19 | 自托管构建：如何使用 Harbor 搭建企业级镜像仓库？

2023-01-20 王炜 来自北京

《云原生架构与GitOps实战》

课程介绍 >

讲述：王炜

时长 10:04　大小 9.19M

你好，我是王炜。

在上一节课，我带你学习了 Tekton 的基本概念以及如何使用 Tekton 构建镜像。这种方案最大的优势在于可以直接在 Kubernetes 集群中构建，我们不再需要单独为镜像构建付费。

但是，在之前的构建方案中，我们是将镜像推送到了 Docker Hub 镜像仓库。实际上，Docker Hub 也是一个收费的服务，对于免费用户来说，它限制每 6 小时最多拉取 200 次镜像，显然，这对团队来说是完全不够用的。

在这节课，我就带你学习如何使用 Harbor 来搭建企业级的镜像仓库，将它集成到我们上一节课创建的 Tekton Pipeline 流程中，最终替换 Docker Hub，进一步降低镜像存储的成本。此外，在安装 Harbor 的过程中，我还会首次介绍 Helm 工具的使用方法。

在开始今天的学习之前，你需要按照上一节课的内容准备好一个云厂商的 Kubernetes 集群，安装 Ingress-Nginx 和 Tekton，并配置好 Pipeline 和 GitHub Webook。

此外，在生产环境下，Harbor 一般都会开启 TLS，**所以你还需要准备一个可用的域名。**

下面，我们进入今天的实战环节。

安装 Helm

在我们之前的实践中，像是安装 Tekton 和 Ingress-Nginx 都是通过 Kubernetes Manifest 来完成的。实际上，安装 Kubernetes 应用并不只有一种方案，这里我们介绍第二种方案。**Helm**。

这节课，我们先学会使用 Helm 就可以了，更详细的内容我们会在后续的课程做介绍。

要使用 Helm，首先需要安装它，你可以通过这个链接查看不同平台的安装方法，这里我使用官方提供的脚本来安装。

```
$ curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | ba
```

安装好 Helm 之后，在正式使用之前，你还要确保本地 Kubectl 和集群的连通性，Helm 和 Kubectl 默认读取的 Kubeconfig 文件路径都是 ~/.kube/config。

安装 Cert-manager

接下来我们安装 Cert-manager，它会为我们自动签发免费的 Let's Encrypt HTTPS 证书，并在过期前自动续期。

首先，运行 helm repo add 命令添加官方 Helm 仓库。

```
$ helm repo add jetstack https://charts.jetstack.io
"jetstack" has been added to your repositories
```

然后，运行 helm repo update 更新本地缓存。

复制代码

```
$ helm repo update
...Successfully got an update from the "jetstack" chart repository
```


接下来，运行 helm install 来安装 Cert-manager。

复制代码

```
$ helm install cert-manager jetstack/cert-manager \
--namespace cert-manager \
--create-namespace \
--version v1.10.0 \
--set ingressShim.defaultIssuerName=letsencrypt-prod \
--set ingressShim.defaultIssuerKind=ClusterIssuer \
--set ingressShim.defaultIssuerGroup=cert-manager.io \
--set installCRDs=true

NAME: cert-manager
LAST DEPLOYED: Mon Oct 17 21:26:44 2022
NAMESPACE: cert-manager
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
cert-manager v1.10.0 has been deployed successfully!
```

此外，还需要为 Cert-manager 创建 ClusterIssuer，用来提供签发机构。将下面的内容保存为 cluster-issuer.yaml。

复制代码

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: "wangwei@gmail.com"
    privateKeySecretRef:
      name: letsencrypt-prod
    solvers:
    - http01:
        ingress:
          class: nginx
```

注意，这里你需要将 spec.acme.email 替换为你真实的邮箱地址。然后运行 kubectl apply 提交到集群内。


```
$ kubectl apply -f cluster-issuer.yaml
clusterissuer.cert-manager.io/letsencrypt-prod created
```

到这里，Cert-manager 就已经配置好了。

安装和配置 Harbor

安装 Harbor

现在，我们同样使用 Helm 来安装 Harbor，首先添加 Harbor 官方仓库。

```
$ helm repo add harbor https://helm.goharbor.io
"harbor" has been added to your repositories
```

然后，更新本地 Helm 缓存。

```
$ helm repo update
...Successfully got an update from the "jetstack" chart repository
...Successfully got an update from the "harbor" chart repository
```

接下来，由于我们需要定制化安装 Harbor，所以需要修改 Harbor 的安装参数，将下面的内容保存为 values.yaml。

```
expose:
  type: ingress
  tls:
    enabled: true
    certSource: secret
    secret:
      secretName: "harbor-secret-tls"
      notarySecretName: "notary-secret-tls"
```

