

Open Source AceCon

2021 智能云边开源峰会

AI x Cloud Native x Edge Computing

人工智能 × 云原生 × 边缘计算

Harbor企业级落地实践

张晨宇

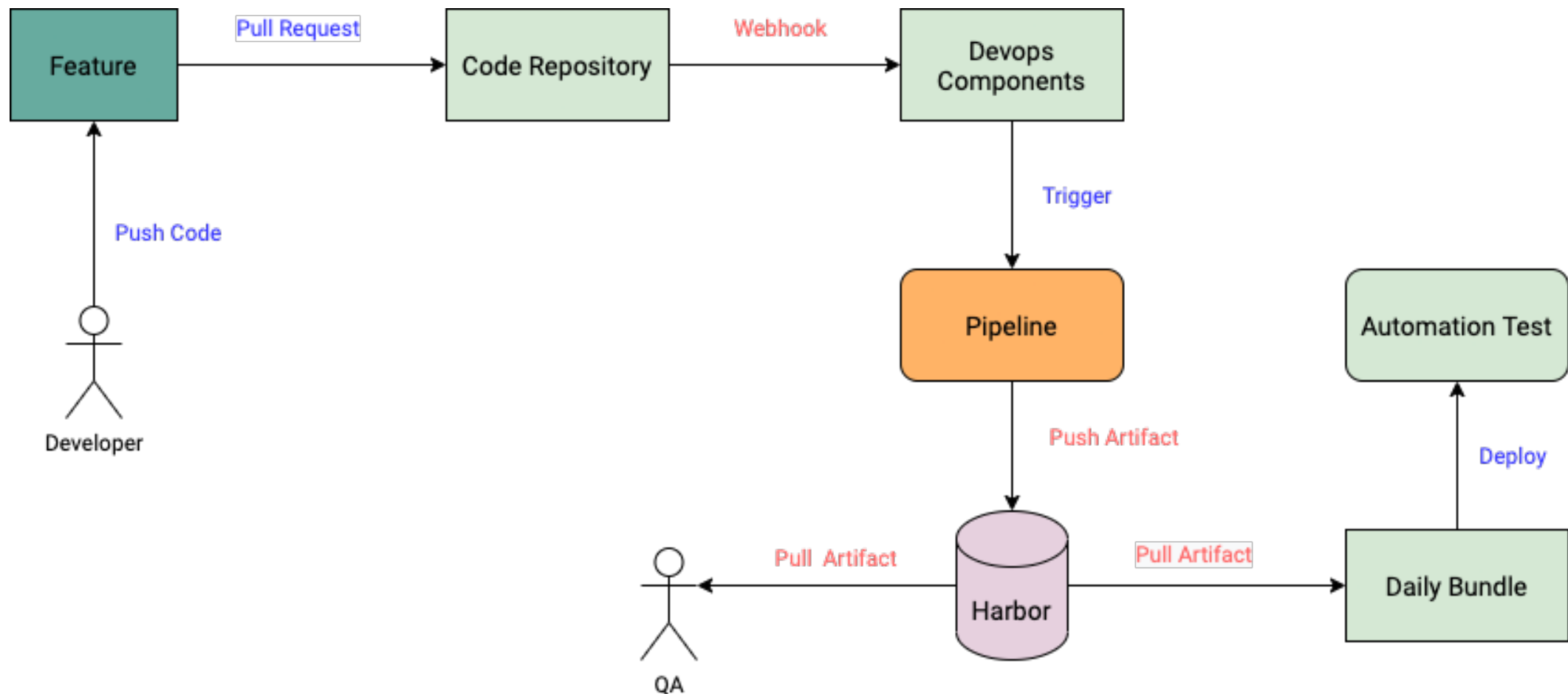
灵雀云工程师

大纲

- 背景介绍
- 内部演进
- 治理维护
- 产品融合
- 开源社区
- Harbor企业版

背景介绍

内部开发流程



背景介绍

数据规模

镜像仓库

500->1000->2000

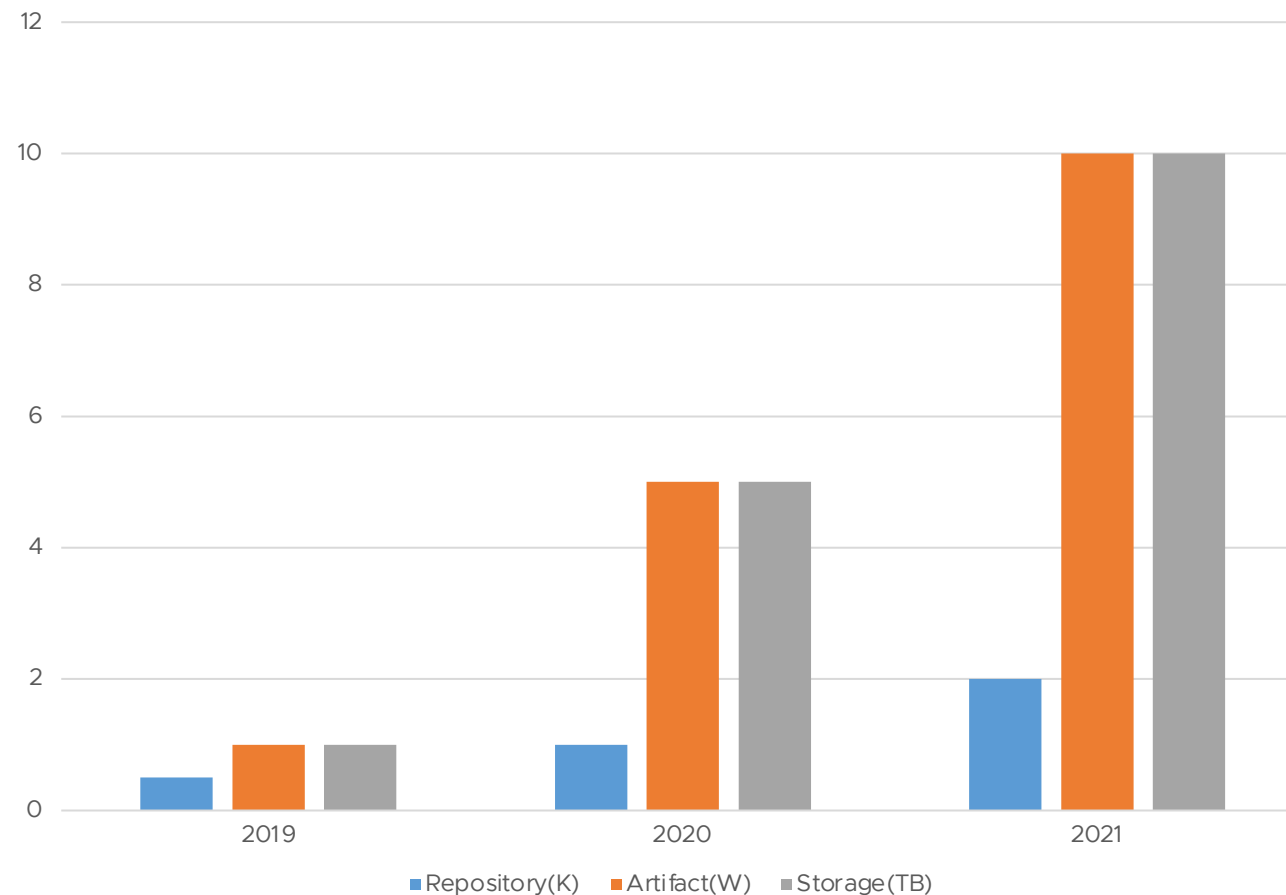
制品数量

