工作整理

TODO

- https://blog.csdn.net/weixin_40242845/article/details/128400291
- 多个 ssh.Channel 时的 status 问题
- https://blog.csdn.net/hui1429/article/details/80144863
- 默认的上传下载速度
- BatchModifyAutoPostRootList
- UserInfo
- 自动创建 policy-data 目录
- 创建 user 表
- SyncStatus, 更新 status, 管理 ssh.Channel
- RootAccount(nil)
- LoadFromFile 去掉 os.MkdirAll
- 更改用户时的 ssh.Channel 处理, r.CloseSftpReqServers
- associateStatus
- ui.probe = &pb
- 日志大小处理
- https://www.gnu.org/software/bash/manual/html_node/Shell-Parameter-Expansion.html

EngineMgr

```
* 引擎名称: 172.18.6.140

* 厂商: WebRAY

* 设备类型: RayGate

* 部署模式: 单机

* 引擎IP: 172.18.6.140

* 协议: https

* 端口: 8443

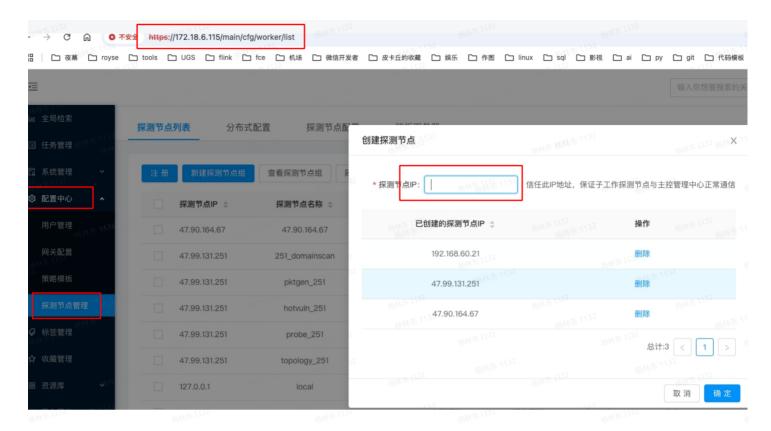
* 所属引擎组: api引擎组

* 用户名: apiuser

* 密码: admin123!
```

```
默认 443 端口
2
3
 4
    /api/v1
 5
        http://127.0.0.1:89
 6
 7
        /usr/lib/python3.7_venv_raygate/bin/uwsgi \
             --ini /rayos/app/webapp/raygate/uwsgi.ini
8
        后端,插入数据
9
10
     引擎的日志在 tail -f /var/log/webraydb/workermsg.log;
11
         2025-04-14 10:33:22,703 INFO: [MainProcess] worker_manage 状态正常 ... [in
12
    worker.py:119]
    2025-04-14 11:03:24,468 INFO: [MainProcess] worker_manage 状态正常 ... [in
13
    worker.py:119]
14
15
16
    */(finger|async|logout|cfg|asset|system|report|logsystem|engine|label|base|acti
    <del>ve)/</del>
       http://127.0.0.1:80
17
18
      /usr/bin/python2 \
19
           /rayos/app/webapp/paladin/run.py
20
21
22
    ^/vm/(vmreport/|download/|css/|js/|img/|signin)
23
24
    http://127.0.0.1:16530
25
26
    /rayos/app/worker_manage
27
        python3 /RayOS/app/worker_manage/worker.py
28
        上报引擎心跳
29
    引擎
30
31
        分布式引擎: jobexe
        api引擎,能登录
32
33
34
        python3 /RayOS/app/worker_manage/worker.py
35
36
```

IP段加白,添加引擎



python

from urllib.parse import quote

C++

https://blog.csdn.net/youlinhuanyan/article/details/142738063

内存布局: https://www.cnblogs.com/zl1991/p/7040687.html

coredump

```
1
    # go race condition check
2
3
    # ulimit -c;
    ulimit -c unlimited;
4
5
    # %e是可执行文件的名称,%s是信号名,%u是用户ID,%g是组ID,%p是进程ID,%t是时间戳
6
7
8
    cp /etc/sysctl.conf{,.bak};
    echo 'kernel.core_pattern=/var/log/rlog/cacheproxy/core-%e-%s-%u-%g-%p-%t' \
9
        >> /etc/sysctl.conf;
10
    sysctl -p;
11
```

Apt

https://wiki.debian.org/zh_CN/Apt

https://wiki.debian.org/AptCLI

Suricata

Sourceinsight rust

https://linuxcpp.0voice.com/?id=112226

https://zhuanlan.zhihu.com/p/632044134

Crontab

• https://segmentfault.com/a/1190000023029219

Docker

```
1 {
2    "insecure-registries": ["172.21.1.73:8080"],
3    "registry-mirrors": ["http://172.21.1.73:8080"]
4 }
```

```
1 # docker mirrors
2 # 目前国内可用Docker镜像源汇总
3 # https://www.coderjia.cn/archives/dba3f94c-a021-468a-8ac6-e840f85867ea
4
5 docker build \
-t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
-f ./dockerfiles/autopost.Dockerfile .
```

Git

```
1 https://kkgithub.com/
2 https://hub.yzuu.cf/
3
4 # 提交时转换为LF,检出时转换为CRLF
5 git config --global core.autocrlf true
6
7 # 提交和检出时都不转换
8 git config --global core.autocrlf false
9
```

Cacheproxy

管理环境

```
虚拟平台:
https://172.21.151.12:8006/#v1:0:=qemu%2F104:4:5:::::
root Webray@0808

https://172.21.15.37/user/login
admin: 2025-01-09 15:02:36
页面: admin ABa21@23Fulv1
```

Env 配置

```
代码块

1 CC=x86_64-linux-musl-gcc
2 CGO_ENABLED=1
3 CGO_LDFLAGS=-02 -g -static
4 GOROOT=C:/Program Files/go
5
```

```
代码块

func(*Server)

type Handler func(context.Context, net.Conn) error

type SFTPMiddleware func(Handler) Handler

func Chain(m ...SFTPMiddleware) SFTPMiddleware {
```

```
7
         return func(next Handler) Handler {
             for i := len(m) - 1; i >= 0; i-- {
 8
                 next = m[i](next)
9
10
             }
             return next
11
       ANT 32
12
     }
13
14
15
     s.middleware = []{
16
         Recovery(),
         SSHAuth(ss.SSHConfigs()),
17
         WhiteList(ss.GetProbeEgressIP),
18
19
         Status(ss.GetProbeStatus),
         Logging(),
20
         ConnectionLimit(ss.Limit)
21
22
     next = ServeSFTP(ctx context.Conext, conn net.Conn) error {
23
24
         s.probe.AppendSftpReqServer(p.Username, channel, connID)
25
         defer s.probe.RemoveSftpReqServer(p.Username, connID)
         defer server.Close()
26
27
         server.Serve()
28
     }
29
     go Recovery(
30
      SSHAuth(
31
             WhiteList(
32
                 Status(
33
                     go ps.Active()
34
                     defer func(){
                         if cnt==0 {
35
                             go. ps.Logoff()
36
                         } ** 11
37
                     }()
38
39
                     Logging(
                         ConnectionLimit(next)
40
41
42
               10.11
43
             )
      杨林东)1
44
     )
45
46
     Logging: info user "ridward" logged out
47
48
     go.ps.Logoff
```

```
https://blog.csdn.net/Eivene/article/details/140774552
reflect
```

```
http://bbs.itying.com/topic/676cf34b5b798701ddf78072
  4
  5
     - 服务器管理
     https://172.21.150.16/ui/#/host/vms/89
  6
     root/Esxi#0926
  7
     - 150.39 的账密 root/Superman@12345
  8
 9
     - proxmox 账户
10
 11
     root/Webray@0808
 12
     - 删除
 13
     点中-more-remove
 14
 15
     探针: https://172.21.15.37/user/login
 16
     admin/ABa2l@23Fulv1
17
 18
     系统管理/集中配置管理
 19
 20
 21
     - img 下载
 22
     - 外部: http://172.18.9.4:8001/products/fap-cacheproxy/
 23
     - 内部: http://172.21.1.25:8000/products/fap-cacheproxy/
 24
 25
 26
     上传 data(ibpd-uat-01), upload,
 27
 28
     创建系统,
 29
     两个盘, 128G, 256G
 30
     2 sockets, 4 cores = 8 cres
 31
     16384 Memory MiB(16G)
 32
 33
     - 登录账号: root/webray++
 34
     - 安装密码: WebRW2r$
 35
     输入 y 确认
 36
 37
     等待好长时间....
 38
 39
     y; xfs; rdata/cacheproxy; y; roboot;
 40
 41
     等好久...
 42
     admin密码默认aDmin@3.21,首次登录需修改密码,修改后使用新密码登录 Forget@12345678
 43
 44
     配置网络
 45
     vlan -A -v MngtVlan -f 172.21.150.107 -m 255.255.255.0;
 46
 47
     vlan −S;
     route -A -i 0.0.0.0 -m 0.0.0.0 -g 172.21.150.1;
 48
     route -S;
 49
```

```
50
51
    settime
    help
52
    upsshd start
53
54
55
    [autopost]
56
57
58
    docker build \
    -t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
59
    -f ./dockerfiles/autopost.Dockerfile .;
60
61
    docker save registry.ibdp.webray.com.cn:51808/fap/autopost--o 'autopost.tar';
62
63
    [autopost]
64
65
    - sftp_policy_path: /policy-data
       pull 模式要忽略的远程 server 的策略存储目录
66
67
       push 不用此配置
       root 账户的家目录 HomeDir (所以单导pull模式能同步此目录下的所有文件)
68
    - file_path: /home/
69
       定期清理老文件的目录, NewMonitorRepo 中的的 DataDir:
70
        pull 模式下文件同步到的目录
71
       push 模式不用此配置
72
    - policy_file_path: /home/policy-data
73
       pull 模式不用此配置
74
        push 的时候,本地策略所在文件夹,同时也在检测文件
75
76
    [cacheproxy]
77
    RootAccount() 一致维护 root 用户
78
    - data_dir: /home/cacheproxy/data
79
       探针文件夹创建的目录
80
       策略解压到的目录
81
       定期清理的目录 (NewMonitorRepo)
82
       root 用户的家目录
83
    - app_dir: /home/cacheproxy
84
       创建一个 policy-data-backup 目录备份
85
       cert 所在目录
86
    data.db 所在目录
87
        处理请求的时候看了这个目录的信息??? HandleConn: IsDiskWork()
88
    - policy_data_dir: /home/cacheproxy/data/policy-data
89
        push 过来的策略的存储路径
90
       定期检测并需要解压 .tgz
91
92
    p.HomeDir 是不同 sftpServer 的目录
93
94
95
    SystemProbeRecover
```

```
scp -0 -l 1024 /d/... root@172.21.150.28:/cacheproxy/
1
 2
 3
 4
    # 172.21.150.27
    export GOPROXY="http://172.21.3.11:3000"
 5
    账密: admin/Admin123, 端口是 8000
 6
 7
8
9
    - cmd/cacheproxy/main.go
10
    ///go:embed web/*
11
     - go.mod
12
13
    toolchain gol.21.4
14
    - configs/config.json
15
16
    data_dir
17
    app_dir
    policy_data_dir
18
19
20
    https://github.com/protocolbuffers/protobuf/releases
21
    # exec: "protoc": executable file not found in %PATH%
22
23
    https://github.com/protocolbuffers/protobuf/releases/download/v29.2/protoc-
    29.2-win64.zip;
24
25
26
    VERSION=1.0;
27
    # https://www.jianshu.com/p/10ae8a5a7704
     # go get github.com/go-kratos/kratos/cmd/kratos/v2@latest;
28
29
    go install github.com/go-kratos/kratos/cmd/kratos/v2@latest;
30
    kratos proto client ./internal/conf/conf.proto;
31
    kratos proto client ./api/cacheproxy/v1/cacheproxy.proto;
32
33
34
35
    go build -ldflags "-X main.Version=$(VERSION)";
36
    chmod a+x /usr/local/go/bin/kratos;
37
    go mod tidy;
38
    cd cmd/cacheproxy;
39
40
    go build;
     ./cacheproxy -conf ../../configs/config.json;
41
    dlv --listen=:2345 --headless=true --api-version=2 --accept-multiclient exec
42
     ./cacheproxy
```

```
43
    缓存转发的 root_passwd 要配置为 root
44
    # ../configs/autopost.yaml 要配置 sftp_address 和 3 个目录
45
    ./autopost -conf ../configs/autopost.yaml -mode push
46
    上传文件到单导的 /cacheproxy/home/policy-data
47
    单导日志类似:
48
    caller=biz/autopost_push.go:203 ... \
49
        policy-data/20241221181530_policy.tgz.part successfully
50
51
    缓存转发的日志类似:
52
        caller=biz/monitor.go .... ∖
53
        job: /home/cacheproxy/data/policy-data/20241221181530_policy.tgz
54
55
    文件会被解压到 /home/cacheproxy/data 下
56
        - 0XSDJI902
57
58
        - 0XSDJI905
        - 0XSDJI908
59
60
        - E04C81F400
        - SXXAtest11
61
    /home/cacheproxy/data/policy-data
62
    /hoem/cacheproxy/policy-data(??? 是什么)
63
```

