Kubernetes Essential

Practical Guide to Learn Kubernetes

J°t Edition

CONTENTS

e two technologies play a key role in shifting to DevOps methodologies and CI/CD (continuous integration/c

2. KEY DEFINITIONS AND CONCEPTS

WHAT IS KUBERNETES?

In order to fully understand Kubernetes and its significance in the Information Technology World, a brief look at recent history will be quite beneficial as you will find out.

Virtualization has come a long way in Information Technology which began when there was need to share the resources of a computer among many users. As it is known, Computer resources can be pretty expensive and hence there is a need to utilize whatever you have to the optimum instead of investing in another expensive venture. In the 1960's and early 1970's IBM embarked on a journey with the objective of finding ways that will make it possible to share computer resources in a robust fashion. The breakthrough was the concept of virtualization that made computing capability costs to plunge remarkably in such proportions that it made organizations and individual entities to use computer resources devoid of owning one. Virtualization has made it possible to improve the utilization of resources and more importantly a reduction in costs.

With the constant development of technology, virtualization has not been left behind in the growth cycle. With more innovative solutions on the rise, containerization in the field of technology is the current standard that is improving efficiency and resource utilization.

WHAT THEREFORE IS CONTAINERIZATION

When cargo is being shipped from one country to another across the ocean, they are normally placed in different containers for easy

management. Instead of piling up every product in one huge container, shoes and clothing for instance are placed in different containers without

either interfering with each other. The same is applied in computing Containerization. Creating a container is basically putting everything you need for your application to work be it libraries, operating system or any other technology. What has been created can be replicated and will work in any environment which saves time and makes it easy for
other processes to continue without re-installing the same components of the container every time you spin a virtual machine. It is a type of a strategy in virtualization that has come about as an alternative to the native or initial hypervisor-based virtualization. Containerization involves creating separate containers at the operating system level which makes it possible to share libraries, file systems and other important components hence saving a lot of space compared to native virtualization where
each virtual machine had its own components in isolation. There are few containerization technologies that offer containerization tools and API such as Docker Engine, Rkt, LXC, OpenVZ, runC and LXD. After understanding the key concepts, Kubernetes can thus be easily defined. Below are few similarities and differences between these container technologies.
Kubernetes is an active open source project founded by Google to assist system developers/administrators orchestrate and manage containers in different kind of environments such as virtual, physical, and cloud infrastructure. Currently, Kubernetes project is hosted by Cloud Native Computing Foundation (CNCF).

WHAT IS A DOCKER CONTAINER?

A docker container is a lightweight software package that includes everything needed to run it, including its own minimal operating system, run-time resources, and dependencies. Docker ecosystem lies at the heart of the mass adoption and the excitement witnessed in the container space. To spin a Specific Docker container, they are developed out of images designed to provide a specific capability, for instance a database such as MariaDB, a base operating system or even a web server such
as Apache. These images of Docker are made from file systems that are layered so that they are capable of sharing common files. Sharing of common files adds the advantage of reducing the usage of disk space and speeding up image download

As compared to virtual machines, containers are more resource-

efficient because they do not require hypervisors. In addition, containers have less memory footprint and can help organizations avoid high costs and hassles associated with server sprawl.

HOW KUBERNETES DIFFERS FROM DOCKER PROJECT?

Docker project aims at defining a container format, building and managing individual containers

WHAT IS ORCHESTRATION?

In order to implement certain applications, many containers need to be spinned and managed. In order to optimize this process, the deployment of these containers can be automated. This is especially beneficial if there is a growth in the number of hosts. This automation process is called orchestration.

FEATURES OF ORCHESTRATION

Preparing and equipping hosts

Instantiating a set of desired

containers

Maintaining failed containers for example through rescheduling them

Merging containers together through interfaces that have been agreed upon

Exposing services to machines outside of the cluster

Docker has several orchestration tolls such as Kubernetes, Docker Machine and Docker swam among others. Kubernetes is one of the most feature- rich orchestration tools and is widely used.

After building the container image you want with Docker, you can use Kubernetes or others to automate deployment on one or more compute nodes in the cluster. In Kubernetes, interconnections between a set of containers are managed by defining Kubernetes services. As demand for individual containers increases or decreases, Kubernetes can start more or stop some container pods as needed using its feature called replication controller.

Kubernetes gives you a capability to easily add new features to your application, manage system resources and ship your applications from development to production effortlessly. It has a flexible plugin-architecture and provides a convenient pathway to a hybrid cloud

implementation.
To conclude this section, many organizations favor the Kubernetes

framework because it is highly portable and provides a smooth migration path for legacy applications. Although containers will never be and are not designed to be the single solution to all enterprise workloads, they are a smart way to accelerate development, deployment, and scaling of cloud- native workloads with the help of tools like Kubernetes.

KEY FEATURES OF KUBERNETES

- Extensibility

This is the ability of a tool to allow an extension of its capacity/capabilities without serious infrastructure changes. Users can freely extend and add services. This means users can easily add their own features such as security updates, conduct server hardening or other custom features.

- Portability

In its broadest sense, this means, the ability of an application to be moved from one machine to the other. This means package can run anywhere. Additionally, you could be running your application on google cloud computer and later along the way get interested in using IBM watson services or you use a cluster of raspberry PI in your backyard. The application-centric nature of Kubernetes allows you to package your app once and enjoy seamless migration from one platform to the other.

- Self-healing

Kubernetes offers application resilience through operations it initiates such as auto start, useful when an app crash, auto-replication of containers and scales automatically depending on traffic. Through service discovery, Kubernetes can learn the health of application process by evaluating the main process and exit codes among others. Kubernetes healing property allows it to respond effectively.

- Load balancing

Kubernetes optimizes the tasks on demand by making them available and avoids undue strain on the resources. In the context of Kubernetes, we have two types of Load balancers – Internal and external load balancer.
The creation of a load balancer is asynchronous process,

information about provisioned load balancer is published in the Service’s status. loadBalancer.

Traffic coming from the external load balancer is directed at the backend pods. In most cases, external load balancer is created with user-specified load balancer IP address. If no IP address is specified, an ephemeral IP will be assigned to the load balancer.

- Automated deployment and even replication of containers

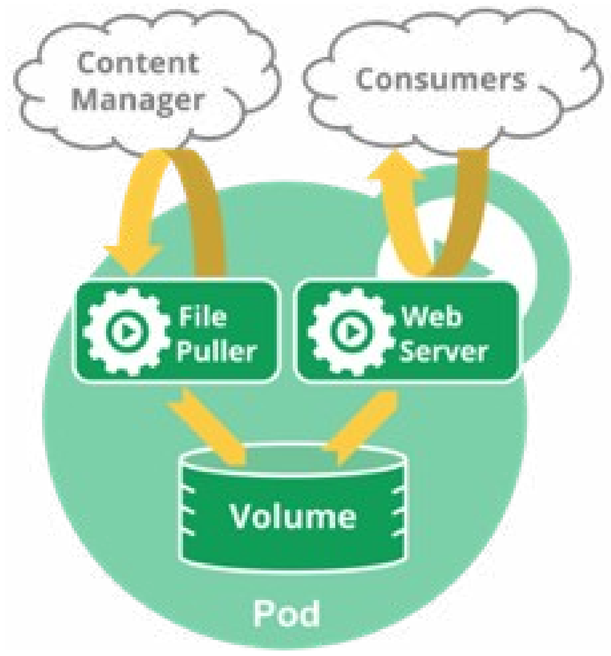
WORK UNITS OF KUBERNETES/

- Cluster

These are the nodes or the collection of virtual machines or bare- metal servers which provide the resources that Kubernetes uses to run applications.

- Pods

Pods are the smallest units of Kubernetes. A pod can be a single or a group of containers that work together. Generally, pods are relatively tightly coupled. A canonical example is pulling and serving some files as shown in the picture below.



It doesn’t make sense to pull the files if you’re not serving them and it doesn’t make sense to serve them if you haven’t pulled them.

Application containers in a pod are in an isolated environment with resource constraints. They all share network space, volumes, cgroups and Linux namespaces. All containers within a pod share an IP address and port space, hence they can find each other via 127.0.0.1 (localhost). They can as well communicate with each other via standard inter-process communications, e.g. SystemV semaphores/POSIX shared memory. Since they are co-located, they are always scheduled together.

When pods are being created, they are assigned a unique ID (UID), and scheduled to run on nodes until they are terminated or deleted. If a node dies, pods that were scheduled to that node are deleted after a timeout period.

- Labels

These are key/value pairs attached to objects like pods. When containers need to be managed as a group, they are given tags called labels.
This can allow them to be exposed to the outside to offer services. A replication controller defined next gives the same label to all containers developed from its templates. Labels make it easy for administration and management of services.

Labels are attached to objects at creation time and can be modified at any time. Each set of key/value must be unique for a given object. Unlike names and UIDs, labels do not provide uniqueness, hence many objects can carry the same label(s).

The client or user identifies a set of objects using a label selector. The label selector can be defined as the core grouping primitive in Kubernetes.

Note: Within a namespace, there should be no overlap of the label collectors that belong to two controllers.

- Services

A service is an abstraction that defines a logical set of pods and access policy. Services include load balancers services for other containers. Pods performing a similar function are grouped together

and represent one entity. If a certain process or application needs a service, a single access point grants it a scalable backend which can be easily replicated making

it optimum and fast. Service can be defined as an abstraction on top of a number of pods.

A Kubernetes service deployment has, at least, two parts. A replication controller and a service. The replication controller defines how many instances should be running, the Container image to use, and a name to identify the service. Additional options can be utilized for configuration and discovery.

- Replication Controller

A Replication Controller ensures that a specified number of pod replicas are running at any one time. It defines pods that are to be scaled horizontally. Pods that have been completely defined are provided as templates which are then added with what the new replication should have. It is the responsibility of Replication controller to make sure that a pod or a homogeneous set of pods is always up and available.

Replication Controller supervises multiple pods across multiple nodes. Pods are automatically replaced, deleted or terminated if they fail. As an example, pods are re-created on a node after disruptive maintenance such as a kernel upgrade. If the number of pods is too few, Replication Controller starts more pods. If there are too many pods, extra pods are terminated.

COMPONENTS OF KUBERNETES

The diagram on the next page gives a representation of the components discussed above.

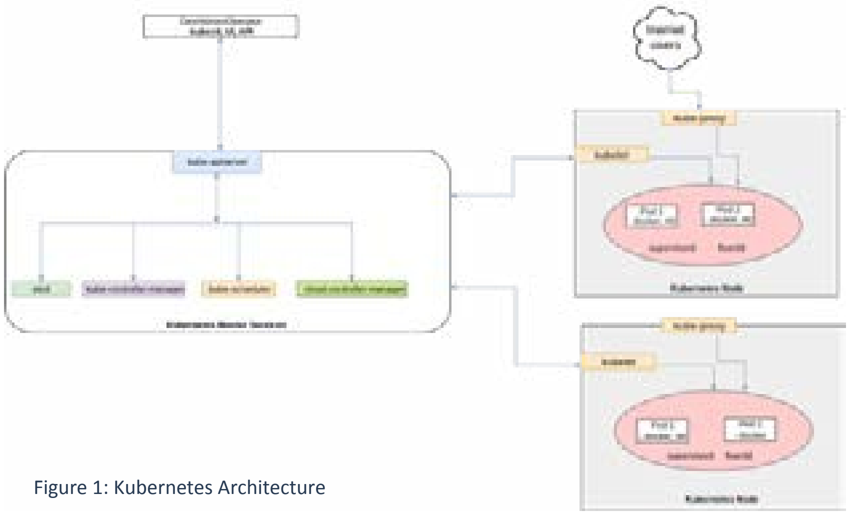


Figure 1: Kubernetes Architecture

Each of these Kubernetes components and how they work is covered in the next table. Note that it's broken into two parts – Kubernetes Master and Kubernetes Node. For Kubernetes Node to function in coordination with master services, there exist control plane within the Master Node.

Kubernetes Master Component	Function of each component
kubectl	• Tmhaisbilsesacoummaunnd clionme mfrraatenrdfisaacaegaWmihsiettrh Kubernetes cluster(s) • Epearchforcmoms manaonpdetrhaattioynouonruonnewoitrhmkuorbe ectl resources. • Epoxadms, psleersviocferse,seonudrpcoeitnytpsees.ta.cre jobs, nodes,
Etcd	• Tvhaisluseisstaohreigthay't'asvuasieladbtloe sdtiosrtreibshuatreedkey- configurations and for service discovery • Kanudbearsnaenteisntuesrefsaceetctdo troegstisotreeriwtsaAtcPhIeorbsjoecnts specific nodes for reliable • Fporordhuigchtioanvaeinlavbirilointymaenndtsd,uy roaubinlieteydinto run etcd as a multi-node cluster • Iotfiosdrdecmomemmbeenrdse(dn/t2o) +ru1,nwehtcedreans aisctlhuester nadudminbgeroonfennooddees.wFiollraalwnyayosdi dn-csirzeeadseclteuhseter, number of nodes • Aqunoertucmd , cltuosatgereneeoendsuapdmaatejosrtiotythoef ncoludsetse,ra state. necessary for quorum • Ylooaudcbaanlaanlscoerr.un etcd cluster in front of a

kube-apiserver **kube-controller-manager**	• Torhcihs essetrrvaitciengprKouvbideerns eatnesAcPlIufsoterr • Isttaptreoovfidthese tchluesftreornatenndsetorvtihcees haallrRedEST operations against the cluster. • Trehgeuklaubees-tchoenstraotleleorf-mKuabnaegrneretseesrvciluecster.
kube-scheduler	• Isttadtoeetshtrhoeugwhathheiunsginofgcalpuiss teerrv'sesrhared component/service. • Idteesnirseudrestsactleuster is operating within the • Kpuobdes-sLcihfeecdyucelel.rT hdoseseisnchludedmesandaegpelomyemnet notf of
Cloud-controller-manager	configured pods, deletion and termination. • IasllaclIsuosgtearthNeords erse.sources information from • Itot wdoeprklosyclKousbeleyrnweitescoobnjtercotlsleir nmthaencalguesrter depending on the resources available.. • TefhKisuwbaesrnineterosduced in 1.6 release version
	• Iwt iitshthaneyfuctluouredof integrating Kubernetes • Tthheiisriosw tonefneatulerecsloinudeppreonvdide enrtslydfervoemlotphe core Kubernetes cycles. • Tinhtehcelocuorde-cKounbtreorlnl etems auntializeersparmoveicdleodud libraries as kube-controller-manager • Acasnoifmv1p.l8e,mtheentclsoeurvdiceonctornotlrleo rllmer,anager rPoeursteiscteonntVrolulemr,enLoadbeeclAodnmtro islsleiornand Controller.

Table 1: Kubernetes Master Services

Kubernetes Node Component	**Role**
Kube-proxy	• Tlohaids bsaelaanycinieginagctosfassearvniectewroурnknpinrogxoyn aansdimdgolees
Kubelet	worker node. • Kcluubstee-rp.roxy usually runs on each node in the • IEtnwdaptocihnets tahdedmitiaosnt/erremforovSaelravnicdedaoneds load bfboarlwanacrnidningtharnodugrohusnimd-prolebtUihnDaPc,riTöeCsnsPastsreetaomf anaycktheindgsaebrovuicteKsuwbiethrnoeuttes oercSlervtisceksnorwing Pods. • Kimupblee-mpreonxtyinisgaalsformesopfovnisritbulaelfIoPr for services of types other than ExternalName. • Teahcihs insatdhe ipnritmhearcylunsotedre agent running on
Supervisord	• IJtSOgeNtsfothrme caot nfrfiogmurtahteioanpoisfearvpeordainndYeAnMsuLr /e tchoantfitghuercaotinotnasinaerresrduensncirnibgea dnidnitnhhoseealthy state. • Ictredaoteesdn'otumtsaindaegKeucboenrtnaeinteesrs which were • Situpisersvyisortemis thatallaof wsecltioenat/sdercveonr ntsryosltaemnutmhabtearlloof
Container Engine –	processes on UNIX-like operating systems. • IcnonKtuabineerrnentegsi,nseuapnedrvkisuobredle mt are asluwreays in running state. These runs the configured pods on worker
Rkt, docker, e.t.c	• nodes • Iatcdtsoaes

	rduonwtnimloeaedninvgironcmoenntatifnoerrciomnatag iensearnsd

Table 2: Kubernetes Node Components

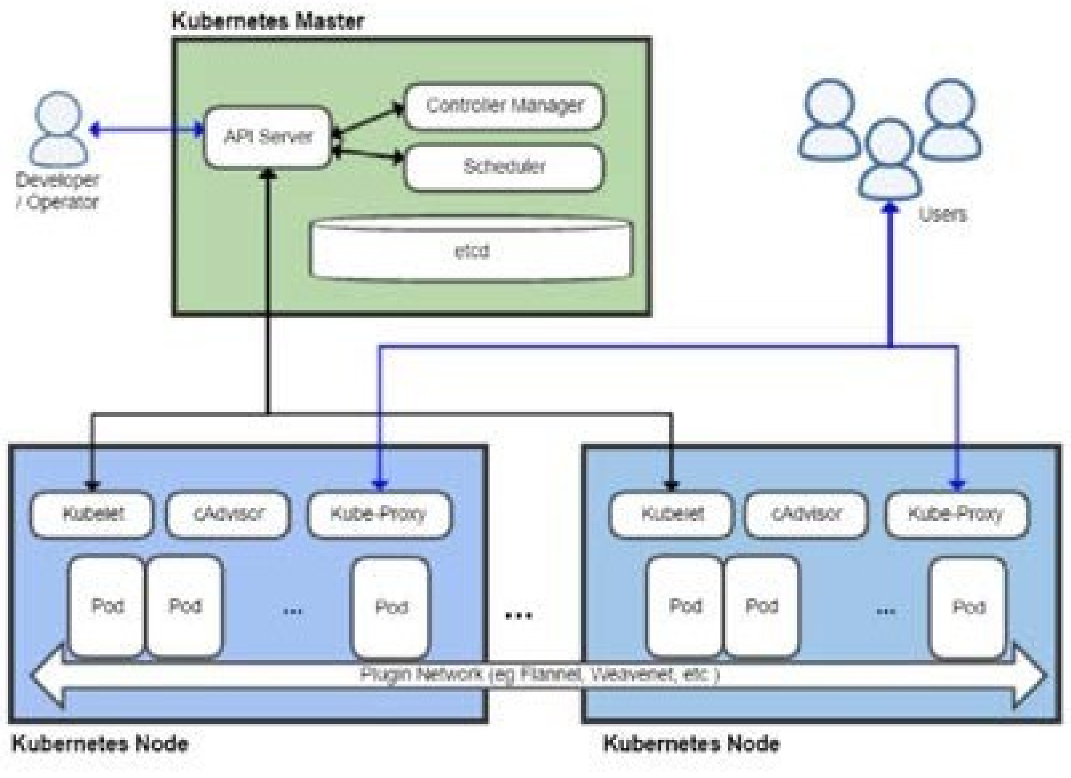
Kubernetes Concepts
Pods
Controllers

Deploying Kubernetes Manually

- **Install Docker Engine on Ubuntu**
- **Installing Kubernetes on Ubuntu**
- **Installing etcd 2.0 on Ubuntu**
- **Installing Addons**
- **Downloading Kubernetes Docker Images**

Kubernetes Concepts

To fully understand Kubernetes operations, you'll need a good foundation on the basics of pods and controllers. We'll refer to the diagram below while explaining these concepts.



Pods

In Kubernetes, a Pod is the smallest deployable object. It is the smallest building unit representing a running process on your cluster. A Pod can run a single container or multiple containers that need to run together.

A Pod can also be defined as a group of containers that share resources like file systems, kernel namespaces, and an IP address.

A pod encapsulates the following pieces

- Application container; single or many containers
- A unique network IP address; each pod has an IP address
- Storage resources; All containers in a pod share same storage
- Options governing how containers should run

A single instance of an application is Pod. This instance of an application can be run on a single container or on a small number of containers that share resources and are tightly coupled. Pods support a number of container runtime environments though docker is the most common in Kubernetes.

There are two models of running pods in Kubernetes:

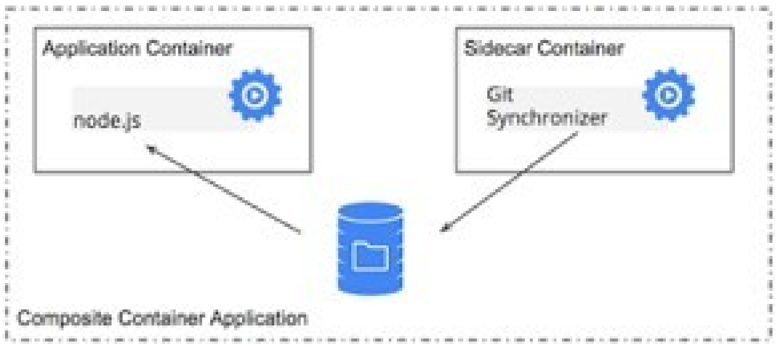
- One container per pod - This is the most common model used in Kubernetes. In this case, a pod is a wrapper around a single container. Kubernetes then manage pods instead of directly interacting with individual containers.

- Multiple containers per pod: In this model, a pod encapsulates an application that runs on a multiple co-located containers that share resources and are tightly coupled. These co-located containers might form one container that serves files from a shared volume to the public while one container tracks and updates changes of these files.

When talking about pods in Kubernetes, there are different types of containers that you need to know:

Sidecar containers

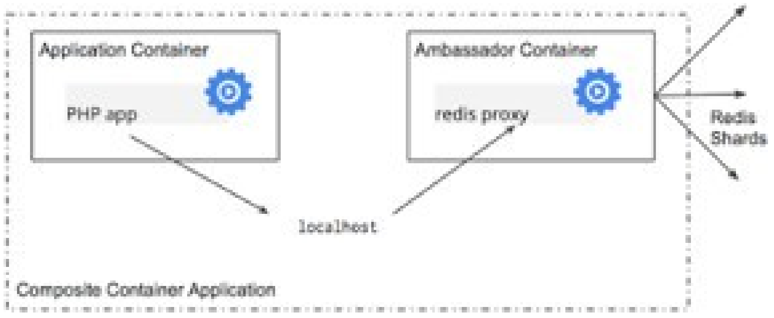
- **These are containers which assist the main container. They take main container better in its functionalities.**



From this diagram, the sidebar container does pulling of updates from git and application controller then serve these files on application server.

Ambassador containers:

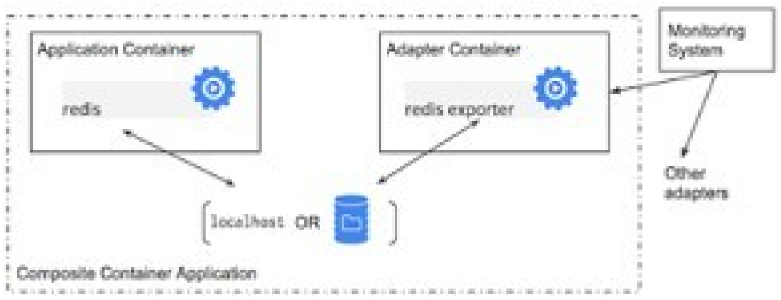
- **These are containers which proxy a local connection to the world.**



As shown above, the ambassador container runs Redis proxy service. This connects to application container via localhost, but the proxy make the application accessible from outside.

Adapter containers:

- **The main work of this is to standardize and normalize output.**



How Pods manage multiple Containers

By design, Pods support multiple cooperating containers which work as a single unit of service. The containers in a pod are automatically co-located and co-scheduled on the same virtual machine or physical node in the cluster. All containers in the same pod communicate with each other and can share resources and dependencies. They coordinate on when and how they get terminated. There are two kinds of shared resources provided by Pods for constituent containers, these are:

s in a Pod access shared volumes and all data in those volumes. In this way, if one container in a Pod is destr

Networking

In Kubernetes, each Pod is assigned a unique IP address. All containers in the same Pod will share network namespace- IP address and ports.
Containers on a Pod are able to communicate with each other through the localhost. When containers in a Pod need to reach the outside, they have to coordinate how they use shared network resources.

Pods creation

A controller can make your task easier by creating and managing multiple Pods for you using a Pod Template you provide. Kubernetes controller also manage replication, rollout and self-healing features. If a Node fails, the Controller can schedule the creation of identical Pod on a different Node.

Pods Lifetime

Pods are mostly managed using a Controller and they are ephemeral entities. When a Pod is created, it is scheduled to run on a Node in the cluster. The Pod will keep running in that specific Node until the parent process is terminated – end of lifecycle, then the Pod object is deleted

Pods by themselves do not self-heal. A pod is deleted if a Node it was running on fails or if the scheduling operation itself fails. If a Pod is evicted for lack of resources, it will be removed as well.

Controllers

The major controller component is ReplicationController which work to ensure that a specified number of pod replicas are running at any one time. It makes sure that a pod or a homogeneous set of pods is always up and available.

When there are too many pods, the ReplicationController will terminate the extra pods. If the number of pods is too low, ReplicationController will start more pods. It does this using application-provided metrics like CPU utilization. Note that Horizontal Pod Autoscaling does not apply to objects that can't be scaled, for example, DaemonSet. Advantage of having pods maintained by ReplicationController is that if a pod fails for any reason, it will automatically create another pod to replace failing one. For this reason, it is always recommended to use ReplicationController even if you have only one pod.

ReplicationController can supervise multiple pods across multiple nodes. Have a look at the below ReplicationController configuration which runs three copies of the caddy web server.

```
$ cat replication.yml

akpiniVde:
rRseiopnli:cva1tionControll
er mneatmadea:tcaa:ddy

spreepcl:icas: 4



  template:

      name: caddy

      app: caddy

      containers:

        image: caddy

        - containerPort: 80
```

From above code snippet, you can see that we have specified that four copies of caddy web server be created. Container image to be used is caddy and port exposed on the container is 80

Create replicas by running the command:

```
        $ kubectl create -f ./replication.yaml replicationcontroller "caddy" created
```

Give it some seconds to pull image and create container, then check for status:

```
$ kubectl describe replicationcontrollers/caddy
-P-o- ds Status:4 Running / 0 Waiting / 0 Succeeded / 0 Failed
```

If you would like to list all the pods that belong to the ReplicationController in a machine readable form, run:

```
$--pooudtps=u`tk=ujsboencptal tghe=t{p.iotedms -s-.s.emleecttaodra=tap.npa=mngei}n`x\
$ echo $pods
```

Rescheduling

ReplicationController makes it easy to do rescheduling. You can just change the value of replicas and redeploy, the specified number of replicas will be scheduled for creation accordingly. ReplicationController always ensure that the specified number of pods exists, even in the event of node failure or pod termination

Scaling

You can easily scale the number of replicas up and down using auto- scaling control agent or through manual process. The only change required is on the number of replicas. Please note that Horizontal Pod Autoscaling does not apply to objects that can't be scaled, for example, DaemonSet.

Rolling updates

ReplicatioController facilitates rolling updates to a service by replacing pods one-by-one. To achieve this, you use a Deployment which in turn uses a ReplicaSet. Replication controller is able to update one pod at a time to avoid any service downtime. The command used is kubectl rolling-update. It is always recommended to use Deployment which is a higher-level controller that automates rolling updates of

applications declaratively.

4. Deploying Kubernetes Manually

In this section, we're going to cover steps used to install Kubernetes on CentOS and Ubuntu Linux base operating systems. There are many ways to deploy Kubernetes, one of them is manual deployment, and second method is automated deployment with configuration management tools like Ansible, Puppet or Chef.

Manual deployment of Kubernetes includes building different components of Kubernetes one by one to create a working Kubernetes cluster.

In our Lab, we'll setup one Kubernetes master and two Kubernetes nodes. This Lab is done on VirtualBox. These three virtual machines will be created using vagrant. Vagrant is a software applications available on Windows, Linux and Mac which allows you to easily build and maintain portable virtual software development environments.

