

eks 部署CICD

前言

1 构建EKS集群

1.1 创建master节点

1.2 创建node节点

创建ebs provisioner (可选)

kubesphere install (可选)

ingress nginx install (可选)

1.3 jenkins install

2 配置ECR权限

2.1 创建私有仓库

2.2 iam角色添加权限

3 配置jenkins

3.1 安装的插件

3.2 配置凭证

3.3 添加cloud

3.4 创建测试环境

3.5 jenkins file

结尾

前言

本导材是基于AWS EKS进行的CICD持续部署方案，由于1.24后，K8s不再使用docker作为底层支持，取而代之的使用containerd作为容器运行时，containerd没有提供构建镜像的方式，所以需要采用其他方式，这里着重介绍一下kaniko，由于它不依赖于Docker 守护进程，并且完全在用户空间中执行Dockerfile中的每个命令，这使得无需借助docker守护进程就可以完成镜像的构建上传一系列操作，关于kaniko的使用方法以及介绍可以参考

<https://github.com/GoogleContainerTools/kaniko>，流程大概分为，用户在jenkins执行构建部署，jenkins在eks中启动agent实例构建镜像然后推送到ECR，然后根据jenkinsfile里的配置，agent从ECR拉取镜像部署在eks中。本导材还附带了eks集群的创建，和ecr仓库的权限部分的设

置，如果这方面您还不够了解，可以参考EKS文档

https://docs.aws.amazon.com/zh_cn/eks/latest/userguide/getting-started.html。

1 构建EKS集群

为了尽可能简单快速的入门，本导材大部分设置采用的default参数，如果你创建的集群是用于生产环境建议你详细了解eks的创建参数，另外在执行下列操作前，请确保你已经安装AWS CLI和kubectl 等工具

1.1创建master节点

cli 和kubectl下载，只适用于amd架构的linux

```
▼ Plain Text |
1  # cli  install
2  curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip -o awscliv2.zip
3  unzip awscliv2.zip
4  ./aws/install
5  mv /usr/local/bin/aws /bin/aws
6  aws --version
7
8  # kubectl install
9  curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
10 chmod +x kubectl
11 mv kubectl /usr/bin/
```

创建master节点需要使用的权限角色

```

1 cat >eks-cluster-role-trust-policy.json <<EOF
2 {
3     "Version": "2012-10-17",
4     "Statement": [
5         {
6             "Effect": "Allow",
7             "Principal": {
8                 "Service": "eks.amazonaws.com"
9             },
10            "Action": "sts:AssumeRole"
11        }
12    ]
13 }
14 EOF
15
16 aws iam create-role --role-name AmazonEKSClusterRole --assume-role-policy-
document file://"eks-cluster-role-trust-policy.json"
17 aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonEKSC
lusterPolicy --role-name AmazonEKSClusterRole

```

创建master节点

```

1 #可以通过命令行获取到创建的eks集群放置的网段，但是要根据你们vpc的内容调整VpcId里的参数
2 #aws ec2 describe-vpcs --query "Vpcs[?InstanceTenancy=='default'].VpcId" -
-output text
3 #aws ec2 describe-subnets --query "Subnets[?VpcId=='vpc-0068f2f40191aace
5'].[SubnetId,AvailabilityZone]"
4
5 # 创建集群的cli命令，可以调整name区域，和版本，以及role的arn
6 aws eks create-cluster --region us-east-2 --name adp-k8s-prod --kubernetes
-version 1.28 \
7     --role-arn arn:aws:iam::547384405015:role/AmazonEKSClusterRole \
8     --resources-vpc-config subnetIds=subnet-01e82b74dc8d09b8c,subnet-0e2bfff
aba035bfd5f
9 # 创建必选插件
10 aws eks create-addon --cluster-name adp-k8s-prod --addon-name coredns
11 aws eks create-addon --cluster-name adp-k8s-prod --addon-name kube-proxy
12 aws eks create-addon --cluster-name adp-k8s-prod --addon-name vpc-cni

```

1.2 创建node节点

创建node节点需要使用的权限角色

```
▼ Plain Text |
1  #node
2  cat >node-role-trust-relationship.json <<EOF
3  {
4      "Version": "2012-10-17",
5      "Statement": [
6          {
7              "Effect": "Allow",
8              "Principal": {
9                  "Service": "ec2.amazonaws.com"
10             },
11             "Action": "sts:AssumeRole"
12         }
13     ]
14 }
15 EOF
16
17 aws iam create-role \
18     --role-name AmazonEKSNodeRole \
19     --assume-role-policy-document file://"node-role-trust-relationship.json"
20 aws iam attach-role-policy \
21     --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
22     --role-name AmazonEKSNodeRole
23 aws iam attach-role-policy \
24     --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
25     \
26     --role-name AmazonEKSNodeRole
27 aws iam attach-role-policy \
28     --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
29     --role-name AmazonEKSNodeRole
```