Go

Protobuf: https://www.cnblogs.com/zhanchenjin/p/17454978.html

```
代码块
     runtime.gcdata: missing Go type information for global symbol .dynsym: size 72
 1
     https://www.cnblogs.com/binHome/p/13020178.html
 2
 3
     https://www.jianshu.com/p/a73cef45b65c
 4
 5
    wire:
 6
     https://www.cnblogs.com/jiujuan/p/16136633.html
 7
 8
    在 service/service.go
 9
     type AppRepo interface {
10
11
12
13
     在 data/data.go
14
     type appRepo struct {
15
16
         *mq
17
         *db
         *dp
18
19
     }
```

```
20
    func NewAppRepo() service.AppRepo {
21
        appRepo := &appRepo{}
22
        return appRepo
23
24
    }
25
26
    google.golang.org/protobuf/types/know/anypb
27
28
    marshal := v1.Flow { Enable: true, Subset: "127.0.0.1" }
29
    any, err := anypb.New(&marshal);
30
     [type.googleapis.com/engmgr.v1.Flow]:{enable:true, subset:"127.0.0.1"}
31
32
    grpcurl -plaintext -H "x-user-id: 1" 172.21.150.109:9000 list;
33
    grpcurl -plaintext -H "x-user-id:1" -d '{}' 172.21.150.109
34
    engmgr.v1.App.CreateGrp;
```

```
代码块
```

```
# 文档
   https://www.topgoer.com/go%E5%9F%BA%E7%A1%80/%E7%BB%93%E6%9E%84%E4%BD%93.html
2
3
  # 自定义 json
4
   https://juejin.cn/post/6844904184651874312
```

```
1
    # select
    https://www.cnblogs.com/qcy-blog/p/18520091
 2
 3
    # docker copy
    https://developer.baidu.com/article/details/3214252
 4
 5
    https://blog.csdn.net/u011461385/article/details/106017483
 7
    https://blog.csdn.net/chenchongg/article/details/86589395
 8
    # nil, close chan
9
    https://dave.cheney.net/2014/03/19/channel-axioms
10
    # strace
11
12
    // 只显示 renameat 的调用情况
13
    strace -f -e trace=renameat go run ./script/download-go go1.22.3
14
15
    # switch语句也可以与false字面值一起使用,提供了一种确定哪些条件未满足的方法
    https://www.cnblogs.com/cheyunhua/p/17945018
16
```

```
for pos, char := range "日本\x80語" { // \x80 is an illegal UTF-8 encoding fmt.Printf("character %#U starts at byte position %d\n", char, pos) } } 4 5
```

```
the module github.com/google/go-cmp contains a package in the directory cmp/.
 2
    That package's import path is github.com/google/go-cmp/cmp.
 3
     Packages in the standard library do not have a module path prefix
 4
     go install example/user/hello;
 5
 6
 7
    The install directory is controlled by the GOPATH and GOBIN environment
 8
     variables.
 9
    If GOBIN is set, binaries are installed to that directory.
    If GOPATH is set, binaries are installed to the bin subdirectory of
10
     the first directory in the GOPATH list.
11
     Otherwise, binaries are installed to the bin subdirectory of
12
     the default GOPATH ($HOME/go or %USERPROFILE%\go)
13
14
15
     go env -u GOBIN;
16
17
     $(dirname $(go list -f '{{.Target}}' .));
18
19
     go help importpath;
20
     GOCACHE: $HOME/.cache/go-build;
21
22
     import "example/user/hello/morestrings";
23
24
     go mod tidy;
25
26
     $HOME/.mygo/pkg/mod/
27
28
29
     go clean -modcache;
```

```
go env -w GOBIN=/path/to/your/bin;
 3
     Named return values;
  4
     A var statement can be at package or function level.
  5
     := 只可以在函数内使用.
  6
     rune is int32;
 7
     uintptr;
 8
11329
 10
     %T, %v
     Zero values;
 11
     different type requires an explicit conversion;
 12
     Gos switch cases need not be constants, and the values involved need not be
 13
     integers;
     Unlike C, Go has no pointer arithmetic.;
 14
15
 16
     type Vertex struct {
 17
             X int
 18
             Y int
     }
19
 20
 21
     (*p).X \rightarrow p.X;
22
 23
     var (
             v1 = Vertex{1, 2} // has type Vertex
 24
             v2 = Vertex{X: 1} // Y:0 is implicit
 25
             v3 = Vertex{}  // X:0 and Y:0
 26
             p = &Vertex{1, 2} // has type *Vertex
 27
 28
29
     Slices are like references to arrays;
 30
 31
     cap();
     a := make([]int, 5) // len(a)=5;
 32
     b := make([]int, 0, 5) // len(b)=0, cap(b)=5;
 33
     strings.Join;
 34
 35
     The range form of the for loop iterates over a slice or map.
36
     for i := range pow;
     delete(m, key)
 37
     elem, ok = m[key]
 38
 39
     The make function returns a map of the given type, initialized and ready for
 40
     use;
 41
42
 43
     func fibonacci() func() int {
             i, j := 0, 1 1 1 3 2
 44
              return func() int {
 45
 46
                     i, j = j, i+j
```

```
47
                   return i
48
           }
49
50
    Methods:
51
52
    The receiver appears in its own argument list
    between the func keyword and the method name.
53
    func (v Vertex) Abs() float64 {
54
55
           return math.Sqrt(v.X*v.X + v.Y*v.Y)
56
57
58
    You can only declare a method with a receiver
59
    whose type is defined in the same package as the method.
60
    Pointer receivers;
61
62
    Go interprets the statement v.Scale(5) as (&v).Scale(5);
63
64
    p.Abs() is interpreted as (*p).Abs();
65
    type Abser interface {
66
67
          Abs() float64
    }
68
69
70
    func (t *T) M() {
            if t == nil {
71
                    fmt.Println("<nil>")
72
73
                    return
74
            fmt.Println(t.S)
75
76
    }
77
    # Type assertions
78
    t := i.(T);
79
    t, ok := i.(T);
80
81
    switch v := i.(type);
82
83
    type Stringer interface {
84
       String() string
85
    }
86
87
    func (p Person) String() string {
88
          return fmt.Sprintf("%v (%v years)", p.Name, p.Age)
89
90
    }
91
92
    type error interface {
93
     Error() string
```

```
94
     }
 95
 96
     %q;
97
     func Index[T comparable](s []T, x T) int
98
99
     type List[T any] struct {
100
101
             next *List[T]
102
             val T
     }林东1132
103
104
     The evaluation of f, x, y, and z happens in the current goroutine and
105
     the execution of f happens in the new goroutine.
106
107
108
     ch <- v // Send v to channel ch.
109
     v:= <-ch // Receive from ch, and
110
111
                // assign value to v.
112
113
114
     Like maps and slices, channels must be created before use:
     ch := make(chan int);
115
116
117
     ch := make(chan int, 100);
118
     v, ok := <-ch;
119
120
     Only the sender should close a channel, never the receiver.
121
     Sending on a closed channel will cause a panic.
122
123
124
125
     select {
     case c <- x:
126
127
       x, y = y, x+y
128
     case <-quit:</pre>
             fmt.Println("quit")
129
130
             return
     } 林东 1132
131
132
133
     mu sync.Mutex;
134
     c.mu.Lock();
135
     c.mu.Unlock();
```

ssh

pg_dump

```
PGPASSWORD='admin1234' pg_dump -d fusion -U postgres --no-comments \
--column-inserts \
-t 'rule_ele|model';

--o --no-owner
--s --schema-only
```

pg

```
/install/modules/ray-postgresql/stop.sh;
 2
    psql postgresql://user:pass@host:port/dbname
    PGPASSWORD='admin1234' psql -U postgres -p 5432 fusion;
 3
 4
    echo "host ti_graph {user} {ip} md5" >> $PGDATA/pg_hba.conf
 5
 6
 7
 8
 9
    PGPASSWORD='Mjolnir' pg_dump -d test -U postgres -h 127.0.0.1 --no-comments \
    --column-inserts \
10
    --schema-only \
11
    -t 'group' \
12
    -t 'device_group' \
13
    -t 'device' \
14
15
    -t 'engine' \
    > enginemgr.sql;
16
17
18
    -s --schema-only,加上只导表结构
19
```

iptables

```
1 /rayos/cfg/iptables_port.conf 6379 5432
2 /rayos/utils/netac -A
```

ipaddr

sudo systemctl restart systemd-resolved.service;

```
flock -xn /var/run/custom_worker.lck -c \
    "nohup python3 -m flask rq worker &>> /var/log/webraydb/custom_worker.log
    &"
```

```
SELECT col.table_name, col.column_name, col.data_type, des.description
FROM information_schema.columns AS col

JOIN pg_statio_all_tables AS st

ON col.table_name=st.relname

LEFT JOIN pg_description AS des

ON (st.relid=des.objoid AND col.ordinal_position=des.objsubid)

WHERE col.table_name = 'document';
```

limits.conf/sysctl.conf

- 1 https://blog.csdn.net/qisianla/article/details/52383426
- 2 https://www.cnblogs.com/netsa/p/15385635.html

Pycharm

1 remote debug, /etc/ssh/sshd_config 中 AllowTcpForwarding yes

Iptables

- 1 [Unit]
- 2 AssertPathExists=/etc/sysconfig/iptables
- 3 iptables -I INPUT 2 -p tcp -m multiport --dport 80,5000,10290 -j ACCEPT;

```
network.host: 0.0.0.0
2
    discovery.seed_hosts: ["127.0.0.1"]
3
4
    network.bind_host
5
    network.public_host
6
    network.host
    discovery.seed_hosts: 集群主机列表
7
    discovery.seed providers: 基于配置文件配置集群主机列表
    cluster.initial master nodes: 启动时初始化的参与选主的node, 生产环境必填
10
    curl -K https://elastic:elastic@172.21.15.234/_settings?pretty
11
12
```

alembic 使用

```
cd fusion_app/module/resources/webapp/rayfusion;
mkdir misc;
cd misc;
alembic init db; # create alembic.ini and dir db/
```

```
1
     apt-get download git;
 2
     ssh-copy-id -i ~/.ssh/id_*.pub root@172.21.15.234;
     useradd -m -s /bin/bash lindyang;
 3
 4
    passwd lindyang;
 5
 6
     cp /etc/yum.conf{,.bak};
     #echo 'proxy=http://172.21.3.11:3128' >> /etc/yum.conf;
     echo "$USER $HOSTNAME= NOPASSWD: ALL" | sudo tee /etc/sudoers.d/$USER;
 8
 9
10
     cat /etc/os-release
11
12
     cat /etc/yum.repos.d/openEuler.repo
```

sudo apt install zlib1g zlib1g-dev;

sudo apt-get install -y git make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libffi-dev;

sudo apt install liblzma-dev;

v=3.9.19; wget https://repo.huaweicloud.com/python/\$v/Python-\$v.tar.xz -P /opt/.pyenv/cache/ --no-check-certificate; pyenv install \$v;

Virtualbox:

Visual C++ Redistributable for Visual Studio 2019 (version 16.11)

- 1. sudo groupadd docker
- 2. sudo usermod -aG docker \$USER
- 3. 【重新登录shell】或者【切换为root再切换成当前用户】

Docker 镜像

https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors

```
1 ARG DEBIAN_FRONTEND=noninteractive
2 ENV TZ=Asia/Shanghai
3 RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y install tzdata;
4
5 https://github.com/Tensho/docker-rbenv;
```

阿里云镜像查找

https://developer.aliyun.com/mirror/

```
#将可用源进行配置
     sudo tee /etc/docker/daemon.json <<EOF</pre>
 3
         "registry-mirrors": [
 4
             "https://dockerproxy.cn",
 5
             "https://docker.rainbond.cc",
 6
             "https://docker.udayun.com",
 7
             "https://docker.211678.top"
 8
9
        1
10
     }
11
     EOF
12
     # 重载,重启Docker服务
13
     sudo systemctl daemon-reload
14
```

