Objective of POC

Objectives of the POC is to demonstrate following capabilities:

- Provision K8s cluster using Ansible scripts . Same scripts can be used later for cluster setup across different environments and various other projects
- Setup Nexus Repository to store docker images and Maven dependency artifacts
- Automate CICD Jenkins pipeline to
    - build artifacts ,
    - Create container/docker images ,
    - Push images to nexus repository
    - Pull images from Nexus repository and deploy to K8s cluster
- Autoscaling of Pods in K8s cluster based on request traffic (i.e. based on CPU and memory usage)
- Monitoring capabilities of K8 cluster (POD, Services, Nodes etc.)

Results achieved from POC

As part of POC , we were able to achieve/demonstrate following:

- Setup K8s cluster using Ansible scripts.
- CICD Jenkins pipeline using shared-library to build images and deploy to K8s cluster
- Autoscaling of Pods in K8s cluster based on request traffic
- Monitoring capabilities of K8 cluster (POD, Services, Nodes etc.)

-

Strategy/Approach to integrate with current production system

[ANIL]TDL

Build Promotion Strategy(wip)

- Traditional approach
  - Master > PROD env
    - Release > UAT env
      - Develop > DEV/QA Env
- Current
  - UAT branch>> app2(DEV)
  - PROD branch >>> app1(QA/UAT)
    - Cherry pick necessary changes from UAT branch and merge to PROD branch
    - QA/UAT signoff
    - Same artifacts is moved to PROD
- Containerized applicaiton
  - Master branch
    - DEV branch
      - Image build, push image to nexus repo with a version
      - Deploy to DEV env, promote same image to QA/UAT env
      - QA/UAT signoff
      - Merge dev to master and tag the release
      - Deploy same image to PROD
    - Option 2: UAT branch
      - Image build, push image to nexus repo with a version
      - Deploy to QA/UATenv,
      - QA/UAT signoff
      - Merge UAT to master and tag the release
      - Deploy same image to PROD

VM's used for POC

#	VM ipaddress	Version
Ansible server	192.168.70.5	
Nexus Server	192.168.70.16	3.68.1-02
Jenkins Master	192.168.70.19	
Jenkins Slave node-1	192.168.70.17	
Jenkins Slave node-2	192.168.70.18	
Nginx server	192.168.70.13	
K8 master	192.168.70.6 192.168.70.7 192.168.70.8	
K8 worker	192.168.70.11 192.168.70.12	
ELK Cluster	192.168.150.40 192.168.150.41 192.168.150.19	
	192.168.70.14 192.168.70.15	Not used
Sonar Qube	192.168.150.41	Used existing one

K8s Cluster setup

Requirements:

Build Server

- **Ansible v2.14+, Jinja 2.11+ and python-netaddr is installed on the machine that will run Ansible commands**
- **Python3 version 3.11.2 is required**
- **Ansible Core Version 2.14.3 is required**
- **Jinja Version 3.1.2 is required**
- **Git**
- **The firewalls are not managed. We will implement our own rules and during script invocation firewall will be disabled**
- **Non root account with full sudo privileges will be required.**

- **Temporary internet access for downloading the above packages**
- **A valid Redhat Subscription manager will be required to be attached to this machine.**

Download Server

- **Download Server with full internet access will be required**
- **Unmanaged firewall. We will manage the firewall on this machine**
- **Non root user with sudo priveleges**

Code Repository hosted for this Setup:
https://apps.trigyn.com/gitlab/emigrate/emigrate-cicd.git

Installation

S.No	Ansible Role	Servers to be run on	Purpose
1	pkg-download	Download Server	1. Install yum utils 2. Add kubernetes Repository 3. Add Docker Repository 4. Download only Kubernetes packages 5. Download haproxy packages 6. Download keepalived process 7. Download all the required docker images for cluster bootstrap 8. Create Tar Zip of RPM packages 9. Create Tar Zip of saved images
2	pkg-xfer-remote	k8s-master k8s-worker	1. Transfer all image tarball to remote servers 2. Transfer kubernetes package tarball to remote server
3	lb-xfer-remote	K8s-worker, k8s-master,	1. Transfer haproxy packages to remote server 2. Transfer keepalived process to remote server
4.	unzip-artifacts		1. Unzips all the tarballs into respective servers
5.	config-iptables	K8s-worker, haproxy, master	1. Configure iptables for Bridged traffic

