SIGHUP

KUBEPRIMER

Deploying Microservices to Kubernetes

11/04/18 - Rome

kubernetes

ABOUT US

SIGHUP is specialized on **Kubernetes** and **Cloud-Native infrastructures**. We help companies and enterprises during their journey towards automated software-defined infrastructures.

The Speakers

Jacopo Nardiello
@jnardiello
Founder & DevOps Engineer
SIGHUP

Giacomo Tirabassi
@gitirabassi
DevOps & Systems Engineer
SIGHUP

Riccardo Setti
@giskarda
DevOps & Software Engineer
SIGHUP

SIGHUP

kubernetes

Introducing Kubernetes

Kubernetes (Greek κυβερνήτης “kyvernítis” for "helmsman" or "pilot")

Is an *open-source platform for* ***automating deployment***, ***scaling***, *and* ***operations*** *of application containers* across clusters of hosts

Inspired and informed by Google’s experiences [Borg, Omega]

SIGHUP

kubernetes

Agenda

- **Software Development and Containers**
  - Docker build, ship, run and compose
  - A look into orchestration with Docker swarm
- **Containers in Production with Kubernetes**
  - Kubernetes principles & core concepts
  - Kubernetes for Docker users
  - Let’s go hands-on and deploy a real-world application
- **KubePrimer final**

SIGHUP

kubernetes

Agenda

- **Kubernetes: the Architecture**
  - Master components
  - Worker components
  - State
  - Kubernetes HA
- **Kubernetes:**
  - Cloud vs On-Premise
  - Managed vs self-provisioned
- **Appendinx A: Beyond Docker containers**
- **Appendix B: Logging and monitoring on Kubernetes**
  - Fluentd + Elasticsearch + Kibana
  - Prometheus + Grafana

Before starting

Setting up the environment

Head over and follow instructions:

https://github.com/sighup-io/kubeprimer/blob/master/README.md#preparation--setup

A quick intro to Minikube

Setting up the environment

Minikube is by far the easiest tool to run locally a *single node cluster*.

Kubernetes is currently embedded in Docker for Mac/Win, still missing on Linux and still missing some network features

It gives you a bunch of options that you can tweak, use `*minikube --help*` to display them. It's basically a pre-configured VM. You can ssh into it with `*minikube ssh*`

SIGHUP

kubernetes

SOFTWARE DEVELOPMENT
AND
CONTAINERS

SIGHUP

Docker

Origins

#dotCloud

Born in the PaaS world

"A simpler way to deploy your web application in the cloud"

"One platform, any stack"

"Be a developer, not a sysadmin"

#containers-community

Struggling with LXC, chroot, cgroups, zones, etc

Enthusiastic reaction to Docker promoting the development of new docker-based tools (e.g. figs)

Open community, with developers and sysadmins collaborating in the definition of the next generation application runtime platform

#docker

Disruptive Innovation

"Open-Source engine to commoditize LXC "

"Proposing a standard format for containers"

From 1.0 to Swarm in 6 months

SIGHUP

kubernetes

Docker

Build

Together with the CLI UX, the Dockerfile is the real docker innovation

Before docker, container images were made via bash scripts and chroot

#dockerfile

```
FROM node:8.1.3

RUN apt-get -y update && \
    apt-get -y install vim wget curl

WORKDIR /kubeprimer-api/

ADD rootfs /
```

*Provides an **easy** and **universal language** **to** **build applications** together with their execution environment*

Runs everywhere, all you need is access to the Docker daemon

*Thanks to its universality, **Developers** **and** **Sysadmins** **have been finally able to** **speak the same language** to define applications infrastructure*

SIGHUP kubernetes

Docker

Ship

#docker #registry

The Maven of Docker Containers

Provides an easy and efficient ***container images distribution system***

The role of the Registry is to ***decouple*** ***the docker images builder from the actual nodes running the containers***

Overall, the Docker Registry does not provide anything different from the AWS AMI MarketPlace, OpenStack Glance, etc.

The most known Docker Registry implementation is the Docker HUB

Recently has been introduced the support to multi-architecture images (arm, ppc64, etc)

SIGHUP

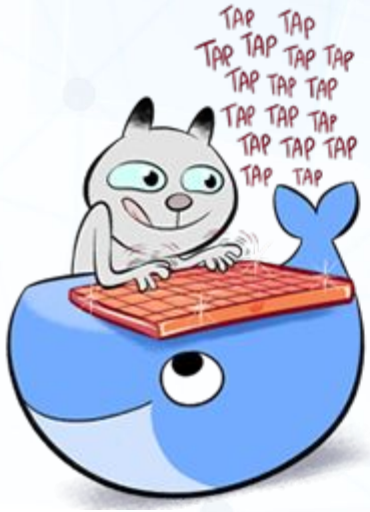
kubernetes

Docker

Run

***Initially developed as a wrapper of LXC**:*
the Docker Daemon was expanding the Docker image in a new FS branch and *forking to execute the image ENTRYPOINT containing the process leveraging LXC* and *attaching* it *to a new cGroups tree*

#docker #magics



Potentially we can run any Docker image on any host. The only requirement is having the Docker daemon up & running (and a compatible system architecture)

*Since Docker 1.11, Docker leverages containerd to interact with **OCI** **compliant container runtime engines***

*The default Docker container runtime is **runc**, potentially this could be **replaced with any OCI compliant runtime***

SIGHUP

kubernetes

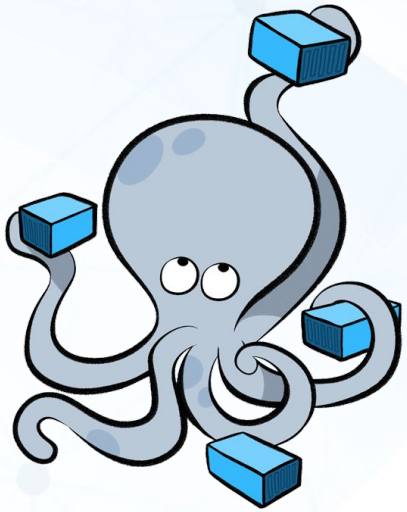
Docker

Compose

Provides the same single-container UX to ***multi-container applications***

Since Docker 1.12 provides a new command to bundle multi-container applications stacks and deploy them in production

#compose



docker-compose is a perfect Software Development companion allowing developers to run external software dependencies with docker-compose

docker-compose is a must have when developing micro-service applications

docker-compose, was initially a community project named fig

docker-compose allows to ***version control the whole services stack*** *(including network and storage)*