Minikube

https://blog.csdn.net/Sindweller5530/article/details/116499761

https://github.com/luksa/kubernetes-in-action/tree/master/Chapter04

• autocmd FileType yaml setlocal tabstop=2 shiftwidth=2 expandtab autoindent

```
minikube service ingress-nginx-controller -n ingress-nginx;
```

```
1
    minikube start \
 2
        --vm-driver none \
3
        --driver=docker \
        --image-mirror-country='cn' \
     -image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers;
 5
 6
7
    minikube status;
 8
    minikube dashboard;
9
    minikube ssh;
    eval $(minikube docker-env);
10
11
12
    kubectl cluster-info;
    kubectl get nodes;
13
    kubectl describe node minikube
14
15
    minikube image load kubia;
16
    minikube ssh;
17
18
    docker tag kubia luksa/kubia:1.0;
19
    exit;
    kubectl create deployment kubia --image=luksa/kubia:1.0;
20
    kubectl get deployments;
21
22
23
    kubectl get pods;
    kubectl describe pod;
24
25
    kubectl expose deployment kubia --type=LoadBalancer --port 8080;
26
    kubectl get svc;
27
28
29
    k get nods/pods/deployments/pods;
30
    kubectl api-resources;
31
32
```

```
33
     # If you use Minikube to create the cluster,
     # no load balancer is created
34
35
    minikube service kubia --url;
36
    minikube ip;
37
     kubectl scale deployment kubia --replicas=3;
38
39
    kubectl get deploy;
40
41
42
    kubectl get pods -o wide;
43
     kubectl get nodes minikube -o yaml;
44
45
     kubectl proxy;
46
    http://127.0.0.1:8001/api/v1/nodes/minikube;
47
48
    kubectl explain nodes;
49
50
     kubectl explain node.spec; # --api-version
51
     kubectl explain pods --recursive;
52
53
     kubectl get ev -o wide;
     kubectl get ev --field-selector type=Warning;
54
55
56
    kubectl explain events;
57
     kubectl run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml >
58
    mypod.yaml;
59
    kubectl apply -f kubia.yaml;
60
61
62
     k run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml;
63
     k describe pod kubia; minikube ssh; curl 172.17.0.X:8080;
64
65
66
     Some of the pod's fields aren't mutable, so the update may fail,
     but you can always delete the pod and then create it again
67
68
     kubectl get pod kubia -o wide;
69
70
    kubectl run \
71
     --image=ubuntu:latest -it \
72
     --restart=Never
73
     --rm client-pod curl 172.17.0.7:8080;
74
75
76
    kubectl port-forward kubia 8080:8080;
77
    kubectl port-forward --help;
     kubectl logs --timestamps=true -f kubia;
78
```

```
79
          --since=2m
 80
          --since-time=2020-02-01T09:50:00Z
          --tail=10
 81
         -p (--previous)
 82
 83
      kubectl exec kubia -- ps aux;
 84
 85
      kubectl attach;
 86
 87
      kubectl port-forward kubia-ssl 8080 8443 9901;
 88
      curl https://example.com:8443 --resolve example.com:8443:127.0.0.1 -k;
 89
 90
      kubectl delete po kubia-ssl --grace-period 10;
 91
         spec.terminationGracePeriodSeconds
 92
         metadata.deletionGracePeriodSeconds
93
 94
 95
 96
      kubectl logs kubia-ssl -c kubia;
 97
      kubectl logs kubia-ssl --all-containers;
 98
      kubectl exec -it kubia-ssl -c envoy -- bash;
 99
100
      k delete pod kubia --wait=false;
101
102
     kubectl delete -f kubia-ssl.yaml;
103
      kubectl delete -f kubia.yaml,kubia-ssl.yaml;
104
      kubectl apply -f Chapter05/;
105
      kubectl delete -f Chapter05/;
106
          --recursive
107
108
109
      kubectl delete po --all;
110
      kubectl delete all --all;
111
112
113
      kubectl delete events,all --all;
114
      kubectl exec kubia-liveness -c envoy -- tail -f /var/log/envoy.admin.log;
115
116
      kubectl logs kubia-liveness -c envoy -p;
117
```

```
1 # NodePort
2 kubectl cluster-info;
3 curl http://192.168.49.2:30123;
```

```
kubectl get nodes -o jsonpath='{.items[*].status.addresses}';
 5
     kubectl get nodes -o \
 6
 7
     jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}';
 8
     # http://kt1bernetes.io/docs/user-guide/jsonpath
 9
10
     minikube service kubia-nodeport;
11
12
     # minikube sevrvice <service-name> [-n <namespace>]
13
     externalTrafficPolicy: Local;
14
15
     minikube addons list;
16
     minikube addons enable ingress;
17
     apiVersion: networking.k8s.io/v1;
18
```

```
1
    apiVersion: networking.k8s.io/v1
 2
    kind: Ingress
    metadata:
 3
    name: kubia
 4
 5
    spec:
      rules:
 6
 7
      - host: kubia.example.com
       http:
         paths:
         - path: /
10
           pathType: Prefix
11
           backend:
12
             service:
13
14
             name: kubia-nodeport
15
              port:
            number: 80
16
```

```
1 docker build -t kubia .;
2 删除 DaemonSet 也会删除 pod;
3 4 # job 资源
5 kubectl get jobs;
```

```
# completions: 5
    # parallelism: 2
 8
 9
    # 有问题
10
    kubectl scale job multi-completion-batch-job --replicas 3;
11
12
13
    activeDeadlineSeconds
    spec.backoffLimit=6 (default)
14
15
16
    CronJob
    分钟 小时 每月中的第几天 月 星期几(⊙是星期天)
17
    startingDeadlineSeconds
18
19
20
    创建 service 后,需要
21
    minikube ssh; # 登录
22
    curl `kubectl get svc | awk '{print $3}'` # CLUSTER-IP
23
24
25
    kubectl exec kubia-manual -- curl -s http://10.99.94.217; # 没有 -s, -- 是非必
    须得。
26
    sessionAffinity: ClientIP; # 不支持http cookie
27
    kubectl exec kubia-manual -- env;
28
29
    dnsPolicy;
30
31
    backend-database.default.svc.cluster.local;
32
    kubectl exec -it kubia-manual -- bash;
33
    curl -s http://kubia.default.svc.cluster.local;
34
    curl http://kubia;
35
36
    cat /etc/resolv.conf;
37
    ping kubia; # 无法ping通,因为是虚拟IP,只有与port结合才有意义
38
39
    kubectl describe svc kubia | grep -i endpoint;
40
    kubectl get endpoints kubia;
    curl http://external-service/echo;
41
42
    # kubia ExternalName
43
```

```
1 kubectl logs mypod --previous;
2 kubectl edit rc kubia;
3
4 export KUBE_EDITOR="/usr/bin/vim";
```

```
export EDITOR="/usr/bin/vim";
    kubectl scale rc kubia --replicas=4;
 6
    kubectl delete rc kubia --cascade=false;
 7
 8
    ReplicaSet 匹配缺少某个标签的 pod, 或包含某个标签名的 pod, 不管其值.
 9
10
    kubectl api-versions;
11
    replicaset.apps/kubia created;
12
13
    kubectl get rs;
    kubectl delete rs kubia;
14
15
    # 使用 DaemonSet 在每个节点上运行一个 pod
16
    # 使用 DaemonSet 只在特定的节点运行 pod
17
    # 节点可以被设置为不可调度,但DaemonSet可以绕过它
18
```

```
apiVersion: apps/v1
 1
     kind: ReplicaSet
 2
     metadata:
 3
       name: kubia
 4
 5
     spec:
       replicas: 3
 6
       selector:
 7
         matchLabels:
 8
           app: kubia
       template:
10
         metadata:
11
12
          labels:
             app: kubia
13
14
         spec:
15
          containers:
16
           - name: kubia
            image: kubia
17
            imagePullPolicy: Never
18
```

matchExpressions

```
1 selector:
2 matchExpressions:
3 - key: app
4 operator: In
```

operator

- In
- NotIn
- Exists
- DoesNotExist

```
apiVersion: v1
 1
 2
     kind: Pod
 3
     metadata:
       name: kubia-liveness
 4
 5
     spec:
       containers:
 6
1327
       - image: kubia-unhealthy
         imagePullPolicy: Never
 9
      name: kubia
         livenessProbe:
10
          httpGet:
11
12
             path: /
13
             port: 8080
           initialDelaySeconds: 15
14
```

```
- pod 共享 network, UTS, IPC, PID(默认未开启)
 1
    - 文件系统隔离, 但是可以通过 Volume 共享文件目录
 2
     - localhost 可以与同一 pod 中的其它容器通信
 3
    kubectl get po kubia-82d24 -o yaml;
 4
     kubectl explain pods;
 5
     kubectl explain pod.spec;
     kubectl create -f lindyang/kubia-manual.yaml;
 7
 8
     kubectl describe pod kubia-manual | grep 172;
1329
10
     minikube ssh;
     curl 172.17.0.7:8080;
11
12
    kubectl logs kubia-manual -c kubia;
13
```

```
14
     # 不通过 service 与 pod 通讯
15
     kubectl port-forward kubia-manual 8888:8080;
16
     curl http://127.0.0.1:8888;
17
18
     kubectl get pods --show-labels;
19
     kubectl get pod -L creation_method,env;
20
21
22
     kubectl label po kubia-manual creation_method=manual --overwrite
     # 删除标签, -
23
     kubectl label po kubia-manual createion method-
24
25
     kubectl get po -l creation_method=manual;
26
27
     kubectl get po -l env;
28
29
    kubectl get po -l '!env';
30
31
     env!=prod
32
     'env in (prod,dev)'
     'env_notin (prod,dev)'
33
34
     kubectl label node minikube gpu=false;
35
36
37
     kubectl annotate pod kubia-manual mycompany.com/someannotation="foo bar";
38
     kubectl get po --namespace kube-system;
39
40
41
     kubectl get ns;
42
     kubectl delete namespace custom-namespace;
43
44
     kubectl create namespace custom-namespace;
     kubectl create -f lindyang/kubia-manual.yaml -n custom-namespace;
45
46
47
     命名空间不允许包含点号.
48
49
     alias kcd='kubectl config set-context $(kubectl config current-context) --
     namespace'
50
     kubectl delete pod po1 po2;
51
52
     kubectl delete po -l creation_method=manual;
53
54
55
     kubectl delete ns custom-namespace;
56
57
     kubectl delete po --all;
58
    kubectl delete all --all;
59
```

Namespace

```
1 apiVersion: v1
2 kind: Namespace
3 metadata:
4 mame: custom-namespace
```

```
apiVersion: v1
1
    kind: Pod
 2
    metadata:
 3
      name: kubia-gpu
 4
5
    spec:
      nodeSelector:
6
27
        gpu: "true"
      containers:
     - name: kubia
        image: kubia
10
        imagePullPolicy: Never
11
```

```
minikube status;
 1
 2
    minikube kubectl -- get pods -A;
     alias kubectl="minikube kubectl --";
    minikube kubectl cluster-info;
 4
     sudo apt install bash-completion;
 5
     source <(kubectl completion bash | sed s/kubectl/k/g);</pre>
 6
7
     kubectl run --image=kubia --port=8080 kubia;
 8
9
     kubectl get pod kubia -o yaml -n default;
     kubectl describe pod kubia;
10
11
    https://www.cnblogs.com/xiao2/p/16047455.html;
12
     minikube image load kubia;
13
     minikube image build -t <IMAGE_NAME> .;
14
15
```

```
16
     kubectl get nodes;
17
     kubectl apply -f lindyang/deployment.yaml;
18
     kubectl delete -f lindyang/deployment.yaml;
19
     kubectl port-forward kubia-868cd55b98-qzz79 8080:8080;
20
     kubectl expose pod kubia-868cd55b98-qzz79 --type=LoadBalancer --name kubia-
21
    http;
     # minikube 不支持 LoadBalancer;
22
23
    minikube service kubia-http;
24
     kubectl get rc;
     kubectl get services; # src
25
26
    kubectl scale --replicas=3 -f lindyang/deployment.yaml;
27
```

```
apiVersion: v1
 1
    kind: ReplicationController
 2
    metadata:
 3
     name: kubia
 4
 5
    spec:
    mreplicas: 3
 6
 7
           selector:
           app: kubia
 8
9
      template:
10
        metadata:
          name: kubia
11
          labels:
12
            app: kubia
13
        spec:
14
          containers:
15
           - name: kubia
16
17
              image: kubia
18
             imagePullPolicy: Never
19
              ports:
              - containerPort: 8080
20
21
    apiVersion: v1
22
    kind: Service
23
    metadata:
24
      name: kubia
25
    spec:
26
27
    type: LoadBalancer
28
      selector:
        app: kubia
29
```

```
1
    apiVersion: apps/v1
    kind: Deployment
 2
    #kind: ReplicaSet
    metadata:
 4
 5
      name: kubia
      labels:
 6
7
        name: kubia
 8
    spec:
      replicas: 3
9
10
      selector:
11
        matchLabels:
        name: kubia
12
13
      template:
        metadata:
14
          labels:
15
            name: kubia
16
        spec:
17
          containers:
18
            - name: kubia
19
              image: kubia
20
              imagePullPolicy: Never
21
              ports:
22
23
                - containerPort: 8080
```

https://github.com/AliyunContainerService/minikube/wiki

https://www.cnblogs.com/hukey/p/18061513

https://kubernetes.oss-cn-hangzhou.aliyuncs.com/minikube/releases/v1.20.0/minikube-linux-amd64

```
https://github.com/kubernetes/minikube/releases/download/v1.24.0/minikube-
linux-amd64;

minikube delete;
minikube delete --all --purge;
```

```
6
     minikube start --force --driver=docker \
 7
     --image-mirror-country='cn' \
 8
     --registry-mirror=https://docker.mirrors.ustc.edu.cn \
 9
     --image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers
10
11
12
13
     minikube start \
     --kubernetes-version=v1.23.8 \
14
     --image-mirror-country='cn' \
15
     --image-repository='registry.cn-hangzhou.aliyuncs.com/google_containers';
16
17
18
     minikube start --image-mirror-country=cn;
19
20
    minikube kubectl -- create clusterrolebinding system:anonymous \
21
22
     --clusterrole=cluster-admin \
23
     --user=system:anonymous;
24
25
     kubectl proxy \
     --port=8888 \
26
     --address='192.168.1.20' \
27
     --accept-hosts='^.*' >/dev/null 2>&1 &;
28
29
30
     minikube kubectl -- get pods -A;
31
     minikube kubectl -- logs kube-proxy-2z5gg -n kube-system;
32
33
     sudo sysctl -w net.netfilter.nf_conntrack_max=524288;
34
     minikube kubectl -- describe pods .. -n kubernetes-dashboard
35
     # minikube start --disk-size="10g" \
36
    #--image-mirror-country="cn" \
37
38
     #--image-repository="registry.cn-hangzhou.aliyuncs.com/google_containers";
```