6	Disable-firewall	K8s-worker, k8s-master, k8s-haproxy	1. Disable firewall
7.			2.

<<Anil:TDL:How to run the ansible scripts>>

Jenkins

VM's

We used following configurations for Jenkins setup

#	vm-ipaddres
Master	192.168.70.19
Slave node-1	192.168.70.17
Slave node-2	192.168.70.18

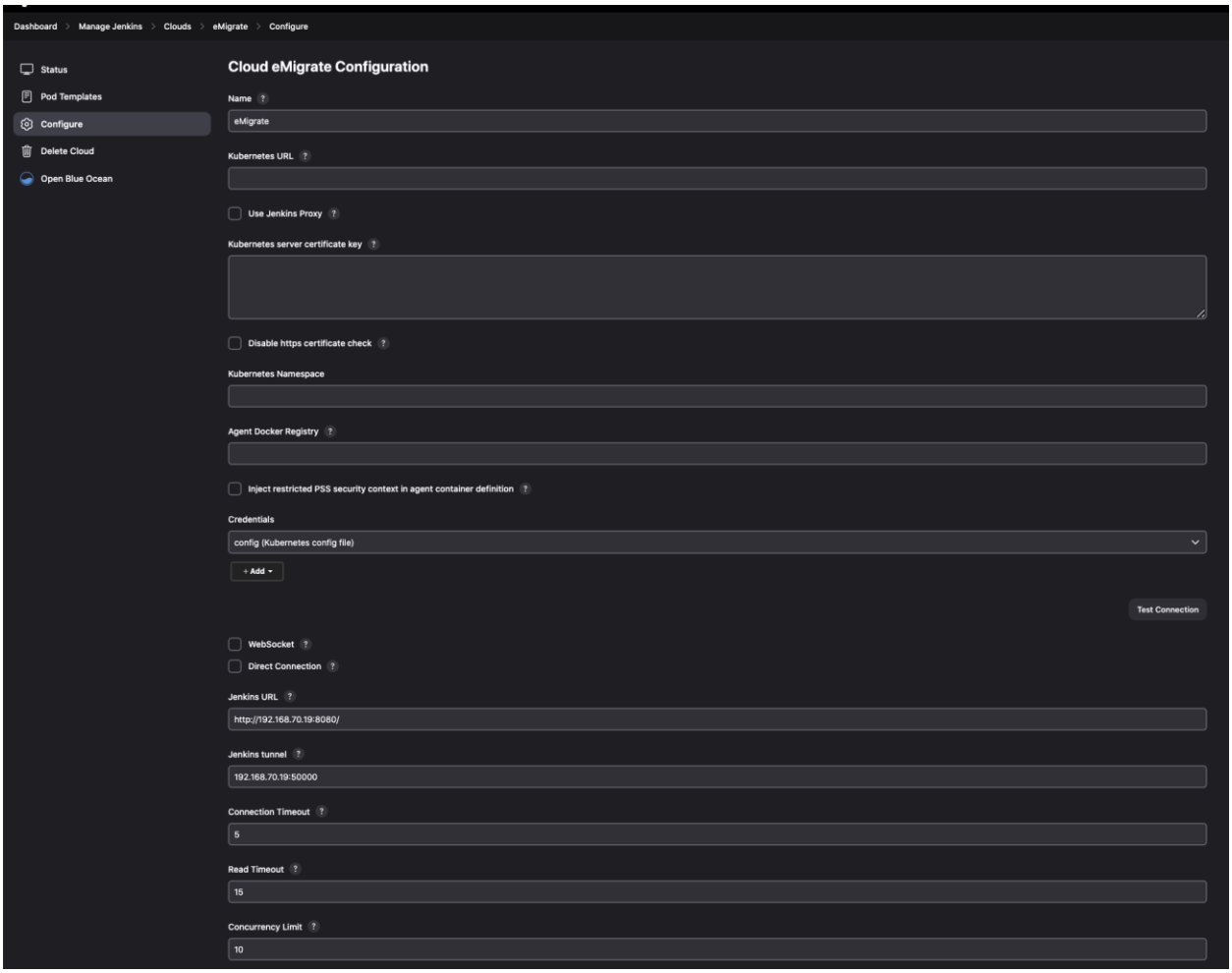
Installation

- Refer following link and section "Long Term Support release" for Jenkins installation on Linux : https://www.jenkins.io/doc/book/installing/linux/.
- Jenkins Version: 2.452.1
- Installation steps to be followed on all servers (**On master and slave nodes**)
    - sudo yum install fontconfig java-11-openjdk
    - sudo yum install git
    - sudo yum install -y yum-utils
    - sudo yum-config-manager --add-repo https://download.docker.com/linux/rhel/docker-ce.repo