10000->50000->100000

存储空间

1T -> 5T -> 10T

制品规模



背景介绍

数据规模

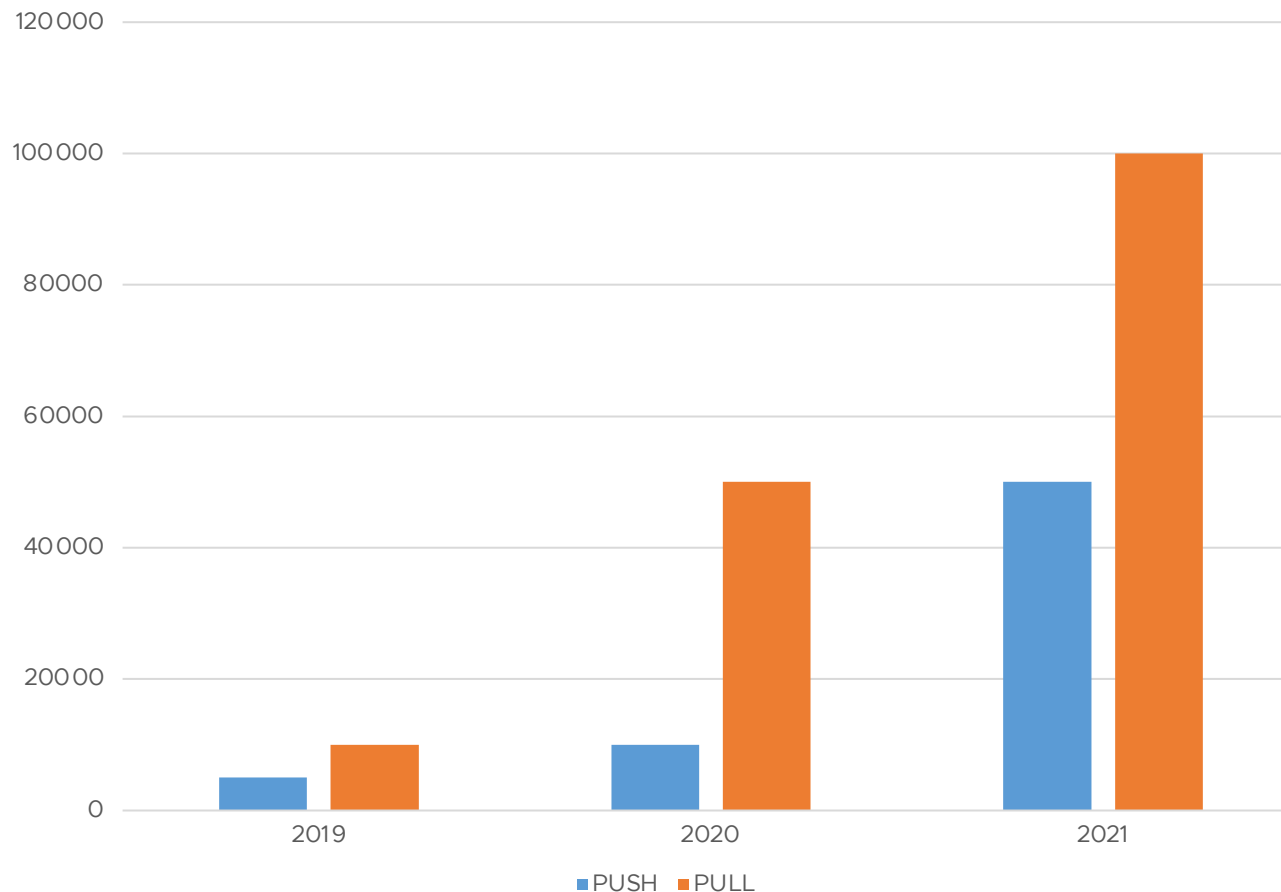
日均推送

5000->10000->50000

日均拉取

10000->50000->100000

请求规模



内部演进

harbor 1.x

业务组件Chart

多架构镜像需求

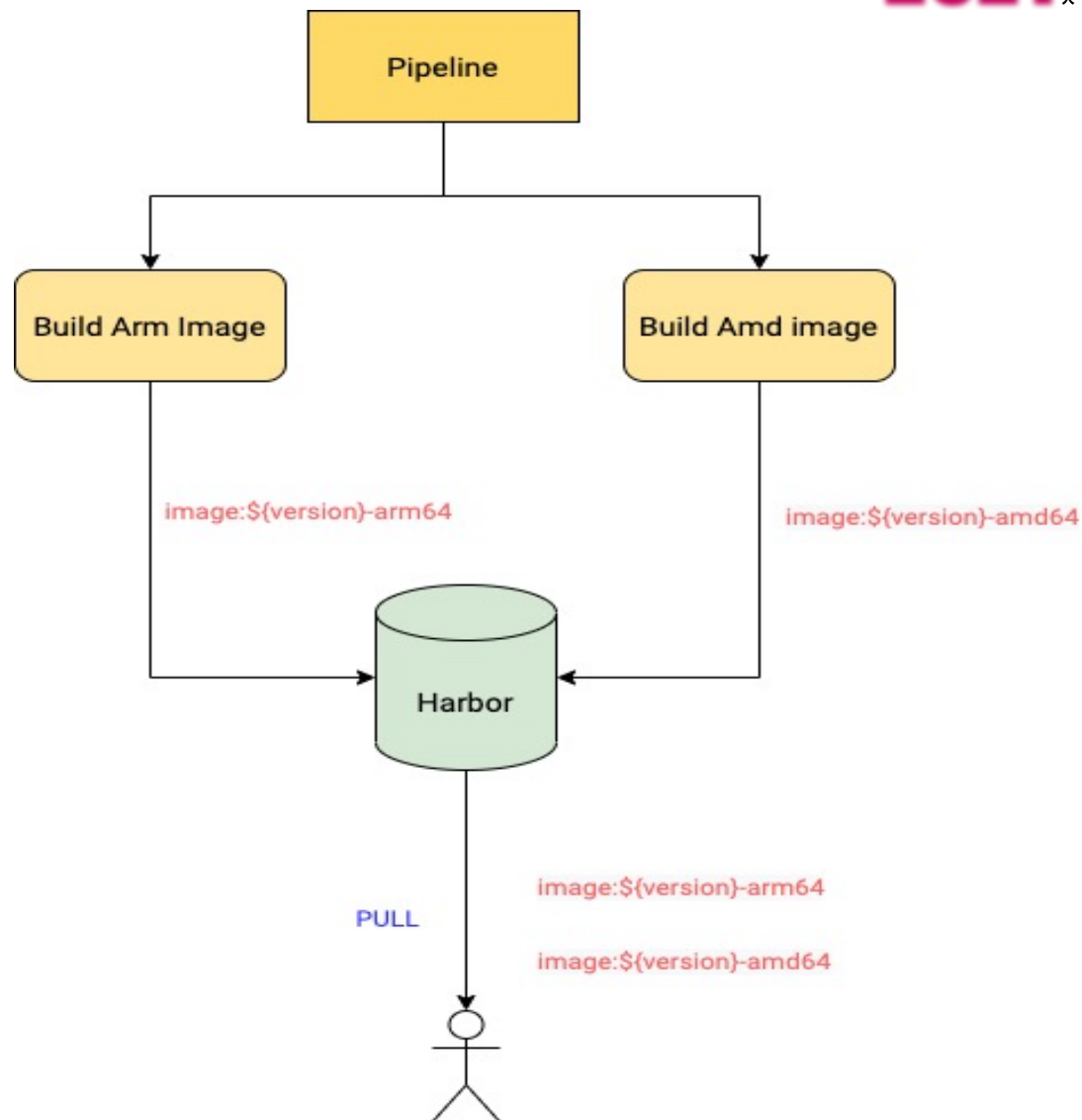
- AMD64
- ARM64

常规方式:

两种架构tag分别维护，如v1-amd64, v1-arm64

缺点:

- 需要维护两个架构版本的chart
- tag信息冗余
- 额外的tag维护成本



内部演进

harbor 1.x

harbor(amd) + arm harbor

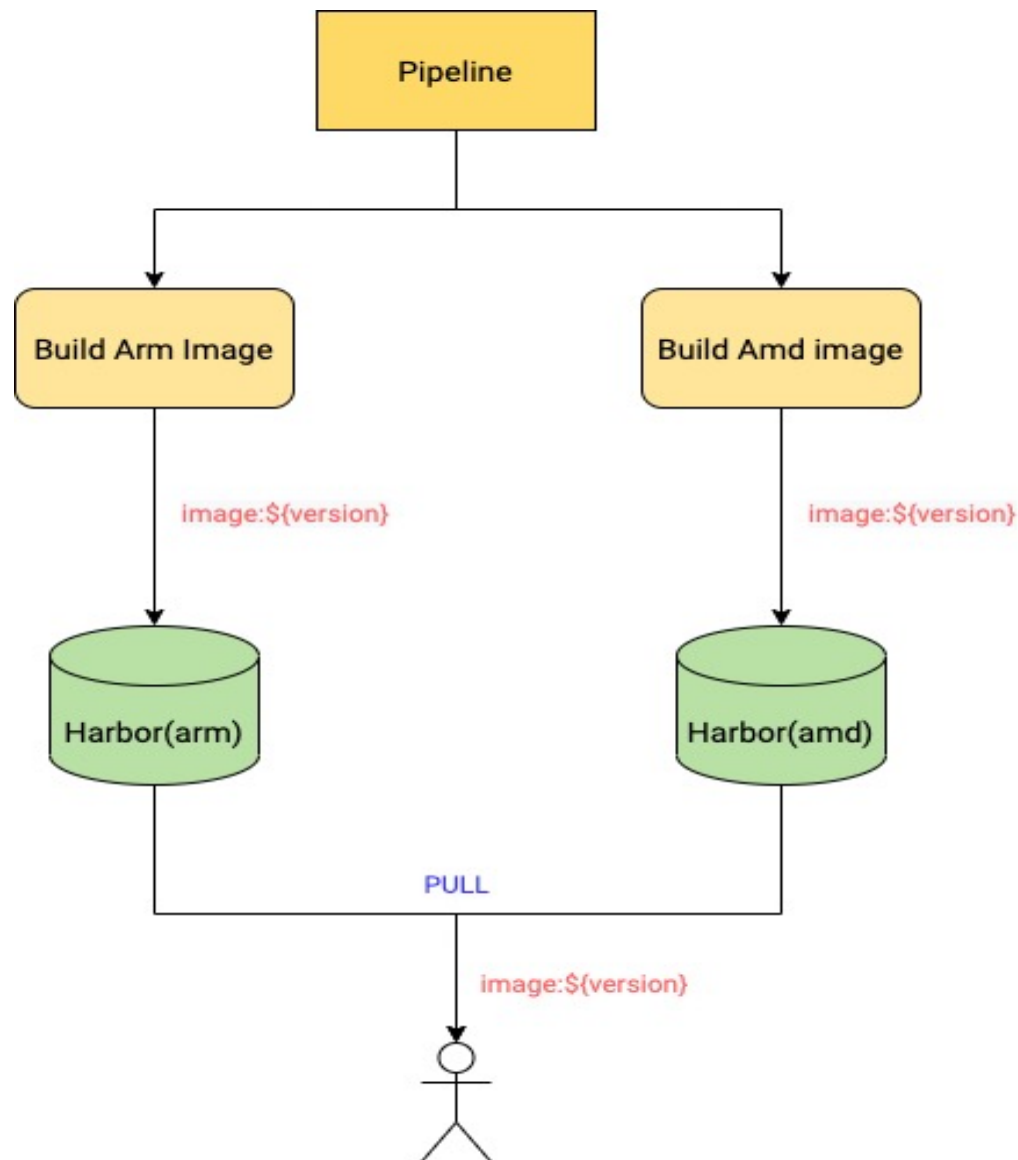
部署两套Harbor，维护两种架构的镜像

优点：

- 业务chart维护一套，通过指定registry拉取不同架构的镜像
- tag比较统一

缺点：

- 额外的harbor维护成本



内部演进

harbor 1.x -> harbor 2.x

Harbor 2.x 版本增加了对OCI制品的支持，可以通过同一个tag维护多架构的镜像。

OCI Image Index

application/vnd.oci.image.index.v1+json

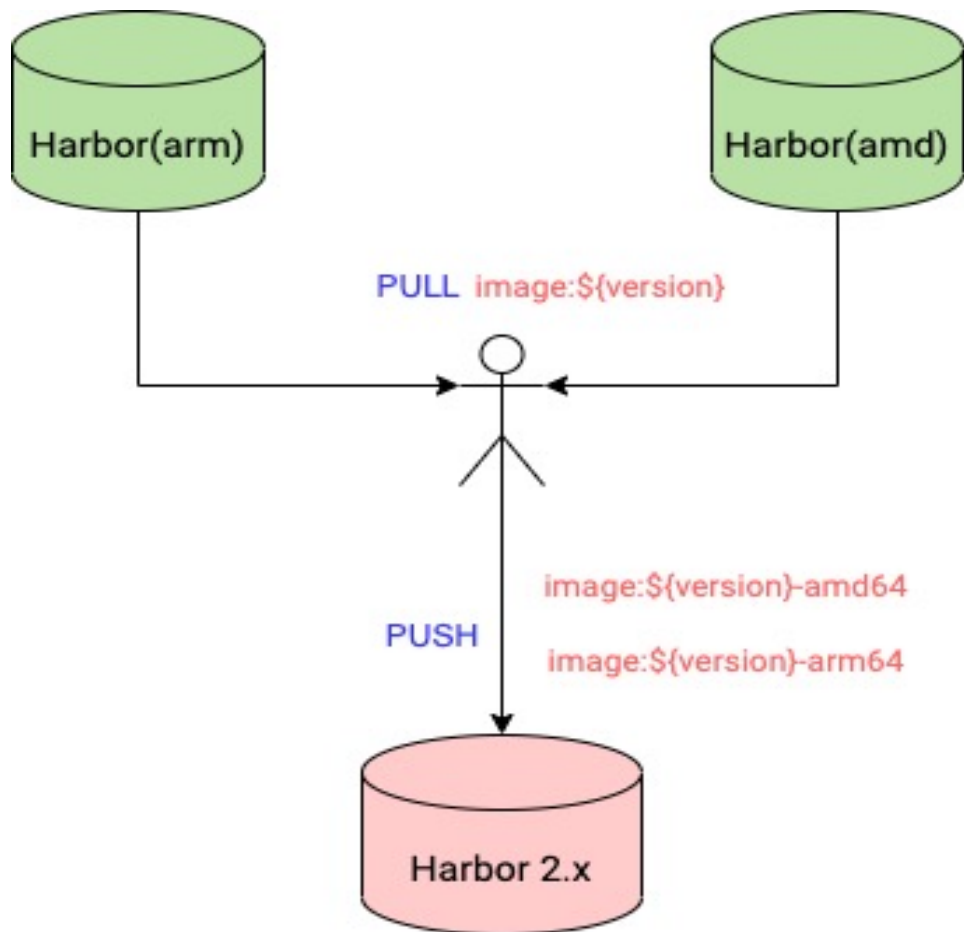
- schemaVersion int
- mediaType string
- manifests []manifest
- annotation map

```
{
  "schemaVersion": 2,
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "size": 7143,
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc331f",
      "platform": {
        "architecture": "arm64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "size": 7682,
      "digest": "sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a4333501270",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    }