```
  ingress:
    hosts:
      core: harbor.n7t.dev
      notary: notary.n7t.dev
    className: nginx
    annotations:
      kubernetes.io/tls-acme: "true"
persistence:
  persistentVolumeClaim:
    registry:
      size: 20Gi
    chartmuseum:
      size: 10Gi
    jobservice:
      jobLog:
        size: 10Gi
      scanDataExports:
        size: 10Gi
    database:
      size: 10Gi
    redis:
      size: 10Gi
    trivy:
      size: 10Gi
```


注意，由于腾讯云的 PVC 最小 10G 起售，所以，除了 registry 的容量以外，我将安装参数中其他的持久化卷容量都配置为了 10G，你可以根据安装集群的实际情况做调整。

另外，我还为 Harbor 配置了 ingress 访问域名，分别是 harbor.n7t.dev 和 notary.n7t.dev，**你需要将它们分别替换成你的真实域名。**

然后，再通过 helm install 命令来安装 Harbor，**并指定参数配置文件 values.yaml**。

复制代码

```
$ helm install harbor harbor/harbor -f values.yaml --namespace harbor --create-
NAME: harbor
LAST DEPLOYED: Mon Oct 17 21:53:28 2022
NAMESPACE: harbor
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Please wait for several minutes for Harbor deployment to complete.
Then you should be able to visit the Harbor portal at https://core.harbor.domai
For more details, please visit https://github.com/goharbor/harbor
```

等待所有 Pod 处于就绪状态。


```
$ kubectl wait --for=condition=Ready pods --all -n harbor --timeout 600s
pod/cm-acme-http-solver-4v4zz condition met
pod/harbor-chartmuseum-79f49f5b4-8f4mb condition met
pod/harbor-core-6c6bc7fb4f-4822f condition met
pod/harbor-database-0 condition met
pod/harbor-jobservice-85d448d5c9-gn5pk condition met
pod/harbor-notary-server-848bcc7ccd-m5m5v condition met
pod/harbor-notary-signer-6897444589-6vssq condition met
pod/harbor-portal-588b64cbdb-gqlbn condition met
pod/harbor-redis-0 condition met
pod/harbor-registry-5c7d58c87c-6bgsj condition met
pod/harbor-trivy-0 condition met
```

到这里，Harbor 就已经安装完成了。

配置 DNS 解析

接下来，我们为域名配置 DNS 解析。首先，获取 Ingress-Nginx Loadbalancer 的外网 IP。


```
$ kubectl get services --namespace ingress-nginx ingress-nginx-controller --out
43.135.82.249
```

然后，为域名配置 DNS 解析。在这个例子中，我需要分别为 harbor.n7t.dev 和 notary.n7t.dev 配置 A 记录，并指向 43.135.82.249。

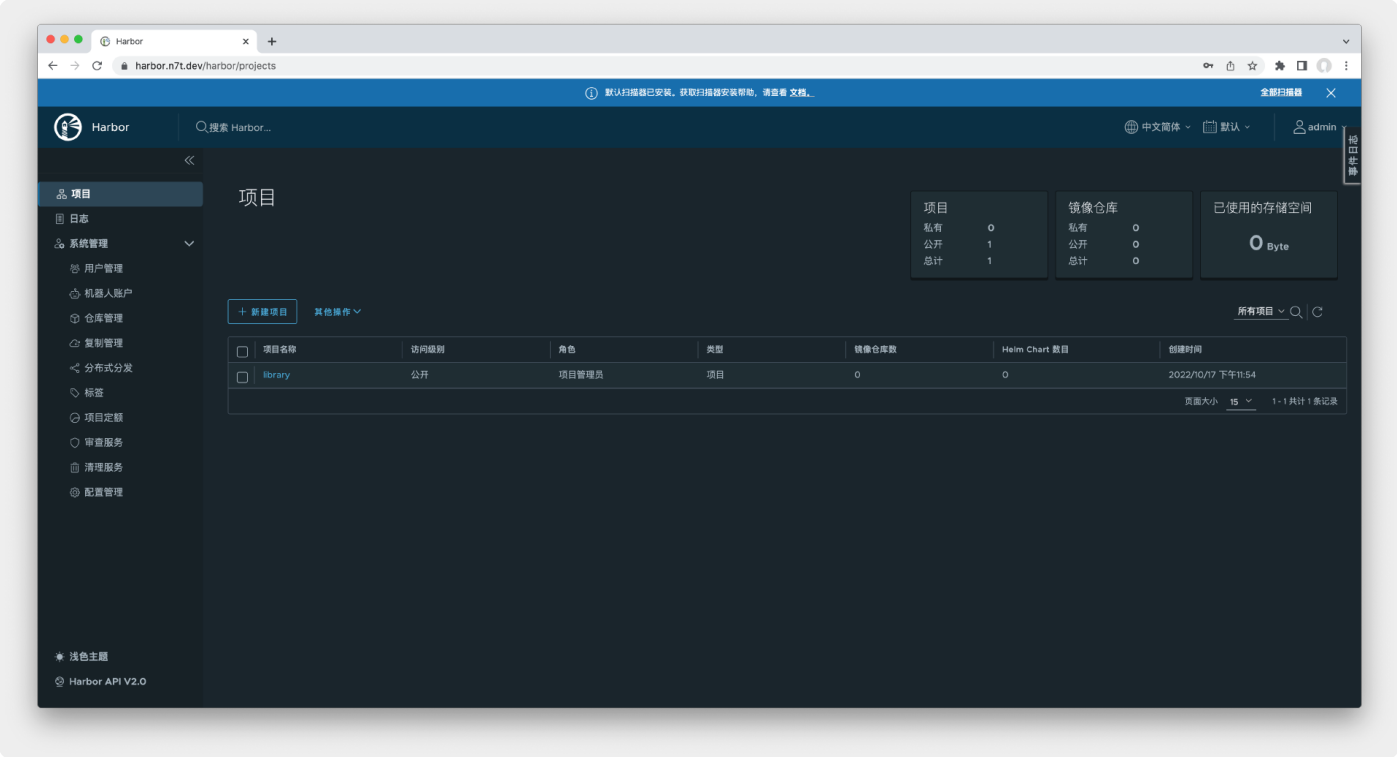
访问 Harbor Dashboard

在访问 Harbor Dashboard 之前，首先我们要确认 Cert-manager 是否已经成功签发了 HTTPS 证书，你可以通过 kubectl get certificate 命令来确认。


