基于drone和openshift的CI/CD实践

孟静

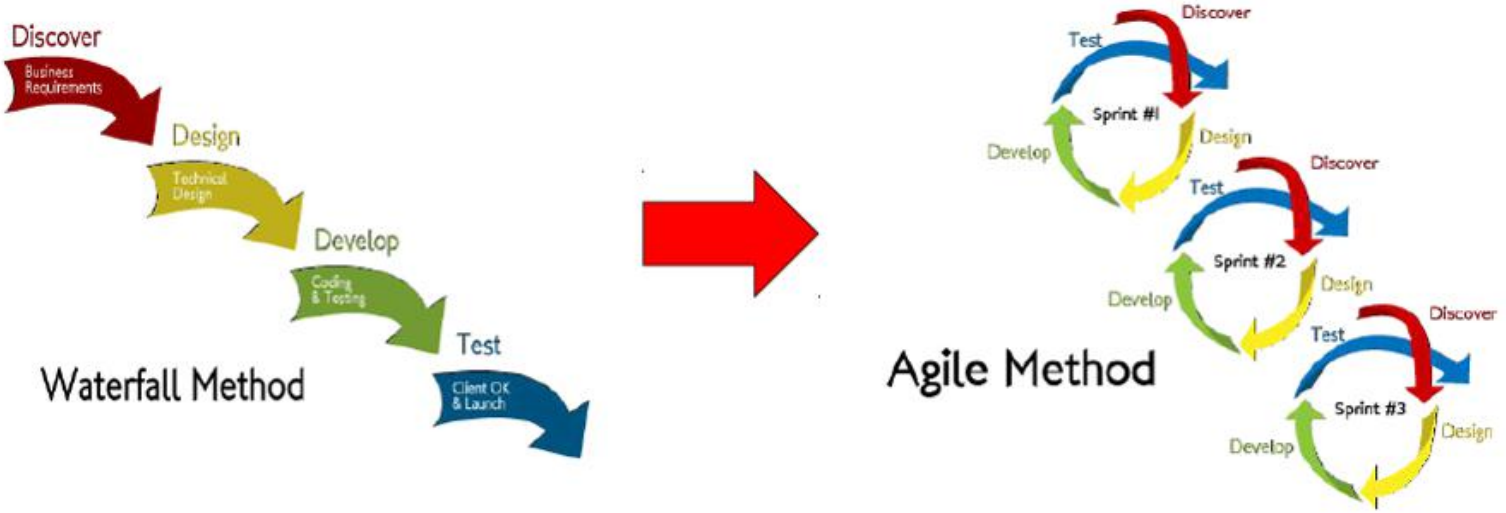
提纲

- CI/CD介绍
- 基于drone的CI/CD
- 私有镜像仓库的搭建

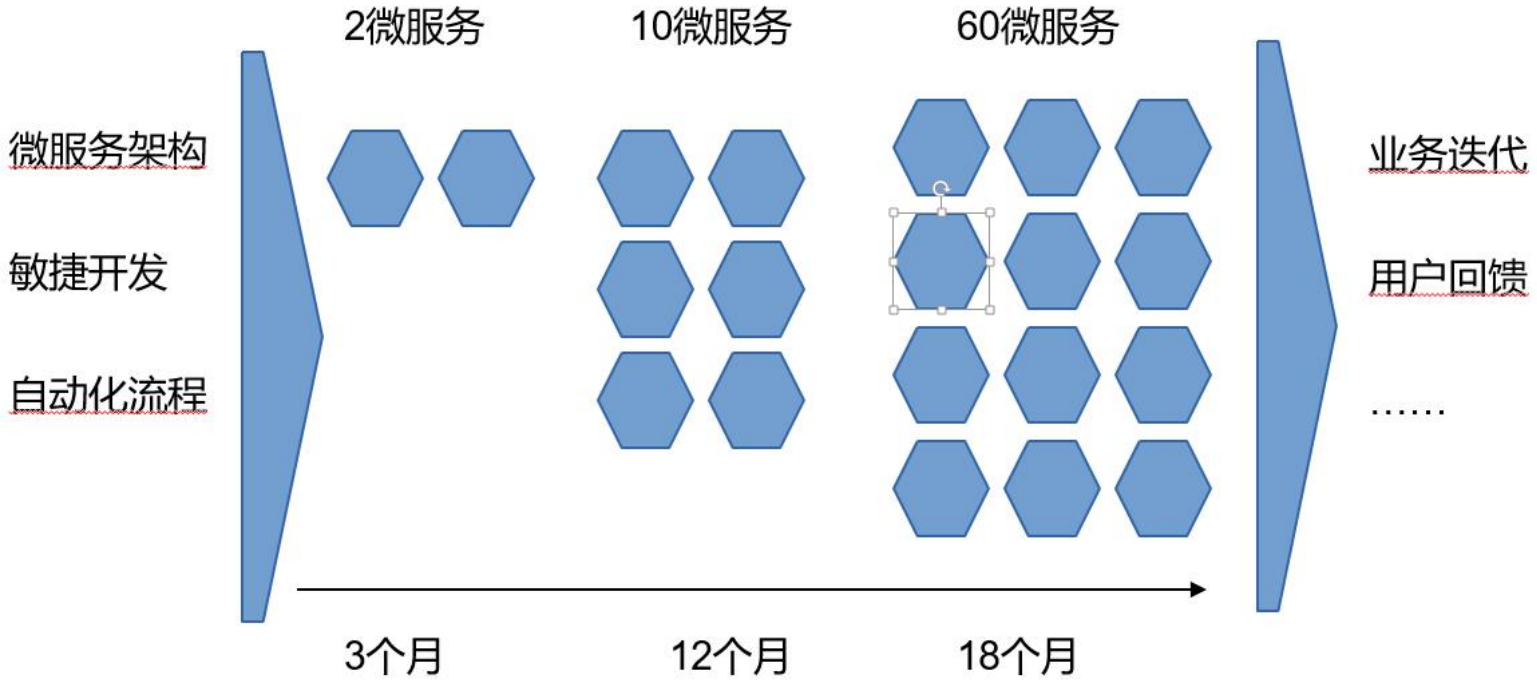
•CI/CD介绍

- 基于drone的CI/CD
- 私有镜像仓库的搭建

敏捷开发


Discover
Business Requirements
Design
Technical Design
Develop
Coding & Testing
Test
Client OK & Launch
Waterfall Method
Agile Method
Discover
Test
Sprint #1
Develop
Design
Discover
Test
Sprint #2
Develop
Design
Discover
Test
Sprint #3
Develop
Design

