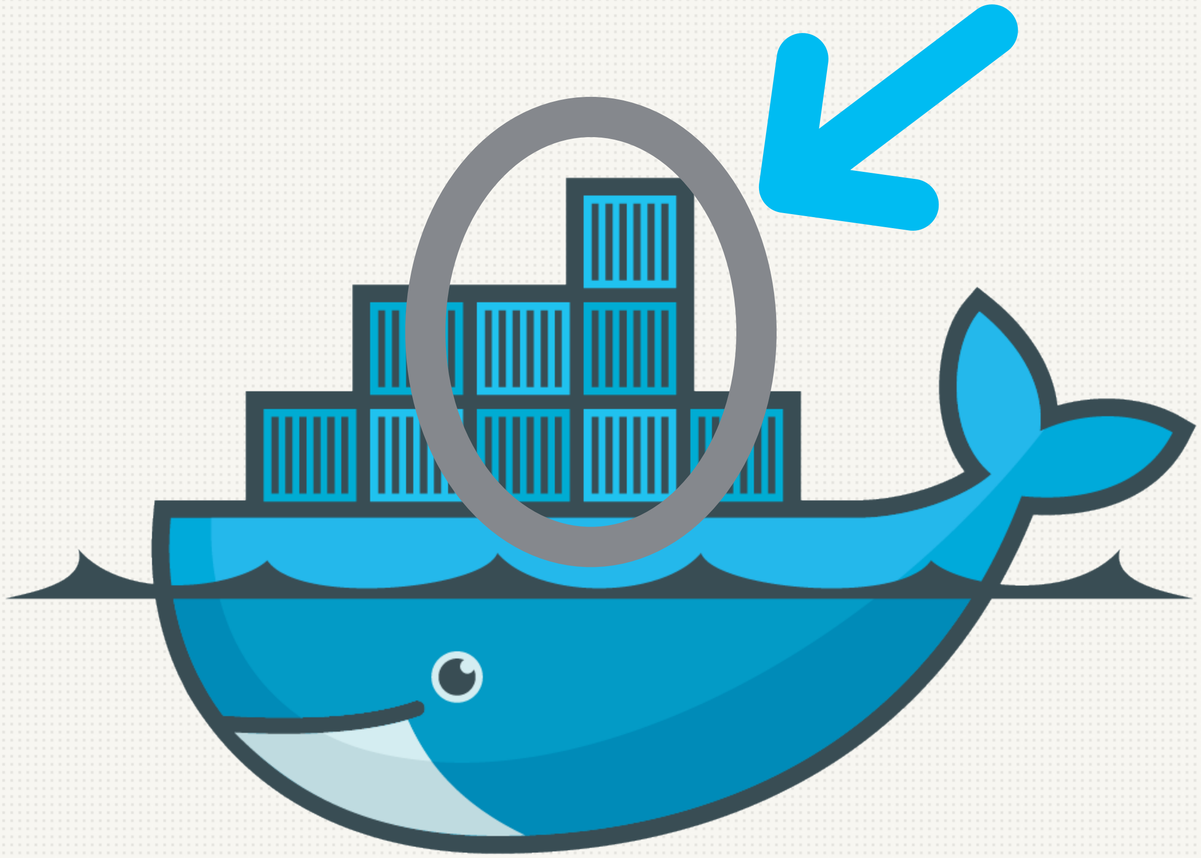
PODS


Ingeniously Crunchy
Pods
176g
HONEYCOMB BITES
ABUZZ WITH HONEYCOMB FLAVOUR
Ingeniously Crunchy
Pods
176g
Mint Slice
Pods
176g
COOKIES & CREAM

... NOT !!



PODS



docker

WORDPRESS_POD.YAML

```
  metadata:
    labels:
      app: wordpress
      tier: frontend
  spec:
    containers:
    - image: wordpress:4.8.0-apache
      name: wordpress
      env:
      - name: WORDPRESS_DB_HOST
        value: wordpress-mysql
      - name: WORDPRESS_DB_PASSWORD
        value: a super secure password for my mysql 1
```

Labels: for reporting + linking to other components

Docker image

env vars

A HOME FOR YOUR CONFIG

CONFIG MAPS

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: logstash-config
  namespace: default
data:
  log4j2_properties: |-
    status = error
    name = LogstashPropertiesConfig

    appender.console.type = Console
    appender.console.name = plain_console
    appender.console.layout.type = PatternLayout
    appender.console.layout.pattern = [%d{ISO8601}][%-5p][%-25c] %m%n

    rootLogger.level = FATAL
    rootLogger.appenderRef.console.ref = ${sys:ls.log.format}_console
```

definition

data

SECRETS