```
$ kubectl get certificate -A
NAMESPACE   NAME                READY   SECRET              AGE
harbor      harbor-secret-tls   True    harbor-secret-tls   8s
harbor      notary-secret-tls   True    notary-secret-tls   8s
```

由于我们在部署 Harbor 的时候需要配置两个域名，所以这里会出现两个证书。当这两个证书的 Ready 状态都为 True 时，说明 HTTPS 证书已经签发成功了。此外，**Cert-manager 自动从 Ingress 对象中读取了 tls 配置**，还自动创建了名为 harbor-secret-tls 和 notary-secret-tls 两个包含证书信息的 Secret。

接下来，打开 https://harbor.n7t.dev 进入 Harbor Dashboard，使用默认账号 admin 和 Harbor12345 即可登录控制台。



Harbor 已经自动为我们创建了 library 项目，我们在后续的阶段将直接使用它。

推送镜像测试

现在，让我们来尝试将本地的镜像推送到 Harbor 仓库。首先，在本地拉取 busybox 镜像。

复制代码

```
$ docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
f5b7ce95afea: Pull complete
Digest: sha256:9810966b5f712084ea05bf28fc8ba2c8fb110baa2531a10e2da52c1efc504698
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest
```

然后，运行 docker login 命令登录到 Harbor 仓库，使用默认的账号密码。


