What's New in Kubernetes 1.11

CLOUD NATIVE
COMPUTING FOUNDATION

Presenters

Ihor Dvoretskyi

1.11 Features Lead

Jun Du

SIG-Networking

Chris O'Haver

SIG-Networking

Michael Taufen

SIG-Node

Hemant Kumar

SIG-Storage

Kaitlyn Barnard

1.11 Communications Coordinator

Agenda

IPVS-Based In-Cluster Service Load Balancing

CoreDNS

Dynamic Kubelet Configuration

Resizing Persistent Volumes

Q&A

IPVS-Based In-Cluster Service Load Balancing

Jun Du, Huawei

What's IPTables?

- What is IPTables?
  - iptables is a user-space application that allows configuring Linux kernel firewall (implemented on top of Netfilter) by configuring chains and rules.
  - What is Netfilter? A framework provided by the Linux kernel that allows customization of networking-related operations, such as packet filtering, **NAT**, port translation etc.
- Issues with IPTables as load balancer
  - Latency to access service (routing latency)
  - Latency to add/remove rule

What's IPVS

- What is IPVS?
  - Transport layer load balancer which directs requests for TCP, UDP and SCTP based services to real servers.
  - Same to IPTables, IPVS is built on top of Netfilter.
  - Support 3 load balancing mode: NAT, DR and IP Tunneling.
- Why using IPVS?
  - Better performance (Hashing vs. Chain)
  - More load balancing algorithm
    - Round robin, source/destination hashing.
    - Based on least load, least connection or locality, can assign weight to server.
  - Support server health check and connection retry
  - Support sticky session

Run Kube-proxy in IPVS mode

- Load required kernel modules
  - ip_vs, ip_vs_rr, ip_vs_wrr, ip_vs_sh, nf_conntrack_ipv4
- Switch proxy mode to IPVS
  - --proxy-mode=ipvs
- Enable feature gateway before v1.10
  - --feature-gates=SupportIPVSProxyMode=true

IPVS Service Network Topology

- When creating a ClusterIP type Service, IPVS proxier will do the following 3 things:
  - Make sure a dummy interface exists in the node, defaults to kube-ipvs0
  - Bind Service IP addresses to the dummy interface
  - Create IPVS virtual servers for each Service IP address respectively

Example

 # kubectl describe svc nginx-service

```
IP:                 10.102.128.4
Port:               http  3080/TCP
Endpoints:          10.244.0.235:8080,10.244.1.237:8080
```

 # ip addr

```
73: kube-ipvs0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN qlen 1000
link/ether 1a:ce:f5:5f:c1:4d brd ff:ff:ff:ff:ff:ff
inet 10.102.128.4/32 scope global kube-ipvs0
```

 # ipvsadm -ln

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.102.128.4:3080 rr
  -> 10.244.0.235:8080            Masq    1      0          0
  -> 10.244.1.237:8080            Masq    1      0
```

CoreDNS

Chris O'Haver (SIG-Networking/Infoblox)

What is CoreDNS?

CoreDNS is a DNS server.

It uses a plugin chain architecture. Each plugin can perform a DNS function.

- Flexible - combine plugins together for advanced functionality
- Extensible - write your own plugins

How does CoreDNS fit into Kubernetes?

The *kubernetes* plugin enables CoreDNS to provide DNS-based service discovery in a Kubernetes cluster.

- Follows Kubernetes DNS-Based Service Discovery Specification
- Can replace kube-dns - functionally equivalent
- CoreDNS is an approved alternative to kube-dns in Kubernetes 1.11
- Default in kubeadm 1.11
- Currently optional in kops, kubeup, minikube, kubespray, and others

CoreDNS fixes a few things...

Some open issues in kube-dns, resolved by CoreDNS.

dns#55 - Allow custom DNS entries for kube-dns

dns#116 - Missing ‘A’ records for headless service with pods sharing hostname

dns#131 - ExternalName not using stubDomains settings

dns#167 - Enable round robin A/AAAA records

dns#190 - kube-dns cannot run as non-root user

dns#232 - Use pod’s name instead of pod’s hostname in DNS SRV records

There are some outward changes with CoreDNS

Containers - Number of containers in the pod

Kube-dns has 3 (kube-dns, dnsmasq, sidecar)

CoreDNS has 1

Metrics - Both report metrics to Prometheus, but the set of metrics differ

Configuration - format of configuration entirely different (migration tools available)

CoreDNS fully configurable via configmap

Kube-dns *not* fully configurable via configmap (e.g. cache)

Here's an example CoreDNS configuration

Send stub domain example.com to 1.2.3.4
Cache locally up to 300 seconds

Serve liveness status on http 8080

Backend to k8s for cluster.local and reverse domains
- Resolve CNAME targets upstream
- Mimic kube-dns pod records behavior
- Continue searching reverse zones

Serve prometheus metrics

Forward other domains to /etc/resolv.conf
Cache for up to 30 seconds