CI/CD的历史背景

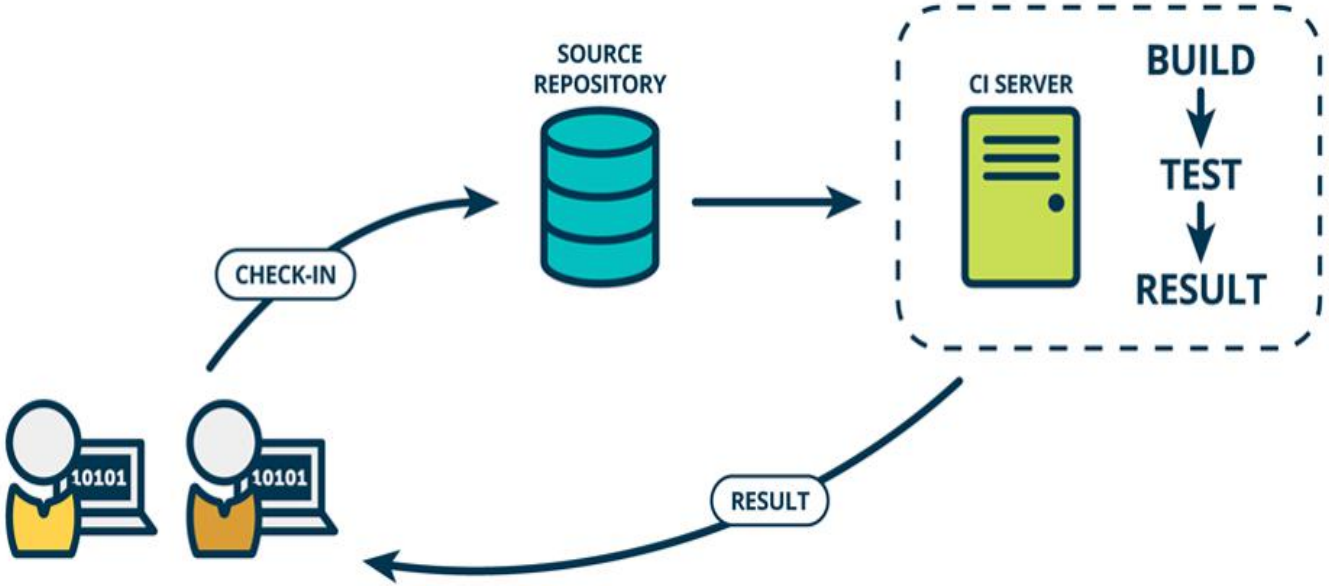


什么是持续集成（CI）

根据敏捷大师Martin Fowler的定义，“持续集成是一种软件开发实践。在持续集成中，团队成员频繁集成他们的工作成果，一般每人每天至少集成一次，也可以多次。每次集成会经过自动构建（包括自动测试）的检验，以尽快发现集成错误。”

可以说，持续集成是敏捷编程的重要实践基础，没有持续集成，所谓的敏捷编程便缺了最重要的一条腿，而基本不可能实现。

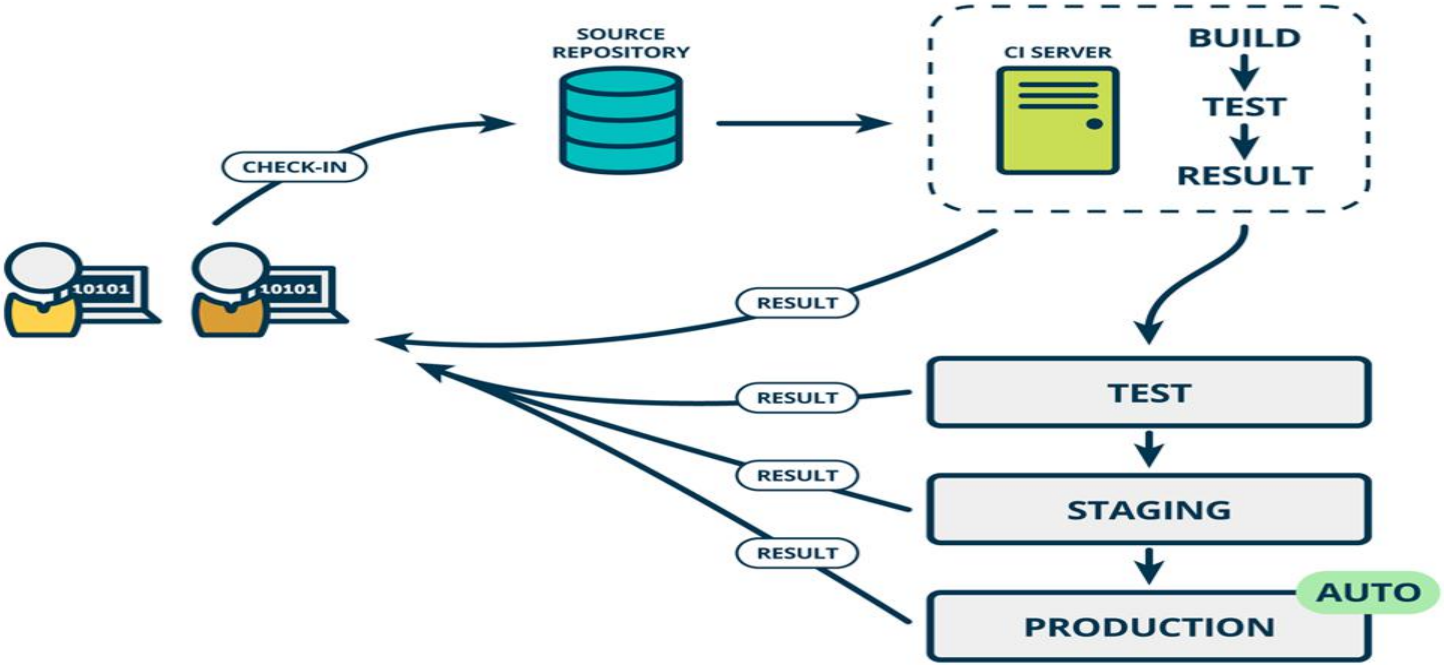
持续集成的工作流


SOURCE
REPOSITORY
CI SERVER
BUILD
TEST
RESULT
CHECK-IN
10101
10101
RESULT

什么是持续部署（CD）

持续部署指的是代码通过评审以后，自动部署到生产环境

持续部署（CD）的工作流


SOURCE
REPOSITORY
CI SERVER
BUILD
TEST
RESULT
CHECK-IN
10101
10101
RESULT
RESULT
RESULT
RESULT
TEST
STAGING
PRODUCTION
AUTO