Prerequisites:

1. **Install VirtualBox**
2. **Install Vagrant**
3. **Spin up three VMs**

Install VirtualBox:

VirtualBox installation varies depending on your base operating system. Refer to official documentation for your specific OS. If you're using Ubuntu 16.x, use the following commands to get latest version of VirtualBox.

```
#coencthroibd>eb/ehtct/tapp:/t//dsourncleosa.dli.svti.rdt/uvairlbtuoaxl.boorgx/.vliisrttualbox/

          #| swudgeota-pqth-ktetpysa:/d/wd w-  w.virtualbox.org/download/oracle_vbox_201

                              #apwt-gkeety-qadhdtt-ps:/w-O- | sudo

   $ sudo apt-get inpsdtaltlevirtualbox
```

Install Vagrant

Vagrant is an open-source software product for building and maintaining portable virtual software development environments, e.g. for VirtualBox.
It makes creation of Virtual Machines easier. If you don't already have Vagrant, install using the following commands:

```
$ sudo apt-get uinpsdtaltlevagrant
```

After successfully installing Vagrant, We can proceed to create three VMs needed for this Lab. Initialize vagrant environment using below commands:

```
mkdir kubernetes_lab
$ cd kubernetes_lab
vim Vagrantfile
```

If you don't have ssh key, you can generate using command below:

```
$Gessnhe-rkaetiynggenpublic/private rsa key pair.
Enter fipalessinphwrhasiceh(etmo
spatvyefothrenkoepya(s/shpohmraes/jem):utai/.ssh/id_rsa): id_rsa
EYonuterridsaemnteifipcaastsiopnhrhaasse baegeanins:aved in id_rsa.
YTohuerkpeuybfilnicgkeerpyrhinast ibse: en saved in id_rsa.pub.
SdHevA.j2m56ta:8i.Fc2oOmbfrwvIa4/rn3oCHjnx5FgEsxVH/MJP1pf17
mgt4 jmutai@
T+-h--e[RkSeAy's2r0a4n8d]-o--m-+art image is:
|      .o+.+o = ||
|     .o. ... +* .+o      ||
|        S +o = =. |.|
|      .o.o.=.@...++oo. |o|
|+----[.S+H=OA=2*5o6E]-.-|---+
```

By default generated ssh keys will be located under $HOME/.ssh directory. Now that you have ssh keys that we'll use to ssh to the VMs, it is time to write Vagrantfile used to automatically bring the three VMs up. Vagrantfile uses ruby programming language syntax to define parameters. Below is a sample Vagrantfile contents used for this Lab.

```
rruubbyy:-*-                                        # A l V  gation is done below. The "2" in V

   conlfiguarerant configur
```

```
# cboanckfiwguareds
tchoemcpoantfiibgiulirtayt)i.oPnlevaesresidoonn('twcehsaunpgpeoirttu
onlldeesrs sytoyuleksnfow w# hyoaut 're doing.
Vcaognrafingt.v.cmon.bfiogxu=re"(u"b2"u)ndtou/|xceonifiagl6| 4"
 cwoenbfi.vgm.vm.n.edtewfionrek""kpuubbelircn_entet
 w-morakst",eirp": d"1o9|2w.1e6b8| .60.2"
 ewndeb.vm.hostname="kubernetes-master"
 cwonefib.gv.mvm.n.deetwfinorek"k"puubbelrinc_enteest
 -wnordke"-,0i1p": d"1o9|2w.1e6b8| .60.3"
 ewndeb.vm.hostname="kubernetes-node-01"
 cwonefib.gv.mvm.n.deetwfinorek"k"puubbelrinc_enteest
 -wnordke"-,0i2p": d"1o9|2w.1e6b8| .60.4"


end
```

Once you have the file saved as Vagrantfile. Now create virtual machines using from the Vagrantfile. Note that you need to be on same directory as Vagrantfile before running the command shown below:

```
griantmupmaster' up with 'virtualbox' provider... Bng ngachine 'kubernetes-node-01' up with 'virtuall

  pBroinvgidinegr..m.  achine 'kubernetes-node-02' up with 'virtualbox'

 proi vidi er..m.

                                   Importing base box 'ubuntu/xenial64'...
=> kubernetes-master: MChaetchking iMf bAoCxa'dudbruenstsuf/oxrenNiAaTl6n4'eitswuoprktoing..

   date...
```

```
=k=u>bekrunbeetrens-emteass-mtera_s1te5r0:9S8e1t9t1in5g76t8h2e_n2a7m2 e of
the VM: kubernetes_lab_
=in=t>erkfuabcesr.n..etes-master: Clearing any previously set network
=co=n>fikguubreartnioente..s.-master: Preparing network
   interfaces based on kubernetes-master: Adapter 12:
   nbraitdged
==k>ukbuebrenrenteste-ms-amsatesrte: r2:2F(ogruweastr)d=in>g2p22o2rt(sh..o.st)
(adapter 1)
==> kubernetes-master: RBouontniningg V'pMr.e.-.boot' VM customizations...
=fe=w> kmuibneurtnese.t.e.s-master: Waiting for machine to boot.
   This may take a kubernetes-master: SSH audsedrrneassm:
   e1:27u.b0u.0n.t1u:2222
   kubernetes-master: SSH auth method: password
   kubernetes-master: RInesmerotvininggiennseercautreedkpeuybflriocmketyhwe
   igtuheinstgiuf eits'st...

   kubernetes-master: Key inserted! Disconnecting and reconnecting

==> kubernetes-master: Machine booted and ready!
==k>ukbuebrenrenteste-ms-amsatesrte: rT:hCehgeuceksint
agdfdoirtigounesstoandtdhiitsioVnMs idnoVnMot..m.
                                                                atch

thkeuinbsetranlleetdesv-emrasisotenro: fVirtualBox! In most cases
this is fine, but in rakreubcaesrenseittesc-amnaster: prevent things
such as shared folders from wokrukbinergnpertoeps-
emrlay.stlefry:osuhasered folder errors, please make sure the
gukeustbaedrndeittieosn-ms wasiterin: vtihretual machine match
the version of
```

```
    kubernetes-master: your host and reload your VM.
    kubernetes-master:
    GViuretustaAlBdodxitVioenrssioVner: s5i.o2n: 5.0.40
==> kubernetes-master:
SCeotntifingguhrionsgtnaanmdee.n..abling network
i=n=t>erkfuabcer.n..etes-master: Mounting shared folders...
==k>ukbuebrenrentete-ms-ansotdeer:-0/1va:
gImrapnotr=t>in/ghobmasee/bjmoxut'uaib/ukunbtue/rxneentieasl_6l4a'.b..
=n=e>twkourbkeinrnge..t.es-node-01: Matching MAC address for NAT
=d=a>tek..u. bernetes-node-01: Checking if box 'ubuntu/xenial64' is up to
=la=b>_kubernetes-node-01_S1e5t0t9in81g9t2h1e0n67a6m_e63o6f8t9he VM:
kubernetes_
=p=o>rtk2u2b0e0r.netes-node-01: Fixed port collision for 22 => 2222. Now on
=in=t>erkfuabcesr.n..etes-node-01: Clearing any previously set network
=co=n>fikguubreartnioente..s.-node-01: Preparing network
    interfaces based on kubernetes-node-01: Adapter 12:
    nbraitdged
==k>ukbuebrenrenteste-ns-ondoed-0e1-0: 12:2F(ogruweastr)d=in>g2p20o0rt(sh..o.st)
(adapter 1)
==> kubernetes-node-01: RBouontniningg V'pMr.e.-.boot' VM customizations...
=a=f>ewkumbienrunteetse.s..-node-01: Waiting for machine to boot.
    This may take kubernetes-node-01: SSH audsedrrneassm:
    e1:27u.b0u.0n.t1u:2200


    kubernetes-node-01: Warning: Authentication failure. Retrying...
```

kubernetes-node-01: Inserting generated public key within guest... prkeusebnetr.n..etes-node-01: Removing insecure key from the guest if it's uskinugbenrenwetSeSsH-nkoedye.-.0. 1: Key inserted! Disconnecting and reconnecting
==> kubernetes-node-01: MChaecchkininegbfortgeudeasnt dadrdeiatdioyn! s in VM... thkeuinbsetranlleetdesv-enrosdioen-0o1:f The guest additions on this VM do not match rakreubcaesrenseittesc-annode-01: VirtualBox! In most cases this is fine, but in wokrukbinergnpertoeps-enrolyd.el-f0y1o:uprseveent things such as shared folders from gukeustbaedrndeittieosn-ns owdieth-0in1:tshheared folder errors, please make sure the VikrtuubaelBrnoexteyos-unhodavee-0i1n:svtairltluedalomnachine match the version of kubernetes-node-01: your host and reload your VM.
kubernetes-node-01: GViuretustaAlBdodxitVioenrssioVner: s5i.o2n: 5.0.40
==> kubernetes-node-01:
SCeotntifingguhrionsgtnaanmdee.n..abling network
i=n=t>erkfuabcer.n..etes-node-01: Mounting shared folders...
==k>ukbuebrenrent ete-ns-ondoed-0e1-0: 2/v:
aImgrpaonrtt=in>g/hboamseeb/jomxu'utabiu/knutub/exrenneitaels6_4l'.a.b.
=n=e>twkourbkeinrnge..t.es-node-02: Matching MAC address for NAT
=d=a>tek..u. bernetes-node-02: Checking if box 'ubuntu/xenial64' is up to
=la=b>_kubernetes-node-02_S1e5t0t9in81g9t2h6e7n47a5m_e56o9f9t4he VM: kubernetes_
=p=o>rtk2u2b0e1r.netes-node-02: Fixed port collision for 22 => 2222. Now on

```
=in=t>erkfuabcesr.n..etes-node-02: Clearing any previously set network
=co=n>fikguubreartnioente..s.-node-02: Preparing network
   interfaces based on kubernetes-node-02: Adapter 12:
   nbraitdged
==k>ukbuebrenrenteste-ns-ondoed-0e2-0: 22:2F(ogruweastr)d=in>g2p20o1rt(sh..o.st)
(adapter 1)
==> kubernetes-node-02: RBouontniningg V'pMr.e.-.boot' VM customizations...
=a=f>ewkumbienrunteetse.s..-node-02: Waiting for machine to boot.
   This may take kubernetes-node-02: SSH audsedrrneassm:
   e1:27u.b0u.0n.t1u:2201
   kubernetes-node-02: SSH auth method: password
   kubernetes-node-02:
   RInesmerotvininggiennseercautreedkpeuybflriocmketyhwe igtuheinstgiuf
   eits'st...

   kubernetes-node-02: Key inserted! Disconnecting and reconnecting

==> kubernetes-node-02: Machine booted and ready!
==k>ukbuebrenrenteste-ns-ondoed-0e2-0: 2T:hCehgeuceksint
agdfdoirtigounesstoandtdhiitsioVnMs fdnoVnMot..m.
                                                                        atch

thkeuinbsetranlleetdesv-enrosdioen-0o2:f VirtualBox! In most cases
this is fine, but in rakreubcaesrenseittesc-annode-02: prevent things
such as shared folders from wokrukbinergnpertoeps-enrolyd.el-
f0y2o:ushsaereed folder errors, please make sure the
gukeustbaedrndeittieosn-ns owdieth-0in2:tvhiertual machine match
the version of VikrtuubaelBrnoexteyos-unhodavee-0i2n:sytoaluler
dhosnt and reload your VM.

   kubernetes-node-02: Guest Additions Version: 5.0.40
```

```
kukberenrent ete-ns-ondoed-0e2-0: 2V:iSrteuttailnBgohxoVsetrnsaiomne:.5...2
==>ubling network

    i=n=t>erkfuabcesr.n..etes-node-02: Configuring and enab  rs...

kukbernrenteste-ns-ondoed-0e2-0: 2/v: aMgoraunntti=n>g/shhoamreed/jmfouldteai/kubernet
```

The command above will download Ubuntu Xenial vagrant image and create three Virtual Machiness with specified names - kubernetes-master, kubernetes-node-01 and kubernetes-node-02. All these VMs will be on the same subnet 192.168.60.0/24.

Confirm that the VMs were successfully created:

```
$ vagrant status
Current machine states:

kubernetes-mnoadsete-r0r1unnnniinngg((vviirrttuuaallbbooxx))

  ubernetes-node-02             running (virtualbox

enwviitrhonthmeeirncturrerpernetsestnattsem. FudrtimploerVeMinsfoTrmheaVtiMons aabreouatllali

  VM, run `vagrant status NAME`.
```

Now ssh to the Kubernetes master node and update apt cache, then do system upgrade:

```
$suvdaogrsaun-t ssh kubernetes-master

  apt-get upgraatede && apt-get dist-upgrade
```

Do the same on both Kubernetes nodes - Perform system update and upgrade.

```
$ vagrant ssh kubernetes-node-01

    apt-get update && apt-get
             upgrade              && apt-get  dist-upgrade

$ vagrant ssh kubernetes-node-02

    apt-get update && apt-get     && apt-get  dist-upgrade
```

```
             upgrade
```

Now that Kubernetes master node is ready, let's proceed to install docker engine community edition on this VM.

Install Docker Engine on Ubuntu

Docker is a platform for developers and system administrators to develop, ship, and run applications. One of the key components of Docker is docker engine which is a lightweight and powerful open source containerization technology combined with a workflow for building and containerizing your applications. Kubernetes depends on docker engine to run containers.
Though other container runtimes like rkt and lxc can be used, the most mature and popular is docker.

Docker engine can be easily installed on Ubuntu from official apt repositories provided by Docker Inc. Follow steps below to setup Docker repository and install docker engine on Ubuntu host.

1. **Update the apt package index:**

```
$ sudo apt-get update
```

2. Install packages to allow apt to use a repository over HTTPS:

```
$ saupdt-otraapnts-gpeotritn-hsttaplls\\

          cu-rcle\rtificates \

           software-properties-common
```

3.Add Docker’s official GPG key:

```
$apctu-krley-fsaSdLdh- ttps://download.docker.com/linux/ubuntu/gpg | sudo
```

4.Set up the stable repository on Ubuntu 16.x

```
oh_oredleebas[ear-cchs)=Satmabdl6e4>] h/ettc/sa:/p/dt/oswounrlcoeasd.l.disot.cdk/edro.ckmer/.lisntux/
```

5.Update the apt package index and install Docker CE.

```
sudo apt-get update && apt-get install docker-ce
```

Another way to install docker is by using script provided at get.docker. com. The scripts attempt to detect your Linux distribution and version and configure your package management system for you. You only need to be root or use sudo to run this script. To install Docker with this script use the commands shown below:

```
$ csudrlo-fsshSgLegt-edt.odcokcekre.srh.com -o get-docker.sh
```

If you would like to use Docker as a non-root user, you should now consider adding your user to the "docker" group with something like:

```
$ sudo usermod -aG docker vagrant
```

If you are using different username, remember to replace vagrant with that name. Check version of docker installed using the command below:

```
$Cldieonckt:er version

 VAePrIsvioenrs:ion1:

 71.1.303.0-ce
                     gfo41ff.d82.35
 Git

 vceormsimo

 nit::

 BOuSi/Al
 tr: ch:     TuleinOucxt/1a7m1d96:404:16 2017


SVeerrvseiro:n:     17.10.0-ce
 AGPoIvveerrssioionn: : g1o.313.8(.m3
 inimum version 1.12) GBuitilcto: mmTitu:
 efO4ffctd127519:02:56 2017
 OExSp/Aercimh:entaliln: ufaxl/saemd64
```

Now that docker is installed, we can proceed to install Kubernetes on the master node.

Install Docker Engine on Ubuntu

This section provides instructions for installing Kubernetes and setting up a Kubernetes cluster. Kubernetes can run on various platforms; Virtual Machine on cloud, Laptop, Virtual machine on VirtualBox, to bare metal servers. The effort required to setup Kubernetes cluster varies depending on the need for setting up the cluster. Test environment can be done using minikube while for production environment, a number of customizations might be required, for this manual setup will work fine.

Install dependencies:

```
apt-get install -y transport-https
```

Add key for new repository:

```
>c/hetocd/aepbt/hstotupr:/c/easp.tl.ikstu.db/ekrunbeetersn.eiote/ sk.ulibsternetes-$(lsb_release -cs) mai
```

Update apt index:

```
>c/hetocd/aepbt/hstotupr:/c/easp.tl.ikstu.db/ekrunbeetersn.eiote/ sk.ulibsternetes-$(lsb_release -cs) mai
```

Install kubelet, kubeadm kubectl and kubernetes-cni:

```
apt-get install -y kubelet kubeadm kubectl kubernetes-cni
```

Short description of installed packages:

Kubelet: This is the core component of Kubernetes. It is a primary "node agent" that runs on each node and does things like starting pods and containers.

Kubeadm: Used to easily bootstrap a secure Kubernetes cluster.

Kubect1: This is a command line interface for running commands against Kubernetes clusters. This is only used on the master.

Kubernetes-cni: This enables cni network on your machine. CNI stands for Container Networking Interface which is a spec that defines how network drivers should interact with Kubernetes

Once you have all the master components installed, the next step is initialize cluster on the master node. The master is the machine where the "control plane" components run, including etcd (the cluster database) and the API server (which the kubectl CLI communicates with). All of these components run in pods started by kubelet.

Initialize master cluster

On your master node, initialize cluster by running the command:

```
kubeadm init
```

This will download and install the cluster database and "control plane" components. This may take several minutes depending on your internet connection speed. The output from this command will give you the exact command you need to join the nodes to the master, take note of this command:

```
.Y.o. ur Kubernetes master has initialized successfully!
To start using your cluster, you need to run (as a regular user):
```

```
msukddoircp-p-i$/HetOcM/kEu/b.keurnbetes/admin.conf $HOME/.kube/config


YRouuns"hkouubledctnloawppdleyp-
Ifo[ypoadpnoedtwneotrwk]o.yrakmtol"thweitchluosntero. f the options
lihsttpd:/a/tk:ubernetes.io/docs/admin/addons/

Yoonueacachn nodwejoin any number of machines by running the following


ckouvebreya-dtomkejoni-nca--ctoekrte-
nha4s5h3as5hea.235d62:f5e3d6efe7e27f85c76cdb4082d1701.103.25.81f5d:76f41493c-
e-cd-is-
```

Before you join a cluster, run the commands:

```
msukddoircp-p-i$/HetOcM/kEu/b.keurnbetes/admin.conf $HOME/.kube/config

    sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

On the two Kubernetes nodes, run the following command to join them to the cluster:

```
$cokvuebreya-tdomkejno-icna-c-teorkte-hna4s5h3sah5ae2.35d62:5f3e6dfe7e2ef785c76cdb408

         csu0bd9o0c3p82-i4/ae9t0c/bkfu8be4rn70eetef1s/fa7dbmd3i4nf.9c4o8n9f2$HOM

                   sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Remember to replace token id and --discovery-token-ca-cert-hash values with the ones you got from the output of ‘kubeadm init’ command.

Installing etcd 2.0 on Ubuntu

Etcd 2.0 is a Distributed reliable key-value store for the most critical data of a distributed system. It focuses mainly on being Secure, Simple, Fast and Reliable. It is written in Go programming language and Raft consensus algorithm that manages a highly-available replication log,

There are two ways to install etcd on Ubuntu – One being building it form source, next being using pre-built binary available for download. If you're interested in getting the latest release, consider building it from source.

To build it from source, you'll need installed git and golang, then run:

```
$ gcdituectlcodne https://github.com/coreos/etcd.git

$ ./binil/detcd
```

Installing Addons

Here I'll show a number of plugins that you can install to extend Kubernetes functionalities. This list is not exclusive, feel free to add what you feel might help.

Deploy Flannel Pod Network

Flannel is an overlay network provider that can be used with Kubernetes. Let's configure Flannel pod network by first installing in on the master node. Follow steps provided here to get flannel network plugin up and running.

```
noneklu/mbeæsttlearp/Dpolycu-fmhetntptast:i//orna/wk.ugbiteh-ufbaunsneerlc.yomntlent.com/coreos/

neklu/mbeæsttlearp/Dpolycu-fmhettnptsa:t/i/oranw/k.guibtheu-lbaunsneercl-ornbtaecn.yt.mcolm/coreo

$ ./buinil/detcd
```

For further reading about Flannel, refer to https://github.com/coreos/flannel

Deploy CoreDNS

CoreDNS is a flexible and extensible DNS server written in Go which you can use on Kubernetes setup to manage Pod DNS records. It chains plugins which manage DNS functions such as Kubernetes service discovery, Prometheus metrics or rewriting queries. CoreDNS is able to integrate with Kubernetes via the Kubernetes plugin, or directly with etcd plugin. Mostly you'll either server zone

To deploy CoreDNS on Kubernetes do:

1. **Install Golang and git:**

```
sudo apt-get install golang git
```

2. **Verify Installation**

```
#gogovevresriosinongo1.6.2 linux/amd64
```

3. **Set GOPATH environment variable.**

```
mexkpdoirrt~G/gOoPATH=$HOME/go

export PATH=$GOPATH/bin:$PATH
```

4. **Compile CoreDNS application using go**

```
go get github.com/coredns/coredns
```

Another way to install CoreDNS is from a binary of latest release. To check latest release visit > https://github.com/coredns/coredns/releases

Then download latest version, uncompress and copy binary to /usr/local/ bin/

```
awpgte-gt ehtttipns:t/a/gllitwhguebt.com/coredns/coredns/releases/download/v0.9.10/

aorrezxdvnfs_c0o.r9e.d1n0_s_li0n.u9.x1_0a_mlindu64x._tagmz  d64.tgz

      p coredns /usr/local/bin/
```

Test that the binary is copied and working by checking coredns version:

```
#Co/uresDr/NloSe-a0l./9b.i1n0/coredns -version

      linux/amd64, go1.9.1, d272e525
```

For more information about CoreDNS, visit official project page on github:

Deploying Kubernetes Dashboard

Kubernetes Dashboard is a general purpose, web-based UI for managing and configuring Kubernetes clusters. This Dashboard aims at making applications running in the cluster easier to manage and troubleshoot. It provides for a secure setup and by default it has minimal set of privileges with access only through https.

To deploy Dashboard execute following command on master node command line interface:

```
atrlda/pmpalyst-efrh/sttrpc/sd:/e/rpalwoy.g/ritehcuobmumseerncdoendte/knut.bceormn/ekteusb-edran

   yaml
```

Before you can start using Kubernetes Dashboard, you'll need to configure the proxy server using kubectl command. This will set url for you which you can use to access the dashboard. Run command below to get proxy configured:

```
$ kubectl proxy
```

After successful execution of this command, dashboard is now accessible on http://localhost:8001/api/v1/namespaces/kube-system/services/ https:kubernetes-dashboard:/proxy/

To learn more about using Kubernetes dashboard, visit Kubernetes Wiki

Getting Kubernetes Images

Now that you have ready environment, let's look at how to download docker images for use with Kubernetes. A docker image is an inert, immutable file that's essentially a snapshot of a container. This contains OS utilities and basic tools required to run an application. In this example, I'll show you how to download busybox docker image:

```
#Usdioncgkdeerfpaullt btaugs:ylbaotexst

afteasdtd: 5P8sufhl2lain6g: Pfrubolbmlccolimbrpalreyt/ebusybox

9S9igce9st7:6df0a32556d: 24f03a03235220b170ba48a157dd097dd1379299370e1ed

tatus: Downloaded newer image for busybox:latest
```

Confirm that the image has been downloaded:

```
 EdPoOckSeIrT      TxeAn      IMd6AfG76Ed
 OimRaYges         Gial       I9Dcc90

                                              CREATED       SIZ2EMB
 ubbusyenrbt

                                                            16.4103MMBB
                     latest     6c3afd87733650404ea9653
                                  152mdoanytshasgaogo

 ruabssye2n
 4g
 phasussein
                   latest    c3f873600e95       5 months ago    640MB
 onge/ r-
 ruby24
```

As you can see above, the image has been downloaded successfully. We'll use this image in the next section.

Testing:

Now that we have ready Kubernetes cluster, let's create a simple pod on this cluster. As an example, consider below simple Pod template manifest for a Pod with a container to print a message. Pod configuration file is defined using YAML syntax:

```
$ cat pod.yaml


akpiniVde:


rPsoiodn: v1
mneatmadea:ttae:st
 app-pod
 laabpepl:s:testapp
```

```
- name: testapp-container

  command: ['sh', '-c', 'echo This is Kubernetes!! && sleep 3600']
  image: busybox
```

Options used:

apiVersion: v1 - Using API version 1 of Kubernetes
kind: Pod - This specify that we want to create a Pod. Other values include Deployment, Job, Service, and so on
metadata: - Here we have to specify the name of the Pod, as well and the label used to identify the pod to Kubernetes
Spec: - Here we are specifying the actual objects that make up the pod. The spec property includes storage volumes, containers and other information that Kubernetes require, as well as properties like whether to restart a container in case it fails e.t.c. In this case, we have minimal definition:

A container name (testapp-container), the image on which it is based (busybox), and the command that will execute on the container upon creation.

Tell Kubernetes to create its contents:

```
$ kubectl create -f pod.yaml
```

Confirm creation by asking for a name of pods:

```
$NkAuMbeEctl getRpEoAdDsYSTATUS
                                                  RESTARTSAGE
         testapp-pod0/1ContainerCreating0              6s
```

After a few seconds, you should see the containers running:

```
$NkAuMbeEctl getRpEoAdDsYSTATUS
                                                  RESTARTSAGE
       testapp-pod1/1Running            0       6s
```

5. Downloading Kubernetes Docker Images

Setting up Kubernetes Cluster:

There are many solutions available for setting up Kubernetes cluster for the different environment. To get started with the Kubernetes, Minikube is one of the most preferred options.

Before starting with Minikube lets look into other options:

In independent solutions, minikube and kubeadm are the best options to bootstrap Kubernetes cluster. Both options work well with the development environment. In hosted solutions Google Kubernetes Engine, Azure Container Service, and IBM Cloud Container Service are options. It is the less time-consuming process to launch Kubernetes cluster with hosted solutions.

Kubespray and KOPS(Kubernetes Operations) are community supported tools for bootstrapping the cluster. With Kubespray it is possible to setup multi-master Kubernetes cluster on on-premise and cloud platforms.

Minikube is a tool written in Golang to set up the cluster locally on the machine. It will require virtualization to be enabled for Operating System. Minikube supports Container Network Interface (CNI Plugins), Domain Name System, Kubernetes Dashboard, Ingress for load balancing, Config Maps and Secrets and Container runtime which can be docker or rkt.

If Linux is used then, it will require Virtual Machine Driver to run minikube. Following are steps to install Minikube on Linux:

(Note: Following steps are preferred for Ubuntu 16.04)

Install a Hypervisor:

Check if CPU supports hardware virtualization: (If value > 0 then it supports)

```
$ egrep -c '(vmx|svm)' /proc/cpuinfo
```

Download VirtualBox:

```
$5.c2u_r5l.2-L.0O-1h1t8t4p3:/1/d~Uowbunnlotaud~.xveirntiuaal_lbaomxd.o6r4g.d/veibrtuall
```

Install using Debian package manager:

```
$ sudodpkg -i virtualbox-5.2_5.2.0-118431-Ubuntu-xenial_amd64.deb
```

For latest version please check https://www.virtualbox.org/ .

Install Kubectl:

Kubectl is the command line utility which interacts with API Server of the Kubernetes.

Download Kubectl Stable Binary:

```
  $ curl -LO https://storage.googleapis.com/kubernetes-release/
  release/$(curl -s https://storage.googleapis.com/kubernetes-release/
elease stable.txt)/bin/linux/amd64/kubectl
```

Make binary executable:

```
$ chmod +x ./kubectl
```

Move binary to the system path:

```
$ sudo mv ./kubectl /usr/local/bin/kubectl
```

Check if Kubectl is configured or not:

```
$kkuubbecetcltlcontrols the Kubernetes cluster manager.

Find more information at

https://github.com/kubernetes/kubernetes.

BcarseiactCeommCarnedaste(Baergeisnonuerrc)e: by filename or stdinex

a controller, service, deployment or pod run-

containuenr aRunr aicpualratriciular image on the clusterset on

objects
```

Install Minikube:

```
ls-eLso/vm0.i2n2i.k3u/mbeinhitktupsb:e/-slitnouraxg-aem.godo6g4l&ea&pcish.mcoomd/+mxinikube/
    minikube&&sudo mv minikube /usr/local/bin/
```

For latest version please checkhttps://github.com/kubernetes/minikube/ releases.

Verify Installation:

```
$Mmininikikuubbeeis a CLI tool that provisions and manages

single-node Umsiangike:ube [command]

Aadvdaiolnabsle
ComMmodainfydsm:
                         inikube'skubernetesaddons
shcoemll (pbleatsiho)n  Outputs minikube shell completion for the given
cdoansfihgboard ModOifpyemnsi/ndiiksupblaeycsotnhfiegkubernetes
dashboard URL for ydoeulreltoecal cluDsetelertes a local kubernetes
cluster
dmoacckheirn  Sets up dockerenv variables; similar to '$(docker-
-e evnv)'
```

 get-k8s-versions Gets the list of available kubernetes versions

available for
ip   Retrieves the IP address of the running cluster
minikube

The message indicates that Kubernetes cluster is started with Minikube. Docker should be running on the host machine. Minikube will use default container engine (docker here) to run the app.

Create Kubernetes Cluster through Minikube:

Run following command:

```
$minikube start
```

Verify the Kubernetes Cluster Started:

```
$SmtarintiinkgubloecsatlaKrtubernetes v1.7.5 cluster...

SGteatrttiinnggVVMMI..P. address...

MSeottviinnggufplecseirntsto...cluster...

CSeotntinnegcutipngkutobeclcuosntefirg....

SKtuabrteicntgl icslunsotwerccoonmfipguonreedntо..u. se the cluster.
```

If you got this return message then, you successfully started MinikubeKubernetes Cluster locally. Kubernetes API server will be accessed with Kubectl utility.

To verify run following command:

```
$kubectlconfig get-clusters
```

It will list the clusters. You should get the result like:

```
$NkAuMbeEctlconfig get-clusters

  minikube
```

Here Kubectl is successfully configured and Kubernetes cluster will be running on local machine.

Dockerizing the App:

**Containerization is the most important aspect to bundle all the dependencies together we called it as a container. The container can be an application container, web container or a database container. The advantage of the containerization is to run app anywhere regardless the dependencies because container bundles it with Images. Container engine is used for communication between containers and underlying
kernel of Operating System. Docker is the most popular container engine. There are other container engines like rkt, lxc.**

Dockerfile consists the set of instruction to dockerize the application. It bundles all the dependencies.

Here we will use sample NodeJs application. The structure of the basic app will be:

```
a--pspe/rver.js

    -- package.json
```

server.js:

```
      const express = require('express');
      const app = express();



arepsp.s.geentd('/(',HfuelnloctWioonr(lrde!'q);, res) {

   });

acopnp.sloislete.lno(g3(0'E0x0a, mfupnlcetiaopnp(l)is{tening on port 3000!');

   });
```

Package.json:

```
{ "nvearmsieo"n: "":h"e1l.l0o.-0w",orld",
st"adretsecrr/ihpetliloon-"w: o"Hrledl.lhotmwol"r,ld app taken from:
  https://expressjs.com/en/ "smcariipnt"s:"":s{erver.js",
  },"stetasrt"t:":"","
  "retyppoes"i:to"gryit"":, {
  },"url":  "git+https://github.com/borderguru/hello-world.git"
  "aiuctehnosre"":: """I,SC",
  "b"uurgls""::"{https://github.com/borderguru/hello-
  world/issues"
  }, omepage":  "https://github.com/borderguru/hello-world#readme",
  "dcehpaein":d"e^n4c.i1e.2s"",: {
    "emxopcrhesas"": :""^^44.0.1.15".,3",
  } "request": "^2.83.0"

}
```

To Dockerize the app, we need to create the

Dockerfile: Dockerfile:

```
#FRTOhMe onfiodceia:bl oimroange of
the node
#ARAGRGVEVRerSsIiOoNn=t1o.d0.e0f
ine app version

#WCOrReKatDeIaRp/pudsri/rsercct/oarp

ypinside image #ENSeVt

NreOqDuEir_eEdNeVnvpirroodnumcetin

otnvariables

#COInPsYtapllacakpapgdee.jpseon .encies by copying package.json to image


# FCoOrPnYppma@ck5agoer.ljastoenr,pcaocpkyagpea-

clkoackge.j-slonck../json as well RUNnpm install

# Bundle app source

# It will start the app on 3000 port of the container

CMD [ "npm", "start" ]
```

This is well-defined dockerfile. Make sure docker should be pre-installed on the machine. If it is not installed then, install via official documentation of the docker.

The Dockerfile consists set of commands which includes making work directory where necessary files will be copied. Installing dependencies using npm. Note the base image node:boron is an official image from NodeJS and it consists stable npm version. After copying all files and installing dependencies exposing 3000 port of the container and the first command will run npm start.

This is sample Dockerfile but it is not limited to just given commands. For more in-depth information please check official guide of the Docker for creating Dockerfile.

To create the images from the Dockerfiledockercli is used. Now the app structure is:

```
    app/ rver.js

--- peackkaegref.ljeson


       Doc
```

Run the following command:

```
$ sudodocker build -t helloworld:1.0 .
```

It will build the image and tag it with helloworld:1.0 here 1.0 is the version of
the image. If nothing is specified then, the latest version will be chosen.

This will download all dependencies to run the app. After a successful build, check the images.

sudodocker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
helloworld	1.0	c812ebca7a95	Abo a	678
node	boron	c0cea7b613ca	m11induaytes ago	661 MB

To run the container docker run command is used. It’s necessary to bind the node port to container port.

```
sudodocker run -it -p 3000:3000 helloworld:1.0
```

Here its binding port 3000 for container port

3000. Check if app is running:

```
$Hceullrol lWocoarllhdo! st:3000
```

Tag the image and push to dockerhub:

R$sudodocker images SIEZPEOSITORY	TAG	IMAGE ID	CREATED
helloworld	1.0	c812ebca7a95	3 hours ago
678 MB			
node	boron	c0cea7b613ca	11 days ago
661 MB			
$sudodocker tag he	lloworld:1.0	helloworld:late	st
RSIEZPEOSITORY	TAG	IMAGE ID	CREATED
helloworld	1.0	c812ebca7a95	3 hours ago
678 MB			
helloworld	1.0	c812ebca7a95	3 hours ago
678 MB			
helloworld	1.0	c812ebca7a95	3 hours ago
678 MB			
node	boron	c0cea7b613ca	11 days ago
661 MB			

Tagging image to latest to indicate this will be the most recent version of the image. Dockerhub is the central registry for storing docker images. However, there are many other registries available like JFROG, Quay and Amazon ECR.

Login to dockerhub:
(Notice: If you don't know dockerhub then please visithttps://hub.docker. com and create account)

```
dHoakbe.rIlfoygoiun dwoitnh't yhoauvreDaoDcokcekreIrDIDto, phuesahd aonvedrptuolhl itmtpasg:/

 dUoscekrnera.mcoem(ktuobcerjeaactke)o: knue.bejack

        PLoasgsinwoSrudc:ceeded
```

```
$sudodocker tpaugsh
klulobweojarcldk:/lhaetellsotwkuobrledja:lcakte/hstelloworld:latest
T50hce9pa7udshd8r3ebfe6r:sPtuosaheredpository
[docker.io/kubejack/helloworld] 9e47d15f58a2e6a44a9b8594::
PPuusshheedd
574c9e696eb49c979c19747e: PMuosuhnetded
from                                   library/node
c73d0f23510ce1b319aea5c:   Mounted   from
library/node           ed97521aa06331b0e7e4:
Mounted             from            library/node
d5d6640efdcc3e443b059b:                           :
MMoouunnteteddfrfroommlilbibrraarryy/n/no
oddee

cla0t1ecs6t:3dc6ig8e2s3td: :

shMao2u5n6:t9e5d2fffro7em89li5b4r7aer1y5/n76o3d0ebe120126a3e1d8717d45e0d-
```

This will be used in Kubernetes manifests. The app is

successfully containerized now and images are pushed to the dockerhub.

Writing Kubernetes Manifest Files for Sample App:

For now, the sample app is containerized but to run the app over Kubernetes it will require the Kubernetes Objects.