  ],
  "annotations": {
    "com.example.key1": "value1",
    "com.example.key2": "value2"
  }
}
```


内部演进

harbor 1.x -> harbor 2.x

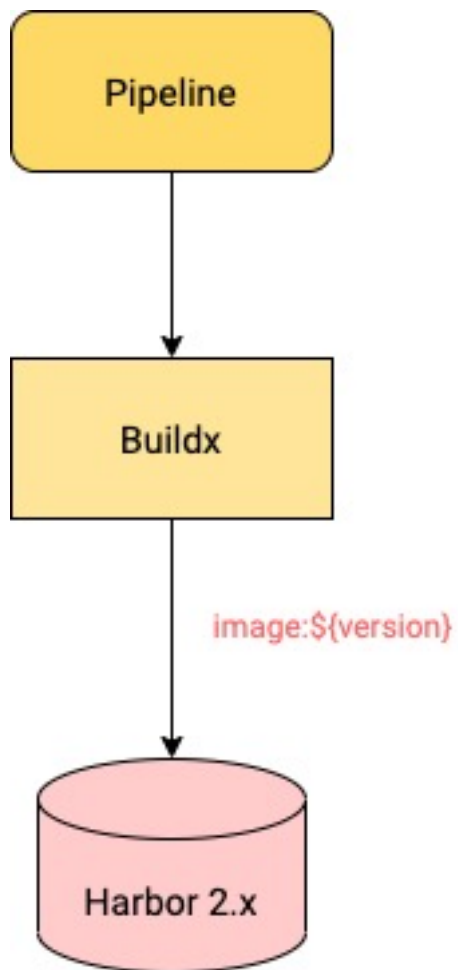
Multi arch images migrations



1. 将1.x版本的单架构镜像以对应架构作为后缀同步到2.x版本中。
(同步方式可选择使用docker cli, skopeo或harbor的镜像复制功能)
2. 创建镜像的manifest index, 并push到2.x版本。



```
# 创建多架构的manifest index
$ docker manifest create image:${version} -a image:${version}-amd64 -a image:${version}-arm64
# 推送该manifest index
$ docker manifest push image:${version}
```



通过Buildx构建多架构镜像

- QEMU
 - 配置简单方便
 - 单点构建，借助qemu模拟其他架构环境，构建速度慢于原生构建
- 异构集群(buildx kubernetes driver)
 - 多架构镜像通过原生平台构建，构建速度快
 - 需部署异构集群，配置及维护成本略高

```
# 开启buildx
$ export DOCKER_CLI_EXPERIMENTAL=enabled
# 构建多架构镜像
$ docker buildx build -t image:${version} --platform=linux/arm64,linux/amd64 . --push
```

