**Cài đặt Jenkins trong kubernetes**

**Jenkins được cài với Helm chart.**

curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

**Kiểm tra:**

helm version

***Chuẩn bị trước khi cài Jenkins vào kubernetes***

**1. Tạo namespaces cho jenkins**

kubectl create namespace jenkins

Xem secret đang tồn tại:

Kubectl get secret -n jenkins

**2. Tạo secret để tạo kubeconfig cho kubect trong pod jenkins thông qua secret:** *(secret đọc file, cung cấp cho tạo aws/credentials trong pod jenkins trên kubernetes, cho phép sử dụng aws thông qua thông tin đăng nhập lưu ở aws/ credentials cho jenkins. Trong lệnh dùng cần dùng “$HOME” thay vì “~” bới vì “~” là shell nên không được hiểu trong câu lệnh kubectl, còn “$HOME” là biến môi trường nên sẽ luôn đúng cho các trường hợp).*

kubectl create secret generic aws-credentials --from-file=credentials=$HOME/.aws/credentials -n jenkins

**3. Chuẩn bị image cho jenkins bằng docker, lưu trên docker hub.**

Login docker trước khi đẩy image lên docker hub

docker login (mở của url trong cửa sổ login, tiêp theo dán mã được cấp từ cửa số login vào trang login trên trình duyệt)

docker build -t vothinhuydt/jenkins:txu .

docker push vothinhuydt/jenkins:txu

**4. Tạo thư mục để mout vào jenkins lưu trữ dữ liệu lâu dài khi pod jenkins bị xóa tạo lại…**

sudo mkdir -p /mnt/data/jenkins

sudo chown -R 1000:1000 /mnt/data/jenkins

Reboot máy để có hiệu lực: reboot

ls -ldn /mnt/data/jenkins

ls -ld /mnt/data/jenkins

**5. Cài đặt Jenkins với helm**

helm install jenkins ./jenkins-chart -n jenkins

**\* Gỡ cài đặt Jenkins**

helm uninstall jenkins -n jenkins

**6. Sau khi cài đặt jenkins bằng helm, chạy jenkins thiết lập ban đầu cho jenkins**

a. jenkins hiện tại được cài đặt và chạy trong kubernetes và được ingress vào theo url: jenkins.txuapp.com. Do đó từ node control-plane, cài đặt cloudflare và tunnel url: jenkins.txuapp.com đến NodePort của Ingress Controller ở port(xxxxx:8080). Các bước từ tạo tunnel, route dns, và run tunnel như sau:

- Tạo tunnel cho jenkins: (sẽ tạo ra một id và file id.json của tunnel, dùng thông tin này để tạo file chạy tunnel route dns)

cloudflared tunnel create jenkins-tunnel

**Tạo một route dns cho gán với tunnel vừa tạo**

cloudflared tunnel route dns jenkins-tunnel jenkins.txuapp.com

**Tạo file “cloudflare-jenkins-tunnel.yml” config thông tin để chạy tunnel, nội dung:**

tunnel: 4a06d22d-240f-4e63-b772-58502933d65f

credentials-file: ~/.cloudflared/4a06d22d-240f-4e63-b772-58502933d65f.json

ingress:

- hostname: jenkins.txuapp.com

service: http://localhost:31179

- service: http\_status:404

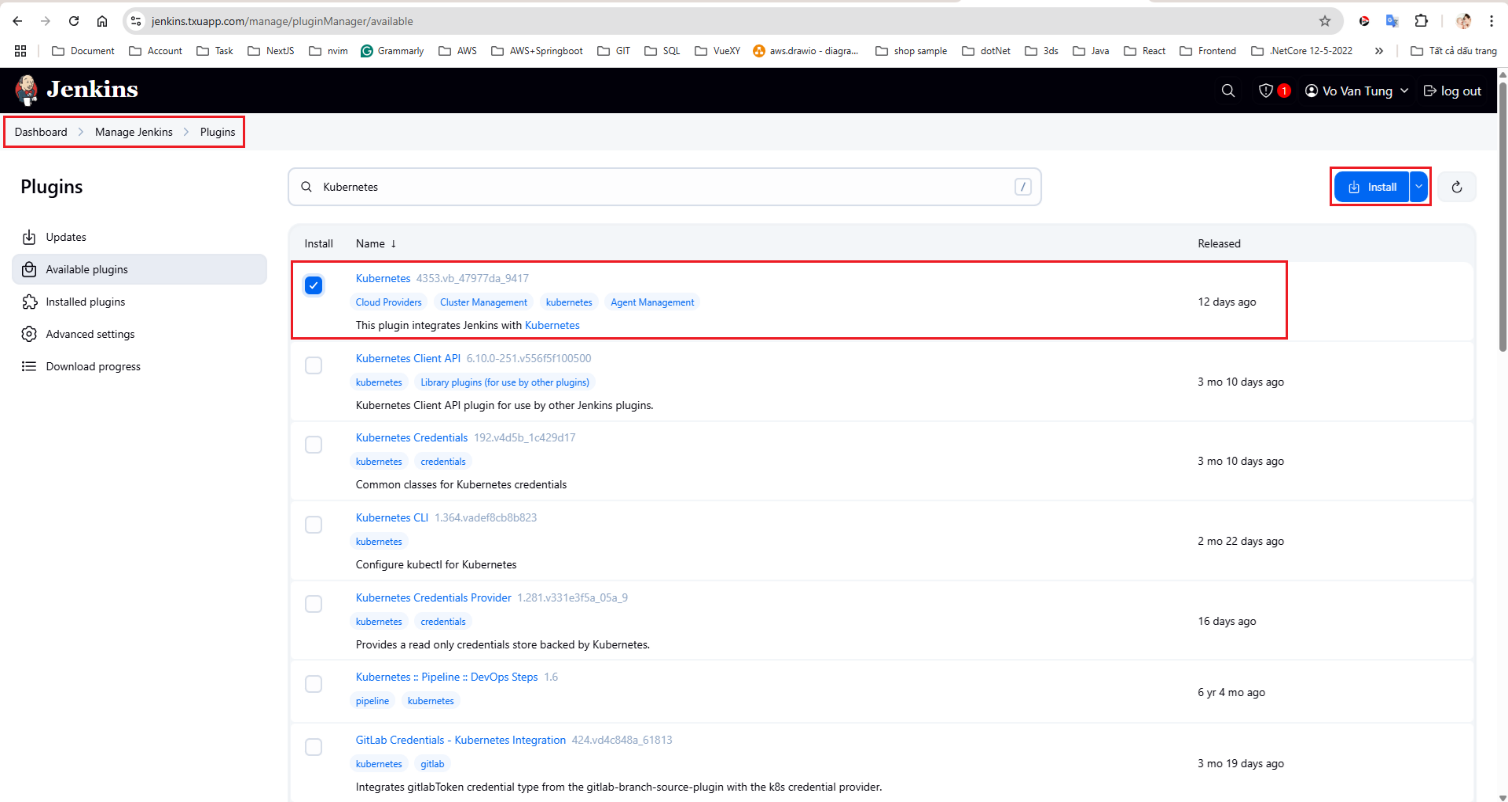
**Chạy tunnel với file config:**

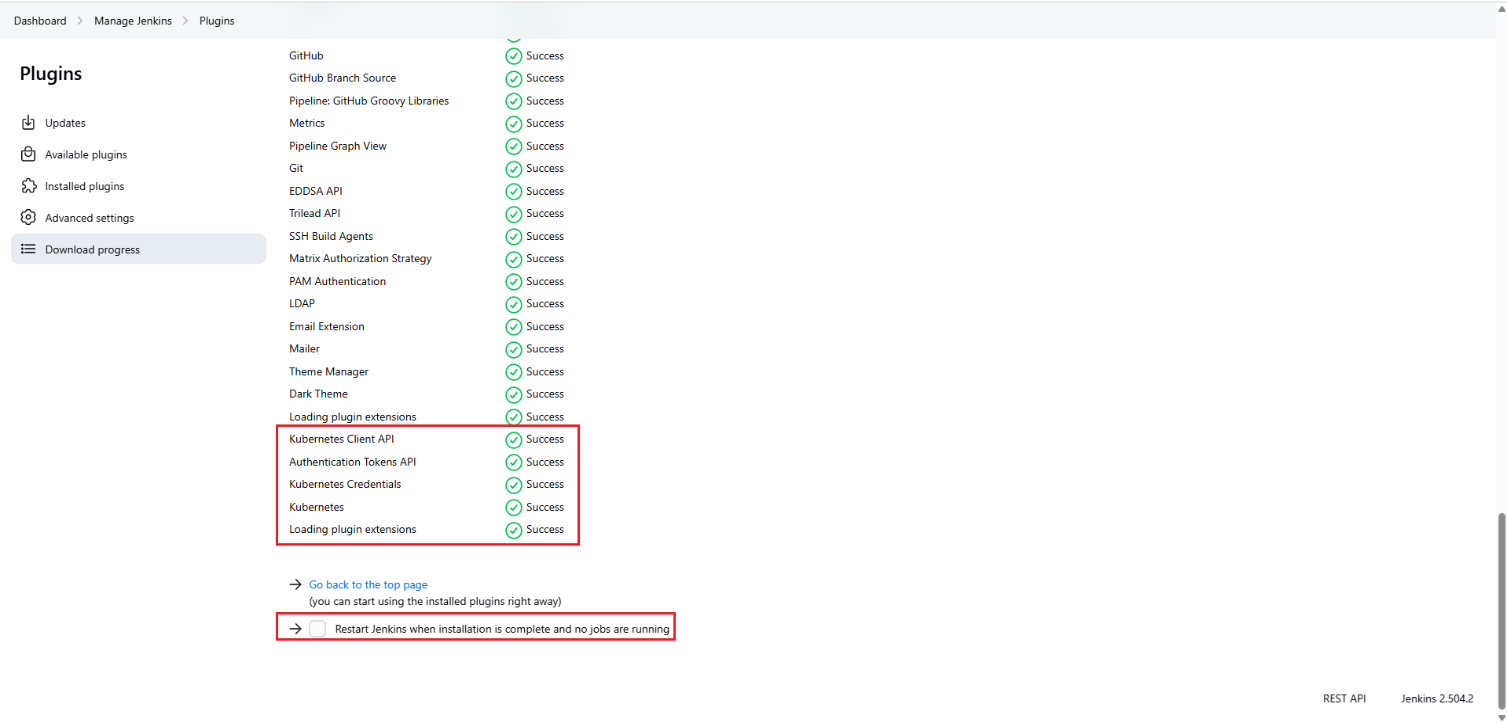
cloudflared tunnel –config cloudflare-jenkins-tunnel.yml run