The following are the Kubernetes objects:

Pod :Kubernetes deployment unit. It wraps single or multiple containers. Multiple containers will share network and storage namespace.

Replication Controller / Replica Sets: Replica Sets is next version of the Replication controller. There is more efficiency while working with Replica Sets using Selectors and Labels. This object will instruct replication controller to maintain the number of replicas. Replication Controller / Replica Sets layered on top of pods.

Deployment: Deployment is layered on top of Replica Sets / Replication Controller for automated rollouts and rollbacks. It’s the most important use case while in production we deploy the different version of the application.

DaemonSets : Daemon sets are used for running the pods on each Kubernetes node. This usually used for monitoring and logging.

Services: Services are the backbone of the Kubernetes service discovery mechanism. It's the internal service discover implemented by the Kubernetes. Service act as the proxy for replicas of the pods and redirect the traffic to the appropriate pod. That means it acts as the load balancer for the pods too.

Here we will be using deployment and service object to create kubernetes manifests. For now, create Kubernetes directory.

```
mkdirkubernetes
```

Create deployment.yml and service.yml files

```
ctodukcuhbdeerpnleotyems ent.yml

    touch service.yml
```

Now the structure of the directory will be:

```
a--p- pspe/ rcvkearg.jes.json

k--u- bDaocnkeetreifsl/e



--- dseerpvliocyem.yemnlt.yml
```

Kubernetes manifests are plain yaml files which will define the desired state of the cluster.

deployment.yml

```
akpiniVde: rDseiopnlo:

yemxtentsions/v1beta1me

abpepl:shello-worldver:



selelcictor:matc  Labe s:
```

```
temmeptaladtaet:a:

    laabpepl:s:hello-world
    ver

   spc: oevc1:tainers:

- inmnaamgee:: hkeulbloe-jwacokr/lhdelloworld:latest


imagp- eoPrutsl:lPolicy: Always

              containerPort: 3000
```

Let's understand the deployments.

Deployment Object:

deployment.yml:

```
akmpineiVde: rDseiopnlo: yemxtentsions/v1beta1

        nlaatmadea::thae: llo-world

          v  abp: epl:shello-world

              er  v1
```

Kubernetes API Server is responsible for all tasks related to Kubernetes. Here the deployment is the object of the Kubernetes. apiVersion indicates which API it is using. apiVersion consists two different type of version. It consists the extensions and apps.

Kind indicates the type of kubernetes object it can be pod, deployment, service etc. Metadata is the naming convention for the kubernetes object. So this deployment will be known by name: hello-world.

Labels are important for service discovery. Labels are attached to

the kubernetes object. And all operations on API server are with Labels.

Replica Set Object Specification:

deployment.yml

```
sprseeplcl:ictoasr: 10

maatecpchpL: ahbeellos:-world



  vte  : vp1late:

    mlaebtaedlsa:ta:

     ver: avp1p: hello-world
```

We discussed previously that deployment is layered above the Replication Controller / ReplicaSets. Here it is replica set.ReplicaSet Object is the specification for the Replica Sets. It indicates numbers of replicas. That means there should be 10 replicas of Replica Sets object. It will select the deployment by a selector which will use matchLabelsto match with pods.

ReplicaSets will create pod-template-hash and recognize pods with Pod labels. These are given in template. That means Labels are used to group the pods. And selectors have used to group right pods according to a label.

Pod Object Specification:

deployment.yml

```
 spc- oenc:tmaien:ehresl:lo-world

imaginmeaauglelP: okluicbye:jaAclkw/ahyeslloworld:latest

          p- oPrtst:ainerPort: 3000

               con
```

The Pod Object Specification wraps one or multiple containers. It's always the best practice to wrap one container within one pod. Multiple container's per pod will create the more complexity in terms of network and storage namespace.

The Pod specification will consist the container specification. It means this specification will give information about image name, container name, container ports to be exposed and what will be pull policy for the image.

imagePullPolicy indicates when the image should get pulled and can be "Always" to make sure updated image is pulled each time.

To group pods and load balance between them Kubernetes internal service discovery mechanism is used.

service.yml

```
akmpineiVde: rSseirovni:cve1

  nlaatmadea:thae: llo-world-svc

   s  nbacemlse:: hello-world-svc

   p-opr:tosr:t: 80


targpertoPtorcto:l3: 0T0C0P

    sealpepc:tohre:llo-world

     ver: v1
```

Services are important to proxy the replicas of ReplicaSets. In above service.yml file apiVersion is used which indicates the Kubernetes API version to be used. Here its v1. It was not same in deployment because v1 is not supporting deployment object. Again the labels and metadata for the services are used to uniquely identify the service within Kubernetes cluster.

In the spec, it's possible to expose the port of the node. The exposed port will be mapped to the container port. Here targetPort is the port of the container which is mapped with node port 80.TCP and UDP protocols are supported.

This service is not exposed publicly. To expose it with cloud provider type: Loadbalanceris used.

The most important is Selector. It will select the group of pods which will match all labels. Even if any pod consist more than, specified labels then it should match. It is not the same case with the selector, if there is any label missed in selector but not in the pod then service will just ignore it.
Labels and Selectors are good ways to maintain version and to

rollback and rollout the updates.

Understanding the Kubectl Utility

In previous sections, we have discussed the master components. The API Server plays the important role of the master.

Everything can be done using API Server only. There are different options for calling REST API's, the User interface of the Kubernetes and Kubectl.

Kubectl is the command line utility to interact with Kubernetes cluster. We created the Kubernetes Cluster through Minikube. Kubectlconfig is used to configure Kubectl for Kubernetes Cluster. To verify the Kubectl is connected to API Server:

```
$apciaVte~r/s.ikounb:ev/1config
c- lculustsetresr::c  i        ate-authority:

 /home/ut/.minikube/ca.crt nam

        inikube

   clustetr: minikubeuser:     e
name: minikubecurrent-
context: minikube kind:
Configpr  ferences: {}

users:- name: minikube



   client-key: /home/ut/.minikube/client.key
```

Kubectl is authenticated to interact with Kubernetes Cluster. Kubectl can manage everything for Kubernetes Cluster.

Kubectl can,
Create the Kubernetes Object:

```
$kubectl create -f deployment.yml
```

Apply the changes to the Kubernetes Object:

```
$kubectl apply -f deployment.yml
```

Get the deployments

```
$kubectl get deployments
```

Get the pods:

```
$kubectl get pods
```

Get the namespaces

```
$kubectl get namespaces
```

Get the replica sets:

```
$kubectl get rs
```

Describe the Kubernetes Object:

```
$kubectl describe deployment hello-world
```

Delete the Kubernetes Object

```
$kubectl delete -f deployment.yml
```

To get the all pods from namespace kube-system

```
$kubectl get po --namespace kube-system
```

All these are few examples of the Kubectl utility. But there are more advanced use cases like scaling with kubectl. All these are imperative methods.

Imperative methods are good at the time of troubleshooting or in the development environment. It’s always the best practice to have declarative methods for creating YAML files.

Launching and Running Container pods with Kubernetes

To create the kubernetes object Kubectl cli will be used for the communication with API Server.

Create deployment object:

```
$kubectl create -f deployment.yml
```

Create service object:

```
$kubectl create -f service.yml
```

It will create the 10 Pods, 1 ReplicaSet and 1 Deployment object in the default namespace.

Get deployment:

```
$NkAuMbeEctl get DdEepSIloRyED   CURRENT h-10
                                          1U0P-TO-DATE  1A0VAILABLE 1AhGE
 weolrlold       10
```

Get replica set:

```
kNuAbMecEtl get rs
                  DESIRED   CURRENTUDAPT-TEO-      AVAILABLEAGE

 hweolrlold--49362160110      10          10          10           1h
```

493621601 is the hash value added over deployment

Get pods:

```
$NkAuMbeEctl get po
                                              READYSTATUSRESTARTSAGE

   hello-world-493621601-2djnqj3wc   1/1          Running0             1h
                                                  Running0             1h 1h 1h 1h
     hello-world-493621601-g35jsjtfl1/1           Running0
                                                  Running0
                                                  Running0
   hello-world-493621601-rb5q7pgr9   1/1

  hello-world-493621601-xvk08h6hds  1/1

      hello-world-493621601-ztp321/1
```

Again the more hash values are added over the replica set. The status indicates all pods are in running state. That means desired state of the cluster is met.

Get the service object and describe it:

```
$NkAuMbeEctl get svc
khuelbloer-wneotrelds -svc    CLUSTER-IP   EIPXTERNAL-PORT(S)          AGE

                              10.0.0.131      <npeone>ing>  48403:3/0T9C1P2/TCP21h
```

Describe service:

```
$kaumbe:ctl describe svchhelellolo-w-woorlrdld-s-vscvc Ned  f

 Laabmelss:pace:           neamaue=lthello-world-svc

   Annotations:               kubectl.kubernetes.io/last-applied-
```

```
ctaotniofingsu"r:{a}t,"iolanb=e{l"sa"p:{i"Vnearmsioe"n:"h"evl1l"o,"-kwionrdl"d:"-
sSvecr"v}i,"cnea","mmee"t:"ahdealtlao"-:w{"oarnlndo-

sSvecle",  espacea"p:"pd=ehfaeulllot-"w},"

c"ntoarm
                osr..l.d,ver=v1

:
                10.0.0L.o13a1dBalancer

TIPy:pe:

PNoordte: Port: <unset>    8309T1C2/PTCP
E0 n+d7points:         172.17.0.10:3000,172.17.0.12:3000,172.17.0.2:300
mSeos
rseio..n
.
          Affinity:       None

Events:                   <none>
```

Endpoints are unique IP's of the Pods. Services group all Pods with the IP's. Kubernetes Services plays the role of Proxy.

Exposing the Service locally through Minikube:

To access the app it's necessary to get the IP and port from the Minikube.

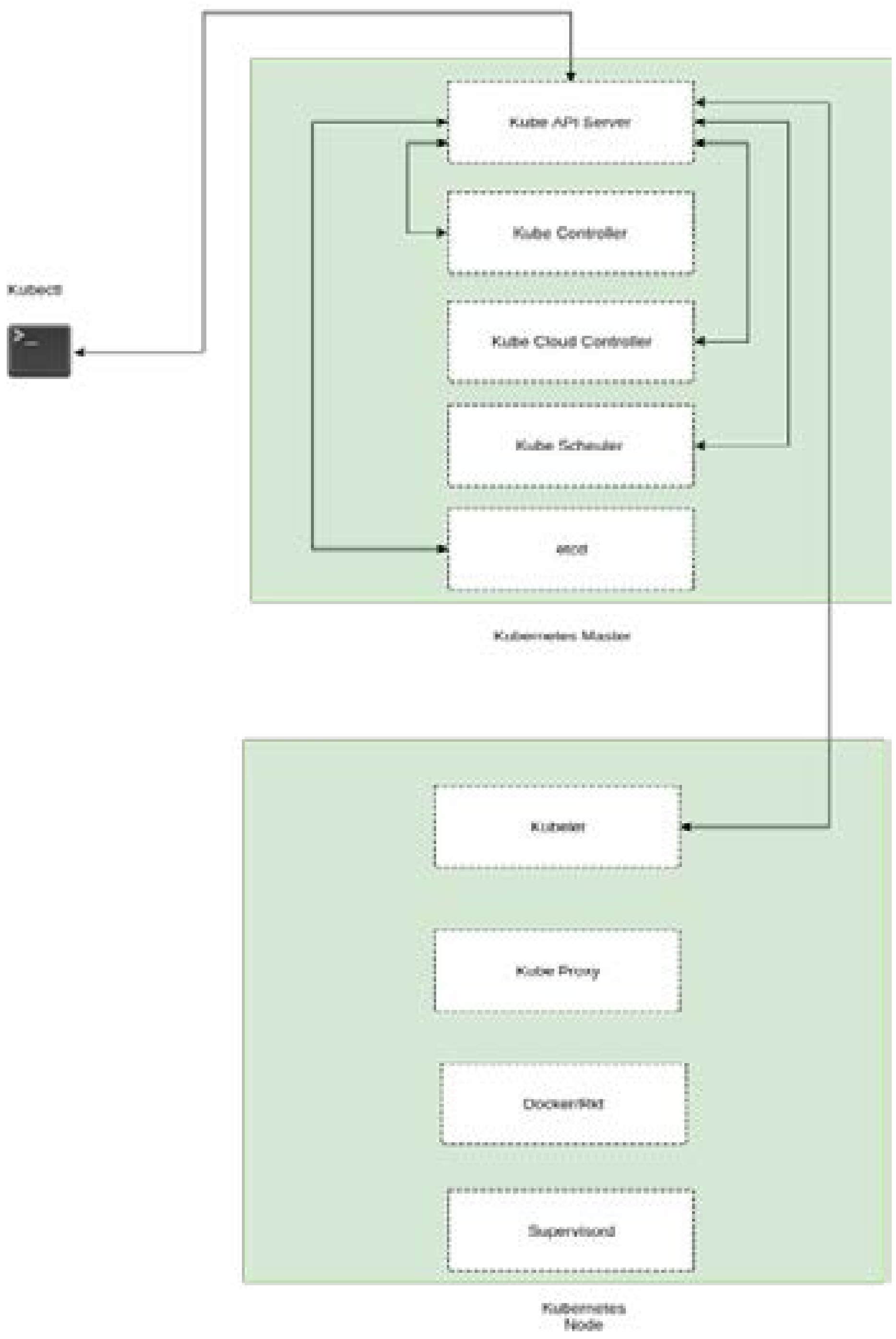
```
$minikube service hello-world-svc
```

It will open the app into default browser.

Understanding the Kubernetes App Flow:

To understand the process in the background when Kubectl creates objects let's look into the master-node communication of Kubernetes.

The following diagram indicates the communication between kubectl, master components, and node components. Here API Server is the responsible component of communication. All the information on Node is associated with kubelet.



When

```
$kubectl create -f deployment.yml
```

Kubectl create command is used for creating Kubernetes object with specification file. Here the two-sided arrows indicate the bi-directional communication between components. Kubelet will inform the Node information to API Server which will be stored in the distributed key-value pair(etcd).The important task of scheduling is done by Kube-scheduler which uses the advanced algorithm to schedule the Kubernetes object on right Node with the help of kube controller manager.

The API Server communicates with Kubelet. If the connection is broken then that node is treated as unhealthy. It will help and maintain the desired state of the cluster by load balancing and autoscaling.

Kubernetes Auto Scaling:

It’s possible to scale up and scale down the replicas of the Kubernetes deployment object. To scale up by 20 replicas.

Scaling up:
deployment.yml

```
akpiniVde: rDseiopnlo:

yemxtentsions/v1beta1me

abpepl:shello-worldver:



selelcictor:matc   Labe s:



 template:metadata:
   labels:app: hello-world



   containers:

    image: kubejack/helloworld:latest



    - containerPort: 3000
```

Update the deployment with,

```
kkaurbneinctgl:akpuapblyec-ftsldaveppp-lloyysmheonutl.dymbuel usetldaopnplryesource created by ei
duepbleocytml cernet t"ehe--lloa-we ocrolnd"ficgoonrfkgubred
```

```
$NkAuMbeEctl        get
pods
                                 world-493621601-

h1ge7ll6os-world-
                            1R/E1ADY  RSTuAnTnUinSg          0RESTARTS
493621601-  h2deqll3oc-               2AhGE

world-493621601-            1/1       Running   0            2h

h2qedllgo3-world-           1/1       Running   0            38s

493621601-   h8jenljlwo-    1/1       Running   0            2h
hcwelmlo1-wj  orld-
493621601-                  1/1       Running   0            2h

hg5esllflo-world-
493621601-     hj3ejjltlo-  1/1       Running   0            2h
world-493621601-
hj3erlrljo-world-           1/1       Running   0            38s
493621601-   hkgelj7loz-
world-493621601-            1/1       Running   0            38s
hlqevlklo4-world-
493621601-   hmerlklotj-    1/1       Running   0            38s
world-493621601-
hnefdll5od-world-           1/1       Running   0            38s
493621601-
                            1/1       Running   0            38s

                            1/1       Running   0            38s
```

```
hpjeblldon-world-
493621601-   hq6exlllog-      1/1        Running   0          38s
world-493621601-
hrbe7llgor-world-             1/1        Running   0          38s
493621601-   hs5eqllpo9-
world-493621601-              1/1        Running   0          2h
hshell7lov-world-
493621601-   hv0e8ll6od-      1/1        Running   0          2hs
world-493621601-
hxkehllho-sworld-             1/1        Running   0          38s
493621601-   hztepl3lo2-
world-493621601-              1/1        Running   0          2hs

                              1/1        Running   0          2hs

                              1/1        Running   0          2hs
```

It will scale up the replicas to the 20.

Scaling down:

To scale down the application change replicas to the 5, deployment.yml

```
akmpineiVde: rDseiopnlo: yemxtentsions/v1beta1

 nlaatmadea::thae: llo-world

  vearbp: epl:shello-world

  spr epcv: 1 as   5

     selelcictor:

     matchLabels:
```

```
vera: pvp1: hello-
world
temmeptaladtaet:a:
laabpepl:s:hello-
world vesrp: evc1:

   c- onnatmaien:ehresl:lo-world
imagimePauglelP:
    okluibcye:jaAclkw/ahyeslloworld:l
    atest p- coortnst:ainerPort: 3000
```

Update the deployment:

```
kduepbleocytml aepnptly"h-ef ldloe-pwlooyrmlde" ncot.nymfiglured

    $kubectl get pods
```

```
NheAlMloE-world-          R1/E1ADY   SRTuAnTnUinSg          R0 ESTARTS
                                              A2hGE
493621601-

he7llos-world-
493621601-  2hdeqll3oc-
world-493621601-          1/1       Terminating   0          2h
2hqedllgo3-world-
493621601-   8hjenljlwo-  1/1       Terminating   0          2m
world-493621601-
                          1/1       Terminating   0          2h

                          1/1       Terminating   0          2m
cwm1j
```

```
hg5esllflo-world-
493621601-     hj3erlrljo-
world-493621601-
hkgelj7loz-world-
493621601-   hlqevlklo4-
world-493621601-
hmerlklotj-world-
493621601-   hnefdll5od-
world-493621601-
hpjeblldon-world-
493621601-   hq6exlllog-
world-493621601-
hrbe7llgor-world-
493621601-   hs5eqllpo9-
world-493621601-
hshell7lov-world-
493621601-   hv0e8ll6od-
world-493621601-
hxkehllho-sworld-
493621601-   hztepl3lo2-
world-493621601-

1/1      Terminating    0          2h
1/1      Terminating    0          2h
1/1      Terminating    0          2m
1/1      Terminating    0          2m
1/1      Terminating    0          2m
1/1      Terminating    0          2m
1/1      Terminating    0          2m
1/1      Terminating    0          2m
1/1      Running        0          2h
1/1      Running        0          2h
1/1      Terminating    0          2m
1/1      Terminating    0          2h
1/1      Terminating    0          2h
1/1      Running        0          2h
```

It will scale down the Kubernetes deployment.

Destroying Kubernetes Cluster and Pods:

Delete Kubernetes Objects:

To destroy the Kubernetes Application:

```
keupbloeycmtlednetle"hteel-lfod-ewpolrolydm" deenlte.ytemdl

$sekruvbiceect"lhdeellot-ew-ofrslder-svvic" .dyemlelted
```

It will delete the Kubernetes Pods, Deployment, ReplicaSets and Services which was created.

Stopping Minikube and Deleting cluster:

```
$SmtopinpiiknugbleocstaolpKubernetes cluster...

    Machine stopped.
```

```
$DmelientinugbloecdaellKetuebernetes cluster...

    Machine deleted.
```

6. Deploying Kubernetes with Ansible

Ansible is the configuration management tool used for the deploying the different types of the infrastructure. It is possible to deploy the Kubernetes with Ansible. Kubespray is a tool used for deploying the Kubernetes cluster with Ansible.

Kubespray can deploy the cluster on AWS, GCE, Azure, OpenStack or Bare Metal. It makes sure the higher availability cluster and composable support with container networking. Kubernetes cluster can be deployed over popular Linux distributions. Even If any CI tests are available then it's possible to run over Kubernetes Cluster created by Kubespray.

Before using Kubespray it's necessary to have following requirements in place:

1. **Ansible v2.4 (or newer) and python-netaddr is installed on the machine that will run Ansible commands**
2. **Jinja 2.9 (or newer) is required to run the Ansible Playbooks**
3. **The target servers must have access to the Internet in order to pull docker images.**
4. **The target servers are configured to allow IPv4 forwarding.**
5. **Your ssh key must be copied to all the servers part of your inventory.**
6. **The firewalls are not managed, you'll need to implement your own rules the way you used to. in order to avoid any issue during deployment, you should disable your firewall.**

Kubespray provides the Terraform and kubespray-cli to provision the environment.

Pre- Requisites for Kubespray

Ansible v2.3 (or newer)

Execute below commands to install the latest ansible on RPM based distributions.

```
$ sudo yum install eapnesilb-rleelease
```

Execute below commands to install latest ansible on debian based distributions.

```
$ sudo apt-gdetdui-nrpsdtaoltlesoftywpaprea-:panrospibelret/iaenss-cibolme mon

$ sudo apt-get upedpatseitorle

                install ansib
```

Jinja 2.9 (or newer)

Execute below commands to install Jinja 2.9 or upgrade existing Jinja to version 2.9

```
$ epaipsy2_iinnssttaallllljpinipja2 --upgrade
```

Allow IPv4 forwarding

You can check IPv4 forwarding is enabled or disabled by executing below command.

```
$ sudo sysctl net.ipv4.ip_forward
```

If the value is 0 then, IPv4 forwarding is disabled. Execute below command to enable it.

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

Your machine ssh key must be copied to all the servers part of your inventory.The firewalls should not be managed and The target servers must have access to the Internet.

For SSH Authentication between machines. On the source machine,

Step 1: Generate Keys

```
$ ssh-keygen -t rsa

GEnenteerrfiatlienignpwuhbilcich/ptorisvaavte trhsae key
p(/ahior.me/user4/.ssh/id_rsa): CEnreteartepdadssirpehcrtaosrey
('/ehmopmtey/fuosrenr4o/.pssahs's.phrase):
EYonuterridsaemnteifipcaastsiopnhrhaasse baegeanins:aved in
/home/user4/.ssh/id_rsa. YTohuerkpeuybfilnicgkeerpyrhinast ibse:
en saved in /home/user4/.ssh/id_rsa.pub.
aTdh:e1ek:e1y4's:ar5a:ncdo7m7:a2r5t:2im9:a9gf:e75is::ee:4f:a4:8f:f5:
65 user4@server1
+| --[ RS.A.2..0| 48]         +
|       o=o oo.o+|.|
|       So ..o+ o| |

|      .o.   . *E+||
|      ...    . +| |

+-----------------+
```

Note that your key pair is id_rsa and id_rsa.pub files in shown directories. Your id_rsa is the private key which will reside on the source machine. id_ rsa.pub is the public key which resides on the destination machine. When SSH attempt is made from source to destination, protocol checks these both keys from source and destination. If they match then the connection will be established without asking password.

Step 2: Copy Keys

Now, we need to copy id_rsa.pub key on the destination machine. It should be copied to a home directory of the intended user in the destination server. It should reside under ~/.ssh/ (i.e. home directory/.ssh/) and with name authorized_keys. You can copy the file using shell or any other file transfer program.

If you are trying from source machine using ssh then use below commands:

```
$ ssh     er4@10.10.4.12 "mkdir ~/.ssh"

RSA key fingerprint is 08:6c:51:09:9f:4c:69:34:84: f:08:af:68:df:5e:24.

Warning: Permanently added '10.10.4.12' (RSA) to the list of known

user4@10.10.4.12's password:

$ cat .ssh/id_rsa.pub | ssh user4@10.10.4.12 'cat >> .ssh/authorized_

user4@10.10.4.12's password:

$ ssh user4@10.10.4.12 "chmod 700 .ssh; chmod 640 .ssh/authorized_

user4@10.10.4.12's password:
```

Here, the first command creates .ssh directory on the destination machine. Second command copies id_rs.pub file's content to destination machine under file ~/.ssh/authorized_keys and last command set proper permissions.

Repeat the copying steps for each Node of the Kubernetes Cluster.

Step 3: Test the Connection

```
$L[ assthlougseinr:4@Tvu1e0.O1c0t.46.1221:59:00 2015 from 10.10.4.11

    user4@ser er2 ~]$
```

Kubespray CLI installation

You can also use kubespray without CLI by directory cloning its git repository. We will use it using CLI. Execute below step to install kubespray.

```
$ pip2 install kubespray
```

You can check the version of kubespray after successful completion of installation

```
$ kubespray -v
```

Inventory File setup

Create new inventory file at ~/.kubespray/inventory/inventory.cfg and Add the contents as shown below.

```
    $ vi ~/.kubespray/inventory/inventory.cfg

mgeancphrionxey-:0810a8n0sible_ssh_host=192.168.0.144 http_proxy=http://

mgeancphrionxey-:0820a8n0sible_ssh_host=192.168.0.145 http_proxy=http://

mgeancphrionxey-:0830a8n0sible_ssh_host=192.168.0.146 http_proxy=http://
```

```
    [mkuacbhei-nmea-0st1er]

machine-02
[metaccdh]ine-0 machine-03



     [mkuacbhei-neo-d0e2]

          machine-03

      [kku8bse-c-nluosdter:children]

          kube-master
```

Here the 3 Nodes of the server with the proxy.Let's start the cluster deployment.

Here Inventory is the term specifically related to Ansible. Ansible works with different systems in the infrastructure. Ansible communicates over ssh using the configuration file called as Inventory. Inventory is a configuration file with the .cfg extension which lists the information about the systems.
Look following information about the Kubernetes Nodes is stored inside inventory.

Kubernetes Cluster Deployment Using Kubespray

Before, starting actual Deployment, Let's see what will be going behind the scenes and how painful manual installation task is executed smoothly.
Kubespray will install kubernetes-api-server, etcd (key-value store), controller, Scheduler will be installed on master machines and kubelet, kube-proxy and Docker (or rkt) will be installed on node machines.
These all components will be installed and configured by ansible roles in kubespray. All, We need to do is to execute one command.

To start deployment of the kubernetes cluster, execute following commands.

```
$ kubespray deploy
```

Based on the number of master and minions, It will take time to deploy the complete cluster. At the end of execution, you will get output something like shown below. If there are no failed task, Your deployment is successful.

```
P19L2A.Y16R8E.0C.1A4P4***:o**k*=*2*7**8***c*h**a*n*g**e*d*=*8*9*****u**n*r*e*a

          l1o9c2a.1lh6o8s.0t.1465:ok=324867changed=71803unreachable=

                              ed=1
```

To check that Everything went good and deployment was successful, you can login to master node and get all the worker node.

```
$mkaucbheincetl-0g2et nodesReady
                                      4m
   machine-03          Ready          4m
```

List pods in all namespaces by executing below command.

```
$NAkuMbEeScPtlAgCeEt poNdAsM--aEll-namespaRcEeAsDY                    STATUS
                                          RESTARTS                    AGE

                                  1/1     Running   0                 5m
kube-system
                                  1/1     Running   0                 5m
d7ynks3mnasq- kube-

system

d5vnfshm0jasq-

kube-system     flmaancnheinl-e-02        2/2         Running         0

                4m kube-system            flmaancnheinl-e-03          2/2

                Running           0       4m
kube-system     kapuibserver-     1/1     Running   0                 5m


kube-system
                mkuabceh-
                ine-01            1/1     Running   0                 5m
                cmoanntrao
kube-system     glelre-r-
                mkuabceh-
kube-system     ipnreo-x0y1-      1/1     Running   0                 4m
                mkuabceh-
kube-system     ipnreo-x0y2-      1/1     Running   0                 4m
                mkuabceh-
                ine-03
                                  1/1     Running   0                 5m
                smcahcehdi

                unle-r0-2

kube-system
kpu8mbekd7ns-                     3/3     Running   0                 4m
 kube-system    mnagcihnixn-                        ep-r0o2xy-
```

```
1/1          Running   0                          2m

        kube-system     nmgaicnhxi-
        npero-0x3y-                    1/1          Running   0                 2m
```

This will install kubernetes using Ansible.

7. Provisioning Storage in Kubernetes

Kubernetes Persistent Volumes

Requesting storage
Using Claim as a Volume
Kubernetes and NFS
Kubernetes and iSCSI

Provisioning Storage with Kubernetes

Storage plays the important role of storing the data. Kubernetes consists pods which are ephemeral. The important aspect of the having persistent storage is to maintain the state of the application. The most useful method is to have persistent storage attached with containers. The state of the database will be maintained with the Persistent Storage. Kubernetes provides the PersistentVolume object.

Kubernetes currently supports the following plugins:

GCEPersistentDisk
AWSElasticBlockStore
AzureFile
AzureDisk
FC (Fibre Channel)
FlexVolume
Flocker
NFS
iSCSI
RBD (Ceph Block Device)
CephFS
Cinder (OpenStack block storage)
Glusterfs
VsphereVolume
Quobyte Volumes
VMware Photon
Portworx Volumes
ScaleIO Volumes
StorageOS

Kubernetes Persistent Volumes

Kubernetes provides PersistentVolume API for users and administrators. It abstracts the details about how the storage is provided for the other Kubernetes objects. There are two new API are introduced to do so:

1. **PersistentVolume (PV)**
2. **PersistentVolumeClaim (PVC)**

A PersistentVolume (PV) **is a piece of the storage from the cluster. This storage could be provisioned by the administrator. This API object is responsible for implementation of the storage like NFS, iSCSI or cloud- specific storage. The PersistentVolume is cluster resource similar to the node.**

A PersistentVolumeClaim (PVC) **is a request made by a user for storage that means PersistentVolume. If we correlate it to the pod, pod consumes the node resources. Similarly, the PersistentVolumeClaim consumes PV resources. Claims can request the specific size and access modes from PersistentVolume.**

A StorageClass gives the abstraction for the PersistentVolume. The implementation details are hidden with StorageClass. It also provides the quality of service levels or backup policies with PersistentVolume. It is described as "Class" of the storage.

Lifecycle of the Volume

The interaction between the Persistent Volume and Persistent Volume Claim consists the lifecycle.

This lifecycle consists several phases:
1. Provisioning
2. Binding
3.Using
4.Reclaiming

1. Provisioning :

There are two types of the Persistent Volume provisioning methods. Those

are Static and Dynamic.In the static method, the administrator creates the PV's. But if PV's does not match with users PVC (Persistent Volume Claim) then the dynamic method is used. The cluster will try to generate the PV dynamically for the PVC.

2. Binding:

A control loop on the master watches the PVC's and binds matched PV with PVC. If no PV found matching PVC then the PVC will remain unbounded.

3. Using:

Pods use the Claim as the Volume. Once the PV matches with required PVC, the cluster inspects the claim to find the bound volume and mounts the volume for a pod.

4.Reclaiming

The reclaim policy decides what to do with the PersistentVolume once it has been released. Currently, volumes can be Retained, Recycled or Deleted.

Retain **policy consists the reclamation of the resource. If the Persistent Volume Claim is deleted and Persistent Volume is released, still Persistent Volume cant be used for other PVC. The reason behind it that Persistent Volume may contain the data. The administrator can manually reclaim the volume:**

a. Delete the PersistentVolume.
b. Manually clean up the data.
c. Manually delete the storage assets.

In Recycling **the volume again available for new claim. The Recycling performs the basic scrub on the volume if it is supported.**

The Deleting **will delete the PersistentVolume object from the Kubernetes Cluster. It will also delete the associated external infrastructure Volumes like AWS EBS, GCE etc.**

Requesting Storage

To request the storage that means PersistentVolume (PV) the PersistentVolumeClaim is used. Here PersistentVolumeClaim is API provided for the user to request PersistentVolume. Similar to Pod, PersistentVolumeClain (PVC) also consists the specification in YAML to request the resource.

```
kapiniVde:
rPseiorsni:stve1ntVolumeClai
m mneatmadea:tma:yclaim
spacece: ssModes:
  re-
  sRoeuardcWesr:iteOnce
  restqoureasgtes:: 8Gi
  seoleractgoerC:lassNam
  e: slow
  mrealtecahsLea:b"setlas:
  ble"

    m- a{ktcehyE: xenpvreirsosniomnse:nt, operator: In, values: [dev]}
```

Kind represents the Kubernetes Object. PersistentVolumeClain (PVC) is available with v1 apiVersion.

