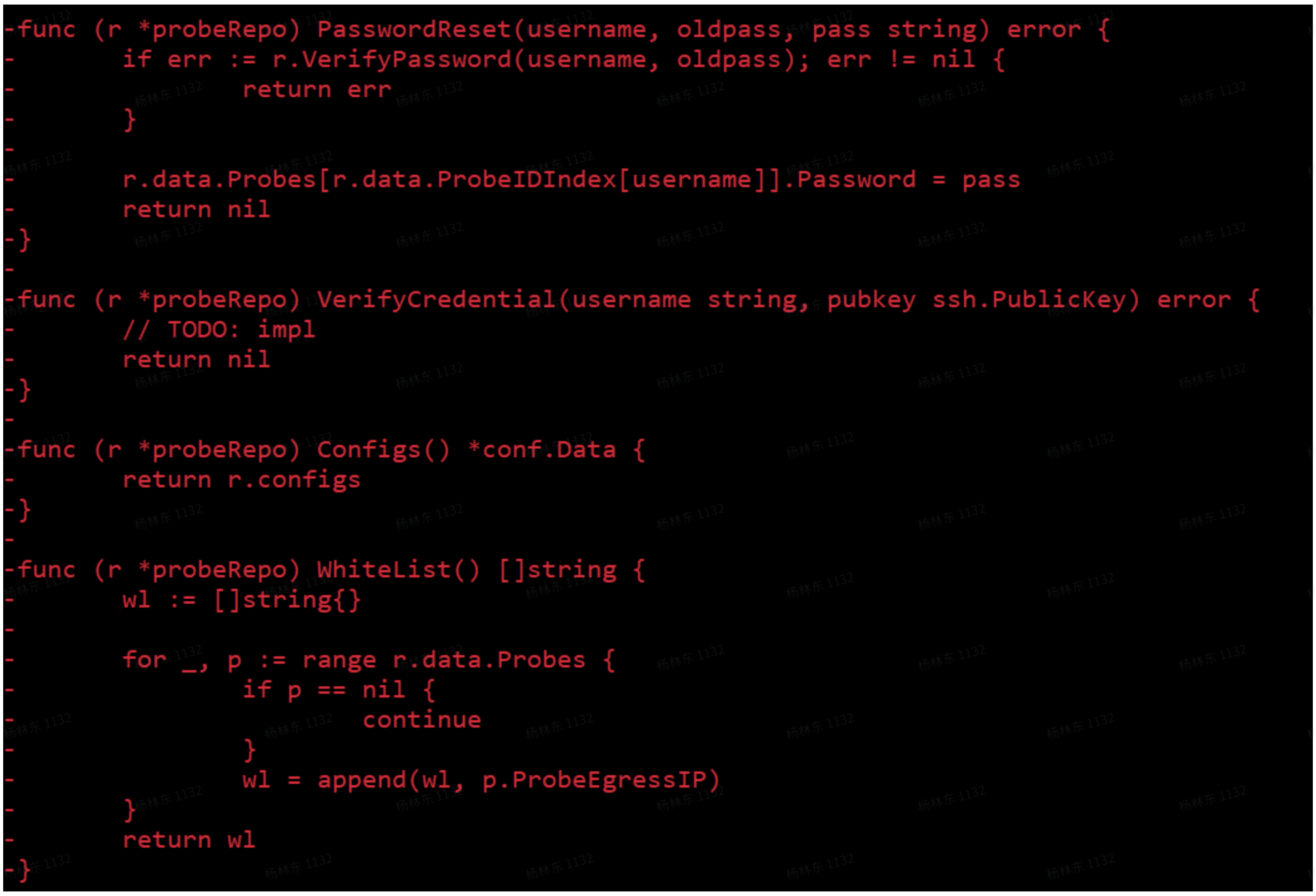
工作整理

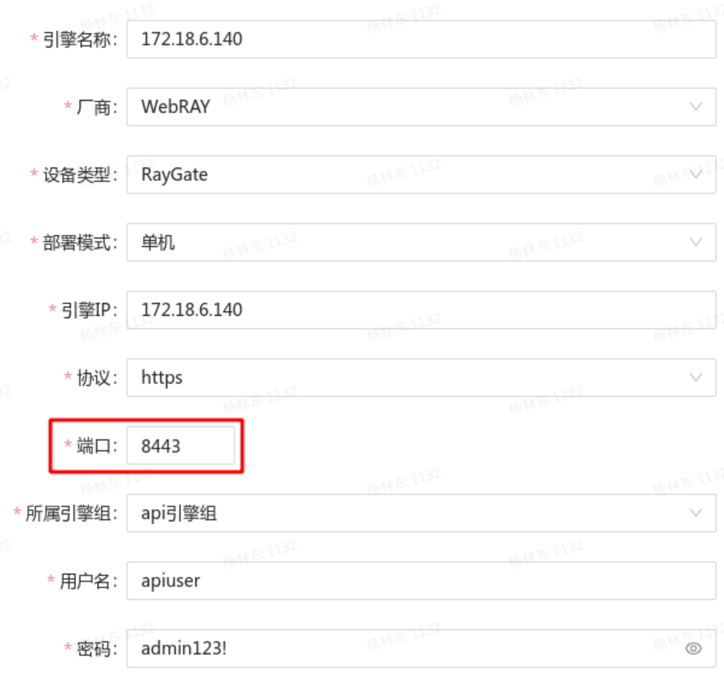
TODO

- 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



```
-func (r *probeRepo) PasswordReset(username, oldpass, pass string) error {
-	if err := r.VerifyPassword(username, oldpass); err != nil {
-		return err
-	}
-
-	r.data.Probes[r.data.ProbeIDIndex[username]].Password = pass
-	return nil
-}
-
-func (r *probeRepo) VerifyCredential(username string, pubkey ssh.PublicKey) error {
-	// TODO: impl
-	return nil
-}
-
-func (r *probeRepo) Configs() *conf.Data {
-	return r.configs
-}
-
-func (r *probeRepo) WhiteList() []string {
-	wl := []string{}
-
-	for _, p := range r.data.Probes {
-		if p == nil {
-			continue
-		}
-		wl = append(wl, p.ProbeEgressIP)
-	}
-	return wl
-}
```

EngineMgr



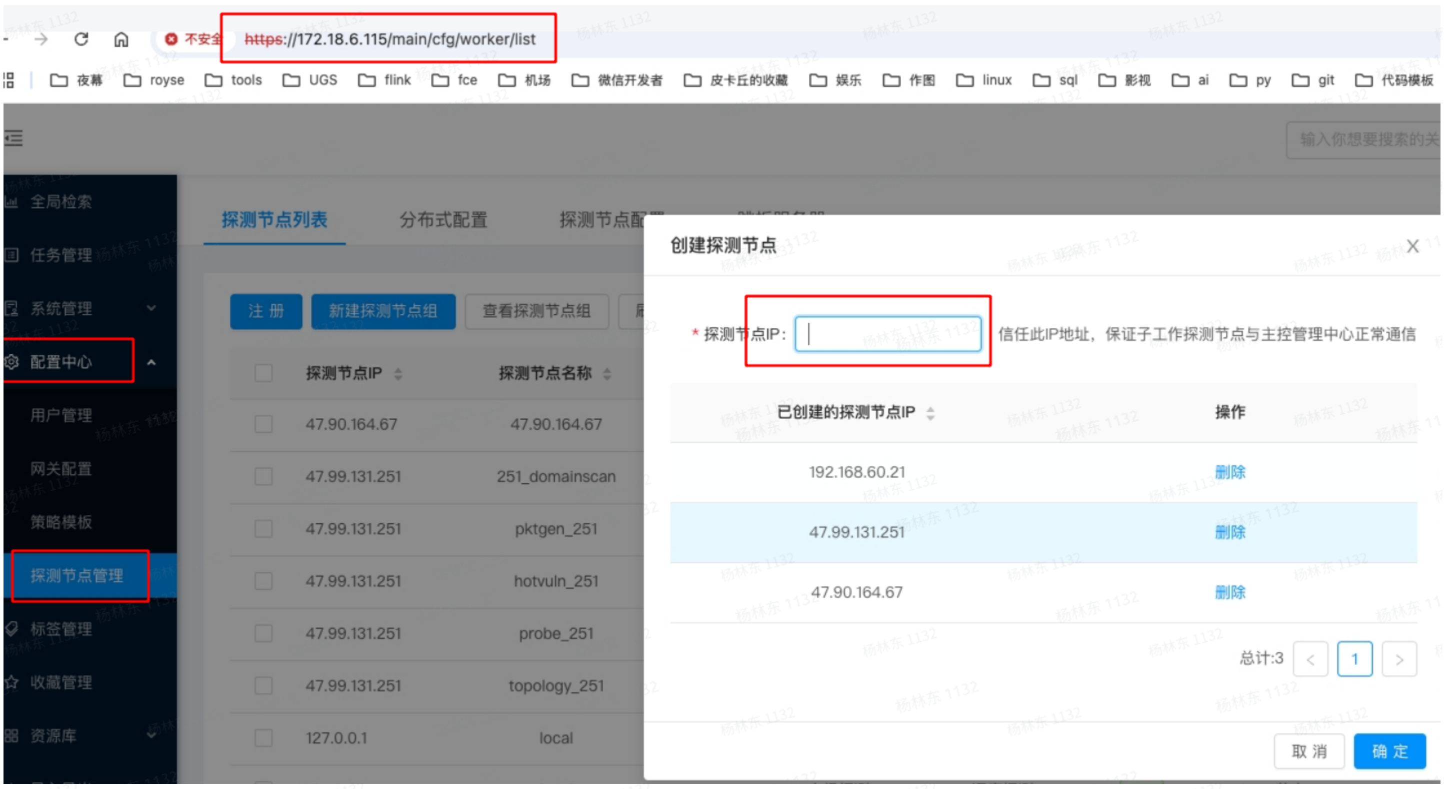
* 引擎名称:	172.18.6.140
* 厂商:	WebRAY
* 设备类型:	RayGate
* 部署模式:	单机
* 引擎IP:	172.18.6.140
* 协议:	https
* 端口:	8443
* 所属引擎组:	api引擎组
* 用户名:	apiuser
* 密码:	admin123!

