Model and Operate Datacenter by Kubernetes at eBay

辛肖刚, Cloud Engineering Manager, ebay
梅岑恺, Senior Operation Manager, ebay
<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>US Data Centers</td>
<td>3</td>
</tr>
<tr>
<td>POPs</td>
<td>15</td>
</tr>
<tr>
<td>Managed Vms</td>
<td>200K+</td>
</tr>
<tr>
<td>Managed BMs</td>
<td>100K</td>
</tr>
<tr>
<td>Applications</td>
<td>4K</td>
</tr>
<tr>
<td>Managed Storage</td>
<td>4.5PB</td>
</tr>
</tbody>
</table>
All of us know that...

It’s not easy to manage fleet and infrastructure at scale
Infrastructure should play magic
Way to Kubernetes

- **Bare metals**
  - 2010-
  - 2010-
  - 2012-
  - 2015-
  - Now

- **OpenStack**
  - 2015-
  - Now

- **Kubernetes**
  - 2015-
  - Now

- **VMware**
  - 2010-
  - Now

- eBay
Kubernetes plays magic

Search Grid

Hadoop

PoP

Database

Frontend

VM
Kubernetes Core concept of Kubernetes - Declarative magic

WISB: What it should be

Converge & Reconcile

WIRI: What it really is
Kubernetes models applications

What is an application looks like?

- Config
- Replica
- Docker Image
- Volume
- LoadBalancer
- RBAC
- Quota
- Rolling Update

How about Kubernetes itself?

How about the fleet Kubernetes running on?
Our thinking of datacenter modeling by extending Kubernetes.
Let’s model a datacenter running Kubernetes

You need onboard something from nothing!

```yaml
apiVersion: infra.tess.io/v1alpha1
kind: GeographicalRegion
metadata:
  name: us-south
spec:

apiVersion: infra.tess.io/v1alpha1
kind: AvailabilityZone
metadata:
  name: slc01
  region: us-central

apiVersion: network.tess.io/v1alpha1
kind: SubNetwork
metadata:
  name: 10-242-158-0-24
spec:
  availabilityZone: slc01
  ipCidrRange: 10.242.158.0/24
l2Domain:
  name: rnb10-ra085
network:
  name: ebay
networkZone:
  name: production

apiVersion: infra.tess.io/v1alpha1
kind: L2Domain
metadata:
  name: rnb10-ra082
spec:
  availabilityZone: slc01
  networkZone:
    name: production

apiVersion: infra.tess.io/v1alpha1
kind: Rack
metadata:
  name: rno--mcc10--01-1110--33--06
  position:
    room: RNO:MCC10:01-1110:33
    row: RNO:MCC10:01-1110

apiVersion: infra.tess.io/v1alpha1
kind: ComputeAsset
metadata:
  name: asset0538903
  manufacturer: haye
  rack:
    name: RNO:MCC10:01-1110:33:06
  sku:
    name: 80366
  devices:
    - Labels: null
      function: network
      index: 1
      l2Domain:
        name: rnb04-ra082
      macAddress: 4C:38:D5:04:AC:85
      networkSwitch:
        name: rnb04-ra082
      speed: 1000000Mb/s
      type: Provision
    - Labels: null
      function: network
      ipAddress: 10.24.113.131
      l2Domain:
        name: rnb10-ra082
      macAddress: 4C:38:D5:04:AC:84
      networkSwitch:
        name: rnb10-ra082
      type: Management
```
Let’s model a datacenter running Kubernetes

After you define your fleet, you want a accessible compute node: Asset + Flavor + OS = ComputeNode

```yaml
apiVersion: compute.tess.io/v1alpha1
kind: OSImage
metadata:
  name: centos-atomic
spec:
  kernel_version: "3.10"
  os_version: "7.5"
  ostree_repo_url: http://ostree.ebay.com/atomic-ostree/centos/7.5.1804/docker-18.03.1-ce/ostree-3-10-3/
image_id: 42feb598-48ab-45c5-a79b-2c9bd6a53232
```
Let’s model a datacenter running Kubernetes