Metadata **is the key-value pair representing the metadata for** PersistentVolumeClaim **object. The specification consists the access modes which is ReadWriteOnce. The access mode is important while accessing the PV resource with given access mode. resources consist the requesting of the storage in Gi. The** storageClassName **represents the classes in the PersistentVolume.**

The selector **can consist of two fields:**

1. matchLabels **- the volume must have a label with this value**
2. matchExpressions **- a list of requirements made by specifying key, list of values, and operator that relates the key and values. Valid operators include In, NotIn, Exists, and DoesNotExist.**

Access Modes for Persistent Storage

A Persistent Volume can be mounted on the host. The access modes play the important role for PV's. Its possible to handle the Persistent Volume with different Access Modes.
The access modes are:

- 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**

In the CLI, the access modes are abbreviated to:

- RWO - **ReadWriteOnce**
- ROX - **ReadOnlyMany**
- RWX - **ReadWriteMany**
-

Using Claim As Volume

Pods are ephemeral and require the storage. Pod uses the claim for the volume. The claim must be in the namespace of the Pod. PersistentVolume is used for the pod which is backing the claim. The volume is mounted to the host and then into the Pod.

```
kapiniVde: rPsoiodn:
v1
mneatmadea:tpar:odu
ction-pv
spcoenc:tainers:

  - inmaamgee:: fdrocnkterfidle/nginx
    v- omluomunetMPaotuhn:
 t"s/:var/www/html" volunmamese:: pv

  - pnearmsies:tepnvtVolumeClaim:
```

This is the example of sample pod where volumes and persistentVolumeClaim is used with the claimName myclaim. This is the way pod will use the persistent volume. That means claim as a volume.

Kubernetes and NFS

Kubernetes PersistentVolume can use external volume for storage. nfs volume allows mounting NFS (Network file system) to be mounted into the pod. NFS can be mounted with multiple writers simultaneously. NFS server can be used for provisioning the volume.

Below is the example of NFS volume as Persistent Volume and Persistent Volume Claim:

PersistentVolume:

```
akmpineiVde: rPseiorasni:stve1ntVolume

  spnatmadea:tnf:s

  csaepc:arcaigtey:: 1Mi

 a-ctcoeesasdMWorditeesM:  any

    n#fsR: IXME: use the right IP

     seFrver: 10.244.1.4

     path: "/"
```

The server is pointing over the NFS

server. PersistentVolumeClaim:

```
akmpineiVde: rPseiorasni:stve1ntVolumeClaim

  spnatmadea:tnf:s

     acece: ssModes:

sret-oRroeaagdecWCelraistseNMaamnye: ""

      resquestss::

        storage: 1Mi
```

Kubernetes and iSCSI

SCSI and IP (iSCSI) can be mounted with Kubernetes using an iscsi volume. When the pod dies, the volume preserves the data and the volume is merely unmounted. The feature of iSCSI is that it can be mounted read- only by multiple consumers simultaneously. iSCSI does not allow write with simultaneous users.

The simple example of the iSCSI volume:

```
-a-p- iVersion:

  v1 name:

  iscsipd

  - onnamaien:eisrsc:sipd-

  rwima  : k

  bernetes/pause namuen:

  iPsacsipd-rwvolu        :

      tarig:etPortal: 10.0.2.15:3260po   als: ['1       0.2.16:3260', '10.0.2.17:3260']

      fsType: ext4
      readOnly: true
```

8. Troubleshooting Kubernetes and Systemd Services

g Commands Networking Constraints
ubernetes Querying the State of Kubernetes Checking Kubernetes yaml or json Files Deleting Kubernetes Co

Kubernetes Troubleshooting Commands

Kubectl utility is used for communicating with cluster through command line environment. It is necessary to debug the Kubernetes services running on master and nodes. Also, the Kubernetes Cluster itself to ensure higher availability of the application. Here few are Kubernetes troubleshooting commands which will help you with troubleshooting Kubernetes.

Getting Information about Pods:

```
Nk AkuMebEectl get po --namespace kRuEbAeD-sYystemSRTuAnTnUinSgR0 ESTARTSA35GmE

   muibnik-audbdeon-manager-          1/1

kuberdnneste-9s1-d0a3s3h0b66o2a-rddp- 7xt32/2
                                                       Running0              35m
   1pc46
```

It will give the status of each pod. Also mentioning the namespace is al- ways the best practice. If you don't mention the namespace, then it will list the pods from the default namespace.

To get the detailed information about the pods:

```
$kukbueb-escytsltedmescribe pods kube-dns-910330662-dp7xt
--namespace Name:spkaucbee:-kdunbs-e9-1s0y3st3e0m662-dp7xt
NStoadrte:Tmimine:ikubeS/u1n92, .
11268N.o9v9.210107 17:15:19 +0530 Labels:
                        kp8osd--atpemp=pkluatbee--
hdanssh=910330662

Ae"n,"napotiVateirosnios:n":"kvu1b",e"rrenfeetreesn.icoe/"c:r{e"katined-"b:"yR=e{"

pklicnadS"e:"tS",e"nriaamlizeesdpRaceefe"r:"eknuc
ba6ea-s9y-s0t8e0m02",7"n64a6m..e.":"kube-dns-910330662","uid":"f5509e26-
c79e-11e7-

SIPta:
tus:
                        sRcuhnendiunlger.alpha.kubernetes.io/critic
                                                        al-pod=

Created By1:72.1R7.e0p.3licaSet/kube-dns-
910330662 ContaoinlleerdsBy:  ReplicaSet/kube-
dns-910330662

        kubedCnosn:tainer ID:
5dcodc4kce8r:1/9/7465d2eb2b5156ec2a4e2a7f385d700eb96c91874137495b9d715
ddf-

Iamdg6e4::1.            gcr.io/google_containers/k8s-dns-kube-dns-
14.4

Image ID:               docker://sha256:a8e00546bcf3fc9ae1f33302c16a6d4c-

Ports:                  10053/UDP, 10053/TCP, 10055/TCP

          main=cluster.local.

       config-map=kube-dns

State:                  Running
```

```
RSetardtye:d:     STurune, 12 Nov 2017 17:15:20 +0530

ReLsitmarittsC:o

unt:  0

mReeqmueosrtys

:: 170Mi
cpu
m:
       e1m00omry:      70Mi

tLjimveenoeusts=:5shpttepr-igoedt=h1t0tps
:#/s:u1c0c0e5s4s/=h1ea#lftahicluhreec=k5/kubedns delay=60s

                    #suchctetsps-=g1et#hfattiplu:/r/e:8=0381/readiness delay=3s
Rperaidoid           timeout=5s

n=e1s0ss:

   EPnRvOirMon
   EmTeHnEtU:

                           S_PORT: 10055

   M/koubnets-d: ns-config from kube-dns-config (rw)
vdlg/vsa(rr/or)un/secrets/kubernetes.io/serviceaccount from default-
token- dCnosmntainqe: r ID:
c1a8ad0o8c3kee7r8:9//b8fd3b9fa37c931a8074e566875e04b66055b1a96dcb4f192ac
b-

Iamdg6e4::1.      gcr.io/google_containers/k8s-dns-dnsmasq-nanny-
14.4

Idmocakgeerl:/D/s:ha256:f7f45b9cb733af946532240cf7e6cde1278b687c
d7094cf- 0P4o3rbts7:68c800cfd5a3fd/UDP, 53/TCP
Arg-sv:=2
    -lcoogntfiosgtDdierr=r/etc/k8s/dns/dnsmasq-nanny
    -kcache-size=1000
    --log-facilictlyu=s-ter.local/127.0.0.1#10053
    --server=/ipn6-
 a.adrdpra.a/r1p2a7/.01.207..10#.01.010#5130053
 Startete: d:        RSuunn,n1i2ngNov 2017 17:15:20
 +0530 Reatdayr:t Count:  T0rue

cpum:

       emor1y5:          20Mi

       0m

 Ltiimveenoeusts=:5s                                    perihotdt=p1-
   g0est#hstutcpc:/e/s:1s0=015#4f/ahieluarlteh=c5heck/dnsmasq
   delay=60s EMnovuirnotsn:ment:        <none>
    /evtacr//kru8sn//dsnecs/rdentss/mkuabsqer-
 nentensy.ifor/osemrvkiucebaec-dconus-nctofnrfiomg    (drwef)ault-token-
 vsdildgesc(arro:)
 doCcoknerta:/i/nsehraI2D5:6:38bac66034a6217abfd44b4a8
 a763b1a4c-
 97I3m04a5gcea:e2763f2gcr.8io5/7gboaoag5lce9_aco87n2tai
 ners/k8s-dns-sidecar- aImadg6e4:I1D.:14.4
 97304d5occakee2r7:6/3sfh2ac2c5865:73b8abaa5cc696a083742a6217abfd44b4a8a7
 63b1a4c-
 PoArtr:gs:  10054/TCP

    --vlo=g2tostderr
```

```
pro-b-
e=kubedns,127.0.0.1:10053,kubernetes.default.svc.cluster.
loca--l.,5,A
pStraotbee:
=dnsmasqR,1u2n7n.0in.0g.1:53,kubernetes.default.svc.cluster.local.,5,
A SRteaardteyd: : STurune, 12 Nov 2017 17:15:19 +0530
ReqtuaersttCs:ount:  0
cmpeum:
                    10m
        ory:            20Mi

timLievoeunte=s5ss: period=10hstt#ps-ugcect

ehststst=p1://#:1fa0i0lu54re/m=5etrics delay=60s

EnMvioruontms:ent:    <none>
vdlg/vsa(rr/or)un/secrets/kubernetes.io/serviceaccount from default-token-
CToynpdeiti
ons:
                    Status

 IRneiatidaylizedTrue
True VPooludmScehse:
duled

True

kuTbyep-de:ns-cCoonnfifigg:Map (a volume populated by a ConfigMap)

NOpamti  ktruubee-dns

oen: al:

 dTeyfapuel:t-toSkeecnr-evtd(lagsv:olume populated by a Secret)


QoS Class:Burstable
```

```
NTooldeera-Stieolencst:ors:Critic<anlAodndeo>nsOnly
EFivresntStse:en  LastSeen   Count                    From SubObjectPath

                  ---------   -----          ---------------

        ---------40m
40m                 ------  1     defau-l-t---s--c-heduler

       N91o0r3m30a6l 62-                Successfully assigned kube-dns-
  dp7xStcthoemduinleikdube

                           4N0omrmal      40m

Succe1ssfulkMuobuenletVt,omluimniekuMbeountVolume.SetUp

s4u0mcceeded4f0omr volum1e “kubkue-bdenles-tc, omnifingik”ube
sNuocrcmeeadl ed for voSulucmceess“dfuelfMauolut-ntotVkoelnu-
mvdelgMs”ountVolume.SetUp

4co0mntainers4{s0i   1     kubelet, minikube    spec.
mdecar}

Ngooorgmlea_lcontainePrus/llke8ds-dns-sidecar-
aCmodn6t4a:i1n.e1r4.i4m”aaglere“agdcyr.ipor/esent on
m40amc
hine
          40m        1     kubelet, minikube    spec.

cNoonrtmaianlers{sidecCarre}ated      Created container

c4o0nmtainers{side          1     kubelet, minikube    spec.
c4a0rm}
N40omr
mal                Starte1
          40m      d           kubelet, mSintairktuebdeconsptaeicn.er

cNoonrtmaianlers{kubePdunllse}d      Container image “gcr.io/google_

cmoanct               haine
```

s/k8s-dns-kube-dns-amd64:1.14.4” already present on

```
c4o0mntainers4{k0mubedns}1                spec.kubelet, minikube

    N40omr
    mal              Creat1e
                     d
              40m                 kubelet, mCinreikautebde cosnpteacin. er

    cNoonrtmaianlers{kubeSdtanrst}ed      Started container


    c4o0nmtainers{dnsm                1      kubelet, minikube    spec.
    40amsq}


    N"gocrr.mioa/lgoogle_cPonultlaeidners/k8s-dns-dnCsomnatasqin-

    neranimnya-gaemd64:1.14.4" a40lrmeady pre4s0emnt on

    m1achinkeubelet, minikube                spec.

    cNoonrtmaianlers{dnsmCraesaqt}ed      Created container
    c4o0nmtainers{dnsm                1      kubelet, minikube    spec.
    40amsq}

    Normal           Started                  Started container
```

Check if Kubernetes nodes are ready or not:

```
$NAkuMbEectl get noSdTeAsTUS
                                        AGE         VERSION
   minikube          Ready              43m         v1.7.5
```

If some node fails, then the the STATUS will be not ready.

It's possible to get the information about Kubernetes Objects using kubectl get

```
$Yokuubmeucstlt gspetecify the type of resource to get. Valid resource types

    * acellrtificatesigningrequests (aka 'csr')
    * clusterrolebs indings
    * cloumstpeorns e(vnatlsitdatounsleysf(oarkfaed'ces'r)ation apiservers)
    * contrgomllearprsev(aiskiaon'csm')
    * cdraoenmjoobnssets (aka 'ds')
    * denepdlpooyimntesn(task(aa'keap'')deploy')
    * ehvoernizto sn(ataklapo'edva')utoscalers (aka 'hpa')
    * ijonbgsresses (aka 'ing')
    * lniammitersapnagceess(a(akkaa'l'inms'i)ts')
    * notdweos r(kapkoal'incoie')s (aka 'netpol')
    * persistentvolumesl(aaikmas'p(avk')a 'pvc')
    * podprisersuept tionbudgets (aka 'pdb')
    * podse(caukrait'pyop'o) licies (aka 'psp')
* repsoliucracstieotns c(aoknatr'orsll')er'squ(aokta')'rc')

      * rolebs indeiqnugostas (aka


         * secretsaccounts (aka 'sa')

    * seartveifcuelsse(atska 'svc')

  * sthtoirdapgeacrltaysrseessources
```

Additionally you can specify the namespace to get the Kubernetes objects from specific namespace. In commands (aka 'shortform') shows the short name for the commands. I.e. kubectl get pods and kubectl get po will re- turn same.

If the pod is crashing / unhealthy:

```
$ kubectl logs ${POD_NAME} ${CONTAINER_NAME}
```

To run the commands inside the container:

```
${kAuRbGe1c}tl$e{AxRecG$2{}P.O.. D$_{ANRAGMNE}} -c ${CONTAINER_NAME} -- ${CMD}
```

For example:

```
$ kubectl exec cassandra -- cat /var/log/cassandra/system.log
```

Check if your cluster process is running with ps:

```
e6f 5| 3g4re6p2sk3u7-b3e6m1m7:1e0nt?min0ik0u:2b5e:3--8st/aurstrv/lmibc/v0ibratubaal8beo-x2/f4b-4

83Bdo3x-Hbde8a3dcle9s787- fcdoe --vrde config
```

In this case minikube process is running.

Also if the container engine is docker,checking the docker containers on each node and master running necessary services will help troubleshoot- ing the cluster.

```
$COdoNcTkAeIrNpEsR ID       IMAGE                              COMMAND
CgcRrE.iAo/TgEoDogle_conStTaAinTeUrSs/heapsPtOerR_TgSrafana:v2N
.6A.M0-2ES57ce"7/bi9n1/3s0h2-6c     /     rcuondssh"                    28
secondsk8asg_ografUapna2.7ebse42e400_influxdb-grafana-qkps6_
kbu93bee3-scystemku_bf4e2r1nde2tecsf-/ch7eaa8p-s1t1eer_7i-n90fl9uax-
d02b4:v20a.c611005e_130df7e"i5n2fl7u82x2d2

-in-cflounxfidgb  7."8c2893dse7c_oinndflsuaxgdob-grUapfa2n7a-

sqekcposn6d_skube-system_fk482s1_d2cf-
c57ba483-01c1ae478-2970c9a-
02g4c2ra.cio1/1g0o0o5gel_e9_aco71nbta6i1n4ers/pause-amd64:3.0
"P/OpaDu.dse8"dbe16c_in2fl9usxedcbon-gdrsafaagnoa-
qkUpps62_8ksuebcoe-nsdystem_f421d2cf-kc87sa_8- 1co1en7ta-
9in09ears-0/p2a4u2asec-
1a1m00d56e4_:633.064783d72155"/9p1a0uasfec"7

gcr.i2o9/gsoeocgolned_s

akguobe-sUyspte2m8 _sef3cfo8nad7sda-c7a8-11e7-9k089sa_-

0P2O4D2.adc81d1b0e0156e_c_f6h3eea1pes2t8er-gfrzl_

ea1m8de6c4f4:v010.65e.10   "/dagcshr.bioo/agrodog--
fpeo_crto.n.."tai2n9ersse/ckounbdesrnageotes-dUapsh2b8osaercdo-nds
```

kusb_ek-usybsetrenme_tef3s-cdda4s0hbbco-ca7rda8.5-1317e47e-79b0a9_ak-
0u2b4e2rance1t1e0s-0d5aes_h6b1o85a7rdf6-z8nmh3_

```
d"/0p8acuasde      2g9csre.icoo/ngodosgalgeo_conUtapin28erss
3" 6b80f           e/cpoanudses-amd64:3.0
                                                        k8s_
Pf3OcDd4.d08bdcb-ce71a68c-_1k1ueb7e-9r0n9eat-e0s2-
d4a2ashc1b1o0a0r5de-z_n0m14h735_6k95udb0e2-sdy0sdtebm56_3c7
gkcurb.ieo-/agdodoognles-.csho"nta3in1 esresc/oknudbse-aagdodonU-
mp a3n0asgeer-amd64:v2
                                                        "/opt
/
cmoanndasger-host01_kukb8es-s_yksutbeme-a_d38d9obnf-
dm8a3n00a2gdedr.c68751e9862bb8522_3k0u9baee-3aad3dco2n_-
a"/fp0aducsbea"7d1db6be3b1bseefcbondsgacgro.io/gUoopg3le0_sceonta
iners/pause-amd64:3.0
choonstd0s1_kube-
systemk_83s8_9PbOfDd8.d380d0b2de1d6cc8_5keu82bbe5-a2d3d0o9ane-
m3aa3nca2g_er-
am9d4624f3:vc1.057.2da6b3c71"/alocalkgcurb.ieos/gtaorotg.l.e."_c3o7n
tsaeicnoenrds/sloacgadkubUep- 36 se
conds            minikube
```

This command will return the all containers which are running on Nodes and Masters.

Networking Constraints

Kubernetes uses the service object to expose the application to the out- side world. The service act as the proxy for the Kubernetes set of pods. To the kubernize application, it is necessary to keep in mind that the applica- tion will support Network Address Translation (NAT) and does not requires forward and reverse proxy.

Inspecting and Debugging Kubernetes

To inspect and debug the Kubernetes it's recommended to have the good understanding of the high-level architecture of the Kubernetes. Kuberne- tes architecture consists masters and nodes. Each master is having mas- ter-components. Also for the nodes, node-components are in-place.

Make sure cluster is up and running on all nodes are ready.

```
    $Nkubectl get nodesAGE
mAinMikEubeSRTeAaTdUy S1h      VERSION
                                v1.7.5
```

If the status is ready for all Nodes that means the all necessary system processes are running on the Kubernetes Nodes.
Check the necessary pods running inside Kubernetes master and Kuber- netes Nodes.

Run following command on every Node and Master:

```
     $NAkuMbEectl get pRoE-oADwYideST--
   AnTaUmSespRaEceSTkAuRbTeS-syAsGteEm

                                                   IP              NODE
kuabnea-
gaedrd-on-   1/1     Running   0          1h     192.168.99.100  mkuibnei-

minikube

910330662-   3/3     Running   0          1h     172.17.0.3      mkuibnei-

kuberne-     1/1     Running   0          1h     172.17.0.2      mkuibnei-

board-
1pc46
```

The result of above command will vary on the type of installation. In this case the cluster is running through minikube.

Alternatively its possible to check containers using docker if the docker is container engine used within Kubernetes.

Use:

```
$ docker ps
```

It will also help to determine the master-level components running on the master. Note that following services are important on masters and nodes:

Master services: Services include: kube-controller-manager, kube-sched- uler, flanneld, etcd, and kube-apiserver. The flanneld service is optional and it is possible to run the etcd services on another system.

Node services: Services include: docker,kube-proxy, kubelet, flanneld. The flanneld service is optional.

Flanneld is optional service and depends on the container networking it may be different.

If any of the service is failed to launch on the Kubernetes then check the setup (YAML) file of the service (usually at /etc/kubernetes) and try to re- start the service.

Checking the system logs through journalctl:
If there is a problem of starting specific services then the best way to de- bug it to check the systemd journalctl log.

Example:

```
# journalctl -l -u kubel-eatpiserver
```

Querying the State of Kubernetes

Kubectl command line tool is used for getting the state of the Kubernetes. Kubectl configured with the API server. API server is the key component of the master. Kubectl utility is used to get the all information about the Ku- bernetes Objects. It includes Pods, Deployments, ReplicationControllers, PersistentStorage etc.

To get the information:
kubectl get command is used.

Following is the example of the few objects retrieved through the Kubectl command line.

```
$ kubectl get pods --namespace kube-system
NAME                        READY    STATUS    RESTARTS  AGE
heapster-q377d              0/1      Error     2           22s
influxdb-grafana-x4h8c      2/2      Running   0           22s
keru-bheo-
satd0d1on-manag-            1/1      Running   0           24s
kbuoabredr-
n4evtnersc-dash-            1/1      Running   0           22s

$ kubectl get services --namespace kube-system
NAME          TYPE        CIPLUSTER-EIPXTERNAL-   PORT(S        AGE
heapster      ClusterIP   10.0.0.90    <none>       80/TCP        33s

kteusb-
dearnshe--    NodePort    10.0.0.115   <none>                     33s

                          8T0C:P30000/ ClusterIP
                                                                  32s
ing-influxdb
                          10.0.0.166   <none>       8083,8/
```

086/

```
   $ kubectl get rc --namespace  kube-system
NAME                    DESIREDCURRENT      READY      AGE
 heapster           1           1           0          44s
  influxdb-grafana  1           1           1          44s
b$uoabredrnetes-dash-      111                         44s
                           t --napace kube-sy
   Nkubectl get deploymen     mes             stem
    o resources found.
```

This is just the querying for getting the information of the Kubernetes objects. To describe the specific object kubectl describe command is used. Let's describe the information about the pod.

```
$ kubectl describe pods influxdb-grafana-x4h8c --namespace

Name:               influxdb-grafana-x4h8c
Namespace:          kube-system
Node:               host01/172.17.0.101
Start Time:         Mon, 13 Nov 2017 14:11:51 +0000
Labels:             kubernenteasm.ioe/=cilnuflsutexrG-
                    srearfvaincaeA=tnrnuoetations:
                    kRuebfeerrennecte",".iaop/icVrerastieodn-"b:y
                    "v=1{""k,"irnedfe"r:"eSnecreia":l{i"zkeidn
                    d"n":a"mReepspliaccaeti"o:"nkCuobnet-roller",
                    sgyrastfeamna","",”nuaimd"e:"9:"7ien1fl7ucxad5
                    b-c-87c-11e7-ac86-...

Status:             Running
IP:                  172.17.0.101
```

```
Created By:          ReplicationController/influxdb-grafana
Controlled By:       ReplicationController/influxdb-grafana
Containers:
influxdb:
Container ID:
docker://d43465cc2b130d736b83f465666c652a71d05a2a169eb72f2369c-
3d96723726c
Image:               kubernetes/heapster_influxdb:v0.6
  Image ID:
                     sdtoecrk_einr-
                     flpuuxldlabb@les:h//ak2u5b6e:7rn0bet3e4s
                     b/h65edaepf-36fd-
Port:                09f7574baafb57705d05ce17427ac41c3c82e
                     086dace9e6a-
State:               <none>
Started:             Running
  Ready:             Mon, 13 Nov 2017 14:11:52 +0000
Restart Count:       True
Environment:         0
Mounts:              <none>
                     /data from influxdb-storage (rw)
                       /var/run/secrets/kubernetes.io/
grafana:             serviceaccount from default-token-3rdpl  (ro)
  Container ID:

Image:
                     7d5o0c3kde7r:a//49a890b86d459b94b841656ba

                     dd8770ef8565fd5e1220330d4f75ce2b-

                     gcrar.fiaon/gao:vo2g.l6e._0c-

                     o2ntainers/heapster_
Image ID:            derosc/kheera-
psutfelar_bglera:/f/agncra.@ios/ghoao2g5f6e:_2c0o8nct9a8ibn--
```

Port:

State:

Started:

767fd142e4108aa5d777705092ce099d5e82b7f787d467a3243bf75b-

Running

Mon, 13 Nov 2017 14:11:52 +0000

```
   Ready:                     True
Restart Count:                0
   Environment:

URLIN:
FLUXDB_SERVICE_               http://localhost:8086
ABLGEFD_A:
UTH_BASIC_EN-                 false
MOGUFS__AEUNTAHB_LA           :  true
ENDO: NY-

GMFO_UAUS_TOHR_GA_N           Admin

RONLEY:-

GF_SERVER_ROOT_URL:           /
Mounts:                       /var from grafana-storage (rw)
                                                          /
                               svearrv/ircuena/csceocurnettsf/rkoumbedrenfe
                                          atuelst.-itoo/ken-3rdpl (ro)
Conditions:

Type                          Status
Initialized                   True
Ready                         True
PodScheduled                  True
Volumes:
influxdb-storage:
Type:                         :shaErmespatypDoidr's(alitfeemtimpoer)ary directory
that


grafana-storage:
Type:                         Ea mpopdty'sDlifre(taimteem) porary directory that
shares


  default-token-3rdpl:
   Type:                      Secret (a volume populated by a Secret)
```

```
SecretName:              default-token-3rdpl
Optional:                false
QoS Class:               BestEffort
Node-Selectors:          <none>
Tolerations:             <none>
Events:

Type       Reason       Age  From           Message
– –        – –          – –  – –            – –

Normal     Scheduled    3m   duelefrault-   Sinuflcucexsdsbf-
                                            uglrlyafaasnsaig-xn4ehd8c to
sched- Normal           Pulled    3m        Chontt0a1iner image
                                            "inkfluubxedrnbe:vte0s.6/h"
           khuobste0l1et,                   eaalrpesatedry_ Cpreaetendt
                                            ocnonmtaaicnheirnwe ith
                                            dSeocukreirtyid:
Normal     Created      3m   khuobste0l1et, [sde4c3c4o6m5cpc=2ubn1c3o;
                                            n- Sfitnaerdte]d container with
                                            Cdonktaeirniedrdi4m3a4g6e5c"
Normal     Started      3m                  gcc2rb.13
           khuobste0l1et, Normal            ihoe/agposotgelre__gcroanfat
           Pulled       3m                  anian:evr2s./6.0-2"
           khuobste0l1et,                   Calrreeaatdeyd
                                            pcroensteanint
                                            eornwmitahchine
                                            dSeocukreirtyid:
Normal     Created      3m   khuobste0l1et, [s9e8c0c8odm5bp4=8u1n5cb
                                            o;n-

Normal     Started      3m   khuobste0l1et,
                                            docker id 9808d5b4815b
```

This consists the in depth level of information about the Pod.

Checking Kubernetes yaml or json Files

Kubernetes supports the declarative method of the specification of the Kubernetes object. The declarative approach is one of the recommended approaches to define the specifications. The specification file can be in YAML or JSON format.

The first thing to do with the declarative approach is to validate the YAML or JSON file which contains the specifications. Online validators are avail- able like http://www.yamllint.com/ and https://jsonlint.com/ . This will re- move the syntax and parsing error from the specification file.

Checking error with kubectl create command. kubectl create command is used for the creating the Kubernetes object with the specification file.

deployment.yaml

```
akmpineiVde: rDseiopnlo: yemxtentsions/v1beta1

  nlaatmadea::thae: llo-world

      avbpepl:shello-world

    spr epecr:: v120

  semlealcictoasrL:abels:

       aveptpc:hh1ello-world

temmeprl:advtaet:a:

         labtaels:

              app: hello-world
    spcveecr:t: v1  rs:

- onnamaien:ehkeulbloe-jwa okr/lhdelloworld:latest

        ipmoratgse:PullPolicy: Always

        - containerPort: 3000
        kubectl create -f deployment.yaml
```

It will throw an error if there is any syntax error or API error.

Deleting Kubernetes Components

With Kubectl it is possible to delete the components of Kubernetes. If declarative definition is used then the similar definition can be used for deletion.

deployment.yaml

```
akmpineiVde: rDseiopnlo: yemxtentsions/v1beta1

 nlaatmadea::thae: llo-world

      avbpepl:shello-world

   spr epecr:: v120

    selelcictoasr:

         matchLabels:
```

```
   aveprp::vh1ello-
 world
 temmeptaladtaet:a:
 laabpepl:s:hello-
 world spveecr:: v1
   c- onnatmaien:ehresl:lo-world
    imagePkuullbPeojlaicyk:/hAelwlloa
    wysorld:latest

    - containerPort: 3000

kubectl delete -f deployment.yaml
```

kubectl delete command will be used for the deleting the kubernetes com- ponents. In imperative method there are different way of deleting the ku- bernetes components.

```
"foo" #kuDbeelcetledpeoledtseapnoddsse,srevricveicsews i-tlhnlaambel=nmaymLaeb=melyLabel. #kuD
```

Similarly it will work with all other object of Kubernetes.

9. Kubernetes Maintenance

ernetes Cluster • Managing Kubernetes with Dashboard Logging Kubernetes Cluster Upgrading Kubernetes

•

•

•

Monitoring Kubernetes Cluster

For reliable applications, it is required to have in place monitoring of the Kubernetes Cluster. It helps to determine availability, scalability, and reli- ability of the application deployed over Kubernetes Cluster.

Heapster aggregator is used for monitoring and event the logs. Heapster stores the information in storage backend. Currently, it supports Google Cloud Monitoring and InfluxDB as the storage backends. Heapster runs as a pod in the Kubernetes Cluster. It communicates with each node of the Kubernetes Cluster. Kubelet agent is responsible to provide the monitoring information to Heapster. Kubelet itself collects the data from cAdvisor.

cAdvisor:

cAdvisor is an open source container usage and performance analysis agent. In Kubernetes cAdvisor is included into Kubelet binary.cAdvisor auto-discovers all containers. It collects the information of CPU, memory, network and file system usage statistics.

Kubelet:

Kubelet bridge the gap between Kubernetes Master and Kubernetes Node. It manages the Pods and Containers on each machine.

InfluxDB and Grafana are used for storing the data and visualizing it. Google cloud monitoring provides the hosted solution for monitoring Kubernetes Cluster. Heapster can be set up to send the

metrics to Google Cloud monitoring.