代码块

```
默认 443 端口


/api/v1
    http://127.0.0.1:89

    /usr/lib/python3.7_venv_raygate/bin/uwsgi \
        --ini /rayos/app/webapp/raygate/uwsgi.ini
    后端，插入数据

 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 worker.py:119]
2025-04-14 11:03:24,468 INFO: [MainProcess] worker_manage 状态正常 ... [in worker.py:119]


~~^/(finger|async|logout|cfg|asset|system|report|logsystem|engine|label|base|active)/~~
~~    http://127.0.0.1:80~~
~~    ~~
~~    /usr/bin/python2 \~~
~~        /rayos/app/webapp/paladin/run.py~~


~~^/vm/(vmreport/|download/|css/|js/|img/|signin)~~
~~    http://127.0.0.1:16530~~


/rayos/app/worker_manage
    python3 /RayOS/app/worker_manage/worker.py
    上报引擎心跳
引擎
    分布式引擎：jobexe
    api引擎，能登录

    python3 /RayOS/app/worker_manage/worker.py


```

IP段加白，添加引擎

> superman/Test123!



https://172.18.6.115/main/cfg/worker/list

	探测节点IP	探测节点名称
☐	47.90.164.67	47.90.164.67
☐	47.99.131.251	251_domainscan
☐	47.99.131.251	pktgen_251
☐	47.99.131.251	hotvuln_251
☐	47.99.131.251	probe_251
☐	47.99.131.251	topology_251
☐	127.0.0.1	local

创建探测节点

* 探测节点IP： 信任此IP地址，保证子工作探测节点与主控管理中心正常通信

已创建的探测节点IP	操作
192.168.60.21	删除
47.99.131.251	删除
47.90.164.67	删除

总计:3 < 1 >

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

```
# go race condition check

# ulimit -c;
ulimit -c unlimited;

# %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

cp /etc/sysctl.conf{,.bak};
echo 'kernel.core_pattern=/var/log/rlog/cacheproxy/core-%e-%s-%u-%g-%p-%t' \
    >> /etc/sysctl.conf;
sysctl -p;
```

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

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

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

docker build \
    -t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
    -f ./dockerfiles/autopost.Dockerfile .
```

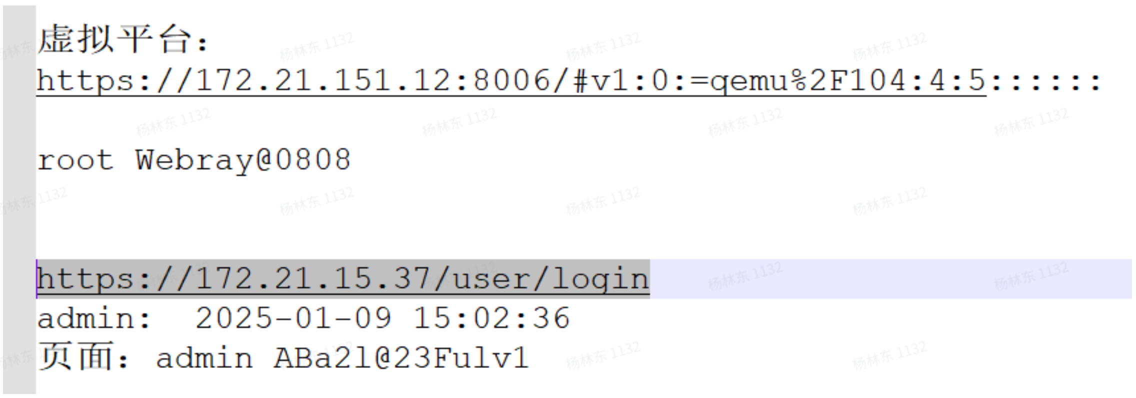
Git

```
https://kkgithub.com/
https://hub.yzuu.cf/

# 提交时转换为LF，检出时转换为CRLF
git config --global core.autocrlf true

# 提交和检出时都不转换
git config --global core.autocrlf false

```

```
# 提交时转换为LF，检出时不转换
git config --global core.autocrlf input
```

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

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

```

Env 配置

代码块

```
CC=x86_64-linux-musl-gcc
CGO_ENABLED=1
CGO_LDFLAGS=-O2 -g -static
GOROOT=C:/Program Files/go

```

代码块

```
func(*Server)

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

func Chain(m ...SFTPMiddleware) SFTPMiddleware {
    return func(next Handler) Handler {
        for i := len(m) - 1; i >= 0; i-- {
            next = m[i](next)
        }
        return next
    }
}

s.middleware = []{
    Recovery(),
    SSHAuth(ss.SSHConfigs()),
    WhiteList(ss.GetProbeEgressIP),
    Status(ss.GetProbeStatus),
    Logging(),
    ConnectionLimit(ss.Limit)
}
next = ServeSFTP(ctx context.Conext, conn net.Conn) error {
    s.probe.AppendSftpReqServer(p.Username, channel, connID)
    defer s.probe.RemoveSftpReqServer(p.Username, connID)
    defer server.Close()
    server.Serve()
}
go Recovery(
    SSHAuth(
        WhiteList(
            Status(
                go ps.Active()
                defer func(){
                    if cnt==0 {
                        go. ps.Logoff()
                    }
                }()
                Logging(
                    ConnectionLimit(next)
                )
            )
        )
    )
)

Logging: info user "ridward" logged out
go.ps.Logoff
```

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

```
http://bbs.itying.com/topic/676cf34b5b798701ddf78072

- 服务器管理
https://172.21.150.16/ui/#/host/vms/89
root/Esxi#0926
- 150.39 的账密 root/Superman@12345

- proxmox 账户
root/Webray@0808

- 删除
点中-more-remove

探针: https://172.21.15.37/user/login
admin/ABa2l@23Fulv1

系统管理/集中配置管理


- img 下载
- 外部: http://172.18.9.4:8001/products/fap-cacheproxy/
- 内部: http://172.21.1.25:8000/products/fap-cacheproxy/


上传 data(ibpd-uat-01), upload,

创建系统,
两个盘, 128G, 256G
2 sockets, 4 cores = 8 cres
16384 Memory MiB(16G)

- 登录账号: root/webray++
- 安装密码: WebRW2r$
输入 y 确认

等待好长时间....

y; xfs; rdata/cacheproxy; y; roboot;

等好久...
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

配置网络
vlan -A -v MngtVlan -f 172.21.150.107 -m 255.255.255.0;
vlan -S;
route -A -i 0.0.0.0 -m 0.0.0.0 -g 172.21.150.1;
route -S;
```

```

settime
help
upsshd start


[**autopost**]

docker build \
-t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
-f ./dockerfiles/autopost.Dockerfile .;

docker save registry.ibdp.webray.com.cn:51808/fap/autopost -o 'autopost.tar';

[autopost]
- sftp_policy_path: /policy-data
    pull 模式要忽略的远程 server 的策略存储目录
    ~~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)~~
- file_path: /home/
    定期清理老文件的目录, NewMonitorRepo 中的的 DataDir:
    pull 模式下文件同步到的目录
    ~~push 模式不用此配置~~
- policy_file_path: /home/policy-data
    ~~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

[cacheproxy]
RootAccount() 一致维护 root 用户
- data_dir: /home/cacheproxy/data
    探针文件夹创建的目录
    策略解压到的目录
    定期清理的目录 (NewMonitorRepo)
    root 用户的家目录
- app_dir: /home/cacheproxy
    创建一个 policy-data-backup 目录备份
    cert 所在目录
    data.db 所在目录
    处理请求的时候看了这个目录的信息??? HandleConn: IsDiskWork()
- policy_data_dir: /home/cacheproxy/data/policy-data
    push 过来的策略的存储路径
    定期检测并需要解压 .tgz

p.HomeDir 是不同 sftpServer 的目录

SystemProbeRecover
```

```
scp -O -l 1024 /d/... root@172.21.150.28:/cacheproxy/


# 172.21.150.27
export GOPROXY="http://172.21.3.11:3000"
账密: admin/Admin123, 端口是 8000


- cmd/cacheproxy/main.go
////go:embed web/*

- go.mod
toolchain go1.21.4

- configs/config.json
data_dir
app_dir
policy_data_dir


https://github.com/protocolbuffers/protobuf/releases
# exec: "protoc": executable file not found in %PATH%
https://github.com/protocolbuffers/protobuf/releases/download/v29.2/protoc-29.2-win64.zip;


VERSION=1.0;
# https://www.jianshu.com/p/10ae8a5a7704
# go get github.com/go-kratos/kratos/cmd/kratos/v2@latest;
go install github.com/go-kratos/kratos/cmd/kratos/v2@latest;

kratos proto client ./internal/conf/conf.proto;
kratos proto client ./api/cacheproxy/v1/cacheproxy.proto;



go build -ldflags "-X main.Version=$(VERSION)";
chmod a+x /usr/local/go/bin/kratos;
go mod tidy;
cd cmd/cacheproxy;
go build;
./cacheproxy -conf ../../configs/config.json;
dlv --listen=:2345 --headless=true --api-version=2 --accept-multiclient exec ./cacheproxy
```

```

缓存转发的 root_passwd 要配置为 root
# ../configs/autopost.yaml 要配置 sftp_address 和 3 个目录
./autopost -conf ../configs/autopost.yaml -mode push
上传文件到单导的 /cacheproxy/home/policy-data
单导日志类似:
caller=biz/autopost_push.go:203 ... \
    policy-data/20241221181530_policy.tgz.part successfully

缓存转发的日志类似:
    caller=biz/monitor.go .... \
    job: /home/cacheproxy/data/policy-data/20241221181530_policy.tgz

文件会被解压到 /home/cacheproxy/data 下
    - 0XSDJI902
    - 0XSDJI905
    - 0XSDJI908
    - E04C81F400
    - SXXAtest11
/home/cacheproxy/data/policy-data
/hoem/cacheproxy/policy-data(??? 是什么)
```

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
https://www.cnblogs.com/binHome/p/13020178.html

https://www.jianshu.com/p/a73cef45b65c

wire:
https://www.cnblogs.com/jiujuan/p/16136633.html

在 service/service.go
type AppRepo interface {
    ...
}

在 data/data.go
type appRepo struct {
    *mq
    *db
    *dp
}
```

```

func NewAppRepo() service.AppRepo {
    appRepo := &appRepo{}
    return appRepo
}

//
google.golang.org/protobuf/types/know/anypb

marshal := v1.Flow { Enable: true, Subset: "127.0.0.1" }
any, err := anypb.New(&marshal);
[type.googleapis.com/engmgr.v1.Flow]:{enable:true, subset:"127.0.0.1"}

grpcurl -plaintext -H "x-user-id: 1" 172.21.150.109:9000 list;
grpcurl -plaintext -H "x-user-id:1" -d '{}' 172.21.150.109 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

