eks 部署CICD

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

```
# cli  install
curl https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip -o awscliv2.zip
unzip awscliv2.zip
./aws/install
mv /usr/local/bin/aws /bin/aws
aws --version

# kubectl install
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
chmod +x kubectl
mv kubectl /usr/bin/
```

创建master节点需要使用的权限角色

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role --role-name AmazonEKSClusterRole --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy --role-name AmazonEKSClusterRole
```

创建master节点

```
#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
#aws ec2 describe-vpcs --query "Vpcs[?InstanceTenancy=='default'].VpcId" --output text
#aws ec2 describe-subnets --query "Subnets[?VpcId=='vpc-0068f2f40191aace5'].[SubnetId,AvailabilityZone]"

# 创建集群的cli命令，可以调整name区域，和版本，以及role的arn
aws eks create-cluster --region us-east-2 --name adp-k8s-prod --kubernetes-version 1.28 \
   --role-arn arn:aws:iam::547384405015:role/AmazonEKSClusterRole \
   --resources-vpc-config subnetIds=subnet-01e82b74dc8d09b8c,subnet-0e2bffaba035bfd5f
# 创建必选插件
aws eks create-addon --cluster-name adp-k8s-prod --addon-name coredns
aws eks create-addon --cluster-name adp-k8s-prod --addon-name kube-proxy
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

```
#node
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSNodeRole
```

创建node节点组

```
aws eks create-nodegroup --cluster-name adp-k8s-dev  --nodegroup-name adp-nodegroup-dev --node-role arn:aws:iam::172229444780:role/AmazonEKSNodeRole \
--subnets subnet-0969dd09119cbc764,subnet-0a5c551ab0c0d47cf --instance-types t3.medium --disk-size 20 --scaling-config minSize=1,maxSize=1,desiredSize=1
```

创建ebs 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

Plain Text

```
# EBS-driver

aws eks describe-cluster --name adp-k8s-prod --query "cluster.identity.oidc.issuer" --output text
cat > aws-ebs-csi-driver-trust-policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::547384405015:oidc-provider/oidc.eks.us-east-1.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/id/1B287476D5B0BA8D494474280FD01B0F:sub": "system:serviceaccount:kube-system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}
EOF
# 替换region-code  和账号id 和OIDC码

aws iam create-role \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-ebs-csi-driver-trust-policy.json"

aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name AmazonEKS_EBS_CSI_DriverRole

aws eks create-addon --cluster-name adp-k8s-dev --addon-name aws-ebs-csi-driver \
  --service-account-role-arn arn:aws:iam::172229444780:role/AmazonEKS_EBS_CSI_DriverRole

```

```
#部署示例应用
git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git
cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
echo "parameters:
  type: gp3" >> manifests/storageclass.yaml
kubectl apply -f manifests/
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。

```
kubectl apply -f https://github.com/kubesphere/ks-installer/releases/download/v3.4.1/kubesphere-installer.yaml



kubectl apply -f https://github.com/kubesphere/ks-installer/releases/download/v3.4.1/cluster-configuration.yaml



# ingress支持websocket
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubesphere-ingress
  namespace: kubesphere-system
  annotations:
    nginx.ingress.kubernetes.io/proxy-body-size: 600m
    nginx.org/client-max-body-size: "10m"
    nginx.ingress.kubernetes.io/proxy-read-timeout: "1800"
    nginx.ingress.kubernetes.io/proxy-send-timeout: "1800"
    nginx.ingress.kubernetes.io/websocket-services: proxy-public
    nginx.org/websocket-services: proxy-public
spec:
  rules:
  - host: ks.lvtujingji.click
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: ks-console
            port:
              number: 80
  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

```
# ingress nginx

wget https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.2/deploy/static/provider/aws/nlb-with-tls-termination/deploy.yaml
aws acm request-certificate --domain-name *.lvtujingji.click --validation-method DNS
aws acm describe-certificate --certificate-arn arn:aws:acm:us-east-1:547384405015:certificate/6843d443-8907-4dee-99b5-ad8f45c27105

aws route53 list-hosted-zones

aws route53 change-resource-record-sets --hosted-zone-id Z03012041JX1FRPDJQ2NX --change-batch file://config.json
cat >> config.json << EOF
{
  "Comment": "optional comment about the changes in this change batch request",
  "Changes": [
    {
      "Action": "UPSERT",
      "ResourceRecordSet": {
        "Name": "ks.lvtujingji.click.",
        "Type": "CNAME",
        "TTL": 60,
        "ResourceRecords": [
          {
            "Value": ""
          }
        ]
      }
    }
  ]
}
EOF

需要修改 proxy-real-ip-cidr
        service.beta.kubernetes.io/aws-load-balancer-ssl-cert
kubectl  apply -f deploy.yaml