```
apiVersion: v1
kind: Secret
metadata:
  name: wordpress-pass
type: Opaque
data:
  password.txt: #base64 encoded
```

1. DEFINE

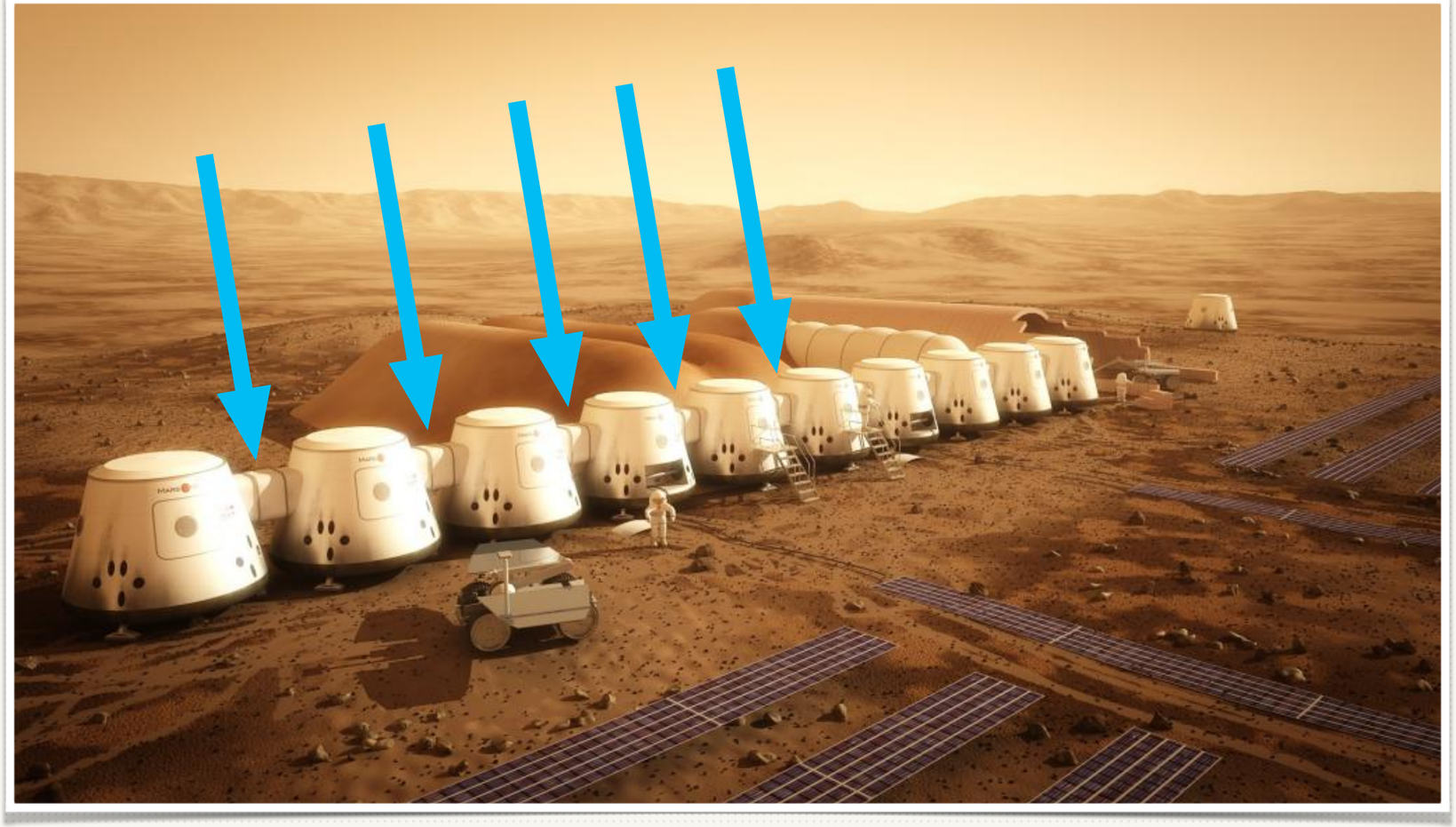
```
env:
- name: WORDPRESS_DB_HOST
  value: wordpress-mysql
- name: WORDPRESS_DB_PASSWORD
  value: a super secure password…
```