```
$ docker login harbor.n7t.dev
username: admin
password: Harbor12345
Login Succeeded
```

接下来，重新给 busybox 镜像打标签，指向 Harbor 镜像仓库。

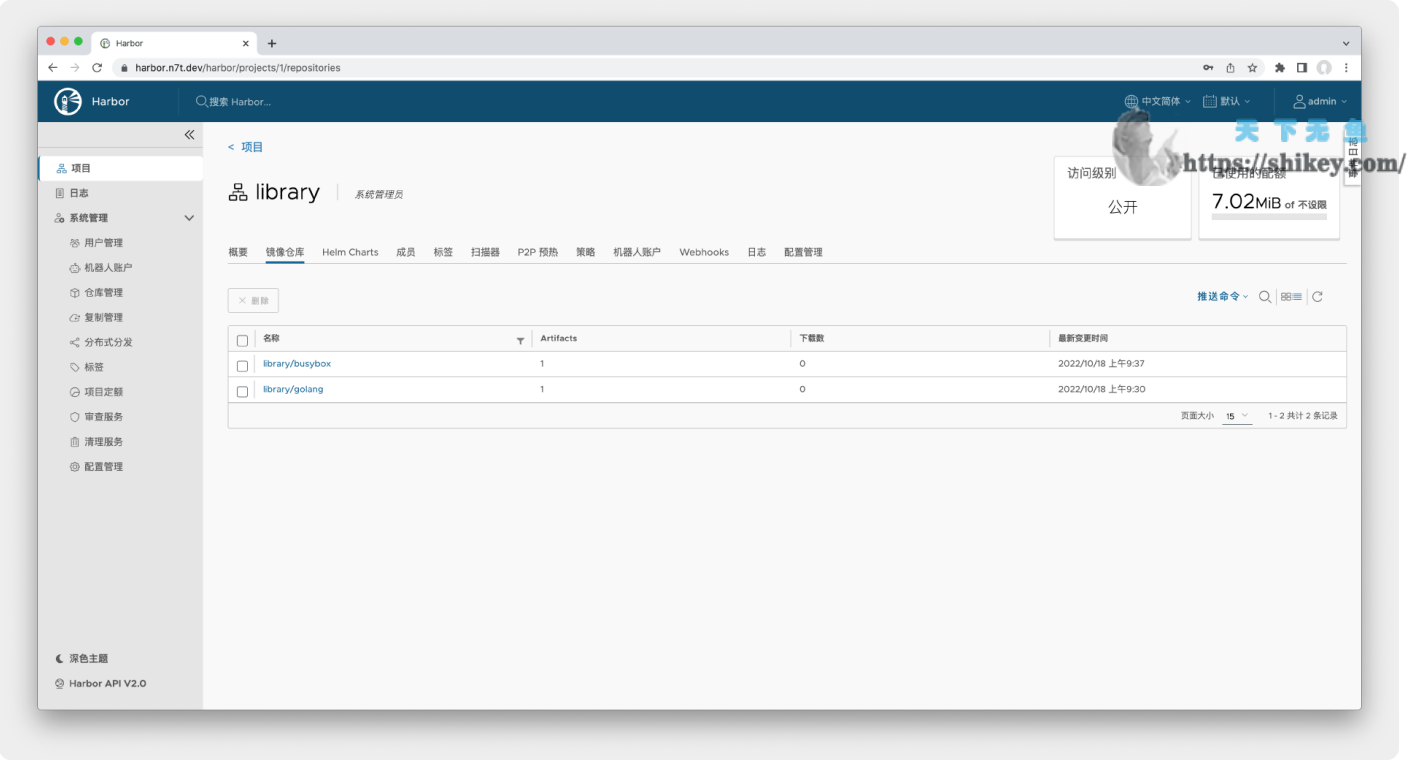

```
$ docker tag busybox:latest harbor.n7t.dev/library/busybox:latest
```

和推送到 Docker Hub 的 Tag 相比，推送到 Harbor 需要指定完整的**镜像仓库地址、项目名和镜像名**。在这里，我使用了默认的 library 项目，当然你也可以新建一个项目，并将 library 替换为新的项目名。

最后，将镜像推送到仓库。


```
$ docker push harbor.n7t.dev/library/busybox:latest
The push refers to repository [harbor.n7t.dev/library/busybox]
0b16ab2571f4: Pushed
latest: digest: sha256:7bd0c945d7e4cc2ce5c21d449ba07eb89c8e6c28085edbcf6f5fa4bf
```

镜像推送成功后，访问 Harbor 控制台，进入 library 项目详情，你将看到我们刚才推送的镜像。



到这里，Harbor 镜像仓库就已经配置好了。

在 Tekton Pipeline 中使用 Harbor

要在 Tekton Pipeline 中使用 Harbor，我们需要将 Pipeline 中的 spec.params.registry_url 变量值由 docker.io 修改为 harbor.n7t.dev，并且将 spec.params.registry_mirror 变量值修改为 library。你可以使用 kubectl edit 命令来修改。

复制代码

```
$ kubectl edit Pipeline github-trigger-pipeline
......
  params:
  - default: harbor.n7t.dev   # 修改为 harbor.n7t.dev
    name: registry_url
    type: string
  - default: "library"   # 修改为 library
    name: registry_mirror
    type: string

pipeline.tekton.dev/github-trigger-pipeline edited
```

（提示：按下 i 进入编辑模式，修改完成后，按下 ESC 退出编辑模式，然后输入 :wq 保存生效。）

然后，再修改镜像仓库的凭据，也就是 registry-auth Secret。


```
$ kubectl edit secret registry-auth
apiVersion: v1
data:
  password: SGFyYm9yMTIzNDUK    # 修改为 Base64 编码：Harbor12345
  username: YWRtaW4K            # 修改为 Base64 编码：admin
kind: Secret