```

1.3 jenkins install

Plain Text

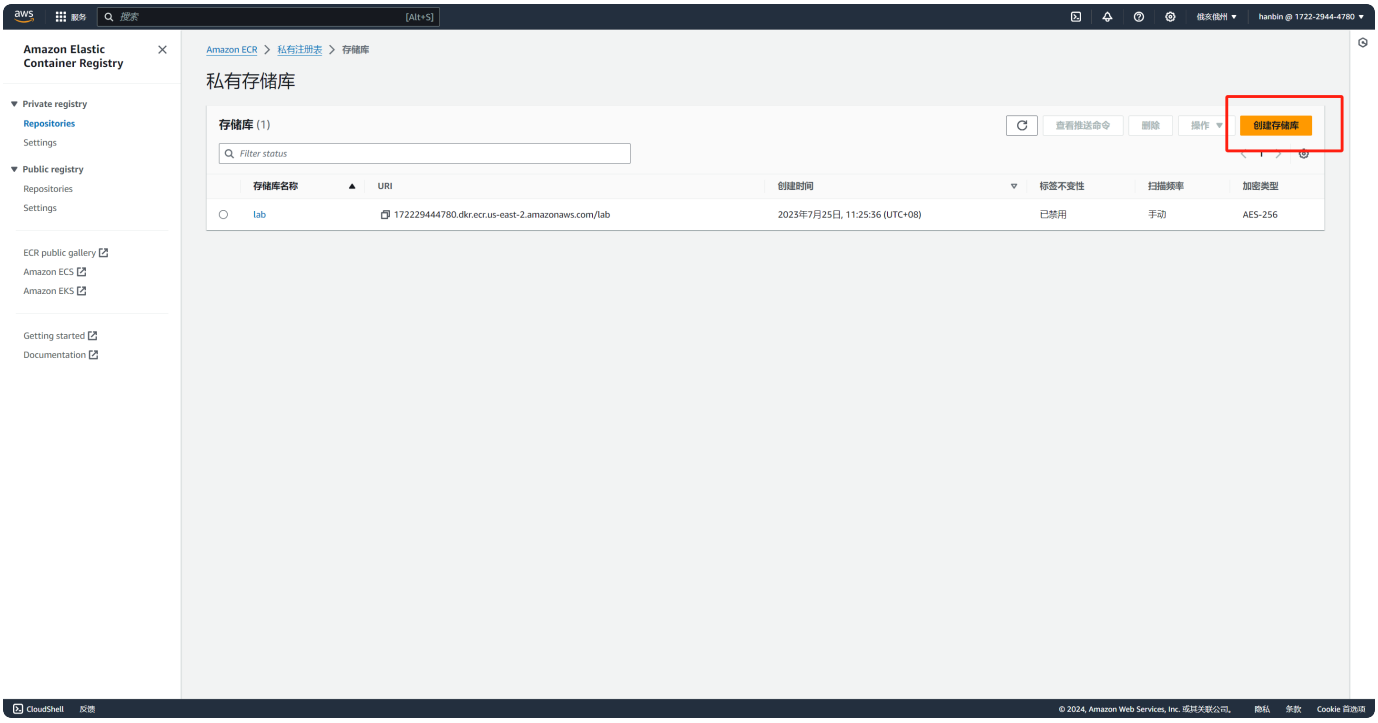
```
git clone https://github.com/scriptcamp/kubernetes-jenkins
kubectl create namespace devops-tools
kubectl apply -f serviceAccount.yaml

# 如果安装了ebs CSI就替换local-storage为 ebs-sc
cat > volume.yaml << EOF
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: jenkins-pv-claim
  namespace: devops-tools
spec:
  storageClassName: local-storage
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 30Gi
EOF

kubectl create -f volume.yaml
kubectl apply -f deployment.yaml


cat >> service.yaml << EOF
apiVersion: v1
kind: Service
metadata:
  name: jenkins-service
  namespace: devops-tools
  annotations:
      prometheus.io/scrape: 'true'
      prometheus.io/path:   /
      prometheus.io/port:   '8080'
spec:
  selector:
    app: jenkins-server
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: 8080
      name: httpport
    - port: 50000
```

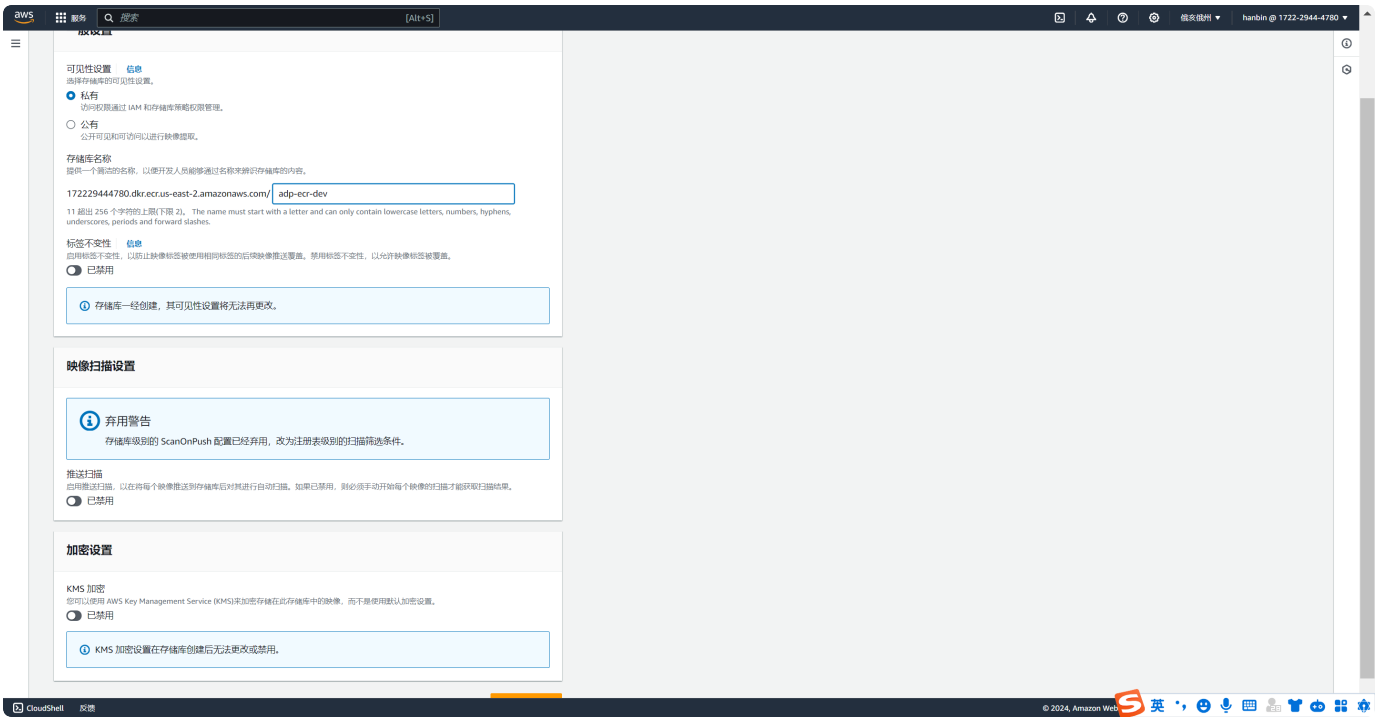
```
      targetPort: 50000
      name: jnlpport
EOF

# 可选
cat >> secret.yaml << EOF
---
apiVersion: v1
kind: Secret
metadata:
  name: jenkins-admin
  namespace: devops-tools
  annotations:
    kubernetes.io/service-account.name: "jenkins-admin"
type: kubernetes.io/service-account-token
EOF

cat > jenkins-ingress.yaml <<EOF
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: jenkins-ingress
  namespace: devops-tools
spec:
  rules:
  - host: js.lvtujingji.click
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: jenkins-service
            port:
              number: 8080
  ingressClassName: nginx
EOF


kubectl apply -f  jenkins-ingress.yaml
```

