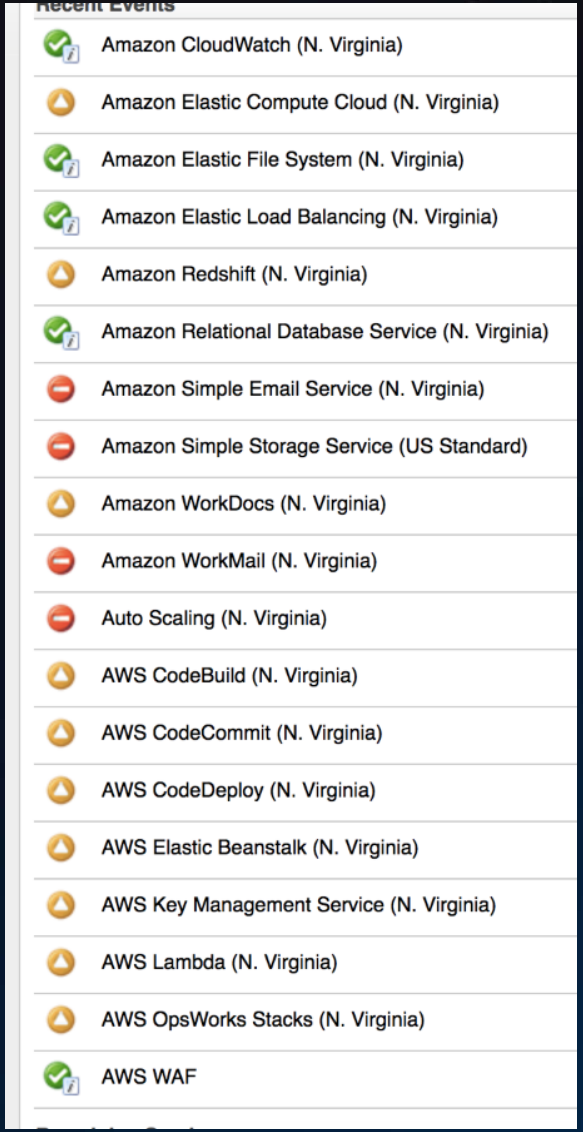
HUAWEI

CloudNative Lives

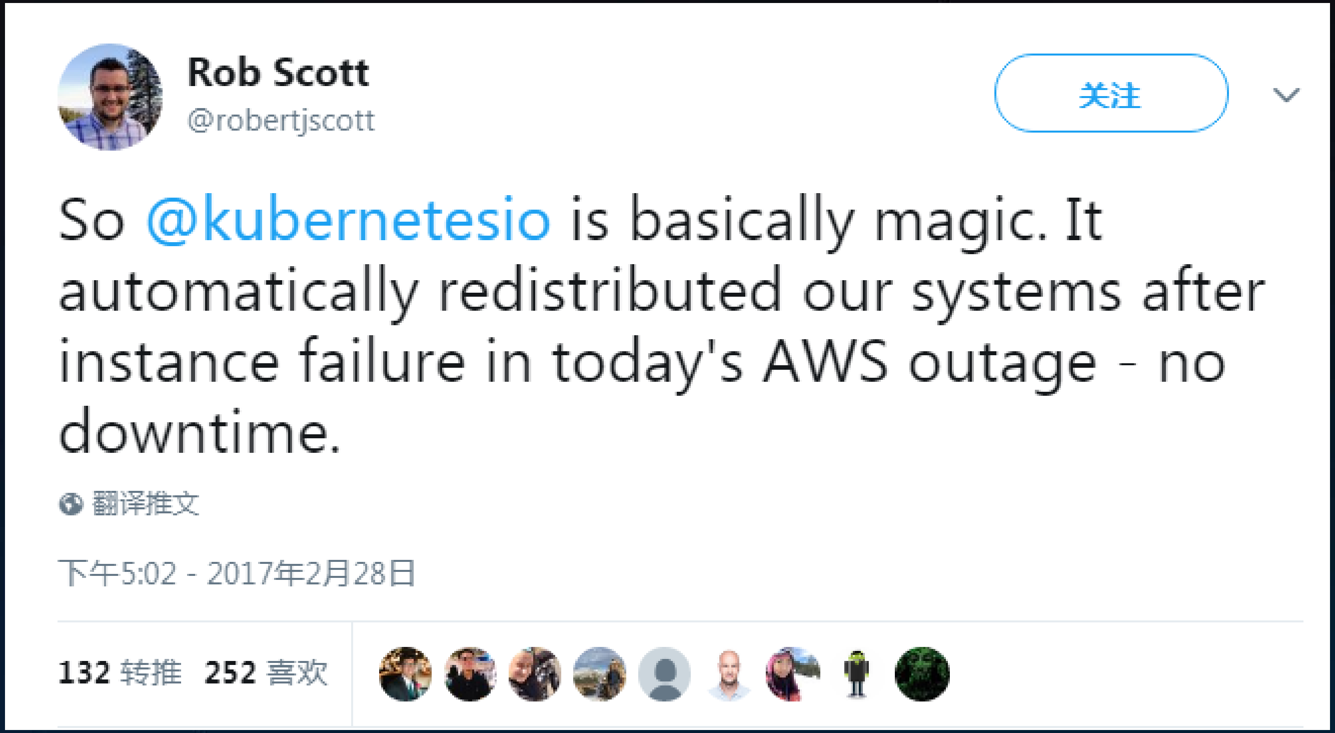
Kubernetes工作负载原理剖析和实践

华为云容器团队核心架构师 & CNCF社区主要贡献者倾心打造

Kubernetes的魔法


Recent Events

- ✓ Amazon CloudWatch (N. Virginia)
- △ Amazon Elastic Compute Cloud (N. Virginia)
- ✓ Amazon Elastic File System (N. Virginia)
- ✓ Amazon Elastic Load Balancing (N. Virginia)
- △ Amazon Redshift (N. Virginia)
- ✓ Amazon Relational Database Service (N. Virginia)
- ⊖ Amazon Simple Email Service (N. Virginia)
- ⊖ Amazon Simple Storage Service (US Standard)
- △ Amazon WorkDocs (N. Virginia)
- ⊖ Amazon WorkMail (N. Virginia)
- ⊖ Auto Scaling (N. Virginia)
- △ AWS CodeBuild (N. Virginia)
- △ AWS CodeCommit (N. Virginia)
- △ AWS CodeDeploy (N. Virginia)
- △ AWS Elastic Beanstalk (N. Virginia)
- △ AWS Key Management Service (N. Virginia)
- △ AWS Lambda (N. Virginia)
- △ AWS OpsWorks Stacks (N. Virginia)
- ✓ AWS WAF


Rob Scott
@robertjscott

关注

So @kubernetesio is basically magic. It automatically redistributed our systems after instance failure in today's AWS outage - no downtime.