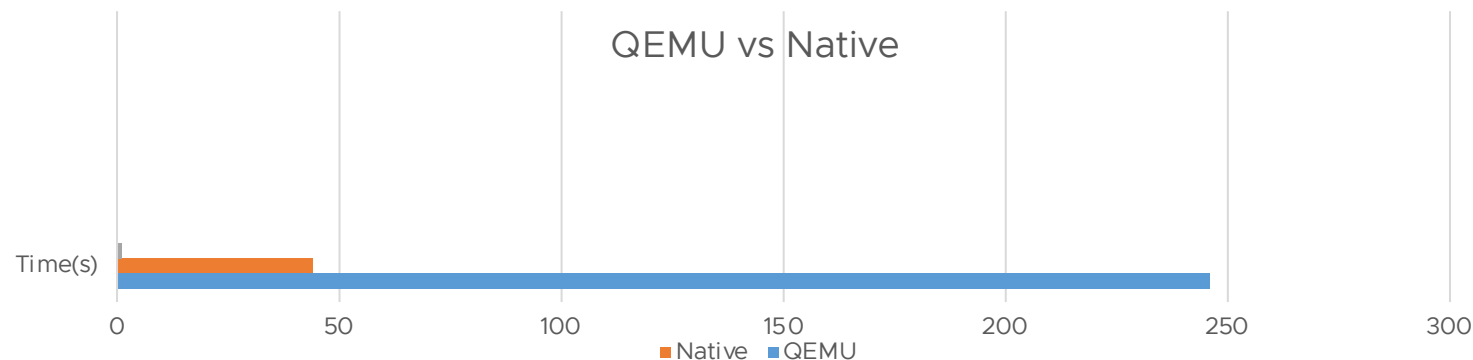
内部演进

harbor 2.x

构建耗时对比

QEMU 246s

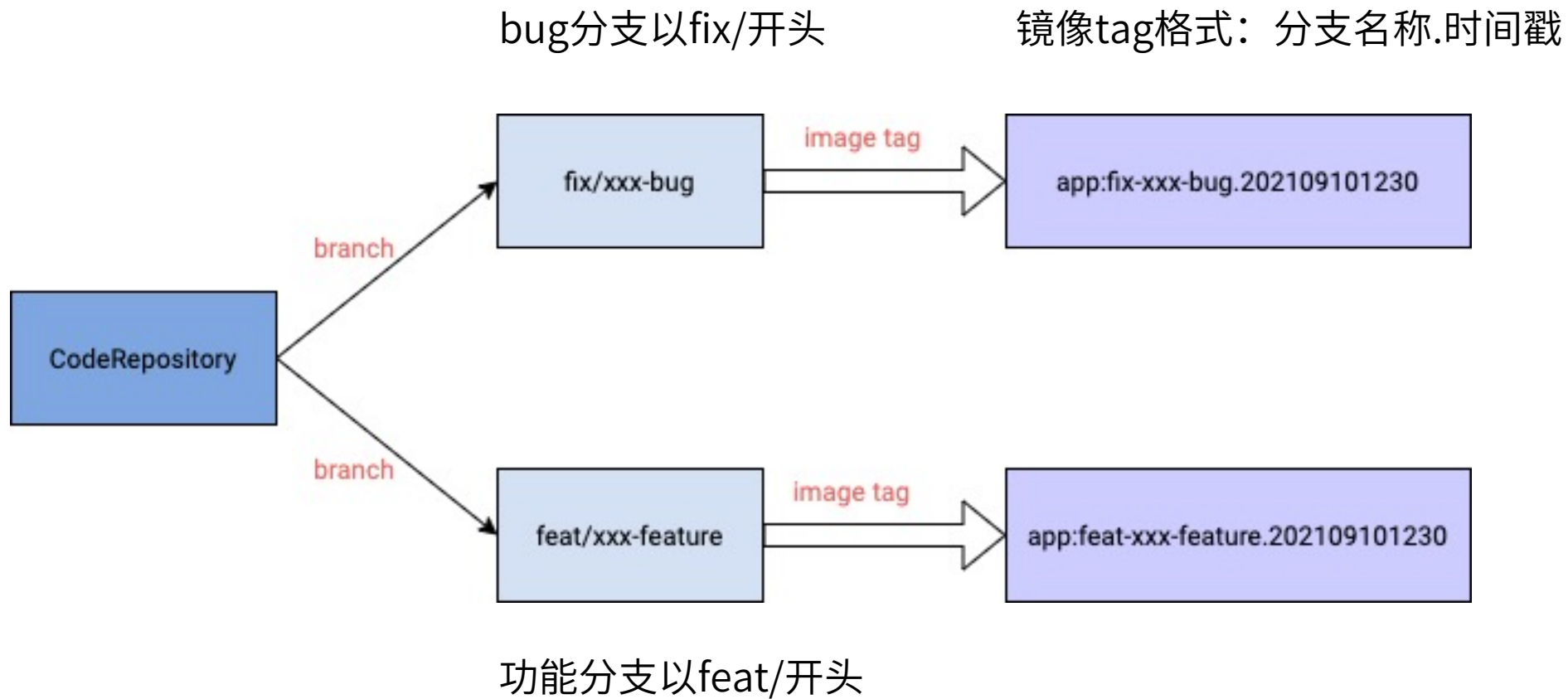
Native 44s



```
# syntax=docker/dockerfile:1
FROM golang:1.16 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go ./
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app ./
CMD [ "./app" ]
```

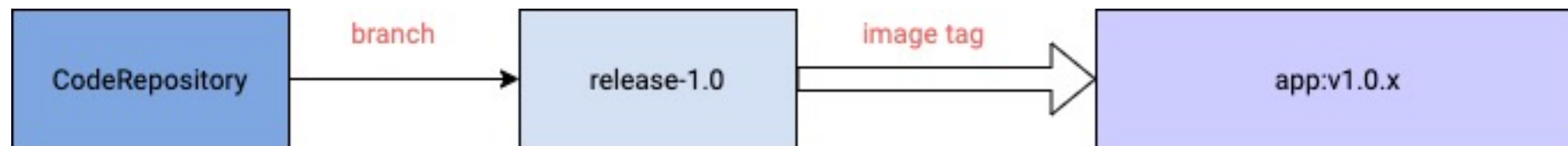
开发阶段



治理维护

命名规范

发版阶段



治理维护

命名规范

规范意义

- 语义化信息，便于溯源
- 统一管理，便于统一的清理维护策略
 - 永久保留release相关的代码分支和制品
 - 按需清理开发阶段相关的代码分支和制品

通过Harbor的Tag Retention实现镜像的清理。

清理周期：

- 定时执行
- 立即执行

清理策略

- 匹配(排除)Repo
- 匹配(排除)Tag
- 保留规则
 - 最近推送的#个
 - 最近拉取的#个
 - 最近#天被推送
 - 最近#天被拉取
 - 全部保留

治理维护

清理策略

保留策略可配置多条，且这些策略之间的关系为 “或”