CI/CD的好处

1.减少风险,尽早发现缺陷并修复缺陷；

2.减少重复的过程，通过减少重复性的动作来节省时间，成本，提高效率；

3.使得项目更加透明。

CI/CD工具

- Jenkins
- Travis CI
- Gitlab CI
- Drone
- Codeship
- CircleCI
- Shippable
- ……

本质都是探测代码库中代码的变更，然后触发一系列的任务或者工作

1.与多种代码库的连接；

2.安装和维护是否耗时；

3.构建环境；

4.构建后的步骤；

5.部署；

6.反馈清晰的执行日志；

7.插件维护难易程度。

提纲

- CI/CD介绍
- **基于drone的CI/CD**
- 私有镜像仓库的搭建

1.自动化测试和构建；

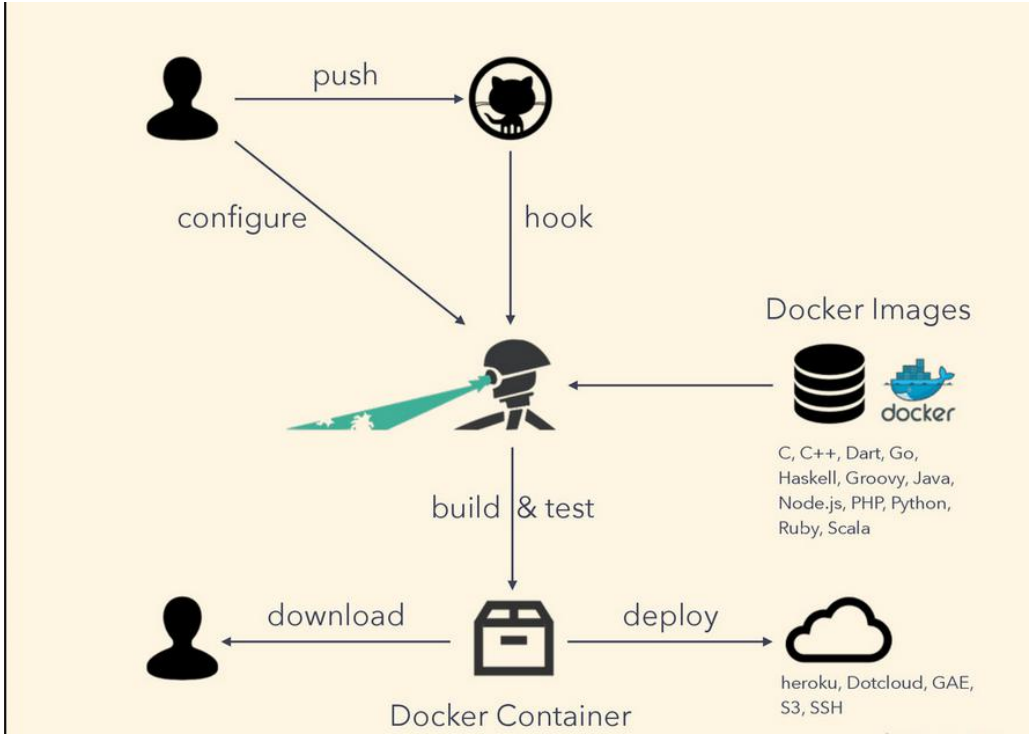
2.所有构建与测试都运行在docker中；

3.安装方便；

4.支持github， gitlab和bitbucket多种代码库；

5.所有插件都以docker方式运行，不需要手工安装和升级。

How it works


push
configure
hook
Docker Images
docker
C, C++, Dart, Go,
Haskell, Groovy, Java,
Node.js, PHP, Python,
Ruby, Scala
build & test
download
deploy
heroku, Dotcloud, GAE,
S3, SSH
Docker Container

.drone.yml


```
build:
    image: golang
    commands:
        - go get
        - go build
        - go test
compose:
    database:
        image: postgres
        environment:
            - POSTGRES_USER=postgres
            - POSTGRES_PASSWORD=mysecretpassword
publish:
    docker:
        username: octocat
        password: password
        email: octocat@github.com
        repo: octocat/hello-world
```