翻译推文

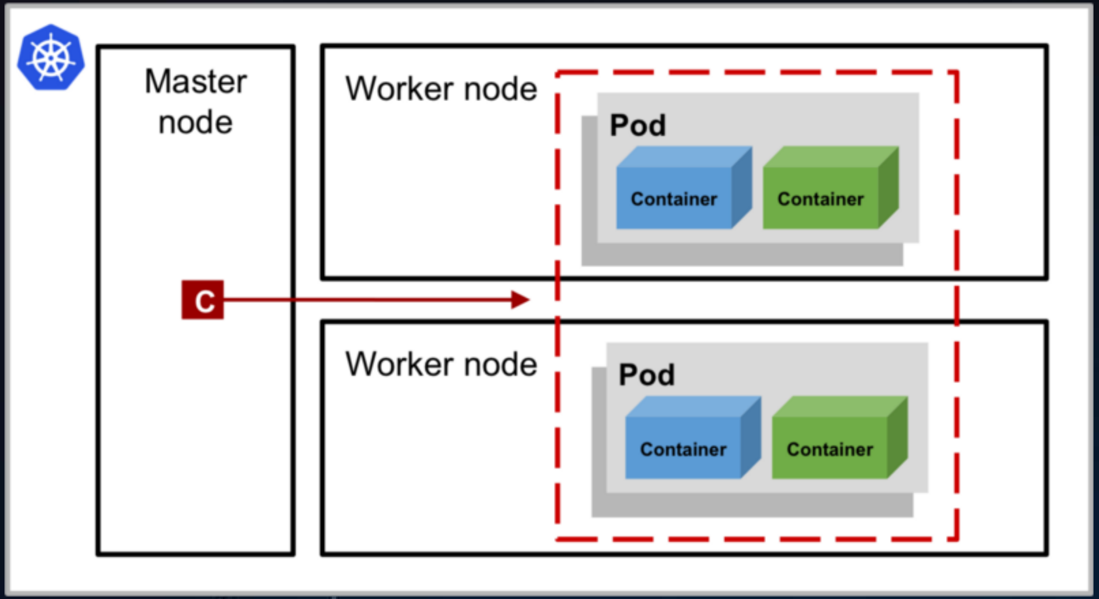
下午5:02 - 2017年2月28日

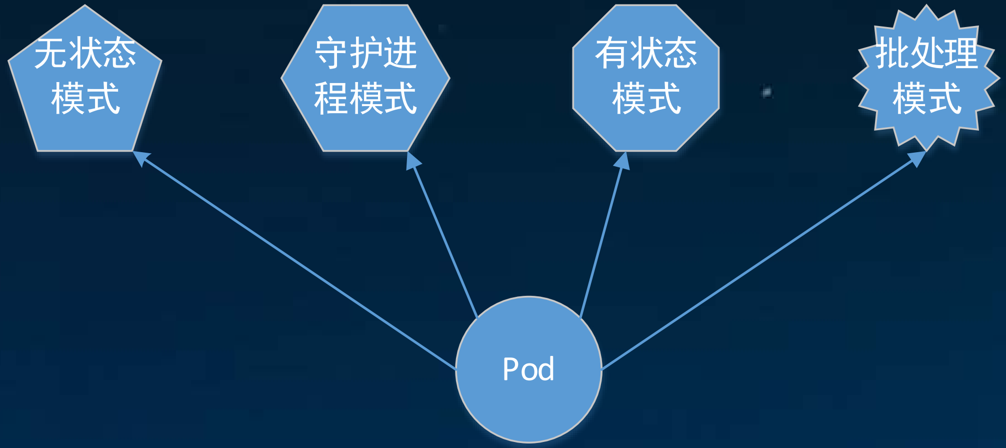
132 转推 252 喜欢

Kubernetes是容器化*应用程序模式*的平台。
这些**模式**使应用程序更易于部署，管理，扩展和从故障中恢复 – 这是”魔法”。

CloudNativeLives

一个简化版的k8s集群





大 纲

- **无状态模式**
- 有状态模式
- 守护进程模式
- 批处理模式

无状态模式

- 不必为你的应用保持状态/持久化数据
- 典型应用代表：Nginx，Tomcat
- Replication Controller
- ReplicaSet
- Deployment

Replication Controller

```
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
```

- ✓ 默认情况下，删除RC会级联删除Pod；只删RC，不影响Pod：
  $ kubectl delete rc --cascade=false
- ✓ 滚动升级，需要两个RC（不同label selector）来配合实现：旧的RC副本数-1，新的RC副本数+1，逐步完成。

ReplicaSet

- **下一代的Replication Controller，区别是支持“基于集合”的label selector**

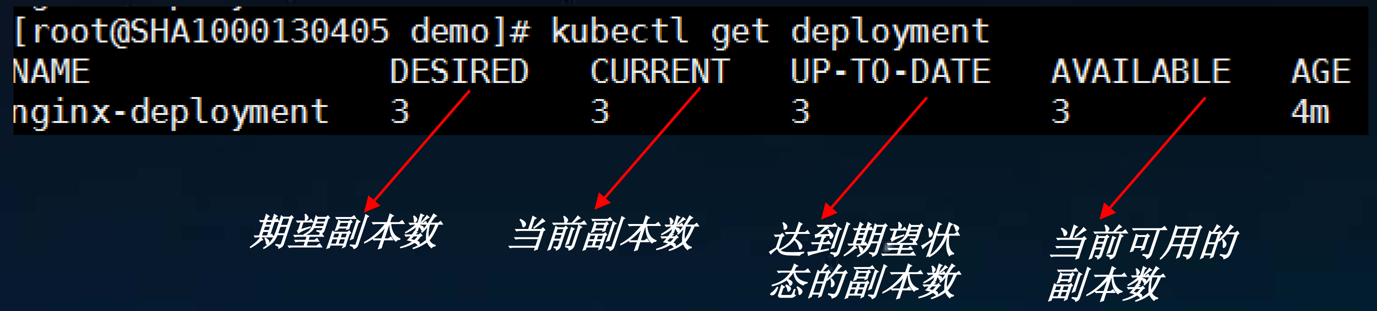
```
kind: ReplicaSet
metadata:
  name: frontend
spec:
replicas: 3
  selector:
    matchLabels:
      app: guestbook
    matchExpressions:
      - {key: app, operator: In, values: [guestbook]} // NotIn, Exists, DoesNotExist
  template:
    metadata:
      labels:
        app: guestbook
```