- Installation steps common to **Slave nodes**
    - sudo yum install docker-ce
    - sudo systemctl start docker
    - sudo yum install -y kubectl
- To start jenkins erver
    - systemctl start jenkins
    - systemctl status jenkins

Plugins

Following plugins were installed in Jenkins

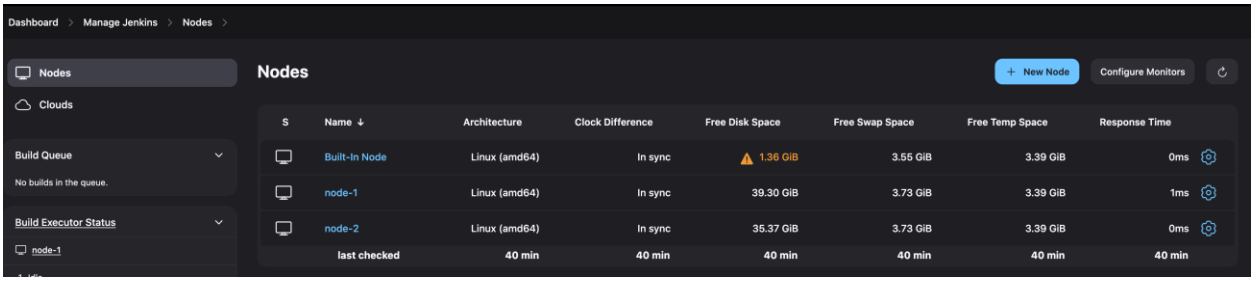
- GIT (latest)
- Maven (3.6.3)
- NodeJS (16.17.0)
- Build Timestamp Plugin
- Sonar Qube server
- Kubernetes plugin
    - Fill in the Kubernetes plugin configuration. Open the Jenkins UI and navigate to **Manage Jenkins -> Manage Nodes and Clouds -> Configure Clouds -> Add a new cloud -> Kubernetes** and enter the *Kubernetes URL* and *Jenkins URL* appropriately



    - Note: Kubeconfig named “config: should be copied to all server (master and slave of jenkins). It is placed in /root/.kube/ directory