Let’s check the Monitoring Kubernetes Cluster created with Minikube:

```
$- admdadinoink-umbaenaadgndeorbn: senliasbt led

    - kdusbheb-ldonasrd:oe: neaebalelded   enabled

  - hinefaapusstte-rs:tidriasgalbecldeladss:

        - regisetrsy::ddissaabbleddisabled

               egistry-creds:
```

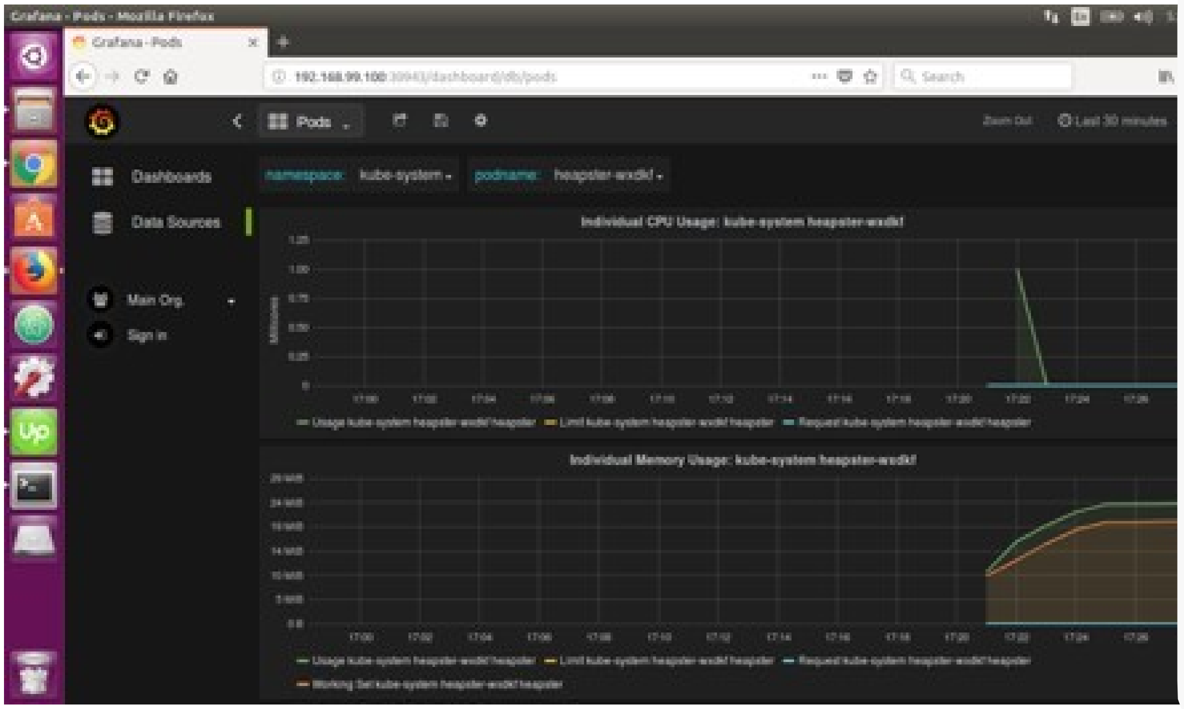
Enable the addon:

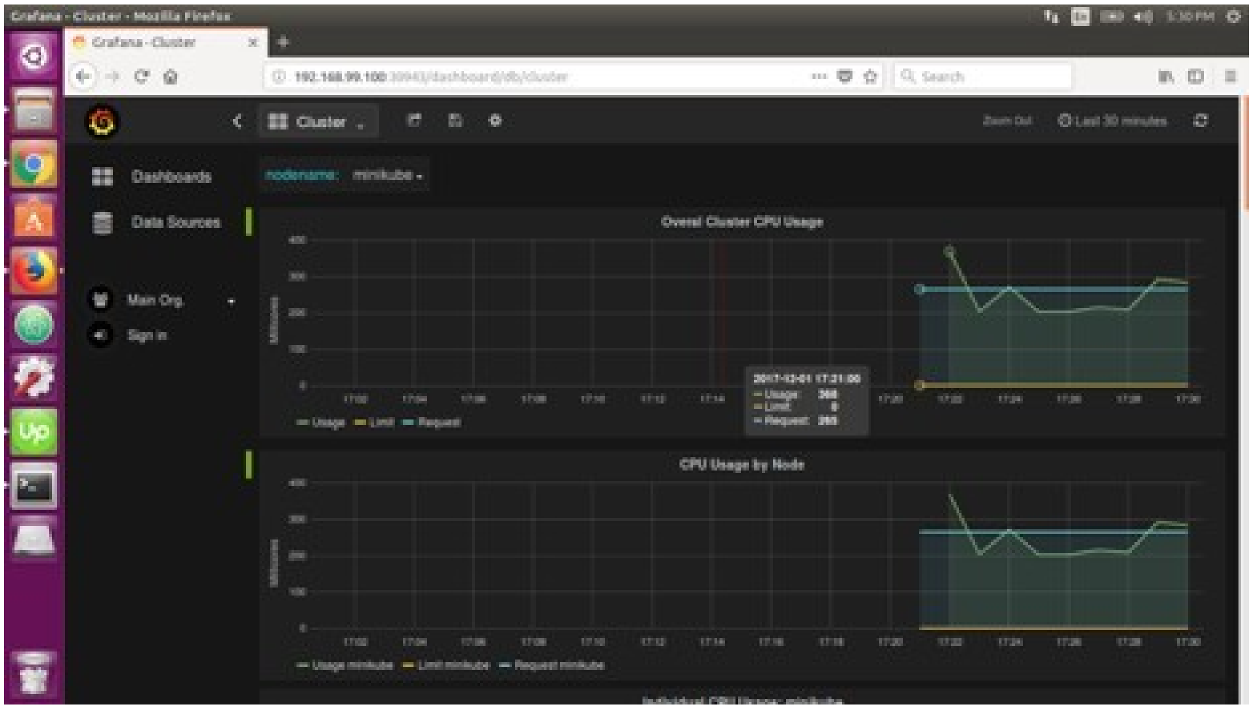
```
$ minikube addons enable heapster
```

To open the web interface:

```
$ minikube addons open heapster
```

The result will be displayed on the grafana.


Grafana - Pods - Mozilla Firefox
Grafana - Pods
192.168.99.100:30943/dashboard/db/pods
Pods
Zoom Out
Last 30 minutes
Dashboards
Data Sources
Main Org.
Sign in
namespace: kube-system
podname: heapster-wxdkf
Individual CPU Usage: kube-system heapster-wxdkf
Individual Memory Usage: kube-system heapster-wxdkf



In Minikube addons are helping for monitoring but it's also possible to add heapster as Kubernetes deployment. This will be the manual installation of heapster, grafana and influxdb.

```
akpiniVde:
rSseirovni:cve1Accou
nt mneatmadea:thae:
apster
--n- amespace: kube-system
akpiniVde: rDseiopnlo:
yemxtentsions/v1beta1
mneatmadea:thae: apster
spneacm: espace: kube-
  system rtemplpiclate: :1
    mlaebtaedlsa:ta:
        tka8ssk-a:
    pmpo:nhietoarpisntger
    spseercv:iceAccountName:
    heapster c- onnatmaien:ehresa:
    pster
        imagePguclIrP.iool/igcoyo:
        glfleN_octoPnrteasiennetrs/heapster-amd64:v1.4.0 c-
        o/hmemapasntedr:
        -
--sinukr=cien=flkuuxbdebrn:hettteps::/h/mttopns:i/t/okruinbegr-
innefltuesx.ddbe.fkaublte-system. s--v-c:8086
akpiniVde: rSseirovni:cve1
```

```
mlaebtaedlsa:ta:
  t#aFsko:r musoenaitsoariCnlguster add-on
(https://github.com/kubernetes/ku-
be#rnIfeyteosu/tarreee/NmOaTstuesr/icnlgutshteisr/asddanonasd)don,
you should comment out thkisulbineern. etes.io/cluster-service:
'true'

  namespace: kube-system

  ports:

    targetPort: 8082

    k8s-app: heapster
```

You can get the latest version of the heapster at https://github.com/kubernetes/heapster/ .

Using Kubectl:

```
$ kubectl create -f heapster.yaml
```

For grafana, use grafana.yaml:

```
akpiniVde: rDseiopnlo:
yemxtentsions/v1beta1
mneatmadea:tma:onitoring-
grafana spneacm: espace:
kube-system rtemplpiclate: :1
   mlaebtaedlsa:ta:
      tka8ssk-a:
   pmpo:ngirtaofraina
   g spcoenc:tainers:
     - inmaamgee:: ggrcarf.iaon/gaoogle_containers/heapster-grafana-
       amd64:v4.4.3 p- coortnst:ainerPort: 3000

       vpolruomtoecMolo: uTnCtPs:
       - nmaomuen:tcPa-tche:r/teitficc/asstel/scerts
       - rmeaoduOnntPlya:th:tr:u/vear

    envm:   e: grafana-storage

       - vnaalmuee::mINoFnLitUoXriDnBg_-iHnflOuSxTdb
       - vnaalmuee::"G30F_0S0E" RVER_HTTP_PORT
cessib#lTe hveiafollowing env variables are required to make Grafana ac-
```

recom#mtheenkdubernetes api-server proxy. On production clusters, we pose #threemgroavfaina these env variables, setup auth for grafana, and ex-

- #nasemrev:icGeFu_AsiUnTgHa _LBoAaSdIBCa_lEanNcAeBrLoErDa public IP.
- vnaalmuee::"GfaFl_sAe"UTH_ANONYMOUS_ENABLED
- vnaalmuee::"GtrFu_eA"UTH_ANONYMOUS_ORG_ROLE
- vnaalmuee::AGdFm_SiEnRVER_ROOT_URL

# Ivfayluoeu:'r/eapoin/vly1/unsainmgetshpeacAePsI/kSuebrvee-srypsrteomxy/,sseertvtihceiss/vmalouneitionrs-tead: ing-grvalaunea: /proxy

v- onlaummee:sc:a-

certificates

hpoasthP:at/eht:c/ssl/

certs

- enmamptey:Dgirra: f{a}na-storage

akpiniVde:
rSseirovni:cve
1
mlaebtaedlsa:t
a:
be#rnFeotreus/stereaes/ma Calsutesrte/crluadstde-ro/and(dhottnpss)://github.com/kubernetes/ku- th#isIlfinyoe.u are NOT using this as an addon, you should comment out kubernetes.io/cnlaumstee:r-mseornviitcoer:in'trgu-ger' afana

---

```
spneacm: espace: kube-system
a#n Ienxtaeprnroadl uLcotaidobnasleatnucpe,rwe recommend
  accessing Grafana through # otyrpteh:rLoouagdhBaaplaunbcleicr IP.
ly#-gYeonuecroauteld palosrot use NodePort to expose the service
  at a random- #potyrtpse:: NodePort
  -
  tpaorrgte:t8P0ort:
  3000 sekl8esc-
  taoprp: : grafana
```

You can get the latest version of grafana.yaml at https://github.com/ku- bernetes/heapster/blob/master/deploy/kube-config/influxdb/grafana. yaml .

Using kubectl :

```
$ kubectl create -f grafana.yaml
```

If influxdb is the storage backend, then use following YAML for deploying influxdb in Kubernetes Cluster:

```
akmpineiVde: rDseiopnlo: yemxtentsions/v1beta1

    natmadea:tma:onitoring-influxdb

        namespace: kube-system
```

```
spreepcl:icas: 1
 temmeptaladtaet:a:
 latabsekls::monitori
 ng spke8cs: -app:
 influxdb c-
 onnatmaien:einrsfl:u
 xdb
      ivmolaugme:egMcor.uion/tgso: ogle_containers/heapster-influxdb-
      amd64:v1.3.3

      -

    nmaomuen:tiPnaflthu:x/dd

    ba-tsatorage v-

    onlaummee:si:nfluxdb-

    storage

---
      emptyDir: {}

akpiniVde:
rSseirovni:cve
1
mlaebtaedlsa:t
a:
   t#aFsko:r                    musoenaitsoariCnlguster                    add-on
(https://github.com/kubernetes/ku-
be#rnIfeyteosu/tarreee/NmOaTstuesr/icnlgutshteisr/asddanonasd)don,
you  should  comment  out  thkisulbineern.  etes.io/cluster-service:
'true'
 nkaumbee:rmneotnesit.oior/ingm-inefl: umxodnbitoring-influxdb



 ports:
```

```
- tpaorrgte:t8P0o8r6t: 8086

  sekl8esc-taoprp: : influxdb
```

You can get the latest version of influxdb.yaml at https://github.com/ kubernetes/heapster/blob/master/deploy/kube-config/influxdb/influxdb. yaml

Using Kubectl:

```
$ kubectl create -f influxdb.yaml
```

To access grafana dashboard with the manual setup, describe the grafa- na service and check endpoint of the service.

To describe the service using Kubectl use following command:

```
$ kubectl describe svc monitoring-grafana --namespace kube-

system Laabmels:pace: adudboen- my

atnemager.kubernetes.io/mode=Reconcile

                 kubernetes.io/minikube-addons-endpoint=heapster

Annotations:     kubectl.kubernetes.io/last-applied-configura-
                 data":{"annotations":{},"labels":{"addonmanager.
                 Type:

Selector:

                 NodePort
```

k
m
u
i
b
n
e
i
k
r
u
n
b
e
t
e
e
-
s
a
.
d
i
o
d
/
o
m
n
o
s
.
d
.
.
e
”
:
”

Reconcile”,”kubernetes.io/
acidlde,onnammaen=aingeflru.kxuGbraefranneates.
io/mode=Recon-

```
IPPo:rt:      NePort:      1<0u.n0s.0e.t6>28TCP
                           <30943/TCP

ESenodspiooninAtsffnity:  N17u2n.s1e7t.>0.9:3000

  Evsents:              <noonnee>
```

Prometheus and Data Dog are also good tools for monitoring the Kubernetes Cluster.

Managing Kubernetes with Dashboard

Kubernetes provides the web interface to manage the Kubernetes Cluster. With the Role Based Access Policy, it becomes simpler and easier to manage Kubernetes Cluster with Dashboard. For Minikube it gives the addon. That means Minikube provides addons for monitoring, dashboard and for managing the cluster.

To enable the addon of the dashboard on minikube:

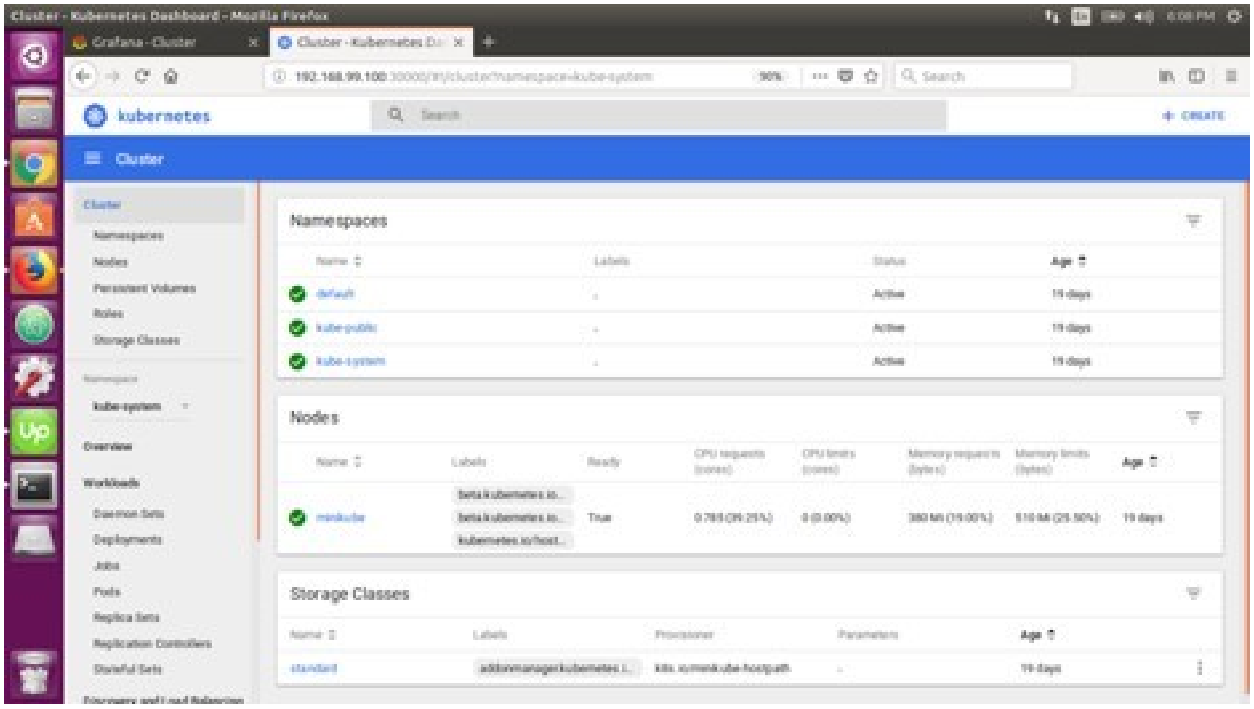
```
$amshinbiokaurbdewaadsdsouncsceenssafbullelydeanshaobaleadrd

 Opmeinniinkgubkeubaedrdnoentsesopseenvidcaeshkubbe-rsdystem/kubernetes-dashboard in

      default browser...
```

With Minikube, it will open the dashboard in the browser. The dashboard consists the Cluster infromation, Namespace information and information about the objects associated with cluster like workloads etc.The dashboard consists the three different informative sections including workloads, discovery and load balancing, config and storage.

In the cluster tab, Kubernetes dashboard shows the information about the Namespaces, Nodes, Persistent Volumes and Storage Classes.



Namespaces

	Name	Labels	Status	Age
✓	default	-	Active	19 days
✓	kube-public	-	Active	19 days
✓	kube-system	-	Active	19 days

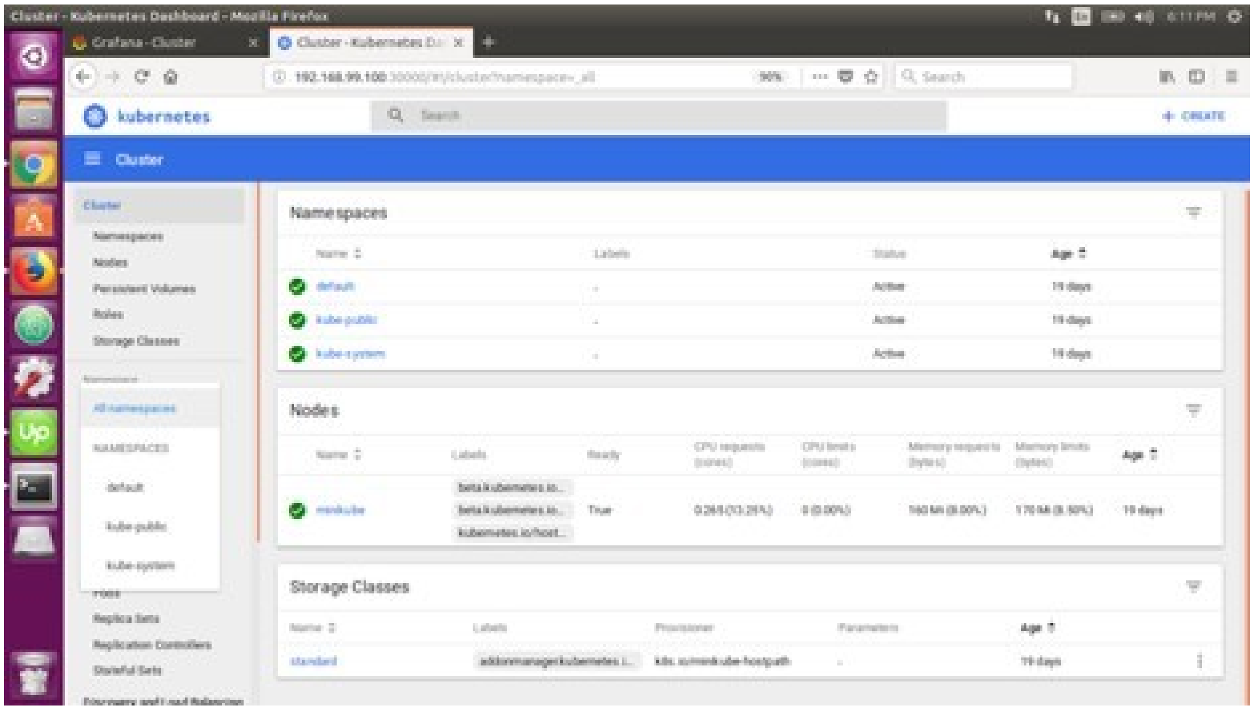
Nodes

	Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
✓	minikube	beta.kubernetes.io... beta.kubernetes.io... kubernetes.io/host...	True	0.785 (39.25%)	0 (0.00%)	380 Mi (19.00%)	510 Mi (25.50%)	19 days

Storage Classes

Name	Labels	Provisioner	Parameters	Age
standard	addonmanager.kubernetes.i...	k8s.io/minikube-hostpath	-	19 days

It gives the brief information about the cluster. In namespaces, it shows the all available namespaces in Kubernetes Cluster. It's possible to select all namespaces or specific namespace.



Namespaces

	Name	Labels	Status	Age
✓	default	-	Active	19 days
✓	kube-public	-	Active	19 days
✓	kube-system	-	Active	19 days

Nodes

	Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
✓	minikube	beta.kubernetes.io... beta.kubernetes.io... kubernetes.io/host...	True	0.265 (13.25%)	0 (0.00%)	160 Mi (8.00%)	170 Mi (8.50%)	19 days

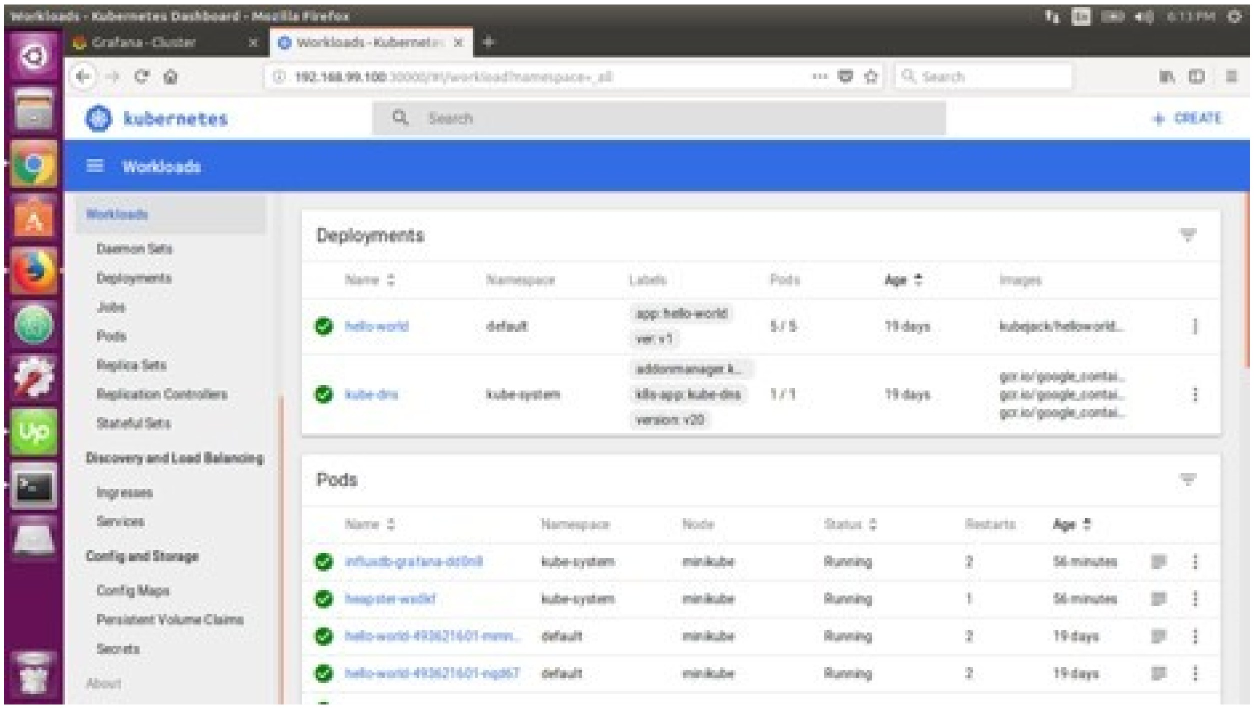
Storage Classes

Name	Labels	Provisioner	Parameters	Age
standard	addonmanager.kubernetes.i...	k8s.io/minikube-hostpath	-	19 days

Depending on the namespace selected previously, further tabs show in depth information of associated Kubernetes object within selected namespace. It consists the workloads of Kubernetes which includes

Deployments, Replica Sets, Replication Controller, Daemon Sets, Jobs, Pods and Stateful Sets. Each of this workload is important to run an

application over the Kubernetes Cluster.



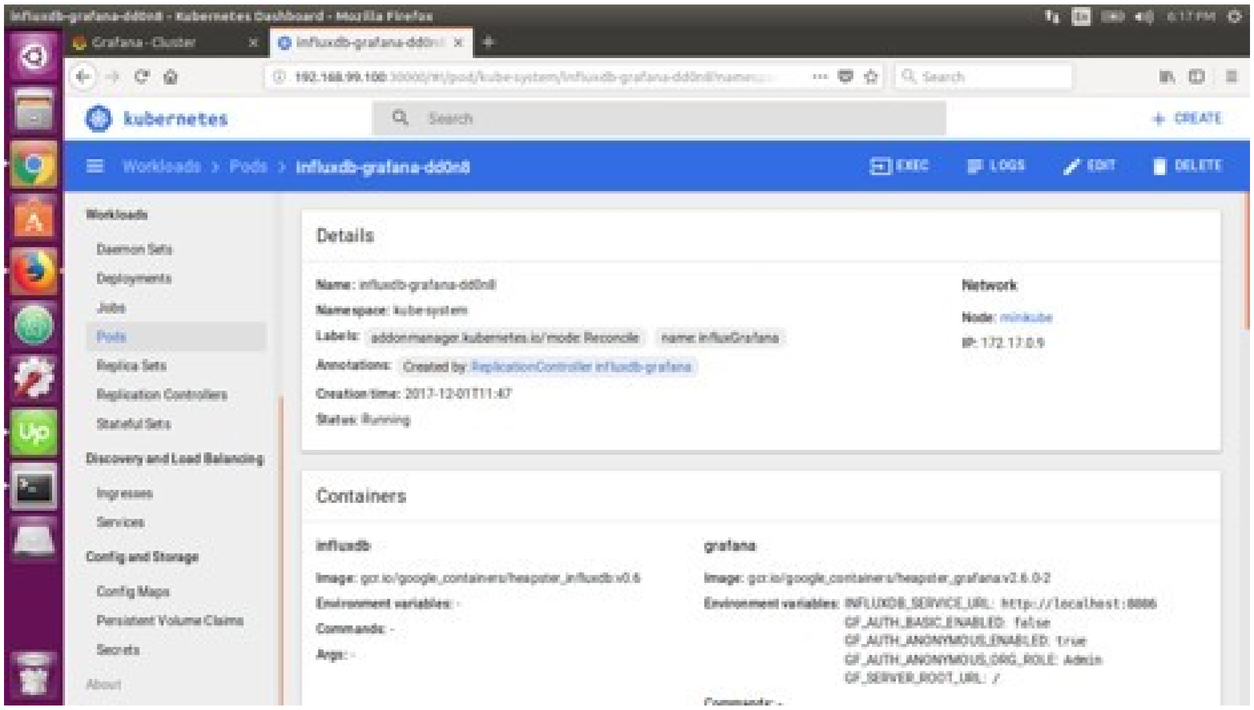
Deployments

	Name	Namespace	Labels	Pods	Age	Images
✓	hello-world	default	app: hello-world ver: v1	5 / 5	19 days	kubejack/helloworld...
✓	kube-dns	kube-system	addonmanager.k... k8s-app: kube-dns version: v20	1 / 1	19 days	gcr.io/google_contai... gcr.io/google_contai... gcr.io/google_contai...

Pods

	Name	Namespace	Node	Status	Restarts	Age
✓	influxdb-grafana-dd0n8	kube-system	minikube	Running	2	56 minutes
✓	heapster-wsdkf	kube-system	minikube	Running	1	56 minutes
✓	hello-world-493621601-mmn...	default	minikube	Running	2	19 days
✓	hello-world-493621601-nqd67	default	minikube	Running	2	19 days

The Dashboard also gives the information of each workload. Following screenshot describes the Kubernetes Pod in detail.



Details

Name: influxdb-grafana-dd0n8
Namespace: kube-system
Labels: addonmanager.kubernetes.io/mode: Reconcile   name: influxGrafana
Annotations: Created by: ReplicationController influxdb-grafana
Creation time: 2017-12-01T11:47
Status: Running

Network
Node: minikube
IP: 172.17.0.9

Containers

influxdb
Image: gcr.io/google_containers/heapster_influxdb:v0.6
Environment variables: -
Commands: -
Args: -

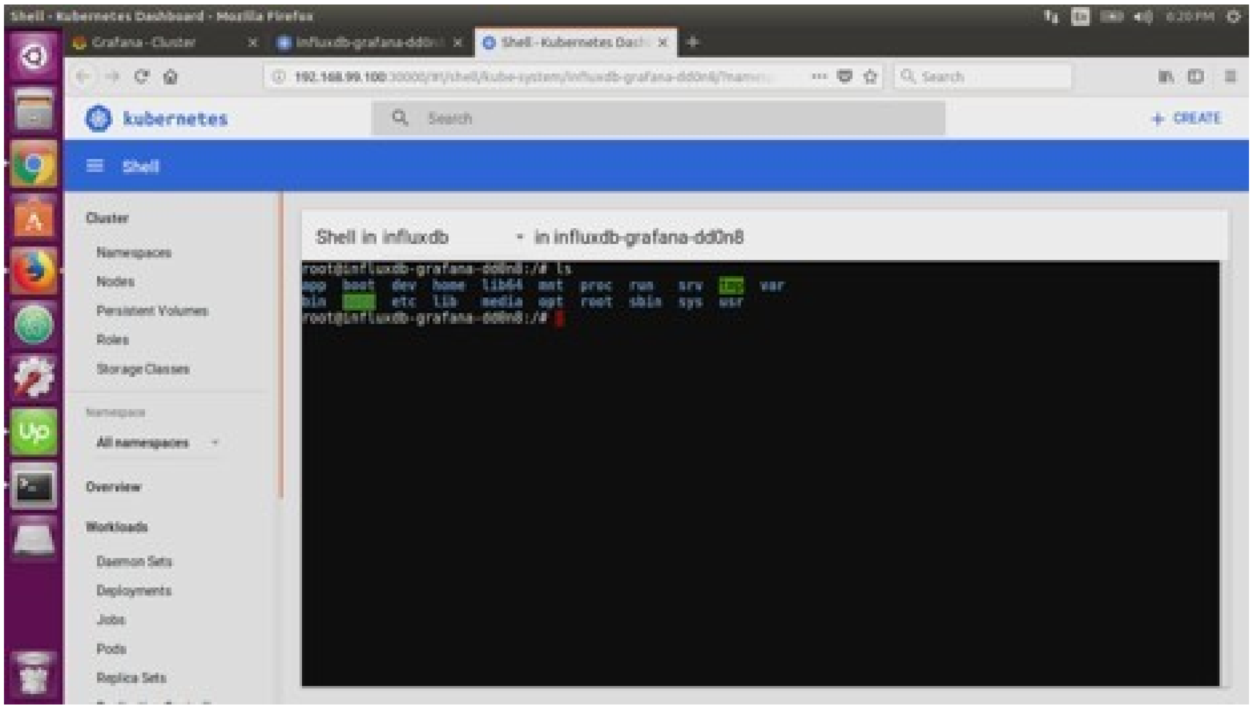
grafana
Image: gcr.io/google_containers/heapster_grafana:v2.6.0-2
Environment variables:

```
INFLUXDB_SERVICE_URL: http://localhost:8086
GF_AUTH_BASIC_ENABLED: false
GF_AUTH_ANONYMOUS_ENABLED: true
GF_AUTH_ANONYMOUS_ORG_ROLE: Admin
GF_SERVER_ROOT_URL: /
```

Commands: -

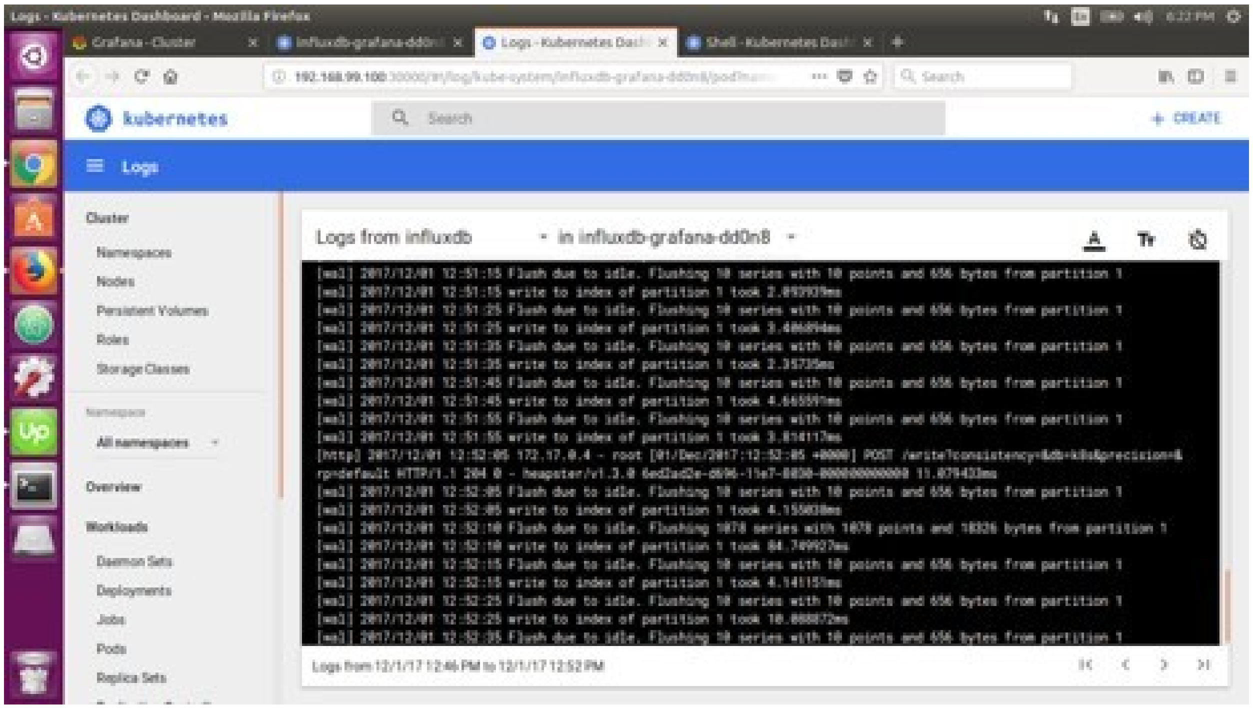
But the dashboard is just not limited to information, it's even possible to ex- ecute in the pod, get the logs of the pod, edit the pod and delete the pod.