Dashbaord 失败

https://blog.csdn.net/jialiang_chen/article/details/140761362

```
1 minikube ssh;
2
3 docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-
    scraper:v1.0.8;
4
5 docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-
    scraper:v1.0.8 \
```

```
docker.io/kubernetesui/metrics-scraper:v1.0.8;
 7
     docker pull registry.cn-
     hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0;
 9
10
     docker tag registry.cn-
     hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0 \
     kubernetesui/dashboard:v2.7.0;
11
12
13
     registry.hub.docker.com/kubernetesui/dashboard:v2.1.0;
14
     registry.hub.docker.com/kubernetesui/metrics-scraper:v1.0.4;
15
```

```
wget https://oss-cdn.nebula-graph.com.cn/package/3.8.0/nebula-graph-
 1
    3.8.0.ubuntu2004.amd64.deb;
    sudo /usr/local/nebula/scripts/nebula.service start all;
 2
 3
    ~/Downloads/nebula-console-linux-amd64-v3.8.0 -addr 127.0.0.1 -port 9669 -u
 4
    root -p dummy -t 3000
    查看集群: 查看集群状态
 5
    查看单个: SHOW HOSTS GRAPH、SHOW HOSTS STORAGE、SHOW HOSTS META;
 6
    ADD HOSTS 127.0.0.1:9779
 7
    心跳: heartbeat_interval_secs
 8
    GO语句采用的是walk类型路径;
 9
    MATCH、FIND PATH和GET SUBGRAPH语句采用的是trail类型路径;
10
11
         表示边的终点;
    $$
12
    $۸
         表示边的起点;
13
         表示管道符前面的查询输出的结果集;
    $-
14
15
    已写入但未构建索引"的数据重建索引,否则无法在MATCH和LOOKUP语句中返回这些数据;
16
    // 重建索引确保能对已存在数据生效。
17
    nebula> REBUILD TAG INDEX player_index_1;
18
19
20
    create edge index follow_index_1 on follow(degree);
    rebuild edge index follow_index_1;
21
22
```

```
1 echo > ~/.minttyrc <<EOF
2 Font=Consolas
3 FontHeight=14
4 EOF
```

sougou_input

```
1 <mark>搜狗输入法</mark>4.0.1可以通过配置文件来实现中文时使用英文标点。
2 vim ~/.config/sogoupinyin/conf/env.ini
3
4 在该配置文件中添加一行
5 DefaultSymbol=0
6
7 然后重新启动输入法即可。
```

postgres

```
sudo -u postgres psql;
林东 1134 1
       ALTER USER postgres WITH PASSWORD 'Mjolnir';
   2
   3
       CREATE DATABASE mydatabase TEMPLATE template0 ENCODING 'UTF8';
   4
       select datid, datname,
   5
           pid,
   6
           usesysid,
           usename,
           client_addr,
  10
        client_port,
  11
           query_start,
  12
           state
  13
           --,query
  14
       FROM pg_stat_activity;
```

```
ps --ppid 2 -p 2 -N -f;
iptables -I INPUT 2 -p tcp -m multiport --dport 80,5000 -j ACCEPT;

3
```

[cookie](https://blog.csdn.net/p312011150/article/details/82179704/)

```
1 proxy_cookie_path / "/; Secure; SameSite=Lax"; # 去掉 Secure
```

Jenkins

```
FROM ubuntu:16.04
 1
     MAINTAINER james@example.com
 2
     ENV REFRESHED AT 2014-06-01
 3
 4
 5
     RUN apt-get update -qq && apt-get install -qqy curl apt-transport-https
     software-properties-common
     #RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
 6
 7
     COPY gpg /tmp/gpg
     RUN cat /tmp/gpg | apt-key add -; rm /tmp/gpg
     RUN apt-get clean && rm -rf /var/lib/apt/lists/*
 9
     #RUN echo deb https://apt.dockerproject.org/repo ubuntu-trusty main >
10
     /etc/apt/sources.list.d/docker.list
     RUN add-apt-repository "deb [arch=amd64]
11
     https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
     RUN apt-get update -qq && apt-get install -qqy --allow-unauthenticated
12
     iptables ca-certificates git-core
     RUN apt-get install -qqy --allow-unauthenticated docker-ce
13
     #RUN apt-get install -qqy --allow-unauthenticated openjdk-9-jdk
14
     RUN add-apt-repository -y ppa:openjdk-r/ppa
15
     RUN apt-get update -qq && apt-get install -qqy openjdk-17-jdk
16
17
     ENV JENKINS_HOME /opt/jenkins/data
18
     #ENV JENKINS_MIRROR http://mirrors.jenkins-ci.org
19
20
21
     RUN mkdir -p $JENKINS_HOME/plugins
22
     #RUN curl -sf -o /opt/jenkins/jenkins.war -L $JENKINS_MIRROR/war-
     stable/latest/jenkins.war
23
     COPY jenkins.war /opt/jenkins/jenkins.war
24
     #RUN for plugin in chucknorris greenballs scm-api git-client git ws-cleanup ;\
25
        do curl -sf -o $JENKINS_HOME/plugins/${plugin}.hpi \
26
             -L $JENKINS_MIRROR/plugins/${plugin}/latest/${plugin}.hpi; done
27
28
     COPY ./*.hpi $JENKINS_HOME/plugins/
29
     ADD ./dockerjenkins.sh /usr/local/bin/dockerjenkins.sh
30
     RUN sed -i 's/^docker_daemon/dockerd/' /usr/local/bin/dockerjenkins.sh
31
     RUN chmod +x /usr/local/bin/dockerjenkins.sh
32
33
     #VOLUME /var/lib/docker
34
35
     EXPOSE 8080
36
37
```

```
38
     # docker daemon & => dockerd
     ENTRYPOINT [ "/usr/local/bin/dockerjenkins.sh" ]
39
40
41
     $ sudo mkdir -p /var/jenkins home
42
     $ cd /var/jenkins_home
43
     $ sudo chown -R 1000 /var/jenkins_home
44
45
46
     docker run --privileged -d \
     -p 8080:8080 \
47
     -v /var/jenkins_home:/opt/jenkins/data \
48
     -v /var/run/docker.sock:/var/run/docker.sock \
49
     --name jenkins \
50
     lindyang/jenkins
51
52
     sudo sed -i 's/https:/http:/'
53
     /var/lib/docker/volumes/data/_data/hudson.model.UpdateCenter.xml;
54
     # http://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json
55
     # https://baijiahao.baidu.com/s?id=1742735186119625775&wfr=spider&for=pc
56
     # https://www.cnblogs.com/yshc/p/10621224.html
57
58
     https://updates.jenkins.io/download/plugins/skip-certificate-check/1.1/skip-
59
     certificate-check.hpi
60
     - Dashboard/Manage Jenkins/插件管理/advanced
61
     - https://www.cnblogs.com/fengwenqian/p/13534786.html
62
63
    http://updates.jenkins.io/update-center.json;
64
     https://gitee.com/Errorcode500/docker-jenkins-sample.git
65
66
     /opt/jenkins/data/jobs/${JOB_NAME}/workspace
67
68
69
     cat > Dockerfile <<EOF
70
     FROM ubuntu:16.04
     RUN echo "while true; do date; echo $WORKSPACE; sleep 1; done" >
71
     /usr/bin/run.sh
     CMD ["/bin/bash", "/usr/bin/run.sh"]
72
73
     EOF
     docker build -t loop_test_$0S .
74
     docker run -d --name loop_test.$0S loop_test_$0S
75
     docker logs loop_test.$0S
76
     docker stop loop_test.$0S
77
     docker rm loop_test.$0S
78
79
     docker rmi loop_test_$0S
80
```

TProv

```
ruby File.exists => File.exist
```

生成证书

```
openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048;
 1
 2
    openssl req -new -key private.key -out request.csr -subj \
 3
    "/C=CN/ST=Beijing/L=Beijing/O=WebRay/OU=IT/CN=cacheproxy.com/emailAddress=ridw
    ard@qq.com"
 5
 6
    openssl x509 -req -days 36135 -in request.csr -signkey private.key -out
    certificate.crt;
 7
    openssl pkey -in private.key -text -noout;
 8
    openssl req -in request.csr -text -noout;
9
10
    openssl x509 -in certificate.crt -text -noout;
```

升级系统

```
1 上传包到 172.21.15.38
2 /var/log/webraydb/Dump/NJ/WebRay/1cf935508eb43c32a229826cf8c1be59
3 在要升级的系统上以 admin 执行
5 # patchup/patchall
6 patchup -u 1cf935508eb43c32a229826cf8c1be59 -p E9DCCB48 \
7 -i ftp://172.21.15.38:10021/fap-cacheproxy-1.0.0-T6-20250211205624-openEuler-x86_64.img
```

Kratos

```
1 kratos proto client api/helloworld/v1/demo.proto -- --go-
http_opt=omitempty=false;
```

Rbenv

```
# https://github.com/Tensho/docker-rbenv/blob/master/Dockerfile
 2
 3
    FROM ubuntu:20.04
 4
 5
    RUN apt-get -yqq update
    RUN DEBIAN_FRONTEND=noninteractive TZ=Asia/Shanghai apt-get -y install tzdata
 6
    RUN apt-get -yqq install build-essential nodejs
 7
    RUN apt-get -yqq install git wget libffi-dev zlib1g-dev libyaml-dev
8
 9
    RUN rm /bin/sh && ln -s /bin/bash /bin/sh
    #RUN wget -q -0 ~/.rbenv/cache/ruby-3.4.1.tar.gz -c https://ftp.ruby-
10
     lang.org/pub/ruby/3.4/ruby-3.4.1.tar.gz;
    RUN git clone --depth=1 https://github.com/rbenv/rbenv.git /root/.rbenv
11
    RUN git clone --depth=1 https://github.com/rbenv/ruby-build.git
12
    /root/.rbenv/plugins/ruby-build
    RUN /root/.rbenv/plugins/ruby-build/install.sh
13
    ENV PATH /root/.rbenv/bin:/root/.rbenv/shims:$PATH
14
    RUN echo 'eval "$(rbenv init -)"' >> /root/.bashrc
15
16
    RUN mkdir -p /root/.rbenv/cache
17
    COPY ruby-3.4.1.tar.gz /root/.rbenv/cache/
    RUN rbenv install 3.4.1
18
    RUN rbenv global 3.4.1
19
    RUN gem sources --remove https://rubygems.org/ --add
20
    https://mirrors.tuna.tsinghua.edu.cn/rubygems/;
21
    RUN gem update --system
    RUN gem install ffi -v 1.17.1
22
    #RUN gem install -s https://gems.ruby-china.com/ jekyll -v 2.5.3
23
    RUN gem install jekyll -v 2.5.3
24
    VOLUME /data
25
    VOLUME /var/www/html
26
    WORKDIR /data
27
28
    ENTRYPOINT ["jekyll", "build", "--destination=/var/www/html"]
29
30
    # gem install faraday-retry
31
```

Uwsig

```
https://github.com/unbit/uwsgi/issues/2117;

ARG uwsgi_dir=/var/lib/gems/2.5.0/gems/uwsgi-2.0.20

RUN mkdir -p $uwsgi_dir/ext/uwsgi

RUN curl -o $uwsgi_dir/ext/uwsgi/install.sh
https://raw.kkgithub.com/unbit/uwsgi/refs/heads/master/install.sh

RUN sed -i 's/curl /curl -k /' $uwsgi_dir/ext/uwsgi/install.sh

RUN apt-get -yqq install python3
```

```
RUN cd $uwsgi_dir/ext/uwsgi; bash install.sh ruby2
$uwsgi_dir/ext/uwsgi/uwsgi.ruby

RUN mkdir -p $uwsgi_dir/bin; cp
$uwsgi_dir/ext/uwsgi/uwsgi_latest_from_installer/bin/uwsgi $uwsgi_dir/bin/;

ENV PATH=$uwsgi_dir/bin:$PATH

RUN uwsgi --version;
```

Docker book

```
https://wangwei1237.github.io/Kubernetes-in-Action-Second-Edition/
 1
 2
 3
     ip link set docker0 up;
    DOCKER_HOST;
 4
 5
     redis-server --protected-mode no;
     docker-compose logs/ps/stop/start/rm/kill;
7
     curl --unix-socket /var/run/docker.sock http://docker/info
 8
9
10
     docker -H tcp://127.0.0.1:2375 images;
```

交叉编译

```
https://github.com/kekeqy/windows-hosted-x86_64-linux-musl-gcc-cross-compiler
https://www.cnblogs.com/nvim/p/18631356
https://juejin.cn/post/7168747781388140580
```

工作整理

待办事项

- https://blog.csdn.net/weixin_40242845/article/details/128400291
- Multiple ssh.Channel status issues
- https://blog.csdn.net/hui1429/article/details/80144863
- 默认的上传下载速度
- 批量修改自动发布根列表
- 用户信息

- 自动创建 policy-data 目录
- 创建 user 表
- SyncStatus, 更新 status, 管理 ssh.Channel
- 根账户(无)
- LoadFromFile 去掉 os.MkdirAll
- 更改用户时的 ssh.Channel 处理, r.CloseSftpReqServers
- 关联状态
- 用户界面.探测器 = &pb
- 日志大小处理
- https://www.gnu.org/software/bash/manual/html_node/Shell-Parameter-Expansion.html