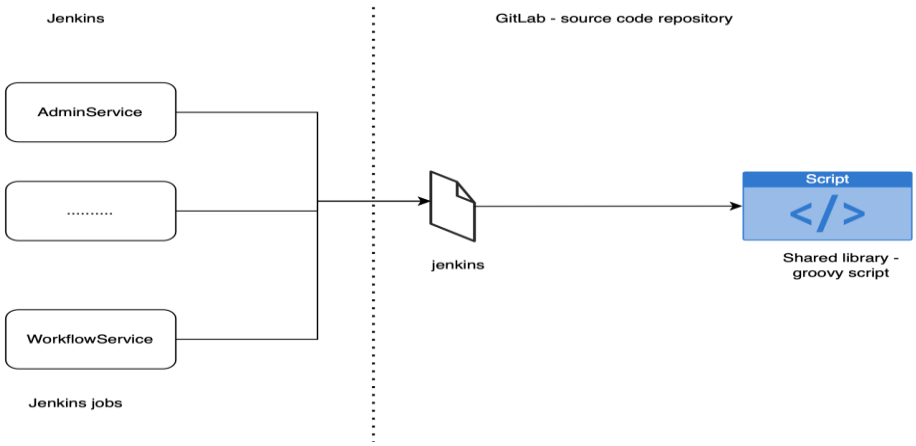
Configuration

- Configure nodes using Jenkins UI and navigate to Manage Jenkins > Nodes > New Node option (by adding node-1 and node-2)



CICD Pipeline

We use Jenkins Pipeline support for creating "Shared Libraries" which can be defined in external source control repositories and loaded into existing Pipelines.



A shared library in Jenkins is a collection of Groovy scripts shared between different Jenkins jobs. To run the scripts, they are pulled into a Jenkinsfile.
Each shared library requires users to define a name and a method of retrieving source code. Shared libraries are stored in git repositories. Developers use shared libraries to avoid writing the same code from scratch for multiple projects. Shared libraries share code across development projects, thus optimizing the software development life cycle. This drastically cuts down time spent on coding and helps avoid duplicate code

Separate Jenkins job pipeline for each micorservice (Admin, Workflow etc) is created. Each Jenkins job refer to a common Jenkin file :Jenkinsfile where we define all the stages required for a pipeline

- Build

- SonarQube-Code Analysis
- Build Docker Image and Push to Nexus
- Deployment to K8S cluster

Jenkinsfile has a reference to shared library using usign (@Library('emigrate-libraries@dev-k8-poc') _). The shared library has implementation of all the stages defined in a common place and can be refereed from Jenkinsfile

Configuring shared Library in Jenkins

- Create a Groovy Script

```
import org.apache.commons.lang.StringUtils

def call (String stageName) {

    if ("${stageName}" == 'Checkout') {
           git branch: '${ENVIRONMENT}',
     credentialsId: '692bbda9-3741-4662-8371-92df1ccfcac3',
     url: 'https://apps.trigyn.com/gitlab/emigrate/emigratebackend.git'
        }

    else if ("${stageName}" == 'Build' ) {

     sh "mvn -f ${SERVICE_NAME}/pom.xml clean install -DskipTests"

    }
```

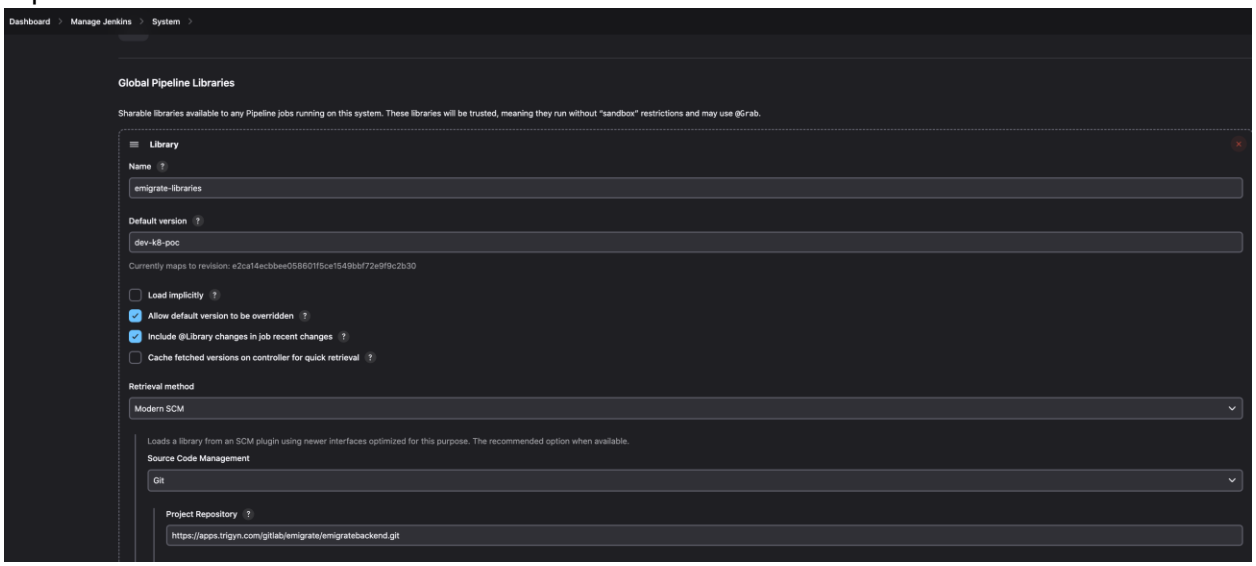
- Add the Script to a Git Repository under /vars folder i.e /vars/emigrate.groovy