2 配置ECR权限

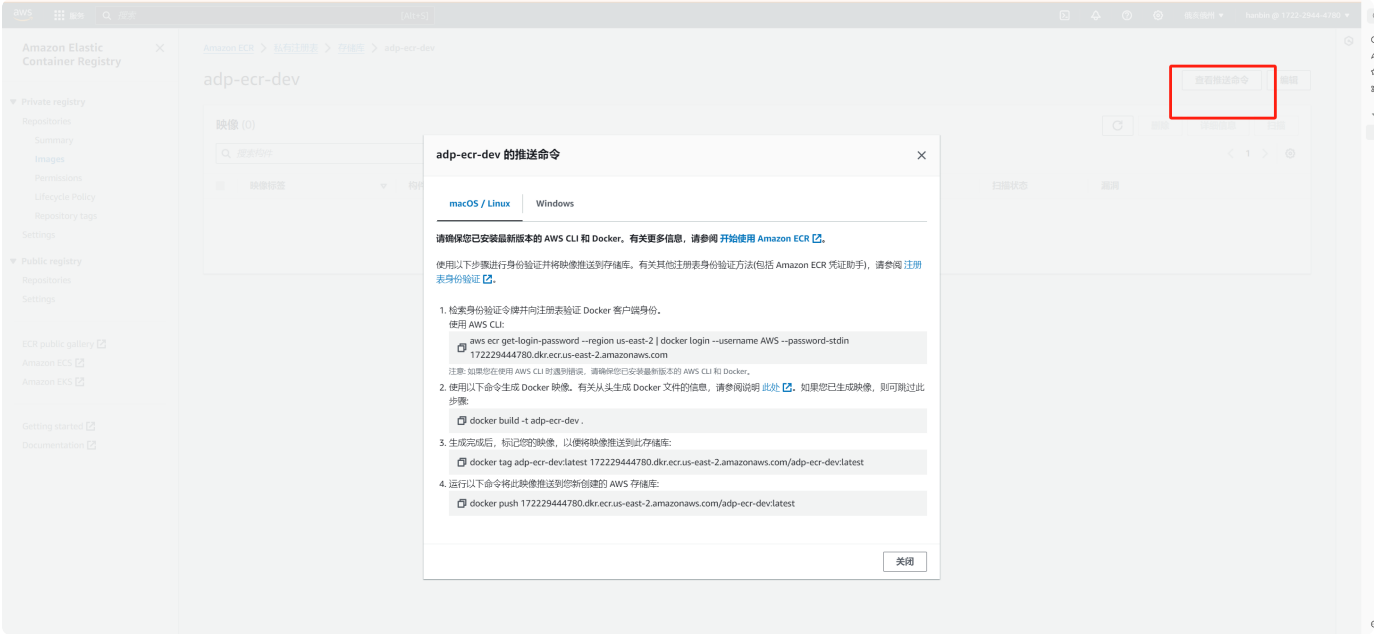
我们使用AWS ECRprivate仓库作为镜像载体，所以需要提前创建私有仓库和进行IAM授权