# 自定义 json
https://juejin.cn/post/6844904184651874312
```

```
# select
https://www.cnblogs.com/qcy-blog/p/18520091
# docker copy
https://developer.baidu.com/article/details/3214252
# break
https://blog.csdn.net/u011461385/article/details/106017483
# chan
https://blog.csdn.net/chenchongg/article/details/86589395
# nil, close chan
https://dave.cheney.net/2014/03/19/channel-axioms
# strace
// 只显示 renameat 的调用情况
strace -f -e trace=renameat go run ./script/download-go go1.22.3

# 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
```

```
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)
}


```

```
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.
Packages in the standard library do not have a module path prefix

go install example/user/hello;


The install directory is controlled by the GOPATH and GOBIN environment variables.
If GOBIN is set, binaries are installed to that directory.
If GOPATH is set, binaries are installed to the bin subdirectory of
the first directory in the GOPATH list.
Otherwise, binaries are installed to the bin subdirectory of
the default GOPATH ($HOME/go or %USERPROFILE%\go)

go env -u GOBIN;

$(dirname $(go list -f '{{.Target}}' .));

go help importpath;

GOCACHE: $HOME/.cache/go-build;

import "example/user/hello/morestrings";

go mod tidy;

$HOME/.mygo/pkg/mod/

go clean -modcache;
```

```
go list -f '{{.Target}}';
go env -w GOBIN=/path/to/your/bin;

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;
uintptr;

%T, %v
Zero values;
different type requires an explicit conversion;
Gos switch cases need not be constants, and the values involved need not be
integers;
Unlike C, Go has no pointer arithmetic.;

type Vertex struct {
        X int
        Y int
}

(*p).X -> p.X;

var (
        v1 = Vertex{1, 2}  // has type Vertex
        v2 = Vertex{X: 1}  // Y:0 is implicit
        v3 = Vertex{}      // X:0 and Y:0
        p  = &Vertex{1, 2} // has type *Vertex
)

Slices are like references to arrays;
cap();
a := make([]int, 5)  // len(a)=5;
b := make([]int, 0, 5) // len(b)=0, cap(b)=5;
strings.Join;
The range form of the for loop iterates over a slice or map.
for i := range pow;
delete(m, key)
elem, ok = m[key]

The make function returns a map of the given type, initialized and ready for
use;


func fibonacci() func() int {
        i, j := 0, 1
        return func() int {
                i, j = j, i+j
                return i
        }
}

Methods:
The receiver appears in its own argument list
between the func keyword and the method name.
func (v Vertex) Abs() float64 {
        return math.Sqrt(v.X*v.X + v.Y*v.Y)
}


You can only declare a method with a receiver
whose type is defined in the same package as the method.
Pointer receivers;

Go interprets the statement v.Scale(5) as (&v).Scale(5);
p.Abs() is interpreted as (*p).Abs();

type Abser interface {
        Abs() float64
}

func (t *T) M() {
        if t == nil {
                fmt.Println("<nil>")
                return
        }
        fmt.Println(t.S)
}

# Type assertions
t := i.(T);
t, ok := i.(T);

switch v := i.(type);

type Stringer interface {
    String() string
}

func (p Person) String() string {
        return fmt.Sprintf("%v (%v years)", p.Name, p.Age)
}

type error interface {
    Error() string
```

```
}

%q;

func Index[T comparable](s []T, x T) int

type List[T any] struct {
        next *List[T]
        val  T
}

The evaluation of f, x, y, and z happens in the current goroutine and
the execution of f happens in the new goroutine.


ch <- v    // Send v to channel ch.
v := <-ch  // Receive from ch, and
           // assign value to v.


Like maps and slices, channels must be created before use:
ch := make(chan int);

ch := make(chan int, 100);

v, ok := <-ch;

Only the sender should close a channel, never the receiver.
Sending on a closed channel will cause a panic.


select {
case c <- x:
        x, y = y, x+y
case <-quit:
        fmt.Println("quit")
        return
}

mu sync.Mutex;
c.mu.Lock();
c.mu.Unlock();
```

ssh

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

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;
psql postgresql://user:pass@host:port/dbname
PGPASSWORD='admin1234' psql -U postgres -p 5432 fusion;

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



PGPASSWORD='Mjolnir' pg_dump -d test -U postgres -h 127.0.0.1 --no-comments \
--column-inserts \
--schema-only \
-t 'group' \
-t 'device_group' \
-t 'device' \
-t 'engine' \
> enginemgr.sql;


-s --schema-only，加上只导表结构
```

iptables

```
/rayos/cfg/iptables_port.conf     6379 5432
/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

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

Pycharm

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

Iptables

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

Es

```
network.host: 0.0.0.0
discovery.seed_hosts: ["127.0.0.1"]

network.bind_host
network.public_host
network.host
discovery.seed_hosts:  集群主机列表
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

curl -K https://elastic:elastic@172.21.15.234/_settings?pretty

```

alembic 使用

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

```

```
apt-get download git;
ssh-copy-id -i ~/.ssh/id_*.pub root@172.21.15.234;
useradd -m -s /bin/bash lindyang;
passwd lindyang;

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;


cat /etc/os-release
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

```
ARG DEBIAN_FRONTEND=noninteractive
ENV TZ=Asia/Shanghai
RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y install tzdata;

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
{
    "registry-mirrors": [
        "https://dockerproxy.cn",
        "https://docker.rainbond.cc",
        "https://docker.udayun.com",
        "https://docker.211678.top"
    ]
}
EOF

# 重载，重启Docker服务
sudo systemctl daemon-reload
sudo systemctl restart docker
```

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;
```

```
minikube start \
    --vm-driver none \
    --driver=docker \
    --image-mirror-country='cn' \
    --image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers;

minikube status;
minikube dashboard;
minikube ssh;
eval $(minikube docker-env);

kubectl cluster-info;
kubectl get nodes;
kubectl describe node minikube

minikube image load kubia;
minikube ssh;
docker tag kubia luksa/kubia:1.0;
exit;
kubectl create deployment kubia --image=luksa/kubia:1.0;
kubectl get deployments;

kubectl get pods;
kubectl describe pod;

kubectl expose deployment kubia --type=LoadBalancer --port 8080;
kubectl get svc;

k get nods/pods/deployments/pods;

kubectl api-resources;

```

```
# If you use Minikube to create the cluster,
# no load balancer is created

minikube service kubia --url;
minikube ip;
kubectl scale deployment kubia --replicas=3;

kubectl get deploy;

kubectl get pods -o wide;

kubectl get nodes minikube -o yaml;

kubectl proxy;
http://127.0.0.1:8001/api/v1/nodes/minikube;

kubectl explain nodes;
kubectl explain node.spec;  # --api-version
kubectl explain pods --recursive;

kubectl get ev -o wide;
kubectl get ev --field-selector type=Warning;

kubectl explain events;

kubectl run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml >
mypod.yaml;

kubectl apply -f kubia.yaml;

k run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml;

k describe pod kubia; minikube ssh; curl 172.17.0.X:8080;

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

kubectl get pod kubia -o wide;

kubectl run \
--image=ubuntu:latest -it \
--restart=Never \
--rm client-pod curl 172.17.0.7:8080;

kubectl port-forward kubia ~~8080:~~8080;
kubectl port-forward --help;
kubectl logs --timestamps~~=true~~ -f kubia;
```

```
    --since=2m
    --since-time=2020-02-01T09:50:00Z
    --tail=10
    -p (--previous)

kubectl exec kubia -- ps aux;

kubectl attach;

kubectl port-forward kubia-ssl 8080 8443 9901;
curl https://example.com:8443 --resolve example.com:8443:127.0.0.1 -k;

kubectl delete po kubia-ssl --grace-period 10;
    spec.terminationGracePeriodSeconds
    metadata.deletionGracePeriodSeconds


kubectl logs kubia-ssl -c kubia;
kubectl logs kubia-ssl --all-containers;

kubectl exec -it kubia-ssl -c envoy -- bash;

k delete pod kubia --wait=false;

kubectl delete -f kubia-ssl.yaml;
kubectl delete -f kubia.yaml,kubia-ssl.yaml;
kubectl apply -f Chapter05/;
kubectl delete -f Chapter05/;
    --recursive

kubectl delete po --all;

kubectl delete all --all;

kubectl delete events,all --all;

kubectl exec kubia-liveness -c envoy -- tail -f /var/log/envoy.admin.log;

kubectl logs kubia-liveness -c envoy -p;
```

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

```

kubectl get nodes -o jsonpath='{.items[*].status.addresses}';
kubectl get nodes -o \
jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}';

# http://kt1bernetes.io/docs/user-guide/jsonpath

minikube service kubia-nodeport;
# minikube sevrvice <service-name> [-n <namespace>]

externalTrafficPolicy: Local;

minikube addons list;
minikube addons enable ingress;
apiVersion: networking.k8s.io/v1;
```

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubia
spec:
  rules:
  - host: kubia.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kubia-nodeport
            port:
              number: 80
```

```
docker build -t kubia .;
删除 DaemonSet 也会删除 pod;

# job 资源
kubectl get jobs;

```

```
# completions: 5
# parallelism: 2

# 有问题
kubectl scale job multi-completion-batch-job --replicas 3;

activeDeadlineSeconds
spec.backoffLimit=6 (default)

CronJob
分钟 小时 每月中的第几天 月 星期几(0是星期天)
startingDeadlineSeconds


创建 service 后, 需要
minikube ssh; # 登录
curl `kubectl get svc | awk '{print $3}'`  # CLUSTER-IP

kubectl exec kubia-manual -- curl -s http://10.99.94.217;  # 没有 -s, -- 是非必须得

sessionAffinity: ClientIP;  # 不支持http cookie
kubectl exec kubia-manual -- env;
dnsPolicy;

backend-database.default.svc.cluster.local;

kubectl exec -it kubia-manual -- bash;
curl -s http://kubia.default.svc.cluster.local;
curl http://kubia;

cat /etc/resolv.conf;
ping kubia;  # 无法ping通, 因为是虚拟IP, 只有与port结合才有意义
kubectl describe svc kubia | grep -i endpoint;
kubectl get endpoints kubia;
curl http://external-service/echo;

# kubia ExternalName
```

```
kubectl logs mypod --previous;
kubectl edit rc kubia;

export KUBE_EDITOR="/usr/bin/vim";
export EDITOR="/usr/bin/vim";
kubectl scale rc kubia --replicas=4;
kubectl delete rc kubia --cascade=false;

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.

kubectl api-versions;
replicaset.apps/kubia created;
kubectl get rs;
kubectl delete rs kubia;

