WIN A $100 AMAZON GIFT CERTIFICATE

Try Zenko Orbit and fill out a short survey to enter the drawing!

Visit https://zenko.io

SCALITY

SCALITY METALK8S

AN OPINIONATED KUBERNETES DISTRIBUTION WITH A FOCUS ON LONG-TERM ON-PREM DEPLOYMENTS

Nicolas Trangez - Technical Architect
nicolas.trangez@scality.com
@eikke | @Scality | @Zenko

SCALITY

ABOUT SCALITY

ONE PURPOSE

GIVING FREEDOM & CONTROL TO PEOPLE WHO CREATE VALUE WITH DATA

SCALITY

GLOBAL PRESENCE

8 OFFICES

200+ PEOPLE

20+ NATIONALITIES

GLOBAL CLIENT BASE

EUROPE

120+

Liberté • Égalité • Fraternité
République Française
Ministère de la Défense

orange™

Metropolitan Police

Mairie de Paris

RTL

SFR

Eurosport

Com Hem

dailymotion

Natixis

molotov.tv

TF1

Hospices Civils de Lyon

Renault

SNCF

AB

Societe Generale

deluxe

eDF

Airbus

AMERICAS

60+

Penn Medicine

Bank of America

Bloomberg Media

Comcast

Staples

logitech

HBO

rackspace

AUSTRALIA

~10

EndemolShine Group

MediaHub

foxtel

JAPAN

20+

KDDI

BIGLOBE

DMM.com

@nifty

SCALITY

OUR JOURNEY TO KUBERNETES

Scality RING, S3 Connector & Zenko

Scality RING

On-premise
Distributed Object & File Storage

- Physical servers, some VMs
- Only the OS available (incl. ‘Legacy’ like CentOS 6)
- Static resource pools
- Static server roles / configurations
- Solution distributed as RPM packages, deployed using SaltStack
- De-facto taking ownership of host, difficult to run multiple instances
- Fairly static post-install

Scality S3 Connector

On-premise S3-compatible Object Storage

- Physical servers, sometimes VMs
- Static resource pools
- “Microservices” architecture
- Solution distributed as Docker container images, deployed using Ansible playbooks
- No runtime orchestration
- Log management, monitoring,... comes with solution

Scality Zenko

Multi-Cloud Data Controller

- Deployed on-prem or ‘in the Cloud’: major paradigm shift
- New challenges, new opportunities
- Multi-Cloud Data Controller, must run on multiple Cloud platforms

Scality Zenko

Deployment Model

- Embraced Docker as distribution mechanism
  - Some shared with Scality S3 Connector
- For Cloud deployments, started with Docker Swarm
  - Ran into scaling, reliability and other technical issues
- Decided to move to Kubernetes
  - Managed platforms for Cloud deployments, where available (GKE, AKS, EKS,...)
  - On-prem clusters

Scality Zenko

Kubernetes Benefits

- Homogenous deployment between in-cloud and on-prem
- Various services provided by cluster:
  - Networking & policies
  - Service restart, rolling upgrades
  - Service log capturing & storage
  - Service monitoring & metering
  - Load-balancing
  - TLS termination
- Flexible resource management
  - If needed, easily add resources to cluster by adding some (VM) nodes
  - HorizontalPodAutoscaler

Scality Zenko

Kubernetes Deployment

- Currently using Helm chart
- Contributed many fixes and features to upstream charts repository
- Contributed new charts and became maintainer of some others
- Looking into Zenko ‘operator’
- Can run in your cluster (https://github.com/Scality/Zenko) or test-drive a hosted instance for free using Zenko Orbit at https://zenko.io/admin

OUR JOURNEY TO KUBERNETES

MetalK8s

SCALITY

On-prem Kubernetes

- Can’t expect a Kubernetes cluster to be available, provided by Scality customer
- Looked into various existing offerings, but in the ends needs to be supported by/through Scality (single offering)
    - Also, many existing solutions don’t cover enterprise datacenter requirements
- Decided to roll our own

SCALITY METALK8S

AN OPINIONATED KUBERNETES DISTRIBUTION WITH A FOCUS ON LONG-TERM ON-PREM DEPLOYMENTS

OPINIONATED

We offer an out-of-the-box experience, no non-trivial choices to be made by users

LONG-TERM

Zenko solution is mission-critical, can’t spawn a new cluster to upgrade and use ELB (or similar) in front