1.每个插件都跑在单独的容器中；

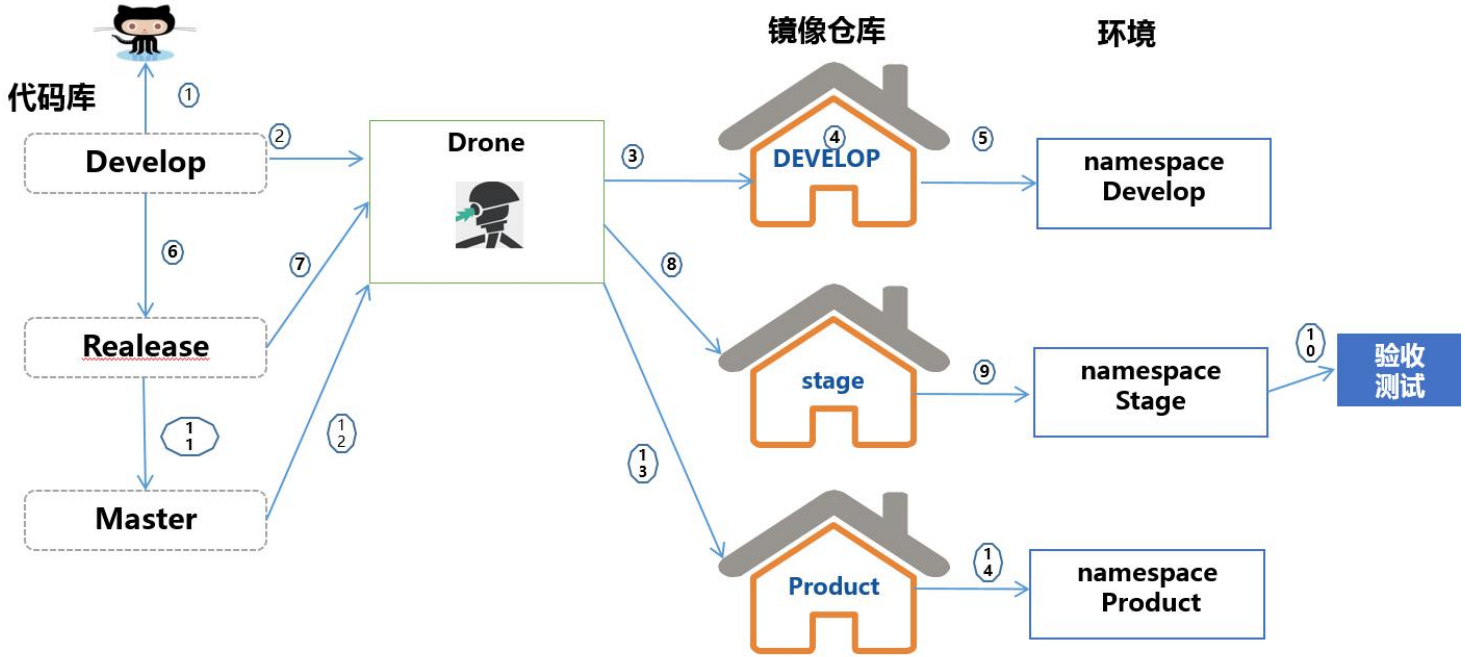
2.大部分使用go语言编写；

3.官方有大约54个plugin；

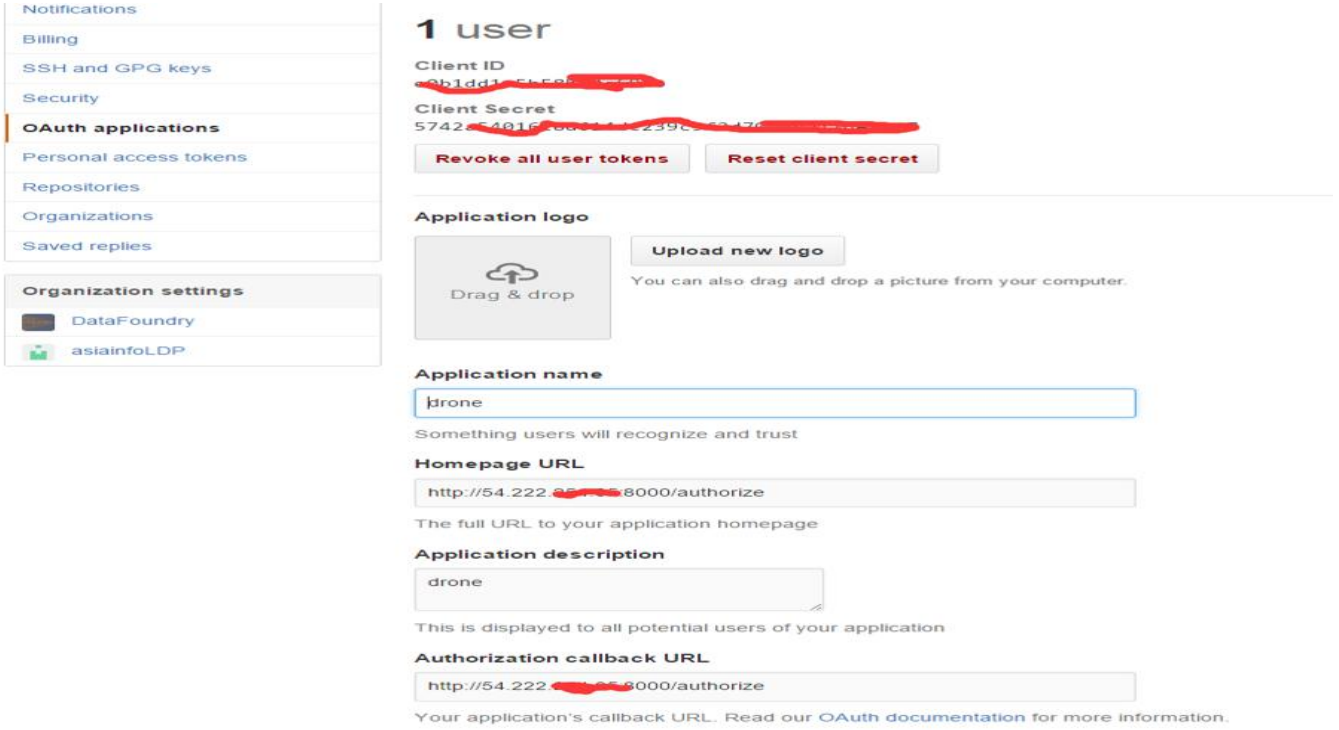
4.每个插件负责一件事情，职责明确；

5.自动更新，自动下载。

LDP云平台CI/CD架构



为drone配置oauth:



搭建drone服务器

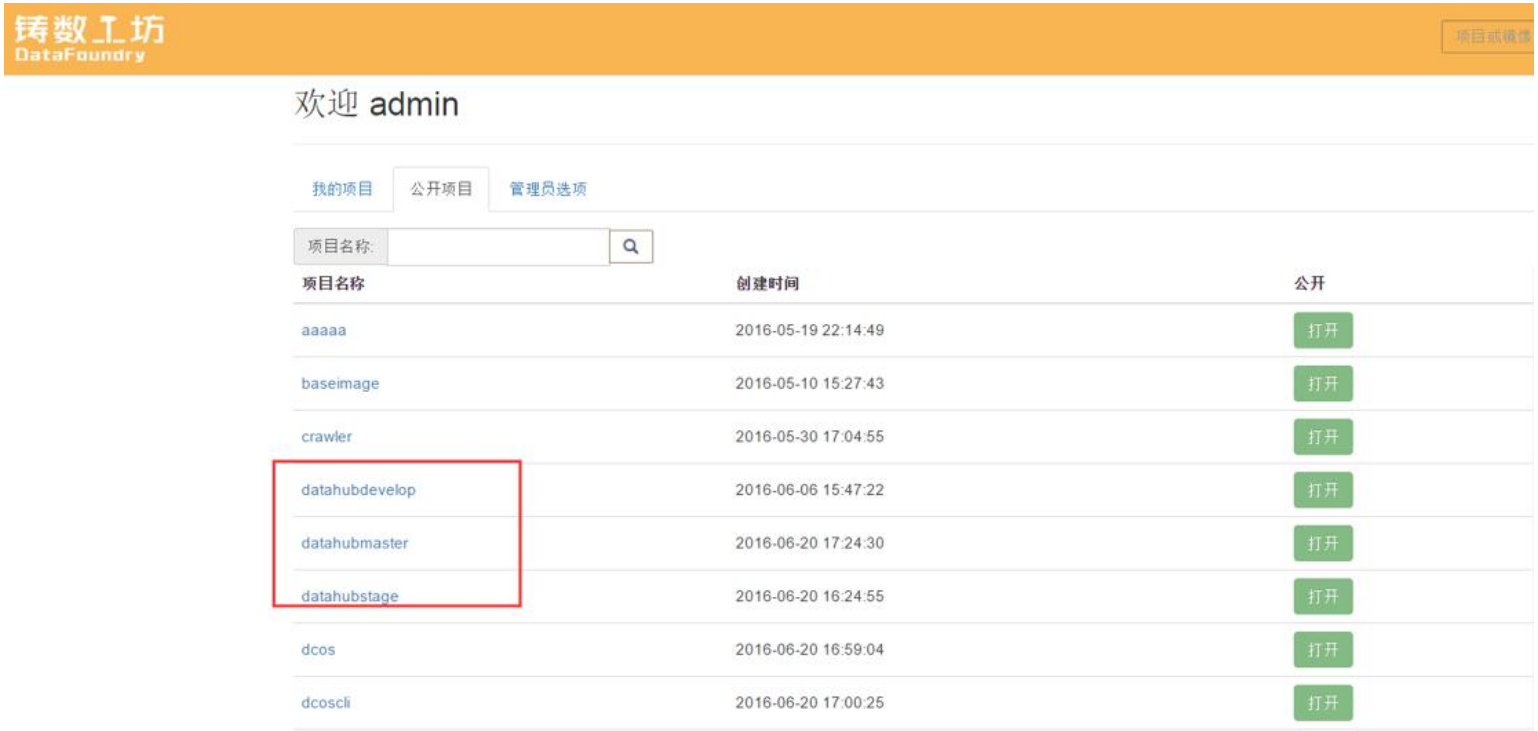
http://readme.drone.io/setup/overview/

```
ubuntu@ip-172-31-6-181:~$ cat /etc/drone/dronerc
REMOTE_DRIVER=github
REMOTE_CONFIG=https://github.com?client_id=[redacted]&client_secret=[redacted]dc239c5f2d7000ba17d
ef217
DATABASE_DRIVER=mysql
DATABASE_CONFIG=root:[redacted]@tcp(drone.[redacted].rds.cn-north-1.amazonaws.com.cn:3306)/drone?parseTime=true
debug=true
PLUGIN_FILTER=plugins/* registry.dataos.io/library/*
```

```
docker run \
--volume /var/lib/drone:/var/lib/drone \
--volume /var/run/docker.sock:/var/run/docker.sock \
--env-file=/etc/drone/dronerc \
--restart=always \
--publish=80:8000 \
--detach=true \
--name=drone \
drone/drone:0.4.2
```

```
compose:
  database:
    image: registry.dataos.io/library/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=1234
      - MYSQL_DATABASE=test
      - MYSQL_USER=test
      - MYSQL_PASSWORD=1234
build:
  image: golang
  branch: develop
  environment:
    - MYSQL_PORT_3306_TCP_ADDR=127.0.0.1
    - MYSQL_PORT_3306_TCP_PORT=3306
    - MYSQL_ENV_MYSQL_DATABASE=test
    - MYSQL_ENV_MYSQL_USER=root
    - MYSQL_ENV_MYSQL_PASSWORD=1234
    - MYSQL_CONFIG_DONT_UPGRADE_TABLES=yes
  commands:
    - mkdir -p /drone/src/github.com/asiainfoLDP
    - cp -R /drone/src/github.com/cherry4477/datahub_subscriptions /drone/src/github.com/asiainfoLDP/
    - export GOPATH=/drone
    - go get github.com/tools/godep
    - cd /drone/src/github.com/asiainfoLDP/datahub_subscriptions
    - $GOPATH/bin/godep restore
    - _db/initdb_v1.0.sh
    - go test -v -cover ./....
    - go build

publish:
  docker:
    environment:
      - DOCKER_LAUNCH_DEBUG=true
    registry: registry.dataos.io
    #mirror: registry.dataos.io
    #image_name: registry.dataos.io/datahubdevelop/sub
    repo: datahubdevelop/sub
    username: admin
    password: [redacted]
    email: mengjing@asiainfo.com
    #file: Dockerfile
    Tag: [$$COMMIT,latest]
    when:
      branch: develop
      event: push
```