secret/registry-auth edited
```

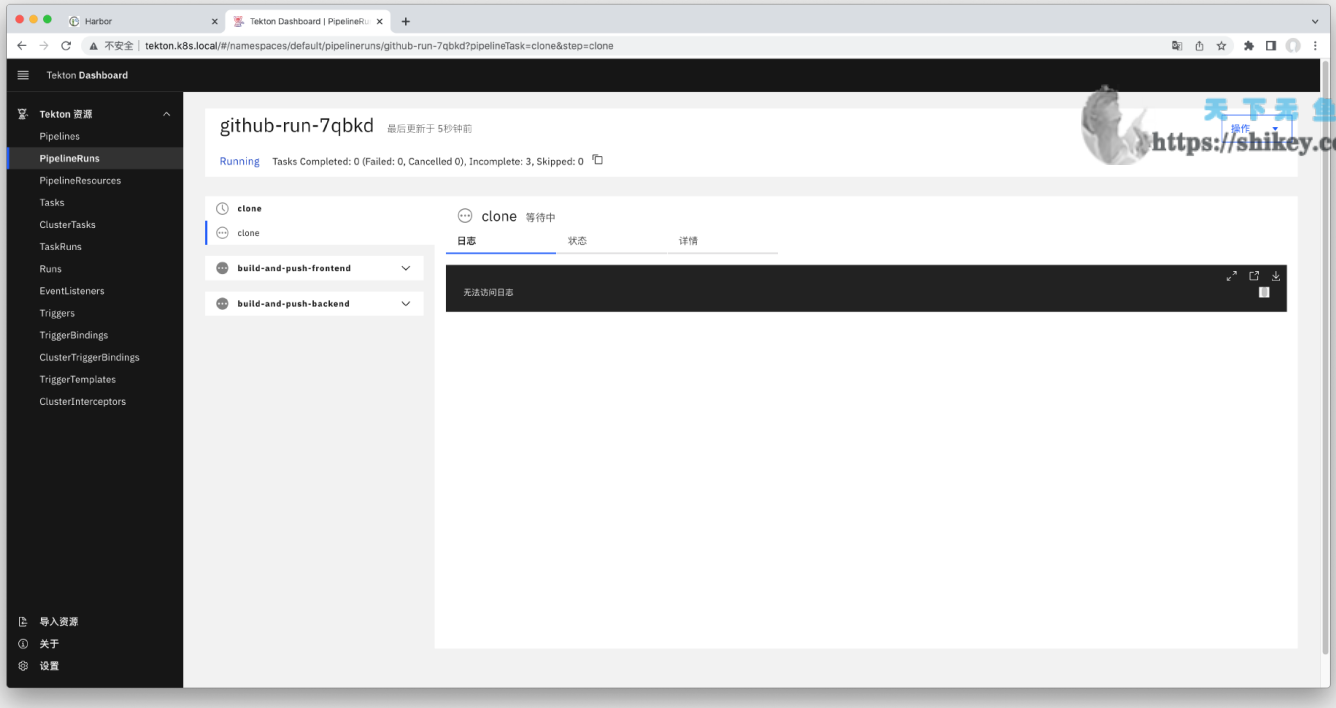
保存后生效。

现在，回到本地示例应用 kubernetes-example 目录，向仓库推送一个空的 commit 来触发 Tekton 流水线。

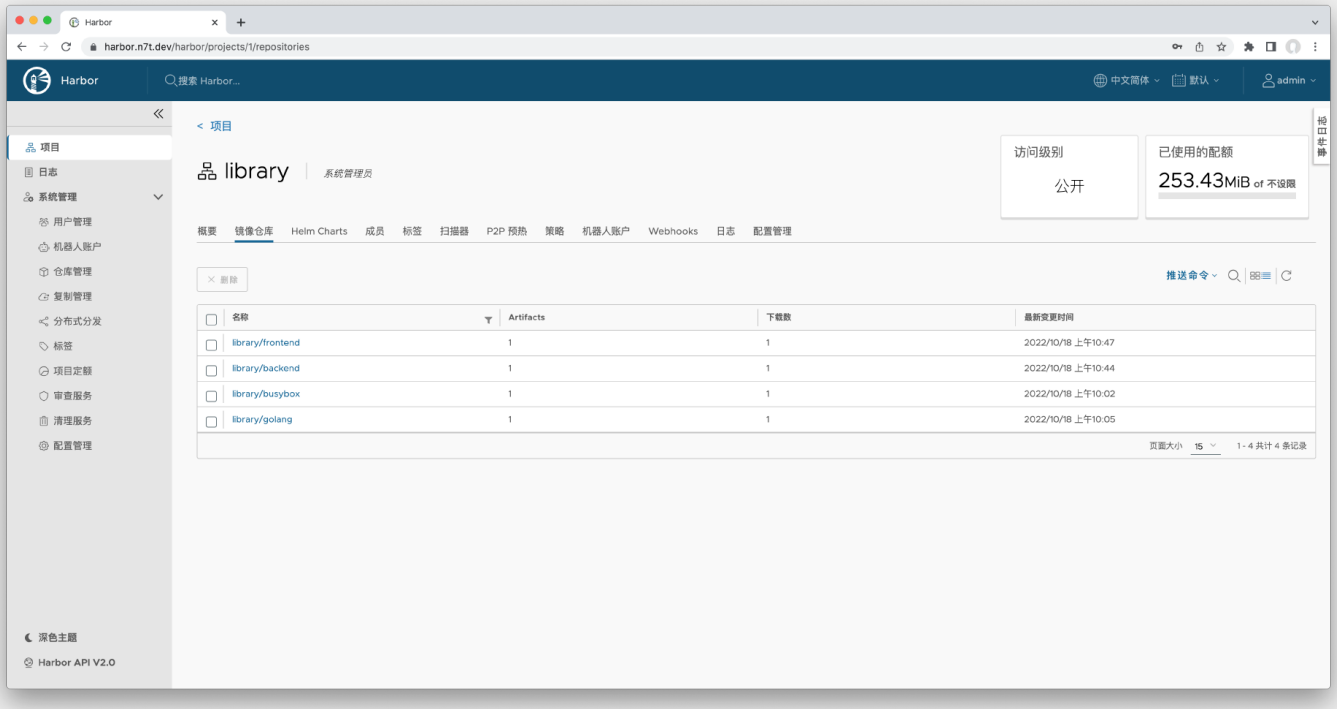