# 使用 DaemonSet 在每个节点上运行一个 pod
# 使用 DaemonSet 只在特定的节点运行 pod
# 节点可以被设置为不可调度, 但DaemonSet可以绕过它
```

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: kubia
        imagePullPolicy: Never
```

- matchExpressions

```
selector:
    matchExpressions:
    - key: app
      operator: In
```

```
        values:
        - kubia
```

- operator
  - In
  - NotIn
  - Exists
  - DoesNotExist

```
apiVersion: v1
kind: Pod
metadata:
  name: kubia-liveness
spec:
  containers:
  - image: kubia-unhealthy
    imagePullPolicy: Never
    name: kubia
    livenessProbe:
      httpGet:
        path: /
        port: 8080
      initialDelaySeconds: 15
```

```
- pod 共享 network, UTS, IPC, PID(默认未开启)
- 文件系统隔离, 但是可以通过 Volume 共享文件目录
- localhost 可以与同一 pod 中的其它容器通信
kubectl get po kubia-82d24 -o yaml;
kubectl explain pods;
kubectl explain pod.spec;
kubectl create -f lindyang/kubia-manual.yaml;

kubectl describe pod kubia-manual | grep 172;
minikube ssh;
curl 172.17.0.7:8080;

kubectl logs kubia-manual -c kubia;
```

```

# 不通过 service 与 pod 通讯
kubectl port-forward kubia-manual 8888:8080;
curl http://127.0.0.1:8888;

kubectl get pods --show-labels;
kubectl get pod -L creation_method,env;

kubectl label po kubia-manual creation_method=manual --overwrite
# 删除标签, -
kubectl label po kubia-manual createion_method-

kubectl get po -l creation_method=manual;

kubectl get po -l env;

kubectl get po -l '!env';
env!=prod
'env in (prod,dev)'
'env notin (prod,dev)'

kubectl label node minikube gpu=false;

kubectl annotate pod kubia-manual mycompany.com/someannotation="foo bar";

kubectl get po --namespace kube-system;

kubectl get ns;

kubectl delete namespace custom-namespace;
kubectl create namespace custom-namespace;
kubectl create -f lindyang/kubia-manual.yaml -n custom-namespace;

命名空间不允许包含点号.

alias kcd='kubectl config set-context $(kubectl config current-context) --namespace'

kubectl delete pod po1 po2;

kubectl delete po -l creation_method=manual;

kubectl delete ns custom-namespace;

kubectl delete po --all;

kubectl delete all --all;
```

- Namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubia-gpu
spec:
  nodeSelector:
    gpu: "true"
  containers:
  - name: kubia
    image: kubia
    imagePullPolicy: Never
```

```bash
minikube status;
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);

kubectl run --image=kubia --port=8080 kubia;
kubectl get pod kubia -o yaml -n default;
kubectl describe pod kubia;

https://www.cnblogs.com/xiao2/p/16047455.html;
minikube image load kubia;
minikube image build -t <IMAGE_NAME> .;

```

```
kubectl get nodes;

kubectl apply -f lindyang/deployment.yaml;
kubectl delete -f lindyang/deployment.yaml;
kubectl port-forward kubia-868cd55b98-qzz79 8080:8080;
kubectl expose pod kubia-868cd55b98-qzz79 --type=LoadBalancer --name kubia-http;
# minikube 不支持 LoadBalancer;
minikube service kubia-http;
kubectl get rc;
kubectl get services; # src

kubectl scale --replicas=3 -f lindyang/deployment.yaml;
```

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: kubia
spec:
  replicas: 3
---    selector:
---    app: kubia
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
        - name: kubia
          image: kubia
          imagePullPolicy: Never
          ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  type: LoadBalancer
  selector:
    app: kubia
  ports:
  - port: 8080
    targetPort: 8080
```

```yaml
apiVersion: apps/v1
kind: Deployment
#kind: ReplicaSet
metadata:
  name: kubia
  labels:
    name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      name: kubia
  template:
    metadata:
      labels:
        name: kubia
    spec:
      containers:
        - name: kubia
          image: kubia
          imagePullPolicy: Never
          ports:
            - 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;


minikube start --force --driver=docker \
--image-mirror-country='cn' \
--registry-mirror=https://docker.mirrors.ustc.edu.cn \
--image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers


minikube start \
--kubernetes-version=v1.23.8 \
--image-mirror-country='cn' \
--image-repository='registry.cn-hangzhou.aliyuncs.com/google_containers';


minikube start --image-mirror-country=cn;

minikube kubectl -- create clusterrolebinding system:anonymous \
--clusterrole=cluster-admin  \
--user=system:anonymous;

kubectl proxy \
--port=8888 \
--address='192.168.1.20' \
--accept-hosts='^.*'  >/dev/null 2>&1 &;


minikube kubectl -- get pods -A;
minikube kubectl -- logs kube-proxy-2z5gg -n kube-system;
sudo sysctl -w net.netfilter.nf_conntrack_max=524288;

minikube kubectl -- describe pods .. -n kubernetes-dashboard
# minikube start --disk-size="10g" \
#--image-mirror-country="cn"  \
#--image-repository="registry.cn-hangzhou.aliyuncs.com/google_containers";
```

Dashboard 失败

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

```
minikube ssh;

docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.8;

docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.8 \
docker.io/kubernetesui/metrics-scraper:v1.0.8;

docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0;

docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0 \
kubernetesui/dashboard:v2.7.0;


registry.hub.docker.com/kubernetesui/dashboard:v2.1.0;
registry.hub.docker.com/kubernetesui/metrics-scraper:v1.0.4;
```

```
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;

~/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
查看单个: SHOW HOSTS GRAPH、SHOW HOSTS STORAGE、SHOW HOSTS META;
ADD HOSTS 127.0.0.1:9779
心跳: heartbeat_interval_secs
GO语句采用的是walk类型路径;
MATCH、FIND PATH和GET SUBGRAPH语句采用的是trail类型路径;

$$     表示边的终点;
$^     表示边的起点;
$-     表示管道符前面的查询输出的结果集;

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;
// 重建索引确保能对已存在数据生效。
nebula> REBUILD TAG INDEX player_index_1;

create edge index follow_index_1 on follow(degree);
rebuild edge index follow_index_1;

```

Windows git bash 设置字体大小

```
echo > ~/.minttyrc <<EOF
Font=Consolas
FontHeight=14
EOF
```

sougou_input

```
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。
vim ~/.config/sogoupinyin/conf/env.ini

在该配置文件中添加一行
DefaultSymbol=0

然后重新启动输入法即可。
```

postgres

```
sudo -u postgres psql;
ALTER USER postgres WITH PASSWORD 'Mjolnir';
CREATE DATABASE mydatabase TEMPLATE template0 ENCODING 'UTF8';

select datid, datname,
    pid,
    usesysid,
    usename,
    client_addr,
    client_port,
    query_start,
    state
    --,query
FROM pg_stat_activity;
```

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

```

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

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

Jenkins

```
FROM ubuntu:16.04
MAINTAINER james@example.com
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 -
COPY gpg /tmp/gpg
RUN cat /tmp/gpg | apt-key add -; rm /tmp/gpg
RUN apt-get clean && rm -rf /var/lib/apt/lists/*
#RUN echo deb https://apt.dockerproject.org/repo ubuntu-trusty main > /etc/apt/sources.list.d/docker.list
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 iptables ca-certificates git-core
RUN apt-get install -qqy --allow-unauthenticated docker-ce
#RUN apt-get install -qqy --allow-unauthenticated openjdk-9-jdk
RUN add-apt-repository -y ppa:openjdk-r/ppa
RUN apt-get update -qq && apt-get install -qqy openjdk-17-jdk

ENV JENKINS_HOME /opt/jenkins/data
#ENV JENKINS_MIRROR http://mirrors.jenkins-ci.org

RUN mkdir -p $JENKINS_HOME/plugins
#RUN curl -sf -o /opt/jenkins/jenkins.war -L $JENKINS_MIRROR/war-stable/latest/jenkins.war
COPY jenkins.war /opt/jenkins/jenkins.war

#RUN for plugin in chucknorris greenballs scm-api git-client git ws-cleanup ;\
#    do curl -sf -o $JENKINS_HOME/plugins/${plugin}.hpi \
#       -L $JENKINS_MIRROR/plugins/${plugin}/latest/${plugin}.hpi ; done
COPY ./*.hpi $JENKINS_HOME/plugins/

ADD ./dockerjenkins.sh /usr/local/bin/dockerjenkins.sh
RUN sed -i 's/^docker daemon/dockerd/' /usr/local/bin/dockerjenkins.sh
RUN chmod +x /usr/local/bin/dockerjenkins.sh

#VOLUME /var/lib/docker

EXPOSE 8080

```

```
# docker daemon & => dockerd
ENTRYPOINT [ "/usr/local/bin/dockerjenkins.sh" ]


$ sudo mkdir -p /var/jenkins_home
$ cd /var/jenkins_home
$ sudo chown -R 1000 /var/jenkins_home

docker run --privileged -d \
-p 8080:8080 \
-v /var/jenkins_home:/opt/jenkins/data \
-v /var/run/docker.sock:/var/run/docker.sock \
--name jenkins \
lindyang/jenkins

sudo sed -i 's/https:/http:/'
/var/lib/docker/volumes/data/_data/hudson.model.UpdateCenter.xml;
# http://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json

# https://baijiahao.baidu.com/s?id=1742735186119625775&wfr=spider&for=pc
# https://www.cnblogs.com/yshc/p/10621224.html

https://updates.jenkins.io/download/plugins/skip-certificate-check/1.1/skip-
certificate-check.hpi

- Dashboard/Manage Jenkins/插件管理/advanced
- https://www.cnblogs.com/fengwenqian/p/13534786.html

http://updates.jenkins.io/update-center.json;
https://gitee.com/Errorcode500/docker-jenkins-sample.git

/opt/jenkins/data/jobs/${JOB_NAME}/workspace

cat > Dockerfile <<EOF
FROM ubuntu:16.04
RUN echo "while true; do date; echo $WORKSPACE; sleep 1; done" >
/usr/bin/run.sh
CMD ["/bin/bash", "/usr/bin/run.sh"]
EOF
docker build -t loop_test_$OS .
docker run -d --name loop_test.$OS loop_test_$OS
docker logs loop_test.$OS
docker stop loop_test.$OS
docker rm loop_test.$OS
docker rmi loop_test_$OS

```

TProv

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

生成证书