SIGHUP

kubernetes

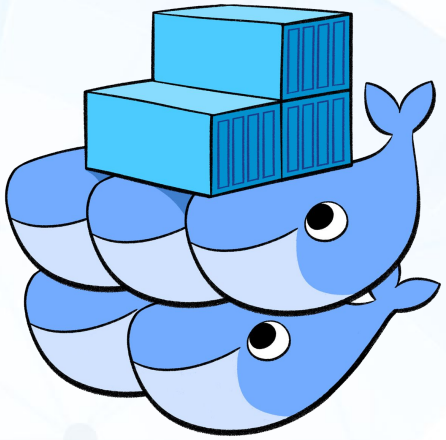
Docker

Swarm

Real production systems can't run on single-node. Docker Swarm provides an ***auto-piloted multi-node Docker environment***

Before "Swarm Mode", Swarm was an external component. Now is ***shipped with the Docker Engine binary***

#swarm



With "Swarm Mode", Docker provides ***native embedded support to Services Discovery and Network Load Balancing***

The cluster creation and scheduling experience with Swarm is amazing

Docker Swarm leverage ***VXLAN*** *to implement multi-host overlay networks*

Swarm mission is to implement ***clustering for common people***

SIGHUP

kubernetes

Docker

Challenges

#support

Being a continuously evolving Open-Source project, ***older Docker releases became unsupported after few months***

The new Docker release cycle aims to target this issue

#security

Docker decided to embed many additional features into the standard docker distribution (e.g. Swarm)

This might **increase the attack surface** for the project

#community

The decision of embedding Swarm into the Docker Engine with "Swarm Mode" had as effect the split of the community between Supporters and Not

This might move many important contributors away from the project

#fragmentation

The recent Moby announcement is surely trying to address some critics Docker received about it's Open-Source Management Policies

The ***risk of Moby is the ecosystem fragmentation***

SIGHUP

kubernetes

Kubernetes

Principles

Open-sourced by Google on June 2014. Donated to Cloud Native Computing Foundation (CNCF) in 2015.

Since the CNCF is a Linux Foundation initiative, ***K8S is part of the Linux Foundation*** *(like the Linux Kernel, XEN, etc)*

#borg #omega #kubernetes

The ***K8S initial design is inspired by Google's experience managing large scale compute clusters with Borg and Omega***

In 2.5 years, Kubernetes has seen plus than 1500 individual contributors from 15 different timezones

"Kubernetes aims to eliminate the burden of orchestrating physical/virtual compute, network, and storage infrastructure, and enable application operators and developers to focus entirely on container-centric primitives for self-service operation."

https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture.md

SIGHUP

kubernetes

Kubernetes

vs Swarm

#modularity

By design, K8S focus on scheduling and monitoring of resources

Storage and Network management is demanded to external software components

Swarm ships some of these features natively (e.g. Overlay Networks)

#focus

Thanks to the Kubernetes modular design, the K8S developers can focus on ***scheduling, security and performances***

The integration of external components is done via well defined extension points (CRI, CNI, CRD)

#multi-tenancy

Thanks to namespaces and the introduction of Role Based Access Control (RBAC), ***Kubernetes is the perfect orchestration platform for multi-operator and multi-tenant workloads***

#pods

Kubernetes pods ease data sharing and scheduling of containers that must leave together without having to build monolithic or side-car containers

SIGHUP

kubernetes

Kubernetes

for Docker users

#kompose

Kompose it's a conversion tool from docker-compose to Kubernetes *(or OpenShift)*

Kompose ***ease the transitions to docker-compose based development environments to production on K8S***

```
± % kompose convert -f docker-compose.yml
WARN Unsupported root level networks key - ignoring
WARN Unsupported root level volumes key - ignoring
WARN Unsupported networks key - ignoring
WARN Kubernetes provider doesn't support build key - ignoring
INFO file "backend-service.yaml" created
INFO file "mongo-service.yaml" created
INFO file "web-service.yaml" created
INFO file "backend-deployment.yaml" created
INFO file "mongo-deployment.yaml" created
INFO file "kubeprimer-db-persistentvolumeclaim.yaml" created
INFO file "web-deployment.yaml" created
```

Overall Kompose is the best option for Docker users to get started with Kubernetes

Kompose does not provide any equivalent to docker-compose build. The docker-images must be built in a CI/CD pipeline and pushed to the Docker registry

SIGHUP kubernetes

KUBERNETES
PRINCIPLES AND **CONCEPTS**

SIGHUP

Kubelet

Core concepts

Main **agent running on each node**.
Does a few things:

- Ensuring that the node is doing its job
- Ensuring state and health of running containers on the node
- Enforce limits and options when containers are scheduled on the node

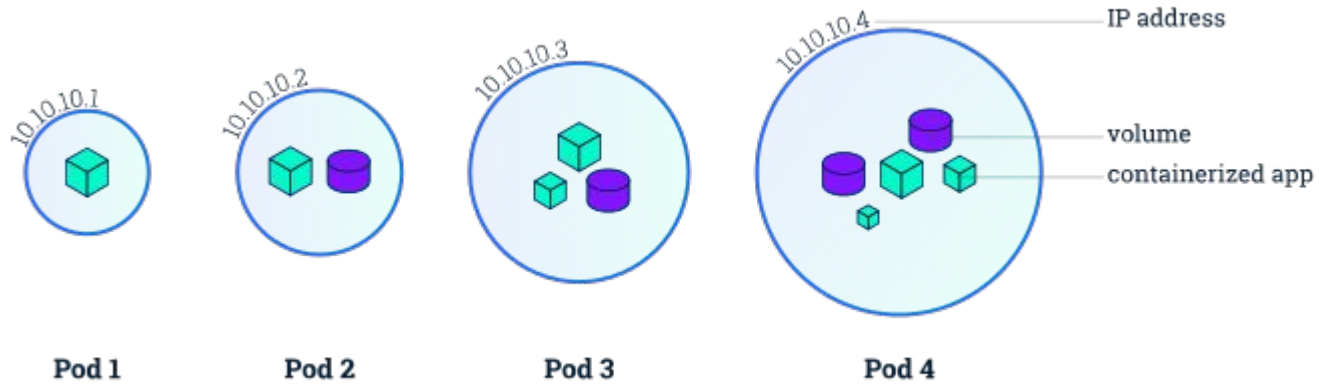
Kubelet are running on each node of the cluster, both on masters and workers.