```
$ git commit --allow-empty -m "Trigger Build"
[main e42ac45] Trigger Build

$ git push origin main
```

进入 Tekton Dashboard http://tekton.k8s.local/，并查看流水线运行状态。



当流水线运行结束后，我们进入 Harbor Dashboard 会看到刚才 Tekton 推送的新镜像。



到这里，Harbor 的安装和配置就完成了。在最开始，我们是通过最小化的配置安装 Harbor，在生产环境下你可能需要注意一些额外的配置。

Harbor 生产建议

Harbor 的安装配置相对较多，这里我也提供几点生产建议供你参考。

1. 确认 PVC 是否支持在线扩容。
2. 尽量使用 S3 作为镜像存储系统。
3. 使用外部数据库和 Redis。
4. 开启自动镜像扫描和阻止漏洞镜像。

接下来我们对每一项进行详细的介绍。

确认 PVC 是否支持在线扩容

如果你是按照这节课的安装方式使用 PVC 持久卷来存储镜像的，那么随着镜像数量的增加，你需要额外注意 Harbor 仓库存储容量的问题。

一个简单的方案是在 Harbor 安装前，提前确认 StorageClass 是否支持在线扩容，以便后续对存储镜像的持久卷进行动态扩容。你可以使用 kubectl get storageclass 命令来确认。

复制代码

```
$ kubectl get storageclass
NAME            PROVISIONER                RECLAIMPOLICY   VOLUMEBINDINGMODE
cbs (default)   com.tencent.cloud.csi.cbs   Delete          Immediate
```

在返回内容中，如果 ALLOWVOLUMEEXPANSION 为 true，就说明支持在线扩容。否则，你需要手动为 StorageClass 添加 AllowVolumeExpansion 字段。

复制代码

```
$ kubectl patch storageclass cbs -p '{"allowVolumeExpansion": true}'
```

推荐使用 S3 存储镜像

除了使用持久卷来存储镜像以外，Harbor 还支持外部存储。如果你希望大规模使用 Harbor 又不想关注存储问题，那么使用外部存储是一个非常的选择。例如使用 AWS S3 存储桶来存储镜像。

S3 存储方案的优势是，它能为我们提供接近无限存储容量的存储系统，并且按量计费的方式成本也相对可控，同时它还具备高可用性和容灾能力。

要使用 S3 来存储镜像，你需要在安装时修改 Harbor 的安装配置 values.yaml。


```yaml
expose:
  type: ingress
  tls:
    enabled: true
    certSource: secret
    secret:
      secretName: "harbor-secret-tls"
      notarySecretName: "notary-secret-tls"
  ingress:
    hosts:
      core: harbor.n7t.dev
      notary: notary.n7t.dev
    className: nginx
    annotations:
      kubernetes.io/tls-acme: "true"
persistence:
  imageChartStorage:
    type: s3
    s3:
      region: us-west-1
      bucket: bucketname
      accesskey: AWS_ACCESS_KEY_ID
      secretkey: AWS_SECRET_ACCESS_KEY
      rootdirectory: /harbor
  persistentVolumeClaim:
    chartmuseum:
      size: 10Gi
    jobservice:
      jobLog:
        size: 10Gi
      scanDataExports:
        size: 10Gi
     ......
```

注意，要将 S3 相关配置 region、bucket、accesskey、secretkey 和 rootdirectory 字段修改为实际的值。

然后，再使用 helm install -f values.yaml 来安装。

使用外部数据库和 Redis

在安装 Harbor 时，会默认自动安装数据库和 Redis。但是为了保证稳定性和高可用，我建议你使用云厂商提供的 Postgres 和 Redis 托管服务。

要使用外部数据库和 Redis，你同样可以在 values.yaml 文件中直接指定。

复制代码