Exec inside the influxdb-grafana pod:


Shell in influxdb · in influxdb-grafana-dd0n8

```
root@influxdb-grafana-dd0n8:/# ls
app  boot  dev  home  lib64  mnt  proc  run  srv  tmp  var
bin  [illegible]  etc  lib  media  opt  root  sbin  sys  usr
root@influxdb-grafana-dd0n8:/#
```

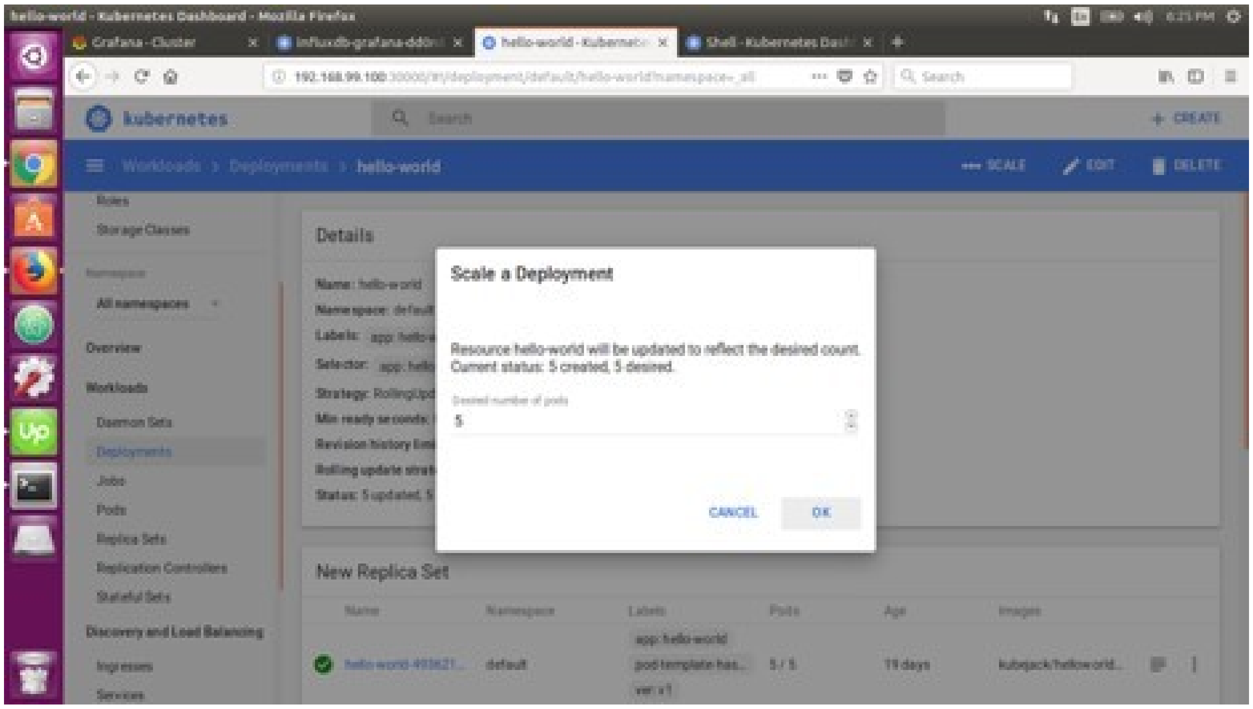
Logs of the influxdb-grafana pod:


Logs from influxdb · in influxdb-grafana-dd0n8

```
[wal] 2017/12/01 12:51:15 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:15 write to index of partition 1 took 2.093929ms
[wal] 2017/12/01 12:51:25 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:25 write to index of partition 1 took 3.486094ms
[wal] 2017/12/01 12:51:35 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:35 write to index of partition 1 took 2.35735ms
[wal] 2017/12/01 12:51:45 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:45 write to index of partition 1 took 4.665597ms
[wal] 2017/12/01 12:51:55 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:55 write to index of partition 1 took 3.814117ms
[http] 2017/12/01 12:52:05 172.17.0.4 - root [01/Dec/2017:12:52:05 +0000] POST /write?consistency=&db=k8s&precision=&
rp=default HTTP/1.1 204 0 - heapster/v1.3.0 6ed2ad2e-d696-11e7-8030-000000000000 11.079433ms
[wal] 2017/12/01 12:52:05 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:05 write to index of partition 1 took 4.155038ms
[wal] 2017/12/01 12:52:10 Flush due to idle. Flushing 1070 series with 1070 points and 18326 bytes from partition 1
[wal] 2017/12/01 12:52:10 write to index of partition 1 took 84.749927ms
[wal] 2017/12/01 12:52:15 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:15 write to index of partition 1 took 4.141151ms
[wal] 2017/12/01 12:52:25 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:25 write to index of partition 1 took 10.000072ms
[wal] 2017/12/01 12:52:35 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
```

Logs from 12/1/17 12:46 PM to 12/1/17 12:52 PM

This is only for Pod. For other objects like deployments, it's possible to scale, edit and delete the deployment.


Scale a Deployment

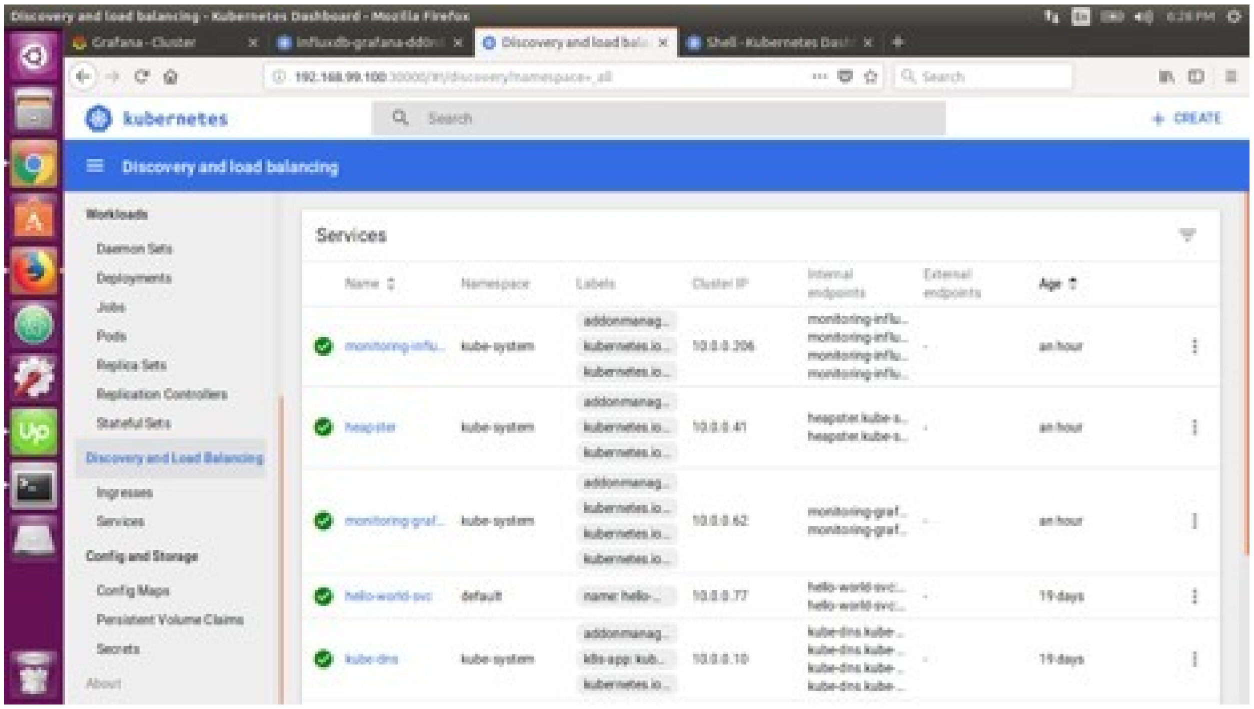
Resource hello-world will be updated to reflect the desired count.
Current status: 5 created, 5 desired.

Desired number of pods: 5

CANCEL OK

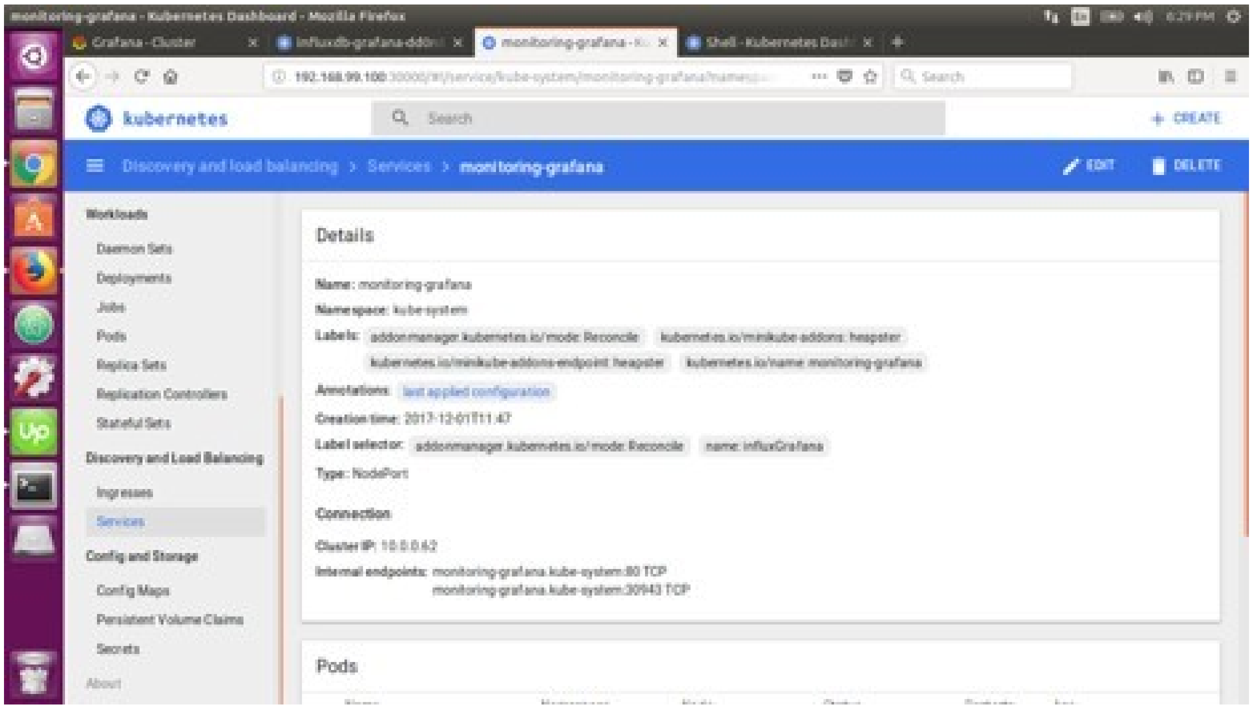
Name	Namespace	Labels	Pods	Age	Images
hello-world-493621...	default	app: hello-world pod-template-has... ver: v1	5 / 5	19 days	kubejack/helloworld...

Discovery and Load Balancing consists the information about the Ingresses and Services.


Services

Name	Namespace	Labels	Cluster IP	Internal endpoints	External endpoints	Age
monitoring-influ...	kube-system	addonmanag... kubernetes.io... kubernetes.io...	10.0.0.206	monitoring-influ... monitoring-influ... monitoring-influ... monitoring-influ...	-	an hour
heapster	kube-system	addonmanag... kubernetes.io... kubernetes.io...	10.0.0.41	heapster.kube-s... heapster.kube-s...	-	an hour
monitoring-graf...	kube-system	addonmanag... kubernetes.io... kubernetes.io... kubernetes.io...	10.0.0.62	monitoring-graf... monitoring-graf...	-	an hour
hello-world-svc	default	name: hello-...	10.0.0.77	hello-world-svc... hello-world-svc...	-	19 days
kube-dns	kube-system	addonmanag... k8s-app: kub... kubernetes.io...	10.0.0.10	kube-dns.kube-... kube-dns.kube-... kube-dns.kube-... kube-dns.kube-...	-	19 days

The dashboard also provides the detail information about Ingresses and Services. It's possible to edit and delete the ingress or service through the dashboard.



Details

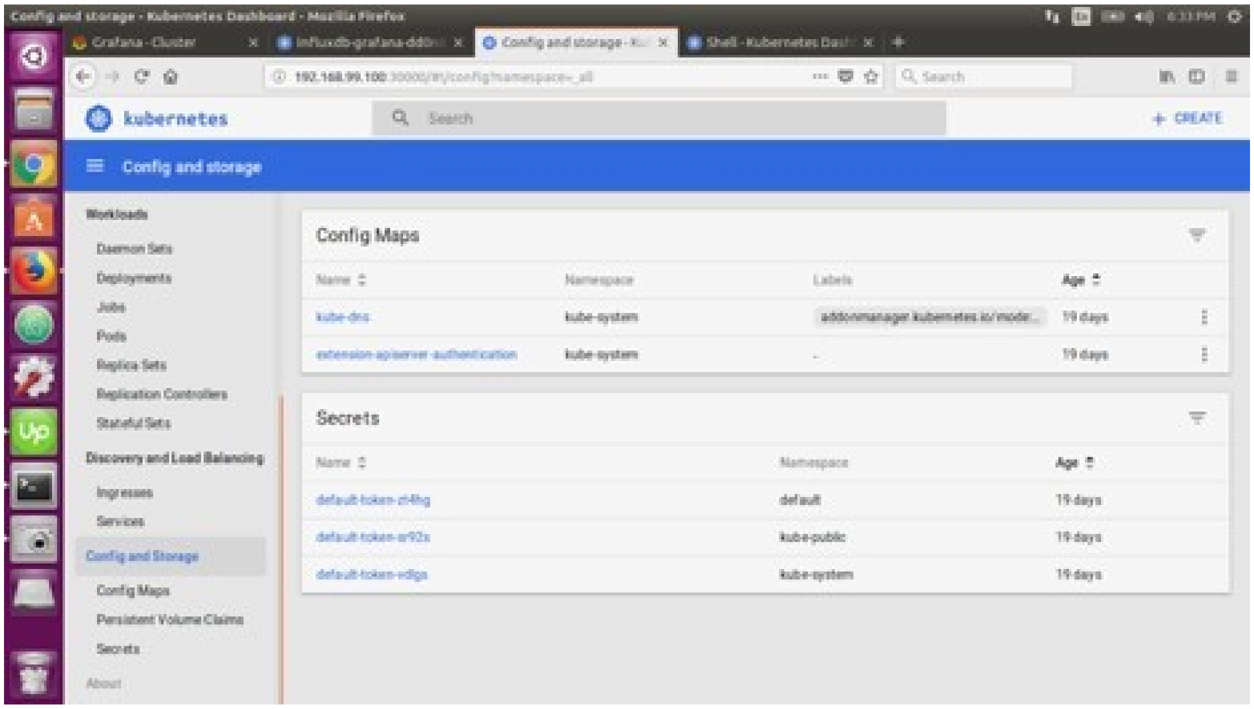
Name: monitoring-grafana
Namespace: kube-system
Labels: addonmanager.kubernetes.io/mode: Reconcile kubernetes.io/minikube-addons: heapster kubernetes.io/minikube-addons-endpoint: heapster kubernetes.io/name: monitoring-grafana
Annotations: last applied configuration
Creation time: 2017-12-01T11:47
Label selector: addonmanager.kubernetes.io/mode: Reconcile name: influxGrafana
Type: NodePort

Connection

Cluster IP: 10.0.0.62
Internal endpoints: monitoring-grafana.kube-system:80 TCP
monitoring-grafana.kube-system:30943 TCP

Pods

Config and Storage consist the information about the Config Maps, Persistent Volume Claims and Secrets.



Config Maps

Name	Namespace	Labels	Age
kube-dns	kube-system	addonmanager.kubernetes.io/mode:...	19 days
extension-apiserver-authentication	kube-system	-	19 days

Secrets

Name	Namespace	Age
default-token-zt4hg	default	19 days
default-token-sr92s	kube-public	19 days
default-token-vdlgs	kube-system	19 days

Also, it's possible to directly deploy the containerized application through web UI. The dashboard will require App name, Container Image, Number of Pod and Service inputs. Service can be internal or external service. The YAML file with required specifications can also be used for the creating the Kubernetes object.

In following example, Minikube addon is used for the Kubernetes dashboard.For manual installation following YAML should be used:

```
Kdausbhebcotlacrdre/mateas-tfehr/tstprcs/:d/
erapwlo.yg/irthecuobmusmerecnodnetde/nktu.cboemrn/keutebse-drnasehtebso/ard.
```

To access the Web UI,

Using Kubectl Proxy:

```
kubectl proxy
```

Using Kubernetes Master API Server: https://<kubernetes-master>/ui

If the username and password are configured and unknown to you then use,

```
kubectl config view
```

Kubernetes dashboard is a flexible and reliable way to manage the Kubernetes Cluster.

Logging Kubernetes Cluster

Application and system level logs are useful to understand the problem with the system. It helps with troubleshooting and finding the root cause of the problem. Like application and system level logs containerized application also requires logs to be recorded and stored somewhere. The most standard method used for the logging is to write it to standard output and standard error streams. When the logs are recorded with separate storage then the mechanism is called as Cluster Level Logging.

Basic Logging with Kubernetes:

In the most basic logging it's possible to write the logs to the standard output using the Pod specification.

For Example:

```
akmpineiVde: rPsoiodn: v1

  spnatmadea:tcao:unter

 c- oenc:tmaien:ecrosu: nt

      inmaage: busybox

     args':i=[/0b;iwn/hsihle, -tcr,ue; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']
```

The logs will be recorded with standard output:

```
$ kubectl create -f log-example-pod.yaml
pod "counter" created
$ kubectl get po
NAME                      READY     STATUS     RESTARTS   AGE
counter                   1/1       Running    0          8s

                          1/1       Running    0          19d
h1ffelrlpo-world-
                          1/1       Running    0          19d
493621601-    hmemllon-
                          1/1       Running    0          19d
zwworld-493621601-

hneqldlo67-world-

493621601-
  hello-world-            1/1       Running    0          19d

$ kubectl logs counter
0: Fri Dec  1 16:37:36 UTC 2017
1: Fri Dec  1 16:37:37 UTC 2017
2: Fri Dec  1 16:37:38 UTC 2017
3: Fri Dec  1 16:37:39 UTC 2017
4: Fri Dec  1 16:37:40 UTC 2017
5: Fri Dec  1 16:37:41 UTC 2017
6: Fri Dec  1 16:37:42 UTC 2017
7: Fri Dec  1 16:37:43 UTC 2017
8: Fri Dec  1 16:37:44 UTC 2017
```

Node level logging with Kubernetes:

The containerize application writes logs to stdout and stderr. Logging driver is the responsible for the writing log to the file in JSON format. In case of docker engine, docker logging driver is responsible

for writing the log.

The most important part of Node level logging is log rotation with the Kubernetes. With the help of log rotation, it ensures the logs will not consume all storage space of the nodes.

Cluster Level Logging with Kubernetes:

Kubernetes does not provide the native cluster level logging. But the clus- ter level logging is possible with following approaches:

1. **Run the agent on each node for log collection**
2. **Run the side container which will be responsible for log collection**
3. **Directly store the logs of the application into the storage**

The most used and recommended method is using the node agent for log collection and storing the logs in log storage.

Stackdriver or Elasticsearch is used for the logging with Kubernetes. How- ever, there are other solutions available like logz.io, sematext logging etc. Fluentd is used with custom configuration along with Stackdriver and Elas- ticsearch. Fluentd acts as the node agent.

For the Kubernetes Cluster created through minikube Giantswarm pro- vides the solution. The solution consists ELK (Elasticsearch , logstash and Kibana) stack logging with minikube. However, it is possible to deploy all these components manually with manifests.

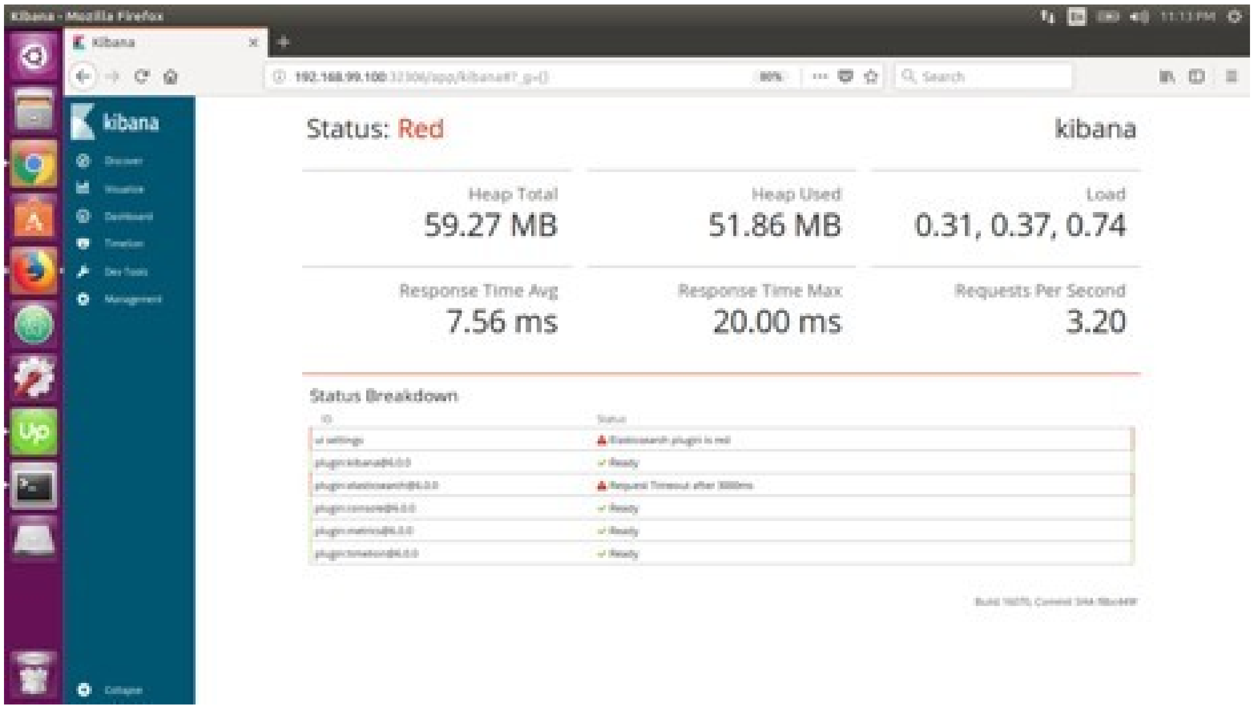
Start the Minikube:

```
minikube start --memory 4096
```

Download all manifests and start Kibana:

```
k-uiƒbleecntlaampeplhytt\ps://raw.githubusercontent.com/giantswarm/kuberne-
    tes-elastic-stack/master/manifests-all.yaml
    minikube service kibana
```

On the Kibana Dashboard :



Status: Red

kibana

Heap Total	Heap Used	Load
59.27 MB	51.86 MB	0.31, 0.37, 0.74

Response Time Avg	Response Time Max	Requests Per Second
7.56 ms	20.00 ms	3.20

Status Breakdown

ID	Status
ui settings	Elasticsearch plugin is red
plugin:kibana@6.0.0	✓ Ready
plugin:elasticsearch@6.0.0	Request Timeout after 3000ms
plugin:console@6.0.0	✓ Ready
plugin:metrics@6.0.0	✓ Ready
plugin:timelion@6.0.0	✓ Ready

Build [illegible], Commit SHA [illegible]

The logging will be enabled and you can check it through Kibana dashboard. If you are using Google Kubernetes Engine, then stackdriver is a default logging option for GKE.

Upgrading Kubernetes

Upgrading the Kubernetes cluster is completely dependent on the platform. Here the platform is the type of installation you have followed for installing the Kubernetes Cluster. The solutions consist Google Cloud Engine, Google Kubernetes Engine, KOPS (Kubernetes Operations), CoreOS tectonic and Kubespray.

Apart from this, there are multiple solutions available including independent solution, hosted solution, and cloud-based solutions.

Upgrading Google Compute Engine Clusters:

**If the cluster is created with cluster/gce/upgrade.sh script. To upgrade the
master for specific version:**

```
cluster/gce/upgrade.sh -M v1.0.2
```

Upgrade the entire cluster to the recent stable version:

```
cluster/gce/upgrade.sh release/stable
```

Upgrading Google Kubernetes Engine Clusters:

Google Kubernetes Engine automatically updates the master components. Example: kube-API server, kube-scheduler. It is also responsible for upgrading the operating system and other master components.

Upgrade Cluster of the Kubespray:

Kubespray provides ansible based the upgrade-cluster role. It consists the following YAML file.

Executing the following role will upgrade the components of the Kubernetes cluster.

upgrade-cluster.yml

```
   - gh-aotshtes:r_lofaccatlsh:oFsatlse


            roles:kubespray-defaults}

- { role: bastion-ssh-config, tags: ["localhost",  "bastion"]}
- ahnoys_tes:rrko8rss-_cflautsatle:r":e{{tacndy:c_aelrircoor-sr_rfatal | default(true) }}"
```

```
  gvarths:er_facts: false
re#quNireetdtytoindsisuadbolerpsispeetl,inwihnigchfomr
baokoestsptriappe-lionsinasgsome systems have
ca#nfbaeil.enboabotlestdr.ap-os fixes this on these systems, so in later
plays it roalnessi:ble_ssh_pipelining: false
    - { role: kbuoobtesstprarapy-o-dse, ftaugslt:sb} ootstrap-os}

- ahnoys_tes:rrko8rss-_cflautsatle:r":{e{tacndy:c_aelrircoor-
  sr_rfatal | default(true) }}" vanrss:ible_ssh_pipelining:
  true



- ahnoys_tes:rrko8rss-_cflautsatle:r":{e{tacndy:c_aelrircoor-
  sr_rfatal | default(true) }}" sroerleiasl:: "{{ serial |
  default('20%') }}"

    - { role: kuberpnreateys-d/perfeaiunlststa} ll, tags: preinstall }
    - {rorloel:er: kdtocker, tags: docker }
      twahges:nr:k"t'rkt' in [etcd_deployment_type,
kubelet_deployment_type, va-u{ltr_odlep:
dlooywmnelnoat_dt,ytpaeg]s": download, skip_downloads: false }

- ahnoys_tes:rreotcrsd_:kfa8tsa-lc:l"u{s{ taenry:v_aeurrltors_fatal | default(true) }}"
```

```
    - { role: kvauublet,sptargasy:-dveafualut,ltvsa,uwlth_ebno:o"tcsetrat_pm: tarnuae,gewmhen:t
      "=c=er'vta_umlta'"n}-

- ahnoys_tes:rreotcrsd_fatal: "{{ any_errors_fatal | default(true) }}"

    - { role: etcd, tags: etcd, etcd_cluster_setup: true }

- ahnoys_tes:rrko8rss-_cflautsatle:r"{{ any_errors_fatal | default(true) }}"

    - { role: etcd, tags: etcd, etcd_cluster_setup: false }

- ahnoys_tes:rreotcrsd_:kfa8tsa-lc:l"u{s{ taenry:v_aeurrltors_fatal | default(true) }}"
  ro- l{erso:le: kubespray-defaults, when: "cert_management == 'vault'"}

#coHmanpdalte.  upgrades to master components first to maintain backwards

- ahnoys_tes:rrkourbs_e-fmatals:te"{r{ any_errors_fatal |
  default(true) }}" sroerleiasl:: 1

    - { role: kuupbgreasdpera/pyr-de-eufapuglrtasd} e, tags: pre-upgrade }
    - { role: kubernetes/nmoadsete, rt,atgasg: sn:omdaes}ter }
    - { role: kubernetes/-calpiepns/tc,ltuasgtse:rc_lriolnets,} tags: cluster-roles }
  - { role: nupetgwraodrke/_ppolustg-iunp,gtraagdse: ,nteatgwso: rpkos}t-upgrade }

#- Fhionsatsll:ykhuabned-nleodweo:r!kuerbeu-pmgarastdeers, based
 on given batch size asenryi_aelr: r"o{{rse_rfaiatal
 l|:d"e{{faaunlyt_(e'2r0r%or')s_}}f"atal | default(true) }}"

 ro- l{erso:le: kubespray-defaults}
  - { role: ukupbgreardne/tperse/n-uopdger,atadges, :tangosd:ep}re-upgrade }


  - { role: kubernetes/kubeadm, tags: kubeadm, when: "kubeadm_en-

  - { role: kubespray-defaults}

- ahnoys_tes:rrkourbs_e-fmatals:tetrru[0e]
 ro- l{erso:le: kubespray-defaults}
"se-c{rreotl_ec:hkaunbgeerdn|deteefsa-ualpt(pfsa/lrsoet)a" t}e_tokens, tags:
rotate_tokens, when:

- ahnoys_tes:rrkourbs_e-fmatals:tetrrue
 ro- l{erso:le: kubespray-defaults}
  - { role: kubernetes-apps/npoetliwcoy_rcko_nptluroglilne,r,tataggss:
  :npeotwlicoyr-kco} ntroller }


- hosts: calico-rr
```

```
  aronlye_se:rrors_fatal: "{{ any_errors_fatal | default(true) }}"
   - { role: knuetbweosprkra_yp-lduegfianu/cltasl}ico/rr, tags: network }

- ahnoys_tes:rrko8rss-_cflautsatle:r"{{ any_errors_fatal |
  default(true) }}" ro- l{ersoːle: kubespray-defaults}
dn-s{mroalseq: }dnsmasq, when: "dns_mode ==
'dnsmasq_kubedns'", tags: so-lv{croonlef_:
mkuobdeern==et'ehso/sptr_erienssotlavlcl,ownfh'e",nt:a"gdsn:
rs_esmoolvdceo!n=f'}none' and re-

- ahnoys_tes:rrkourbs_e-fmatals:te"{r{[0a]ny_errors_fatal | default(true) }}"



   - { role: kubernetes-apps, tags: apps }
```

It will upgrade the Kubernetes Kubespray installation of the Kubernetes Cluster.