ON-PREM

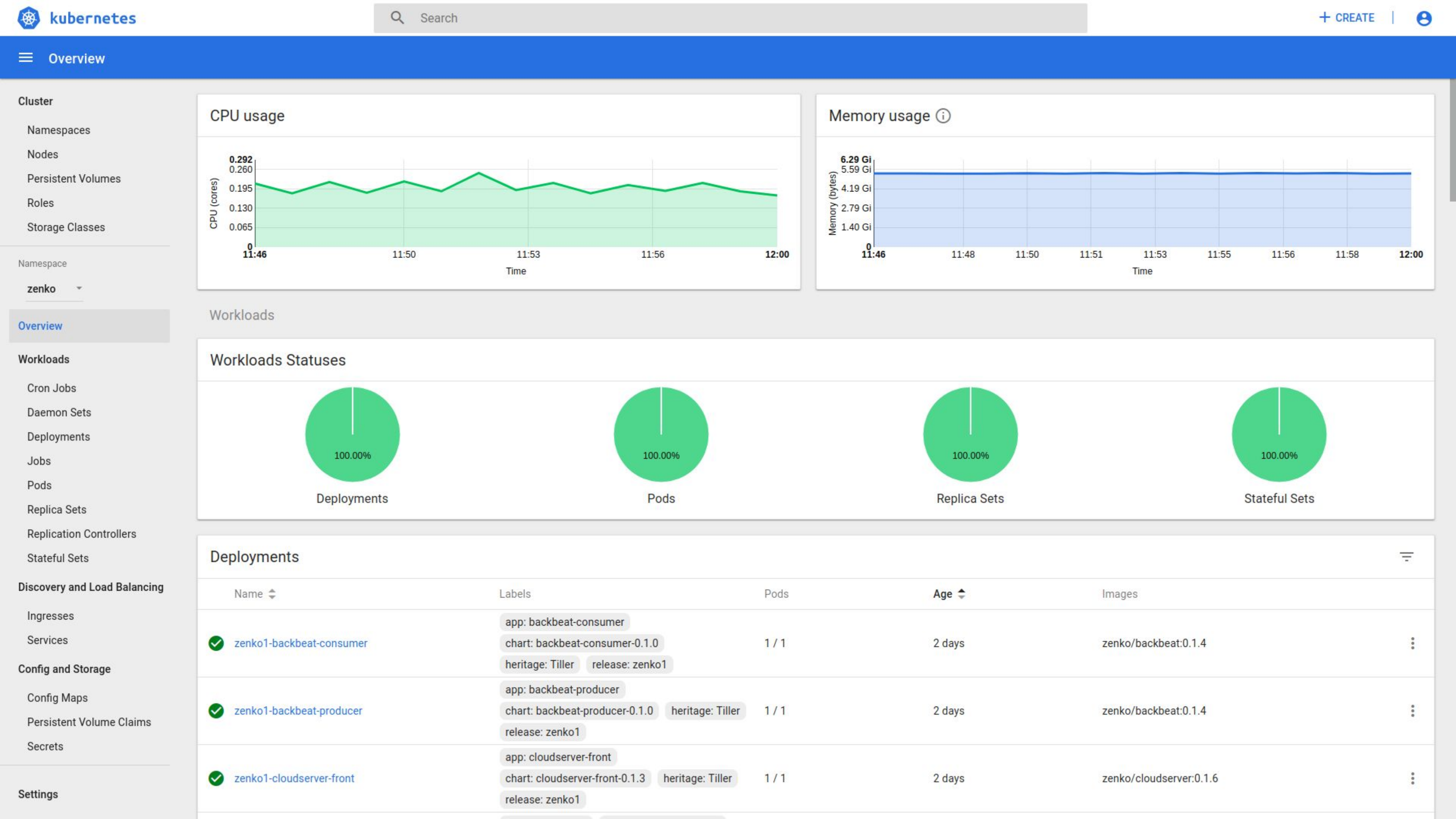
Can’t expect anything to be available but (physical) servers with a base OS