铸数工坊
DataFoundry
欢迎 admin
我的项目
公开项目
管理员选项
项目名称:
项目名称
创建时间
公开
aaaaa
2016-05-19 22:14:49
打开
baseimage
2016-05-10 15:27:43
打开
crawler
2016-05-30 17:04:55
打开
datahubdevelop
2016-06-06 15:47:22
打开
datahubmaster
2016-06-20 17:24:30
打开
datahubstage
2016-06-20 16:24:55
打开
dcos
2016-06-20 16:59:04
打开
dcoscli
2016-06-20 17:00:25
打开

datahubdevelop

所有者: admin

镜像仓库

用户

日志

镜像名称:

datahubdevelop/sub

标签	Pull 命令
08b86cc5776574af29d6677c84d55e6e0b54dc61	docker pull registry.dataos.io/datahubdevelop/sul
8d478b2bf2074667fdb807132122bcf5d869eefb	docker pull registry.dataos.io/datahubdevelop/sul
aeff356d351f4776e6374bfbe3dd33ae6d9edc79	docker pull registry.dataos.io/datahubdevelop/sul
bede5ae87ce6c1b8eb9c143da4e9b5b7a7bae030	docker pull registry.dataos.io/datahubdevelop/sul
deb7caf41e2f5011cb5bf2aec8e07bc3b25f36df	docker pull registry.dataos.io/datahubdevelop/sul
ecb59991eec07941cc42366d767acb5a13b48b7c	docker pull registry.dataos.io/datahubdevelop/sul
f6d317399ceb5ccf4f3c8490349eee68a7bbb444	docker pull registry.dataos.io/datahubdevelop/sul
latest	docker pull registry.dataos.io/datahubdevelop/sul

```
compose:
  database:
    image: registry.dataos.io/library/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=1234
      - MYSQL_DATABASE=test
      - MYSQL_USER=test
      - MYSQL_PASSWORD=1234
build:
  image: golang
  branch: release1.3
  environment:
    - MYSQL_PORT_3306_TCP_ADDR=127.0.0.1
    - MYSQL_PORT_3306_TCP_PORT=3306
    - MYSQL_ENV_MYSQL_DATABASE=test
    - MYSQL_ENV_MYSQL_USER=root
    - MYSQL_ENV_MYSQL_PASSWORD=1234
    - MYSQL_CONFIG_DONT_UPGRADE_TABLES=yes
  commands:
    - mkdir -p /drone/src/github.com/asiainfoLDP
    - cp -R /drone/src/github.com/cherry4477/datahub_subscriptions /drone/src/github.com/asiainfoLDP/
    - export GOPATH=/drone
    - go get github.com/tools/godep
    - cd /drone/src/github.com/asiainfoLDP/datahub_subscriptions
    - $GOPATH/bin/godep restore
    - _db/initdb_v1.0.sh
    - go test -v -cover ./....
    - go build
    - integration.sh

publish:
  docker:
    environment:
      - DOCKER_LAUNCH_DEBUG=true
    registry: registry.dataos.io
    #mirror: registry.dataos.io
    #image_name: registry.dataos.io/datahubstage/sub
    repo: datahubstage/sub
    username: admin
    password: [redacted]
    email: mengjing@asiainfo.com
    #file: Dockerfile
    Tag: [$$COMMIT,latest]
    when:
      branch: release1.3
      event: [push,tag]
```

```
publish:
  docker:
    environment:
      - DOCKER_LAUNCH_DEBUG=true
    registry: [redacted]
    #mirror: registry.dataos.io
    #image_name: registry.dataos.io/datahubdevelop/sub
    repo: datahubmaster/sub
    username: [redacted]
    password: [redacted]
    email: mengjing@asiainfo.com
    #file: Dockerfile
    Tag: [$$COMMIT,latest]
    when:
      branch: master
      event: [tag,push]
```

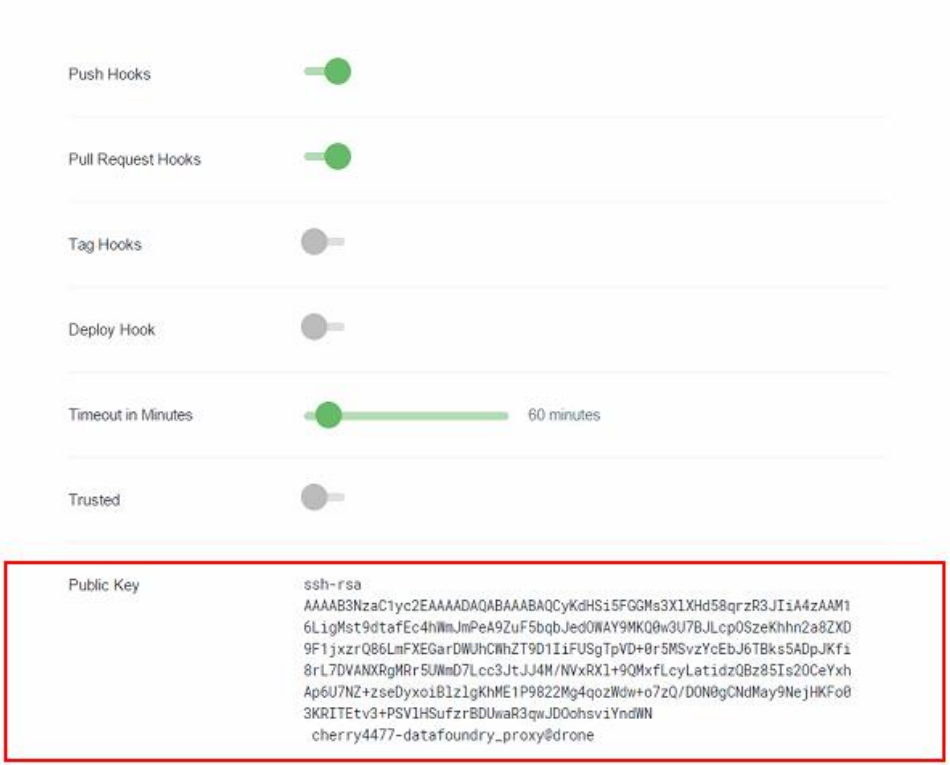
K8s插件(custom plugin)

https://github.com/geofeedia/drone-k8s

```
# perform a rolling-update
publish:
  drone-k8s:
    image: your-repo/your-org/drone-k8s:1.0.0
    replication_controller: some-rc
    namespace: some-ns
    docker_image: some-repo/some-org/some-image:1.0.0
    path_to_cert_authority: /path/to/ca.pem
    path_to_client_key: /path/to/worker-key.pem
    path_to_client_cert: /path/to/worker.pem
    update_period: 5s
    timeout: 30s

# perform an update for a deployment
publish:
  drone-k8s:
    image: your-repo/your-org/drone-k8s:1.0.0
    namespace: some-ns
    is_deployment: true
    deployment_resource_name: some-deployment
    container_name: some-container
    docker_image: some-repo/some-org/some-image:1.0.0
    path_to_cert_authority: /path/to/ca.pem
    path_to_client_key: /path/to/worker-key.pem
    path_to_client_cert: /path/to/worker.pem
```

Openshift中触发自动部署



Openshift中触发自动部署


```
55
56 deploy:
57   ssh:
58     host: [redacted]
59     user: mengjing
60     port: 22
61     commands:
62       - ssh [redacted]
63       - oc login dev.dataos.io --username=mengjing --password=[redacted]
64       - oc project datahub
65       - oc deploy subscription --latest
66
67
```