```
example.com:53 {
    proxy . 1.2.3.4
    cache 300
}

.:53 {
    health
    kubernetes cluster.local in-addr.arpa ip6.arpa {
       upstream
       pods insecure
       fallthrough in-addr.arpa ip6.arpa
    }
    prometheus :9153
    proxy . /etc/resolv.conf
    cache 30
}
```

New Feature: Verified Pod Records

Example Pod records:

```
172-16-10-6.default.pod.cluster.local.   5   IN   A   172.16.10.6
```

In Kube-dns, all IP style names have a record, even if the pod doesn't exist.

In CoreDNS, you have the option to only create records for real pods.

```
kubernetes {
    pods verified
}
```

New: Server side domain search with Autopath

The Kubernetes ndots:5 problem...

Pod lookup for `infoblox.com`

```
infoblox.com.my-namespace.svc.cluster.local.  ->  NXDOMAIN
infoblox.com.svc.cluster.local.               ->  NXDOMAIN
infoblox.com.cluster.local.                   ->  NXDOMAIN
infoblox.com.my-internal-domain.io.           ->  NXDOMAIN
infoblox.com.my-other-internal-domain.io.     ->  NXDOMAIN
infoblox.com.                                 ->  A record returned
```

NXDOMAIN is not cached in kube-dns

= 6 trips between pod and DNS service.

Autopath does this on the server. Answer in 1 trip.