2.1 创建私有仓库



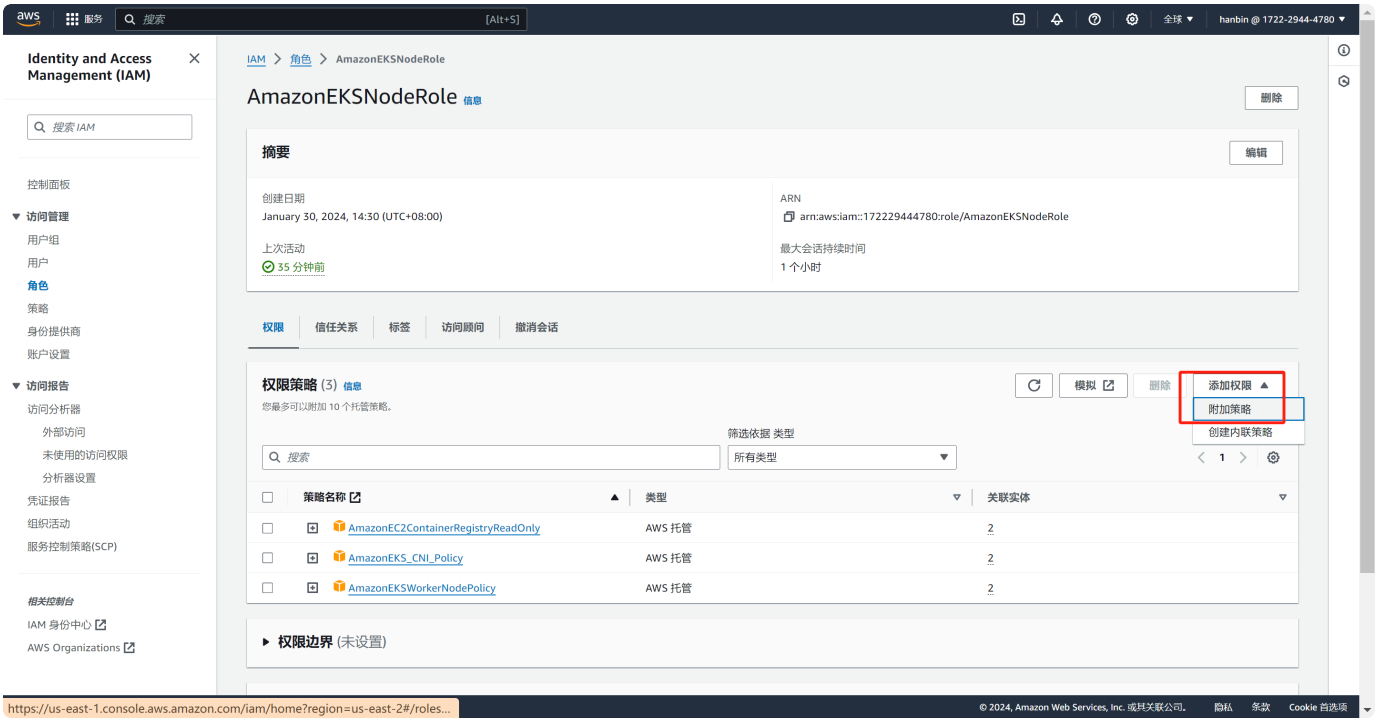


如果是docker推送镜像的话，直接复制推送命令就能上传或下载镜像，但我们用的是kaniko，所以需要一些其他授权方式

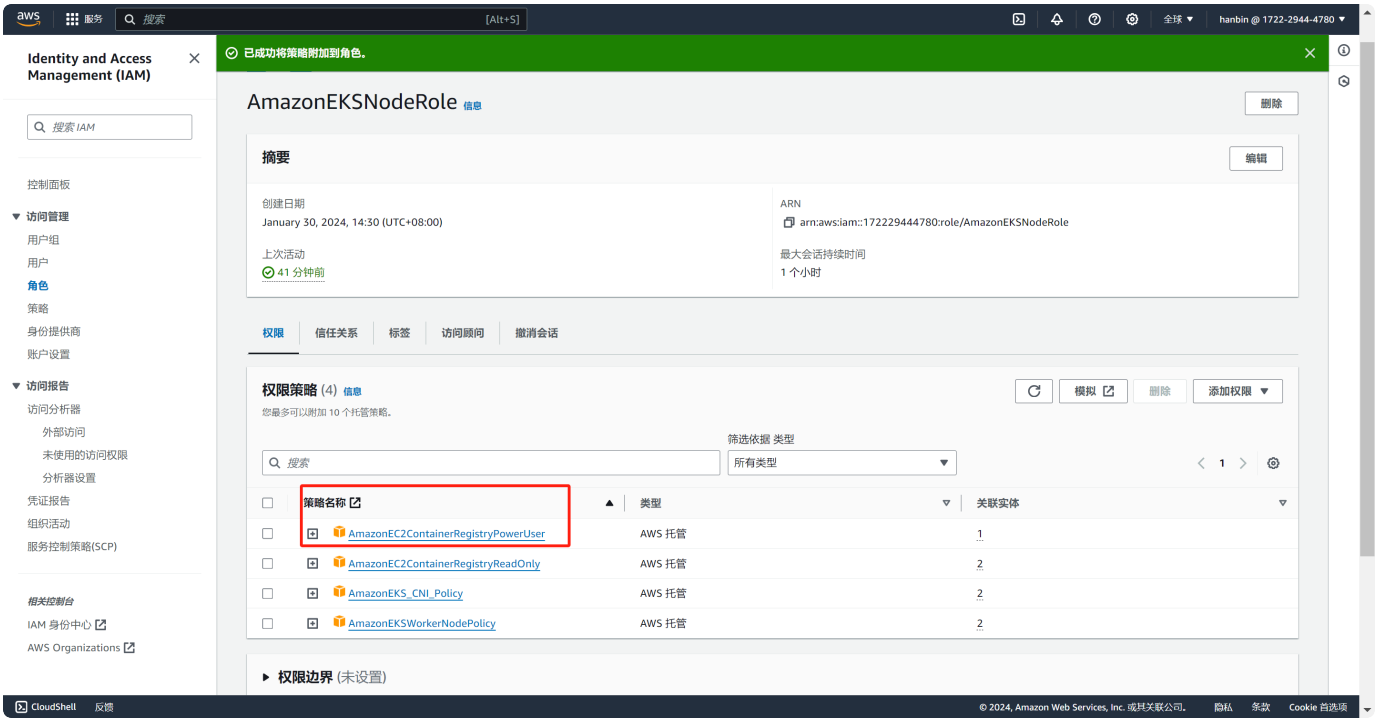


2.2 iam角色添加权限

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