After you define your fleet, you want a accessible compute node: Asset + Flavor + OS = ComputeNode

apiVersion: infra.tess.io/v1alpha1
kind: ComputeFlavor
metadata:
  name: p3g6-minion-nudata
spec:
cpu: 
  spec: 
    cores: 48
    frequency: "0"
memory: 
  - DRAMType: "" 
    memoryModuleFormat: ""
spec: 
  frequency: "0"
  size: 364G

storage: 
  - disk: 
    blockDevice: 
      device: 
        Labels: null
        function: disk
        name: sda
      partitions: 
        - blockDevice: 
          device: 
            Labels: null
            size: 3M
          bootPartition: true
          fileSystem: 
            fsType: bioboot
            name: bioboot
            size: 1M
            type: gpt
        - blockDevice: 
          device: 
            Labels: null
            size: 3M
          bootPartition: true
          fileSystem: 
            fsType: ext4
            name: /boot
            primary: true
            size: 384M
            type: gpt
      - blockDevice: 
        device: 
          Labels: null
          size: 8552M
          name: pv.01
          primary: true
          type: gpt
Let’s model a datacenter running Kubernetes

After you define your fleet, you want an accessible compute node: Asset + Flavor + OS = ComputeNode

```yaml
apiVersion: compute.tess.io/v1alpha1
kind: ComputeNode
metadata:
  name: tess-node-zzq4c
spec:
  assetName: asset0538093
  flavor: bd3g6-minion-hadoop
  livenessProbes:
    - failureThreshold: 3
      initialDelaySeconds: 600
      periodSeconds: 3
      tcpSocket:
        port: "22"
        timeoutSeconds: 3
  osImage:
    name: centos-atomic-hadoop
    provider: foreman
```
Let’s model a datacenter running Kubernetes

---

You have your compute node now, all you need is to configure it by a configuration management orchestration. We use SaltStack.

```yaml
apiVersion: salt.tess.io/v1alpha1
kind: SaltMaster
metadata:
  name: salt-master-test
  namespace: default
spec:
  computeNodeRef: 
    name: group
    secretRef:
      name: pillarsecret
    pillar:
    - environment: other0.28
  gitRepo:
    directory: salt/other/other0.28
    path: /srv/salt/other0.28
    repository: https://git.ebay.com/tessops.git
    name: saltStateother0.28
  pillars:
  - environment: other0.28
  gitRepo:
    directory: pillar/other/other0.28
    path: /srv/pillar/other0.28
    repository: https://git.ebay.com/tessops.git
    name: pillar28
  - environment: hrtv0.28
  gitRepo:
    directory: salt/hrtv/hrtv0.28
    path: /srv/salt/hrtv0.28
    repository: https://git.ebay.com/tessops.git
    name: saltStatehrtv0.28
  pillars:
  - environment: hrtv0.28
  gitRepo:
    directory: pillar/hrtv/hrtv0.28
    path: /srv/pillar/hrtv0.28
    repository: https://git.ebay.com/tessops.git
    name: pillarhrtv0.28
```

---

```yaml
apiVersion: salt.tess.io/v1alpha1
kind: SaltMinion
metadata:
  name: salt-minion1
  namespace: default
spec:
  computeNodeRef: 
    name: group
    secretRef:
      name: secret1-grains
      name: defaultGrain
      secretRef: 
        name: secretgrain
      saltMaster: salt-master-test
```

---

```yaml
apiVersion: salt.tess.io/v1alpha1
kind: SaltDeployment
metadata:
  namespace: 90
  name: sd-kubernetes-master-99
spec:
  minionsConfig:
    selector:
      matchLabels:
        k8s.tess.io/workload: "99"
      k8s.tess.io/role: master
    strategy:
      name: rack-by-rack
    template:
      spec:
        grains:
          - secretRef:
            name: mastergrain
          - secretRef:
            name: salt-minion-grain
          roles:
            - kubernetes-master
        saltEnvironments:
          - gitrepo:
            directory: kubernetes/salt/99
            repository: https://git.ebay.com/tessops.git
            revision: ab336e19b530f3e7299e9480f601d89b2da
            targetPillar:
              - pillar2
            ...
        applicationInstances:
          - etcd
          - apiserver
          - controller-manager
          - scheduler
    status:
      minionsStatus:
        desiredMinions: 4
        lastProbeTime: 2018-06-31T22:45:18Z
        transactionName: sd-kubernetes-master-99
        updatedMinions: 2
      pause: true
```
Let’s model a datacenter running Kubernetes

It’s time to spin up a Kubernetes cluster!

```
apiVersion: k8s.tess.io/vlalpha1
kind: K8sCluster
metadata:
  labels:
    az: phx02
    realm: production
    region: us-central-1
  name: "21"
spec:
  availabilityZone:
    name: phx02
  description: Tess production cluster in phx02
  networkZone:
    name: production
    provider: c3
status:
  loadBalancer:
    dns: api.system.svc.21.tess.io
    ip: 10.137.209.14
```
Easy operation
How to flex up some nodes?

