EKS Workshop v1.0

准备工作

IAM 配置

- EKS 集群角色（通过 Console 创建时必须）
  - Trusted Entity: eks.amazonaws.com
  - AWS 托管策略:
    - AmazonEKSClusterPolicy
    - AmazonEKSServicePolicy
- EKS 节点组角色（通过 Console 创建时必须）
  - Trusted Entity: ec2.amazonaws.com.cn
  - AWS 托管策略:
    - AmazonEKS_CNI_Policy
    - AmazonEKSWorkerNodePolicy
    - AmazonEC2ContainerRegistryReadOnly
- eksctl 管理用户（通过命令行创建时必须）
  - AWS 托管策略:
    - AmazonEC2FullAccess
    - AWSCloudFormationFullAccess
    - AmazonEC2ContainerRegistryReadOnly
  - 客户托管策略:
    - 从以下 GitHub 拉取 IAM Policy

      ```
      $ git clone https://github.com/ryanlao67/aws-iam-template.git
      ```

    - 在 IAM 策略中依次创建

安装 EKS 相关工具

- 安装 eksctl: https://docs.aws.amazon.com/zh_cn/eks/latest/userguide/eksctl.html
  - 0.15.0 正式版本：
    - Add support for KMS encryption provider (#1897)
    - Add support for China regions (#1860)
    - Add region Beijing (cn-north-1) (#1741)
    - Add region Ningxia (#1720)
    - Support addons for China regions, refactor setting container image for addons (#1867)
    - Add support for Kubernetes 1.15 (#1917 #1916)
    - Add support for Bottlerocket NodeGroups (#1918 #1919)

```
# MacOS: https://github.com/weaveworks/eksctl/releases/download/0.15.0/eksctl
# Linux: https://github.com/weaveworks/eksctl/releases/download/0.15.0/eksctl
# Windows: https://github.com/weaveworks/eksctl/releases/download/0.15.0/eksc
```

```
# Example installation on MacOS
$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/dow
$ sudo mv /tmp/eksctl /usr/local/bin
$ eksctl version
  0.15.0
```

- 安装 aws-iam-authenticator: https://docs.aws.amazon.com/zh_cn/eks/latest/userguide/install-aws-iam-authenticator.html
- 安装 Kubectl: https://docs.aws.amazon.com/zh_cn/eks/latest/userguide/install-kubectl.html
  - 中国区域可以直接从 kops-cn 的 GitHub 下载：

```
# Available version:
### v1.15.5 / v1.15.7
### v1.14.6 / v1.14.8
### v1.13.2 / v1.13.5 / v1.13.10 / v1.13.12
### v1.12.7 / v1.12.8 / v1.12.9 / v1.12.10
### v1.11.6 / v1.11.7 / v1.11.8 / v1.11.9
### v1.10.3 / v1.10.6 / v1.10.11
### v1.9.3 / v1.9.6 / v1.9.8
$ VERSION='<kubectl_version>'
$ curl -L https://s3.cn-north-1.amazonaws.com.cn/kops-bjs/fileRepository/kube
$ chmod +x kubectl
$ sudo mv kubectl /usr/local/bin
$ kubectl version --short --client
  Client Version: <kubectl_version>
```

- 安装 AWS CLI: https://docs.amazonaws.cn/cli/latest/userguide/install-cliv1.html

准备密钥对

- 在 Console 上访问 EC2 页面，并定位到密钥对
- 点击 “Create key pair”，指定名称，例如：bjs
- 文件格式选择 pem，并点击创建密钥对
- 文件会下载至本地，打开 Terminal，并 cd 到密钥对下载目录
- 运行如下命令创建 pub 文件用于之后访问工作节点：

```
$ chmod 400 bjs.pem
$ ssh-keygen -y -f bjs.pem > bjs.pub
$ mv bjs.pub ~/.ssh/bjs.pub
```

准备 EKS 集群配置

CLUSTER YAML

```
$ vim cluster.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eks-demo-cluster
  region: cn-north-1
```

NODEGROUP1 YAML

- 创建一个由2个 m5.large 节点组成的 worker group，并部署在私有子网中
- 能够使用本地的 ssh key 登录相应的 worker 节点
- 制定两个label：
  - role: workers
  - usage: stable_cluster

```
$ vim nodegroup1.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eks-demo-cluster
  region: cn-north-1

nodeGroups:
  - name: eks-demo-workers1
    labels: { role: workers, usage: stable_cluster }
    instanceType: m5.large
    desiredCapacity: 2
    privateNetworking: yes
    ssh:
      publicKeyPath: ~/.ssh/bjs.pub
```