- CI/CD介绍
- 基于drone的CI/CD
- **私有镜像仓库的搭建**

What is Harbor

Harbor是一个企业级Registry服务。Harbor对开源的Docker Registry服务进行了扩展,添加了更多企业用户需要的功能。Harbor被设计用于部署一套组织内部使用的私有环境，这个私有Registry服务对于非常关心安全的组织来说是十分重要的。

Harbor的特点

1.基于角色的访问控制 - 用户与Docker镜像仓库通过“项目”进行组织管理；

2.图形化用户界面 - 用户可以通过浏览器来浏览，检索当前Docker镜像仓库，管理项目；

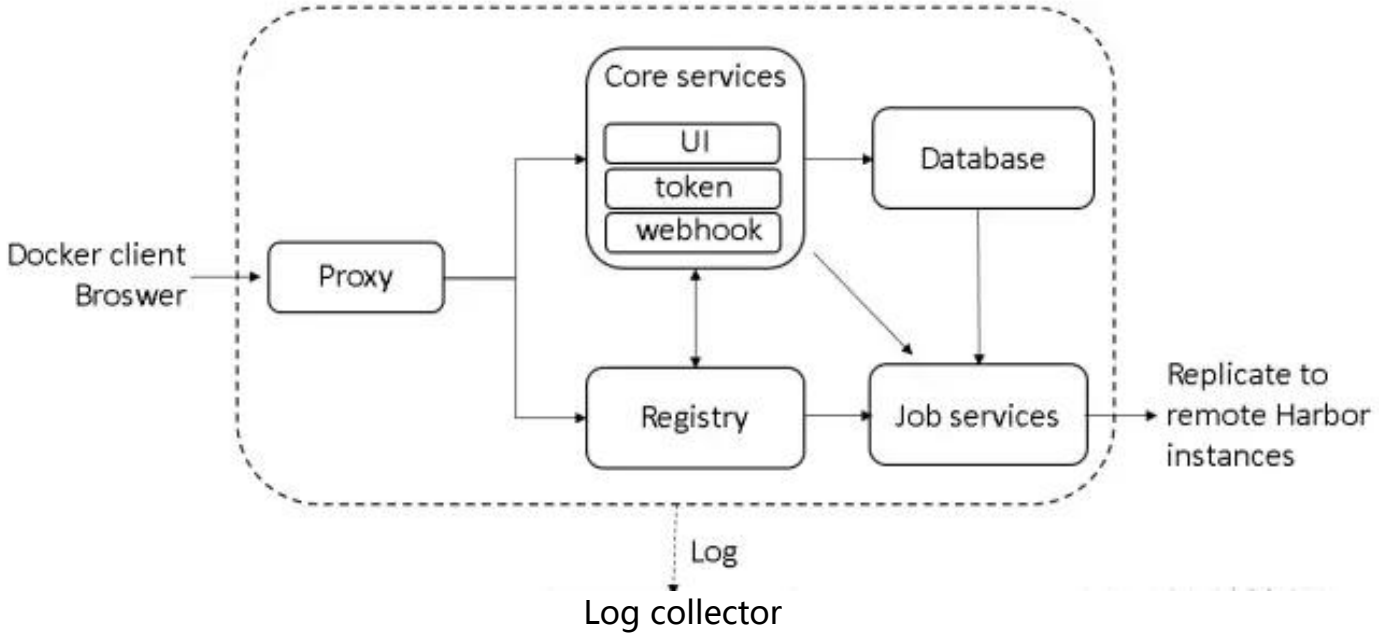
3.审计管理 - 所有针对镜像仓库的操作都可以被记录追溯，用于审计管理；

4.国际化 - 基于英文与中文语言进行了本地化，可以增加更多的语言支持；

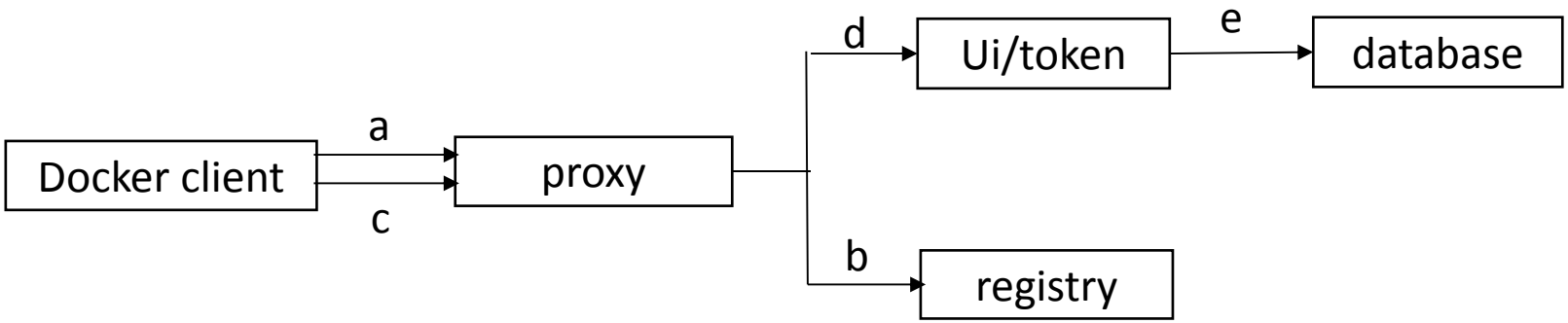
5.RESTful API - RESTful API 提供给管理员对于Harbor更多的操控, 使得与其它管理软件集成变得更容易；

6.多实例间镜像的复制-目前只支持多harbor实例间复制镜像。

Harbor的架构


Core services
UI
token
webhook
Database
Docker client
Broswer
Proxy
Registry
Job services
Replicate to
remote Harbor
instances
Log
Log collector

Harbor是如何工作的



启动registry

```
apiVersion: v1
kind: ReplicationController
metadata:
  creationTimestamp: null
  generation: 1
  labels:
    name: registry
  name: registry
spec:
  replicas: 1
  selector:
    name: registry
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: registry
    spec:
      containers:
      - env:
        - name: REGISTRY_HTTP_SECRET
          value: cherry4477
        - name: REGISTRY_STORAGE_S3_BUCKET
          value: registry3
        - name: REGISTRY_STORAGE_S3_ACCESSKEY
          value: [redacted]
        - name: REGISTRY_STORAGE_S3_SECRETKEY
          value: [redacted]
        - name: REGISTRY_STORAGE_S3_REGION
          value: cn-north-1
        image: caicloud/harbor_registry:2.3.0
        imagePullPolicy: IfNotPresent
        name: registry
        ports:
        - containerPort: 5000
```

启动registry

```
        protocol: TCP
      - containerPort: 5001
        protocol: TCP
      resources: {}
      terminationMessagePath: /dev/termination-log
      volumeMounts:
      - mountPath: /etc/registry
        name: portus-config
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    securityContext: {}
    terminationGracePeriodSeconds: 30
    volumes:
    - name: portus-config
      secret:
        secretName: portus-config
status:
  replicas: 0
```

registry的重要配置

```
version: 0.1
log:
  level: info
  fields:
    service: registry

storage:
  s3:
    accesskey: [redacted]
    secretkey: [redacted]
    region: cn-north-1
    bucket: registry3
    encrypt: false
    secure: true
    v4auth: true
    rootdirectory: /

notifications:
  endpoints:
    - name: "portus"
      url: http://ui/service/notifications
      headers: null
      timeout: 5s
      threshold: 5
      backoff: 1s
      disabled: false

auth:
  token:
    issuer: registry-token-issuer
    realm: http://registry.dataos.io/service/token
    rootcertbundle: /etc/registry/root.crt
    service: token-service
```