```
expose:
  type: ingress
  tls:
    enabled: true
    certSource: secret
    secret:
      secretName: "harbor-secret-tls"
      notarySecretName: "notary-secret-tls"
  ingress:
    hosts:
      core: harbor.n7t.dev
      notary: notary.n7t.dev
    className: nginx
    annotations:
      kubernetes.io/tls-acme: "true"
database:
  type: external
  external:
  host: "192.168.0.1"
  port: "5432"
  username: "user"
    password: "password"
  coreDatabase: "registry"
  notaryServerDatabase: "notary_server"
  notarySignerDatabase: "notary_signer"
redis:
  type: external
  external:
  addr: "192.168.0.2:6379"
    password: ""
persistence:
  ......
```

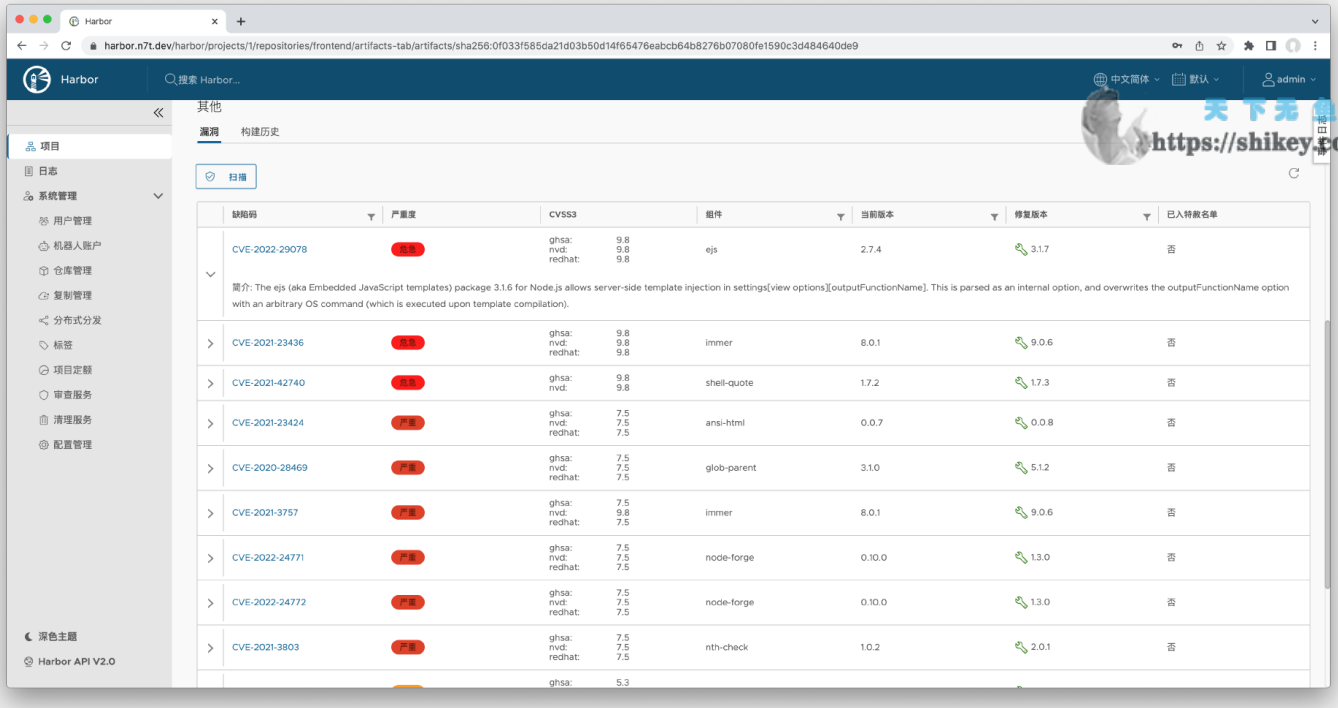
注意，要将 database 和 redis 字段的连接信息修改为实际的内容，并提前创建好 registry、notary_server 和 notary_signer 数据库。

开启自动镜像扫描和阻止漏洞镜像

最后一个建议。由于 Harbor 自带 Trivy 镜像扫描功能，可以帮助我们发现镜像的漏洞，并提供修复建议。因此在生产环境下，我推荐你开启“自动镜像扫描”和“阻止潜在漏洞镜像”功能，你可以进入项目的“配置管理”菜单开启它们。

开启“自动扫描镜像”功能后，所有推送到 Harbor 的镜像都会自动执行扫描，你可以进入镜像详情查看镜像漏洞数量。

进入“Artifacts”详情可以查看漏洞详情和修复建议。



开启“阻止潜在漏洞镜像”功能之后，接下来我们尝试在本地拉取 frontend 镜像，会发现 Harbor 阻止了这个行为。