Kubernetes Maintenance

Monitoring Kubernetes Cluster

For reliable applications, it is required to have in place monitoring of the Kubernetes Cluster. It helps to determine availability, scalability, and reliability of the application deployed over Kubernetes Cluster.

Heapster aggregator is used for monitoring and event the logs. Heapster stores the information in storage backend. Currently, it supports Google Cloud Monitoring and InfluxDB as the storage backends. Heapster runs as a pod in the Kubernetes Cluster. It communicates with each node of the Kubernetes Cluster. Kubelet agent is responsible to provide the monitoring information to Heapster. Kubelet itself collects the data from cAdvisor.

cAdvisor:

cAdvisor is an open source container usage and performance analysis agent. In Kubernetes cAdvisor is included into Kubelet binary.cAdvisor auto-discovers all containers. It collects the information of CPU, memory, network and file system usage statistics.

Kubelet:

Kubelet bridge the gap between Kubernetes Master and Kubernetes Node. It manages the Pods and Containers on each machine.

InfluxDB and Grafana are used for storing the data and visualizing it. Google cloud monitoring provides the hosted solution for monitoring Kubernetes Cluster. Heapster can be set up to send the metrics to Google Cloud monitoring.

Let's check the Monitoring Kubernetes Cluster created with

Minikube: Kubernetes Cluster created locally by minikube

supports addons.

```
$- admdadinoink-umbaenaadgndeorbn: senliasbt led

        - kdusbheb-ldonasrd:oe: neaebalelded   enabled

      - hinefaapusstte-rs:tidriasgalbecldeladss:

          - regisetrsy::ddissaabbleddisabled

                egistry-creds:
```

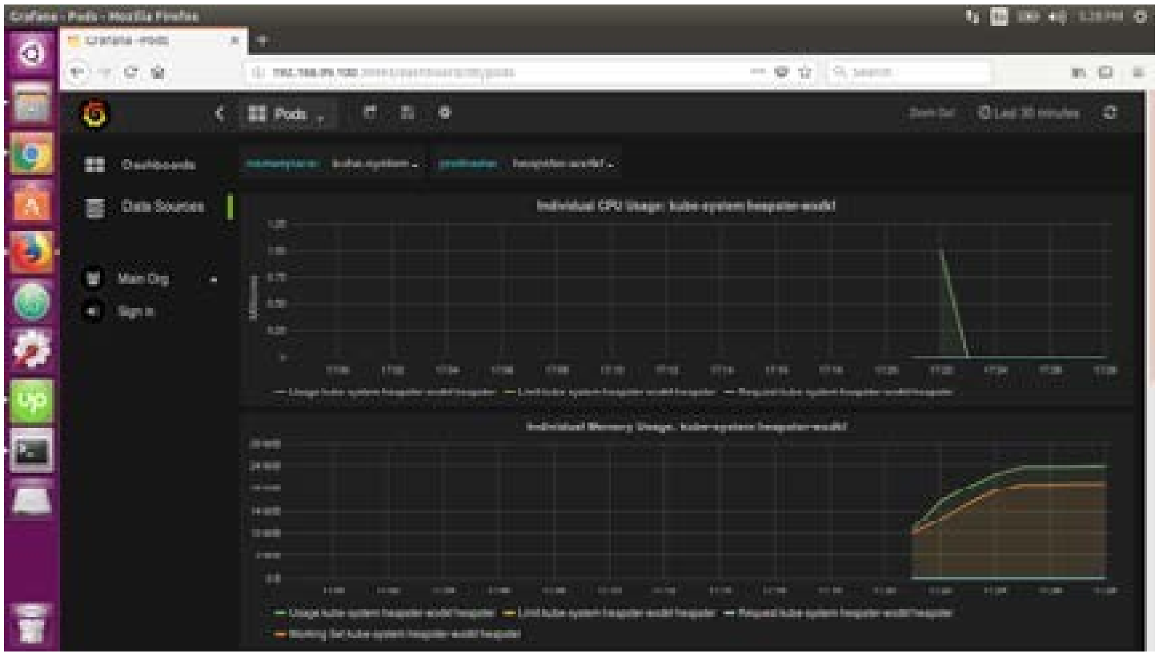
Enable the addon:

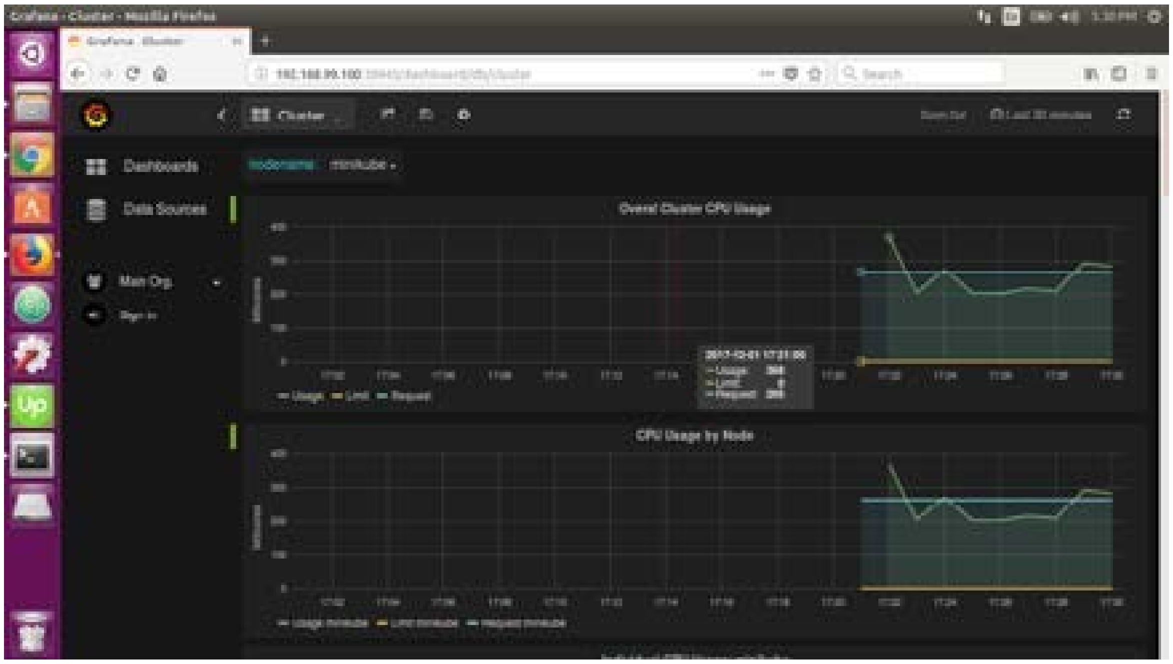
```
$ minikube addons enable heapster
```

To open the web interface:

```
$ minikube addons open heapster
```

The result will be displayed on the grafana.


Grafana - Pods - Mozilla Firefox
Pods
Dashboards
Data Sources
Main Org
Sign In
Individual CPU Usage: kube-system heapster
Individual Memory Usage: kube-system heapster



In Minikube addons are helping for monitoring but it’s also possible to add heapster as Kubernetes deployment. This will be the manual installation of heapster, grafana and influxdb.

Following is the heapster.yaml :

```
akpiniVde:
rSseirovni:cve1Accou
nt mneatmadea:thae:
apster
--n- amespace: kube-system
akpiniVde: rDseiopnlo:
yemxtentsions/v1beta1


  namespace: kube-system

  r  plic    : 1

    metadata:
     latabsekls::monitor
    ing spke8cs: -app:
    heapster
      sceornvtiacienAecrcs:ountName: heapster
      - inmaamgee:: hgecar.pios/tgeorogle_containers/heapster-
        amd64:v1.4.0 icmomagmePaunldlP: olicy: IfNotPresent

        - /--hseoauprsctee=rkubernetes:https://kubernetes.default
s--v-c:8- 0--8s6ink=influxdb:http://monitoring-influxdb.kube-
system. akpiniVde: rSseirovni:cve1
mlaebtaedlsa:ta:
    t#aFsko:r musoenaitsoariCnlguster add-on
(https://github.com/kubernetes/ ku#bIefrynoeuteasr/terNeeO/mT
ausstienrg/ctlhuisstaesr/anddaodndso)n, you should comment out
thkisulbineern. etes.io/cluster-service: 'true'


  namespace: kube-system

  ports:

    targetPort: 8082

    k8s-app: heapster
```

You can get the latest version of the heapster at https://github.com/ kubernetes/heapster/ .

Using Kubectl:

```
$ kubectl create -f heapster.yaml
```

For grafana, use grafana.yaml:

```
akpiniVde: rDseiopnlo:
yemxtentsions/v1beta1
mneatmadea:tma:onitoring-
grafana spneacm: espace:
kube-system rtemplpiclate: :1
    mlaebtaedlsa:ta:
        tka8ssk-a:
    pmpo:ngirtaofraina
    g spcoenc:tainers:
      - inmaamgee:: ggrcarf.iaon/gaoogle_containers/heapster-grafana-
        amd64:v4.4.3 p- coortnst:ainerPort: 3000

        vpolruomtoecMolo: uTnCtPs:
        - nmaomuen:tcPa-tche:r/teitficc/asstel/scerts
        - nmaomuen:tgPraathfa:n/vaa-srtorage
        e- nva:me: INFLUXDB_HOST
        - vnaalmuee::mGFo_nSiEtoRrVinEgR-i_nHflTuTxPd_bPORT
          v#aTluhe f"o3l0lo0w0"ing env variables are required to make
Grafana acces#sitbhle kvuiabernetes api-server proxy. On
production clusters, we recom#mreemnodving these env variables,
setup auth for grafana, and expos#estehrevigcreafuasninag a
LoadBalancer or a public IP.
        - vnaalmuee::"GfaFl_sAe"UTH_BASIC_ENABLED
        - vnaalmuee::"GtrFu_eA"UTH_ANONYMOUS_ENABLED
        - vnaalmuee::AGdFm_AinUTH_ANONYMOUS_ORG_ROLE
        - #naIfmyeo:uG'rFe_oSnElRyVuEsRin_gRtOhOeTA_PUIRSLerver proxy, set
this value instead: grafa#nav/aplruoex: y/api/v1/namespaces/kube-
system/services/monitoring-

      vovlualmues:/
      - hnoasmtPea: tcha:-certificates
      - npaamthe:: /gertacf/assnla/c-setrotrsage

---
        emptyDir: {}
apiVersion: v1
```

```
kminetda:dSaetrav:ice
  la#bFeolsr:use as a Cluster add-on (https://github.com/kubernetes/
ku#bIefrynoeuteasr/terNeeO/mT
ausstienrg/ctlhuisstaesr/anddaodndso)n, you should comment out
thkisulbineern. etes.io/cluster-service: 'true'
  nkaumbee:rmneotnesit.oior/ingm-gera:
fmanoanitoring-grafana spneacm: espace: kube-
system
a#n Ienxtaeprnroadl uLcotaidobnasleatnucpe,rwe recommend
  accessing Grafana through # otyrpteh:rLoouagdhBaaplaunbcleicr IP.
g#enYeoruatceoduplдoratlso use NodePort to expose the service at a
  randomly- #potyrtpse:: NodePort
  -
  tpaorrgte:t8P0ort:
  3000 sekl8esc-
  taoprp: : grafana
```

You can get the latest version of grafana.yaml at https://github.com/ kubernetes/heapster/blob/master/deploy/kube-config/influxdb/grafana. yaml .

Using kubectl :

```
$ kubectl create -f grafana.yaml
```

If influxdb is the storage backend, then use following YAML for deploying influxdb in Kubernetes Cluster:

```
akpiniVde: rDseiopnlo:
yemxtentsions/v1beta1
mneatmadea:tma:onitoring-
influxdb spneacm: espace:
kube-system rtemplpiclate: :1
  mlaebtaedlsa:ta:
    tka8ssk-a:
  pmpo:ninitflourixndgb
  spcoenc:tainers:
   - inmaamgee:: igncflr.uiox/dgboogle_containers/heapster-influxdb-
     amd64:v1.3.3 v- omluomunetMPaotuhn: t/sd:ata
   vonluamees::influxdb-storage

   - enmamptey:Dinirfl: u{}xdb-storage


akpiniVde:
rSseirovni:cve
1
mlaebtaedlsa:t
a:

  t#aFsko:r musoenaitsoariCnlguster add-on (https://github.com/kubernetes/
```

```
        th#isIlfinyoe.u are NOT using this as an addon, you should comment out

kube:rmneotnesit.oior/icnlaumstee:r-mseornviitcoer:in'trgu-ien' fluxdb

    snacm: espace: kubeg-s-iynsifeumxdb


    p- poortrst:: 8086

    setalergcetotPr:ort: 8086

         k8s-app: influxdb
```

You can get the latest version of influxdb.yaml at https://github.com/ kubernetes/heapster/blob/master/deploy/kube-config/influxdb/influxdb. yaml

Using Kubectl:

```
$ kubectl create -f influxdb.yaml
```

To access grafana dashboard with the manual setup, describe the grafana service and check endpoint of the service.

To describe the service using Kubectl use following command:

```
  mSadeildneicktourb:e-addons...  rnetes.io/mode=Recon-

cTiyle,onnammaen=aingeflru.kxuGbraefaNnoadePort

    IPP:pe:
                          1<0u.n0s.0e.t6>2
     ort:                                        80/TCP
```

```
NESenodspePoionrtts::     <17u2n.s1e7t.>0.9:3000      30943/TCP
                          <none>
   Evsenitosn:  Affinity
```

Prometheus and Data Dog are also good tools for monitoring the Kubernetes Cluster.

Managing Kubernetes with Dashboard

Kubernetes provides the web interface to manage the Kubernetes Cluster. With the Role Based Access Policy, it becomes simpler and easier to manage Kubernetes Cluster with Dashboard. For Minikube it gives the addon. That means Minikube provides addons for monitoring, dashboard and for managing the cluster.

To enable the addon of the dashboard on minikube:

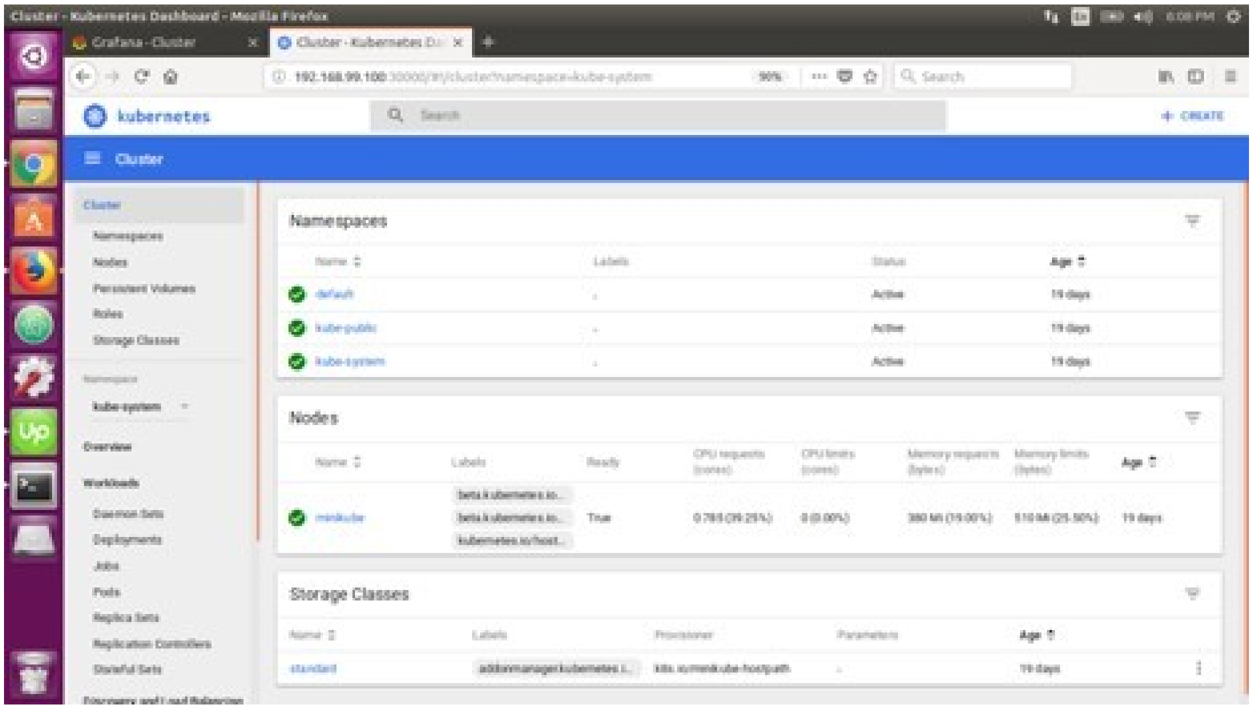
```
damshinbiokaurbdewaadsdsouncsceenssafbullelydeanshableadrd

$Opmeinniinkgubkeubaedrdnoentsesopseenvidcaeshkubboea-rsdystem/kubernetes-dashboard in

     default browser...
```

With Minikube, it will open the dashboard in the browser. The dashboard consists the Cluster infromation, Namespace information and information about the objects associated with cluster like workloads etc.The dashboard consists the three different informative sections including workloads, discovery and load balancing, config and storage.

In the cluster tab, Kubernetes dashboard shows the information about the Namespaces, Nodes, Persistent Volumes and Storage Classes.



Namespaces

Name	Labels	Status	Age
default	-	Active	19 days
kube-public	-	Active	19 days
kube-system	-	Active	19 days

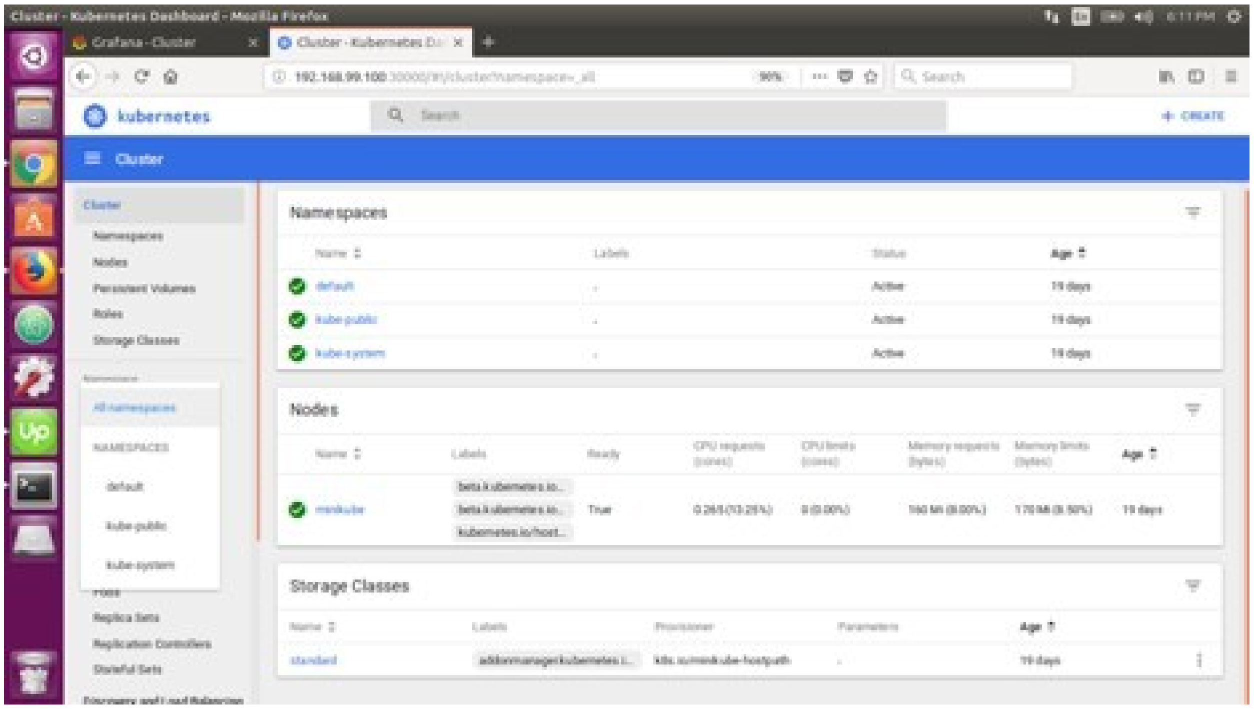
Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
minikube	beta.kubernetes.io... beta.kubernetes.io... kubernetes.io/host...	True	0.785 (39.25%)	0 (0.00%)	380 Mi (19.00%)	510 Mi (25.50%)	19 days

Storage Classes

Name	Labels	Provisioner	Parameters	Age
standard	addonmanager.kubernetes.i...	k8s.io/minikube-hostpath	-	19 days

It gives the brief information about the cluster. In namespaces, it shows the all available namespaces in Kubernetes Cluster. It's possible to select all namespaces or specific namespace



Namespaces

Name	Labels	Status	Age
default	-	Active	19 days
kube-public	-	Active	19 days
kube-system	-	Active	19 days

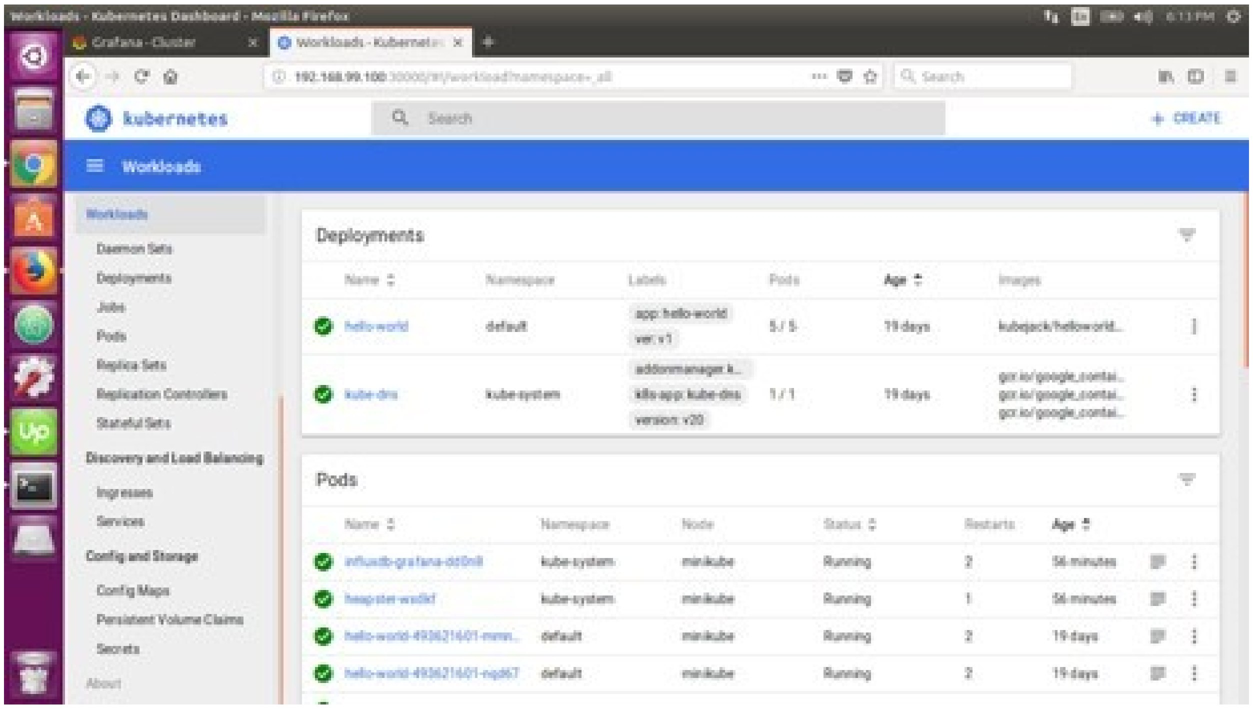
Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
minikube	beta.kubernetes.io... beta.kubernetes.io... kubernetes.io/host...	True	0.265 (13.25%)	0 (0.00%)	160 Mi (8.00%)	170 Mi (8.50%)	19 days

Storage Classes

Name	Labels	Provisioner	Parameters	Age
standard	addonmanager.kubernetes.i...	k8s.io/minikube-hostpath	-	19 days

Depending on the namespace selected previously, further tabs show in depth information of associated Kubernetes object within selected namespace. It consists the workloads of Kubernetes which includes
Deployments, Replica Sets, Replication Controller, Daemon Sets, Jobs, Pods and Stateful Sets. Each of this workload is important to run an application over the Kubernetes Cluster.



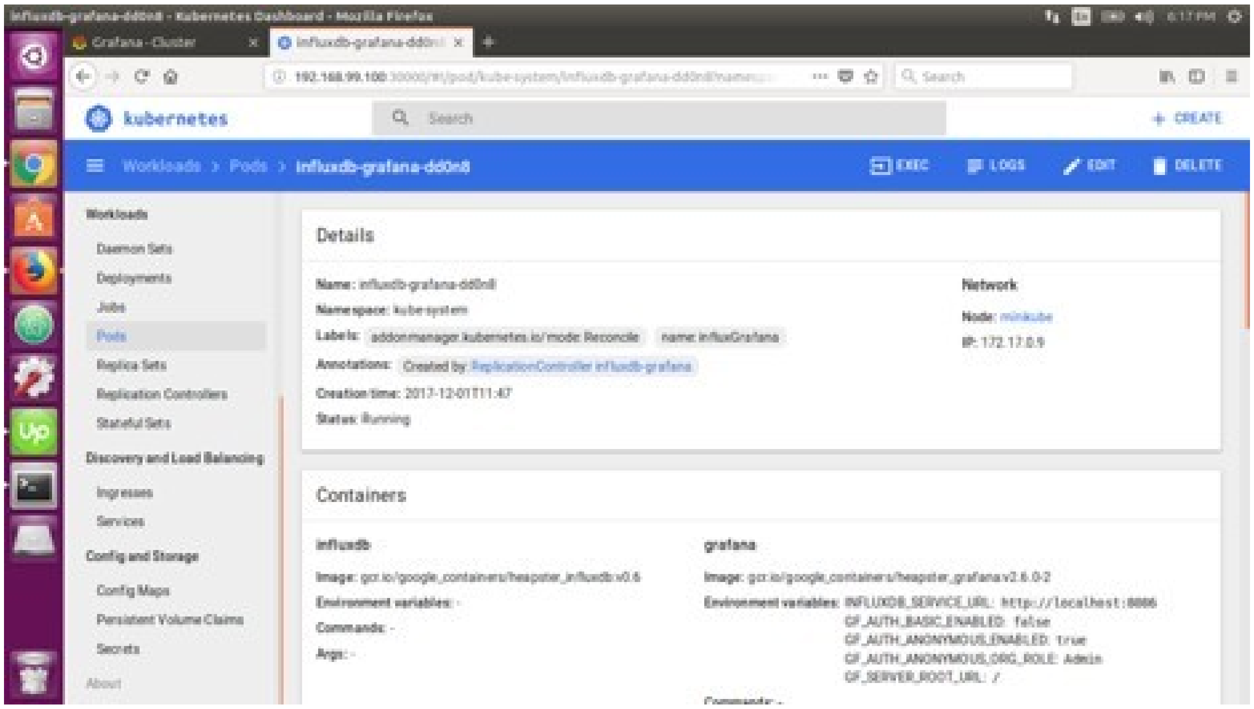
Deployments

Name	Namespace	Labels	Pods	Age	Images
hello-world	default	app: hello-world ver: v1	5 / 5	19 days	kubejack/helloworld...
kube-dns	kube-system	addonmanager.k... k8s-app: kube-dns version: v20	1 / 1	19 days	gcr.io/google_contai... gcr.io/google_contai... gcr.io/google_contai...

Pods

Name	Namespace	Node	Status	Restarts	Age
influxdb-grafana-dd0n8	kube-system	minikube	Running	2	56 minutes
heapster-wxdkf	kube-system	minikube	Running	1	56 minutes
hello-world-493621601-mmn...	default	minikube	Running	2	19 days
hello-world-493621601-nqd67	default	minikube	Running	2	19 days

The Dashboard also gives the information of each workload. Following screenshot describes the Kubernetes Pod in detail.



Details

Name: influxdb-grafana-dd0n8
Namespace: kube-system
Labels: addonmanager.kubernetes.io/mode: Reconcile name: influxGrafana
Annotations: Created by: ReplicationController influxdb-grafana
Creation time: 2017-12-01T11:47
Status: Running

Network
Node: minikube
IP: 172.17.0.9

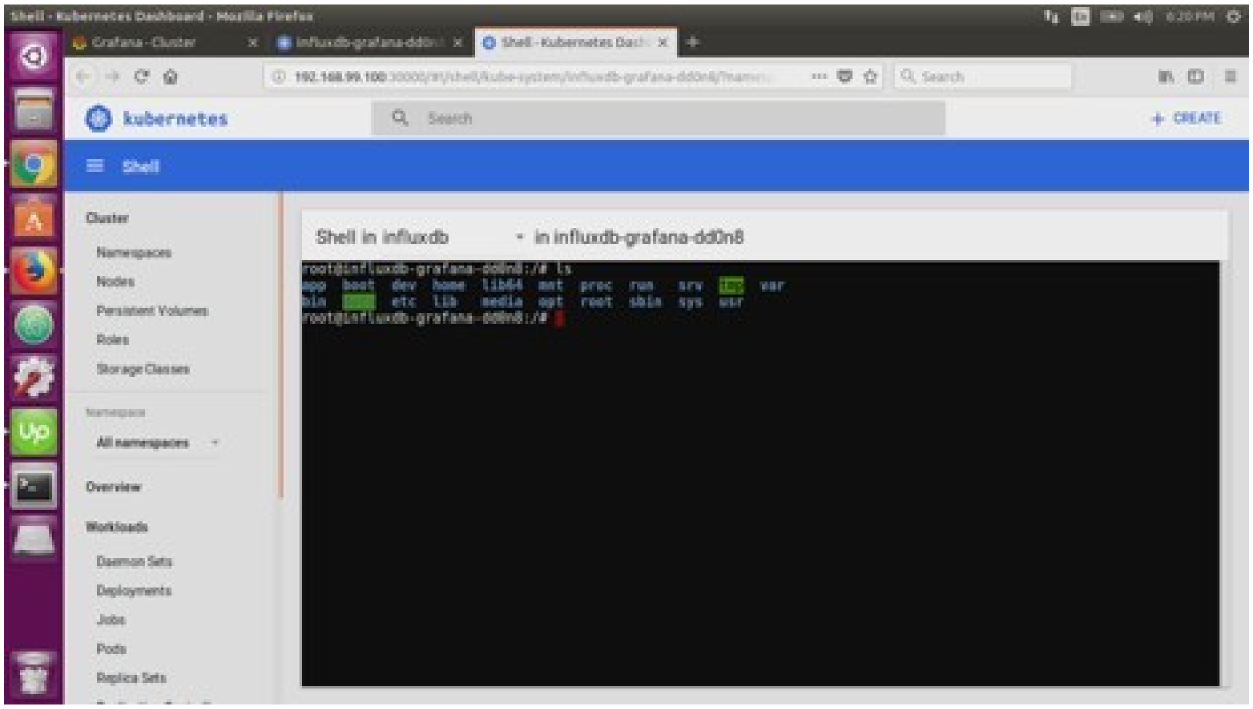
Containers

influxdb
Image: gcr.io/google_containers/heapster_influxdb:v0.6
Environment variables: -
Commands: -
Args: -

grafana
Image: gcr.io/google_containers/heapster_grafana:v2.6.0-2
Environment variables: INFLUXDB_SERVICE_URL: http://localhost:8086
GF_AUTH_BASIC_ENABLED: false
GF_AUTH_ANONYMOUS_ENABLED: true
GF_AUTH_ANONYMOUS_ORG_ROLE: Admin
GF_SERVER_ROOT_URL: /
Commands: -

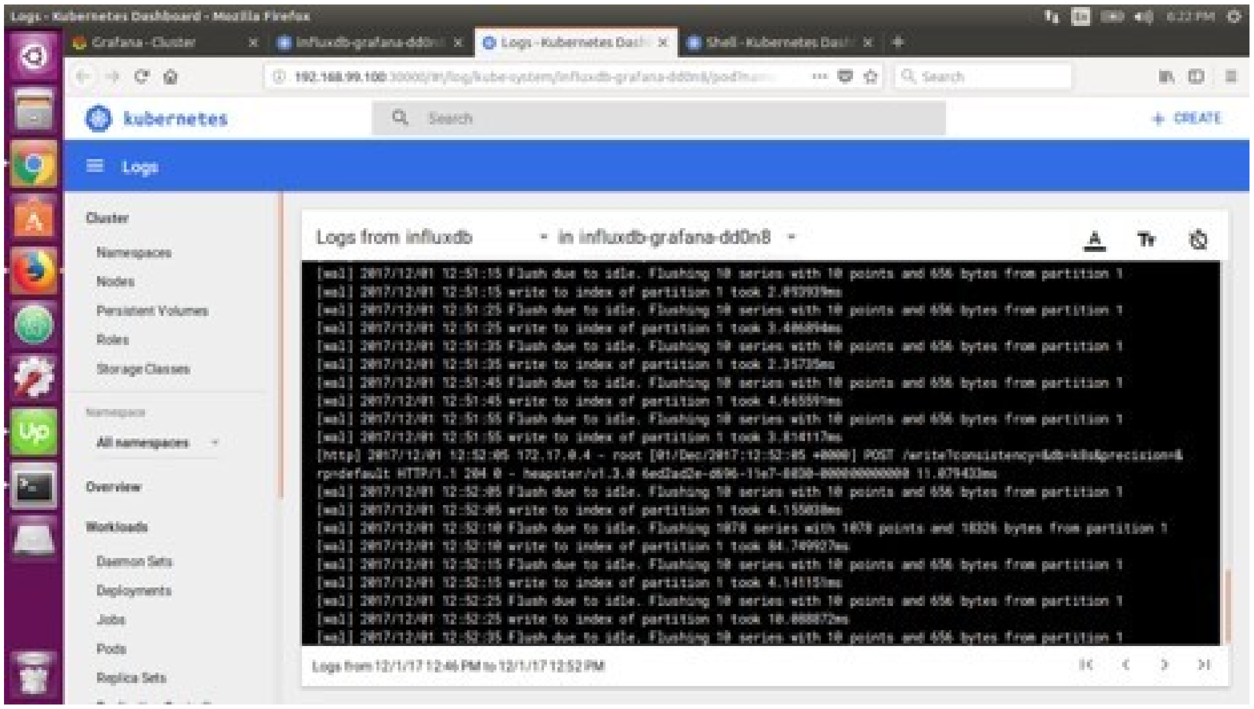
But the dashboard is just not limited to information, it's even possible to execute in the pod, get the logs of the pod, edit the pod and delete the pod.