b. Lấy mật khẩu để đăng nhập và thiết lập ban đầu cho jenkins:

kubectl exec -it -n jenkins <jenkins-pod-name> -- cat /var/jenkins\_home/secrets/initialAdminPassword

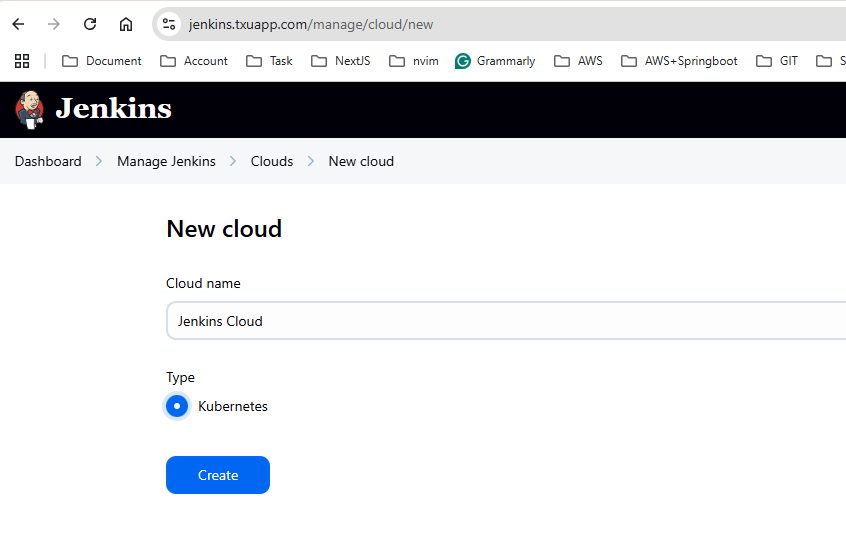
c. Tiếp theo cài đặt plusgin kubernetes cho jenkins trên UI của jenkins (việc cài đặt plusgin kubernetes cho phép jenkins chạy các jenkins agent và giao tiếp với jenkins controller thông qua dns nội bộ của jenkins “http://jenkins.jenkins.svc.cluster.local.:8080” ):