示例：

添加 Tag 保留规则

为当前项目指定 tag 保留规则。所有 tag 保留规则独立计算并且适用于所有符合条件的仓库。

应用到仓库

匹配

▼

**

使用逗号分隔repos,repo*和**

以 artifact 数量或天数为条件

保留全部 artifacts

▼

Tags

匹配

▼

v1.*|

无 Tag 的 Artifacts

☐

输入多个逗号分隔的 Tags, Tag*或**。可通过勾选将未加 Tag 的 artifacts 作为此策略的一部分。

取消

添加

匹配项目下所有仓库

保留所有tag为v1.**制品

人工智能x 云原生分论坛

16

治理维护

清理策略

示例：

添加 Tag 保留规则

为当前项目指定 tag 保留规则。所有 tag 保留规则独立计算并且适用于所有符合条件的仓库。

应用到仓库

匹配

▼

**

使用逗号分隔repos,repo*和**

以 artifact 数量或天数为条件

保留最近#天被推送过的 artifacts

▼

天数

7

Tags

匹配

▼

fix-**|

无 Tag 的 Artifacts ☐

输入多个逗号分隔的 Tags, Tag*或**。可通过勾选将未加 Tag 的 artifacts 作为此策略的一部分。

取消

添加

匹配项目下所有仓库

保留最近一周内推送的所有tag为fix-**制品

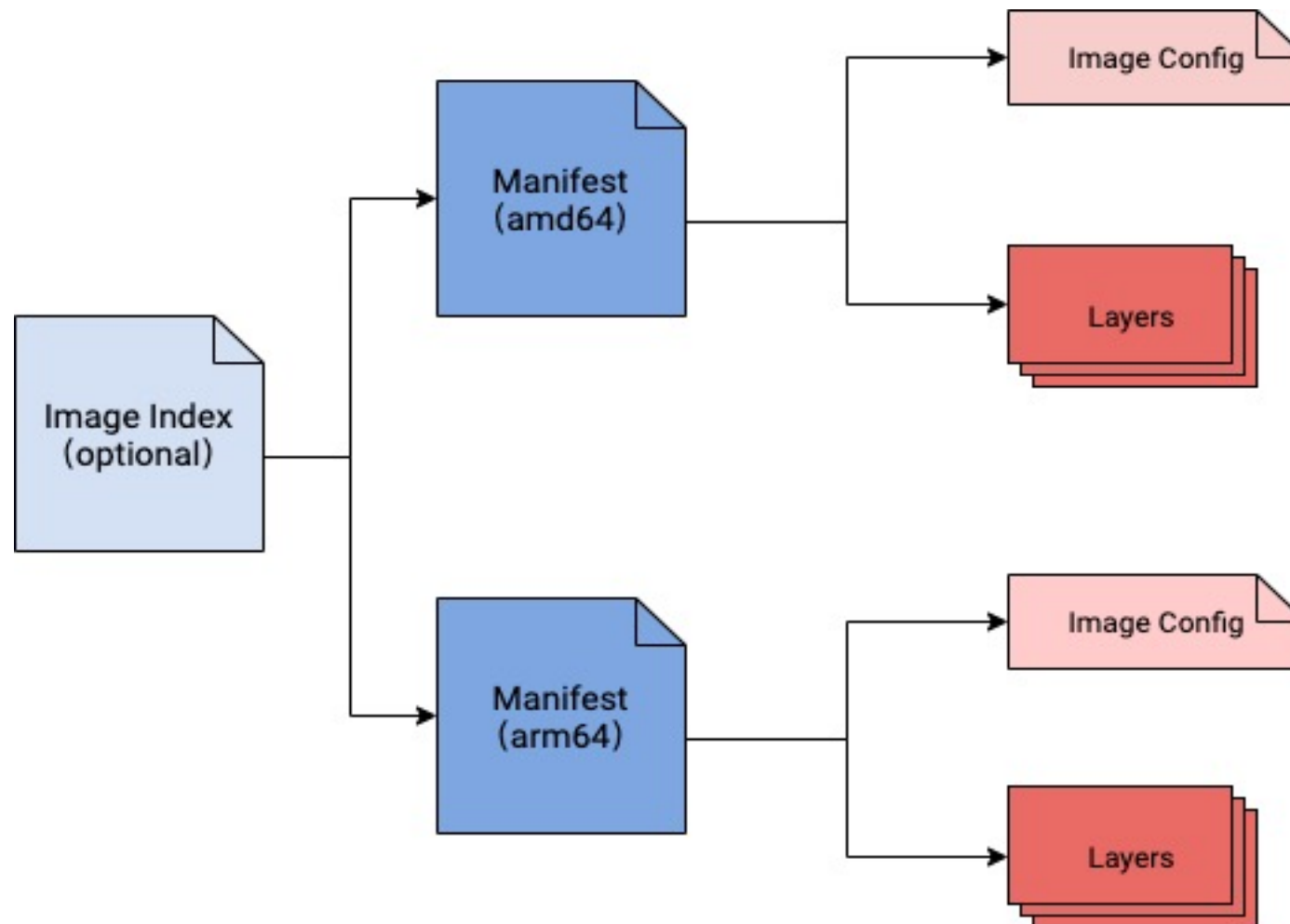
人工智能x 云原生分论坛

17

治理维护

垃圾清理

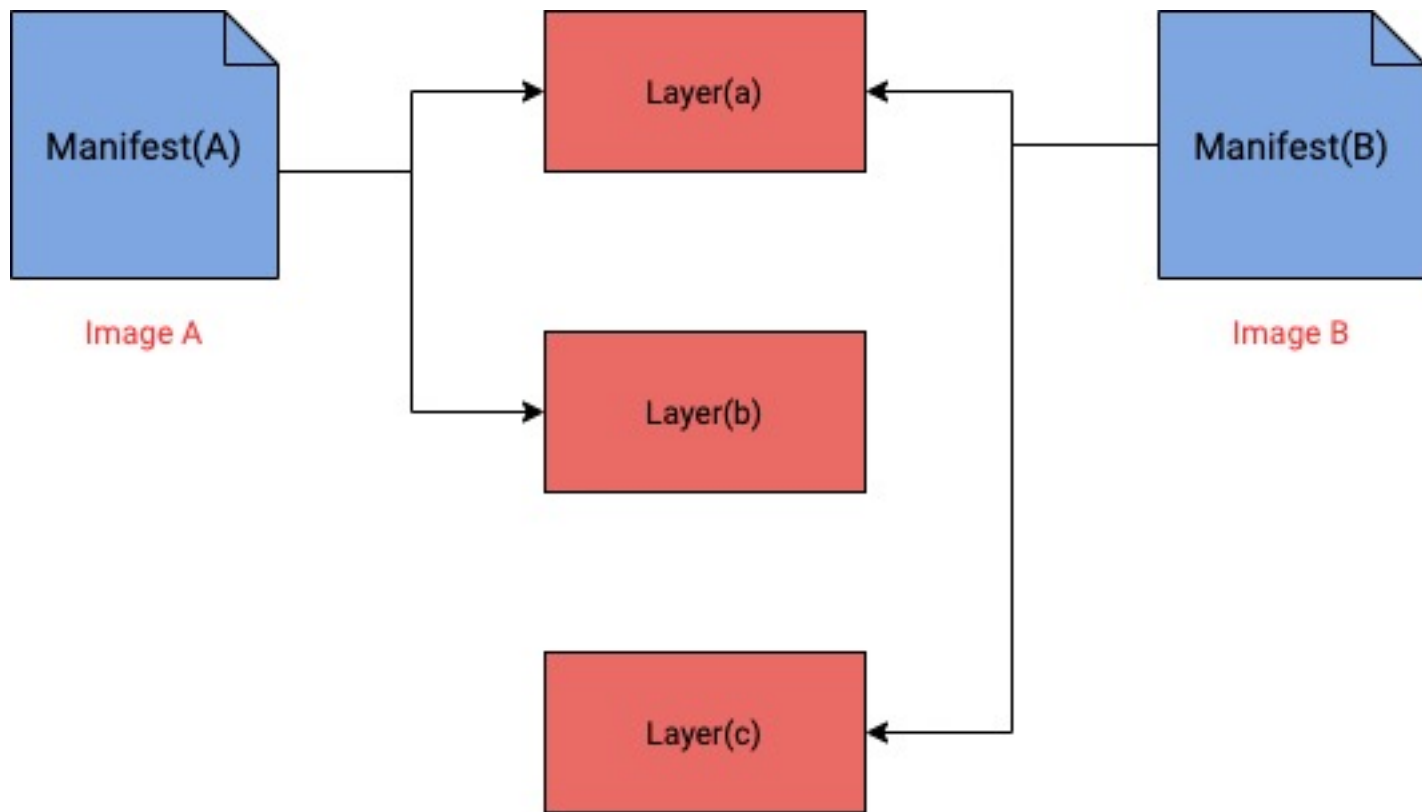
镜像结构



治理维护

垃圾清理

思考：为什么需要GC?