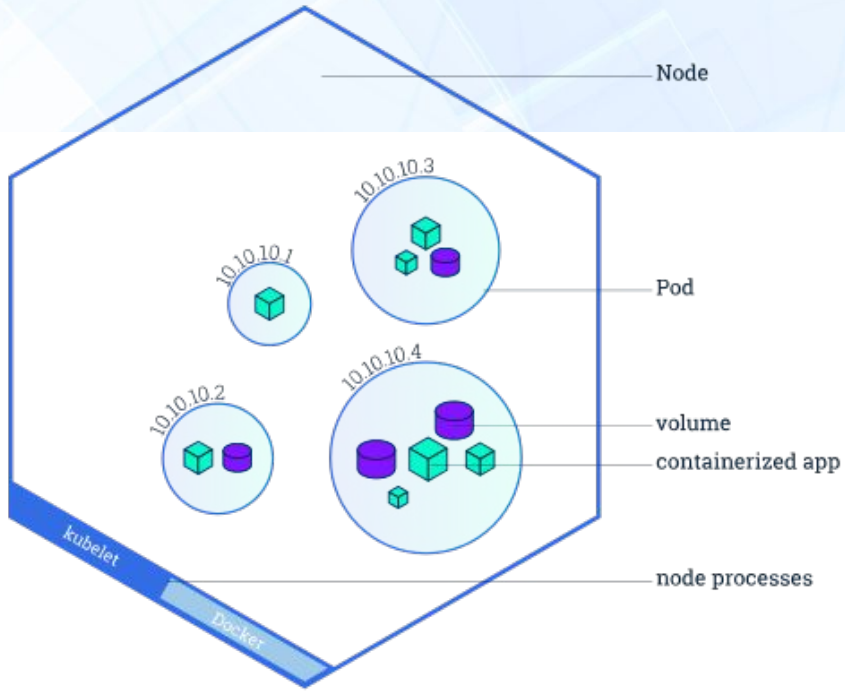
Pods

Core concepts

Pods are ***cohesive collection of containers*** who will **share Networking and Storage**.

- Pods are always **co-located** and **co-scheduled**, they ***run in a shared context***.
- Each Pod has a **virtual ip** associated managed by kubernetes
- Containers within a pod **share an IP address and port space**, and can find each other via localhost

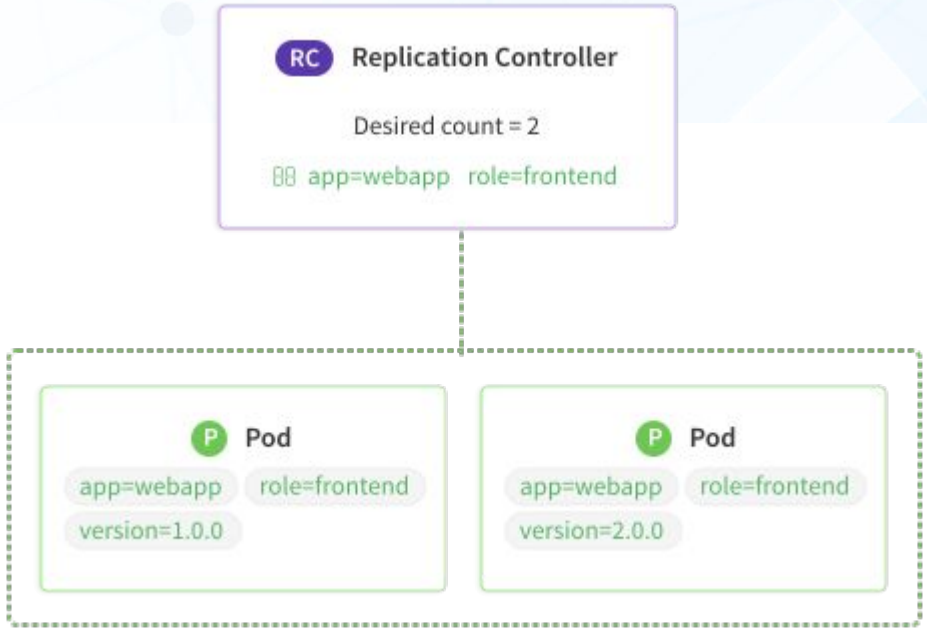




Replication Controllers / ReplicaSets

Core concepts

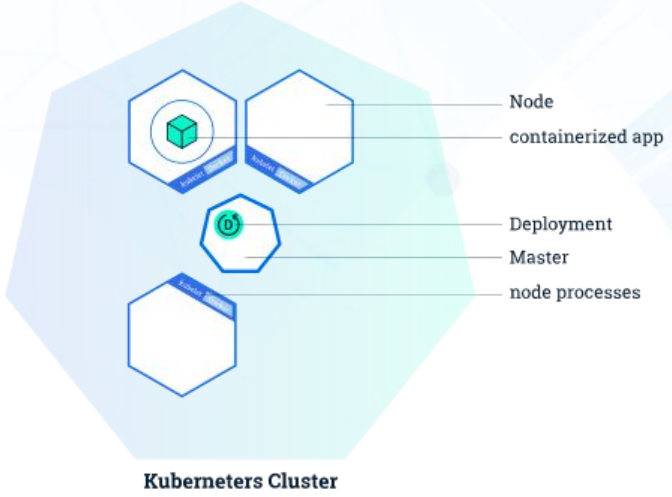
A ***ReplicationController*** **ensures that a specified number of pod "replicas" are running at any time**. **ReplicaSets** are *used by Deployments*, supporting **labels-based selectors**.



Deployments

Core concepts

A Deployment is an *abstraction of Pods and ReplicaSets*, provides **rolling updates and rollbacks**.



SIGHUP

kubernetes

Deployments in action

https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/deployments-in-action.md

SIGHUP

kubernetes

All fun and games but...