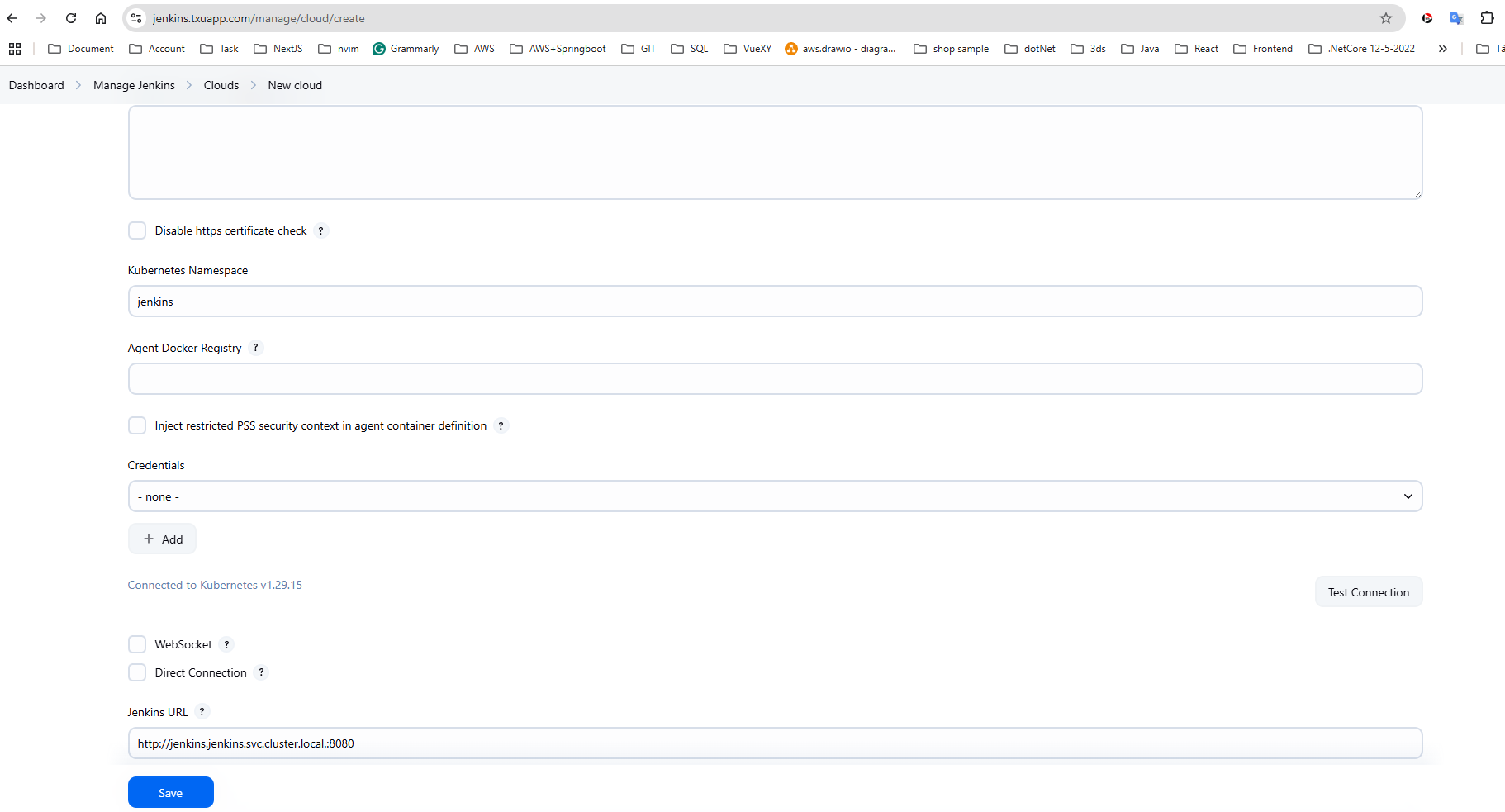


d. Cấu hình jenkins cloud kiểu kubernetes để cho phép các jenkins agent có thể giao tiếp jenkins controller. Do các jenkins agent chạy trong cluster cùng với jenkins controller nên cần cấu hình “URL Jenkins: <http://jenkins.jenkins.svc.cluster.local.:8080>”. Lưu ý: kết thúc url có đặt dấu “.” Để báo cho dns trong kubernetes biết đây là url đầy đủ mà không thêm phần searches, gây sai url và dns không phân giải chính xác. Mặt khác nếu không dùng url đầy đủ như trên thì cũng xảy ra tình trạng searches ở các container khác nhau sẽ không có những hậu tố “ví dụ: cluster.local” theo yêu cầu. Ngoài ra vì các jenkins agent và controller đều giao tiếp nội bộ nên cách hợp lý nhất là dùng dns nội bộ thay vì dùng dns public “jenkins.txuapp.com” vì khi đó yêu cầu phải mở thêm port cho jenkins ra bên ngoài, các agent phải truy cập từ bên ngoại thay vì truy cập ngay bên trong, điều đó không hợp lý và cần nhiều cấu hình hơn.