NODEGROUP2 YAML

- 创建一个由1个 c5.large 节点组成的 worker group，并部署在私有子网中
- 能够使用本地的 ssh key 登录相应的 worker 节点
- 制定两个label：
  - role: workers
  - usage: stable_cluster

```
$ vim nodegroup2.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eks-demo-cluster
  region: cn-north-1

nodeGroups:
  - name: eks-demo-workers2
    labels: { role: workers, usage: stable_cluster }
    instanceType: c5.large
    desiredCapacity: 1
    privateNetworking: yes
    ssh:
      publicKeyPath: ~/.ssh/bjs.pub
```

NODEGROUP3 YAML

- 创建一个由1个 t3.large 和 t3.medium 组成的 spot fleet worker group，并部署在私有子网中
- 可以根据实际用量进行弹性伸缩，最少2个节点，最多5个节点
- 能够使用本地的 ssh key 登录相应的 worker 节点
- 制定两个label：
    - role: workers
    - usage: spot_cluster

```
$ vim nodegroup3.yaml

apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: eks-demo-cluster
  region: cn-north-1

nodeGroups:
  - name: eks-demo-workers3
    labels: { role: workers, usage: spot_cluster }
    minSize: 2
    maxSize: 5
    instancesDistribution:
      maxPrice: 0.2
      instanceTypes: ["t3.large", "t3.medium"]
      onDemandBaseCapacity: 0
      onDemandPercentageAboveBaseCapacity: 50
      spotInstancePools: 2
    privateNetworking: yes
    ssh:
      publicKeyPath: ~/.ssh/bjs.pub
```

创建 EKS 集群

创建 MASTER

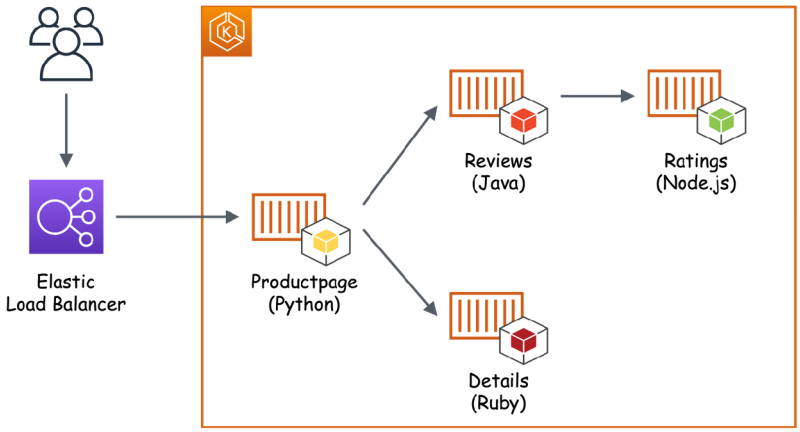
```
# Create EKS cluster
$ eksctl create cluster \
    --config-file=cluster.yaml \
    --profile bjs

# Verify created cluster
$ eksctl get cluster \
    --profile bjs
  NAME                REGION
  eks-demo-cluster    cn-north-1
```

创建 NODEGROUP

```
# Create node groups
$ eksctl create nodegroup \
    --config-file=nodegroup1.yaml \
    --profile bjs

$ eksctl create nodegroup \
    --config-file=nodegroup2.yaml \
    --profile bjs

$ eksctl create nodegroup \
    --config-file=nodegroup3.yaml \
    --profile bjs

# Verify node groups
$ eksctl get nodegroup \
    --cluster eks-demo-cluster \
    --profile bjs
  CLUSTER           NODEGROUP          CREATED               MIN SIZE  MAX SIZE  DESIR
  eks-demo-cluster  eks-demo-workers1  2020-03-XXTXX:XX:XXZ  2         2         2
  eks-demo-cluster  eks-demo-workers2  2020-03-XXTXX:XX:XXZ  1         1         1
  eks-demo-cluster  eks-demo-workers3  2020-03-XXTXX:XX:XXZ  2         5         0
```