- *How do I access my nginx? You didn't show us.*

- *You are tricking us with a trivial example, reality is more complex!* ***I've got micro-services to handle****.*

- *How do my micro-services interact with each others? how do I expose them? How do I serve traffic?*

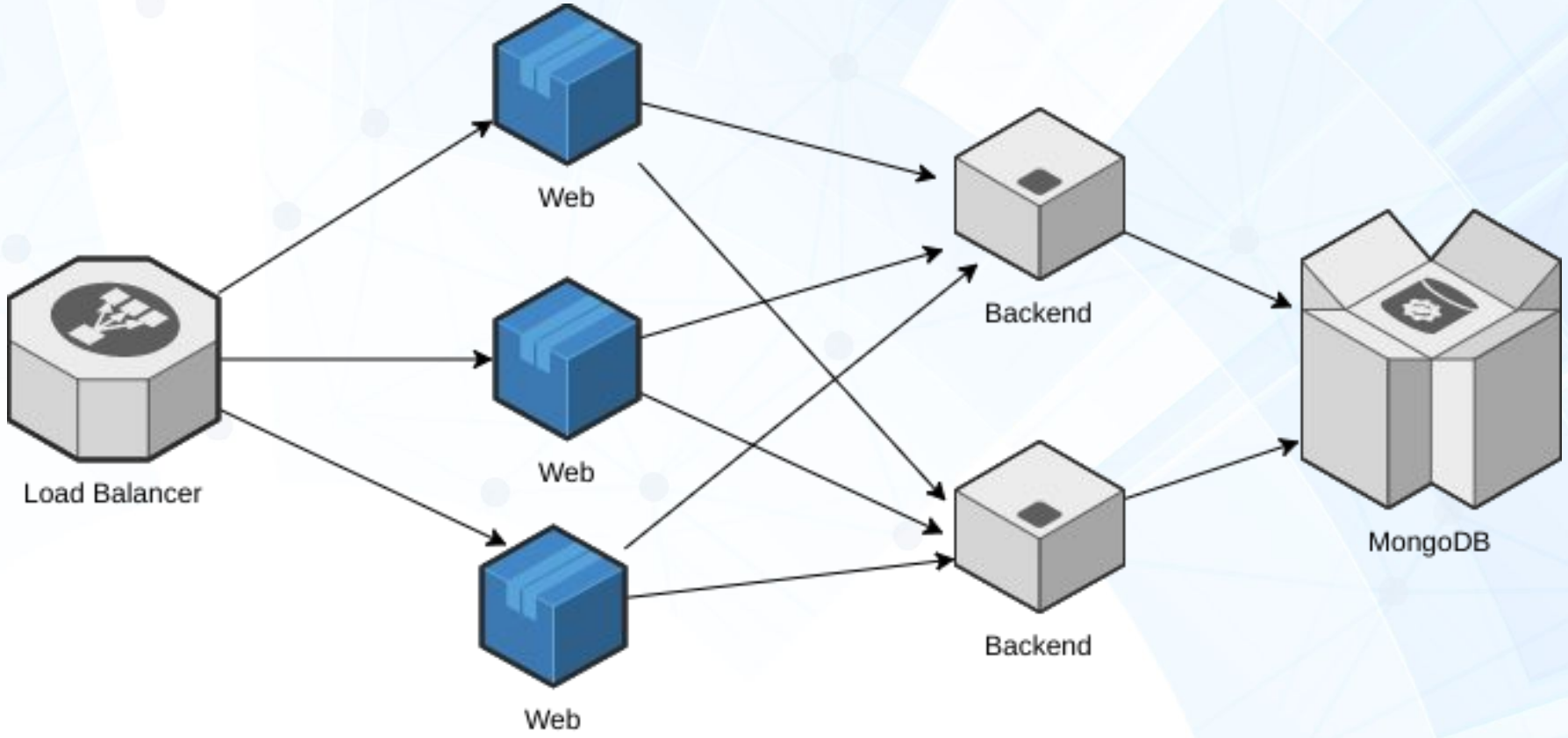
- *How do I handle configuration and passwords and everything?*

SIGHUP

kubernetes

Micro-services Architecture

Hands-on



SIGHUP

kubernetes

Handle configuration: ConfigMaps & Secrets

Conceptually, *ConfigMaps* and *Secrets* are *simple key=value data stored* for configuration and secrets management. The only difference between the two, at the moment, is simply that Secrets are stored encoded base64. Over time they will differentiate more.

Keys set in ConfigMaps and Secrets can be used in pods both as **environment variables** and files if **mounted as external volumes.**

The core idea is that **configuration should be decoupled from pod definition**, **secrets management should give a mechanism to not hardcode secrets in multiple places** (all pods definitions)

Let’s create our ConfigMaps/Secrets

https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/configmaps/README.md

https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/secrets/README.md

SIGHUP

kubernetes

Services

Pods are mortal, they are constantly created/destroyed dynamically by RS. Their virtual IPs are ephemeral.

Services are the abstraction that identifies a set of pods, how to make them accessible beyond single pod lifecycle.

Services have **associated a persistent Virtual IP**, they both support TCP and UDP

Services: exposing pods

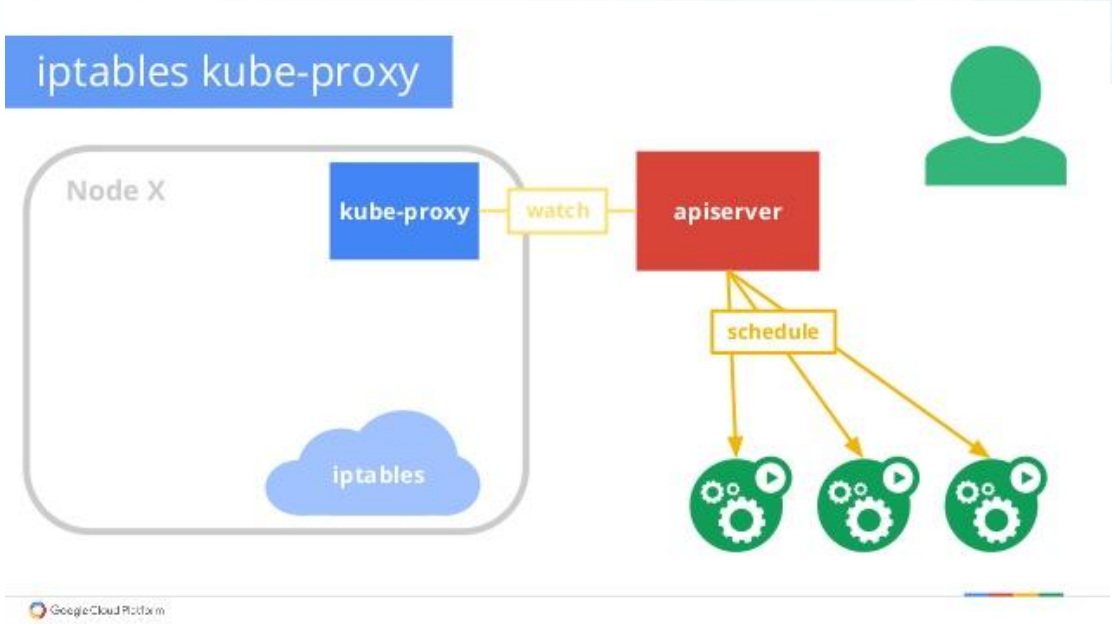
By default pods are not reachable from the outside world.