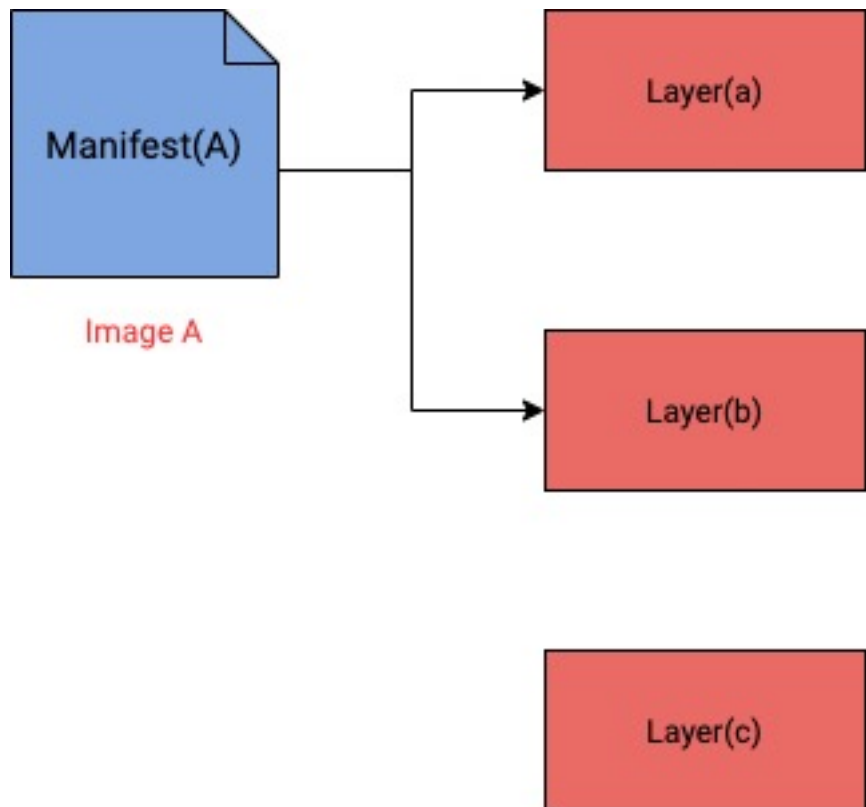


镜像A依赖层a和层b
镜像B依赖层a和层c
镜像A和镜像B都依赖层a

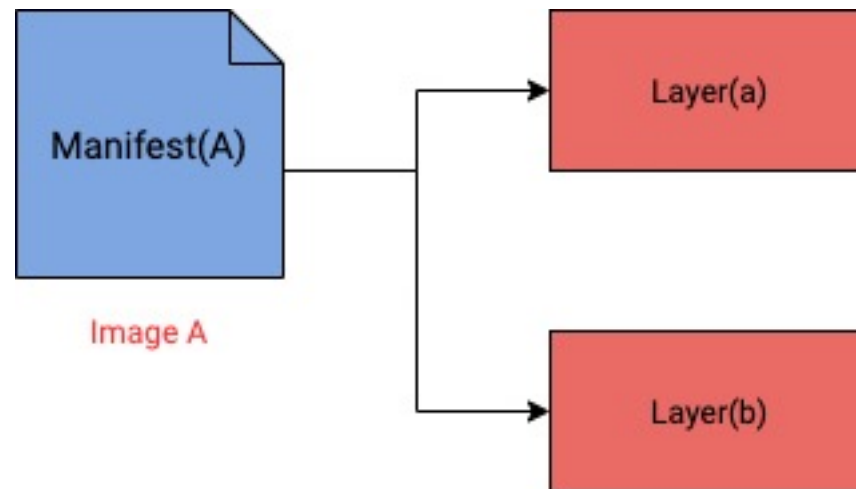
治理维护

垃圾清理

删除镜像B后：



执行GC后：



产品融合

开箱即用的部署体验

DevOps 工具链 / 快速部署并集成

快速部署并集成

表单YAML

 Harbor Registry
制品仓库

重新选择工具

部署位置

* 集群:

1.建议部署在 DevOps 工具专用的集群中。
2.不建议部署在global集群中, 因为此工具在高并发执行时会消耗大量资源, 这可能会影响整个平台的使用。

* 命名空间:

架构

高可用:

☐

开启“高可用”时, 需要确保事先在同集群和命名空间下部署了“Postgres”操作器 (postgres-operator) 。
暂不支持“高可用实例”与“非高可用实例”的互相转换。实例创建后, 当前高可用设置不可修改。

资源

* 规模:

推荐配置
适合并发要求高

最低配置
适合基础使用

自定义
适合专业用户配置

?

* 资源请求:

* CPU

4

核

* 内存

8

Gi

* 资源限制:

* CPU

4

核

* 内存

8

Gi

配置基本信息

产品融合

开箱即用的部署体验

DevOps 工具链 / 快速部署并集成

网络

方式: PVC 存储 本地存储

* PVC 名称: 创建 PVC

网络

方式: Ingress Node Port

* 域名:

协议: HTTP HTTPS

* 证书: 创建证书

服务访问地址:

账号

用户名: admin

* 初始密码: ☒ 自定义 ☐ 随机


必须包含字母、数字、特殊符号中的两种，特殊符号支持!@#%&*(), 长度6~20位

部署并集成 取消

产品融合

开箱即用的部署体验

DevOps 工具链 / high / high-zxnv8

 high-zxnv8 操作 ▾

基本信息

部署状态: 🔄 部署中 🔍 产品入口: [Harbor 工具](#) 🔗

部署时间: 2021-09-09

部署位置

集群: global 命名空间: devops

架构

高可用: 已开启

资源

资源请求: 🖥️ 4 核 💾 8 Gi 资源限制: 🖥️ 4 核 💾 8 Gi

存储

方式: LocalPath 节点名称: 192.168.128.40

存储路径: /tmp/high

网络

方式: NodePort HTTP 端口: 30111

Notary 端口: 30113

查看部署状态，部署信息或更新实例

产品融合

租户绑定分配

项目 / devops / 绑定镜像仓库 (harbor3)

