Day7 容器进阶之Kubernetes 应用生命周期原理分析

大 纲

- 监控集群组件
- 监控应用
- 管理组件日志
- 管理应用日志
- Deployment升级和回滚
- 配置应用的不同方法
- 应用弹性伸缩
- 应用自恢复

监控集群组件

集群整体状态：

```
$ kubectl cluster-info
```

```
Kubernetes master is running at https://10.142.0.2:6443
KubeDNS is running at https://10.142.0.2:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

更多集群信息：

```
$ kubectl cluster-info dump
```

通过插件部署：

```
$ kubectl get pod etcd -n kube-system
$ kubectl describe pod kube-apiserver -n kube-system
```

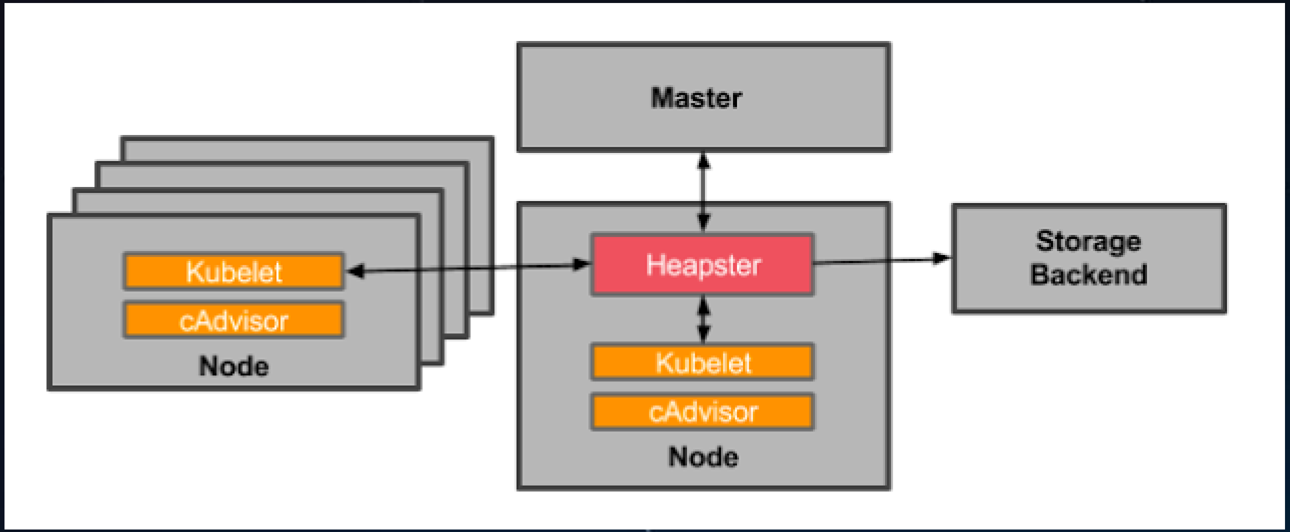
组件metrics：