Opening Pods ports won't actually make them available

We need a service to match the pod, it's the Service that will **expose the pod within the cluster** or **to the outside world** depending on Service type

Services & Networking

Each node has a **kube-proxy** service running. Kube-proxy job is to **keep the kubernetes services virtual network consistent**. It implements routing translating virtual IPs into physical IPs by dynamically changing **iptable** rules on the nodes.



Service Discovery

Service discovery happens either via **Environment Variables** or via **DNS.**

ENVIRONMENT VARIABLES
When a pod is scheduled on a node, the kubelet automatically adds environment variables for each running service into the pod.

DNS
Kube-dns is a cluster addon (present in most of the cluster implementation and strongly recommended) that watches for services and creates a set of records related to it.

This is useful as pods can access services in their own namespace referring to them by name.

Service Types (1)

There are different types of services with different characteristics:

ClusterIP

Exposes services with an internal virtual IP, this makes them reachable only within the cluster itself. This is the default.

NodePort

Expose service on each node IP on a given port. This exposes your services to the outside world and you can reach them <node_ip>:<port>. A ClusterIP service to which the NodePort service will route is automatically created

Service Types (2)

Ingress

This isn't really a service, **Ingresses are themselves first class citizens** and won't work out of the box as they require their own component running on the master. They implement **routing towards services via virtualhost**, to be more specific `myapp.com` can point to our cluster. If it has an ingress associated, this will automatically route the request toward the correct service.

LoadBalancer

LoadBalancers are **available only on cloud infrastructures** who supports them (AWS, GCE to mention a few). Kubernetes will automatically create an external load balancers and will attach it to our nodes routing traffic towards our service and related pods.

DEPLOYING A MULTI-TIER APPLICATION

SIGHUP

Deploying our multi-tier application and exposing pods as services

https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/multi-tier-application.md

SIGHUP

kubernetes

Other types of pods

DaemonSet

DaemonSets are abstractions that make sure that **each node execute a specific type of pod** at any time.

They can either embody a pod definition or they can be attached to an existing pod definition using labels and selectors.

An example of DaemonSet:
https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/monitoring/node-exporter.yml

Jobs & CronJobs

Normal pods **must be alive**, if the process exits kubernetes will try to reschedule the pod indefinitely (or according to your policies). Yet, there are cases where you actually want that a pods process ends, ie: **batch jobs** where you process some data everyonce in a while.

Jobs are special kind of pods specifically designed to execute batch jobs. A Job will make sure that **a job completed successfully, aka the pod will exit successfully**.

Jobs in action

https://github.com/sighup-io/kubeprimer/blob/master/kubernetes/playing-with-jobs.md