```
$ docker pull harbor.n7t.dev/library/frontend:8d64515
Error response from daemon: unknown: current image with 14 vulnerabilities cann
```

总结

在这节课，我向你介绍了如何使用 Harbor 搭建企业级镜像仓库，在安装 Harbor 的过程中，我还简单介绍了 Helm 工具的使用方法，包括添加 Helm 仓库和安装应用。

在生产环境下，我们一般会使用 HTTPS 协议来加密访问请求，所以在安装 Harbor 之前，我向你介绍了 Cert-manager 组件，它可以帮助我们自动签发 Let’s Encrypt HTTPS 证书，并按照 Ingress 的配置生成 Secret。需要注意的是，Let’s Encrypt 证书的有效期是 90 天，不过 Cert-manager 在到期前会自动帮助我们续期，使用起来非常方便。

其次，我还介绍了如何在 Tekton Pipeline 中使用 Harbor，这里的重点是要修改 Pipeline 的仓库地址以及在 Secret 中配置的镜像仓库用户名和密码，以便 Tekton 能顺利地将镜像推送到 Harbor 仓库中。这里还有一个小细节，当我们将镜像推送到 Docker Hub 时，只需要在镜像 Tag 前面加上用户名前缀，例如 lyzhang1999/frontend:latest 即可。但当我们需要将镜像推送

到其他镜像仓库时，则需要把 Tag 配置为完整的地址，例如 harbor.n7t.dev/library/frontend:latest 才能够推送。

最后，我还给你提了 4 个 Harbor 的生产建议，你可以结合项目的实际情况来选择性地采用。

到这里，自动化镜像构建这个章节就全部结束了。在接下来的课程中，我会为你介绍 Manifest 以外其他两种更高级的应用定义格式：Kustomize 和 Helm Chart。显然，这节课提到的 Cert-manager 和 Harbor 就是以 Helm Chart 的方式来定义的。同时，我也会将示例应用以这两种方式进行封装改造，带你深入了解 Kubernetes 的应用定义。

思考题

最后，给你留一道思考题吧。

请你尝试改造第 16 讲中 GitHub Action Workflow，实现将镜像推送到 Harbor 中。

提示：为 docker/login-action@v2 插件增加 registry 参数，并将 docker/build-push-action@v3 插件的 Tag 字段修改为包含完整 Harbor 仓库的 URL。

欢迎你给我留言交流讨论，你也可以把这节课分享给更多的朋友一起阅读。我们下节课见。

上一篇 18丨自托管构建：如何使用 Tekton 构建镜像？

下一篇 20丨应用定义：如何使用 Kustomize 定义应用？

精选留言 (5)

Noel ZHANG
2023-01-30 来自江苏

Harbor搭建过程中使用了大量的云厂商组件，而且还有数倍的运维工作。我不明白，为什么不直接用云厂商的镜像库。

1

ghostwritten
2023-02-01 来自北京

界面https可以登陆。但命令行登陆：docker login harbor.geekcloudnative.com -u admin -p Harbor12345
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Error response from daemon: Get "https://harbor.geekcloudnative.com/v2/": Get "https://core.harbor.domain/service/token?account=admin&client_id=docker&offline_token=true&service=harbor-registry": dial tcp: lookup core.harbor.domain on 8.8.8.8:53: no such host

ghostwritten
2023-02-01 来自北京

记录一次部署遇到的问题：
安装 Cert-manager指定了自己的email
我www.namesilo.com申请了geekcloudnative.com域名，分别为二级域名 `harbor.geekcloudnative.com` 和 `notary.geekcloudnative.com` 配置 A 记录，并指向自己的ip
但我的kubectl get certificate -A 的READY一直是False，
NAMESPACE NAME READY SECRET AGE
harbor harbor-secret-tls False harbor-secret-tls 29s
harbor notary-secret-tls False notary-secret-tls 29s

kubectl get certificate -n harbor harbor-secret-tls -oyaml显示 message: Issuing certificate as Secret does not exist，

k logs cert-manager-5d595f9fdf-mnw9n -n cert-manager 发现 以下报错日志
error"="failed to perform self check GET request 'http://harbor.geekcloudnative.com/.well-known/acme-challenge/ldnFkzhB2euqIZdtrTo7-ryM492_HCrmpcnWOBH6TmI': Get \"http://harbor.geekcloudnative.com/.well-known/acme-challenge/ldnFkzhB2euqIZdtrTo7-ryM492_HCrmpcnWOBH6TmI\": dial tcp: lookup harbor.geekcloudnative.com on 172.16.253.166:53: no such host"

通过报错可以判断本地没有做域名解析。在/etc/hosts添加

<ip> harbor.geekcloudnative.com notary.geekcloudnative.com

READY为false得到解决

共 1 条评论 >

Jich

2023-01-29 来自上海

kubectl get certificate 我本地报 Issuing certificate as Secret does not exist

橙汁

2023-01-27 来自北京

在阿里云的k8s上使用cert-manager需要accesskey来验证域名归属和dns记录验证，文章里没看到 是域名的关系吗，后续跟课程也验证下