```
openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048;

openssl req -new -key private.key -out request.csr -subj \
"/C=CN/ST=Beijing/L=Beijing/O=WebRay/OU=IT/CN=cacheproxy.com/emailAddress=ridward@qq.com"

openssl x509 -req -days 36135 -in request.csr -signkey private.key -out certificate.crt;

openssl pkey -in private.key -text -noout;
openssl req -in request.csr -text -noout;
openssl x509 -in certificate.crt -text -noout;
```

升级系统

```
上传包到 172.21.15.38
/var/log/webraydb/Dump/NJ/WebRay/1cf935508eb43c32a229826cf8c1be59

在要升级的系统上以 admin 执行
# patchup/patchall
patchup -u 1cf935508eb43c32a229826cf8c1be59 -p E9DCCB48 \
-i ftp://172.21.15.38:10021/fap-cacheproxy-1.0.0-T6-20250211205624-openEuler-x86_64.img
```

Kratos

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

Rbenv

```

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

FROM ubuntu:20.04
RUN apt-get -yqq update
RUN DEBIAN_FRONTEND=noninteractive TZ=Asia/Shanghai apt-get -y install tzdata
RUN apt-get -yqq install build-essential nodejs
RUN apt-get -yqq install git wget libffi-dev zlib1g-dev libyaml-dev
RUN rm /bin/sh && ln -s /bin/bash /bin/sh
#RUN wget -q -O ~/.rbenv/cache/ruby-3.4.1.tar.gz -c https://ftp.ruby-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
RUN git clone --depth=1 https://github.com/rbenv/ruby-build.git /root/.rbenv/plugins/ruby-build
RUN /root/.rbenv/plugins/ruby-build/install.sh
ENV PATH /root/.rbenv/bin:/root/.rbenv/shims:$PATH
RUN echo 'eval "$(rbenv init -)"' >> /root/.bashrc
RUN mkdir -p /root/.rbenv/cache
COPY ruby-3.4.1.tar.gz /root/.rbenv/cache/
RUN rbenv install 3.4.1
RUN rbenv global 3.4.1
RUN gem sources --remove https://rubygems.org/ --add https://mirrors.tuna.tsinghua.edu.cn/rubygems/;
RUN gem update --system
RUN gem install ffi -v 1.17.1
#RUN gem install -s https://gems.ruby-china.com/ jekyll -v 2.5.3
RUN gem install jekyll -v 2.5.3
VOLUME /data
VOLUME /var/www/html
WORKDIR /data

ENTRYPOINT ["jekyll", "build", "--destination=/var/www/html"]

# gem install faraday-retry
```

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/

ip link set docker0 up;
DOCKER_HOST;
redis-server --protected-mode no;
docker-compose logs/ps/stop/start/rm/kill;

curl --unix-socket /var/run/docker.sock http://docker/info

docker -H tcp://127.0.0.1:2375 images;
```

交叉编译

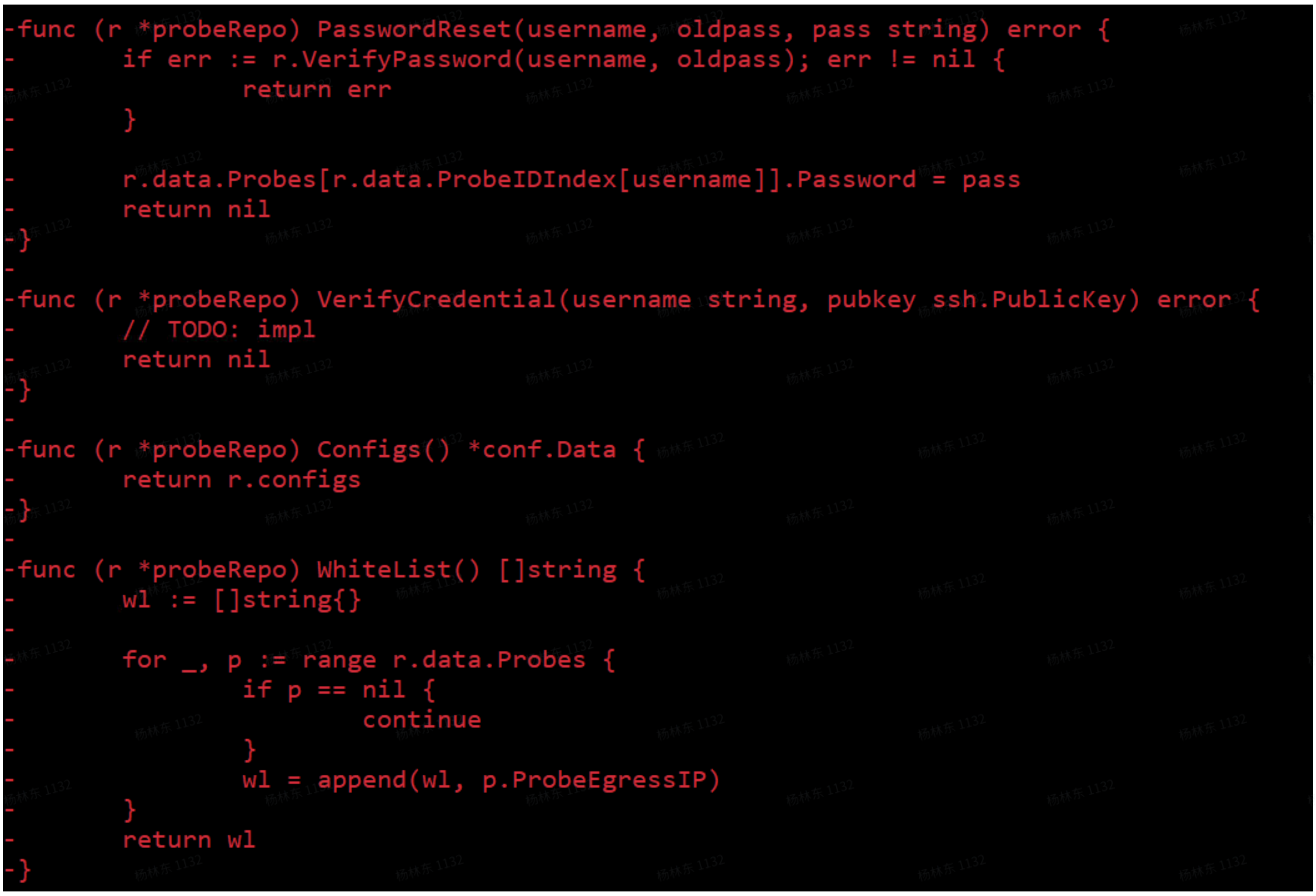
```
https://github.com/kekeqy/windows-hosted-x86_64-linux-musl-gcc-cross-compiler
https://www.cnblogs.com/nvim/p/18631356
https://juejin.cn/post/7168747781388140580


```

工作整理

待办事项

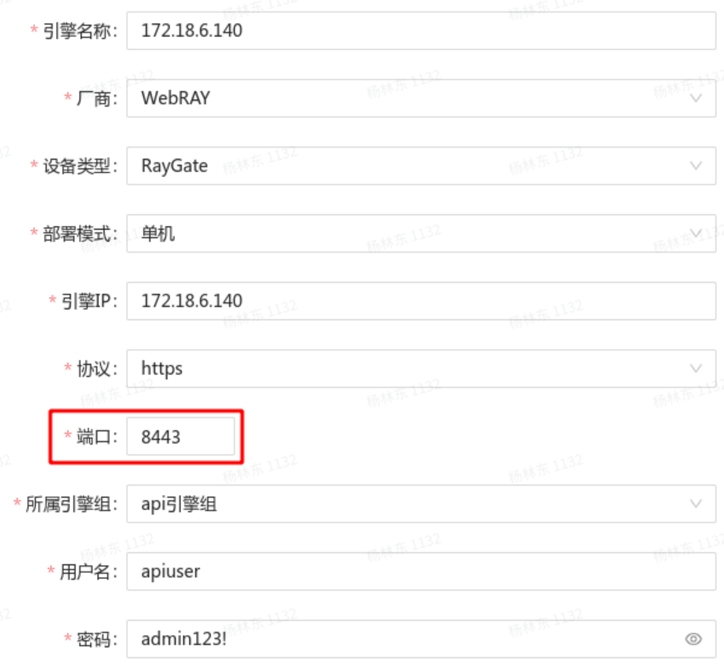
- https://blog.csdn.net/weixin_40242845/article/details/128400291
- Multiple ssh.Channel status issues
- https://blog.csdn.net/hui1429/article/details/80144863
- 默认的上传下载速度
- 批量修改自动发布根列表
- 用户信息

- 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



```
-func (r *probeRepo) PasswordReset(username, oldpass, pass string) error {
-       if err := r.VerifyPassword(username, oldpass); err != nil {
-               return err
-       }
-
-       r.data.Probes[r.data.ProbeIDIndex[username]].Password = pass
-       return nil
-}
-
-func (r *probeRepo) VerifyCredential(username string, pubkey ssh.PublicKey) error {
-       // TODO: impl
-       return nil
-}
-
-func (r *probeRepo) Configs() *conf.Data {
-       return r.configs
-}
-
-func (r *probeRepo) WhiteList() []string {
-       wl := []string{}
-
-       for _, p := range r.data.Probes {
-               if p == nil {
-                       continue
-               }
-               wl = append(wl, p.ProbeEgressIP)
-       }
-       return wl
-}
```

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!