绑定账号

2 分配仓库

✓ 账号绑定成功，请为项目分配镜像仓库

未分配仓库

已分配仓库

▼ 库 /

> 库 devops

▼ 库 library


↕

library/busybox

分配仓库

取消


1. 绑定Harbor给项目
2. 分配镜像仓库

 harbor3

操作 ▾

类型：制品仓库 / Harbor Registry

集成时间：2021-09-02 10:25:38

访问地址：<http://192.168.128.40:30222> 

API 地址：<http://192.168.128.40:30222>

实例：[harbor3-f4pxh](#)

项目绑定


绑定

名称	项目	凭据	绑定时间	
harbor	devops	harbor	2021-09-03 17:40:25	⋮

查看绑定信息

产品融合

租户绑定分配

 harbor

集成名称: harbor3

凭据: harbor

类型: 制品仓库 / Harbor Registry

绑定时间: 2021-09-03 17:40:25

描述: -


制品仓库

分配仓库

按仓库地址过滤

基于产品性能考虑，数据获取可能会有短时间的延迟

仓库地址

 192.168.128.40:30222/library/busybox

192.168.128.40:30222

查看分配的镜像仓库

更新全局绑定策略

×

⚠ 请谨慎更改凭据！更改凭据可能会影响下面已选绑定策略的最终使用效果。

认证

认证方式：☒ 用户名/密码 ☐ 无认证

凭据：

绑定策略

🌐 同名项目自动绑定

☐

- 1、镜像仓库会自动创建和 DevOps 平台项目同名的项目，若镜像仓库中同名项目已存在则不会重复创建，项目数量过多时可能会发生几分钟的延迟
- 2、自动将镜像仓库项目与 DevOps 平台的同名项目进行绑定，并建立分配关系供 DevOps 平台同名项目独立使用
- 3、DevOps 平台新建项目时也会触发上述动作

🌐 公共镜像仓库

☐

- 1、将所选镜像仓库项目作为“公共镜像仓库”与 DevOps 平台的所有项目进行绑定，并建立分配关系供 DevOps 平台所有项目共同使用
- 2、DevOps 平台新建项目时也会触发上述动作

Harbor项目：

删除全局绑定策略

完成配置

取消

在Harbor上创建同名项目，实现自动绑定

将Harbor上的仓库作为公共仓库共享给所有项目使用

创建镜像清理策略



* 名称: 以 a-z, 0-9 开头结尾, 支持使用 a-z, 0-9, -, 最多 36 个字符

* 保留时间:

天

保留所填写天数的镜像, 如填写N, 则保留最近N天的镜像, N天前的镜像将会被清理

* TAG 匹配规则:

符合规则的镜像将会被清理, 多个筛选条件是“或”的关系

* 适用仓库:

选择镜像仓库, 清理规则将对选择的镜像仓库生效

清理时间规则:

快捷选择

自定义选择

* 清理时间: ☒ 星期一 ☒ 星期二 ☒ 星期三 ☒ 星期四 ☒ 星期五 ☒ 星期六 ☒ 星期日

⌚ 00:00 x

+ 配置时间

创建

取消

创建镜像保护策略



* 名称: 以 a-z, 0-9 开头结尾, 支持使用 a-z, 0-9, -, 最多 36 个字符

* TAG 匹配规则:

符合规则的镜像将被设置保护, 不会被清理, 多个筛选条件是“或”的关系

* 适用仓库:

选择镜像仓库, 保护策略将对选择的镜像仓库生效

创建

取消

开源社区

Multi Architecture WorkGroup(多架构工作组)

<https://github.com/goharbor/community/tree/master/workgroups/wg-multiarch>

背景：

此前Harbor社区只提供了标准的x86架构的组件镜像，对于其他架构的环境无法部署harbor，且社区有很多用户有其他架构支持的需求。

目标：

致力于以统一通用的方式持续构建多架构的Harbor镜像，如arm，龙芯等。

开源社区

Multi Architecture WorkGroup(多架构工作组)

成果：

- <https://github.com/goharbor/harbor-arm> (ARM架构，开发中，预计harbor 2.4提供)
- <https://github.com/goharbor/harbor-loongson> (国产化龙芯架构，开发中)

加入：

Slack: # harbor-multi-arch-workgroup

开源社区

Performance WorkGroup(性能工作组)

<https://github.com/goharbor/community/tree/master/workgroups/wg-performance>

背景：

Harbor在一些数据规模场景下，会存在一些性能问题，此前也没有相关的性能测试工具和官方的性能测试报告，需要长期专注于解决性能问题和性能优化领域的贡献者。

目标：

致力于提高Harbor在大数据集或高并发规模下的稳定性和性能，建设统一的性能测试工具，提供标准的性能测试报告等。

成果：

- 性能测试工具集：<https://github.com/goharbor/perf>
 - 准备测试数据
 - 执行并发测试(参数配置化)
 - 自动生成测试报告
 - harbor2.3性能测试报告：<https://github.com/goharbor/perf/wiki/Harbor-2.3.0-Performance-Test-Reports>
- 性能问题修复
 - <https://github.com/goharbor/harbor/issues/14819>
 - <https://github.com/goharbor/harbor/issues/14815>
 - <https://github.com/goharbor/harbor/issues/14814>
 - <https://github.com/goharbor/harbor/issues/14813>
 - <https://github.com/goharbor/harbor/issues/14716>

- 性能提升

concurrency 300
artifacts 10w

API	version < 2.3 (p95)	version = 2.3 (p95)	improvement
Catalog	15s	5s	x3
Projects	8s	2s	x4
Artifacts	46s	3s	x15
Tags	4s	1s	x4
Audit logs	40s	4s	x10

Harbor企业版

Alauda Registry Service for Harbor

背景：

近年来，灵雀云已服务了数百家来自政府、金融、能源、制造、地产、交通等领域的头部客户，Harbor作为ACP平台中默认且推荐的镜像仓库，为其中超过100家客户部署了Harbor开源版，同时提供了高质量的运维和兜底服务，获得了客户的高度评价。灵雀云愿意基于自身丰富的Harbor运维经验帮助更多Harbor企业客户解决后顾之忧，让客户获得稳定、可靠、性能优异的制品库产品使用体验。

Harbor企业版

Alauda Registry Service for Harbor

企业版服务内容：

- 1、Harbor云原生制品仓库的高可用解决方案；
 - a) 企业级高可用解决方案；
 - b) 基于Kubernetes Operator的高可用解决方案；
- 2、Harbor云原生制品仓库的版本升级、漏洞修复；
- 3、Harbor云原生制品仓库运行环境的运维支持服务，包括：镜像复制迁移、故障恢复、环境部署等；
- 4、DevOps工具链解决方案。

Harbor企业版

Alauda Registry Service for Harbor

咨询方式

电话：4006-252-832

微信：



York



扫一扫上面的二维码图案，加我微信

Open Source AceCon

2021 智能云边开源峰会

AI x Cloud Native x Edge Computing

人工智能 × 云原生 × 边缘计算

Thank You