创建 Demo 应用

应用架构



制作镜像

- 从以下 GitHub 拉取示例应用代码
  - https://github.com/ryanlao67/bookinfo-productpage.git
  - https://github.com/ryanlao67/bookinfo-reviews.git
  - https://github.com/ryanlao67/bookinfo-ratings.git
  - https://github.com/ryanlao67/bookinfo-details.git

```
$ git clone https://github.com/ryanlao67/bookinfo-productpage.git
$ git clone https://github.com/ryanlao67/bookinfo-reviews.git
$ git clone https://github.com/ryanlao67/bookinfo-ratings.git
$ git clone https://github.com/ryanlao67/bookinfo-details.git
```

- 为 bookinfo-productpage 制作 Docker 镜像

```
$ cd bookinfo-productpage
$ docker build -t bookinfo-productpage:1.0 .
```

- 为 bookinfo-reviews 制作 Docker 镜像

```
# 原始代码需要先通过如下命令生成war包
$ cd bookinfo-reviews
$ docker run --rm -u root -v "$(pwd)":/home/gradle/project -w /home/gradle/projec

# 生成war包后，制作Docker镜像
# 目前示例代码中已包含war包，可以通过如下命令直接制作Docker镜像
$ cd bookinfo-reviews/reviews-wlpcfg
$ docker build -t bookinfo-reviews:1.0 .
```

- 为 bookinfo-ratings 制作 Docker 镜像

```
$ cd bookinfo-ratings
$ docker build -t bookinfo-ratings:1.0 .
```

- 为 bookinfo-details 制作 Docker 镜像

```
$ cd bookinfo-details
$ docker build -t bookinfo-details:1.0 .
```

- 完成镜像制作后，查看本地镜像

```
$ docker images
  REPOSITORY             TAG                 IMAGE ID            CREATED
  bookinfo-ratings       1.0                 1ee994c63a8f        2 minutes ago
  bookinfo-reviews       1.0                 49772a269c04        3 minutes ago
  bookinfo-details       1.0                 09518b297a40        3 minutes ago
  bookinfo-productpage   1.0                 1cd26d55be02        13 minutes ago
  python                 3.6-slim            d3ae39a2a3a1        2 weeks ago
  node                   12-slim             26932a190e66        2 weeks ago
  ruby                   2.6.3-slim          1c75fac01180        7 months ago
  websphere-liberty      19.0.0.5-javaee8    99f7cc7fc995        8 months ago
  gradle                 4.8.1               8b2989808a5c        20 months ago
```

镜像推送

- 通过 AWS CLI 创建 ECR 仓库

```
$ aws ecr create-repository \
    --repository-name eks-demo/bookinfo-productpage \
    --profile bjs

$ aws ecr create-repository \
    --repository-name eks-demo/bookinfo-reviews \
    --profile bjs

$ aws ecr create-repository \
    --repository-name eks-demo/bookinfo-ratings \
    --profile bjs
```

```
$ aws ecr create-repository \
    --repository-name eks-demo/bookinfo-details \
    --profile bjs
```

- 通过 AWS CLI 验证 ECR 仓库

```
$ aws ecr describe-repositories \
    --query 'repositories[*].[repositoryName]' \
    --output text \
    --profile bjs
  # ECR显示如下
  eks-demo/bookinfo-reviews
  eks-demo/bookinfo-details
  eks-demo/bookinfo-ratings
  eks-demo/bookinfo-productpage
```

- 登录 ECR

```
$ $(aws ecr get-login --no-include-email --region cn-north-1 --profile bjs)
  Login Succeeded
```

- 打标签并推送到创建的 ECR

```
# Tag images
# Account ID shown as 12-digits
$ docker tag bookinfo-productpage:1.0 855501529395.dkr.ecr.cn-north-1.amazonaws.c
$ docker tag bookinfo-reviews:1.0 855501529395.dkr.ecr.cn-north-1.amazonaws.com.c
$ docker tag bookinfo-ratings:1.0 855501529395.dkr.ecr.cn-north-1.amazonaws.com.c
$ docker tag bookinfo-details:1.0 855501529395.dkr.ecr.cn-north-1.amazonaws.com.c

# Push to ECR
# Account ID shown as 12-digits
$ docker push 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo-
$ docker push 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo-
$ docker push 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo-
$ docker push 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo-
```