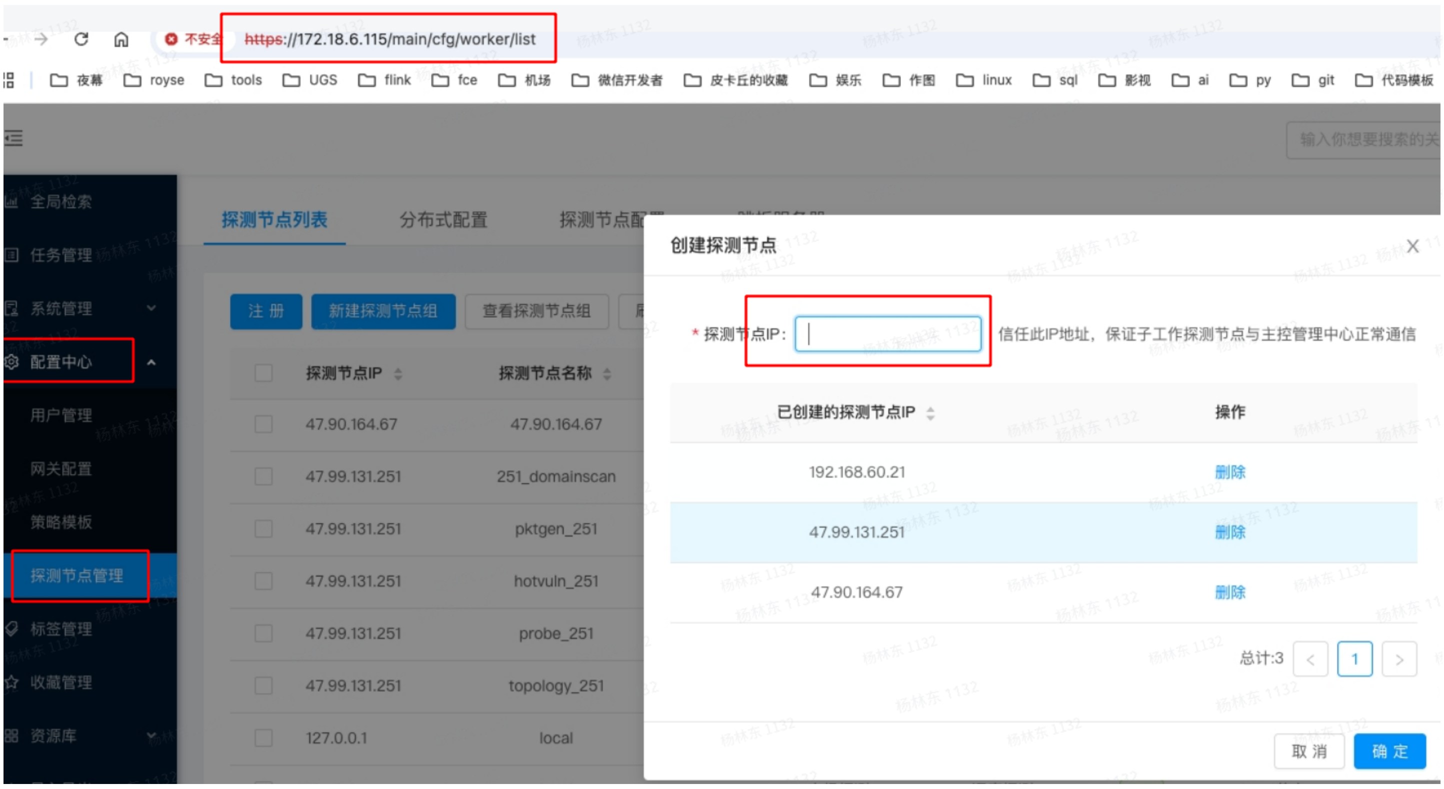
代码块

```
默认 443 端口


/api/v1
    http://127.0.0.1:89

    /usr/lib/python3.7_venv_raygate/bin/uwsgi \
        --ini /rayos/app/webapp/raygate/uwsgi.ini
    后端, 插入数据

 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 worker.py:119]
2025-04-14 11:03:24,468 INFO: [MainProcess] worker_manage 状态正常 ... [in worker.py:119]


~~^/(finger|async|logout|cfg|asset|system|report|logsystem|engine|label|base|active)/~~
~~    http://127.0.0.1:80~~
~~    ~~
~~    /usr/bin/python2 \~~
~~        /rayos/app/webapp/paladin/run.py~~


~~^/vm/(vmreport/|download/|css/|js/|img/|signin)~~
~~    http://127.0.0.1:16530~~
```

```


/rayos/app/worker_manage
    python3 /RayOS/app/worker_manage/worker.py
    上报引擎心跳
引擎
    分布式引擎: jobexe
    api引擎, 能登录

    python3 /RayOS/app/worker_manage/worker.py


```

IP段加白, 添加引擎

> 超人/测试123!



https://172.18.6.115/main/cfg/worker/list

探测节点列表 | 分布式配置 | 探测节点配置

注 册 | 新建探测节点组 | 查看探测节点组

探测节点IP	探测节点名称
47.90.164.67	47.90.164.67
47.99.131.251	251_domainscan
47.99.131.251	pktgen_251
47.99.131.251	hotvuln_251
47.99.131.251	probe_251
47.99.131.251	topology_251
127.0.0.1	local

创建探测节点

* 探测节点IP: 信任此IP地址，保证子工作探测节点与主控管理中心正常通信

已创建的探测节点IP	操作
192.168.60.21	删除
47.99.131.251	删除
47.90.164.67	删除

总计:3

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

代码块
```
# go race condition check

# ulimit -c;
ulimit -c unlimited;

# %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

cp /etc/sysctl.conf{,.bak};
echo 'kernel.core_pattern=/var/log/rlog/cacheproxy/core-%e-%s-%u-%g-%p-%t' \
    >> /etc/sysctl.conf;
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
```
{
    "insecure-registries": ["172.21.1.73:8080"],
    "registry-mirrors": ["http://172.21.1.73:8080"]
}
```

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

```

docker build \
    -t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
    -f ./dockerfiles/autopost.Dockerfile .
```

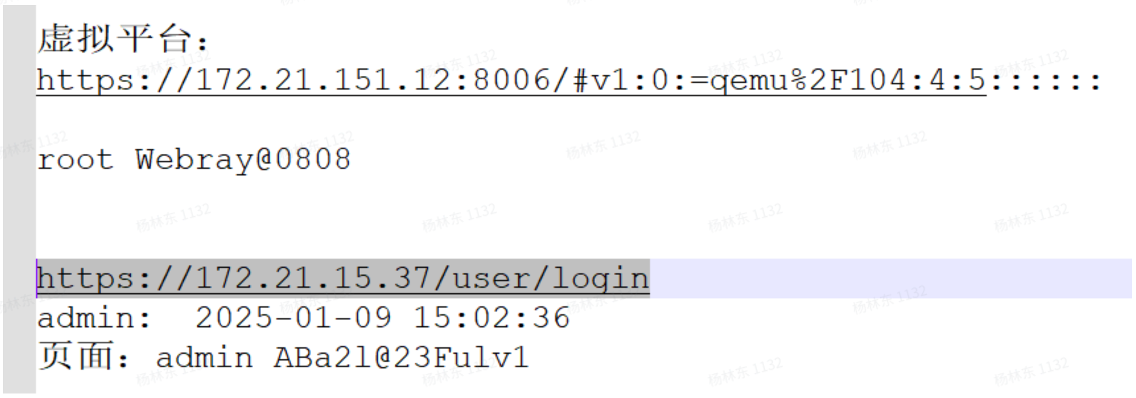
Git

代码块

```
https://kkgithub.com/
https://hub.yzuu.cf/

# 提交时转换为LF，检出时转换为CRLF
git config --global core.autocrlf true

# 提交和检出时都不转换
git config --global core.autocrlf false

# 提交时转换为LF，检出时不转换
git config --global core.autocrlf input
```

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

```

Env 配置

代码块

```
CC=x86_64-linux-musl-gcc
CGO_ENABLED=1
CGO_LDFLAGS=-O2 -g -static
GOROOT=C:/Program Files/go

```

代码块

```
func(*Server)

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

func Chain(m ...SFTPMiddleware) SFTPMiddleware {
    return func(next Handler) Handler {
        for i := len(m) - 1; i >= 0; i-- {
            next = m[i](next)
        }
        return next
    }
}

s.middleware = []{
    Recovery(),
    SSHAuth(ss.SSHConfigs()),
    WhiteList(ss.GetProbeEgressIP),
    Status(ss.GetProbeStatus),
    Logging(),
    ConnectionLimit(ss.Limit)
}
next = ServeSFTP(ctx context.Conext, conn net.Conn) error {
    s.probe.AppendSftpReqServer(p.Username, channel, connID)
    defer s.probe.RemoveSftpReqServer(p.Username, connID)
    defer server.Close()
    server.Serve()
}
go Recovery(
    SSHAuth(
        WhiteList(
            Status(
                go ps.Active()
                defer func(){
                    if cnt==0 {
                        go. ps.Logoff()
                    }
                }()
                Logging(
                    ConnectionLimit(next)
                )
            )
        )
    )
)

Logging: info user "ridward" logged out
go.ps.Logoff
```

代码块

```
https://blog.csdn.net/Eivene/article/details/140774552
- reflect
http://bbs.itying.com/topic/676cf34b5b798701ddf78072

- 服务器管理
https://172.21.150.16/ui/#/host/vms/89
root/Esxi#0926
- 150.39 的账密 root/Superman@12345

- proxmox 账户
root/Webray@0808

- 删除
点中-more-remove

探针: https://172.21.15.37/user/login
admin/ABa2l@23Fulv1

系统管理/集中配置管理


- img 下载
- 外部: http://172.18.9.4:8001/products/fap-cacheproxy/
- 内部: http://172.21.1.25:8000/products/fap-cacheproxy/


上传 data(ibpd-uat-01), upload,

创建系统,
两个盘, 128G, 256G
```

```
2 sockets, 4 cores = 8 cres
16384 Memory MiB(16G)

- 登录账号: root/webray++
- 安装密码: WebRW2r$
输入 y 确认

等待好长时间....

y; xfs; rdata/cacheproxy; y; roboot;

等好久...
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

配置网络
vlan -A -v MngtVlan -f 172.21.150.107 -m 255.255.255.0;
vlan -S;
route -A -i 0.0.0.0 -m 0.0.0.0 -g 172.21.150.1;
route -S;

settime
help
upsshd start


[autopost]

docker build \
-t registry.ibdp.webray.com.cn:51808/fap/autopost:latest \
-f ./dockerfiles/autopost.Dockerfile .;

docker save registry.ibdp.webray.com.cn:51808/fap/autopost -o 'autopost.tar';

[autopost]
- sftp_policy_path: /policy-data
    pull 模式要忽略的远程 server 的策略存储目录
    ~~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)~~
- file_path: /home/
    定期清理老文件的目录, NewMonitorRepo 中的的 DataDir:
    pull 模式下文件同步到的目录
    ~~push 模式不用此配置~~
- policy_file_path: /home/policy-data
    ~~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

[cacheproxy]
```

```
RootAccount() 一致维护 root 用户
- data_dir: /home/cacheproxy/data
    探针文件夹创建的目录
    策略解压到的目录
    定期清理的目录 (NewMonitorRepo)
    root 用户的家目录
- app_dir: /home/cacheproxy
    创建一个 policy-data-backup 目录备份
    cert 所在目录
    data.db 所在目录
    处理请求的时候看了这个目录的信息??? HandleConn: IsDiskWork()
- policy_data_dir: /home/cacheproxy/data/policy-data
    push 过来的策略的存储路径
    定期检测并需要解压 .tgz

p.HomeDir 是不同 sftpServer 的目录

SystemProbeRecover
```