dev-k8-poc emigratebackend / vars / + History

Delete sonarQube.groovy
Siddharth Ingle authored 3 days ago

Name	Last commit
..	
.gitkeep	Added new directory
cicd.groovy	Updated cicd.groovy
emigrate.groovy	Updated emigrate.groovy

- Add a Shared Library in Jenkins job using Manage Jenkins > Configure System > Global Pipeline Libraries



- Use a Shared Library in a Jenkins Pipeline

emigrate > emigratebackend > **Repository**

dev-k8-poc emigratebackend / **Jenkinsfile**

Updated Jenkinsfile
Siddharth Ingle authored 3 days ago

Jenkinsfile 1.2 KB

```
@Library('emigrate-libraries@dev-k8-poc') _
emigrate
pipeline {

    agent {
        label "${AGENT}"
    }

    tools {
        maven "Maven 363"
    }

 /*   environment {
        SERVICE_NAME = "${param.SERVICE}"

    } */

    stages {
        stage('Build') {
            steps {
              emigrate ('Build')
            }
        }

        stage('SonarQube-Code Analysis') {
            steps {
                echo "Code Analysis"
                emigrate ('codeQualityAnalysis')
            }
        }



            stage('Docker Image and Push to Nexus') {
                steps {
                    echo "Creating an Docker Image"
                   emigrate ('dockerImagePushToNexus')
               }

        }



            stage('Deployment to K8S') {
                steps {
               emigrate ('deploymentToKuberentes')

                }
```

Jenkinsfile#L2

Note: Prerequisites
/tmp folder should have exec rights in all Jenkin servers(master, slave-node-1, slave-node2)

Jenkins URL

http://192.168.70.19:8080/

Nexus

VM's

#	VM ipaddress	Version	
Ansible server	192.168.70.5		
Nexus Server	192.168.70.16	3.68.1-02	
Nginx server	192.168.70.16		For DNS routing

Installation

- We used Ansible for Nexus installation. Please refer following ansible script from git repository https://apps.trigyn.com/gitlab/emigrate/emigrate-cicd/-/blob/master/kubernetes-setup/k8-ansible/playbook-nexus.yml
- Connect/ssh to Ansible server and
    - Copy/cloned the above repo https://apps.trigyn.com/gitlab/emigrate/emigrate-cicd/-/blob/master/kubernetes-setup/ in following location on ansible sevre /home/meauser/
    - cd kubernetes-setup/k8-ansible i.e. /home/meauser/kubernetes-setup/k8-ansible
    - install Nexus by running ansible nexus playbook using following command

    sudo ansible-playbook playbook-nexus.yml

Options for Pulling images from Nexus

Option1: Without SSL (insecure registry , skipping SSL verification)