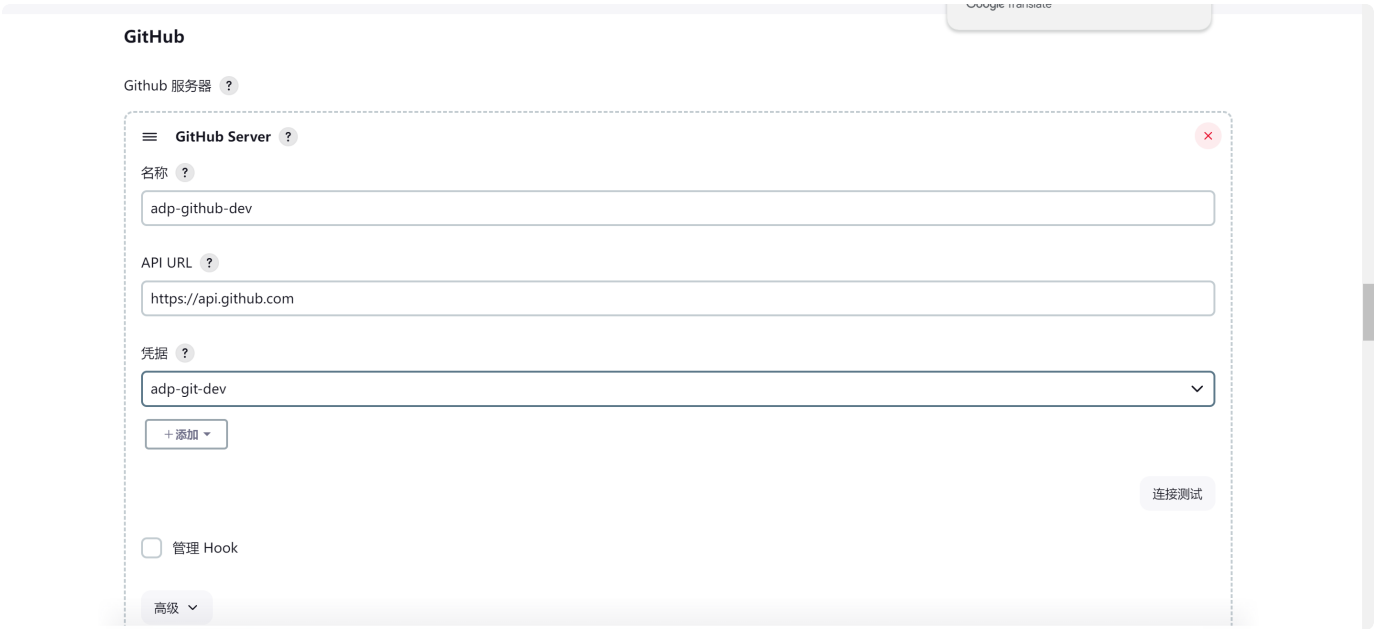
3.1 安装的插件

需确保下列插件全部都已安装在jenkins中

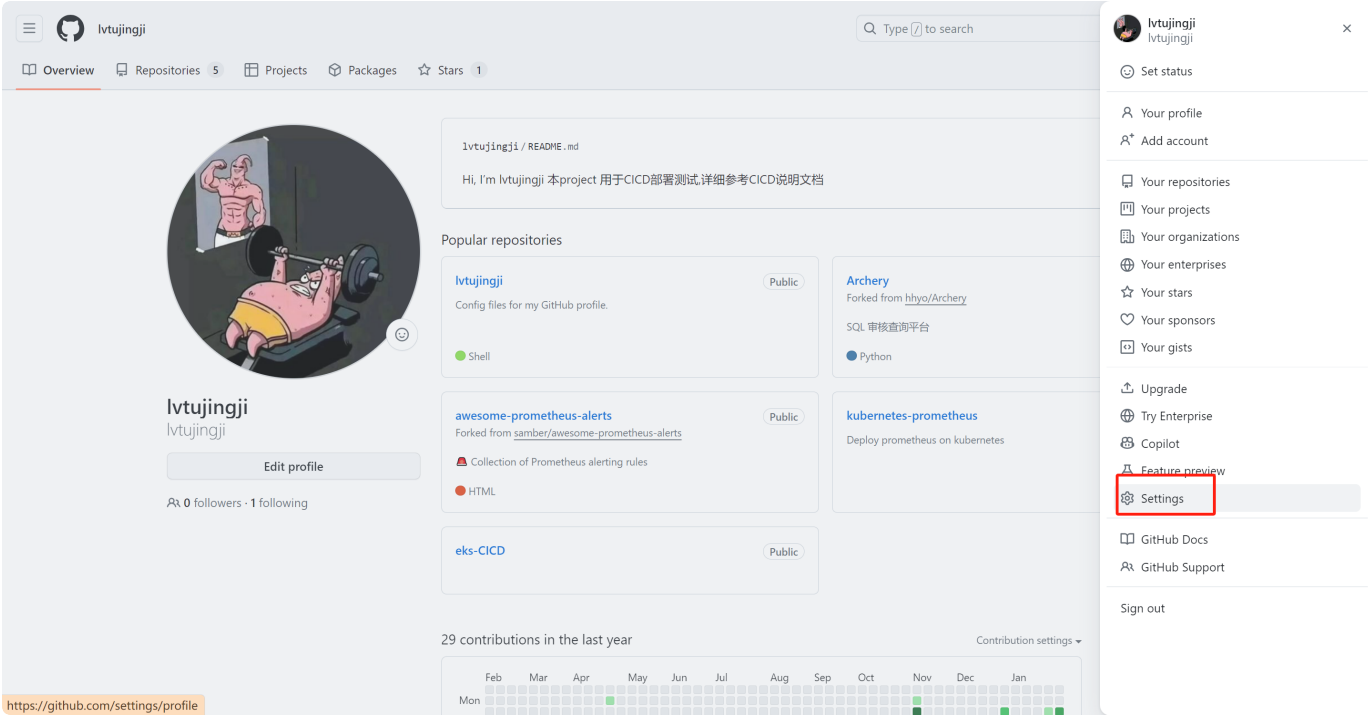
```
Kubernetes plugin
Kubernetes CLI Plugin
Pipeline
GitHub plugin