```
autopath {
     @kubernetes
}

kubernetes {
     pods verified
}
```

Other New Features

Zone transfers - list all records, or copy records to another server

Namespace and label filtering - expose a limited set of services

Adjustable TTL - adjust up/down default service record TTL

Negative Caching - By default caches negative responses (e.g. NXDOMAIN)

Extensibility with Plugins

There are many *built in* plugins (34). A few interesting ones...

- file - serve a zone from an RFC-1035 style zone file
- rewrite - rewrite incoming requests
- template - define responses using regex and go templates

… or build your own *external* plugins.

Some interesting *external* plugins...

- unbound - recursive dns
- pdsql - serve records from a backend database
- redisc - use redis as a shared cache across multiple CoreDNS instances
- kubernetai - connect to multiple kubernetes, for cross-cluster discovery

CoreDNS roadmap in Kubernetes

CoreDNS as an Alpha feature

CoreDNS was introduced as an Alpha feature and could be installed as an option instead of kube-dns in kubeadm, kube-up and minikube.

Kubernetes v1.9

Kubernetes v1.10

CoreDNS graduated to Beta

CoreDNS was graduated to Beta. In kubeadm, we could now automatically translate the kube-dns ConfigMap to corresponding CoreDNS ConfigMap.

CoreDNS graduated to GA

CoreDNS is currently GA and also available as the default DNS in kubeadm.

Kubernetes v1.11

Future releases

CoreDNS as the default DNS

Currently working towards making CoreDNS the default across all Kubernetes install tools (kube-up, minikube, kops etc.)

kube-dns deprecated

Support and Resources

Issues/Questions/Support

github: http://github.com/coredns/coredns (also kubernetes/dns)

slack: https://slack.cncf.io #coredns

security related: security@coredns.io

Documentation/Resources

http://coredns.io - plugin docs. blogs.

Also, The CoreDNS GA for Kubernetes Cluster DNS blog by John Belamaric, from which I borrowed very heavily to create this presentation.

Thanks!

We are conducting a survey of Kubernetes users who are using CoreDNS.

https://www.surveymonkey.com/r/SKZQSLK

Dynamic Kubelet Configuration

Mike Taufen (SIG-Node)

Why Dynamic Kubelet Configuration?

- Kubernetes offers **declarative APIs** hosted in a **central control plane**.
  - Unless you don't run in a Pod, like Kubelet in most deployments.
- Lifting Kubelet configuration into the control plane makes it **more visible** and **convenient to manipulate**.
- Some Alternatives:
  - SSH and edit a command-line in a systemd file, manually restart Kubelet
  - Use third-party configuration management automation tools
  - Create new VMs with desired configuration already installed, migrate work
  - **Infinity other ways that are not built for typical Kubernetes workflows.**

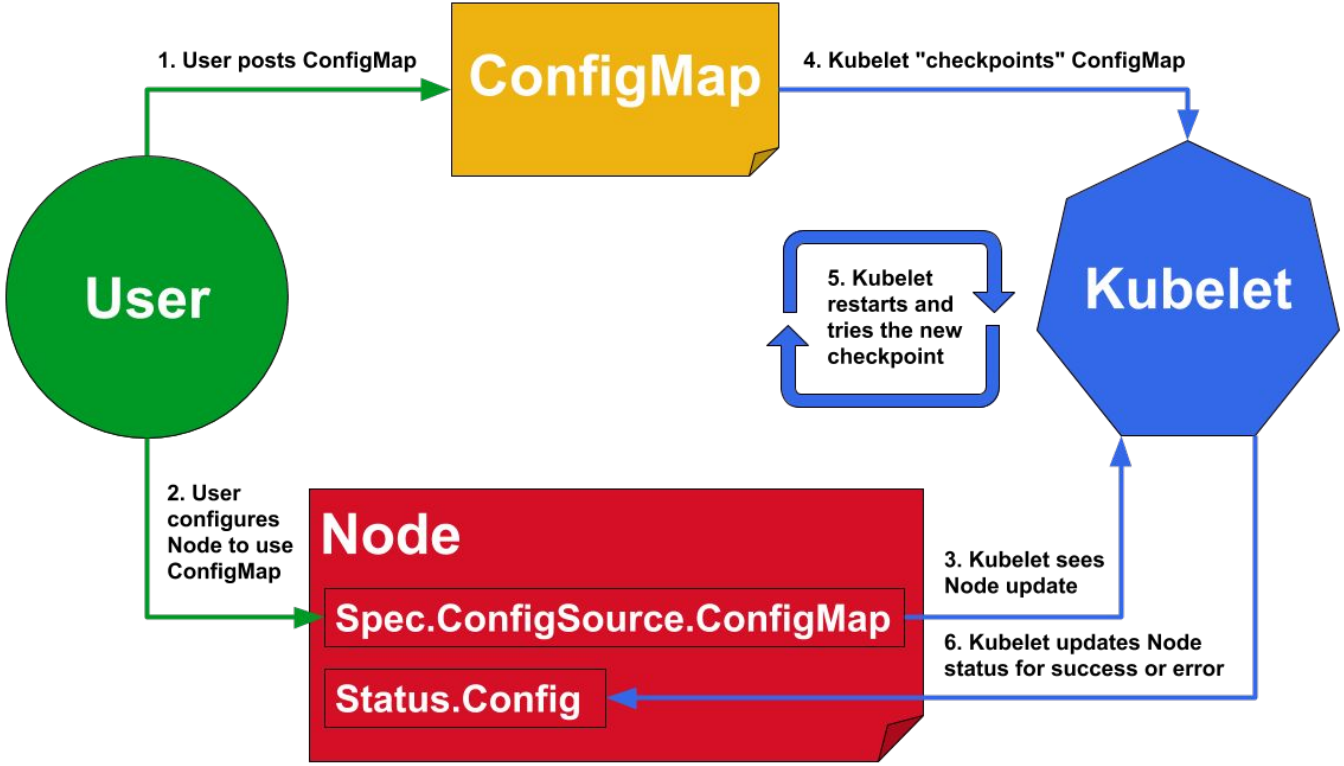
What is Dynamic Kubelet Configuration?

- **v1.10**: Configure Kubelet via **structured**, **versioned**, **K8s-style config file** (beta)
  - Write the file, then start Kubelet with --config=path/to/file.

- **v1.11**: *Dynamically* (for a live node) configure Kubelet (beta)
  - Exact same structured, versioned, K8s-style config file format.
  - **Config in K8s control plane via ConfigMap**
  - **Nodes reference a ConfigMap** that contains the config file, **Kubelet downloads, restarts, and uses**.

 |

Workflow (single-node config update)



For multi-node rollouts, simply update each Node's spec to reference the new ConfigMap.

Since each spec is updated independently, you control the rate/policy.

Example Control Plane