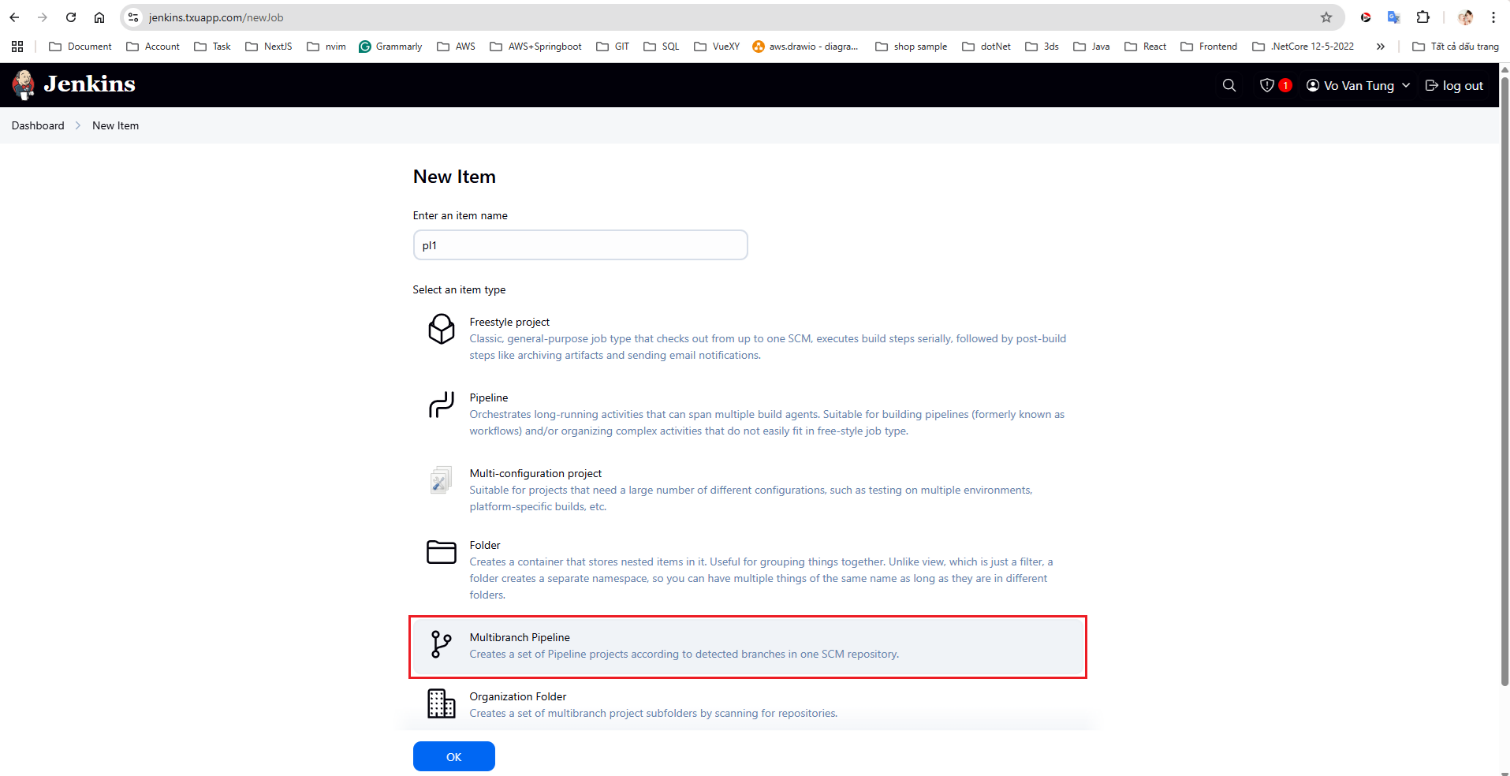
Để cấu hình jenkins cloud kiểu kubernetes thì trước hết cần tạo một cloud kiểu kubernetes:



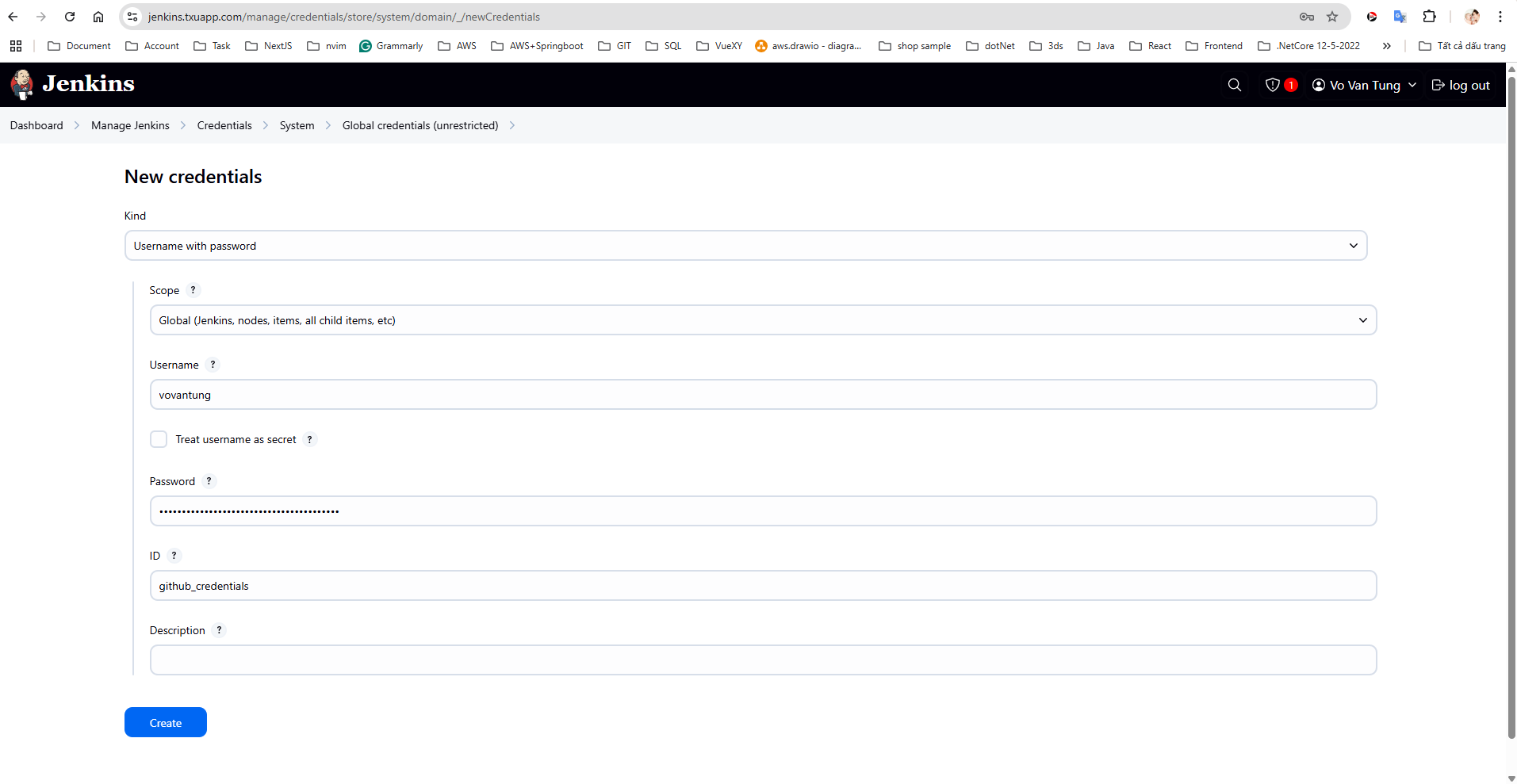
Chú ý: hai mục “Kubernetes Namespaces” và “Jenkins URL”