Exec inside the influxdb-grafana pod:



```
root@influxdb-grafana-dd0n8:/# ls
app  boot  dev  home  lib64  mnt  proc  run  srv  tmp  var
bin  data  etc  lib   media  opt  root  sbin  sys  usr
root@influxdb-grafana-dd0n8:/#
```

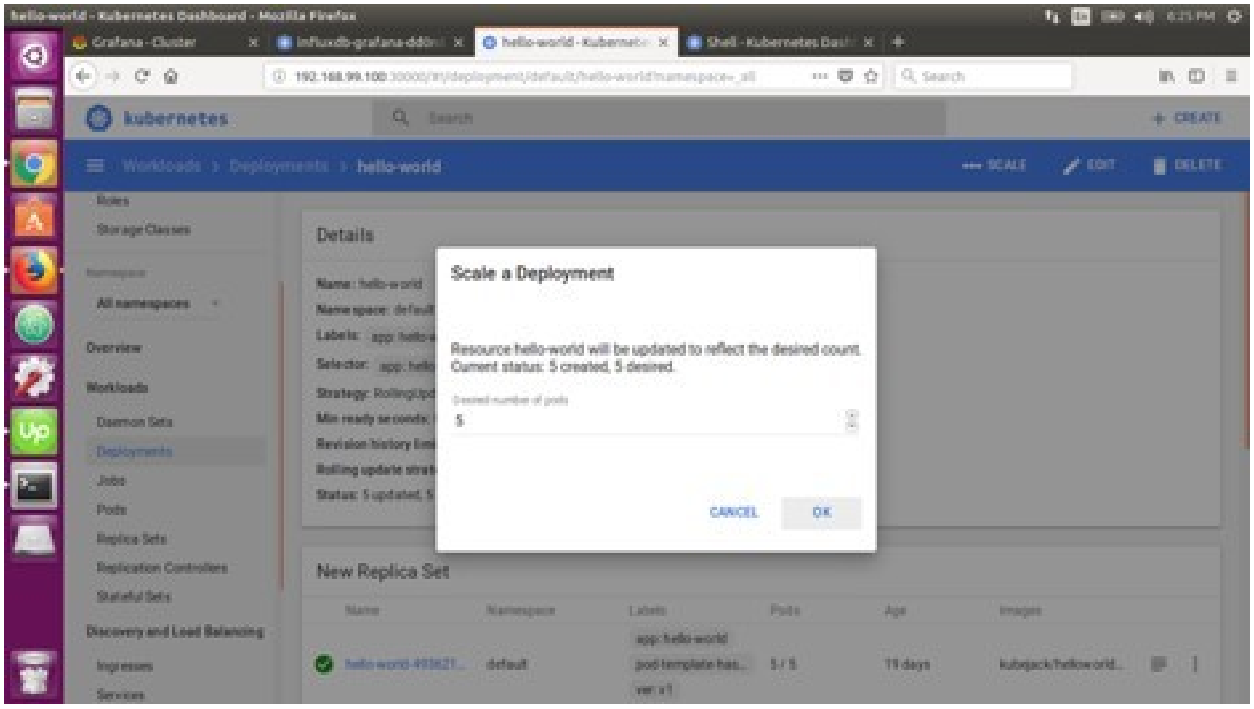
Logs of the influxdb-grafana pod:



```
[wal] 2017/12/01 12:51:15 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:15 write to index of partition 1 took 2.093939ms
[wal] 2017/12/01 12:51:25 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:25 write to index of partition 1 took 3.486894ms
[wal] 2017/12/01 12:51:35 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:35 write to index of partition 1 took 2.35735ms
[wal] 2017/12/01 12:51:45 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:45 write to index of partition 1 took 4.665591ms
[wal] 2017/12/01 12:51:55 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:51:55 write to index of partition 1 took 3.814117ms
[http] 2017/12/01 12:52:05 172.17.0.4 - root [01/Dec/2017:12:52:05 +0000] POST /write?consistency=&db=k8s&precision=&
rp=default HTTP/1.1 204 0 - heapster/v1.3.0 6ed2ad2e-d696-11e7-8030-000000000000 11.079433ms
[wal] 2017/12/01 12:52:05 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:05 write to index of partition 1 took 4.155038ms
[wal] 2017/12/01 12:52:10 Flush due to idle. Flushing 1078 series with 1078 points and 18326 bytes from partition 1
[wal] 2017/12/01 12:52:10 write to index of partition 1 took 84.749927ms
[wal] 2017/12/01 12:52:15 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:15 write to index of partition 1 took 4.141151ms
[wal] 2017/12/01 12:52:25 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
[wal] 2017/12/01 12:52:25 write to index of partition 1 took 10.080872ms
[wal] 2017/12/01 12:52:35 Flush due to idle. Flushing 10 series with 10 points and 656 bytes from partition 1
```

Logs from 12/1/17 12:46 PM to 12/1/17 12:52 PM

This is only for Pod. For other objects like deployments, it's possible to scale, edit and delete the deployment.



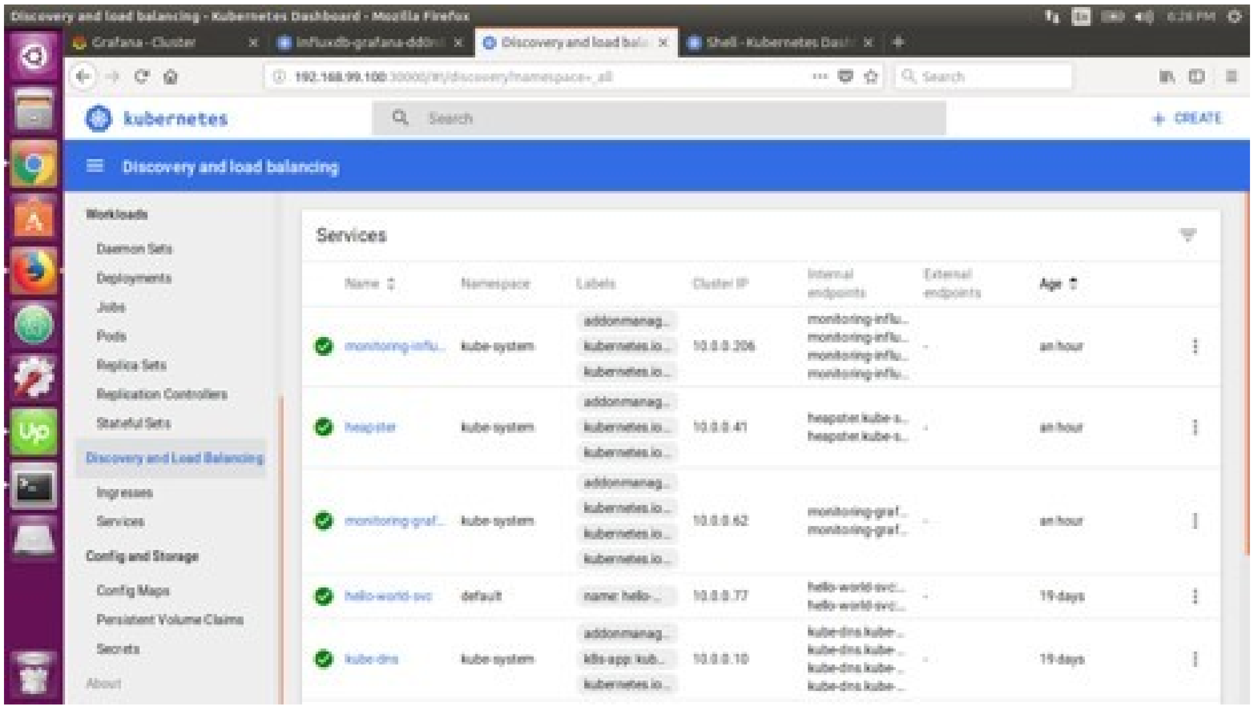
Scale a Deployment

Resource hello-world will be updated to reflect the desired count.
Current status: 5 created, 5 desired.

Desired number of pods: 5

hello-world - Kubernetes Dashboard - Mozilla Firefox
kubernetes
Workloads > Deployments > hello-world
SCALE
EDIT
DELETE
CREATE
Details
New Replica Set
CANCEL
OK

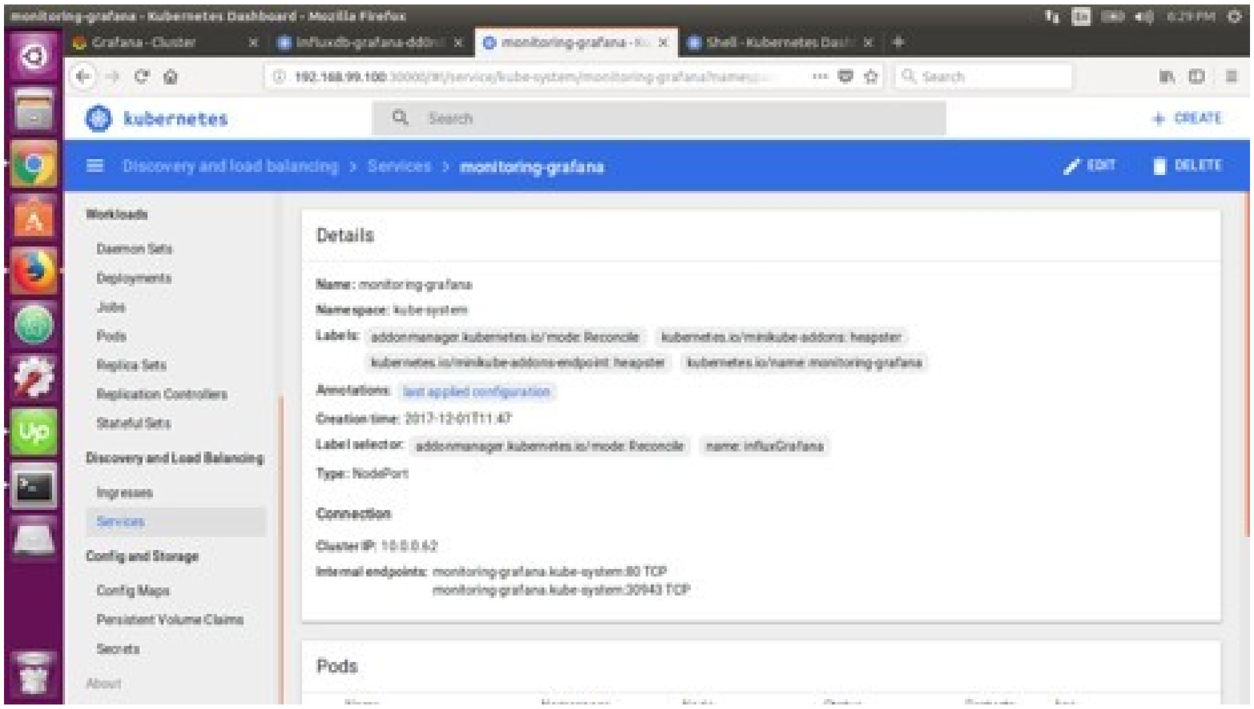
Discovery and Load Balancing consists the information about the Ingresses and Services.



Services

Name	Namespace	Labels	Cluster IP	Internal endpoints	External endpoints	Age
monitoring-influ...	kube-system	addonmanag... kubernetes.io... kubernetes.io...	10.0.0.206	monitoring-influ... monitoring-influ... monitoring-influ... monitoring-influ...	-	an hour
heapster	kube-system	addonmanag... kubernetes.io... kubernetes.io...	10.0.0.41	heapster.kube-s... heapster.kube-s...	-	an hour
monitoring-graf...	kube-system	addonmanag... kubernetes.io... kubernetes.io... kubernetes.io...	10.0.0.62	monitoring-graf... monitoring-graf...	-	an hour
hello-world-svc	default	name: hello-...	10.0.0.77	hello-world-svc... hello-world-svc...	-	19 days
kube-dns	kube-system	addonmanag... k8s-app: kub... kubernetes.io...	10.0.0.10	kube-dns.kube-... kube-dns.kube-... kube-dns.kube-... kube-dns.kube-...	-	19 days

The dashboard also provides the detail information about Ingresses and Services. It's possible to edit and delete the ingress or service through the dashboard.



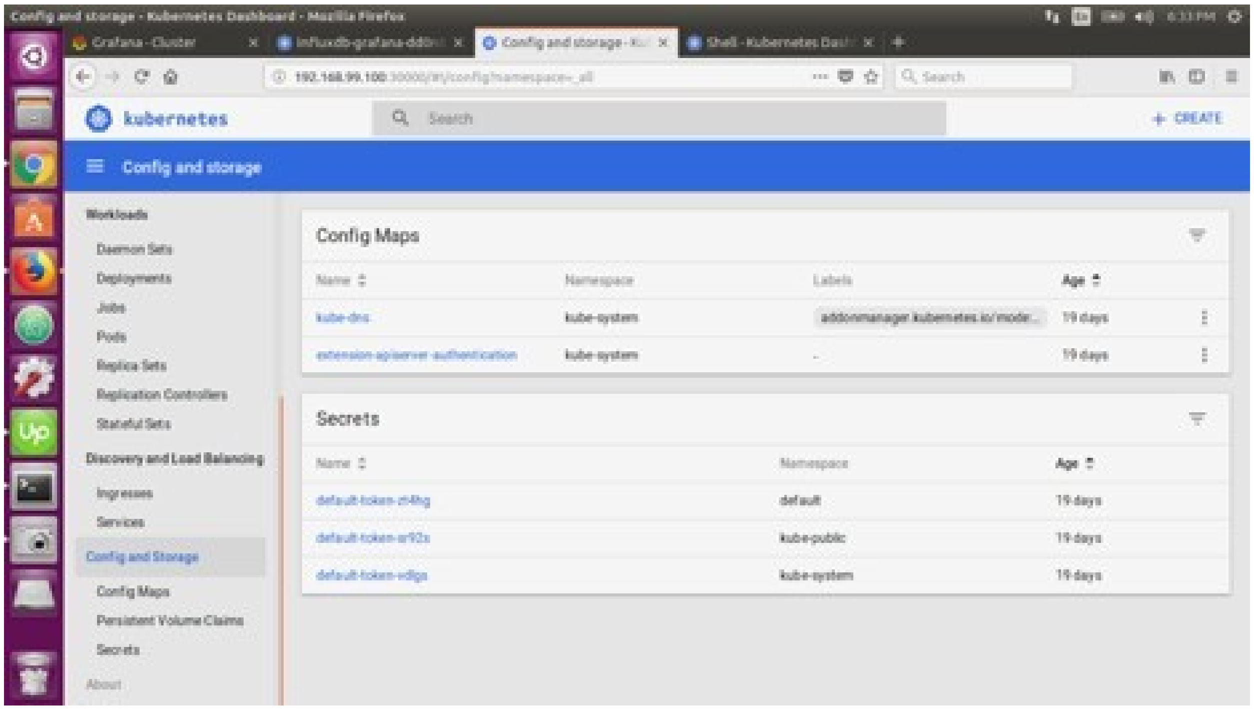
Details

Name: monitoring-grafana
Namespace: kube-system
Labels: addonmanager.kubernetes.io/mode: Reconcile kubernetes.io/minikube-addons: heapster kubernetes.io/minikube-addons-endpoint: heapster kubernetes.io/name: monitoring-grafana
Annotations: last applied configuration
Creation time: 2017-12-01T11:47
Label selector: addonmanager.kubernetes.io/mode: Reconcile name: influxGrafana
Type: NodePort

Connection

Cluster IP: 10.0.0.62
Internal endpoints: monitoring-grafana.kube-system:80 TCP
monitoring-grafana.kube-system:30943 TCP

Config and Storage consist the information about the Config Maps, Persistent Volume Claims and Secrets.



Config Maps

Name	Namespace	Labels	Age
kube-dns	kube-system	addonmanager.kubernetes.io/mode:...	19 days
extension-apiserver-authentication	kube-system	-	19 days

Secrets

Name	Namespace	Age
default-token-zl4hg	default	19 days
default-token-sr92s	kube-public	19 days
default-token-vdlgs	kube-system	19 days

Also, it's possible to directly deploy the containerized application through web UI. The dashboard will require App name, Container Image, Number of Pod and Service inputs. Service can be internal or external service. The YAML file with required specifications can also be used for the creating the Kubernetes object.

example, Minikube addon is used for the Kubernetes dashboard.For manual installation following YAML sho

```
Kdausbhebcotlacrdre/mateas-tfehr/tstprcs/:d/ erapwlo.yg/irthecuobmusmerecnodnetde/nktu.d
yaml
```

To access the Web

UI, Using Kubectl

Proxy:

```
kubectl proxy
```

Using Kubernetes Master API Server: https://<kubernetes-master>/ui

If the username and password are configured and unknown to you then use,

```
kubectl config view
```

Kubernetes dashboard is a flexible and reliable way to manage the Kubernetes Cluster.

Logging Kubernetes Cluster

Application and system level logs are useful to understand the problem with the system. It helps with troubleshooting and finding the root cause of the problem. Like application and system level logs containerized application also requires logs to be recorded and stored somewhere.
The most standard method used for the logging is to write it to standard output and standard error streams. When the logs are recorded with separate storage then the mechanism is called as Cluster Level Logging.

Basic Logging with Kubernetes:

In the most basic logging it's possible to write the logs to the standard output using the Pod specification.

For Example:

```
akmpineiVde: rPsoiodn: v1

spnatmadea:tcao:unter

c- oenc:tmaien:ecrosu: nt

    inmaage: busybox

  args':i=[/0b;iwn/hsihle, -tcr,ue; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']
```

The logs will be recorded with standard output:

```
$p$ okdub"ceocutlnctreera" tcere-faltoegd-example-pod.yaml

    kubectl get po
```

```
NcoAu                    R1/E1ADY  SRTuAnTnUinSg          R0 ESTARTS
                                                   A8sGE
MntEer

h1ffelrlpo-world-            1/1      Running       3          19d

493621601-   hmemllon-       1/1      Running       3          19d

zwworld-493621601-           1/1      Running       3          19d

hneqldlo67-world-            1/1      Running       3          19d

493621601-   hqkelflcox-     1/1      Running       3          19d

world-493621601-

hxbelfl6os-world-

493621601-

$0:kFurbi eDcetcl
1
lo1g1s6c:3o7u:n36teUr TC 2017
3
2: Fri Dec 1 16:37:38 UTC

2017

4: Fri Dec  1 16:37:40 UTC 2017
5
6: Fri Dec  1 16:37:42 UTC 2017
7
8: Fri Dec  1 16:37:44 UTC 2017
```

Node level logging with Kubernetes:

The containerize application writes logs to stdout and stderr. Logging driver is the responsible for the writing log to the file in JSON format. In case of docker engine, docker logging driver is responsible for writing the log.

The most important part of Node level logging is log rotation with the Kubernetes. With the help of log rotation, it ensures the logs will not consume all storage space of the nodes.

Cluster Level Logging with Kubernetes:

Kubernetes does not provide the native cluster level logging.
But the cluster level logging is possible with following approaches:

1. **Run the agent on each node for log collection**
2. **Run the side container which will be responsible for log collection**
3. **Directly store the logs of the application into the storage**

The most used and recommended method is using the node agent for log collection and storing the logs in log storage.

Stackdriver or Elasticsearch is used for the logging with Kubernetes. However, there are other solutions available like logz.io, sematext logging etc. Fluentd is used with custom configuration along with Stackdriver and Elasticsearch. Fluentd acts as the node agent.

For the Kubernetes Cluster created through minikube Giantswarm provides the solution. The solution consists ELK (Elasticsearch , logstash and Kibana) stack logging with minikube. However, it is possible to deploy all these components manually with manifests.

Start the Minikube:

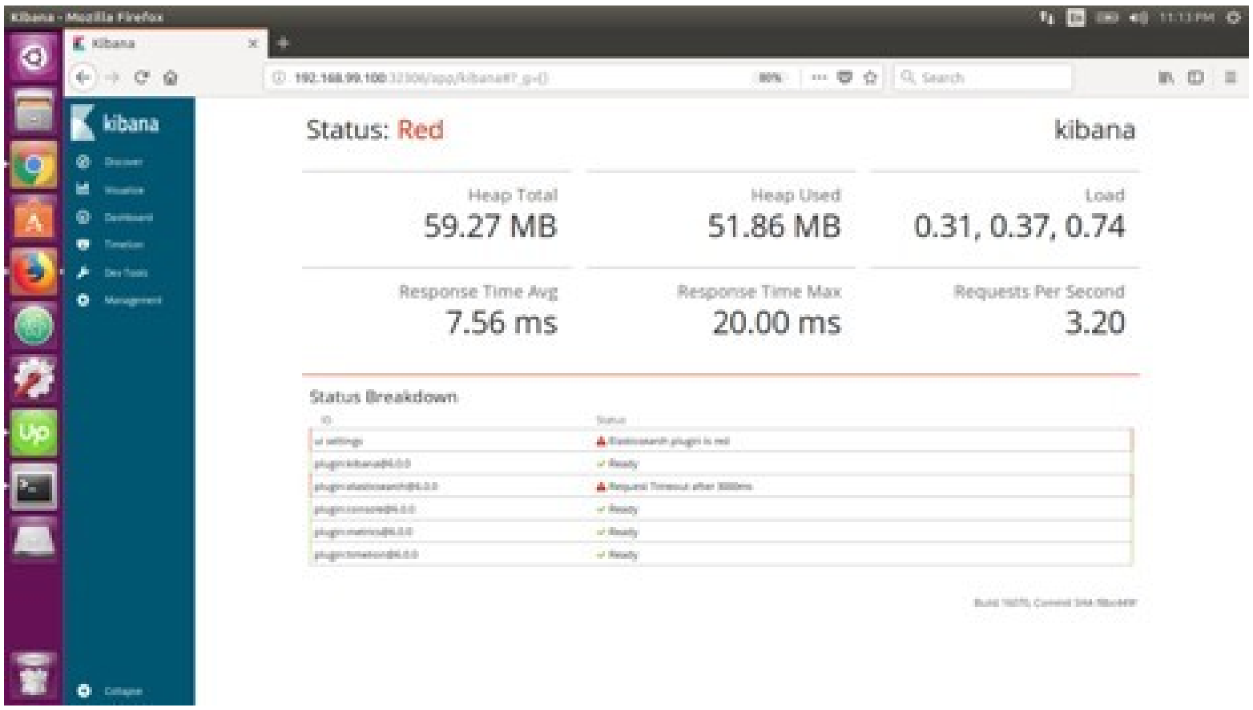
```
minikube start --memory 4096
```

Download all manifests and start Kibana:

```
k-uɨƀleecntlaampeplhytt\ps://raw.githubusercontent.com/giantswarm/

      kubernetes-elastic-stack/master/manifests-all.yaml
      minikube service kibana
```

On the Kibana Dashboard :


Status: Red

kibana

Heap Total	Heap Used	Load
59.27 MB	51.86 MB	0.31, 0.37, 0.74

Response Time Avg	Response Time Max	Requests Per Second
7.56 ms	20.00 ms	3.20

Status Breakdown

ID	Status
ui settings	Elasticsearch plugin is red
plugin:kibana@6.0.0	✓ Ready
plugin:elasticsearch@6.0.0	Request Timeout after 3000ms
plugin:console@6.0.0	✓ Ready
plugin:metrics@6.0.0	✓ Ready
plugin:timelion@6.0.0	✓ Ready

Build [illegible], Commit SHA [illegible]

The logging will be enabled and you can check it through Kibana dashboard. If you are using Google Kubernetes Engine, then stackdriver is a default logging option for GKE.

Upgrading Kubernetes

Upgrading the Kubernetes cluster is completely dependent on the platform. Here the platform is the type of installation you have followed for installing the Kubernetes Cluster. The solutions consist Google Cloud Engine, Google Kubernetes Engine, KOPS (Kubernetes Operations), CoreOS tectonic and Kubespray.

Apart from this, there are multiple solutions available including independent solution, hosted solution, and cloud-based solutions.

Upgrading Google Compute Engine Clusters:

**If the cluster is created with cluster/gce/upgrade.sh script. To upgrade the
master for specific version:**

```
cluster/gce/upgrade.sh -M v1.0.2
```

Upgrade the entire cluster to the recent stable version:

```
cluster/gce/upgrade.sh release/stable
```

Upgrading Google Kubernetes Engine Clusters:

Google Kubernetes Engine automatically updates the master components.Example: kube-API server, kube-scheduler. It is also responsible for upgrading the operating system and other master components.

Upgrade Cluster of the Kubespray:

**Kubespray provides ansible based the upgrade-cluster role. It consists the following YAML file.
Executing the following role will upgrade the components of the Kubernetes cluster.
upgrade-cluster.yml**

```
- h- osts: localhost
  groaltehse:r_facts:
  False

    - { role: kbuasbteiospnr-sasyh-d-ceofanufilgts,}tags: ["localhost", "bastion"]}

- ahnoys_tes:rrko8rss-_cflautsatle:r":{e{tacndy:c_aelrircoor-
  sr_rfatal | default(true) }}" gvarths:er_facts: false
re#quNireetdtytoindsisuadbolerpsispeetl,inwihnigchfomr
baokoestsptriappe-lionsinasgsome systems have
ca#nfbaeil.enboabotlestdr.ap-os fixes this on these systems, so in later
plays it roalnessi:ble_ssh_pipelining: false
    - { role: kbuoobtesstprarapy-o-dse, ftaugslt:sb} ootstrap-os}

- ahnoys_tes:rrko8rss-_cflautsatle:r":{e{tacndy:c_aelrircoor-
  sr_rfatal | default(true) }}" vanrss:ible_ssh_pipelining:
  true


- ahnoys_tes:rrko8rss-_cflautsatle:r":{e{tacndy:c_aelrircoor-
  sr_rfatal | default(true) }}" sroerleiasl:: "{{ serial |
  default('20%') }}"

    - { role: kuberpnreateys-d/perfeaiunlststa} ll, tags: preinstall }
  - role:

    when: "'rkt' in [e cd_deployment_type, kubelet_deployment_type,

  - { role: download, tags: download, skip_downloads: false }

- ahnoys_tes:rreotcrsd_:kfa8tsa-lc:l"u{s{ taenry:v_aeurrltors_fatal | default(true) }}"
  ro- l{erso:le: kubespray-defaults, when: "cert_management ==
'vault'" } ma- n{ arogleem: veanutl=t,=t'avgasu:lvt'a"u}lt,
vault_bootstrap: true, when: "cert_

- ahnoys_tes:rreotcrsd_fatal: "{{ any_errors_fatal | default(true) }}"


  - { role: etcd, tags: etcd, etcd_cluster_setup: true }

- ahnoys_tes:rrko8rss-_cflautsatle:r"{{ any_errors_fatal | default(true) }}"


  - { role: etcd, tags: etcd, etcd_cluster_setup: false  }

- ahnoys_tes:rreotcrsd_:kfa8tsa-lc:l"u{s{ taenry:v_aeurrltors_fatal | default(true) }}"
  ro- l{erso:le: kubespray-defaults, when: "cert_management == 'vault'"}


#coHmanpdalte.  upgrades to master components first to maintain backwards

- hosts: kube-master
```

asenryi_aelr: r1ors_fatal: "{{ any_errors_fatal |
default(true) }}" ro- l{erso:le: kubespray-defaults}
- { role: ukupbgreardne/tperse/n-uopdger,atadges, :tangosd:ep}re-upgrade }

kubernetes-apps/cluster_roles, tags: cluster-roles }

- { role: upgrade/post-upgrade, tags: post-upgrade }

#- Fhionsatsll:ykhuabned-nleodweo:r!kuerbeu-pmgarastdeers,
based on given batch size asenryi_aelr: r"o{{rse_rfaiatal
l|:d"e{{faaunlyt_(e'2r0r%or')s_}}f"atal | default(true) }}"

ro- l{erso:le: kubespray-defaults}
- { role: ukupbgreardne/tperse/n-uopdger,atadges, :tangosd:ep}re-upgrade }

- { role: kubernetes/kubeadm, tags: kubeadm, when: "kubeadm_

- { role: kubespray-defaults}

- ahnoys_tes:rrkourbs_e-fmatals:tetrru[0e]
ro- l{erso:le: kubespray-defaults}
"se-c{rreotl_ec:hkaunbgeerdn|deteefsa-ualpt(pfsa/lrsoet)a" t}e_tokens, tags:
rotate_tokens, when:

- hosts: kube-master

```
  aronlye_se:rrors_fatal: true
   - { role: kuberpnreateys-d-aepfapus/lntse}twork_plugin, tags: network }
   - { role: kubernetes/calpiepns/tp, otaligcsy:_ccloienntrto}ller, tags: policy-controller }

- ahnoys_tes:rrcoarlisc_ofa-rtral: "{{ any_errors_fatal | default(true) }}"


   - { role: network_plugin/calico/rr, tags: network }

- ahnoys_tes:rrko8rss-_cflautsatle:r"{{ any_errors_fatal |
  default(true) }}" ro- 1{erso:le: kubespray-defaults}
dn-s{mroalseq: }dnsmasq, when: "dns_mode ==
'dnsmasq_kubedns'", tags: re-so{ lrvocloen:
kf_umbeordnee=te=s'/hporseti_nrsetsaolll,vwcohnefn"':,
"tdangs:_rmeosodlevc!=on'nfo}ne' and

- ahnoys_tes:rrkourbs_e-fmatals:te"{r{[0a]ny_errors_fatal | default(true) }}"


   - { role: kubernetes-apps, tags: apps }
```

It will upgrade the Kubernetes Kubespray installation of the Kubernetes Cluster.