Scality MetalK8s

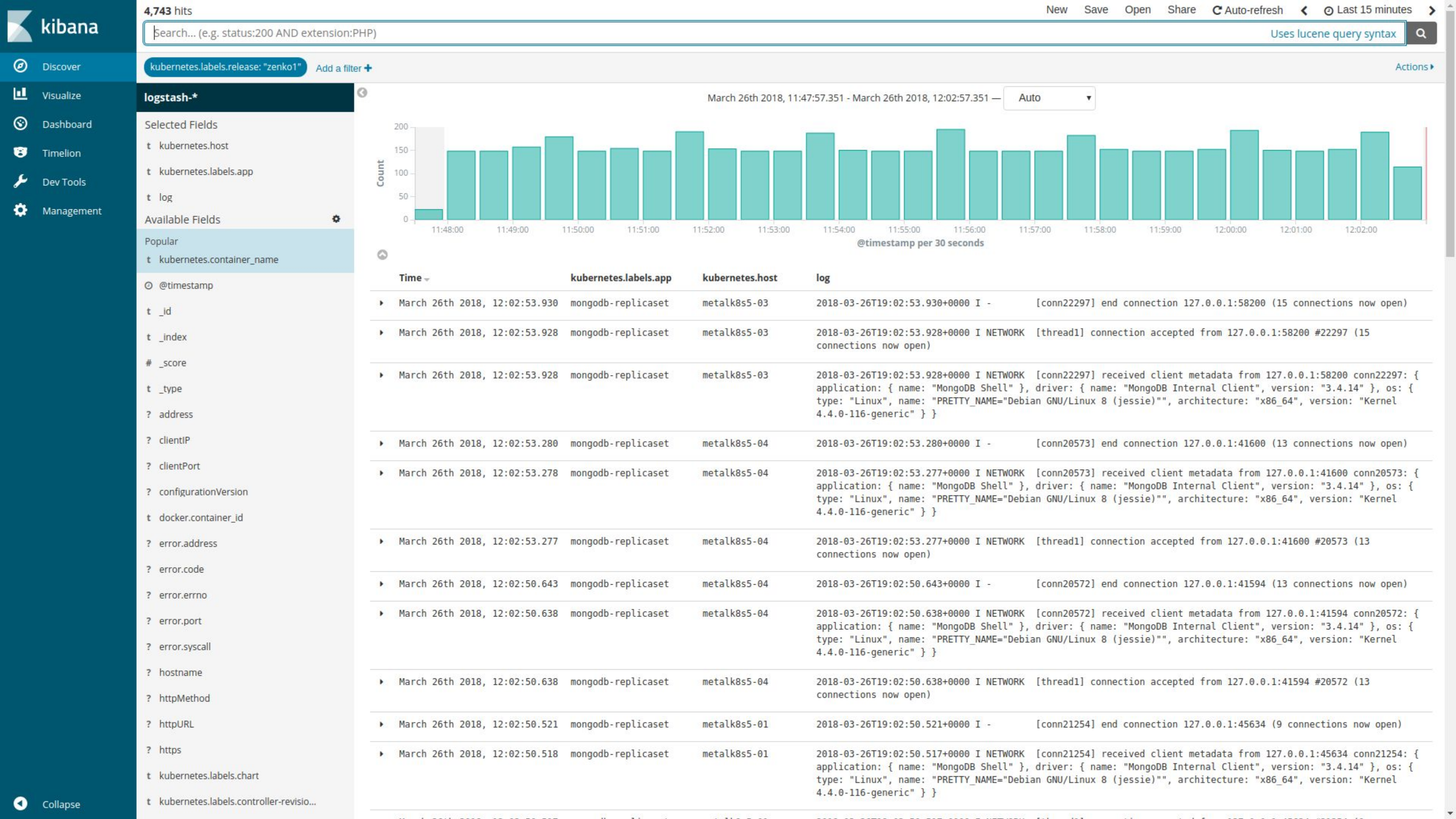
- Scope: 3-20 physical machine, pre-provisioned by customer or partner
- Built on top of the Kubespray Ansible playbook
- Use Kubespray to lay out a base Kubernetes cluster
  - Also: etcd, CNI
- Add static & dynamic inventory validation pre-checks, OS tuning, OS security
  - Based on experience from large-scale Scality RING deployments
- Augment with various services, deployed using Helm
  - Operations
  - Ingress
  - Cluster services
- Take care of on-prem specific storage architecture

Scality MetalK8s: Cluster Services

- “Stand on the shoulders of giants”
- Heapster for dashboard graphs, `kubectl top`,...
- metrics-server for HorizontalPodAutoscaler
    - Looking into k8s-prometheus-adapter
- Ingress & TLS termination: nginx-ingress-controller
- Cluster monitoring & alerting: Prometheus, prometheus-operator, Alertmanager, kube-prometheus, Grafana
    - Host-based node_exporter on all servers comprising the cluster, including etcd
- Host & container logs: ElasticSearch, Curator, fluentd, fluent-bit, Kibana
- All of the above gives a great out-of-the-box experience for operators