```
$ curl localhost:10250/stats/summary
```

组件健康状况：

```
$ curl localhost:10250/healthz
```

Heapster + cAdvisor监控集群组件

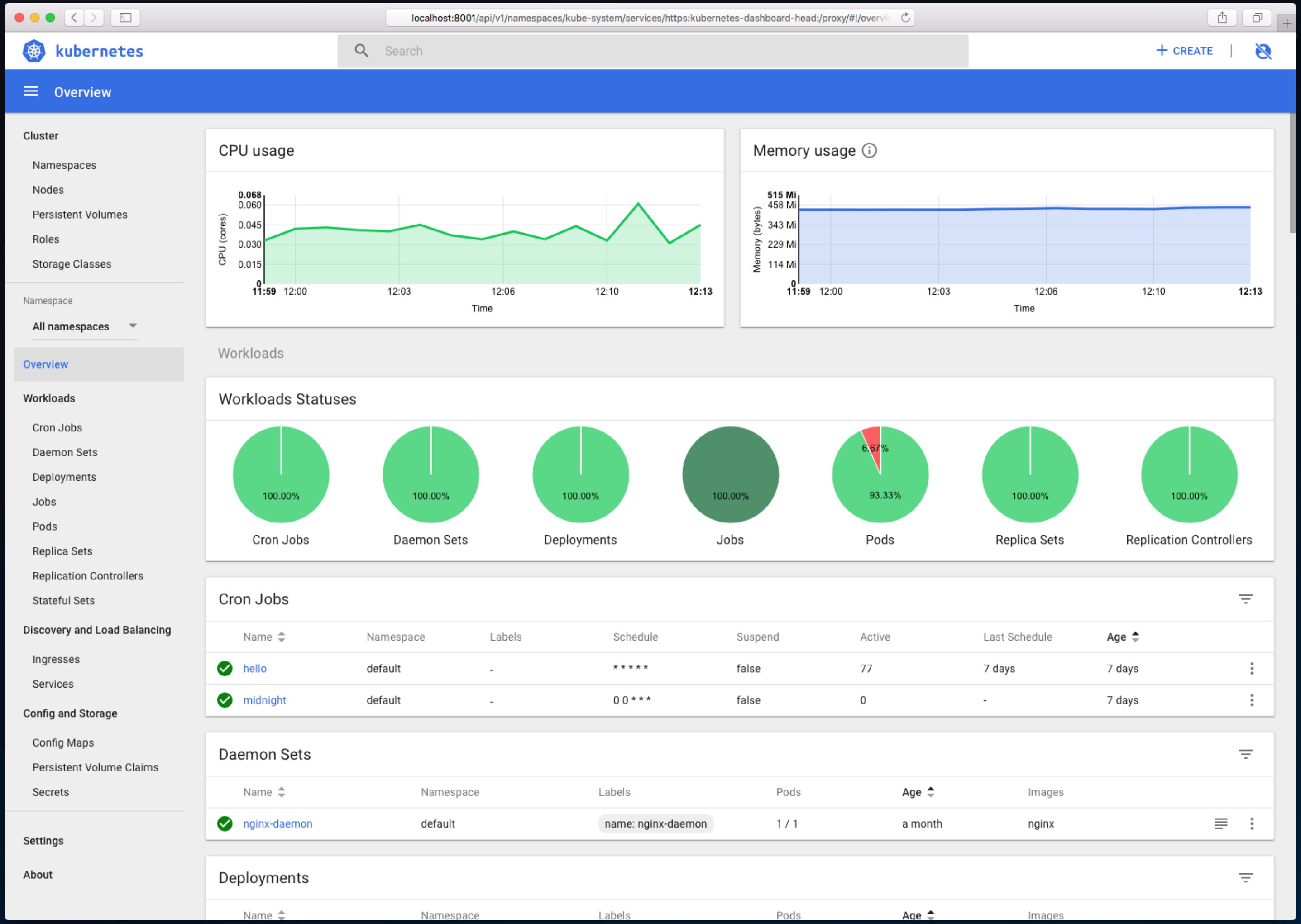


对接了heapster或metrics-server后展示Node CPU/内存/存储资源消耗：

```
$ kubectl top node {node name}
```

cAdvisor既能收集容器CPU、内存、文件系统和网络使用统计信息，还能采集节点资源使用情况；
cAdvisor和Heapster都不能进行数据存储、趋势分析和报警。因此，还需要将数据推送到InfluxDB，Grafana等后端进行存储和图形化展示。
Heapster即将被metrics-server替代

Kuberneetes Dashboard UI



Cron Jobs

	Name	Namespace	Labels	Schedule	Suspend	Active	Last Schedule	Age
✓	hello	default	-	* * * * *	false	77	7 days	7 days
✓	midnight	default	-	0 0 * * *	false	0	-	7 days

Daemon Sets

	Name	Namespace	Labels	Pods	Age	Images
✓	nginx-daemon	default	name: nginx-daemon	1 / 1	a month	nginx

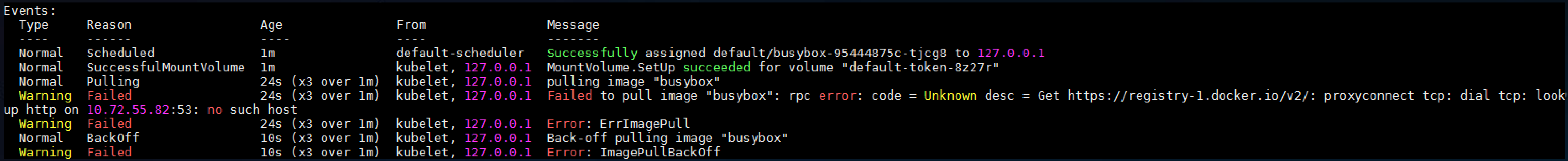
Deployments

Name	Namespace	Labels	Pods	Age	Images

Kubernetes Dashboard用于监控/展示Kubernetes所有的资源对象：
Cluster（Node，PV等）
Workload（Pod，Deployment等）
Config（Configmap，Secrets等）
…

监控应用

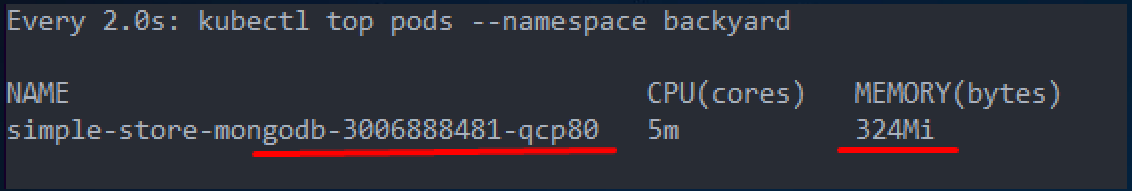
$ kubectl describe pod