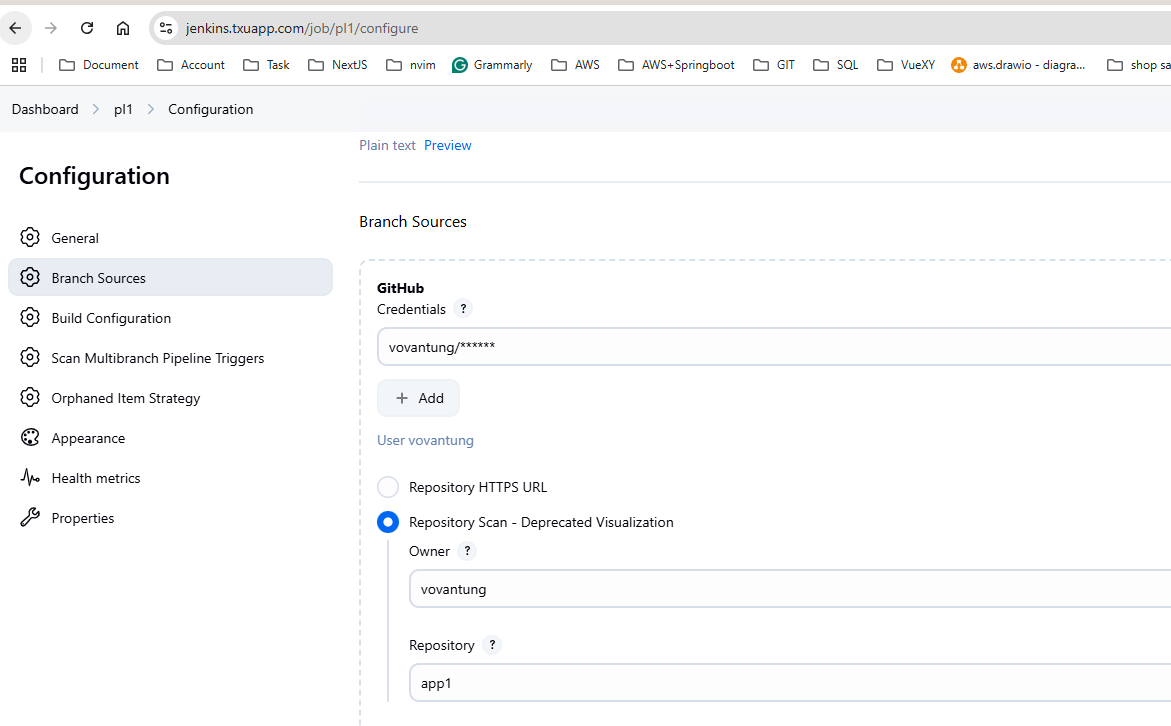
e. Tạo và chạy một job kiểu “multi branch”