```

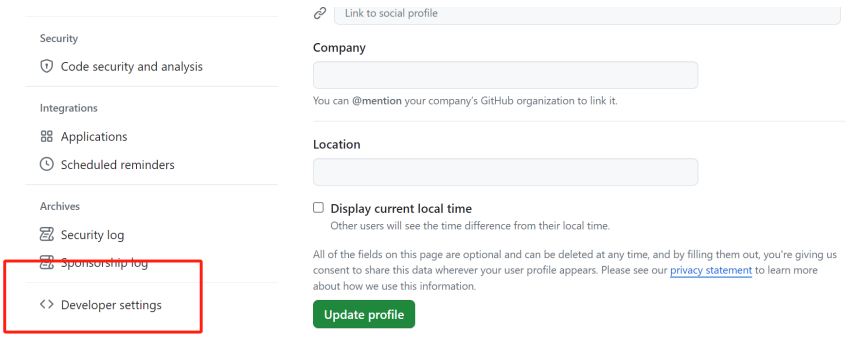
3.2 配置凭证

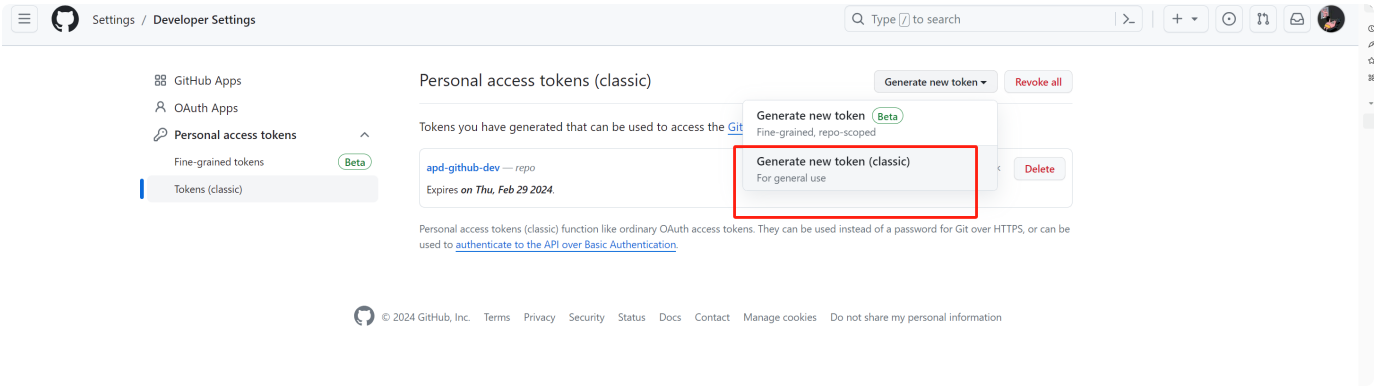
jenkins面板，点击系统管理，系统配置，添加GitHub服务器

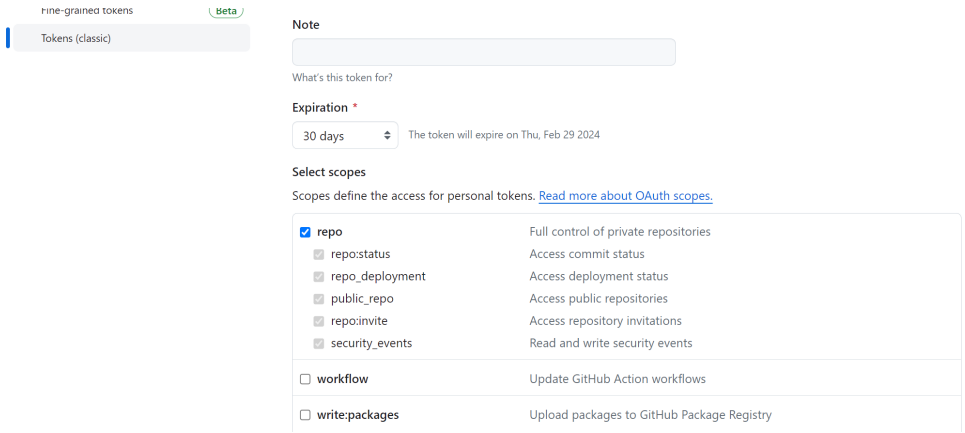


从github中获取到repo的访问凭证，记录token值

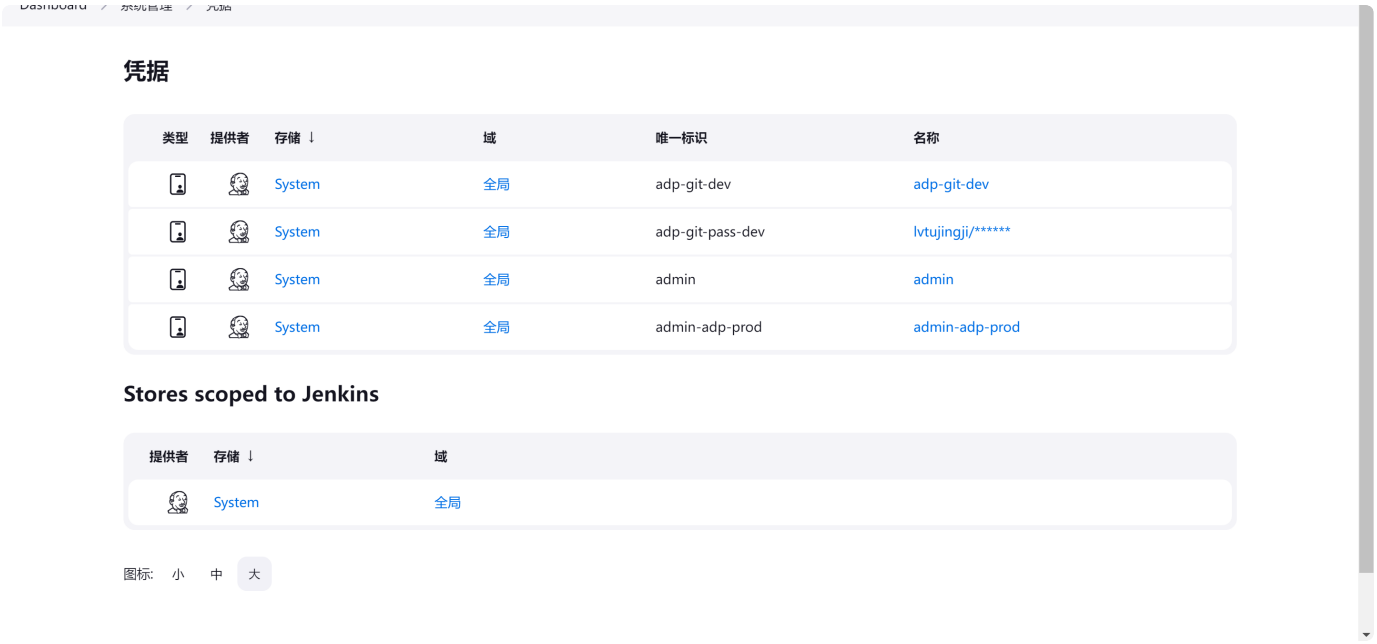




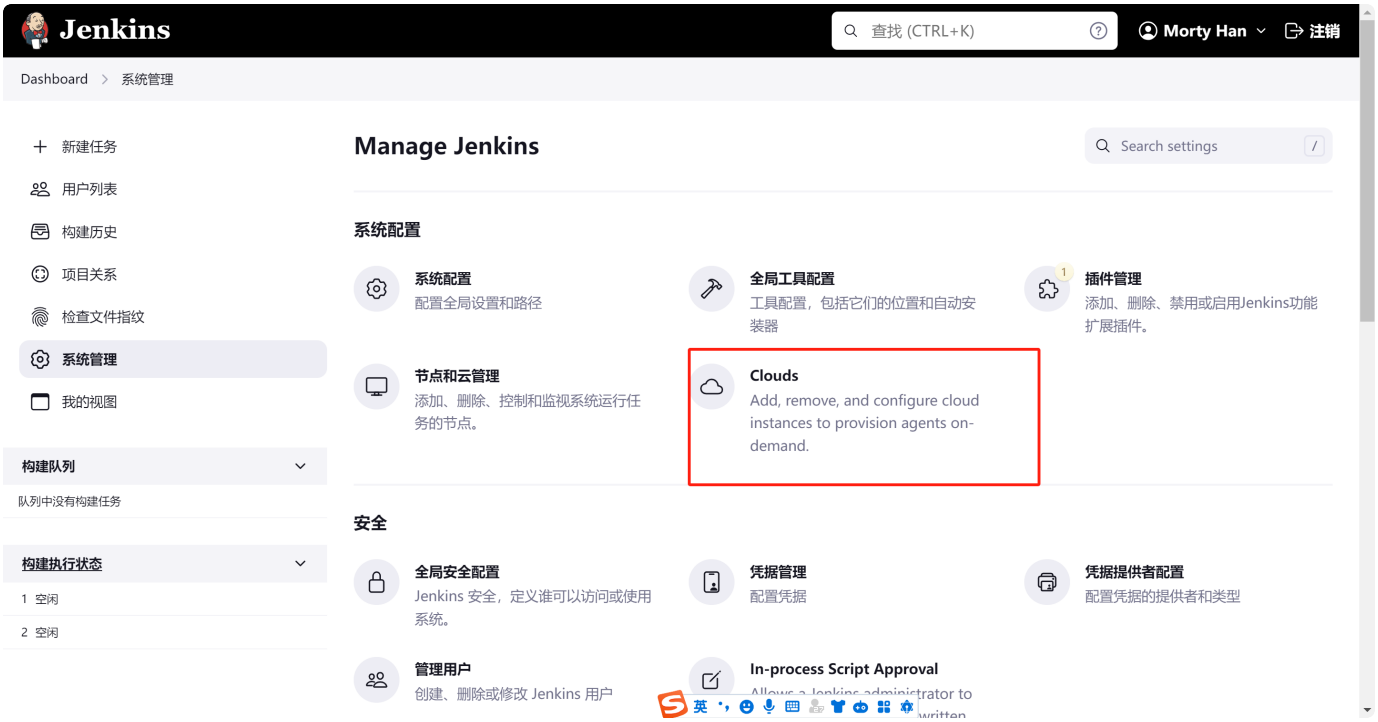




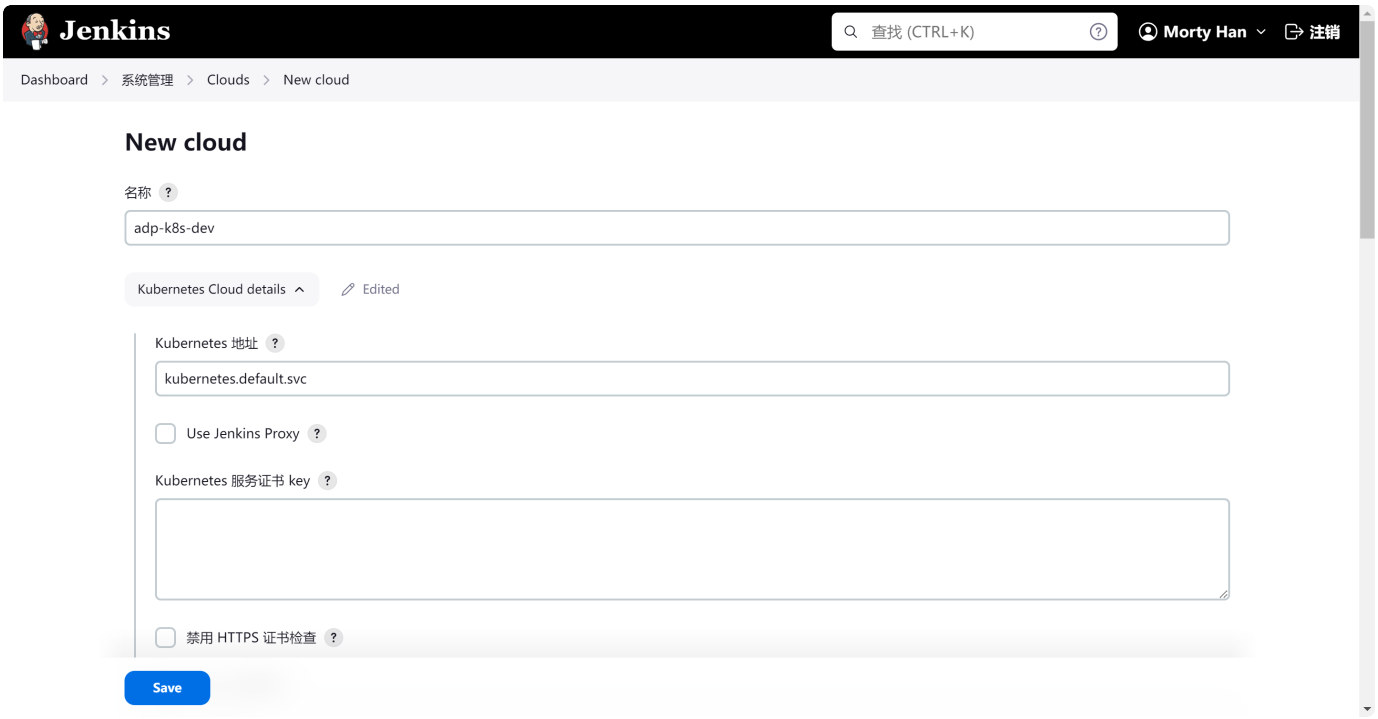
将token添加到jenkins凭证列表中

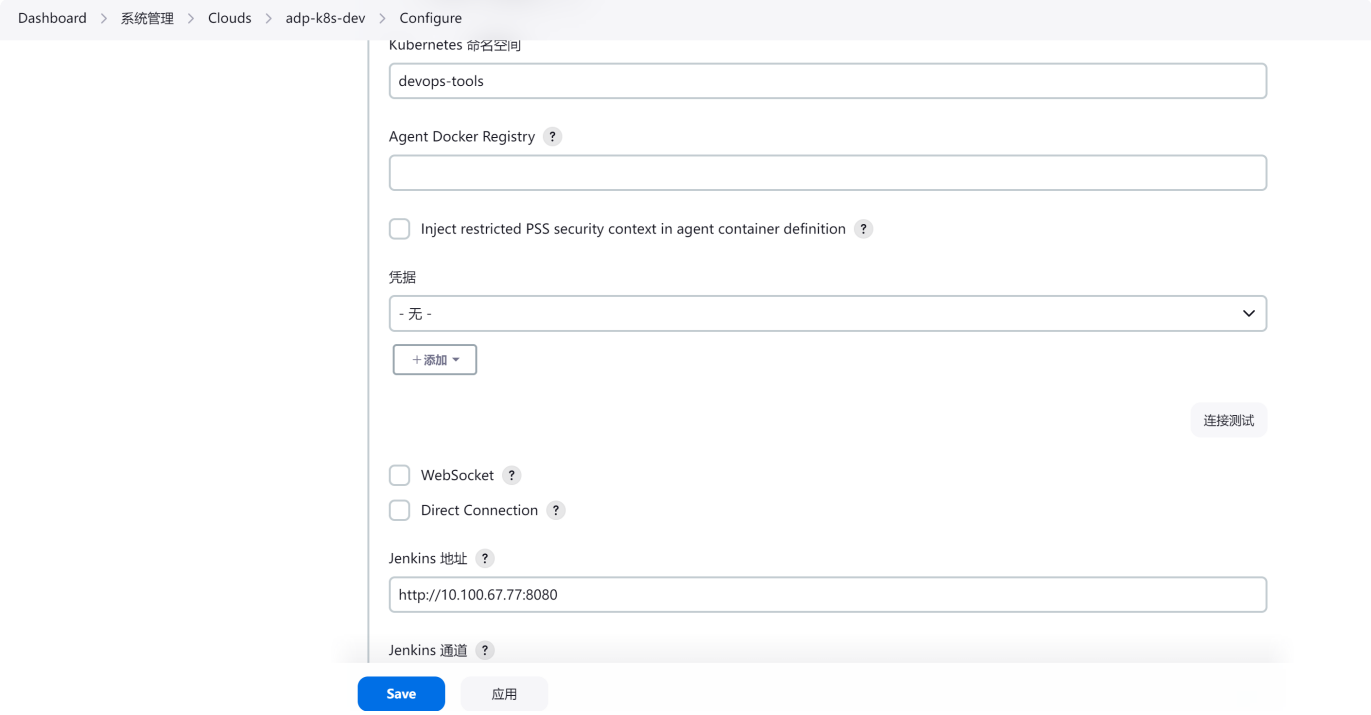


3.3 添加cloud



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





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

```Plain Text
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ecrconfig
  namespace: devops-tools
data:
  config.json: |
    { "credsStore": "ecr-login"}

```

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

```
kubectl create namespace adp-dev
kubectl create namespace adp-test
kubectl create namespace adp-prod

cat > kaniko-configmap.yaml << EOF
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ecrconfig
  namespace: devops-tools
data:
  config.json: |
    { "credsStore": "ecr-login"}
EOF
kubectl apply -f kaniko-configmap.yaml

```

拉取测试代码,并将其放在你自己的git仓库

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

3.5 jenkins file

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

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

sh "/kaniko/executor --context git://github.com/lvtujingji/lvtujingji.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