代码块

```
scp -O -l 1024 /d/... root@172.21.150.28:/cacheproxy/


# 172.21.150.27
export GOPROXY="http://172.21.3.11:3000"
账密: admin/Admin123, 端口是 8000


- cmd/cacheproxy/main.go
////go:embed web/*

- go.mod
toolchain go1.21.4

- configs/config.json
data_dir
app_dir
policy_data_dir


https://github.com/protocolbuffers/protobuf/releases
# exec: "protoc": executable file not found in %PATH%
https://github.com/protocolbuffers/protobuf/releases/download/v29.2/protoc-
29.2-win64.zip;
```

```


VERSION=1.0;
# https://www.jianshu.com/p/10ae8a5a7704
# go get github.com/go-kratos/kratos/cmd/kratos/v2@latest;
go install github.com/go-kratos/kratos/cmd/kratos/v2@latest;

kratos proto client ./internal/conf/conf.proto;
kratos proto client ./api/cacheproxy/v1/cacheproxy.proto;



go build -ldflags "-X main.Version=$(VERSION)";
chmod a+x /usr/local/go/bin/kratos;
go mod tidy;
cd cmd/cacheproxy;
go build;
./cacheproxy -conf ../../configs/config.json;
dlv --listen=:2345 --headless=true --api-version=2 --accept-multiclient exec
./cacheproxy

缓存转发的 root_passwd 要配置为 root
# ../configs/autopost.yaml 要配置 sftp_address 和 3 个目录
./autopost -conf ../configs/autopost.yaml -mode push
上传文件到单导的 /cacheproxy/home/policy-data
单导日志类似:
caller=biz/autopost_push.go:203 ... \
    policy-data/20241221181530_policy.tgz.part successfully

缓存转发的日志类似:
    caller=biz/monitor.go .... \
    job: /home/cacheproxy/data/policy-data/20241221181530_policy.tgz

文件会被解压到 /home/cacheproxy/data 下
    - 0XSDJI902
    - 0XSDJI905
    - 0XSDJI908
    - E04C81F400
    - SXXAtest11
/home/cacheproxy/data/policy-data
/hoem/cacheproxy/policy-data(??? 是什么)
```

走

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

```
runtime.gcdata: missing Go type information for global symbol .dynsym: size 72
https://www.cnblogs.com/binHome/p/13020178.html

https://www.jianshu.com/p/a73cef45b65c

wire:
https://www.cnblogs.com/jiujuan/p/16136633.html

在 service/service.go
type AppRepo interface {
    ...
}

在 data/data.go
type appRepo struct {
    *mq
    *db
    *dp
}

func NewAppRepo() service.AppRepo {
    appRepo := &appRepo{}
    return appRepo
}

//
google.golang.org/protobuf/types/know/anypb

marshal := v1.Flow { Enable: true, Subset: "127.0.0.1" }
any, err := anypb.New(&marshal);
[type.googleapis.com/engmgr.v1.Flow]:{enable:true, subset:"127.0.0.1"}

grpcurl -plaintext -H "x-user-id: 1" 172.21.150.109:9000 list;
grpcurl -plaintext -H "x-user-id:1" -d '{}' 172.21.150.109
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

# 自定义 json
https://juejin.cn/post/6844904184651874312
```

代码块

```
# select
https://www.cnblogs.com/qcy-blog/p/18520091
# docker copy
https://developer.baidu.com/article/details/3214252
# break
https://blog.csdn.net/u011461385/article/details/106017483
# chan
https://blog.csdn.net/chenchongg/article/details/86589395
# nil, close chan
https://dave.cheney.net/2014/03/19/channel-axioms
# strace
// 只显示 renameat 的调用情况
strace -f -e trace=renameat go run ./script/download-go go1.22.3

# 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
```

代码块

```
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)
}


```

代码块

```
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.
Packages in the standard library do not have a module path prefix

go install example/user/hello;


The install directory is controlled by the GOPATH and GOBIN environment
variables.
If GOBIN is set, binaries are installed to that directory.
If GOPATH is set, binaries are installed to the bin subdirectory of
the first directory in the GOPATH list.
Otherwise, binaries are installed to the bin subdirectory of
the default GOPATH ($HOME/go or %USERPROFILE%\go)

```

```
go env -u GOBIN;

$(dirname $(go list -f '{{.Target}}' .));

go help importpath;

GOCACHE: $HOME/.cache/go-build;

import "example/user/hello/morestrings";

go mod tidy;

$HOME/.mygo/pkg/mod/

go clean -modcache;
```

代码块

```
go list -f '{{.Target}}';
go env -w GOBIN=/path/to/your/bin;

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;
uintptr;

%T, %v
Zero values;
different type requires an explicit conversion;
Gos switch cases need not be constants, and the values involved need not be integers;
Unlike C, Go has no pointer arithmetic.;

type Vertex struct {
        X int
        Y int
}

(*p).X -> p.X;

var (
        v1 = Vertex{1, 2}  // has type Vertex
        v2 = Vertex{X: 1}  // Y:0 is implicit
        v3 = Vertex{}      // X:0 and Y:0
        p  = &Vertex{1, 2} // has type *Vertex
```

```
)

Slices are like references to arrays;
cap();
a := make([]int, 5)  // len(a)=5;
b := make([]int, 0, 5) // len(b)=0, cap(b)=5;
strings.Join;
The range form of the for loop iterates over a slice or map.
for i := range pow;
delete(m, key)
elem, ok = m[key]

The make function returns a map of the given type, initialized and ready for use;


func fibonacci() func() int {
        i, j := 0, 1
        return func() int {
                i, j = j, i+j
                return i
        }
}

Methods:
The receiver appears in its own argument list
between the func keyword and the method name.
func (v Vertex) Abs() float64 {
        return math.Sqrt(v.X*v.X + v.Y*v.Y)
}


You can only declare a method with a receiver
whose type is defined in the same package as the method.
Pointer receivers;

Go interprets the statement v.Scale(5) as (&v).Scale(5);
p.Abs() is interpreted as (*p).Abs();

type Abser interface {
        Abs() float64
}

func (t *T) M() {
        if t == nil {
                fmt.Println("<nil>")
                return
        }
        fmt.Println(t.S)
}

# Type assertions
t := i.(T);
t, ok := i.(T);

switch v := i.(type);

type Stringer interface {
    String() string
}

func (p Person) String() string {
        return fmt.Sprintf("%v (%v years)", p.Name, p.Age)
}

type error interface {
    Error() string
}

%q;

func Index[T comparable](s []T, x T) int

type List[T any] struct {
        next *List[T]
        val  T
}

The evaluation of f, x, y, and z happens in the current goroutine and
the execution of f happens in the new goroutine.


ch <- v    // Send v to channel ch.
v := <-ch  // Receive from ch, and
           // assign value to v.


Like maps and slices, channels must be created before use:
ch := make(chan int);

ch := make(chan int, 100);

v, ok := <-ch;

```

```
Only the sender should close a channel, never the receiver.
Sending on a closed channel will cause a panic.


select {
case c <- x:
        x, y = y, x+y
case <-quit:
        fmt.Println("quit")
        return
}

mu sync.Mutex;
c.mu.Lock();
c.mu.Unlock();
```

安全外壳协议

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

pg_dump

代码块

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


-O --no-owner
-s --schema-only
```

pg

代码块

```
/install/modules/ray-postgresql/stop.sh;
psql postgresql://user:pass@host:port/dbname
PGPASSWORD='admin1234' psql -U postgres -p 5432 fusion;

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



PGPASSWORD='Mjolnir' pg_dump -d test -U postgres -h 127.0.0.1 --no-comments \
--column-inserts \
--schema-only \
-t 'group' \
-t 'device_group' \
-t 'device' \
-t 'engine' \
> enginemgr.sql;


-s --schema-only，加上只导表结构
```

iptables防火墙

代码块

```
/rayos/cfg/iptables_port.conf     6379 5432
/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 \
    "nohup python3 -m flask rq worker &>> /var/log/webraydb/custom_worker.log
&"
```

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

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

Pycharm

代码块

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

防火墙

代码块

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

Es

代码块

```
network.host: 0.0.0.0
discovery.seed_hosts: ["127.0.0.1"]

network.bind_host
network.public_host
network.host
discovery.seed_hosts:  集群主机列表
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

curl -K https://elastic:elastic@172.21.15.234/_settings?pretty

```

alembic 使用

代码块

```
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;
ssh-copy-id -i ~/.ssh/id_*.pub root@172.21.15.234;
useradd -m -s /bin/bash lindyang;
passwd lindyang;

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;


cat /etc/os-release
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：

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

```
ARG DEBIAN_FRONTEND=noninteractive
ENV TZ=Asia/Shanghai
RUN DEBIAN_FRONTEND=noninteractive TZ=Etc/UTC apt-get -y install tzdata;

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
{
    "registry-mirrors": [
        "https://dockerproxy.cn",
        "https://docker.rainbond.cc",
        "https://docker.udayun.com",
        "https://docker.211678.top"
    ]
}
EOF

# 重载，重启Docker服务
sudo systemctl daemon-reload
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

代码块

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

代码块

```
minikube start \
    --vm-driver none \
    --driver=docker \
    --image-mirror-country='cn' \
    --image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers;

minikube status;
minikube dashboard;
minikube ssh;
eval $(minikube docker-env);

kubectl cluster-info;
kubectl get nodes;
kubectl describe node minikube

minikube image load kubia;
minikube ssh;
docker tag kubia luksa/kubia:1.0;
exit;
kubectl create deployment kubia --image=luksa/kubia:1.0;
kubectl get deployments;

kubectl get pods;
kubectl describe pod;

kubectl expose deployment kubia --type=LoadBalancer --port 8080;
kubectl get svc;

k get nods/pods/deployments/pods;

kubectl api-resources;

# If you use Minikube to create the cluster,
# no load balancer is created

minikube service kubia --url;
minikube ip;
kubectl scale deployment kubia --replicas=3;

kubectl get deploy;

kubectl get pods -o wide;

kubectl get nodes minikube -o yaml;

kubectl proxy;
http://127.0.0.1:8001/api/v1/nodes/minikube;

kubectl explain nodes;
kubectl explain node.spec;  # --api-version
kubectl explain pods --recursive;

kubectl get ev -o wide;
kubectl get ev --field-selector type=Warning;

kubectl explain events;

kubectl run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml > mypod.yaml;

kubectl apply -f kubia.yaml;

k run kubia --image=luksa/kubia:1.0 --dry-run=client -o yaml;

k describe pod kubia; minikube ssh; curl 172.17.0.X:8080;