引擎管理器

```
* 引擎名称: 172.18.6.140

* 厂商: WebRAY

* 设备类型: RayGate

* 部署模式: 单机

* 引擎IP: 172.18.6.140

* 协议: https

* 端口: 8443

* 所属引擎组: api引擎组

* 用户名: apiuser

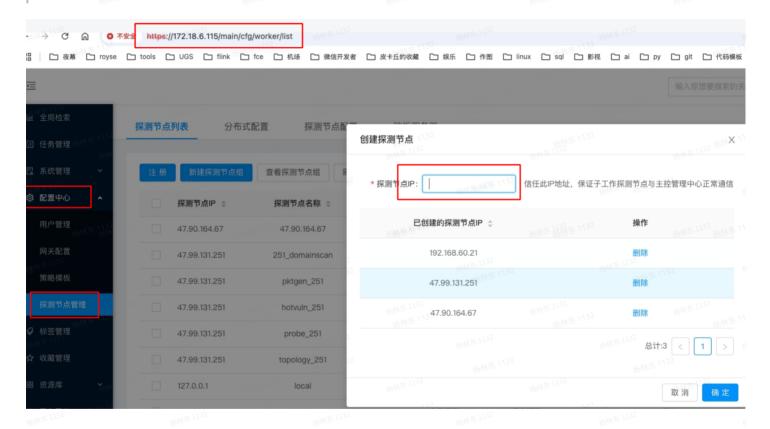
* 密码: admin123!
```

```
代码块
     默认 443 端口
 1
 2
 3
 4
     /api/v1
5
         http://127.0.0.1:89
 6
      /usr/lib/python3.7_venv_raygate/bin/uwsgi \
 7
 8
             --ini /rayos/app/webapp/raygate/uwsgi.ini
 9
         后端,插入数据
10
11
      引擎的日志在 tail -f /var/log/webraydb/workermsg.log;
          2025-04-14 10:33:22,703 INFO: [MainProcess] worker_manage 状态正常 ... [in
12
     worker.py:119]
13
     2025-04-14 11:03:24,468 INFO: [MainProcess] worker_manage 状态正常 ... [in
     worker.py:119]
14
15
    ^/(finger|async|logout|cfg|asset|system|report|logsystem|engine|label|base|acti
16
     <del>∨e)/</del>
     http://127.0.0.1:80
17
18
      <del>/usr/bin/python2 \</del>
19
            /rayos/app/webapp/paladin/run.py
20
21
22
     ^/vm/(vmreport/|download/|css/|js/|img/|signin)
23
24
       http://127.0.0.1:16530
```

```
25
26
    /rayos/app/worker_manage
27
        python3 /RayOS/app/worker_manage/worker.py
28
        上报引擎心跳
29
    引擎
30
        分布式引擎: jobexe
31
        api引擎, 能登录
32
33
34
        python3 /RayOS/app/worker_manage/worker.py
35
36
```

IP段加白,添加引擎

超人/测试123!



蟒蛇

• from urllib.parse import quote

C++

https://blog.csdn.net/youlinhuanyan/article/details/142738063

内存布局: https://www.cnblogs.com/zl1991/p/7040687.html

核心转储

```
代码典 go race condition check
2
3
    # ulimit -c;
 4
    ulimit -c unlimited;
 5
    # %e是可执行文件的名称,%s是信号名,%u是用户ID,%g是组ID,%p是进程ID,%t是时间戳
 6
 7
    cp /etc/sysctl.conf{,.bak};
8
9
    echo 'kernel.core_pattern=/var/log/rlog/cacheproxy/core-%e-%s-%u-%g-%p-%t' \
    >> /etc/sysctl.conf;
10
11
    sysctl -p;
```

公寓

- https://wiki.debian.org/zh_CN/Apt
- https://wiki.debian.org/AptCLI

苏瑞卡塔

Sourceinsight rust

https://linuxcpp.0voice.com/?id=112226

https://zhuanlan.zhihu.com/p/632044134

定时任务

https://segmentfault.com/a/1190000023029219

Docker

```
代码块
1 {
2 "insecure-registries": ["172.21.1.73:8080"],
3 "registry-mirrors": ["http://172.21.1.73:8080"]
4 }
```

代码块

- 1 # docker mirrors
- 2 # 目前国内可用Docker镜像源汇总
- 3 # https://www.coderjia.cn/archives/dba3f94c-a021-468a-8ac6-e840f85867ea

Git

```
代码块
    https://kkgithub.com/
1
    https://hub.yzuu.cf/
2
3
    # 提交时转换为LF, 检出时转换为CRLF
    git config --global core.autocrlf true
5
    # 提交和检出时都不转换
7
    git config --global core.autocrlf false
8
9
    # 提交时转换为LF, 检出时不转换
10
    git config --global core.autocrlf input
11
```

缓存代理

管理环境

```
虚拟平台:
https://172.21.151.12:8006/#v1:0:=qemu%2F104:4:5:::::
root Webray@0808

https://172.21.15.37/user/login
admin: 2025-01-09 15:02:36
页面: admin ABa21@23Fulv1
```

```
代码块

https://172.21.151.12:8006/#v1:0:=qemu%2F104:4:5:::::

root/Webray@0808

172.21.150.109

admin/Webray@12345678

7
```

Env 配置

```
代码块

1 CC=x86_64-linux-musl-gcc
2 CGO_ENABLED=1
3 CGO_LDFLAGS=-02 -g -static
4 GOROOT=C:/Program Files/go
5
```

```
代码块
    func(*Server)
 1
 2
 3
    type Handler func(context.Context, net.Conn) error
     type SFTPMiddleware func(Handler) Handler
 4
 5
     func Chain(m ...SFTPMiddleware) SFTPMiddleware {
 6
         return func(next Handler) Handler {
 7
 8
            for i := len(m) - 1; i >= 0; i-- {
                next = m[i](next)
9
            }
10
       return next
11
12
13
14
     s.middleware = []{
15
16
         Recovery(),
17
         SSHAuth(ss.SSHConfigs()),
18
        WhiteList(ss.GetProbeEgressIP),
         Status(ss.GetProbeStatus),
19
20
         Logging(),
         ConnectionLimit(ss.Limit)
21
22
     }
     next = ServeSFTP(ctx context.Conext, conn net.Conn) error {
23
         s.probe.AppendSftpReqServer(p.Username, channel, connID)
24
25
         defer s.probe.RemoveSftpReqServer(p.Username, connID)
26
         defer server.Close()
         server.Serve()
27
28
29
     go Recovery(
30
         SSHAuth(
            WhiteList(
31
32
                Status(
33
                    go ps.Active()
                    defer func(){
34
35
                         if cnt==0 {
```

```
36
                              go. ps.Logoff()
                         }
37
                     }()
38
                     Logging(
39
                         ConnectionLimit(next)
40
41
                     )
42
43
44
         )
     ) 杨林东11
45
46
     Logging: info user "ridward" logged out
47
     go.ps.Logoff
48
```

```
代码块
    https://blog.csdn.net/Eivene/article/details/140774552
1
2
    http://bbs.itying.com/topic/676cf34b5b798701ddf78072
3
4
5
    https://172.21.150.16/ui/#/host/vms/89
6
    root/Esxi#0926
7
    - 150.39 的账密 root/Superman@12345
8
9
    - proxmox 账户
10
    root/Webray@0808
11
12
    - 删除
13
    点中-more-remove
14
15
    探针: https://172.21.15.37/user/login
16
    admin/ABa2l@23Fulv1
17
18
    系统管理/集中配置管理
19
20
21
    - img 下载
22
    - 外部: http://172.18.9.4:8001/products/fap-cacheproxy/
23
    - 内部: http://172.21.1.25:8000/products/fap-cacheproxy/
24
25
26
    上传 data(ibpd-uat-01), upload,
27
28
    创建系统,
29
    两个盘,128G,256G
30
```

```
31
    2 sockets, 4 cores = 8 cres
32
    16384 Memory MiB(16G)
33
34
    - 登录账号: root/webray++
    - 安装密码: WebRW2r$
35
    输入 y 确认
36
37
    等待好长时间....
38
39
    y; xfs; rdata/cacheproxy; y; roboot;
40
41
    等好久...
42
    admin密码默认aDmin@3.21,首次登录需修改密码,修改后使用新密码登录。Forget@12345678
43
44
45
46
    vlan -A -v MngtVlan -f 172.21.150.107 -m 255.255.255.0;
47
    vlan -S;
48
    route -A -i 0.0.0.0 -m 0.0.0.0 -g 172.21.150.1;
49
    route -S;
50
51
    settime
    help
52
53
    upsshd start
54
55
    [autopost]
56
57
    docker build \
58
    -t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
59
    -f ./dockerfiles/autopost.Dockerfile .;
60
61
    docker save registry.ibdp.webray.com.cn:51808/fap/autopost -o 'autopost.tar';
62
63
64
    [autopost]
65
    - sftp_policy_path: /policy-data
        pull 模式要忽略的远程 server 的策略存储目录
66
        push 不用此配置
67
     root 账户的家目录 HomeDir (所以单导pull模式能同步此目录下的所有文件)
68
69
    - file_path: /home/
        定期清理老文件的目录, NewMonitorRepo 中的的 DataDir:
70
71
        pull 模式下文件同步到的目录
        push 模式不用此配置
72
    - policy_file_path: /home/policy-data
73
74
        pull 模式不用此配置
       push 的时候,本地策略所在文件夹,同时也在检测文件
75
76
77
    [cacheproxy]
```

```
RootAccount() 一致维护 root 用户
78
79
    - data_dir: /home/cacheproxy/data
        探针文件夹创建的目录
80
        策略解压到的目录
81
        定期清理的目录 (NewMonitorRepo)
82
        root 用户的家目录
83
84
    - app_dir: /home/cacheproxy
        创建一个 policy-data-backup 目录备份
85
        cert 所在目录
86
        data.db 所在目录
87
        处理请求的时候看了这个目录的信息??? HandleConn: IsDiskWork()
88
    - policy_data_dir: /home/cacheproxy/data/policy-data
89
        push 过来的策略的存储路径
90
91
        定期检测并需要解压 .tgz
92
    p.HomeDir 是不同 sftpServer 的目录
93
94
95
    SystemProbeRecover
```

```
代码块
     scp -0 -l 1024 /d/... root@172.21.150.28:/cacheproxy/
 1
 2
 3
     # 172.21.150.27
 4
     export GOPROXY="http://172.21.3.11:3000"
 5
     账密: admin/Admin123, 端口是 8000
 6
 7
 8
9
     - cmd/cacheproxy/main.go
     ///go:embed web/*
10
11
12
     - go.mod
    toolchain gol.21.4
13
14
15
     - configs/config.json
    data_dir
16
     app_dir
17
     policy_data_dir
18
19
20
21
     https://github.com/protocolbuffers/protobuf/releases
     # exec: "protoc": executable file not found in %PATH%
22
     https://github.com/protocolbuffers/protobuf/releases/download/v29.2/protoc-
23
     29.2-win64.zip;
```

```
24
25
    VERSION=1.0;
26
    # https://www.jianshu.com/p/10ae8a5a7704
27
    # go get github.com/go-kratos/kratos/cmd/kratos/v2@latest;
28
    go install github.com/go-kratos/kratos/cmd/kratos/v2@latest;
29
30
    kratos proto client ./internal/conf/conf.proto;
31
32
    kratos proto client ./api/cacheproxy/v1/cacheproxy.proto;
33
34
35
    go build -ldflags "-X_main.Version=$(VERSION)";
36
    chmod a+x /usr/local/go/bin/kratos;
37
    go mod tidy;
38
39
    cd cmd/cacheproxy;
    go build;
40
41
     ./cacheproxy -conf ../../configs/config.json;
42
    dlv --listen=:2345 --headless=true --api-version=2 --accept-multiclient exec
     ./cacheproxy
43
    缓存转发的 root passwd 要配置为 root
44
    # ../configs/autopost.yaml 要配置 sftp_address 和 3 个目录
45
46
     ./autopost -conf ../configs/autopost.yaml -mode push
    上传文件到单导的 /cacheproxy/home/policy-data
47
    单导日志类似:
48
    caller=biz/autopost_push.go:203 ... \
49
        policy-data/20241221181530_policy.tgz.part successfully
50
51
    缓存转发的日志类似:
52
53
      caller=biz/monitor.go .... \
        job: /home/cacheproxy/data/policy-data/20241221181530_policy.tgz
54
55
    文件会被解压到 /home/cacheproxy/data 下
56
57
        - 0XSDJI902
58
        - 0XSDJI905
59
        - 0XSDJI908
      - E04C81F400
60
        - SXXAtest11
61
    /home/cacheproxy/data/policy-data
62
    /hoem/cacheproxy/policy-data(??? 是什么)
63
```

