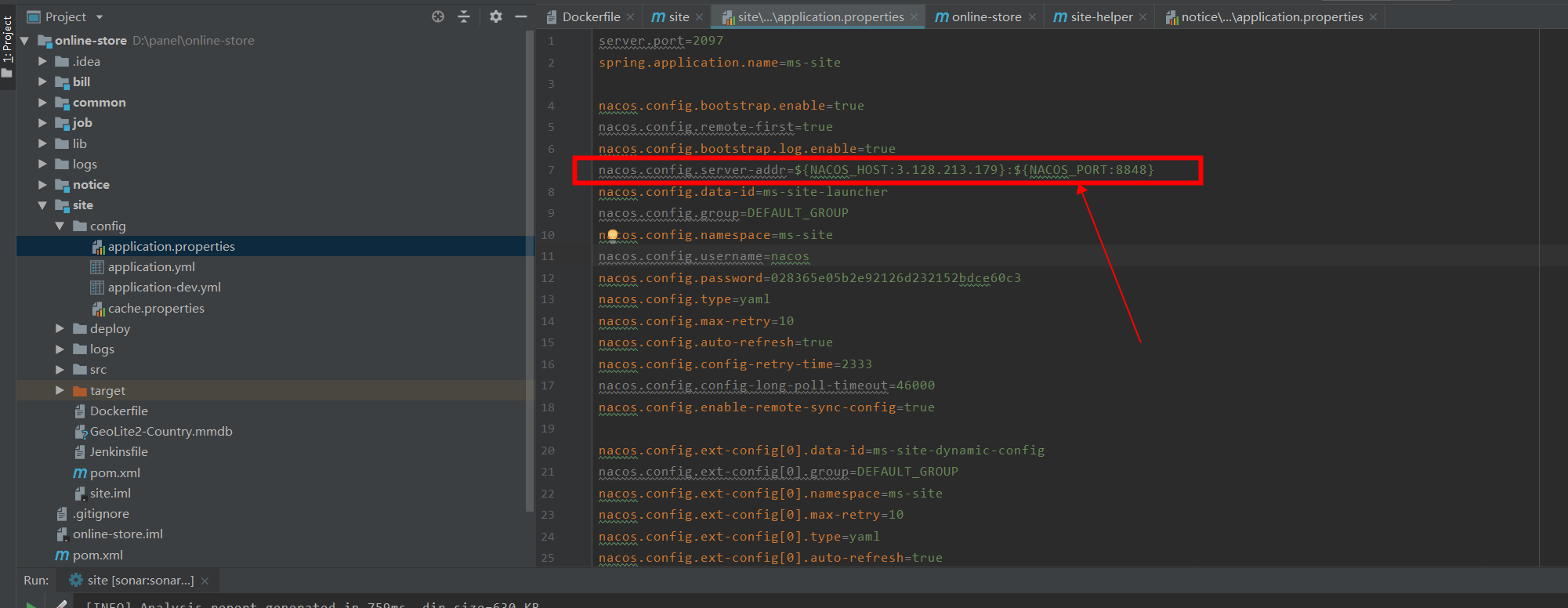
1. 工程改造
2. 业务配置文件从jar包分离，需将环境配置文件打包到jar内
3. 环境的配置需支撑动态替换，如下图中的NACOS\_HOST与NACOS\_PORT



1. 需在各模块工程添加Dockerfile文件。

①后端通用Dockerfile如下：

FROM openjdk:8u302-jre

ENV SERVER\_PORT 2097

WORKDIR /homeCOPY target/\*.jar /home/app.jar

COPY config/application.properties /home/application.propertiesENTRYPOINT ["java","-jar","app.jar","--spring.config.location=./application.properties"]