SIGHUP

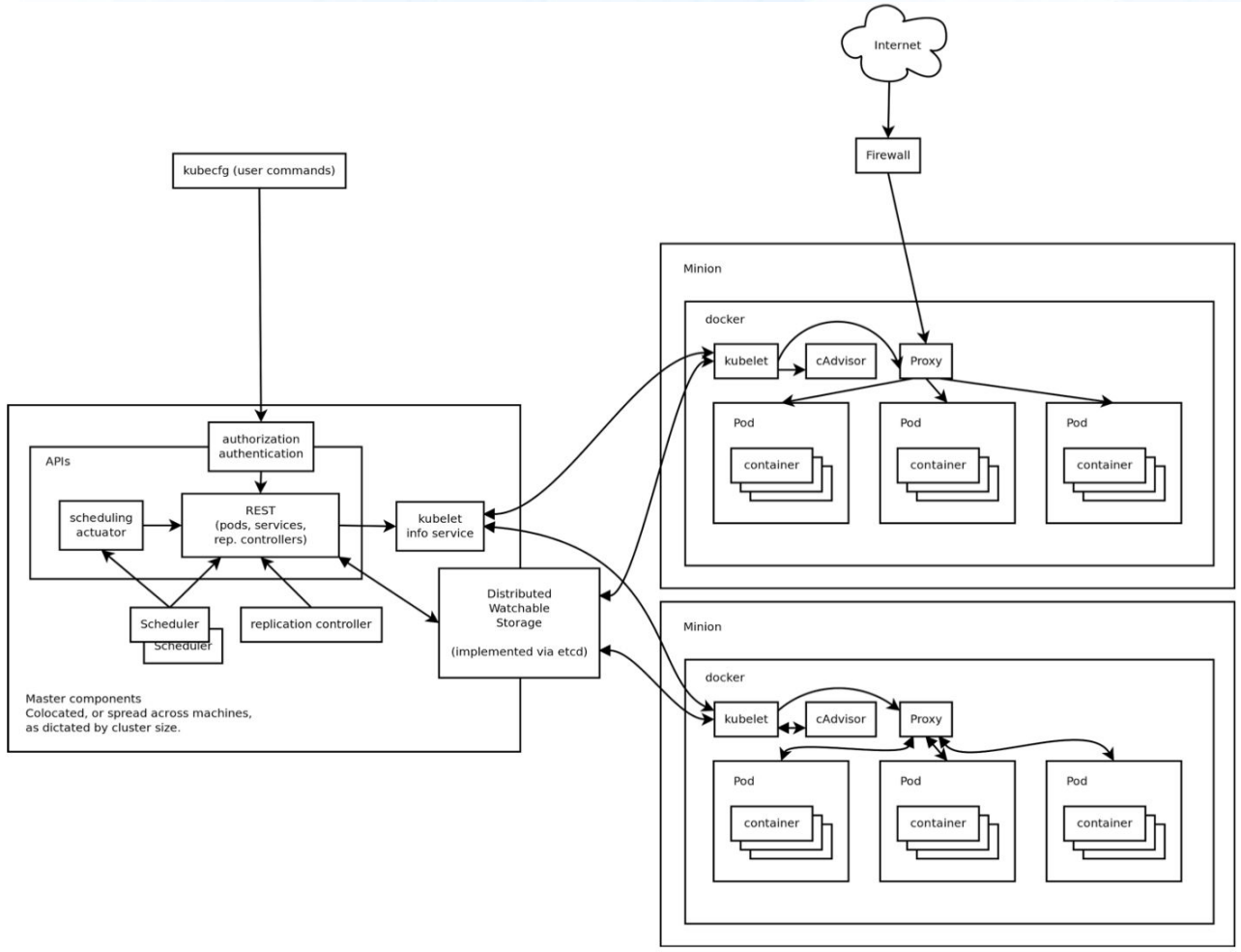
kubernetes

THE KUBERNETES ARCHITECTURE

SIGHUP

Kubernetes Architecture

System Design



https://speakerdeck.com/thockin/architecture-of-kubernetes

SIGHUP

kubernetes

Kubernetes

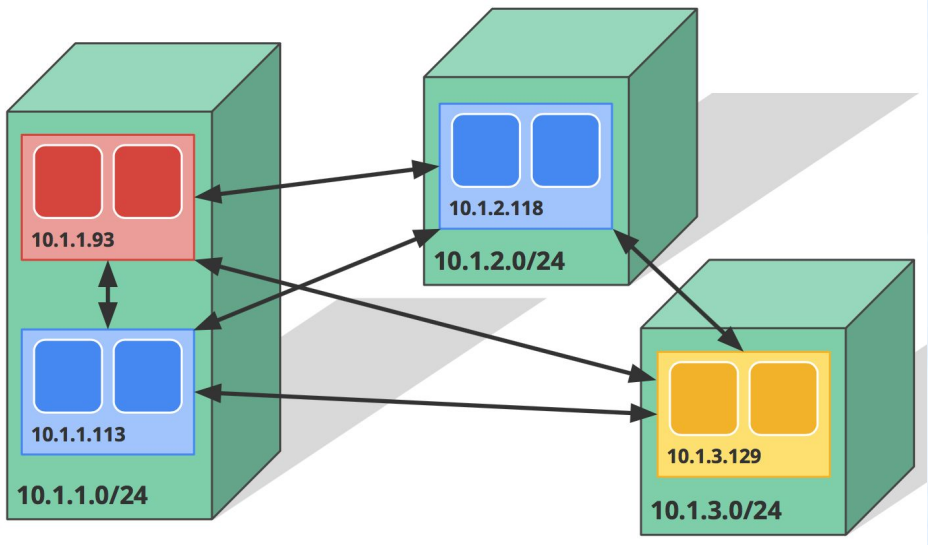
Design Principles

- **Declarative > imperative**
  - State your desired results, let the system actuate
- **Control loops**
  - Observe, rectify, repeat
- **Simple > Complex**
  - Try to do as little as possible
- **Modularity**
  - Components, interfaces, & plugins
- **Network-centric**
  - IP addresses are cheap
  - POD level networking

Kubernetes

Pod Networking

- **Pod IPs are routable**
  - Docker default is private IP
- **Pods can reach each other without NAT**
  - Even across nodes
- **Pods can egress traffic**
  - if allowed by cloud environment
- **No brokering of port numbers**



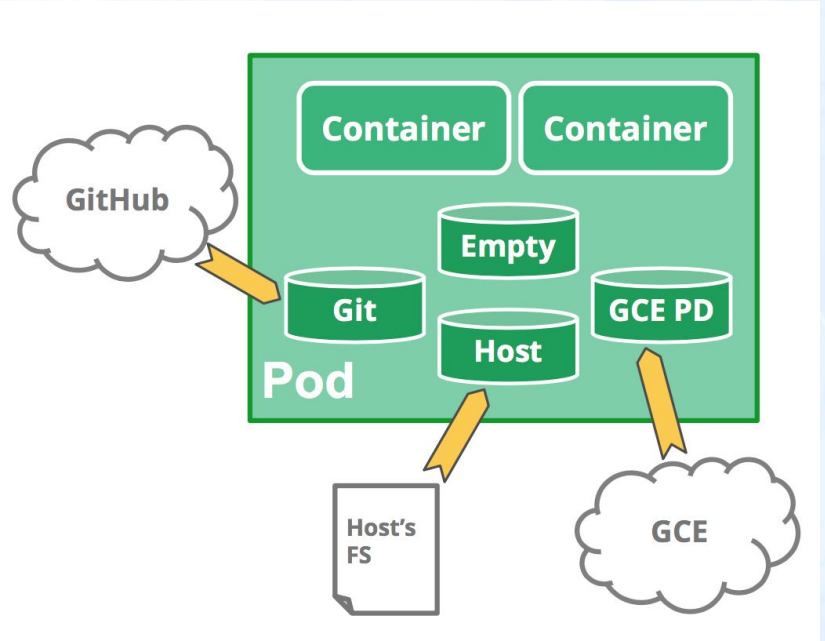
SIGHUP

kubernetes

Kubernetes

Volumes vs PersistentVolumes

- **Pod Scoped**
  - Multiple containers in the same Pod shares storage
- **Volumes**
  - Share Pod's lifetime & fate
  - "Ephemeral"
- **PersistentVolumes**
  - Persist to Pod's lifetime & fate
  - "Persistent"
  - Different Access Policies
    - ReadWriteOnce – the volume can be mounted as read-write by a single node
    - ReadOnlyMany – the volume can be mounted read-only by many nodes
    - ReadWriteMany – the volume can be mounted as read-write by many nodes