走

• 专业bu协议: https://www.cnblogs.com/zhanchenjin/p/17454978.html

```
1 truntime.gcdata: missing Go type information for global symbol .dynsym: size 72
     https://www.cnblogs.com/binHome/p/13020178.html
 2
 3
 4
     https://www.jianshu.com/p/a73cef45b65c
 5
     wire:
 6
 7
     https://www.cnblogs.com/jiujuan/p/16136633.html
8
     在 service/service.go
 9
     type AppRepo interface {
10
11
12
     }
13
14
     在 data/data.go
15
     type appRepo struct {
16
         *mq
      *db
17
18
         *dp
19
     }
20
     func NewAppRepo() service.AppRepo {
21
22
         appRepo := &appRepo{}
         return appRepo
23
24
     } 杨林芬
25
26
     //
27
     google.golang.org/protobuf/types/know/anypb
28
29
     marshal := v1.Flow { Enable: true, Subset: "127.0.0.1" }
     any, err := anypb.New(&marshal);
30
     [type.googleapis.com/engmgr.v1.Flow]:{enable:true, subset:"127.0.0.1"}
31
32
     grpcurl -plaintext -H "x-user-id: 1" 172.21.150.109:9000 list;
33
34
     grpcurl -plaintext -H "x-user-id:1" -d '{}' 172.21.150.109
     engmgr.v1.App.CreateGrp;
```

```
代码块

1 #文档
2 https://www.topgoer.com/go%E5%9F%BA%E7%A1%80/%E7%BB%93%E6%9E%84%E4%BD%93.html
3 # 自定义 json
5 https://juejin.cn/post/6844904184651874312
```

```
代码埃 select
    https://www.cnblogs.com/qcy-blog/p/18520091
    # docker copy
 3
    https://developer.baidu.com/article/details/3214252
 4
 5
    https://blog.csdn.net/u011461385/article/details/106017483
 6
 7
    # chan
    https://blog.csdn.net/chenchongg/article/details/86589395
8
 9
    # nil, close chan
    https://dave.cheney.net/2014/03/19/channel-axioms
10
11
    # strace
    // 只显示 renameat 的调用情况
12
    strace -f -e trace=renameat go run ./script/download-go go1.22.3
13
14
    # switch语句也可以与false字面值一起使用,提供了一种确定哪些条件未满足的方法
15
16
    https://www.cnblogs.com/cheyunhua/p/17945018
```

```
代码块

1 for pos, char := range "日本\x80語" { // \x80 is an illegal UTF-8 encoding
2 fmt.Printf("character %#U starts at byte position %d\n", char, pos)
3 }
4
5
```

```
代码块
 1
    the module github.com/google/go-cmp contains a package in the directory cmp/.
    That package's import path is github.com/google/go-cmp/cmp.
 2
     Packages in the standard library do not have a module path prefix
 3
 4
    go install example/user/hello;
 5
 6
7
    The install directory is controlled by the GOPATH and GOBIN environment
    variables.
    If GOBIN is set, binaries are installed to that directory.
10
    If GOPATH is set, binaries are installed to the bin subdirectory of
    the first directory in the GOPATH list.
11
     Otherwise, binaries are installed to the bin subdirectory of
12
     the default GOPATH ($HOME/go or %USERPROFILE%\go)
13
14
```

```
15
      go env -u GOBIN;
 16
      $(dirname $(go list -f '{{.Target}}' .));
 17
 18
      go help importpath;
 19
 20
 21
      GOCACHE: $HOME/.cache/go-build;
 22
 23
      import "example/user/hello/morestrings";
 24
      go mod tidy;
 25
 26
 27
      $HOME/.mygo/pkg/mod/
 28
 29
      go clean -modcache;
```

```
代码块
    go list -f '{{.Target}}';
     go env -w GOBIN=/path/to/your/bin;
 2
 3
    Named return values;
     A var statement can be at package or function level.
    := 只可以在函数内使用.
     rune is int32;
 7
     uintptr;
 8
 9
10
     %T, %v
     Zero values;
11
12
     different type requires an explicit conversion;
13
     Gos switch cases need not be constants, and the values involved need not be
     integers;
     Unlike C, Go has no pointer arithmetic.;
14
15
     type Vertex struct {
16
           X int
17
      杨琳东1132 Y int
18
19
     }
20
21
     (*p).X -> p.X;
22
23
     var (
           v1 = Vertex{1, 2} // has type Vertex
24
      v2 = Vertex{X: 1} // Y:0 is implicit
25
            v3 = Vertex{} // X:0 and Y:0
26
            p = &Vertex{1, 2} // has type *Vertex
27
```

```
28
29
    Slices are like references to arrays;
30
31
    a := make([]int, 5) // len(a)=5;
32
    b := make([]int, 0, 5) // len(b)=0, cap(b)=5;
33
    strings.Join;
34
    The range form of the for loop iterates over a slice or map.
35
    for i := range pow;
36
37
    delete(m, key)
38
    elem, ok = m[key]
39
    The make function returns a map of the given type, initialized and ready for
40
    use;
41
42
    func fibonacci() func() int {
43
44
            i, j := 0, 1
           return func() int {
45
46
                   i, j = j, i+j
47
                   return i
           } 杨林东 1132
48
49
    }
50
51
    Methods:
    The receiver appears in its own argument list
52
    between the func keyword and the method name.
53
    func (v Vertex) Abs() float64 {
54
           return math.Sqrt(v.X*v.X + v.Y*v.Y)
55
    }
56
57
58
    You can only declare a method with a receiver
59
    whose type is defined in the same package as the method.
60
61
    Pointer receivers;
62
    Go interprets the statement v.Scale(5) as (&v).Scale(5);
63
    p.Abs() is interpreted as (*p).Abs();
64
65
    type Abser interface {
66
         Abs() float64
67
    }
68
69
    func (t *T) M() {
70
     71
72
                   fmt.Println("<nil>")
73
                   return
```

```
74
            fmt.Println(t.S)
 75
 76
 77
     # Type assertions
78
     t := i.(T);
 79
     t, ok := i.(T);
 80
81
 82
     switch v := i.(type);
83
 84
     type Stringer interface {
     String() string
 85
 86
 87
     func (p Person) String() string {
88
      return fmt.Sprintf("%v (%v years)", p.Name, p.Age)
 89
     } 杨林东1132
 90
 91
 92
     type error interface {
       Error() string
 93
 94
     }
95
 96
     %q;
97
98
     func Index[T comparable](s []T, x T) int
99
100
     type List[T any] struct {
101
        next *List[T]
            val T
102
     }
103
104
     The evaluation of f, x, y, and z happens in the current goroutine and
105
     the execution of f happens in the new goroutine.
106
107
108
109
     ch <- v // Send v to channel ch.
     v := <-ch // Receive from ch, and
110
              // assign value to v.
111
112
113
     Like maps and slices, channels must be created before use:
114
115
     ch := make(chan int);
116
     ch := make(chan int, 100);
117
118
119
     v, ok := <-ch;
120
```

```
121
      Only the sender should close a channel, never the receiver.
      Sending on a closed channel will cause a panic.
122
123
124
125
      select {
      case c <- x:
126
127
             x, y = y, x+y
128
      case <-quit:</pre>
129
             fmt.Println("quit")
130
             return
131
      }
132
133
      mu sync.Mutex;
134
      c.mu.Lock();
135
      c.mu.Unlock();
```

安全外壳协议

/rayos/app/config-ssh/scli/upsshd start

pg_dump

```
代码块

1 PGPASSWORD='admin1234' pg_dump -d fusion -U postgres --no-comments \
2 --column-inserts \
3 -t 'rule_ele|model';
4
5
6 -0 --no-owner
7 -s --schema-only
```

pg

```
代码块

1 /install/modules/ray-postgresql/stop.sh;

2 psql postgresql://user:pass@host:port/dbname

3 PGPASSWORD='admin1234' psql -U postgres -p 5432 fusion;

4 
5 echo "host ti_graph {user} {ip} md5" >> $PGDATA/pg_hba.conf

6 
7 
8 
9 PGPASSWORD='Mjolnir' pg_dump -d test -U postgres -h 127.0.0.1 --no-comments \
10 --column-inserts \
```

```
11
    --schema-only \
    -t 'group' \
12
    -t 'device_group' \
13
    -t 'device' \
14
    -t 'engine' \
15
    > enginemgr.sql;
16
17
18
19
    -s --schema-only,加上只导表结构
```

iptables防火墙

```
代码块

1 /rayos/cfg/iptables_port.conf 6379 5432

2 /rayos/utils/netac -A
```

ip地址

代码块

sudo systemctl restart systemd-resolved.service;

代码块

- flock -xn /var/run/custom_worker.lck -c \

代码块

```
SELECT col.table_name, col.column_name, col.data_type, des.description
FROM information_schema.columns AS col

JOIN pg_statio_all_tables AS st
ON col.table_name=st.relname

LEFT JOIN pg_description AS des
ON (st.relid=des.objoid AND col.ordinal_position=des.objsubid)
WHERE col.table_name = 'document';
```

limits.conf/sysctl.conf

代码块

- 1 https://blog.csdn.net/qisianla/article/details/52383426
- 2 https://www.cnblogs.com/netsa/p/15385635.html

Pycharm

代码块

1 remote debug, /etc/ssh/sshd_config 中 AllowTcpForwarding yes

防火墙

代码块

- 1 [Unit]
- 2 AssertPathExists=/etc/sysconfig/iptables
- 3 iptables -I INPUT 2 -p tcp -m multiport --dport 80,5000,10290 -j ACCEPT;

Es

```
代码块
    network.host: 0.0.0.0
1
    discovery.seed_hosts: ["127.0.0.1"]
2
 3
4
    network.bind_host
    network.public_host
    network.host
6
    discovery.seed_hosts: 集群主机列表
7
    discovery.seed_providers: 基于配置文件配置集群主机列表
8
9
    cluster.initial_master_nodes: 启动时初始化的参与选主的node, 生产环境必填
10
11
    curl -K https://elastic:elastic@172.21.15.234/_settings?pretty
12
```

alembic 使用

```
cd fusion_app/module/resources/webapp/rayfusion;
mkdir misc;
cd misc;
alembic init db; # create alembic.ini and dir db/
```

```
代码块
     apt-get download git;
 1
     ssh-copy-id -i ~/.ssh/id_*.pub root@172.21.15.234;
 2
 3
    useradd -m -s /bin/bash lindyang;
     passwd lindyang;
 4
 5
6
    cp /etc/yum.conf{,.bak};
     #echo 'proxy=http://172.21.3.11:3128' >> /etc/yum.conf;
 7
 8
     echo "$USER $HOSTNAME= NOPASSWD: ALL" | sudo tee /etc/sudoers.d/$USER;
 9
10
    cat /etc/os-release
11
    cat /etc/yum.repos.d/openEuler.repo
12
```

sudo apt install zlib1g zlib1g-dev;

sudo apt-get install -y git make build-essential libssl-dev zlib1g-dev libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev libncursesw5-dev xz-utils tk-dev libffi-dev; sudo apt install liblzma-dev;

v=3.9.19; wget https://repo.huaweicloud.com/python/\$v/Python-\$v.tar.xz -P /opt/.pyenv/cache/ --no-check-certificate; pyenv install \$v;

Virtualbox:

适用于 Visual Studio 2019(版本 16.11)的 Visual C++ 可再发行组件

- 1. sudo groupadd docker
- 2. sudo usermod -aG docker \$用户
- 3. 【Relogin shell】 or 【switch to root and then switch to the current user】

Docker 镜像

https://cr.console.aliyun.com/cn-hangzhou/instances/mirrors

```
代码块

1 ARG DEBIAN_FRONTEND=noninteractive

2 ENV TZ=Asia/Shanghai

3 RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y install tzdata;

4 https://github.com/Tensho/docker-rbenv;
```

阿里云镜像查找

https://developer.aliyun.com/mirror/

```
代码块
     # 将可用源进行配置
    sudo tee /etc/docker/daemon.json <<EOF</pre>
 3
 4
         "registry_mirrors": [
             "https://dockerproxy.cn",
 5
             "https://docker_rainbond.cc",
             "https://docker.udayun.com",
 7
             "https://docker.211678.top"
 9
    } 101
10
     EOF
11
12
     # 重载,重启Docker服务
13
     sudo systemctl daemon-reload
14
15
     sudo systemctl restart docker
```