pipline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

Plain Text

```
pipeline{
  agent{
    kubernetes{
      cloud 'adp-k8s-dev'
      yaml '''
apiVersion: v1
kind: Pod
metadata:
  name: kaniko
  namespace: devops-tools
spec:
  serviceAccountName: jenkins-admin
  containers:
    - name: kaniko
      image: gcr.io/kaniko-project/executor:debug
      env:
      - name: AWS_SDK_LOAD_CONFIG
        value: "true"
      command:
        - sleep
      args:
        - 99d
      volumeMounts:
      - name: ecrconfig
        mountPath: /kaniko/.docker/
  restartPolicy: Never
  volumes:
  - name: ecrconfig
    configMap:
      name: ecrconfig
'''
    }
  }
  parameters{
    string( name:'ENVIRONMENT',defaultValue:'dev',description:'Target environment (dev, test, prod)')
    string( name:'VERSION',defaultValue:'1.10',description:'Target Verison to deply')
  }
  stages('Begging Deply'){
    stage('Pull Code'){
      steps {
                // 使用 checkout 步骤拉取代码
                checkout([$class: 'GitSCM',
```

```
                            branches: [[name: 'main']],
                            doGenerateSubmoduleConfigurations: false,
                            extensions: [],
                            submoduleCfg: [],
                            userRemoteConfigs: [[credentialsId: 'adp-git-pass-dev', url: 'https://github.com/lvtujingji/lvtujingji.git']]])