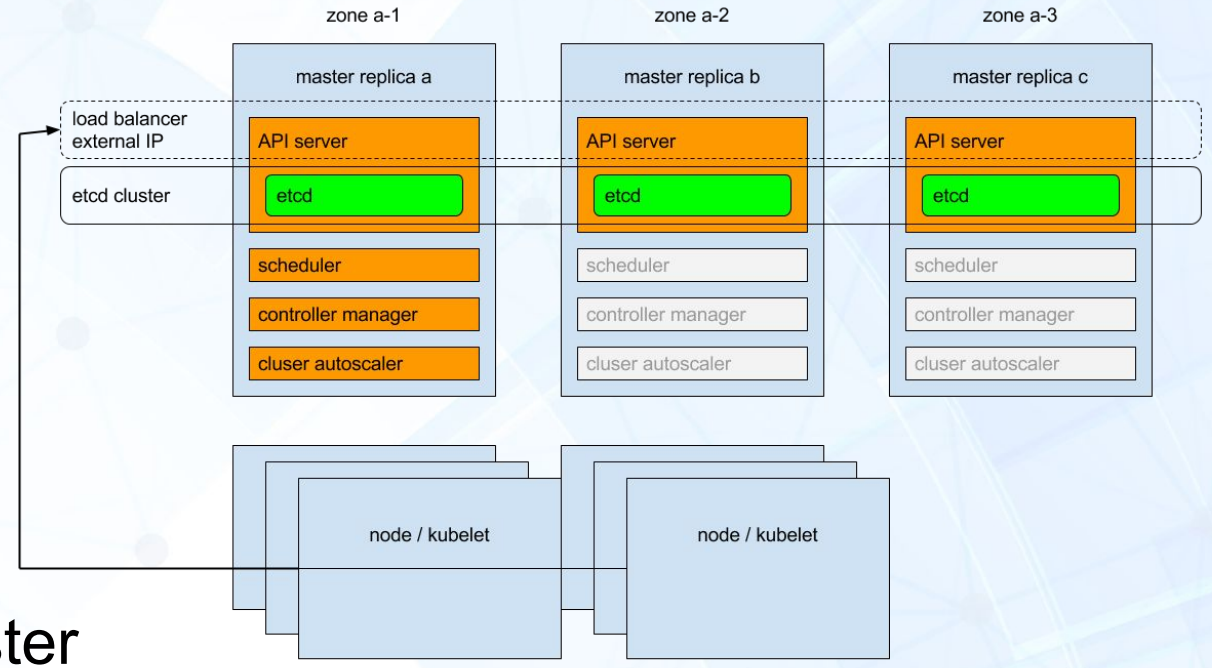


SIGHUP kubernetes

Kubernetes

Highly available deployment

- **API Server**
  - External Load Balancing
- **ETCD**
  - Native “Leader-Follower” Clustering
- **Controllers and Schedulers**
  - will use lease mechanism
  - only one instance of each of them will be active in the cluster



SIGHUP

kubernetes

KUBERNETES
IN
PRODUCTION

SIGHUP

Kubernetes in Production

Managed / as a Service

Huawei, RedHat OpenShift, GCP, Platform 9, Weaveworks, Microsoft Azure, Apprenda, Coreos

Kubernetes in Production

Do It Yourself

#deployment

$ manually

https://kubernetes.io/docs/getting-started-guides/scratch

$ kube up

https://github.com/kubernetes/kubernetes/blob/master/cluster/kube-up.sh

$ kops

https://github.com/kubernetes/kops

$ kube-spray

https://github.com/kubernetes-incubator/kubespray

$ conjure up

https://conjure-up.io/

#networking

$ with overlay

An overlay network obscures the underlying network architecture from the pod network through traffic encapsulation (e.g. vxlan)

$ without overlay

Configure the underlying network fabric (switches, routers, etc.) to be aware of pod IP addresses.
This does not require the encapsulation provided by an overlay, and so can achieve better performance

#storage

PersistentVolumeClaims require to have a Clustered Persistent Storage backend

While on the Cloud PersistenStorage is something provided out-of-the-box, on-premise you might want to evaluate NFS, Ceph or SAN Appliances

#upgrades

How to upgrade a Kubernetes cluster is still an open topic

Some deployment and/or cloud solutions provide out-of-the-box upgrade solutions, however upgrading the Control Plane of a K8S on-premise is still painful

Suggestion?! Double check etcD! Do rolling update of the control plane

SIGHUP

kubernetes

kube up

The “out-of-the-box” tools

The old way of getting started with K8S

Perfect for quick-starts on the cloud (e.g. AWS)

Very limited functionalities

Black-box

“Almost deprecated” (see issue #49213)

kubeadm

The Kube answer to docker-swarm Cluster Management User Experience

Works everywhere!

Sill requires lot of the Cluster design decisions to be taken in advance

Lifecycle management still in alpha

SIGHUP

kubernetes

kops (kubernetes/kops)

The “official” tools

“Production Grade K8s Installation, Upgrades, and Management”

So far, officially supported only on AWS

Limited to the Cloud

Allows to generate configuration files in CloudFormation and/or Terraform format

Deploys Highly Available (HA) Kubernetes Masters

kubespray (kubernetes-contrib/kubespray)

Written on top of Ansible!!!

Runs everywhere, also on bare-metal

Deploys Highly Available (HA) Kubernetes Masters

Still a kubernetes-incubator project

SIGHUP

kubernetes

kubicorn (kris-nova/kubicorn)

"Third-party" community tools

"kubicorn is an unofficial project that solves the Kubernetes infrastructure problem and gives users a rich golang library to work with infrastructure."

It's a kind of library to code your own Kubernetes cluster

Deeply relies on kubeadm to bootstrap the cluster

conjure-up (conjure-up/spells)

"Kubernetes for the enterprise"

Built on top of Juju, Canonical's configuration-management / deployment orchestration tool

Easy configuration and spin-up of k8s via interactive configuration menus (ncourses)

Runs everywhere, on Ubuntu

SIGHUP

kubernetes

Finals

SIGHUP

Appendix A

SIGHUP

BEYOND DOCKER CONTAINERS

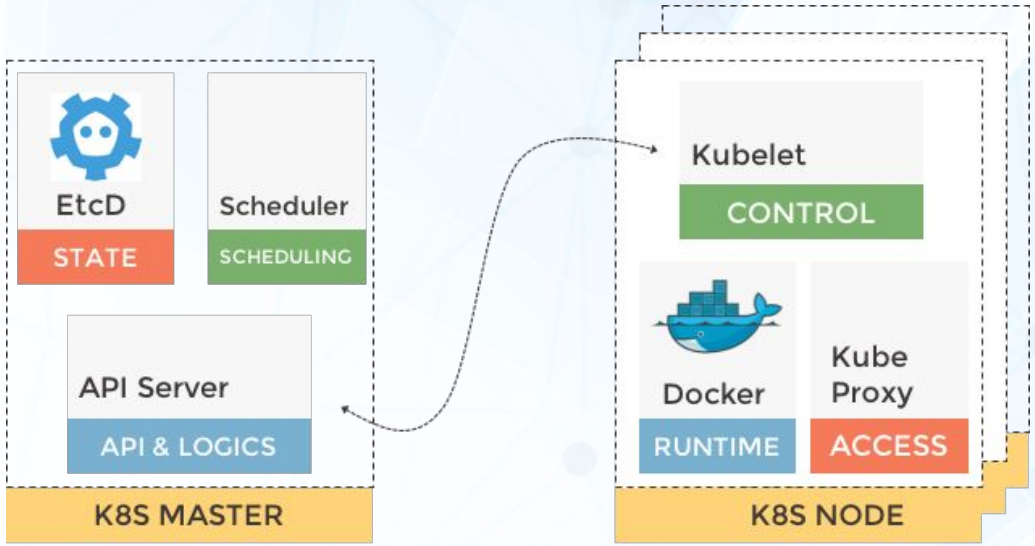
SIGHUP

Kubernetes - Control Plane

Every change of status in the Kubernetes cluster is recorded in EtcD

EtcD is the cluster unique source of truth

Leveraging EtcD Watchers the proper components are triggered every status change and demanded to perform their job to actuate / react to it



Overall the Kubelet performs two main tasks:
- takes care of containers execution
- periodically execute containers, pods and nodes liveness probes and reports their results to the API Server

The communication between the API Server and the Kubelet is bidirectional: the API Server instructs the Kubelet to perform actions, the Kubelet continuously report informations back to the API Server