"containerd" configuration for k8s master and worker node

```
vi /etc/containerd/config.toml
config_path = "/etc/containerd/certs.d"     // add this line

mkdir   /etc/containerd/certs.d/192.168.70.16:5081/
cd /etc/containerd/certs.d/192.168.70.16:5081/
vi hosts.toml

server = "http://192.168.70.16:5081"

[host."http://192.168.70.16:5081"]
capabilities = ["pull", "resolve", "push"]
skip_verify = true

systemctl restart containerd
```

Option 2: With SSL

Generating CSR on Nexus Server

```
openssl req -new -newkey rsa:2048 -nodes -keyout nexuspoc.mea-emigrate.com.key -out
nexuspoc.mea-emigrate.com.csr
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:Delhi
Locality Name (eg, city) [Default City]:Delhi
Organization Name (eg, company) [Default Company Ltd]:trigyn
Organizational Unit Name (eg, section) []:IT
Common Name (eg, your name or your server's hostname) []:*.mea-emigrate.com
Email Address []:symc@trigyn.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:          enter only
An optional company name []:   enter only

[root@EMA-PoC-Nexus home]# ll
total 36
drwx------. 2 root    root     16384 Jan 18 23:38 lost+found
drwx------. 6 meauser  meauser   4096 Jun 10 20:41 meauser
drwx------. 5 nexus   nexus     4096 May 30 14:16 nexus
-rw-r-----. 1 root    root      1050 Jun 11 12:16 nexuspoc.mea-emigrate.com.csr
-rw-------. 1 root    root      1704 Jun 11 12:15 nexuspoc.mea-emigrate.com.key
drwx------. 3 sysadmin sysadmin  4096 Jan 19 00:47 sysadmin

We got the Key and we have to provide csr file to IT team(Kiran) they will generate cert and  will
provide cert file
```

Configuring Nginx server on nexus server 192.168.70.16

```
yum update -y
yum install nginx
vi /etc/nginx/nginx.conf

events {
}

http {

proxy_send_timeout      120;
proxy_read_timeout      300;
proxy_buffering         off;
keepalive_timeout       5 5;
tcp_nodelay             on;
```

```
# cert path
ssl_certificate /etc/nginx/certs/9ca25b051f4f8c07.crt;
# key path
ssl_certificate_key /etc/nginx/certs/nexuspoc.mea-emigrate.com.key;

client_max_body_size     0;

server {
listen 80;
server_name nexuspoc.mea-emigrate.com;
return 301 https://nexuspoc.mea-emigrate.com$request_uri;
}
server {
listen              443 ssl;
location ~ ^/(v1|v2)/[^/]+/?[^/]+/blobs/ {
if ($request_method ~* (POST|PUT|DELETE|PATCH|HEAD) ) {
rewrite ^/(.*)$ /repository/emigrate-docker-hosted-repo/$1 last;
}
rewrite ^/(.*)$ /repository/emigrate-docker-group-repo/$1 last;
}

location ~ ^/(v1|v2)/ {
if ($request_method ~* (POST|PUT|DELETE|PATCH) ) {
rewrite ^/(.*)$ /repository/emigrate-docker-hosted-repo/$1 last;
}
rewrite ^/(.*)$ /repository/emigrate-docker-group-repo/$1 last;
}


location / {
proxy_pass          http://192.168.70.16:8081/;
proxy_redirect      off;
proxy_set_header     Host $host;
proxy_set_header     X-Real-IP $remote_addr;
proxy_set_header      X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header      X-Forwarded-Host $server_name;
proxy_set_header      X-Forwarded-Proto $scheme;
}
}
}


to verify
nginx -t
```

```
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
setenforce 0
reboot

systemctl restart nginx.service
```