迷你库贝

https://blog.csdn.net/Sindweller5530/article/details/116499761

https://github.com/luksa/kubernetes-in-action/tree/master/Chapter04

• 自动命令文件类型yaml设置本地制表位=2 shiftwidth=2 expandtab自动缩进

```
1 minikube service ingress-nginx-controller -n ingress-nginx;
```

```
--image-mirror-country='cn' \
         --image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers;
 5
 6
 7
     minikube status;
     minikube dashboard;
 8
     minikube ssh;
 9
     eval $(minikube docker-env);
10
11
12
     kubectl cluster-info;
13
     kubectl get nodes;
     kubectl describe node minikube
14
15
     minikube image load kubia;
16
     minikube ssh;
17
     docker tag kubia luksa/kubia:1.0;
18
19
     kubectl create deployment kubia --image=luksa/kubia:1.0;
20
21
     kubectl get deployments;
22
     kubectl get pods;
23
     kubectl describe pod;
24
25
     kubectl expose deployment kubia --type=LoadBalancer --port 8080;
26
27
     kubectl get svc;
28
     k get nods/pods/deployments/pods;
29
30
31
     kubectl api-resources;
32
33
     # If you use Minikube to create the cluster,
     # no load balancer is created
34
35
     minikube service kubia --url;
36
37
     minikube ip;
38
     kubectl scale deployment kubia --replicas=3;
39
40
     kubectl get deploy;
41
42
     kubectl get pods -o wide;
43
     kubectl get nodes minikube -o yaml;
44
45
46
     kubectl proxy;
47
     http://127.0.0.1:8001/api/v1/nodes/minikube;
48
49
     kubectl explain nodes;
     kubectl explain node.spec; # --api-version
50
```

```
51
    kubectl explain pods --recursive;
52
    kubectl get ev -o wide;
53
    kubectl get ev --field-selector type=Warning;
54
55
56
    kubectl explain events;
57
    kubectl run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml >
58
    mypod.yaml;
59
    kubectl apply -f kubia.yaml;
60
61
    k run kubia --image=luksa/kubia:1.0 --dry=run=client -o yaml;
62
63
    k describe pod kubia; minikube ssh; curl 172.17.0.X:8080;
64
65
    Some of the pod's fields aren't mutable, so the update may fail,
66
67
    but you can always delete the pod and then create it again
68
    kubectl get pod kubia -o wide;
69
70
    kubectl run \
71
72
    --image=ubuntu:latest -it \
73
    --restart=Never \
    --rm client-pod curl 172.17.0.7:8080;
74
75
    kubectl port-forward kubia 8080:8080;
76
77
    kubectl port-forward --help;
    kubectl logs --timestamps=true -f kubia;
78
79
        --since=2m
      --since-time=2020-02-01T09:50:00Z
80
        --tail=10
81
        -p (--previous)
82
83
84
    kubectl exec kubia -- ps aux;
85
86
    kubectl attach;
87
    kubectl port-forward kubia-ssl 8080 8443 9901;
88
    curl https://example.com:8443 --resolve example.com:8443:127.0.0.1 -k;
89
90
91
    kubectl delete po kubia-ssl --grace-period 10;
        spec.terminationGracePeriodSeconds
92
        metadata.deletionGracePeriodSeconds
93
94
95
96
    kubectl logs kubia-ssl -c kubia;
```

```
97
      kubectl logs kubia-ssl --all-containers;
 98
      kubectl exec -it kubia-ssl -c envoy -- bash;
99
100
      k delete pod kubia --wait=false;
101
102
      kubectl delete -f kubia-ssl.yaml;
103
      kubectl delete -f kubia.yaml,kubia-ssl.yaml;
104
105
      kubectl apply -f Chapter05/;
106
      kubectl delete -f Chapter05/;
107
         --recursive
108
      kubectl delete po --all;
109
110
     kubectl delete all --all;
111
112
     kubectl delete events,all --all;
113
114
115
      kubectl exec kubia-liveness -c envoy -- tail -f /var/log/envoy.admin.log;
116
117
      kubectl logs kubia-liveness -c envoy -p;
```

```
代码块
 1
    # NodePort
    kubectl cluster-info;
 3
    curl http://192.168.49.2:30123;
 4
    kubectl get nodes -o jsonpath='{.items[*].status.addresses}';
 5
     kubectl get nodes -o \
6
     jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}';
7
8
9
     # http://kt1bernetes.io/docs/user-guide/jsonpath
10
     minikube service kubia-nodeport;
11
     # minikube sevrvice <service-name> [-n <namespace>]
12
13
14
    externalTrafficPolicy: Local;
15
    minikube addons list;
16
    minikube addons enable ingress;
17
18
     apiVersion: networking.k8s.io/v1;
```

```
代码块
    apiVersion: networking.k8s.io/v1
 2
    kind: Ingress
    metadata:
 3
       name: kubia
4
 5
    spec:
 6
       rules:
 7
       - host: kubia.example.com
 8
         http:
           paths:
9
10
           - path: /
             pathType: Prefix
11
             backend:
12
13
               service:
                 name: kubia-nodeport
14
15
                 port:
                   number: 80
16
```

```
代码块
    docker build -t kubia .;
1
 2
    删除 DaemonSet 也会删除 pod;
 3
    # job 资源
4
5
    kubectl get jobs;
 6
7
    # completions: 5
8
    # parallelism: 2
9
    # 有问题
10
    kubectl scale job multi-completion-batch-job --replicas 3;
11
12
13
    activeDeadlineSeconds
14
    spec.backoffLimit=6 (default)
15
    CronJob
16
17
    分钟 小时 每月中的第几天 月 星期几(⊙是星期天)
    startingDeadlineSeconds
18
19
20
21
    创建 service 后,需要
22
    minikube ssh; # 登录
```

```
curl `kubectl get svc | awk '{print $3}'` # CLUSTER-IP
23
24
25
    kubectl exec kubia-manual -- curl -s http://10.99.94.217; # 没有 -s, -- 是非必须
26
    sessionAffinity: ClientIP; # 不支持http_cookie
27
    kubectl exec kubia-manual -- env;
28
    dnsPolicy;
29
30
31
    backend-database.default.svc.cluster.local;
32
    kubectl exec -it kubia-manual -- bash;
33
    curl -s http://kubia.default.svc.cluster.local;
34
    curl http://kubia;
35
36
    cat /etc/resolv.conf;
37
    ping kubia; # 无法ping通, 因为是虚拟IP, 只有与port结合才有意义
38
    kubectl describe svc kubia | grep -i endpoint;
39
40
    kubectl get endpoints kubia;
    curl http://external-service/echo;
41
42
    # kubia ExternalName
43
```

```
代码块
    kubectl logs mypod --previous;
    kubectl edit rc kubia;
2
3
    export KUBE_EDITOR="/usr/bin/vim";
4
5
    export EDITOR="/usr/bin/vim";
    kubectl scale rc kubia --replicas=4;
6
    kubectl delete rc kubia --cascade=false;
7
8
    ReplicaSet 匹配缺少某个标签的 pod, 或包含某个标签名的 pod, 不管其值.
9
10
    kubectl api-versions;
11
    replicaset.apps/kubia created;
12
13
    kubectl get rs;
    kubectl delete rs kubia;
14
15
    # 使用 DaemonSet 在每个节点上运行一个 pod
16
    # 使用。DaemonSet 只在特定的节点运行 pod
17
    # 节点可以被设置为不可调度,但DaemonSet可以绕过它
18
```

```
代码块
    apiVersion: apps/v1
    kind: ReplicaSet
 2
    metadata:
 3
      name: kubia
 5
    spec:
      replicas: 3
      selector:
 7
8
        matchLabels:
         app: kubia
9
10
      template:
        metadata:
11
         labels:
12
13
      app: kubia
        spec:
14
15
         containers:
         - name: kubia
16
          image: kubia
17
          imagePullPolicy: Never
18
```

• 匹配表达式

```
代码块

1 selector:
2 matchExpressions:
3 - key: app
4 operator: In
5 values:
6 - kubia
```

操作员

- 在
- 不在
- 存在
- 不存在

```
1 apiVersion: v1
 2
     kind: Pod
 3
     metadata:
       name: kubia-liveness
 4
 5
     spec:
 6
       containers:
       - image: kubia-unhealthy
 7
         imagePullPolicy: Never
8
 9
         name: kubia
       livenessProbe:
10
11
          httpGet:
            path: /
 12
            port: 8080
13
           initialDelaySeconds: 15
14
```

```
代码块
    - pod 共享 network, UTS, IPC, PID(默认未开启)
1
    - 文件系统隔离, 但是可以通过 Volume 共享文件目录
2
    - localhost 可以与同一 pod 中的其它容器通信
3
    kubectl get po kubia-82d24 -o yaml;
4
    kubectl explain pods;
5
    kubectl explain pod.spec;
6
    kubectl create -f lindyang/kubia-manual.yaml;
7
8
    kubectl describe pod kubia-manual | grep 172;
    minikube ssh;
10
    curl 172.17.0.7:8080;
11
12
13
    kubectl logs kubia-manual -c kubia;
14
    # 不通过 service 与 pod 通讯
15
16
    kubectl port-forward kubia-manual 8888:8080;
    curl http://127.0.0.1:8888;
17
18
    kubectl get pods --show-labels;
19
    kubectl get pod -L creation_method,env;
20
21
    kubectl label po kubia-manual creation_method=manual --overwrite
22
23
    # 删除标签, -
    kubectl label po kubia-manual createion_method-
24
25
    kubectl get po -l creation_method=manual;
26
27
28
    kubectl get po -l env;
```

```
29
     kubectl get po -l '!env';
30
     env!=prod
31
32
     'env in (prod,dev)'
     'env notin (prod,dev)'
33
34
35
     kubectl label node minikube gpu=false;
36
37
     kubectl annotate pod kubia-manual mycompany.com/someannotation="foo bar";
38
     kubectl get po --namespace kube-system;
39
40
41
     kubectl get ns;
42
43
     kubectl delete namespace custom-namespace;
44
     kubectl create namespace custom-namespace;
     kubectl create -f lindyang/kubia-manual.yaml -n custom-namespace;
45
46
     命名空间不允许包含点号.
47
48
49
     alias kcd='kubectl config set-context $(kubectl config current-context) --
     namespace'
50
51
     kubectl delete pod po1 po2;
52
     kubectl delete po -l creation_method=manual;
53
54
55
     kubectl delete ns custom-namespace;
56
57
     kubectl delete po --all;
58
59
     kubectl delete all --all;
```

命名空间

```
代码块

1 apiVersion: v1
2 kind: Namespace
3 metadata:
4 name: custom-namespace
```

```
兴码域piVersion: v1
     kind: Pod
 2
 3
     metadata:
       name: kubia-gpu
 4
 5
     spec:
       nodeSelector:
 6
         gpu: "true"
 7
8
       containers:
 9
       - name: kubia
       image: kubia
10
11
         imagePullPolicy: Never
```

```
代码块
1 minikube status;
2
    minikube kubectl -- get pods -A;
    alias kubectl="minikube kubectl --";
    minikube kubectl cluster-info;
    sudo apt install bash-completion;
    source <(kubectl completion bash | sed s/kubectl/k/g);</pre>
6
7
    kubectl run --image=kubia --port=8080 kubia;
8
9
    kubectl get pod kubia -o yaml -n default;
    kubectl describe pod kubia;
10
11
    https://www.cnblogs.com/xiao2/p/16047455.html;
12
    minikube image load kubia;
13
    minikube image build -t <IMAGE_NAME> .;
14
15
16
    kubectl get nodes;
17
    kubectl apply -f lindyang/deployment.yaml;
18
19
    kubectl delete -f lindyang/deployment.yaml;
    kubectl port-forward kubia-868cd55b98-qzz79 8080:8080;
20
21
    kubectl expose pod kubia-868cd55b98-qzz79 --type=LoadBalancer --name kubia-
    http;
22
    # minikube 不支持 LoadBalancer;
23
    minikube service kubia-http;
    kubectl get rc;
24
25
    kubectl get services; # src
26
    kubectl scale =-replicas=3 -f lindyang/deployment.yaml;
27
```

```
代码块
    apiVersion: v1
    kind: ReplicationController
 2
 3
    metadata:
      name: kubia
4
 5
    spec:
      replicas: 3
 6
 7
           selector:
 8
           app: kubia
9
      template:
        metadata:
10
          name: kubia
11
12
          labels:
      app: kubia mas 1132
13
        spec:
14
15
          containers:
            - name: kubia
16
              image: kubia
17
              imagePullPolicy: Never
18
19
              ports:
20
             - containerPort: 8080
21
22
    apiVersion: v1
    kind: Service
23
24
    metadata:
25
      name: kubia
26
    spec:
      type: LoadBalancer
27
      selector:
28
29
        app: kubia
30
      ports:
      - port: 8080
31
32
        targetPort: 8080
```

代码块

6

```
1 apiVersion: apps/v1
2 kind: Deployment
3 #kind: ReplicaSet
4 metadata:
5 name: kubia
```

labels:

```
name: kubia
 8
     spec:
 9
       replicas: 3
10
       selector:
         matchLabels:
11
         name: kubia
12
13
       template:
         metadata:
14
15
           labels:
16
             name: kubia
17
         spec:
           containers:
18
             - name: kubia
19
               image: kubia
20
               imagePullPolicy: Never
21
22
               ports:
                 - containerPort: 8080
23
```

https://github.com/AliyunContainerService/minikube/wiki

https://www.cnblogs.com/hukey/p/18061513

https://kubernetes.oss-cn-hangzhou.aliyuncs.com/minikube/releases/v1.20.0/minikube-linux-amd64

```
代码块
    https://github.com/kubernetes/minikube/releases/download/v1.24.0/minikube-
     linux-amd64;
 2
    minikube delete;
3
    minikube delete --all --purge;
 4
 5
 6
 7
     minikube start --force --driver=docker \
     --image-mirror-country='cn' \
 8
     --registry-mirror=https://docker.mirrors.ustc.edu.cn \
 9
10
     --image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers
11
12
13
     minikube start \
     --kubernetes-version=v1.23.8 \
14
15
     --image-mirror-country='cn' \
16
     --image-repository='registry.cn-hangzhou.aliyuncs.com/google_containers';
17
18
```

```
19
     minikube start --image-mirror-country=cn;
20
     minikube kubectl -- create clusterrolebinding system:anonymous \
21
     --clusterrole=cluster-admin \
22
     --user=system:anonymous;
23
24
25
    kubectl proxy \
    --port=8888 \
26
27
     --address='192.168.1.20' \
     --accept-hosts='^.*' >/dev/null 2>&1 &;
28
29
30
     minikube kubectl -- get pods -A;
31
     minikube kubectl -- logs kube-proxy-2z5gg -n kube-system;
32
     sudo sysctl -w net.netfilter.nf_conntrack_max=524288;
33
34
     minikube kubectl -- describe pods .. -n kubernetes-dashboard
35
36
     # minikube start --disk-size="10g" \
37
     #--image-mirror-country="cn" \
     #--image-repository="registry.cn-hangzhou.aliyuncs.com/google_containers";
38
```