```
apiVersion: v1
kind: Node
metadata:
  name: pool1-node-foo
spec:
  configSource:
    configMap:
      name: pool1-config-abcde
      namespace: node-pool-configs
      kubeletConfigKey: kubelet-config
status:
  config:
    assigned:
      configMap:
        name: pool1-config-abcde
        namespace: node-pool-configs
        kubeletConfigKey: kubelet-config
        uid: 18c93a87-84b6-11e8-94d0-42010a80008c
        resourceVersion: "57"
    active:
      configMap:
        name: pool1-config-edcba
        namespace: node-pool-configs
        kubeletConfigKey: kubelet-config
        uid: 101e77e8-84b6-11e8-94d0-42010a80008c
        resourceVersion: "23"
    lastKnownGood:
      configMap:
        name: pool1-config-edcba
        namespace: node-pool-configs
        kubeletConfigKey: kubelet-config
        uid: 101e77e8-84b6-11e8-94d0-42010a80008c
        resourceVersion: "23"
    error: "failed to validate config, see Kubelet
```

Reference ConfigMap from Node spec

assigned: The config the Kubelet believes it has been told to use.
active: The config the Kubelet is using right now.
lastKnownGood: The last config the Kubelet qualified as stable.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: pool1-config-abcde
  namespace: node-pool-configs
  uid: 18c93a87-84b6-11e8-94d0-42010a80008c
  resourceVersion: "57"
data:
  kubelet-config: |-
    apiVersion: kubelet.config.k8s.io/v1beta1
    kind: KubeletConfiguration
    evictionHard:
      memory.available: "100Mi"
      nodefs.available: "10%"
      nodefs.inodesFree: "5%"
    …
    kubeAPIQPS: "invalid junk to make example error"
```

Node status explicit on uid/resourceVersion Kubelet using to help debugging.*

Any errors also reported in Node status.

*No uid/rv pinning in spec is intentional:
- Can't statically specify these versions.
- No history → hard to debug intent in live environment.

You should use the name as a version and treat the ConfigMap as immutable (kubectl --append-hash). A new config rollout implies a new ConfigMap.

Gotchas

- **Legacy flags specified on the command line take precedence.**
  - Config fields may be settable by deprecated, but not-yet-removed, flags.
  - Flags are invisible to K8s control plane, so you still need to know how the node was bootstrapped.

- **With great power comes great responsibility.**
  - Using the same file format **lets you set the whole config** dynamically. This is powerful, but you must be careful:
    - **Some things are very safe**: QPS
    - **Some you probably shouldn't touch**: names of cgroups
  - See the inline docs for per-field advice on using dynamic Kubelet config:
    - https://godoc.org/k8s.io/kubernetes/pkg/kubelet/apis/kubeletconfig/v1beta1#KubeletConfiguration
  - Remember: **Feature targets system experts/service providers**.

Future Work

- Many **low level knobs** → users actually want **high-level policy**
  - Users don't just change knobs for the hell of it, **they have an end goal**.
  - Can we have/offer **high-level "opinions"** that satisfy common use/tuning cases?
  - Are there **principles that work for everyone** we can bake into our components?
  - **Higher-level** → **friendlier** to non-experts
- No "node pool" orchestration built-in, but Cluster API is focused on solving this, among other things.
  - https://github.com/kubernetes-sigs/cluster-api
- Continue migrating Kubelet flags to the config file API.