创建node节点组

```
▼ Plain Text |
1  aws eks create-nodegroup --cluster-name adp-k8s-dev --nodegroup-name adp-n
2  odegroup-dev --node-role arn:aws:iam::172229444780:role/AmazonEKSNodeRole \
3  --subnets subnet-0969dd09119cbc764,subnet-0a5c551ab0c0d47cf --instance-type
4  s t3.medium --disk-size 20 --scaling-config minSize=1,maxSize=1,desiredSize
5  =1
```

创建efs provisioner (可选)

EBS CSI驱动程序管理EBS卷的生命周期，为EKS集群创建基于EBS的PV持久存储，这部分可以根据你们的需求，选择是否创建，另外创建EBS CSI插件之前，请确保你已经部署过了OIDC身份提供商，OIDC参考下列链接

<https://docs.aws.amazon.com/eks/latest/userguide/enable-iam-roles-for-service-accounts.html>

```
1 # EBS-driver
2
3 aws eks describe-cluster --name adp-k8s-prod --query "cluster.identity.oidc.issuer" --output text
4 cat > aws-ebs-csi-driver-trust-policy.json << EOF
5 {
6     "Version": "2012-10-17",
7     "Statement": [
8         {
9             "Effect": "Allow",
10            "Principal": {
11                "Federated": "arn:aws:iam::547384405015:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F"
12            },
13            "Action": "sts:AssumeRoleWithWebIdentity",
14            "Condition": {
15                "StringEquals": {
16                    "oidc.eks.region-code.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F:aud": "sts.amazonaws.com",
17                    "oidc.eks.region-code.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F:sub": "system:serviceaccount:kube-system:ebs-csi-controller-sa"
18                }
19            }
20        }
21    ]
22 }
23 EOF
24 # 替换region-code 和账号id 和OIDC码
25
26 aws iam create-role \
27     --role-name AmazonEKS_EBS_CSI_DriverRole \
28     --assume-role-policy-document file://aws-ebs-csi-driver-trust-policy.json"
29
30 aws iam attach-role-policy \
31     --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
32     --role-name AmazonEKS_EBS_CSI_DriverRole
33
34 aws eks create-addon --cluster-name adp-k8s-dev --addon-name aws-ebs-csi-driver \
35     --service-account-role-arn arn:aws:iam::172229444780:role/AmazonEKS_EBS_CSI_DriverRole
36
```

```
37 #部署示例应用
38 git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git
39 cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
40 echo "parameters:
41     type: gp3" >> manifests/storageclass.yaml
42 kubectl apply -f manifests/
43 kubectl get pv
```

kubesphere install (可选)

KubeSphere 提供了运维友好的向导式操作界面，帮助企业快速构建一个强大和功能丰富的容器云平台。KubeSphere 为用户提供构建企业级 Kubernetes 环境所需的多项功能，例如多云与多集群管理、Kubernetes 资源管理、DevOps、应用生命周期管理、微服务治理（服务网格）、日志查询与收集、服务与网络、多租户管理、监控告警、事件与审计查询、存储管理、访问权限控制、GPU 支持、网络策略、镜像仓库管理以及安全管理等。

```
1  kubectl apply -f https://github.com/kubesphere/ks-installer/releases/download/v3.4.1/kubesphere-installer.yaml
2
3
4
5  kubectl apply -f https://github.com/kubesphere/ks-installer/releases/download/v3.4.1/cluster-configuration.yaml
6
7
8
9  # ingress支持websocket
10 apiVersion: networking.k8s.io/v1
11 kind: Ingress
12 metadata:
13   name: kubesphere-ingress
14   namespace: kubesphere-system
15   annotations:
16     nginx.ingress.kubernetes.io/proxy-body-size: 600m
17     nginx.org/client-max-body-size: "10m"
18     nginx.ingress.kubernetes.io/proxy-read-timeout: "1800"
19     nginx.ingress.kubernetes.io/proxy-send-timeout: "1800"
20     nginx.ingress.kubernetes.io/websocket-services: proxy-public
21     nginx.org/websocket-services: proxy-public
22 spec:
23   rules:
24   - host: ks.lvtujingji.click
25     http:
26       paths:
27       - path: /
28         pathType: Prefix
29         backend:
30           service:
31             name: ks-console
32             port:
33               number: 80
34   ingressClassName: nginx
```

ingress nginx install (可选)