Để cấu hình job kiểu “multibranch pipeline”, cần tạo **credentials** để đăng nhập github, thông tin cần có là username, passwork (token generate từ github)



Cấu hình job

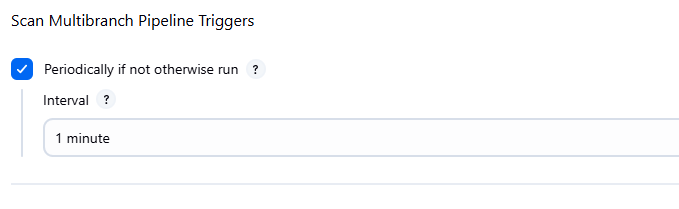
Mục “Behaviours” add thêm Filter by name (with wildcards), nhập Include: “main” nếu muốn chỉ trigger job khi push code trên nhánh “main”



Mục “Build Configuration” chọn “Script Path” là đường dẫn Jenkinsfile trong thư mục gốc của code project

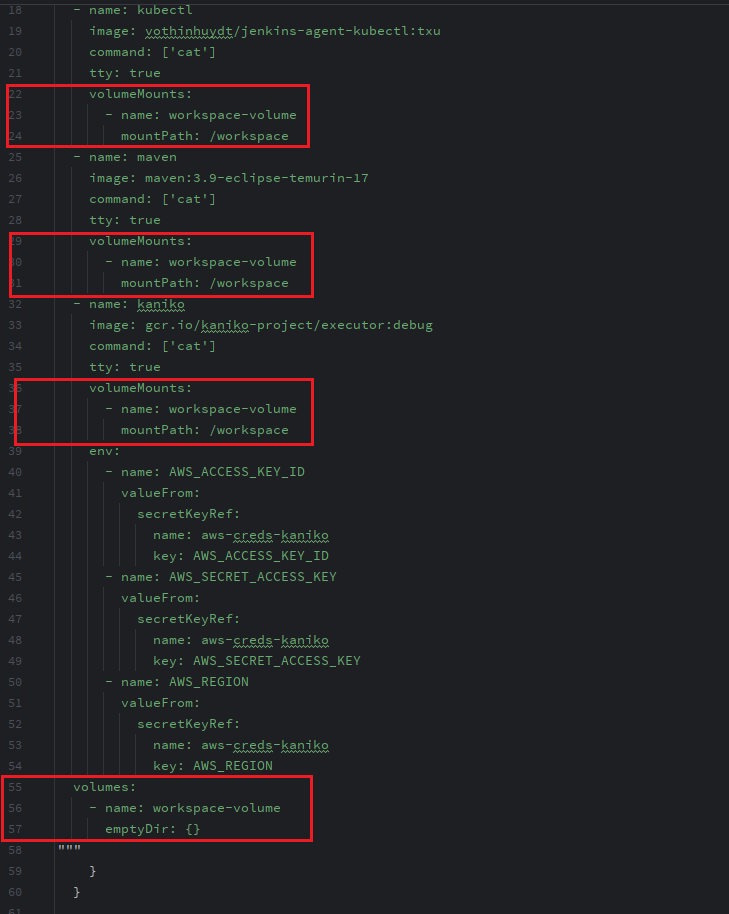


Mục “Scan Multibranch Pipeline Triggers”:



Khi đã tạo job kiểu “multibranch pipeline” như trên, và khi có sự kiện push code lên branch được chỉ định của repo (ví dụ push code lên branch “main” của repo “https://github.com/vovantung/app1.git”). Job được trigger sẽ chạy pipeline được định nghĩa trong Jenkinsfile. Khi cloud kubernetes được cài đặt trong jenkins thì các jenkins agent sẽ thực hiện các stage và giao tiếp với jenkins controller qua “Jenkins URL” mà ta đã chỉ định trong phần cấu hình cloud của jenkins (phần trên đã chỉ định giao tiếp qua dns nội bộ “<http://jenkins.jenkins.svc.cluster.local.:8080>”). Tuy nhiên một số jenkins agent ngoài giao tiếp với jenkins controller qua dns nội bộ đã được chỉ định (full url) thì chúng cũng cần thực hiện với giao tiếp với bên ngoài internet (chẳng hạn kaniko cần giao tiếp với docker.io để pull image “eclipse-temurin:17-jdk-alpine” cho việc build image từ dockerfile). Do đó với container kaniko cần đặt “ndots:1” để tránh việc dns nội bộ thực hiện gắn thêm phần searches (chẳng hạn “cluster.local”) gây ra không thể phân giải dns này.

Trong mô hình jenkins agent thực hiện nhiệm vụ được giao từ jenkins controller và các jenkins agent giao tiếp trở lại với jenkins controller thông qua dns nội bộ “<http://jenkins.jenkins.svc.cluster.local.:8080>” thì nhiệm vụ chủ yếu được thực hiện bởi các jenkins agent (là các container riêng biệt và có thể chia sẻ dữ liệu chung).