	Name	Labels	Pods	Age	Images
✓	zenko1-backbeat-consumer	app: backbeat-consumer chart: backbeat-consumer-0.1.0 heritage: Tiller release: zenko1	1 / 1	2 days	zenko/backbeat:0.1.4
✓	zenko1-backbeat-producer	app: backbeat-producer chart: backbeat-producer-0.1.0 heritage: Tiller release: zenko1	1 / 1	2 days	zenko/backbeat:0.1.4
✓	zenko1-cloudserver-front	app: cloudserver-front chart: cloudserver-front-0.1.3 heritage: Tiller release: zenko1	1 / 1	2 days	zenko/cloudserver:0.1.6



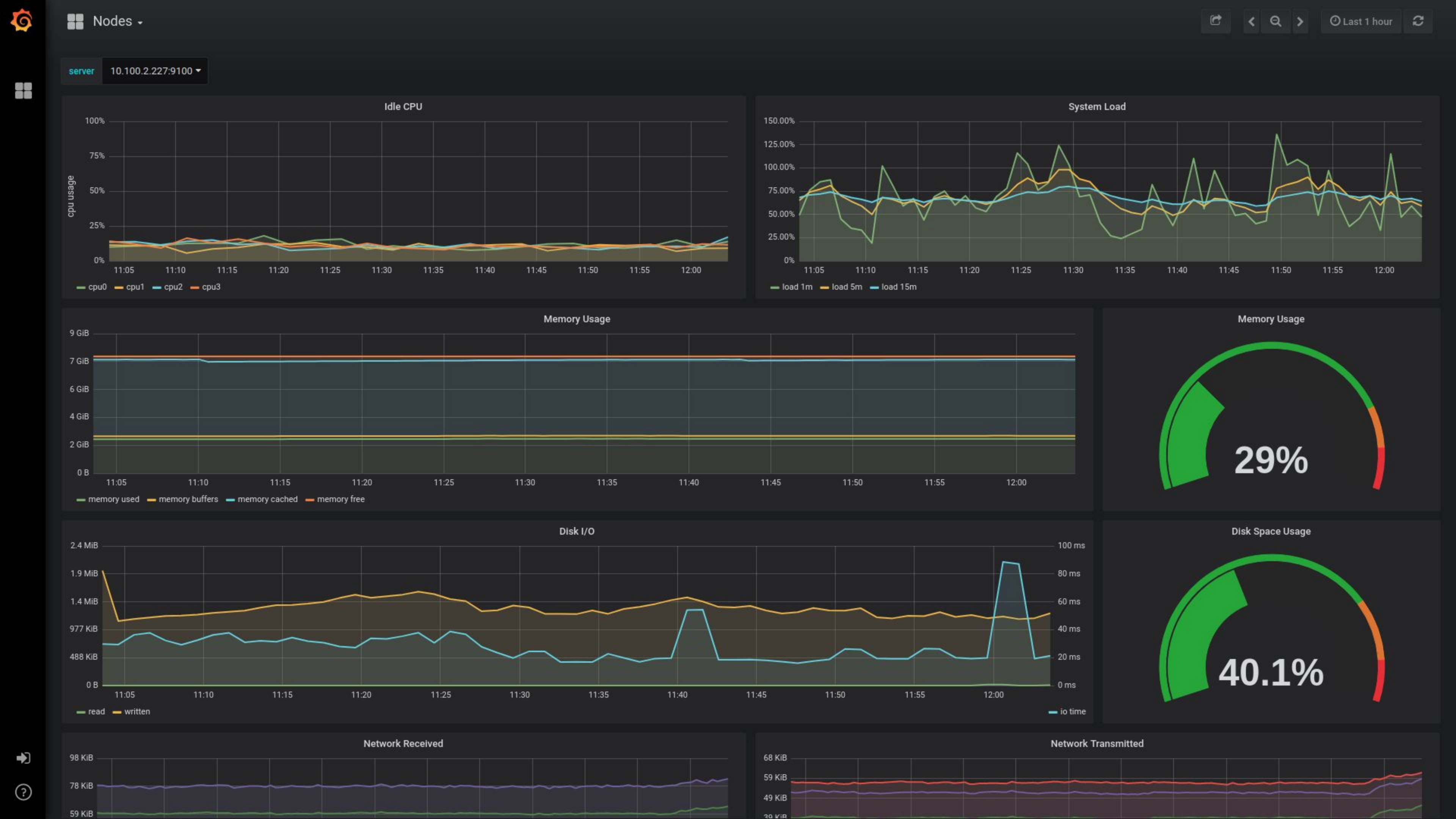
4,743 hits

Search... (e.g. status:200 AND extension:PHP)

kubernetes.labels.release: "zenko1"