Dashbaord 失败

https://blog.csdn.net/jialiang_chen/article/details/140761362

```
代码块
    minikube ssh;
    docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-
     scraper:v1.0.8;
 4
     docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-
 5
     scraper:v1.0.8 \
     docker.io/kubernetesui/metrics-scraper:v1.0.8;
 6
 7
     docker pull registry.cn-
 8
     hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0;
9
     docker tag registry.cn-
10
     hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0 \
11
     kubernetesui/dashboard:v2.7.0;
12
13
     registry.hub.docker.com/kubernetesui/dashboard:v2.1.0;
14
     registry.hub.docker.com/kubernetesui/metrics-scraper:v1.0.4;
15
```

```
代码块
    wget https://oss-cdn.nebula-graph.com.cn/package/3.8.0/nebula-graph-
    3.8.0.ubuntu2004.amd64.deb;
    sudo /usr/local/nebula/scripts/nebula.service start all;
 3
    ~/Downloads/nebula-console-linux-amd64-v3.8.0 -addr 127.0.0.1 -port 9669 -u
    root -p dummy -t 3000
    查看集群: 查看集群状态
 5
    查看单个: SHOW HOSTS GRAPH、SHOW HOSTS STORAGE、SHOW HOSTS META;
 6
    ADD HOSTS 127.0.0.1:9779
7
    心跳: heartbeat_interval_secs
8
    GO语句采用的是walk类型路径;
9
    MATCH、FIND PATH和GET SUBGRAPH语句采用的是trail类型路径;
10
11
    $$
         表示边的终点;
12
    $1
       表示边的起点;
13
         表示管道符前面的查询输出的结果集;
14
15
    已写入但未构建索引"的数据重建索引,否则无法在MATCH和LOOKUP语句中返回这些数据;
16
    // 重建索引确保能对已存在数据生效。
17
    nebula> REBUILD TAG INDEX player_index_1;
18
19
    create edge index follow_index_1 on follow(degree);
20
    rebuild edge index follow_index_1;
21
22
```

Windows git bash设置字体大小

代码块

- 1 echo > ~/.minttyrc <<EOF</pre>
- 2 Font=Consolas
- 3 FontHeight=14
- 4 EOF

搜狗输入法

代码块

- 1 搜狗输入法4.0.1可以通过配置文件来实现中文时使用英文标点。
- 2 vim ~/.config/sogoupinyin/conf/env.ini

3

```
4 在该<mark>配置文件</mark>中添加一行
5 DefaultSymbol=0
6 然后重新启动输入法即可。
```

postgres

```
代码块
    sudo -u postgres psql;
    ALTER USER postgres WITH PASSWORD 'Mjolnir';
    CREATE DATABASE mydatabase TEMPLATE template0 ENCODING 'UTF8';
 4
    select datid, datname,
5
         pid,
 6
 7
        usesysid,
 8
         usename,
9
         client_addr,
        client_port,
10
         query_start,
11
12
         state
13
         --,query
     FROM pg_stat_activity;
14
```

```
代码块

1 ps --ppid 2 -p 2 -N -f;

2 iptables -I INPUT 2 -p tcp -m multiport --dport 80,5000 -j ACCEPT;

3
```

[cookie](https://blog.csdn.net/p312011150/article/details/82179704/)

```
1 proxy_cookie_path / "/; Secure; SameSite=Lax"; # 去掉 Secure
```

Jenkins

```
代码块

1 FROM ubuntu:16.04

2 MAINTAINER james@example.com

3 ENV REFRESHED_AT 2014-06-01
```

```
RUN apt-get update -qq && apt-get install -qqy curl apt-transport-https
     software-properties-common
     #RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
 6
     COPY gpg /tmp/gpg
 7
     RUN cat /tmp/gpg | apt-key add -; rm /tmp/gpg
 8
     RUN apt-get clean && rm -rf /var/lib/apt/lists/*
9
     #RUN echo deb https://apt.dockerproject.org/repo ubuntu-trusty main >
10
     /etc/apt/sources.list.d/docker.list
11
     RUN add-apt-repository "deb [arch=amd64]
     https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
     RUN apt-get update -qq && apt-get install -qqy --allow-unauthenticated
12
     iptables ca-certificates git-core
     RUN apt-get install -qqy --allow-unauthenticated docker-ce
13
     #RUN apt-get install -qqy --allow-unauthenticated openjdk-9-jdk
14
15
     RUN add-apt-repository -y ppa:openjdk-r/ppa
16
     RUN apt-get update -qq && apt-get install -qqy openjdk-17-jdk
17
18
     ENV JENKINS_HOME /opt/jenkins/data
     #ENV JENKINS_MIRROR http://mirrors.jenkins-ci.org
19
20
21
     RUN mkdir -p $JENKINS_HOME/plugins
22
     #RUN curl -sf -o /opt/jenkins/jenkins.war -L $JENKINS_MIRROR/war-
     stable/latest/jenkins.war
     COPY jenkins.war /opt/jenkins/jenkins.war
23
24
     #RUN for plugin in chucknorris greenballs scm-api git-client git ws-cleanup ;\
25
     # do curl -sf -o $JENKINS_HOME/plugins/${plugin}.hpi \
26
            -L $JENKINS_MIRROR/plugins/${plugin}/latest/${plugin}.hpi ; done
27
     COPY ./*.hpi $JENKINS_HOME/plugins/
28
29
30
     ADD ./dockerjenkins.sh /usr/local/bin/dockerjenkins.sh
     RUN sed -i 's/^docker daemon/dockerd/' /usr/local/bin/dockerjenkins.sh
31
     RUN chmod +x /usr/local/bin/dockerjenkins.sh
32
33
34
     #VOLUME /var/lib/docker
35
     EXPOSE 8080
36
37
     # docker daemon & => dockerd
38
     ENTRYPOINT [ "/usr/local/bin/dockerjenkins.sh" ]
39
40
41
42
     $ sudo mkdir -p /var/jenkins_home
     $ cd /var/jenkins_home
43
     $ sudo chown -R 1000 /var/jenkins_home
44
45
46
     docker run --privileged -d \
```

```
47
     -p 8080:8080 \
     -v /var/jenkins_home:/opt/jenkins/data \
48
     -v /var/run/docker.sock:/var/run/docker.sock \
49
     --name jenkins \
50
    lindyang/jenkins
51
52
     sudo sed -i 's/https:/http:/'
53
     /var/lib/docker/volumes/data/_data/hudson.model.UpdateCenter.xml;
54
     # http://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json
55
     # https://baijiahao.baidu.com/s?id=1742735186119625775&wfr=spider&for=pc
56
     # https://www.cnblogs.com/yshc/p/10621224.html
57
58
     https://updates.jenkins.io/download/plugins/skip-certificate-check/1.1/skip-
59
     certificate-check.hpi
60
     - Dashboard/Manage Jenkins/插件管理/advanced
61
62
     - https://www.cnblogs.com/fengwenqian/p/13534786.html
63
64
     http://updates.jenkins.io/update-center.json;
     https://gitee.com/Errorcode500/docker-jenkins-sample.git
65
66
     /opt/jenkins/data/jobs/${JOB_NAME}/workspace
67
68
69
     cat > Dockerfile <<EOF
70
     FROM ubuntu:16.04
     RUN echo "while true; do date; echo $WORKSPACE; sleep 1; done" >
71
     /usr/bin/run.sh
     CMD ["/bin/bash", "/usr/bin/run.sh"]
72
     EOF
73
     docker build -t loop_test_$0S .
74
    docker run -d --name loop_test.$0S loop_test_$0S
75
     docker logs loop_test.$0S
76
77
    docker stop loop_test.$0S
78
     docker rm loop_test.$0S
79
     docker rmi loop_test_$0S
80
```

TProv

```
代码块

1 ruby File.exists => File.exist

musilia

musilia
```

```
代码块
    openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048;
1
 2
    openssl req -new -key private.key -out request.csr -subj \
 3
    "/C=CN/ST=Beijing/L=Beijing/O=WebRay/OU=IT/CN=cacheproxy.com/emailAddress=ridwa
 4
    rd@qq.com"
5
    openssl x509 -req -days 36135 -in request.csr -signkey private.key -out
    certificate.crt;
 7
    openssl pkey -in private.key -text -noout;
8
9
    openssl req -in request.csr -text -noout;
    openssl x509 -in certificate.crt -text -noout;
10
```

升级系统

```
代码块

1 上传包到 172.21.15.38

2 /var/log/webraydb/Dump/NJ/WebRay/1cf935508eb43c32a229826cf8c1be59

3 
4 在要升级的系统上以 admin 执行

5 # patchup/patchall

6 patchup -u 1cf935508eb43c32a229826cf8c1be59 -p E9DCCB48 \
7 -i ftp://172.21.15.38:10021/fap-cacheproxy-1.0.0-T6-20250211205624-openEuler-x86_64.img
```

科瑞托斯

代码块

1 kratos proto client api/helloworld/v1/demo.proto -- --gohttp_opt=omitempty=false;

Rbenv

```
代码块

1
2 # https://github.com/Tensho/docker-rbenv/blob/master/Dockerfile

3
4 FROM ubuntu:20.04

5 RUN apt-get -yqq update

6 RUN DEBIAN_FRONTEND=noninteractive TZ=Asia/Shanghai apt-get -y install tzdata
```

```
7
     RUN apt-get -yqq install build-essential nodejs
     RUN apt-get -yqq install git wget libffi-dev zlib1g-dev libyaml-dev
 8
     RUN rm /bin/sh && ln -s /bin/bash /bin/sh
 9
     #RUN wget -q -0 ~/.rbenv/cache/ruby-3.4.1.tar.gz -c https://ftp.ruby-
10
     lang.org/pub/ruby/3.4/ruby-3.4.1.tar.gz;
     RUN git clone --depth=1 https://github.com/rbenv.git /root/.rbenv
11
     RUN git clone --depth=1 https://github.com/rbenv/ruby-build.git
12
     /root/.rbenv/plugins/ruby-build
13
     RUN /root/.rbenv/plugins/ruby-build/install.sh
     ENV PATH /root/.rbenv/bin:/root/.rbenv/shims:$PATH
14
     RUN echo 'eval "$(rbenv init -)" >> /root/.bashrc
15
     RUN mkdir -p /root/.rbenv/cache
16
     COPY ruby-3.4.1.tar.gz /root/.rbenv/cache/
17
     RUN rbenv install 3.4.1
18
     RUN rbenv global 3.4.1
19
20
     RUN gem sources --remove https://rubygems.org/ --add
     https://mirrors.tuna.tsinghua.edu.cn/rubygems/;
     RUN gem update --system
21
     RUN gem install ffi -v 1.17.1
22
23
     #RUN gem install -s https://gems.ruby-china.com/ jekyll -v 2.5.3
24
     RUN gem install jekyll -v 2.5.3
    VOLUME /data
25
    VOLUME /var/www/html
26
    WORKDIR /data
27
28
     ENTRYPOINT ["jekyll", "build", "--destination=/var/www/html"]
29
30
31
     # gem install faraday-retry
```

Uwsig

```
代码块
    https://github.com/unbit/uwsgi/issues/2117;
2
3
    ARG uwsgi_dir=/var/lib/gems/2.5.0/gems/uwsgi-2.0.20
    RUN mkdir -p $uwsgi_dir/ext/uwsgi
4
    RUN curl -o $uwsgi_dir/ext/uwsgi/install.sh
 5
    https://raw.kkgithub.com/unbit/uwsgi/refs/heads/master/install.sh
    RUN sed -i 's/curl /curl -k /' $uwsgi_dir/ext/uwsgi/install.sh
6
    RUN apt-get -yqq install python3
    RUN cd $uwsgi_dir/ext/uwsgi; bash install.sh ruby2
    $uwsgi_dir/ext/uwsgi/uwsgi.ruby
    RUN mkdir -p $uwsgi_dir/bin; cp
9
    $uwsgi_dir/ext/uwsgi/uwsgi_latest_from_installer/bin/uwsgi $uwsgi_dir/bin/;
10
    ENV PATH=$uwsgi_dir/bin:$PATH
```

```
11 RUN uwsgi --version;
```

《Docker实战》

```
代码块
    https://wangwei1237.github.io/Kubernetes-in-Action-Second-Edition/
 2
3
    ip link set docker0 up;
4
    DOCKER_HOST;
5
    redis-server --protected-mode no;
    docker-compose logs/ps/stop/start/rm/kill;
6
7
8
    curl --unix-socket /var/run/docker.sock http://docker/info
9
    docker -H tcp://127.0.0.1:2375 images;
10
```

交叉编译

```
代码块

1 https://github.com/kekeqy/windows-hosted-x86_64-linux-musl-gcc-cross-compiler

2 https://www.cnblogs.com/nvim/p/18631356

3 https://juejin.cn/post/7168747781388140580

4

5
```