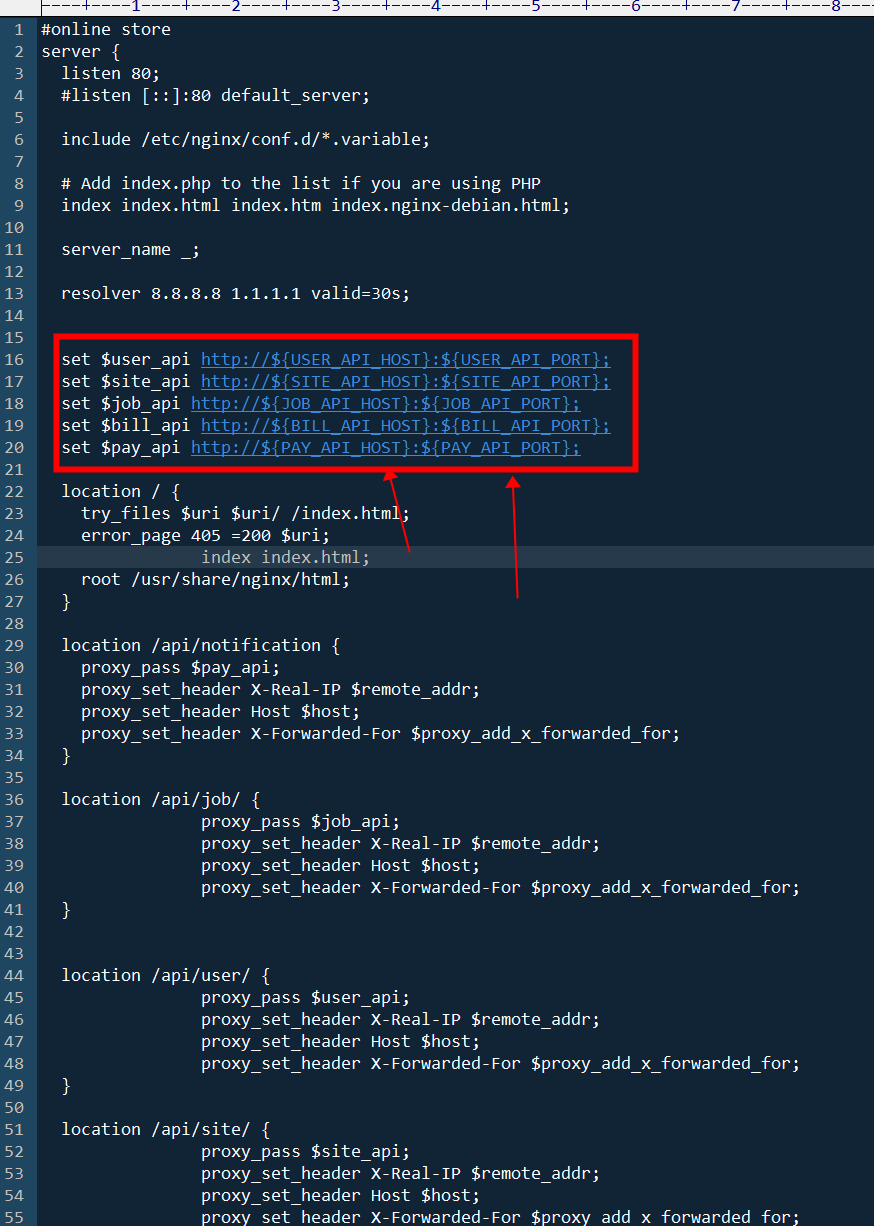
②VUE技术栈通用Dockerfile如下：

FROM nginx:1.21.1

COPY dist/ /usr/share/nginx/html/

COPY default.conf.template /etc/nginx/templates/

其中default.conf.template为现网nginx反向代理文件。其中，${USER\_API\_HOST} 等为环境变量，如：



1. 端口规划

EndUser：30080 ~ 31080

Panel：31080 ~ 32080

Protal：32080 ~ 33080

1. 镜像仓库管理

<http://69.234.235.70:30002/> admin/Valor123

每次CI，会将制品(\*.Jar或 VUE)构建成docke镜像，推送到企业内镜像仓库。

附录：

1. Jenkinsfile 示例

pipeline {

agent {

node {

label 'maven'

}

}

parameters {

string(name:'TAG\_NAME',defaultValue: '',description:'')

}

environment {

HARBOR\_CREDENTIAL\_ID = 'member-site-images'

GITLAB\_CREDENTIAL\_ID = 'member-site-gitlab'