March 26th 2018, 11:47:57.351 - March 26th 2018, 12:02:57.351 —

Time	kubernetes.labels.app	kubernetes.host	log
March 26th 2018, 12:02:53.930	mongodb-replicaset	metalk8s5-03	2018-03-26T19:02:53.930+0000 I - [conn22297] end connection 127.0.0.1:58200 (15 connections now open)
March 26th 2018, 12:02:53.928	mongodb-replicaset	metalk8s5-03	2018-03-26T19:02:53.928+0000 I NETWORK [thread1] connection accepted from 127.0.0.1:58200 #22297 (15 connections now open)
March 26th 2018, 12:02:53.928	mongodb-replicaset	metalk8s5-03	2018-03-26T19:02:53.928+0000 I NETWORK [conn22297] received client metadata from 127.0.0.1:58200 conn22297: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.14" }, os: { type: "Linux", name: "PRETTY_NAME="Debian GNU/Linux 8 (jessie)"", architecture: "x86_64", version: "Kernel 4.4.0-116-generic" } }
March 26th 2018, 12:02:53.280	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:53.280+0000 I - [conn20573] end connection 127.0.0.1:41600 (13 connections now open)
March 26th 2018, 12:02:53.278	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:53.277+0000 I NETWORK [conn20573] received client metadata from 127.0.0.1:41600 conn20573: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.14" }, os: { type: "Linux", name: "PRETTY_NAME="Debian GNU/Linux 8 (jessie)"", architecture: "x86_64", version: "Kernel 4.4.0-116-generic" } }
March 26th 2018, 12:02:53.277	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:53.277+0000 I NETWORK [thread1] connection accepted from 127.0.0.1:41600 #20573 (13 connections now open)
March 26th 2018, 12:02:50.643	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:50.643+0000 I - [conn20572] end connection 127.0.0.1:41594 (13 connections now open)
March 26th 2018, 12:02:50.638	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:50.638+0000 I NETWORK [conn20572] received client metadata from 127.0.0.1:41594 conn20572: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.14" }, os: { type: "Linux", name: "PRETTY_NAME="Debian GNU/Linux 8 (jessie)"", architecture: "x86_64", version: "Kernel 4.4.0-116-generic" } }
March 26th 2018, 12:02:50.638	mongodb-replicaset	metalk8s5-04	2018-03-26T19:02:50.638+0000 I NETWORK [thread1] connection accepted from 127.0.0.1:41594 #20572 (13 connections now open)
March 26th 2018, 12:02:50.521	mongodb-replicaset	metalk8s5-01	2018-03-26T19:02:50.521+0000 I - [conn21254] end connection 127.0.0.1:45634 (9 connections now open)
March 26th 2018, 12:02:50.518	mongodb-replicaset	metalk8s5-01	2018-03-26T19:02:50.517+0000 I NETWORK [conn21254] received client metadata from 127.0.0.1:45634 conn21254: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.14" }, os: { type: "Linux", name: "PRETTY_NAME="Debian GNU/Linux 8 (jessie)"", architecture: "x86_64", version: "Kernel 4.4.0-116-generic" } }


Nodes
Last 1 hour
server
10.100.2.227:9100
Idle CPU
cpu usage
100%
75%
50%
25%
0%
11:05
11:10
11:15
11:20
11:25
11:30
11:35
11:40
11:45
11:50
11:55
12:00
cpu0
cpu1
cpu2
cpu3
System Load
150.00%
125.00%
100.00%
75.00%
50.00%
25.00%
0%
load 1m
load 5m
load 15m
Memory Usage
9 GiB
7 GiB
6 GiB
4 GiB
2 GiB
0 B
memory used
memory buffers
memory cached
memory free
Memory Usage
29%
Disk I/O
2.4 MiB
1.9 MiB
1.4 MiB
977 KiB
488 KiB
0 B
100 ms
80 ms
60 ms
40 ms
20 ms
0 ms
read
written
io time
Disk Space Usage
40.1%
Network Received
98 KiB
78 KiB
59 KiB
Network Transmitted
68 KiB
59 KiB
49 KiB
39 KiB