                  sh 'sed -i -E "/server_name/s/web/${ENVIRONMENT}/" lvtujingji.conf'
        }
      }
      stage('Build Image'){
        steps{
          container('kaniko'){
                   sh "/kaniko/executor --context git://github.com/lvtujingji/lvtujingji.git#refs/heads/main --dockerfile dockerfile --destination 547384405015.dkr.ecr.us-east-1.amazonaws.com/adp-ecr-dev:nginx-v${params.VERSION}"
          }
        }
      }
      stage('Deploy !!!'){
        steps{
          script{
            sh 'curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl && chmod +x kubectl'
            if (params.ENVIRONMENT == 'prod') {
              sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSION}/g" nginx-prod.yaml'
                          sh "./kubectl apply -f nginx-prod.yaml"
                      } else if (params.ENVIRONMENT  == 'test') {
              sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSION}/g" nginx-test.yaml'
                          sh "./kubectl apply -f nginx-test.yaml"
                      } else {
              sh 'sed -i -E "/image/s/nginx-v[0-9]?\\.[0-9]+/nginx-v${VERSION}/g" nginx-dev.yaml'
                          sh "./kubectl apply -f nginx-dev.yaml"
                      }
          }
        }
      }
    }
}
```

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pipline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 lvtujingji@163.com