See Also

- Blog post
  - https://kubernetes.io/blog/2018/07/11/dynamic-kubelet-configuration/
- Official docs
  - https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/
  - https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
- Versioned Component Configuration Files (use versioned files, not flags)
  - https://goo.gl/GM8KyH
- Declarative application management in Kubernetes
  - https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/declarative-application-management.md

Resizing Persistent Volumes

Hemant Kumar (sig-storage/Red Hat)

What is Persistent Volume?

- Persistent Volumes provide **storage** layer in Kubernetes which is useful for running applications that need to persist their data.

- Building blocks of Persistent Volumes:
  - StorageClass
  - PersistentVolume (**PV**)
  - PersistentVolumeClaim (**PVC**)

- StorageClass and PersistentVolume are cluster scoped resources and **PersistentVolumeClaim** is namespaced resource.

 |

What problem resizing solves?

- Once provisioned - **PV** and **PVCs** remain of fixed size and as applications that use them need more storage it becomes essential to move them to larger **PVs** which can be tricky.

- Allows in-place expansion of dynamically provisioned **PVCs** in stateful sets.

What does resizing Persistent Volume mean?

- In-place expansion of Volume by editing the **PersistentVolumeClaim(PVC)** object.
- **Shrinking** of persistent volumes is not supported.
- Supports expansion of remote volume object (Such as **EBS**, **GCE-PD**) and expansion of **file system** on node.
- Supported volume types - **AWS-EBS, GCE-PD, Azure Disk, Azure File, GlusterFS, Cinder, Portworx and Ceph RBD.**
- **File system** expansion requires pod restart.

 |

How to expand persistent volumes?

- In Kubernetes **v1.11** the persistent volume expansion feature is being promoted to **beta**.
- A cluster admin can enable expansion of **PVCs** created from certain **StorageClass** by setting **`allowVolumeExpansion`** property of StorageClass **`true`**.

```
~> kubectl edit sc standard
```

```
kind: StorageClass
metadata:
  name: standard
allowVolumeExpansion: true
```

How to expand persistent volume?

```
~> kubectl edit pvc www-web-0
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: www-web-0
spec:
  resources:
    requests:
      storage: 10Gi
status:
  capacity:
    storage: 5Gi
```

How to expand persistent volumes?

- After editing **PVC -** for shared file systems like **Glusterfs**, **Azure File** expanded storage is immediately available to the pod.

- For block storage volume types (**EBC**, **GCE-PD**, **Azure Disk**, **Ceph RBD etc**) requires file system expansion before additional space could become available to the pod.

 |

Expanding File system

- Expanding file system requires restarting the pod once underlying volume has been resized.
  1. Edit the **PVC** to request more space.
  2. Once underlying volume has been expanded by storage provider(**GCE, AWS etc**) - **PVC** will have `FileSystemResizePending` condition.
  3. Wait for PVC to have `FileSystemResizePending` and restart the pod.

```
~> kubect describe pvc www-web-0
Name:          www-web-0
Namespace:     default
StorageClass:  standard
Status:        Bound
Volume:        pvc-8b3e65bb-9406-11e8-bde0-42010af00065
Capacity:      5Gi
Access Modes:  RWO
Conditions:
  Type                      Status  LastProbeTime                     LastTransitionTime
Reason  Message
  FileSystemResizePending   True    Mon, 01 Jan   Mon, 30 Jul 2018 Waiting for user to (re-)start a pod to
finish file system resize of volume on node.
```

Online File system expansion

- Kubernetes-**1.11** introduces **alpha** feature `ExpandInUsePersistentVolumes` which will allow expansion of in-use volumes without need of restarting the pod.

- Automatic (online) file system expansion only supported for volumes that are in-use by a running pod.

Future work

- Support volume expansion for CSI volumes (https://github.com/kubernetes/features/issues/556).
- Support volume expansion for Flex volumes (https://github.com/kubernetes/features/issues/304)
- General stability and expanding file system type support.

Suggested Resources

- Persistent Volumes in Kubernetes
- https://kubernetes.io/blog/2018/07/12/resizing-persistent-volumes-using-kubernetes/
- https://github.com/container-storage-interface/spec/pull/222

Questions?

CLOUD NATIVE
COMPUTING FOUNDATION

Thank You

CLOUD NATIVE
COMPUTING FOUNDATION