Scality MetalK8s: Storage

- On-prem: no EBS, no GCP Persistent Disks, no Azure Storage Disk,...
- Also: can’t rely on NAS (e.g. through OpenStack Cinder) to be available
- Lowest common denominator: local disks in a node
- PVs bound to a node, hence PVCs bound, hence Pods bound
    - Thanks PersistentLocalVolumes & VolumeScheduling!
- Decided not to use LocalVolumeProvisioner, but static approach (for now)
    - Based on LVM2 Logical Volumes for flexibility
    - PV, VG, LVs defined in inventory, created/formatted/mounted by playbook
    - K8s PV objects created by playbook
    - May support whole partitions/drives depending on application need
- Dynamic local volume provisioning through CSI (using LVM) is getting there…
    - Future: volume encryption?

Scality MetalK8s: Deployment

- Based on years of years of experience deploying Scality RING at enterprise customers, service providers,...
- Constraints in datacenters often very different from ‘VMs on EC2’
  - No direct internet access: everything through HTTP(S) proxy, no non-HTTP traffic
  - Dynamic server IP assignment
  - Security rules requiring services to bind to specific IPs only, different subnets for control & workload,...
  - Fully air gapped systems: requires 100% offline installation
  - Non-standard OS/kernel
  - Integration with corporate authn/authz systems
- Not all of the above supported yet, tackling one by one
  - Relevant patches to be upstreamed to Kubespray
- Only support RHEL/CentOS family of Linux distributions
  - Support for Ubuntu and others can be community-driven, Kubespray supports them
  - RHEL/CentOS sometimes difficult targets for containers/Docker/Kubernetes

Scality MetalK8s: Ease of Deployment

```
$ # Requirements: a Linux or OSX machine with Python and coreutils-like

$ # Create inventory

$ vim inventory/...

$ make shell # Launches a ‘virtualenv’ with Ansible & deps, ‘kubectl’, ‘helm’

$ # Demo @ https://asciinema.org/a/9kNIpBWg4KiwjT5mNSrH0tmj9

$ ansible-playbook -i inventory -b playbooks/deploy.yml

$ # Grab a coffee, and done
```

Scality MetalK8s: Non-technical goodies

- Documentation
  - Various guides: Installation, Operations, Reference
  - https://metal-k8s.readthedocs.io
- Extensive testing
  - Installation
  - Upgrade
  - Services
  - Failure testing

Future Directions

Scality MetalK8s: Shifting focus

- Today: general-purpose deployment tool, fulfil K8s cluster pre-req of $product

- Future: use-case specific component a vendor (you!) can embed in on-prem solution/product running on K8s without being a K8s product
  - More configurable to match exact solution requirements and deployment environment
  - Tighten out-of-the-box security depending on application ‘insecurity’ needs

Scality MetalK8s: The road forward

- Increase documentation coverage
- Considering removing Kubespray
  - Too ‘big’ for our purposes
  - kubeadm brought kubelet TLS bootstrapping and many other goodies
  - Non-trivial to implement certain requirements/features
- Migrate Docker to containerd or cri-o
- Work with Cluster API, implement bare-metal provider?
- Looking into cluster federation (multi-site solutions), built-in over-the-wire encryption (Wireguard?), ‘active’ cluster controller (refresh short-TTL TLS certs, provision new nodes,...), netboot (like CoreOS/Matchbox/Tectonic, but plain CentOS/RHEL), other CNIs, integration of failover (VIP) and load-balancing service, optional deployment of more cluster-provided services (Istio, Jaeger,...), security (TUF/Notary, OPA,...), KubeVirt etc.

SCALITY METALK8S

AN OPINIONATED KUBERNETES DISTRIBUTION WITH A FOCUS ON LONG-TERM ON-PREM DEPLOYMENTS

https://zenko.io

https://github.com/scality/metalk8s

@Scality | @Zenko

Visit https://zenko.io

WIN A $100 AMAZON GIFT CERTIFICATE

Try Zenko Orbit and fill out a short survey to enter the drawing!

SCALITY