KUBECONFIG\_CREDENTIAL\_ID = 'member-site-kubeconfig'

REGISTRY = '69.234.235.70:30002'

HARBOR\_NAMESPACE = 'member-site'

GITLAB\_ACCOUNT = 'panel'

APP\_NAME = 'ms-site'

SONAR\_CREDENTIAL\_ID = 'sonar-token'

}

stages {

stage ('checkout scm') {

steps {

checkout(scm)

}

}

stage ('unit test') {

steps {

container ('maven') {

dir('site') {

sh "mvn clean test"

}

}

}

}

stage('sonarqube analysis') {

steps {

container ('maven') {

dir('site') {

withCredentials([string(credentialsId: "$SONAR\_CREDENTIAL\_ID", variable: 'SONAR\_TOKEN')]) {

withSonarQubeEnv('sonar') {

sh "mvn clean package sonar:sonar -Dsonar.login=$SONAR\_TOKEN"

}

}

timeout(time: 1, unit: 'HOURS') {

waitForQualityGate abortPipeline: true

}

}

}

}

}

stage ('build & push') {

steps {

container ('maven') {

dir('common') {

sh 'mvn clean install'

}

dir('site') {

sh 'mvn clean package'

sh 'docker build -f Dockerfile -t $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:SNAPSHOT-$BRANCH\_NAME-$BUILD\_NUMBER .'

withCredentials([usernamePassword(passwordVariable : 'DOCKER\_PASSWORD' ,usernameVariable : 'DOCKER\_USERNAME' ,credentialsId : "$HARBOR\_CREDENTIAL\_ID" ,)]) {

sh '''echo $DOCKER\_PASSWORD | docker login -u 'robot$ms' --password-stdin $REGISTRY'''

sh 'docker push $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:SNAPSHOT-$BRANCH\_NAME-$BUILD\_NUMBER'

}

}

}

}

}

stage('push latest'){

when{

branch 'dev-4.2.0.0-containerized'

}

steps{

container ('maven') {

sh 'docker tag $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:SNAPSHOT-$BRANCH\_NAME-$BUILD\_NUMBER $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:latest '

sh 'docker push $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:latest '

}

}

}

stage('deploy to dev') {

when{

branch 'dev-4.2.0.0-containerized'

}

steps {

input(id: 'deploy-to-dev', message: 'deploy to dev?')

dir('site') {

kubernetesDeploy(configs: 'deploy/dev/\*\*', enableConfigSubstitution: true, kubeconfigId: "$KUBECONFIG\_CREDENTIAL\_ID")

}

}

}

stage('push with tag'){

when{

expression{

return params.TAG\_NAME =~ /v.\*/

}

}

steps {

container ('maven') {

input(id: 'release-image-with-tag', message: 'release image with tag?')

withCredentials([usernamePassword(credentialsId: "$GITLAB\_CREDENTIAL\_ID", passwordVariable: 'GIT\_PASSWORD', usernameVariable: 'GIT\_USERNAME')]) {

sh 'git config --global user.email "lovelyhuangxingyao@163.com" '

sh 'git config --global user.name "bowen.huang" '

sh 'git tag -a $TAG\_NAME -m "$TAG\_NAME" '

sh 'git push http://$GIT\_USERNAME:$GIT\_PASSWORD@project.valorosoltd.com:9981/$GITLAB\_ACCOUNT/onlinestore/online-store.git --tags --ipv4'

}

sh 'docker tag $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:SNAPSHOT-$BRANCH\_NAME-$BUILD\_NUMBER $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:$TAG\_NAME '

sh 'docker push $REGISTRY/$HARBOR\_NAMESPACE/$APP\_NAME:$TAG\_NAME '

}

}

}

stage('deploy to production') {

when{

expression{

return params.TAG\_NAME =~ /v.\*/

}

}

steps {

input(id: 'deploy-to-production', message: 'deploy to production?')

dir('site') {

kubernetesDeploy(configs: 'deploy/prod/\*\*', enableConfigSubstitution: true, kubeconfigId: "$KUBECONFIG\_CREDENTIAL\_ID")

}

}

}

}

}