升级重器 - Deployment

- Deployment提供了声明式、自定义策略的Pod*升级*支持

  - 重建/滚动

```
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```



```
[root@SHA1000130405 demo]# kubectl get deployment
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            3           4m
```

升级**Deployment:**

```
$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
```

查询升级状态：

```
$ kubectl rollout status deployment/nginx-deployment
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
deployment "nginx-deployment" successfully rolled out
```

查询升级历史

```
$ kubectl rollout history deploy/nginx-deployment
```

弹性扩/缩容

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

Deployment常见用法

- 升级/回滚

```
kubectl set resources deployment nginx-deployment -c=nginx --limits=cpu=200m,memory=512Mi
kubectl rollout history deployment/nginx-deployment --revision=2
kubectl rollout undo deployment/nginx-deployment --to-revision=2
```

- 暂停/恢复

```
kubectl rollout pause deployment/nginx-deployment
kubectl rollout resume deploy/nginx-deployment
```

- 弹性、按比例扩/缩容

```
kubectl scale deployment nginx-deployment --replicas=10
kubectl autoscale deployment nginx-deployment --min=10 --max=15 --cpu-percent=80 # 需要和HPA联动
```

maxSurge=3; **25%(默认)**

maxUnavailable=2; **25%（默认）**

有状态模式

- 典型应用：Zookeeper, MongoDB, MySQL, etcd
- StatefulSet（曾用名：PetSet）
- StatefulSet的Pod和普通Pod区别: 有身份的！
- StatefulSet身份三要素：

  - 域名（网络）<- 容器IP易变
  - PVC（存储）
  - Pod Name（主机名）
- 配合headless service，PVC一起使用
- 严格的启动/删除顺序：0, 1, 2...

```yaml
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: gcr.io/google_containers/nginx-slim:0.8
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      resources:
        requests:
          storage: 1Gi
```

守护进程模式

- 典型应用：fluentd, linkderd, ceph, kube-proxy
- DaemonSet：保证每个节点**总是**运行一个Pod实例
  - NodeSelector或NodeAffinity指定Node
  - 经过（1.11 Alpha特性）/不经过调度器（不管Node状态）
  - 支持滚动升级
  - 支持级联/非级联删除

DaemonSet

```
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd-elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
```

批处理模式

- 典型应用：并发执行的作业 – batch job
  - 相关但独立的工作项：发邮件、数据扫描、文件转码
- Job
  - **保证**指定数量Pod成功运行**结束** - completions
  - 支持并行 - parallelism
  - 支持错误自动重试（10s, 20s, 40s,... 6min）
  - 删除Job会触发对应Pod删除
- CronJob
  - 基于时间调度的Job（*Cron*格式）
  - 用户可以暂停/恢复Job的周期性调度 **.spec.suspend={true,false}**
  - **管理Job -> Pod**

Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi-with-timeout
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
```

Kubernetes Job常见使用方法

- 做不同的事情

  - 扩展Job Expansion，传入参数、环境变量

- 做同样的事情

  - 工作队列形式, 与Work Queue（RabbitMQ）结合

CronJob

```
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

Kubernetes工作负载总结

- 无状态模式: 使用Deployment提供高可用、弹性扩/缩容、升级 /回滚
- 有状态模式：使用StatefulSet提供一致性，Pod的唯一/粘性的身份标识、存储，按序部署、扩缩容
- 守护进程模式：一个节点部署一个（可自定义节点范围）
- 批处理模式：并行跑多个Pod，并且保证都成功返回

预告

- 如何更好地发挥k8s的魔法，让应用的不同副本跨主机、跨机架、跨AZ，进一步提高应用的高可用，容灾，自恢复？
- 欢迎关注本系列的下一个分享：k8s调度器与调度策略。

实操！

HUAWEI

Thank You

http://zhibo.huaweicloud.com/watch/2174406

直播 每周四 晚20:00