```
env:
- name: WORDPRESS_DB_PASSWORD
  valueFrom:
    secretKeyRef:
      name: wordpress-pass
      key: password.txt
```

2. USE

A HOME FOR YOUR NETWORK

SERVICES



SERVICES

- Provide Internal Load-Balancing
- Distribute traffic to Pods with specific labels
- Port mappings
- Can be bound to external Load Balancer (AWS ELB/ALB, Google LB, etc.)

PORTS

```
spec:
  containers:
  - image: wordpress:4.8.0-apache
    name: wordpress
    ports:
      - containerPort: 80
        name: wordpress
```

define ports on container

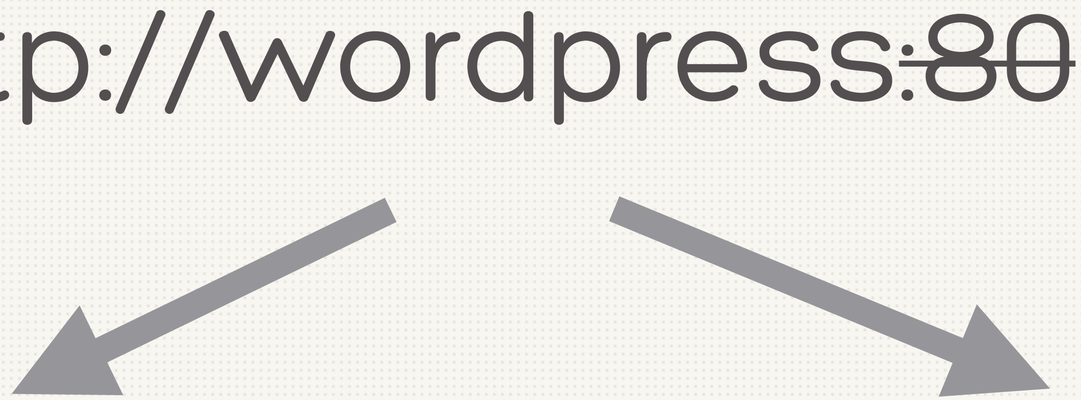
SERVICES

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  ports:
    - port: 80
  selector:
    app: wordpress
    tier: frontend
  (type: LoadBalancer)
```

ports the service listens on

bind to pods

SERVICES



Wordpress Pod 1

...

Wordpress Pod n

A HOME FOR YOUR STORAGE

VOLUMES

- “Same” as docker volumes
- Same lifecycle as Pod. Persists across restarts.
- Not same lifecycle as Container.
- Works fine on “localhost”.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: gcr.io/google_containers/
test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

PERSISTENT VOLUMES

- Persist across nodes. Lifecycle outlasts pod.

```
spec:
    containers:
    - image: wordpress:4.8.0-apache
      name: wordpress
     volumeMounts:
      - name: wordpress-persistent-storage
        mountPath: /var/www/html
    volumes:
    - name: wordpress-persistent-storage
      persistentVolumeClaim:
        claimName: wp-pv-claim
```

A HOME FOR YOUR WORK(LOAD)

JOBS

* A job creates one or more pods and ensures that a specified number of them successfully terminate.
* As pods successfully complete, the job tracks the successful completions.
* When a specified number of successful completions is reached, the job itself is complete.
* Use cases: long-running background jobs (e.g. migrations, cleanups, etc.)

DEPLOYMENT

- Uses ReplicaSets / ReplicationController
- Ensures a specified number of pod replicas are running at any one time.
- Homogeneous pods, always up and available. (if one crashes, starts a new one).
- Supports rolling updates.
- Canary updates possible through Deployments.

STATEFUL SET

- identical container spec, but unique identity for each pod
  - ordinal index
  - stable network id
  - stable storage
- Deployment guarantees:
  - Pods created sequentially, in order, from {0..N-1}.
  - Pods are terminated in reverse order, from {N-1..0}.
  - Before scaling, all predecessors must be Running and Ready.
  - Before termination, all successors must be completely shutdown.

DAEMON SET

- Ensures that all (or some) Nodes run a copy of a Pod.
- Some typical uses of a DaemonSet are:
  - cluster storage daemon, such as glusterd, ceph, on each node.
  - logs collection daemon on every node, such as fluentd or logstash.
  - node monitoring daemon, such as Prometheus Node Exporter, collectd, Datadog agent, New Relic agent, or Ganglia gmond.

SO, BACK TO WORDPRESS...


That's all Folks!

THANK YOU!

Yorgos Saslis
@gsaslis
github.com/gsaslis