AWS上部署ingress入口控制器参考: <https://kubernetes.github.io/ingress-nginx/deploy/#aws>


```
1 # ingress nginx
2
3 wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller
-v1.8.2/deploy/static/provider/aws/nlb-with-tls-termination/deploy.yaml
4 aws acm request-certificate --domain-name *.lvtujiangji.click --validation-
method DNS
5 aws acm describe-certificate --certificate-arn arn:aws:acm:us-east-1:54738
4405015:certificate/6843d443-8907-4dee-99b5-ad8f45c27105
6
7 aws route53 list-hosted-zones
8
9 aws route53 change-resource-record-sets --hosted-zone-id Z03012041JX1FRPDJ
Q2NX --change-batch file://config.json
10 cat >> config.json << EOF
11 {
12     "Comment": "optional comment about the changes in this change batch requ
est",
13     "Changes": [
14         {
15             "Action": "UPSERT",
16             "ResourceRecordSet": {
17                 "Name": "ks.lvtujiangji.click.",
18                 "Type": "CNAME",
19                 "TTL": 60,
20                 "ResourceRecords": [
21                     {
22                         "Value": ""
23                     }
24                 ]
25             }
26         }
27     ]
28 }
29 EOF
30
31 需要修改 proxy-real-ip-cidr
32     service.beta.kubernetes.io/aws-load-balancer-ssl-cert
33 kubectl apply -f deploy.yaml
34
```

1.3 jenkins install

```
1  git clone https://github.com/scriptcamp/kubernetes-jenkins
2  kubectl create namespace devops-tools
3  kubectl apply -f serviceAccount.yaml
4
5  # 如果安装了ebs CSI就替换local-storage为 ebs-sc
6  cat > volume.yaml << EOF
7  ---
8  apiVersion: v1
9  kind: PersistentVolumeClaim
10 metadata:
11   name: jenkins-pv-claim