启动Harbor

```
apiVersion: v1
kind: ReplicationController
metadata:
  creationTimestamp: null
  generation: 1
  labels:
    name: ui
  name: ui
spec:
  replicas: 1
  selector:
    name: ui
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: ui
    spec:
      containers:
      - env:
        - name: MYSQL_HOST
          value: [redacted]
        - name: MYSQL_PORT
          value: "3306"
        - name: MYSQL_USR
          value: portus
        - name: MYSQL_PWD
          value: [redacted]
        - name: REGISTRY_URL
          value: http://registry:5000
        - name: CONFIG_PATH
```

启动Harbor

```
        - name: CONFIG_PATH
          value: /etc/ui/app.conf
        - name: HARBOR_REG_URL
          value: registry.app.dataos.io
        - name: HARBOR_ADMIN_PASSWORD
          value: [redacted]
        - name: HARBOR_URL
          value: registry.app.dataos.io
        - name: AUTH_MODE
          value: ldap_auth
        - name: LDAP_URL
          value: ldap://[redacted]:389
        - name: LDAP_BASE_DN
          value: cn=%s,ou=Users,dc=openstack,dc=org
        - name: LOG_LEVEL
          value: debug
        image: 172.30.188.59:5000/harbor/harbor
        imagePullPolicy: Always
        name: ui
        ports:
        - containerPort: 80
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
      dnsPolicy: ClusterFirst
      nodeSelector:
        role: reserve
      restartPolicy: Always
      securityContext: {}
      terminationGracePeriodSeconds: 30
status:
  replicas: 0
```

启动proxy

```
apiVersion: v1
kind: ReplicationController
metadata:
  creationTimestamp: null
  generation: 1
  labels:
    name: proxy
  name: proxy
spec:
  replicas: 1
  selector:
    name: proxy
  template:
    metadata:
      creationTimestamp: null
      labels:
        name: proxy
    spec:
      containers:
      - image: caicloud/harbor_proxy:latest
        imagePullPolicy: IfNotPresent
        name: proxy
        ports:
        - containerPort: 80
          protocol: TCP
        - containerPort: 443
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        volumeMounts:
        - mountPath: /etc/nginx
          name: nginx-config
      dnsPolicy: ClusterFirst
```

启动proxy

```
      restartPolicy: Always
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
      - name: nginx-config
        secret:
          secretName: nginx-config
status:
  replicas: 0
```

nginx重要配置

```
location /v2/ {
  add_header 'Docker-Distribution-Api-Version' 'registry/2.0' always;
  proxy_pass http://registry/v2/;
  proxy_set_header Host $host;
  proxy_set_header X-Real-IP $remote_addr;
  proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

  # When setting up Harbor behind other proxy, such as an Nginx instance, remove the below line if the proxy already has similar settings

  proxy_set_header X-Forwarded-Proto https;

  proxy_buffering off;
  proxy_request_buffering off;

}
```

为proxy的router添加认证

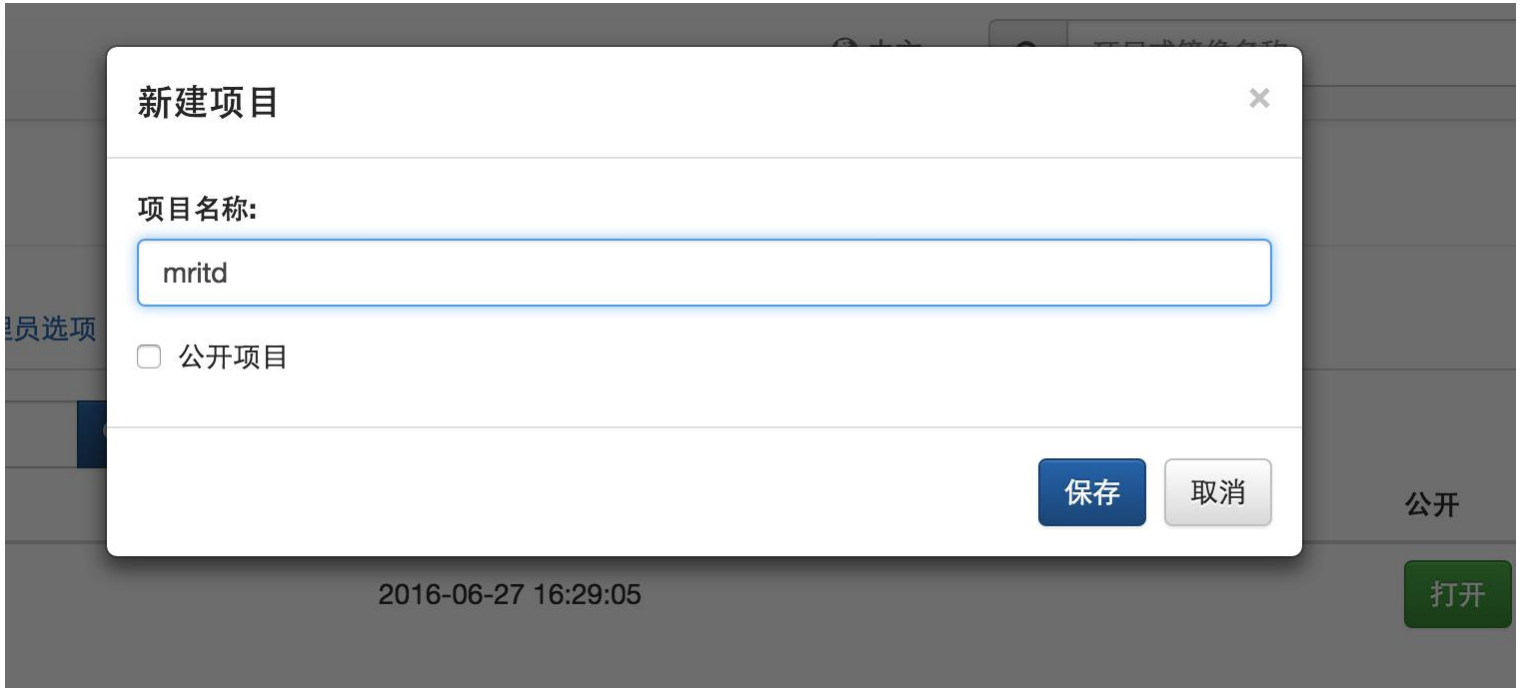
- $ oc create route edge --service=proxy \
- --cert=aaa/ca.crt \
- --key=aaa/ca.key \
- --hostname=www.example.com