Configure DNS entry

```
# given details to Infra team(Kiran) for DNS entry
nexuspoc.mea-emigrate.com
IP 192.168.70.16

# internal ip
# https://nexuspoc.mea-emigrate.com/    Nexus Server
# DNS server IP 192.168.110.14
# this changes will required on K8s worker node and CICD server

cat /etc/resolv.conf
# Generated by NetworkManager
#nameserver 8.8.8.8
// added this line
nameserver 192.168.110.14

# to work resolve.conf with reboot  ( Both on Jenkins Node and Nexus node)
vi /etc/NetworkManager/system-connections/ens33.nmconnection
        dns=192.168.110.14

systemctl restart NetworkManager

#to trust the certificate so we can do docker login
#copied zip file of certificate given by kiran which contains root certificate
#copied .crt file ( root and chained certificates) inside below directories

/etc/pki/ca-trust/source/anchors
```

Trust the certificates

for k8s worker node servers only

```
/etc/pki/ca-trust/source/anchors
```

```
update-ca-trust extract
systemctl restart docker

# for k8s worker node only servers
systemctl restart containerd
docker login nexuspoc.mea-emigrate.com
docker-admin
account@123
```

Create secret to pull the image from Nexus

```
kubectl create secret -n emigrate-dev-ns docker-registry nexuspoc --docker-server=nexuspoc.mea-emigrate.com --docker-username=docker-admin --docker-password=account@123
```

```
replicas: 1
  selector:
    matchLabels:
      app: emigrateui
  template:
    metadata:
      labels:
        app: emigrateui
    spec:
      containers:
      - name: emigrateui
        image: nexuspoc.mea-emigrate.com/emigrate-ui:latest
        imagePullPolicy: Always
        ports:
        - containerPort: 80
      imagePullSecrets:
       - name: nexuspoc
```

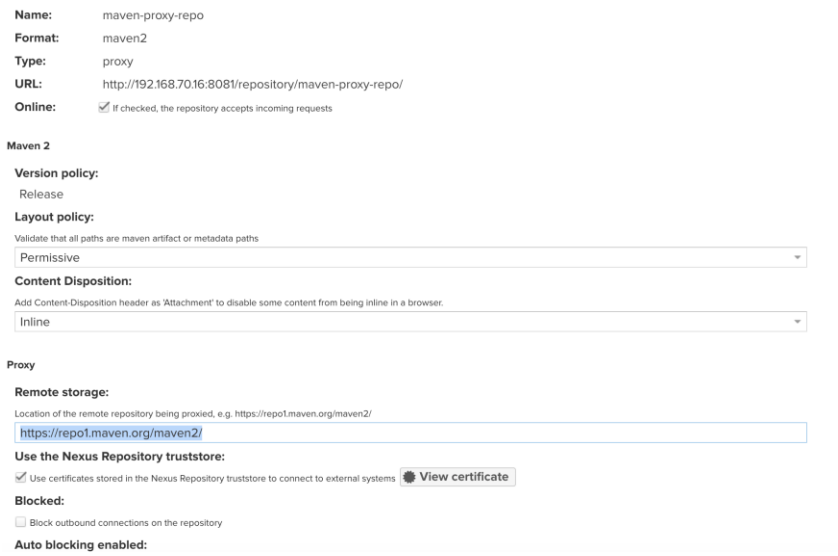
Caching Dependencies Artifacts

For maven build

Through Nexus UI please create a new repository of

- type maven2(proxy),
- Maven2(versionPolicy:Relase, Layout:permissive,Contentdisposition:Inline)

- Proxy (remote stoarage:https://repo1.maven.org/maven2/, Auto blockign enabled:false)