12   namespace: devops-tools
13 spec:
14   storageClassName: local-storage
15   accessModes:
16     - ReadWriteOnce
17   resources:
18     requests:
19       storage: 30Gi
20 EOF
21
22 kubectl create -f volume.yaml
23 kubectl apply -f deployment.yaml
24
25
26 cat >> service.yaml << EOF
27 apiVersion: v1
28 kind: Service
29 metadata:
30   name: jenkins-service
31   namespace: devops-tools
32   annotations:
33     prometheus.io/scrape: 'true'
34     prometheus.io/path: /
35     prometheus.io/port: '8080'
36 spec:
37   selector:
38     app: jenkins-server
39   type: ClusterIP
40   ports:
41     - port: 8080
42       targetPort: 8080
43       name: httpport
44     - port: 50000
```

```

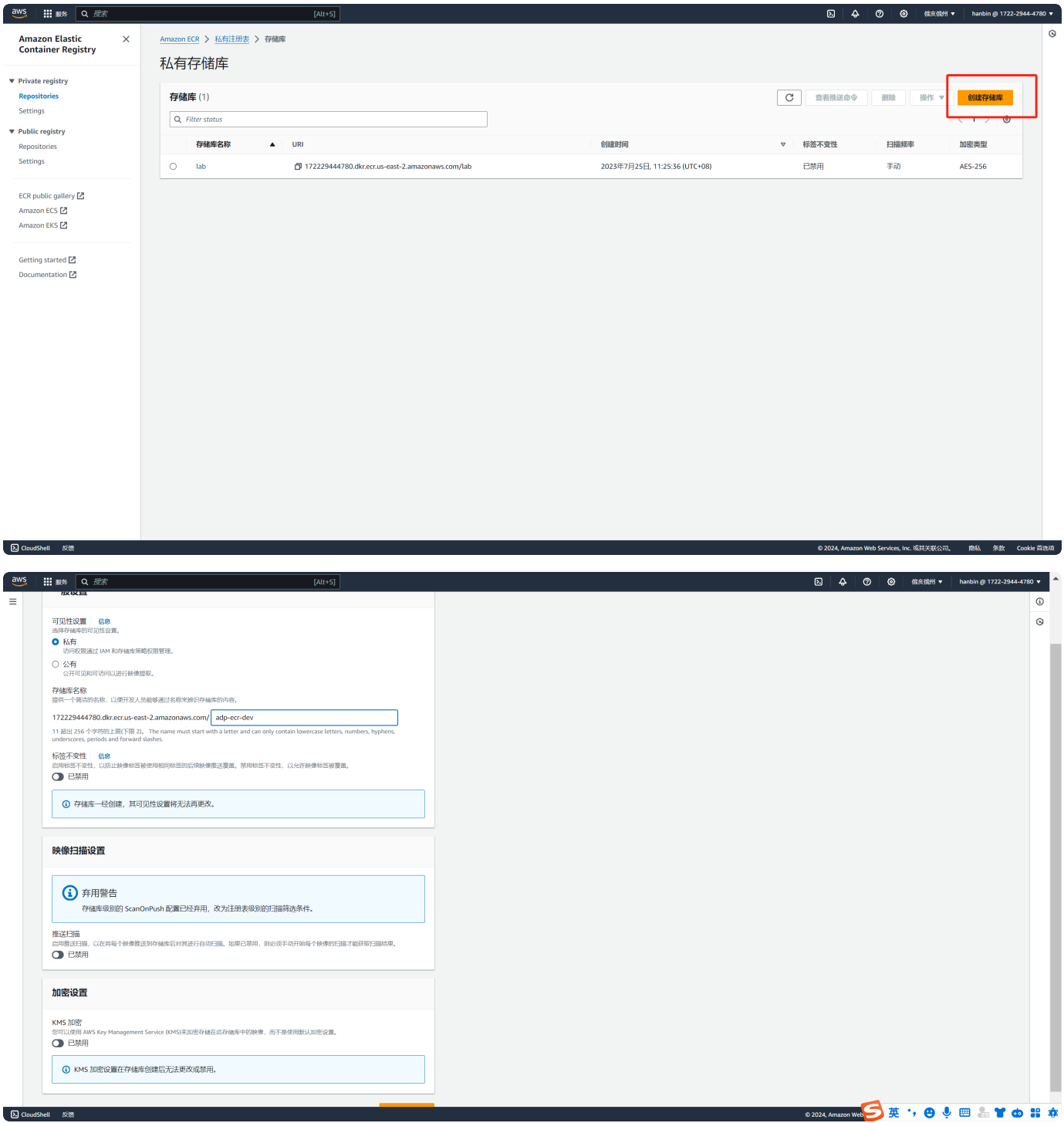
45         targetPort: 50000
46         name: jnlpport
47 EOF
48
49 # 可选
50 cat >> secret.yaml << EOF
51 ---
52 apiVersion: v1
53 kind: Secret
54 metadata:
55     name: jenkins-admin
56     namespace: devops-tools
57     annotations:
58         kubernetes.io/service-account.name: "jenkins-admin"
59 type: kubernetes.io/service-account-token
60 EOF
61
62 cat > jenkins-ingress.yaml <<EOF
63 apiVersion: networking.k8s.io/v1
64 kind: Ingress
65 metadata:
66     name: jenkins-ingress
67     namespace: devops-tools
68 spec:
69     rules:
70     - host: js.lvtujingji.click
71       http:
72         paths:
73         - path: /
74           pathType: Prefix
75         backend:
76           service:
77             name: jenkins-service
78             port:
79               number: 8080
80     ingressClassName: nginx
81 EOF
82
83
84 kubectl apply -f jenkins-ingress.yaml
85
86
87