Router中特别需要注意的

```
HRw\r\nOi8vb2NzcC5zdGFydHNzbC5jb20vY2EwMAYIKwYBBQUHMAKGJGh0dHA6Ly9haWEu\r\nc3RhcnRzc2wuY
\nOi8vY3JsLnN0YXJ0c3NsLmNvbS9zZnNjYS5jcmwwDQYJKoZIhvcNAQELBQADggIB\r\nAI274T7wqbpK6IUpiN
VHNhW+bXyJIc+UBs8RdjcgJMmGHSVhMPs2zvncMwx5b2FHvr4ajWwXcL3INlF\r\nvwJYifZgzi9FSe+Xg9KLqfQ
8V/TJNYlJYNjLF9xSHxgpLYu5WHPFzHzyFdPlDdtWZbgoTyXJ0zKwsMV\r\nw1zS6WDAGRWHgWlRRXPbmBSBPhXk
waTg7jKDLpzC34/PEvPYcF7X5pybb/TQVjjGzWZg61sQERjfN7B\r\nePHm6SGg74pOkX4XVVuX8SnaH/Y+dTmaZ
KhtlH9xnb2l9vxF581/Tlxea6OUtHfk4J7GvGcowyADI8g\r\n7OOYkS8e9GQ2DBVqacq0VrnIFA0uAXGTL27Oh6
f1L8qBqy5FMQDhlET6Lwh0FT1iTvz03xlQMRZBzyS\r\nEkE9+JcNYJ8zFJKmjFvuStRRmMJTdkorYuwFw2tQZLF
    CERTIFICATE-----\r\n"
  insecureEdgeTerminationPolicy: Allow
  key: |-
    -----BEGIN RSA PRIVATE KEY-----
    MIIEogIBAAKCAQEAuio7aqYAEYS6frRqNXNW2We3kgp4U08b3kjARaDReFQGPzfg
    p4pbnrgN4Ybi/LpprwTuAt8gdOcTmnDJ+0u/sksJnz4biYTH8tpcR/RZIUhHEFxw
    ON+Otx5KLGRmf1Iewq3IZg0uSaa1csFlv3VcOH/JbdRUVE4Q8neY5k6mG1uKpIil
    z7n8bkj4tnGvBp4G5yTDQ6zfKQtdiFBmGuM9EndOjjpx7N/qzZA2pzcip3umRd/x
    fscIBkPwmeliqvl1om3TVVxrGpiKwr4jCCcqcHL7IIfpn8cO8zhUlWxnKSKkZ5ju
    K41UaXaXHDQBNHEcxg5Ctpt4LBJC1AsTZD4sTwIDAQABAoIBAG9bh8MZnPzdqjjU
    bpuebJsLQXFoNeWPcpoaZni/48zYZgw2vnk5d+iPLC51Yx3N3B3HEyBAm8eR2dYO
    FaPuAbMC07SfkBVIidoo5/5amV4hP/D3emFqyJGc2r2HKRCL7L6C+VaiF8gSooMM
    UHKgHxPkzHaYBgGP2O9QvvM0pIPDzs0RF5n1f+B/WaE62+Xn5l3+fexKqZM7t+zD
    NjhbSkPgiPrPsAVpIzI8ERbHfn7z4uuOqfnlvA/9tZ9XG3z/wwfZfg0ufKevRJKP
    wUpB6iJ46GMG+vzQnBGlk1b0+AAZcXj1opPyGlQTNQY5LakPL+Ln2npW45WVEGJe
    rF8M1tECgYEA5KhqP8Br/jaYAEZpquGuo2b7ifbAuKYeZfCrnM08KPsb6gjfJwEc
    fMbB880qCD200RHYKJZJBa5BiCErZYt1/xwtqtrI2c9HGj/V8LF+LdYa5Xi5scN/
```

通过www.example.com访问私有镜像仓库

Harbor的使用-创建私有项目

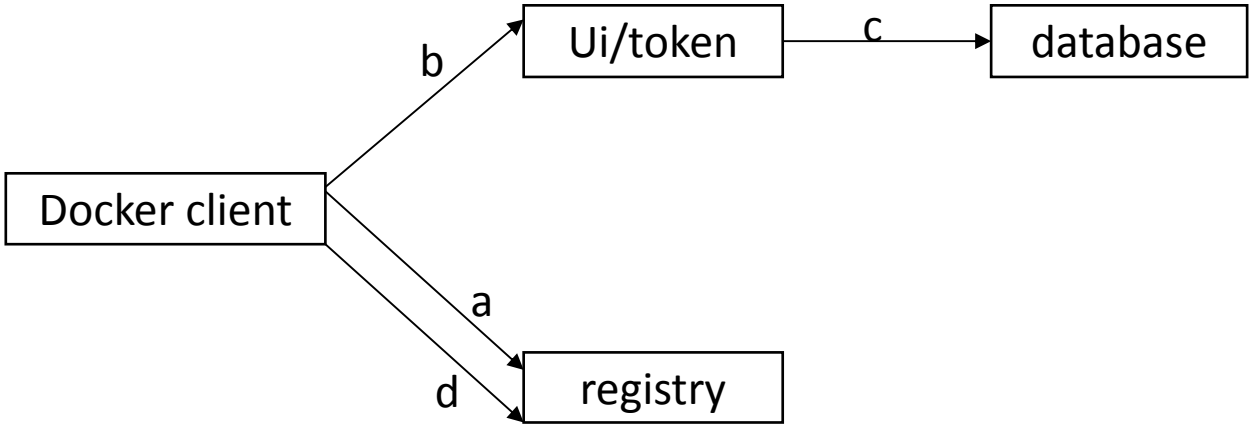


Harbor的使用-push 镜像

1.先使用docker tag为镜像打标签；

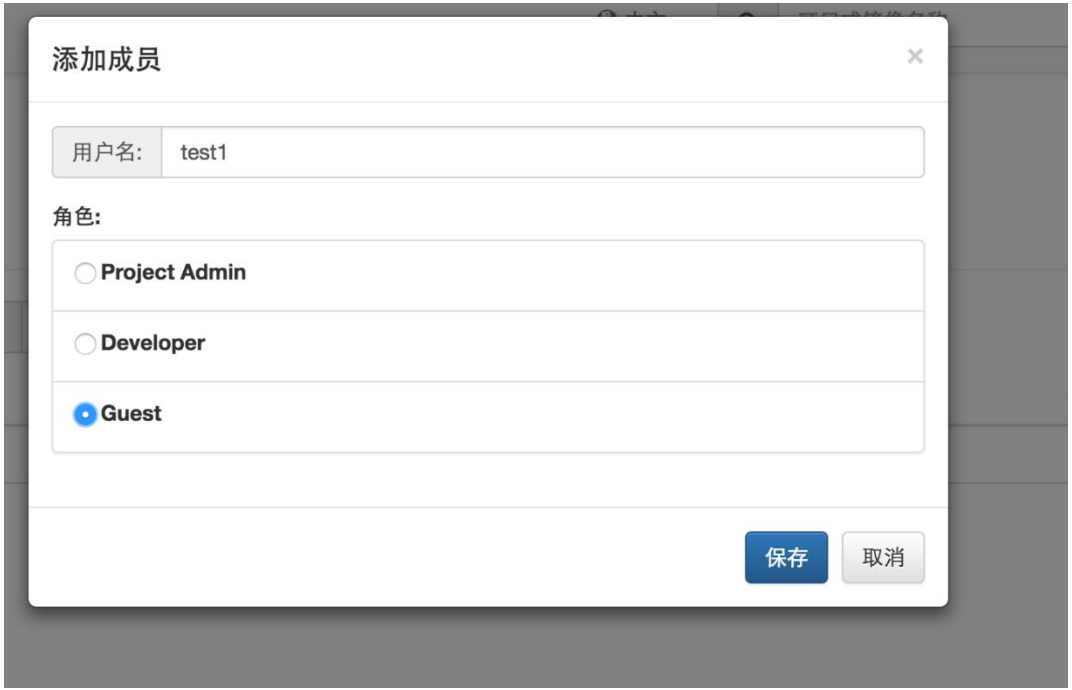
2.使用docker push镜像名push镜像。

push 镜像的工作流



Harbor的使用-管理成员

Harbor的使用-管理成员




谢谢！