SIGHUP kubernetes

Kubernetes

Custom Resource Definitions

#crd

When you create a new CustomResourceDefinition (CRD), the Kubernetes API Server reacts by creating a new RESTful resource path

To handle CRDs you have to write your own Controller

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  # name must match the spec fields below, and be in the form: <plural>.<group>
  name: crontabs.stable.example.com
spec:
  # group name to use for REST API: /apis/<group>/<version>
  group: stable.example.com
  # version name to use for REST API: /apis/<group>/<version>
  version: v1
  # either Namespaced or Cluster
  scope: Namespaced
  names:
    # plural name to be used in the URL: /apis/<group>/<version>/<plural>
    plural: crontabs
    # singular name to be used as an alias on the CLI and for display
    singular: crontab
    # kind is normally the CamelCased singular type. Your resource manifests use this.
    kind: CronTab
    # shortNames allow shorter string to match your resource on the CLI
    shortNames:
    - ct
```

```yaml
apiVersion: "stable.example.com/v1"
kind: CronTab
metadata:
  name: my-new-cron-object
spec:
  cronSpec: "* * * * /5"
  image: my-awesome-cron-image
```

After the CustomResourceDefinition object has been created, you can create custom objects. Custom objects can contain custom fields

A usage example of TPR (now deprecates by CRD) are CoreOS Operators

SIGHUP

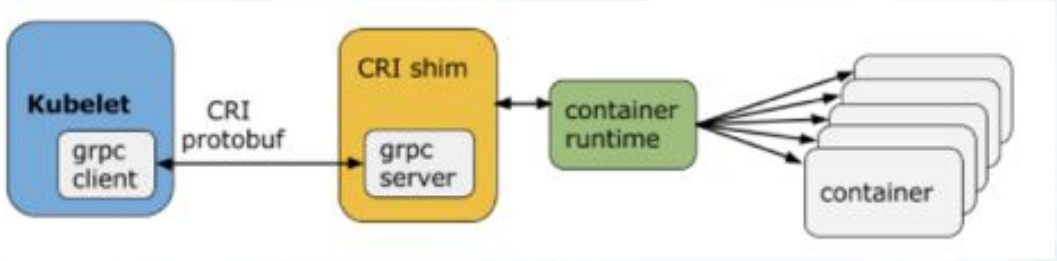
kubernetes

Kubernetes

Custom Runtime Interface

#cri

The Kubelet communicates with the container runtime (or a CRI shim for the runtime) over Unix sockets using the gRPC framework, where kubelet acts as a client and the CRI shim as the server.



The protocol buffers API includes two gRPC services, ImageService, and RuntimeService. The ImageService provides RPCs to manage images operations. The RuntimeService contains RPCs to manage the lifecycle of the pods and containers, as well as calls to interact with containers (exec/attach/port-forward).

Supporting interchangeable container runtimes is not a new concept in Kubernetes. Since the 1.3 release, in supports rktnetes project to enable rkt container engine as an alternative to the Docker container runtime.

SIGHUP

kubernetes

Kubernetes - Beyond Containers

#virtlet

Virtlet is a Kubernetes runtime server developed by Mirantis which allows you to run VM workloads, based on QCOW2 images.
Virtlet consists of the following components:
- Virtlet manager, implementing CRI interface for virtualization and image handling,
- vmwrapper, which is responsible for preparing environment for emulator, currently qemu,
- CRI Proxy, which provides the possibility to mix docker-shim and VM based workloads on the same k8s node

https://github.com/Mirantis/virtlet

#kubevirt

KubeVirt extends Kubernetes by adding additional virtualization resource types through Kubernetes's third party resource concept.
KubeVirt consists in:
virt-api, the HTTP RESTfull entrypoint to manage the virtual machines within the cluster,
virt-controller, the component managing the state of each VM within the Kubernetes cluster,
virt-handler, the daemon running on each Kubernetes node providing an interface to libvirtd.

https://github.com/kubevirt/kubevirt

SIGHUP

kubernetes

Appendix B

SIGHUP

LOGGING & MONITORING

SIGHUP

Monitoring: Prometheus

Prometheus is a monitoring tool that can support natively **kubernetes** offering out of the box service discovery and basic cluster metrics gathering. Prometheus is actually composed by different pieces that offer different functionalities:

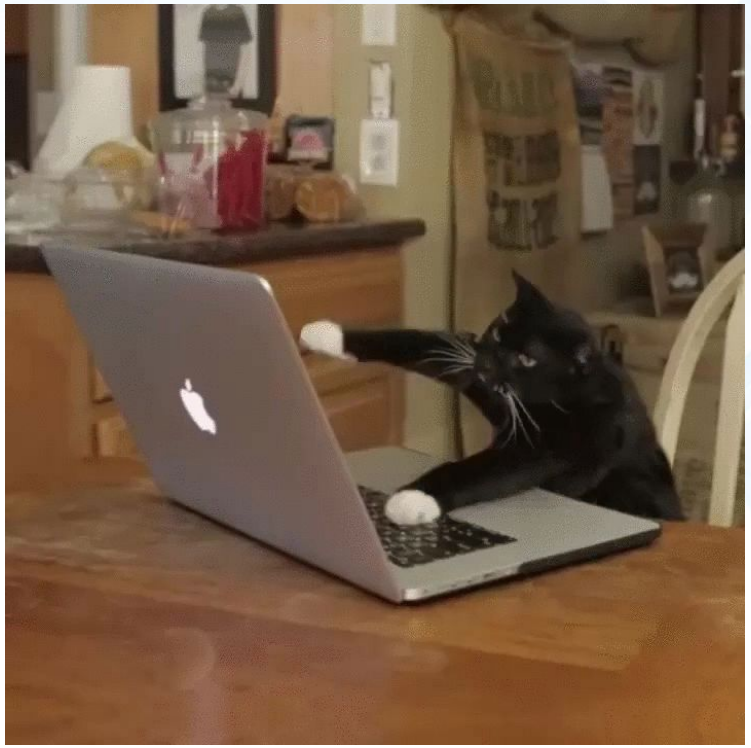
- Prometheus Time series database
- AlertManager
- PushGateway
- Native integration with Grafana

It works with a **pull model**.

SIGHUP

kubernetes

Let’s deploy monitoring and logging to our multi-tier application



SIGHUP

kubernetes

FOLLOW-UP

AND

FEEDBACK

SIGHUP

ARRIVEDERCI

http://sighup.io

Via Feltre 28/6 – 20132 – Milano - Email: jacopo@sighup.io