```

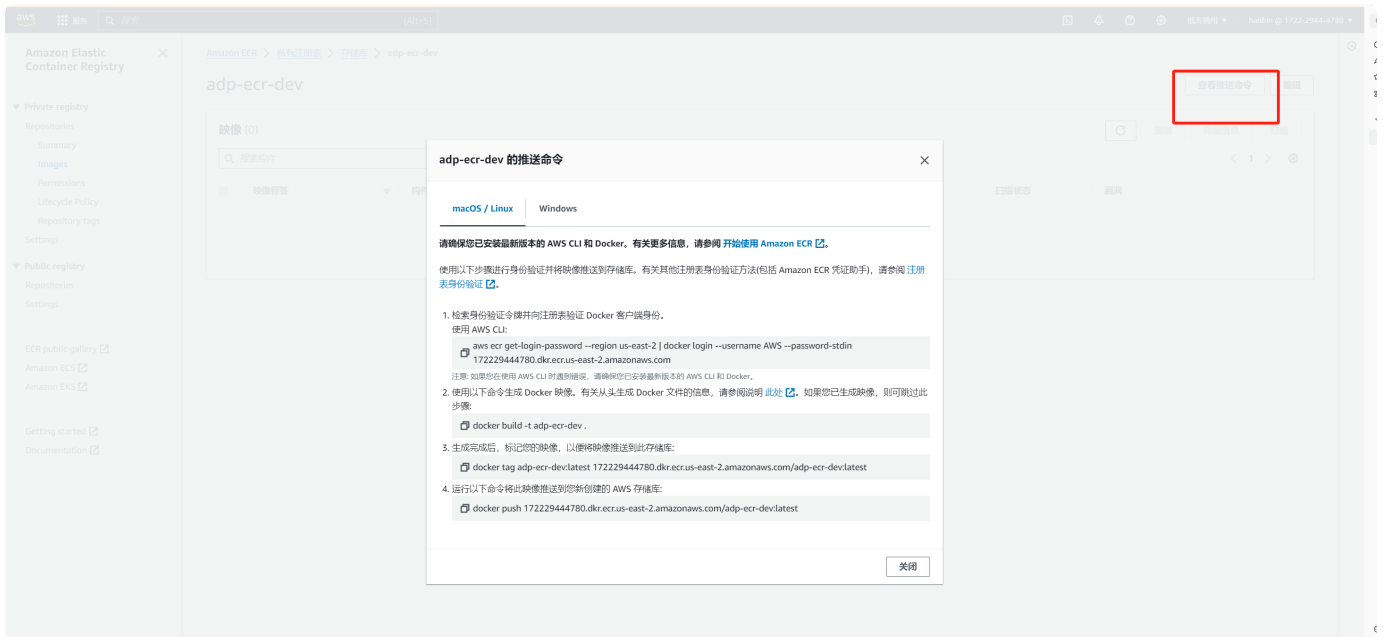
2 配置ECR权限

我们使用AWS ECRprivate仓库作为镜像载体，所以需要提前创建私有仓库和进行IAM授权

2.1 创建私有仓库

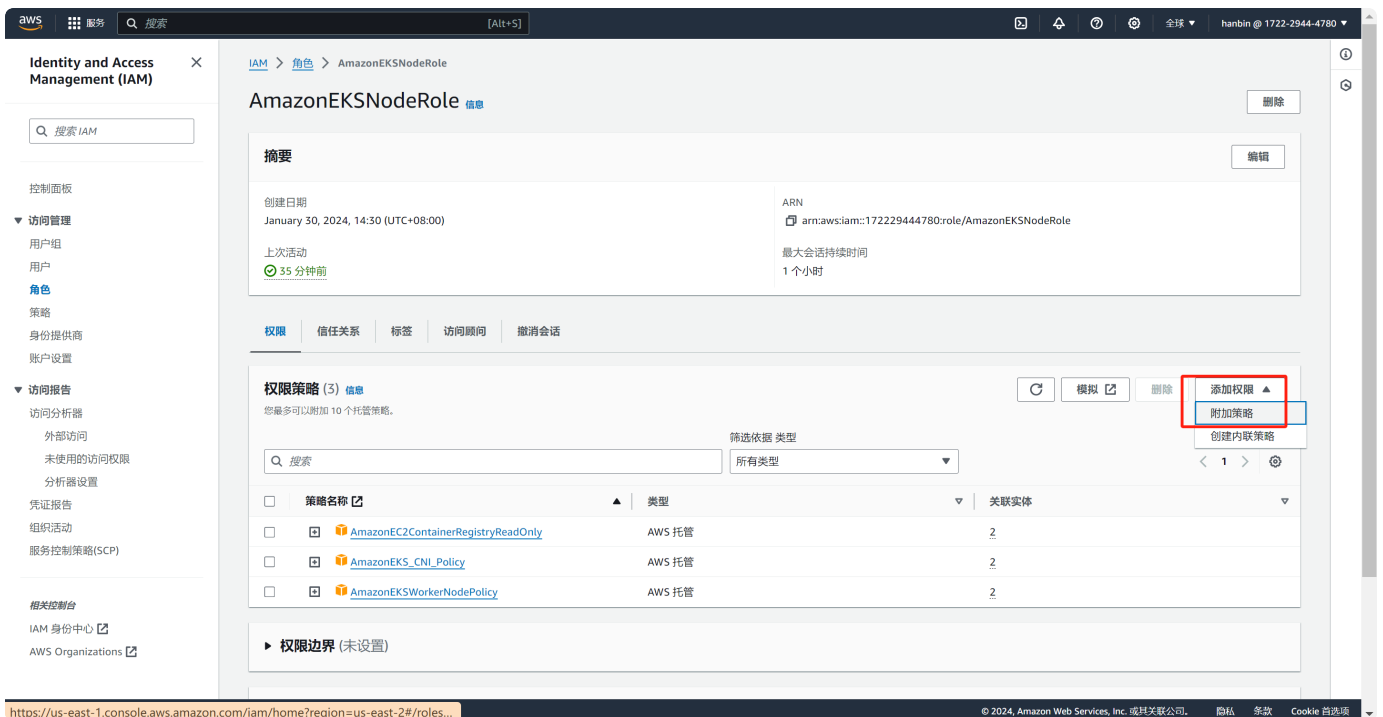


如果是docker推送镜像的话，直接复制推送命令就能上传或下载镜像，但我们用的是kaniko，所以需要一些其他授权方式

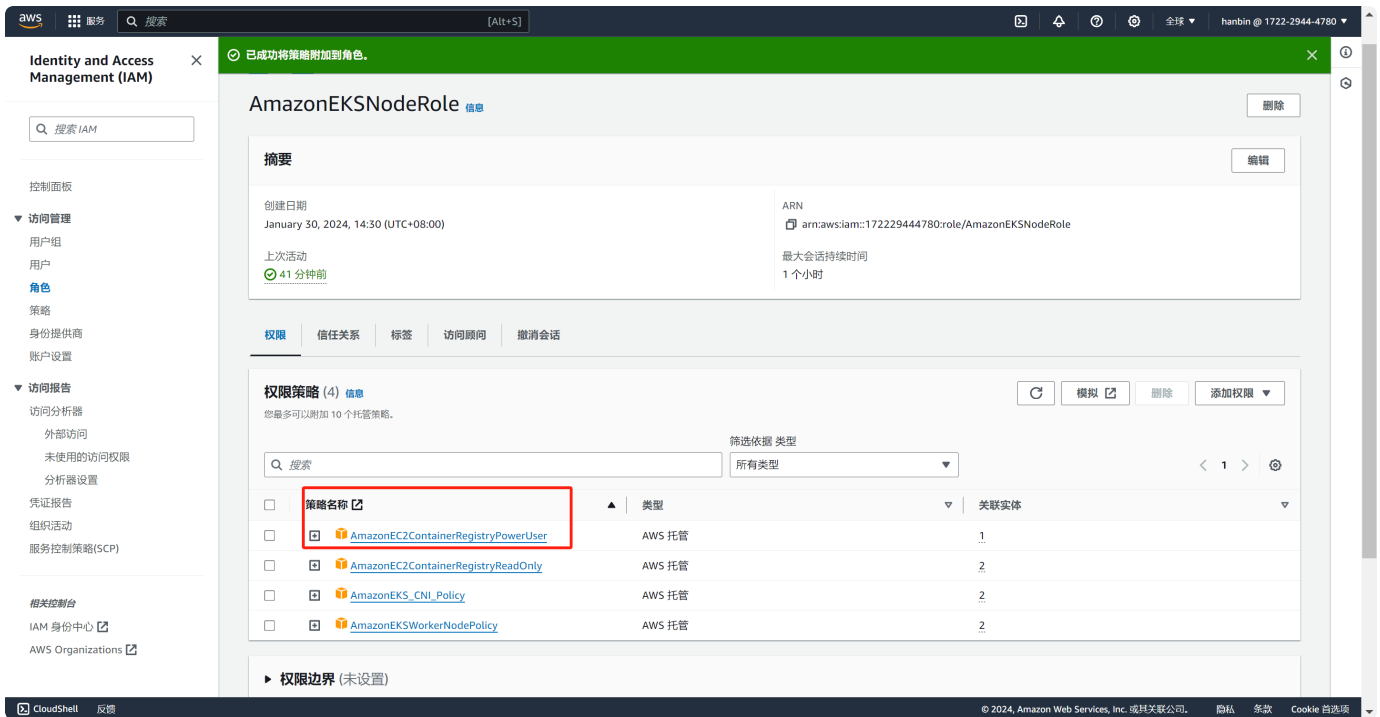


2.2 iam角色添加权限

kaniko使用node上的角色身份获取推送权限，但是我们默认附加的策略里只有get image 没有推送的权限，所以还需要添加下新的策略



添加策略里搜索下ECR的托管策略然后添加AmazonEC2ContainerRegistryPowerUser托管策略



3 配置jenkins

3.1 安装的插件

需确保下列插件全部都已安装在jenkins中

▼ Plain Text |

- 1 Kubernetes plugin
- 2 Kubernetes CLI Plugin
- 3 Pipeline
- 4 GitHub plugin
- 5

3.2 配置凭证

jenkins面板，点击系统管理，系统配置，添加GitHub服务器

GitHub

GitHub 服务器 ?

GitHub Server ?

名称 ?
adp-github-dev

API URL ?
https://api.github.com

凭据 ?
adp-git-dev
+ 添加

连接测试


☐ 管理 Hook

高级

从github中获取到repo的访问凭证，记录token值

ltvujingji

Overview Repositories 5 Projects Packages Stars 1



ltvujingji

ltvujingji

Edit profile

0 followers · 1 following

ltvujingji / README.md

Hi, I'm ltvujingji 本项目 用于CICD部署测试,详细参考CICD说明文档

Popular repositories

ltvujingji

Config files for my GitHub profile.

Public

Archery

Forked from hhyo/Archery

SQL 审核查询平台

Python

awesome-prometheus-alerts

Forked from samber/awesome-prometheus-alerts

Collection of Prometheus alerting rules

HTML

kubernetes-prometheus

Deploy prometheus on kubernetes

eks-CICD

Public

29 contributions in the last year

Contribution settings

Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan

Mon

Link to social profile

Company

You can @mention your company's GitHub organization to link it.