创建应用 YAML

- bookinfo-productpage

```
$ vim bookinfo-productpage.yaml

apiVersion: v1
kind: Service
metadata:
  name: productpage
  labels:
    app: productpage
    service: productpage
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: productpage
```

```
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: productpage-v1
  labels:
    app: productpage
    version: v1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: productpage
      version: v1
  template:
    metadata:
      labels:
        app: productpage
        version: v1
    spec:
      serviceAccountName: default
      containers:
      - name: productpage
        image: 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
      nodeSelector:
        usage: stable_cluster
```

- bookinfo-reviews

```
$ vim bookinfo-reviews.yaml

apiVersion: v1
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v1
  labels:
    app: reviews
    version: v1
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v1
  template:
    metadata:
      labels:
        app: reviews
        version: v1
    spec:
      serviceAccountName: default
      containers:
      - name: reviews
        image: 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo
        imagePullPolicy: Always
        ports:
        - containerPort: 9080
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - productpage
            topologyKey: "kubernetes.io/hostname"
      nodeSelector:
        usage: stable_cluster
```

- bookinfo-ratings

```
$ vim bookinfo-ratings.yaml

apiVersion: v1
kind: Service
metadata:
  name: ratings
  labels:
    app: ratings
    service: ratings
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: ratings
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-v1
  labels:
```

```
    app: ratings
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ratings
      version: v1
  template:
    metadata:
      labels:
        app: ratings
        version: v1
    spec:
      serviceAccountName: default
      containers:
      - name: ratings
        image: 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
      nodeSelector:
        usage: spot_cluster
```

- bookinfo-details

```
$ vim bookinfo-details.yaml

apiVersion: v1
kind: Service
metadata:
  name: details
  labels:
    app: details
    service: details
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: details
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: details-v1
  labels:
    app: details
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: details
      version: v1
```

```
  template:
    metadata:
      labels:
        app: details
        version: v1
    spec:
      serviceAccountName: default
      containers:
      - name: details
        image: 855501529395.dkr.ecr.cn-north-1.amazonaws.com.cn/eks-demo/bookinfo
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
      nodeSelector:
        usage: stable_cluster
```

部署 Demo 应用

- 通过 kubectl 部署上述4个服务

```
$ kubectl apply -f bookinfo-productpage.yaml
$ kubectl apply -f bookinfo-reviews.yaml
$ kubectl apply -f bookinfo-ratings.yaml
$ kubectl apply -f bookinfo-details.yaml
```

- 查看 pod 运行状态

```
$ kubectl get pods -owide

NAME                              READY   STATUS    RESTARTS   AGE   IP
details-v1-56d6c64f9-jfxrc        1/1     Running   0          36s   192.168.180.1
productpage-v1-6cf5b9797-gjtzc    1/1     Running   0          61s   192.168.96.76
productpage-v1-6cf5b9797-lrdlw    1/1     Running   0          61s   192.168.174.2
ratings-v1-6d5885c596-sg2rv       1/1     Running   0          43s   192.168.103.2
reviews-v1-b75ddfc84-k2rzv        1/1     Running   0          52s   192.168.108.1
```

- 查看 service 运行状态

```
$ kubectl get svc

NAME          TYPE           CLUSTER-IP       EXTERNAL-IP
details       ClusterIP      10.100.76.194    <none>
kubernetes    ClusterIP      10.100.0.1       <none>
productpage   LoadBalancer   10.100.93.199    aee3c98f5645611eaa2d406ab5237157-91
ratings       ClusterIP      10.100.223.103   <none>
reviews       ClusterIP      10.100.240.8     <none>
```

- 访问应用 http://{elb_endpoint}:9080/productpage

参考

- eksctl 配置文件示例：https://github.com/weaveworks/eksctl/tree/master/examples
- 如果拉取镜像比较慢可以使用 nwcd 镜像，如下：

```
$ docker pull dockerhub.mirror.nwcdcdn.cn/library/<image>:<tag>
# 例如：docker pull dockerhub.mirror.nwcdcdn.cn/library/node:12-slim
$ docker tag dockerhub.mirror.nwcdcdn.cn/library/node:12-slim node:12-slim
```

思考

- Q1: 能不能通过一个 YAML 文件创建包含所有 NodeGroup 的 EKS 集群？
- Q2: 有什么方式可以快速实现应用之间的安全隔离吗？
- Q3: 怎么使用 ALB 作为运行在 EKS 集群里的应用的前端入口？