Step 1. Find some assets not used

```
xminELM-SHC-16501473:~$ tctl get computeassets
NAME    AGE
asset00489020 276d
asset00489021 276d
asset00489022 276d
asset00489023 276d
asset00489024 276d
asset00489025 276d
asset00489026 276d
asset00489027 276d
asset00489028 276d
asset00489029 276d
asset00489030 276d
asset00489031 276d
asset00489032 276d
asset00489033 276d
```

Step 2. Create a ComputeNode

```
xminELM-SHC-16501473:~$ cat computenode.yaml
kind: ComputeNode
apiVersion: tess.io/v1
metadata:
  name: kubernetes-master-6
labels:
  role: master
spec:
  selector:
    role: master
tess.io/etcdd-storage-provider: host
assetName: asset00489191
flavor: p2bg5-master-std
osImage: centos-atomic
provider: foreman
livenessProbes:
  - Exec:
      command:
      - HCGet:
        - TCP
        - '22'
        - Host:
        - '10.5.0.1'
        InitialDelaySeconds: 600
        TimeoutSeconds: 3
        PeriodSeconds: 0
        SuccessThreshold: 0
        FailureThreshold: 3
```

Step 3. Relax and have a cup of coffee
What if salt master down?

Step 1. Find some compute nodes

Step 2. Create a SaltMaster

Step 3. Relax and have a cup of coffee
How to upgrade a cluster?

Upgrade Kubernetes core components

```yaml
apiVersion: k8s.tess.io/v1alpha1
kind: K8sDeployment
metadata:
  namespace: default
  name: k8s-99
labels:
  k8s.tess.io/cluster: 99
spec:
  version: release-0.33.0
  repository: https://git.ebay.com/tess/tessops.git
  saltDeployments:
    - name: sd-kubernetes-master-99
      type: kubernetes-master
    saltEnvironments:
      - name: kubernetes
        revision: ab136e189b1881ebf2769a949bf6b1cd80a38c6a
        grains:
          - name: mastergrain
            roles:
              - kubernetes-master
              - name: sd-minion-nudata-99
        deploymentStrategy: rack-by-rack
      type: kubernetes-node
      saltEnvironments:
        - name: kubernetes
          revision: ab136e189b1881ebf2769a949bf6b1cd80a38c6a
          grains:
            - name: miniongrain
              roles:
              - kubernetes-pool
              - name: sd-minion-gpu-99
      status:
```

Upgrade addons

```yaml
apiVersion: k8s.tess.io/v1alpha1
kind: K8sDeployment
metadata:
  namespace: default
  name: k8s-addon-release-0.33.0
labels:
  k8s.tess.io/cluster: 99
spec:
  version: release-0.33.0
  repository: https://git.ebay.com/tess/tessops.git
  saltDeployments:
    - name: sd-salt-master-99
      saltEnvironments:
        - name: addon
          secretPillars:
            - name: kubesuds-99
            - name: models-99
          applicationInstances:
            - etcd
            - apiserver
            - kubelet
            - kube-proxy
            - models
            - kubesuds
      status:
        version: release-0.33.0
        saltDeployments:
          - name: sd-salt-master-99
```

......
Kubernetes is amazing on its simple architecture
Model + Controller is the key concept of Kubernetes
It’s easy to extend Kubernetes API and write your controller based on list/watch
ebay uses Kubernetes to model and operate its datacenter

<table>
<thead>
<tr>
<th>KafkaCluster, HadoopCluster, MongoDB, ESCluster ……</th>
<th>Application Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>K8sCluster, K8sAddons, K8sDeployment</td>
<td>Infrastructure Service</td>
</tr>
<tr>
<td>SaltMaster, SaltMinion, SaltDeployment</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>Region, AvailabilityZone, NetworkZone, L2Domain Rack, NetworkDevice, ComputeAsset</td>
<td>Fleet (Compute, Network, Storage)</td>
</tr>
</tbody>
</table>
We are hiring!

xnxin@ebay.com
cmei@ebay.com