Location

☐ Display current local time

Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

Update profile

Security

Code security and analysis

Integrations

Applications

Scheduled reminders

Archives

Security log

Sponsorship log

Developer settings

ltvujingji

Set status

Your profile

Add account

Your repositories

Your projects

Your organizations

Your enterprises

Your stars

Your sponsors

Your gists

Upgrade

Try Enterprise

Copilot

Feature preview

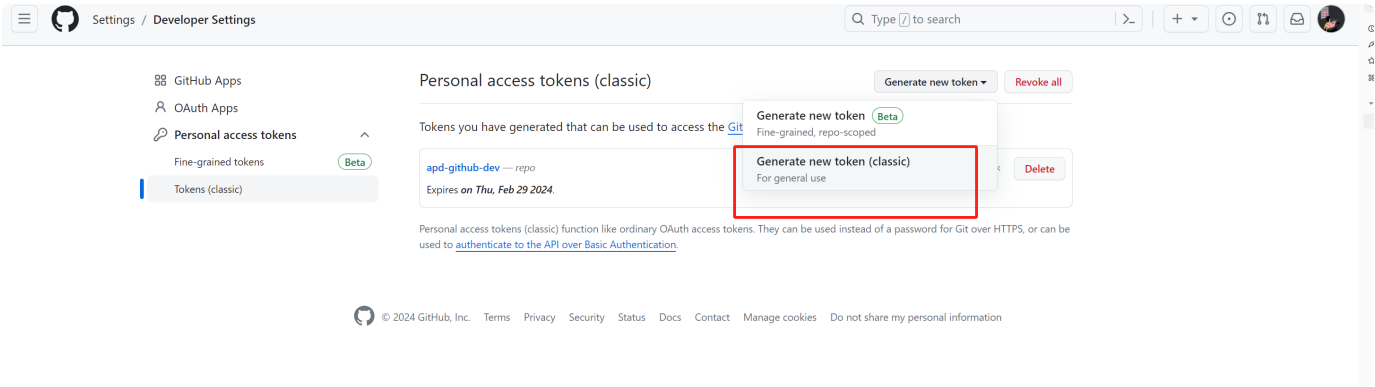
Settings

GitHub Docs

GitHub Support

Sign out

15



Personal access tokens (classic)

What's this token for?

Expiration *
30 days The token will expire on Thu, Feb 29 2024

Select scopes
Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry

将token添加到jenkins凭证列表中

凭据

类型	提供者	存储 ↓	域	唯一标识	名称
	System	System	全局	adp-git-dev	adp-git-dev
	System	System	全局	adp-git-pass-dev	lvtujingji/*****
	System	System	全局	admin	admin
	System	System	全局	admin-adp-prod	admin-adp-prod

Stores scoped to Jenkins

提供者	存储 ↓	域
	System	全局

图标: 小 中 大

3.3 添加cloud

Jenkins

Dashboard > 系统管理

Manage Jenkins

系统配置

- 系统配置: 配置全局设置和路径
- 全局工具配置: 工具配置, 包括它们的位置和自动安装器
- 插件管理: 添加、删除、禁用或启用Jenkins功能扩展插件。
- 节点和云管理: 添加、删除、控制和监视系统运行任务的节点。
- Clouds: Add, remove, and configure cloud instances to provision agents on-demand.

安全

- 全局安全配置: Jenkins 安全, 定义谁可以访问或使用系统。
- 凭据管理: 配置凭据
- 凭据提供者配置: 配置凭据的提供者 and 类型
- 管理用户: 创建、删除或修改 Jenkins 用户
- In-process Script Approval: Allow a Jenkins administrator to write

新建任务

用户列表

构建历史

项目关系

检查文件指纹

系统管理

我的视图

构建队列

队列中没有构建任务

构建执行状态

1 空闲

2 空闲

需要调整的地方不多，配置下jenkins的地址就行，如果master jenkins是在eks中启动

Jenkins

Dashboard > 系统管理 > Clouds > New cloud

New cloud

名称 ?

adp-k8s-dev

Kubernetes Cloud details ^ Edited

Kubernetes 地址 ?

kubernetes.default.svc

☐ Use Jenkins Proxy ?

Kubernetes 服务证书 key ?

☐ 禁用 HTTPS 证书检查 ?

Save

Dashboard > 系统管理 > Clouds > adp-k8s-dev > Configure

Kubernetes 命名空间 ?

devops-tools

Agent Docker Registry ?

☐ Inject restricted PSS security context in agent container definition ?

凭据

- 无 -

+ 添加

连接测试

☐ WebSocket ?

☐ Direct Connection ?

Jenkins 地址 ?

http://10.100.67.77:8080

Jenkins 通道 ?