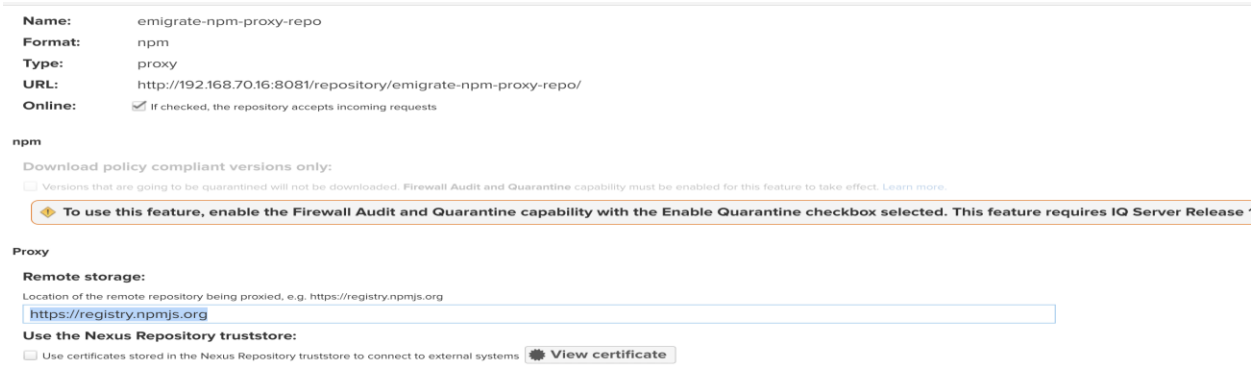


- Create a settings.xml to be used with maven command in Jenkins build . The settings file should refer to maven repository which we have creatded above i.e. to https://nexuspoc.mea-emigrate.com/repository/maven-proxy-repo/. Refer settings.xml from https://apps.trigyn.com/gitlab/emigrate/emigratebackend/-/blob/dev-k8-poc/Maven/settings.xml
- Refer custom settings.xml file during maven build
  using mvn -s <<YourOwnSettings.xml>> clean install
  OR
  configure the settings.xml file in Jenkins using JenkinsUI > Manage Jenkins > System Configuration > Manage files

For NPM Build

Similarly for npm dependencies used in building UI application, create a repository of type maven2(proxy) with Proxy (remote storage: https://registry.npmjs.org)

.nprmc(ANIL:TDL)



Nexus URL

http://192.168.70.16:8081/

Configure docker image repository

Through Nexus UI, select option Administration > Repository > Repositories > Create Repository

- Name: emigrate-docker-hosted-repo
- Format:docker
- Type:hosted
- HTTP Port:5081

Nginx ingress controller

On Ansible server (192.168.70.5)

- Install helm using /home/meauser/helm-v3.15.1-linux-amd64.tar.gz i.e. unzip helm-v3.15.1-linux-amd64.tar.gz and get the helm executable. Optional: If needed copy helm executable to /root/bin
- **emigrate-docker-group-repo:** for pull docker images from 2 proxy and 2 hosted repository
- **emigrate-docker-hosted-repo: to push emigrate applicaiton images**
- Create a **docker proxy** repository named **emigrate-k8s-prox**y and mapped to remote repository **https://registry.k8s.io/**
- Create repository in nexus named **emigrate-helm-proxy-repo** (http://192.168.70.16:8081/repository/emigrate-helm-proxy-repo/) with
  - Format:helm, type:Proxy
  - Proxy remote storage: https://kubernetes.github.io/ingress-nginx

    - Auto blocking enabled: false
- Add ingress-nginx-4.10.1.tgz to Nexus repo emigrate-helm-proxy-repo
- Unzip ingress-nginx-4.10.1.tgz in /home/meauser to get values.yml file
- Modify values.yml for registry to be used for following elements in yml:
    - controller , opentelemetry, revisionHistoryLimit
        - r**egistry: nexuspoc.mea-emigrate.com**
- Install Nginx contoller server on 192.168.70.13 for application routing
    - Install Nginx ingress contrller usign helm
        - helm install myrelease testrepo/ingress-nginx -f values.yml
    - Configure nginx.conf (Anil:pending)
        - Same wildcard certificates used i.e ***.mea-emigrate.com**
        - Port mapping (ingress controller ip-address, port)
    - Kubectl apply ingress2.yml
- Verify the UI and Services endpoint
    - curl --header 'Host: emigrateapi.mea-emigrate.com' http://192.168.70.11:32014/api/admin/business
    - curl --header 'Host: emigrateui.mea-emigrate.com' http://192.168.70.11:32014

MISC