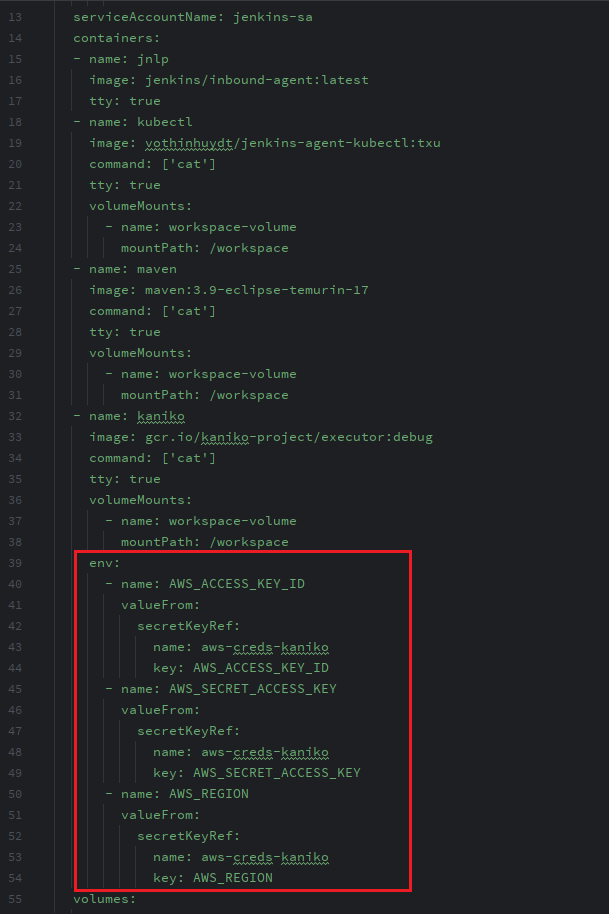
Khi đó jenkins controller chỉ đóng vai trò trung tâm giao nhiệm vụ (nằm trong các stage, chẳng hạn clone code, build code với maven, build code thành iamge với kanino và push lên các repo như ecr, docker hub;…) cho các jenkins agent, các jenkins agent thực hiện và trả lời lại jenkins controller.

*Chi tiết cụ thể các agent được định nghĩa và thực hiện nhiệm vụ cụ thể (xem thêm trong Jenkinsfile)*

Ví dụ, trong jenkins agent (kaniko) thực hiện nhiệm vụ nhận dữ liệu chung được chia sẻ từ jenkins agent (maven), nhận dockerfile và file app2.jar đã build bới jenkins agent (maven) ở stage trước đó, và thực hiện trước pull image “eclipse-temurin:17-jdk-alpine” từ internete, sau đó build thành image chứa ứng dụng app1.jar và push image mới build lên ecr. Kaniko được thiết kế có khả năng build image mà không cần docker meaon, và push image lên ecr hoặc docker hub… Quá trình push image lên erc chẳng hạn cần phải có quyền push image, do đó cần gắn các biến môi trường (thông qua secret trong kubernetes) chứa thông tin xác thực với aws thực hiện quyền push image lên ecr.

kubectl create secret generic aws-creds-kaniko --from-literal=AWS\_ACCESS\_KEY\_ID=aws-access-key-id --from-literal=AWS\_SECRET\_ACCESS\_KEY=aws-secret-key --from-literal=AWS\_REGION=ap-southeast-1 -n jenkins

Việc thực hiện gắn secret cho kaniko được thực hiện bằng cách tạo một secret dạng secret key và gắn thông tin đó vào container kaniko ở bước định nghĩa kamoko trong jenkinsfile.



Tương tự kaniko, thì jenkins agent (kubectl) cũng cần thực hiện việc kết nối với kubernetes api server để có thể dùng command “kubectl”. Để thực hiện được việc này, cần thực hiện tạo một “ServiceAccount” tên “” có quyền “rbac.authorization.k8s.io/v1” tức có quyền lấy thông tin xác thực của kubernetes.

“apiVersion: v1

kind: ServiceAccount

metadata:

name: jenkins-sa

namespace: {{ .Release.Namespace }}

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: jenkins-sa-cluster-admin

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: jenkins-sa

namespace: {{ .Release.Namespace }}”

Khi gán “serviceAccountName: jenkins-sa” cho một pod thì các contriner của pod sẽ chứa các quyền xác thực với k8s, khi đó chỉ cần cài kubectl cli, kubectl cli sẽ thực hiện lấy quyền xác thực được cấp và cấu hình cho kubeconfig (nằm trong /var/jenkins\_home/.kube/config). Có nghĩa là , các container được gán quyền “rbac.authorization.k8s.io/v1” và cài đặt kubectl cli sẽ thực hiện các lệnh “kubectl”.