```
Events:
  Type     Reason                 Age                From               Message
  ----     ------                 ----               ----               -------
  Normal   Scheduled              1m                 default-scheduler  Successfully assigned default/busybox-95444875c-tjcg8 to 127.0.0.1
  Normal   SuccessfulMountVolume  1m                 kubelet, 127.0.0.1  MountVolume.SetUp succeeded for volume "default-token-8z27r"
  Normal   Pulling                24s (x3 over 1m)   kubelet, 127.0.0.1  pulling image "busybox"
  Warning  Failed                 24s (x3 over 1m)   kubelet, 127.0.0.1  Failed to pull image "busybox": rpc error: code = Unknown desc = Get https://registry-1.docker.io/v2/: proxyconnect tcp: dial tcp: look
up http on 10.72.55.82:53: no such host
  Warning  Failed                 24s (x3 over 1m)   kubelet, 127.0.0.1  Error: ErrImagePull
  Normal   BackOff                10s (x3 over 1m)   kubelet, 127.0.0.1  Back-off pulling image "busybox"
  Warning  Failed                 10s (x3 over 1m)   kubelet, 127.0.0.1  Error: ImagePullBackOff
```

对接了heapster或metrics-server后，展示Pod CPU/内存/存储资源消耗：

$ kubectl top pod {pod name}



```
Every 2.0s: kubectl top pods --namespace backyard

NAME                                 CPU(cores)   MEMORY(bytes)
simple-store-mongodb-3006888481-qcp80   5m           324Mi
```

$ kubectl get pod {pod name} --watch

管理K8S组件日志

组件日志：

```
/var/log/kube-apiserver.log
/var/log/kube-proxy.log
/var/log/kube-controller-manager.log
/var/log/kubelet.log
```

使用systemd管理：

```
$ journalctl –u kubelet
```

使用K8S插件部署：

```
$ kubectl logs -f kube-proxy
```

管理K8S应用日志

# 从容器标准输出截获：

```
$ kubectl logs -f {pod name} –c {container name}
$ docker logs -f {docker name}
```

# 日志文件挂载到主机目录：

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: gcr.io/google_containers/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /log
      name: log-volume
  volumes:
  - name: log-volume
    hostPath:
      # directory location on host
      path: /var/k8s/log
```

直接进入容器内查看日志：

```
$ kubectl exec -it {pod} -c {container} /bin/sh
$ docker exec -it {container} /bin/sh
```

Deployment升级与回滚 - 1

```
# 创建Deployment：
$ kubectl run {deployment} –image={image} –replicas={rep.}
# 或使用yaml文件形式，重点配置replicas和image字段。

# 升级Deployment：
$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
$ kubectl set resources deployment/nginx-deployment -c=nginx --limits=cpu=200m,memory=512Mi

# 升级策略：
minReadySeconds: 5
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 1 #默认25%
    maxUnavailable: 1 #默认25%
```

Deployment升级与回滚 - 2

```
# 暂停Deployment：
$ kubectl rollout pause deployment/nginx-deployment

# 恢复Deployment：
$ kubectl rollout resume deployment/nginx-deployment

# 查询升级状态：
$ kubectl rollout status deployment/nginx-deployment

# 查询升级历史：
$ kubectl rollout history deploy/nginx-deployment
$ kubectl rollout history deploy/nginx-deployment --revision=2

# 回滚：
$ kubectl rollout undo deployment/nginx-deployment --to-revision=2
```

应用弹性伸缩

```
$ kubectl scale deployment nginx-deployment --replicas=10

# 对接了heapster，和HPA联动后：
$ kubectl autoscale deployment nginx-deployment --min=10 --max=15 --cpu-percent=80
```

应用自恢复：restartPolicy + livenessProbe

Pod Restart Policy：Always, OnFailure, Never
livenessProbe：http/https Get, shell exec, tcpSocket
# tcp socket的liveness探针 + always restart例子

```
apiVersion: v1
kind: Pod
metadata:
  name: goproxy
spec:
  restartPolicy: Always
  containers:
  - name: goproxy
    image: k8s.gcr.io/goproxy:0.1
    ports:
    - containerPort: 8080
    livenessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 20
```