Save 应用

需要提前在devops-tools名称空间创建ecrconfig configmap

Plain Text

```
1 ---
2 apiVersion: v1
3 kind: ConfigMap
4 metadata:
5   name: ecrconfig
6   namespace: devops-tools
7 data:
8   config.json: |
9     { "credsStore": "ecr-login"}
10
```

3.4 创建测试环境

需要提前创建名称空间，需要在devops-tools空间下创建kaniko的configmap

```
1 kubectl create namespace adp-dev
2 kubectl create namespace adp-test
3 kubectl create namespace adp-prod
4
5 cat > kaniko-configmap.yaml << EOF
6 ---
7 apiVersion: v1
8 kind: ConfigMap
9 metadata:
10   name: ecrconfig
11   namespace: devops-tools
12 data:
13   config.json: |
14     { "credsStore": "ecr-login"}
15 EOF
16 kubectl apply -f kaniko-configmap.yaml
17
```

拉取测试代码,并将其放在你自己的git仓库

```
1 git clone https://github.com/lvtujiangji/eks-CICD.git
```

3.5 jenkins file

这是一个配套的jenkinsfile 你只需要在创建pipeline的时候复制pipeline里的内容然后修改下列部分

<https://github.com/lvtujiangji/lvtujiangji.git> 替换成你github代码托管的地址

```
sh "/kaniko/executor --context git://github.com/lvtujiangji/lvtujiangji.git#refs/heads/main --
dockerfile dockerfile --destination 547384405015.dkr.ecr.us-east-1.amazonaws.com/adp-ecr-
dev:nginx-v${params.VERSION}" 替换拉取代码的地址, 和--destination推送的ecr地址
```

pipeline分为4个步骤

parameter中包含了俩个参数

ENVIRONMENT VERSION 分别代表了哪些环境DEV还是PROD 和 要启动构建image的版本

1 定义agent模板, 在模板中启动kaniko container

2 github中拉取代码

3 通过kaniko 构建镜像并推送

4 Deploy 根据你运行任务的参数选择部署到哪些目标环境和部署的版本

```
1  pipeline{
2    agent{
3      kubernetes{
4        cloud 'adp-k8s-dev'
5        yaml '''
6  apiVersion: v1
7  kind: Pod
8  metadata:
9    name: kaniko
10   namespace: devops-tools
11  spec:
12    serviceAccountName: jenkins-admin
13    containers:
14      - name: kaniko
15        image: gcr.io/kaniko-project/executor:debug
16        env:
17          - name: AWS_SDK_LOAD_CONFIG
18            value: "true"
19        command:
20          - sleep
21        args:
22          - 99d
23        volumeMounts:
24          - name: ecrconfig
25            mountPath: /kaniko/.docker/
26      restartPolicy: Never
27    volumes:
28      - name: ecrconfig
29        configMap:
30          name: ecrconfig
31    '''
32      }
33    }
34    parameters{
35      string( name:'ENVIRONMENT',defaultValue:'dev',description:'Target environment (dev, test, prod)')
36      string( name:'VERSION',defaultValue:'1.10',description:'Target Version to deploy')
37    }
38    stages('Begging Deploy'){
39      stage('Pull Code'){
40        steps {
41          // 使用 checkout 步骤拉取代码
42          checkout([$class: 'GitSCM',
```

```

43         branches: [[name: 'main']],
44         doGenerateSubmoduleConfigurations: false,
45         extensions: [],
46         submoduleCfg: [],
47         userRemoteConfigs: [[credentialsId: 'adp-git-pas
s-dev', url: 'https://github.com/lvtujingji/lvtujingji.git']]])
48         sh 'sed -i -E "/server_name/s/web/${ENVIRONMENT}/" lvtujin
gji.conf'
49     }
50 }
51 stage('Build Image'){
52     steps{
53         container('kaniko'){
54             sh "/kaniko/executor --context git://github.com/lvtujingj
i/lvtujingji.git#refs/heads/main --dockerfile dockerfile --destination 547
384405015.dkr.ecr.us-east-1.amazonaws.com/adp-ecr-dev:nginx-v${params.VERS
ION}"
55         }
56     }
57 }
58 stage('Deploy !!!'){
59     steps{
60         script{
61             sh 'curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.co
m/1.18.9/2020-11-02/bin/linux/amd64/kubectl && chmod +x kubectl'
62             if (params.ENVIRONMENT == 'prod') {
63                 sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSIO
N}/g" nginx-prod.yaml'
64                 sh "./kubectl apply -f nginx-prod.yaml"
65             } else if (params.ENVIRONMENT == 'test') {
66                 sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSIO
N}/g" nginx-test.yaml'
67                 sh "./kubectl apply -f nginx-test.yaml"
68             } else {
69                 sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSIO
N}/g" nginx-dev.yaml'
70                 sh "./kubectl apply -f nginx-dev.yaml"
71             }
72         }
73     }
74 }
75 }
76 }

```

结尾

如果一切顺利的话，可以在eks集群中，查看到deployment已经正常运行，可以通过task里的运行日记，查看是否有任何执行报错，关于pipeline的部分可以参考jenkins的官方文档进行修改，如果在部署过程中有其他问题，也可以邮箱联系作者 lvtingji@163.com