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

kubectl get pod kubia -o wide;

kubectl run \
--image=ubuntu:latest -it \
--restart=Never \
--rm client-pod curl 172.17.0.7:8080;

kubectl port-forward kubia ~~8080:~~8080;
kubectl port-forward --help;
kubectl logs ~~--timestamps=true~~ -f kubia;
    --since=2m
    --since-time=2020-02-01T09:50:00Z
    --tail=10
    -p (--previous)

kubectl exec kubia -- ps aux;

kubectl attach;

kubectl port-forward kubia-ssl 8080 8443 9901;
curl https://example.com:8443 --resolve example.com:8443:127.0.0.1 -k;

kubectl delete po kubia-ssl --grace-period 10;
    spec.terminationGracePeriodSeconds
    metadata.deletionGracePeriodSeconds


kubectl logs kubia-ssl -c kubia;
kubectl logs kubia-ssl --all-containers;

kubectl exec -it kubia-ssl -c envoy -- bash;

k delete pod kubia --wait=false;

kubectl delete -f kubia-ssl.yaml;
kubectl delete -f kubia.yaml,kubia-ssl.yaml;
kubectl apply -f Chapter05/;
kubectl delete -f Chapter05/;
    --recursive

kubectl delete po --all;

kubectl delete all --all;

kubectl delete events,all --all;

kubectl exec kubia-liveness -c envoy -- tail -f /var/log/envoy.admin.log;

kubectl logs kubia-liveness -c envoy -p;
```

代码块

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

kubectl get nodes -o jsonpath='{.items[*].status.addresses}';
kubectl get nodes -o \
jsonpath='{.items[*].status.addresses[?(@.type=="InternalIP")].address}';

# http://kt1bernetes.io/docs/user-guide/jsonpath

minikube service kubia-nodeport;
# minikube sevrvice <service-name> [-n <namespace>]

externalTrafficPolicy: Local;

minikube addons list;
minikube addons enable ingress;
apiVersion: networking.k8s.io/v1;
```

代码块

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: kubia
spec:
  rules:
  - host: kubia.example.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: kubia-nodeport
            port:
              number: 80
```

代码块

```
docker build -t kubia .;
删除 DaemonSet 也会删除 pod;

# job 资源
kubectl get jobs;

# completions: 5
# parallelism: 2

# 有问题
kubectl scale job multi-completion-batch-job --replicas 3;

activeDeadlineSeconds
spec.backoffLimit=6 (default)

CronJob
分钟 小时 每月中的第几天 月 星期几(0是星期天)
startingDeadlineSeconds


创建 service 后, 需要
minikube ssh; # 登录
```

```
curl `kubectl get svc | awk '{print $3}'`  # CLUSTER-IP

kubectl exec kubia-manual -- curl -s http://10.99.94.217;  # 没有 -s, -- 是非必须得

sessionAffinity: ClientIP;  # 不支持http cookie
kubectl exec kubia-manual -- env;
dnsPolicy;

backend-database.default.svc.cluster.local;

kubectl exec -it kubia-manual -- bash;
curl -s http://kubia.default.svc.cluster.local;
curl http://kubia;

cat /etc/resolv.conf;
ping kubia;  # 无法ping通, 因为是虚拟IP, 只有与port结合才有意义
kubectl describe svc kubia | grep -i endpoint;
kubectl get endpoints kubia;
curl http://external-service/echo;

# kubia ExternalName
```

代码块

```
kubectl logs mypod --previous;
kubectl edit rc kubia;

export KUBE_EDITOR="/usr/bin/vim";
export EDITOR="/usr/bin/vim";
kubectl scale rc kubia --replicas=4;
kubectl delete rc kubia --cascade=false;

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.

kubectl api-versions;
replicaset.apps/kubia created;
kubectl get rs;
kubectl delete rs kubia;

# 使用 DaemonSet 在每个节点上运行一个 pod
# 使用 DaemonSet 只在特定的节点运行 pod
# 节点可以被设置为不可调度, 但DaemonSet可以绕过它
```

代码块

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubia
  template:
    metadata:
      labels:
        app: kubia
    spec:
      containers:
      - name: kubia
        image: kubia
        imagePullPolicy: Never
```

- 匹配表达式

代码块

```yaml
selector:
    matchExpressions:
    - key: app
      operator: In
      values:
      - kubia
```

- 操作员

> - 在
> - 不在
> - 存在
> - 不存在

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubia-liveness
spec:
  containers:
  - image: kubia-unhealthy
    imagePullPolicy: Never
    name: kubia
    livenessProbe:
      httpGet:
        path: /
        port: 8080
      initialDelaySeconds: 15
```

代码块

```
- pod 共享 network, UTS, IPC, PID(默认未开启)
- 文件系统隔离，但是可以通过 Volume 共享文件目录
- localhost 可以与同一 pod 中的其它容器通信
kubectl get po kubia-82d24 -o yaml;
kubectl explain pods;
kubectl explain pod.spec;
kubectl create -f lindyang/kubia-manual.yaml;

kubectl describe pod kubia-manual | grep 172;
minikube ssh;
curl 172.17.0.7:8080;

kubectl logs kubia-manual -c kubia;

# 不通过 service 与 pod 通讯
kubectl port-forward kubia-manual 8888:8080;
curl http://127.0.0.1:8888;

kubectl get pods --show-labels;
kubectl get pod -L creation_method,env;

kubectl label po kubia-manual creation_method=manual --overwrite
# 删除标签, -
kubectl label po kubia-manual createion_method-

kubectl get po -l creation_method=manual;

kubectl get po -l env;

kubectl get po -l '!env';
env!=prod
'env in (prod,dev)'
'env notin (prod,dev)'

kubectl label node minikube gpu=false;

kubectl annotate pod kubia-manual mycompany.com/someannotation="foo bar";

kubectl get po --namespace kube-system;

kubectl get ns;

kubectl delete namespace custom-namespace;
kubectl create namespace custom-namespace;
kubectl create -f lindyang/kubia-manual.yaml -n custom-namespace;

命名空间不允许包含点号.

alias kcd='kubectl config set-context $(kubectl config current-context) --namespace'

kubectl delete pod po1 po2;

kubectl delete po -l creation_method=manual;

kubectl delete ns custom-namespace;

kubectl delete po --all;

kubectl delete all --all;
```

- 命名空间

代码块

```
apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace
```

代码块

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: kubia-gpu
spec:
  nodeSelector:
    gpu: "true"
  containers:
  - name: kubia
    image: kubia
    imagePullPolicy: Never
```

代码块

```bash
minikube status;
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);

kubectl run --image=kubia --port=8080 kubia;
kubectl get pod kubia -o yaml -n default;
kubectl describe pod kubia;

https://www.cnblogs.com/xiao2/p/16047455.html;
minikube image load kubia;
minikube image build -t <IMAGE_NAME> .;

kubectl get nodes;

kubectl apply -f lindyang/deployment.yaml;
kubectl delete -f lindyang/deployment.yaml;
kubectl port-forward kubia-868cd55b98-qzz79 8080:8080;
kubectl expose pod kubia-868cd55b98-qzz79 --type=LoadBalancer --name kubia-http;
# minikube 不支持 LoadBalancer;
minikube service kubia-http;
kubectl get rc;
kubectl get services; # src

kubectl scale --replicas=3 -f lindyang/deployment.yaml;
```

代码块

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: kubia
spec:
  replicas: 3
---    selector:
---    app: kubia
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
        - name: kubia
          image: kubia
          imagePullPolicy: Never
          ports:
          - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  type: LoadBalancer
  selector:
    app: kubia
  ports:
  - port: 8080
    targetPort: 8080
```

代码块

```yaml
apiVersion: apps/v1
kind: Deployment
#kind: ReplicaSet
metadata:
  name: kubia
  labels:
    name: kubia
spec:
  replicas: 3
  selector:
    matchLabels:
      name: kubia
  template:
    metadata:
      labels:
        name: kubia
    spec:
      containers:
      - name: kubia
        image: kubia
        imagePullPolicy: Never
        ports:
          - 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

代码块

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

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


minikube start --force --driver=docker \
--image-mirror-country='cn' \
--registry-mirror=https://docker.mirrors.ustc.edu.cn \
--image-repository=registry.cn-hangzhou.aliyuncs.com/google_containers


minikube start \
--kubernetes-version=v1.23.8 \
--image-mirror-country='cn' \
--image-repository='registry.cn-hangzhou.aliyuncs.com/google_containers';


```

```
minikube start --image-mirror-country=cn;

minikube kubectl -- create clusterrolebinding system:anonymous \
--clusterrole=cluster-admin  \
--user=system:anonymous;

kubectl proxy \
--port=8888 \
--address='192.168.1.20' \
--accept-hosts='^.*'  >/dev/null 2>&1 &;


minikube kubectl -- get pods -A;
minikube kubectl -- logs kube-proxy-2z5gg -n kube-system;
sudo sysctl -w net.netfilter.nf_conntrack_max=524288;

minikube kubectl -- describe pods .. -n kubernetes-dashboard
# minikube start --disk-size="10g" \
#--image-mirror-country="cn"  \
#--image-repository="registry.cn-hangzhou.aliyuncs.com/google_containers";
```

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;

docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/metrics-scraper:v1.0.8 \
docker.io/kubernetesui/metrics-scraper:v1.0.8;

docker pull registry.cn-hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0;

docker tag registry.cn-hangzhou.aliyuncs.com/google_containers/dashboard:v2.7.0 \
kubernetesui/dashboard:v2.7.0;


registry.hub.docker.com/kubernetesui/dashboard:v2.1.0;
registry.hub.docker.com/kubernetesui/metrics-scraper:v1.0.4;
```

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;

~/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
查看单个: SHOW HOSTS GRAPH、SHOW HOSTS STORAGE、SHOW HOSTS META;
ADD HOSTS 127.0.0.1:9779
心跳: heartbeat_interval_secs
GO语句采用的是walk类型路径;
MATCH、FIND PATH和GET SUBGRAPH语句采用的是trail类型路径;

$$     表示边的终点;
$^     表示边的起点;
$-     表示管道符前面的查询输出的结果集;

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;
// 重建索引确保能对已存在数据生效。
nebula> REBUILD TAG INDEX player_index_1;

create edge index follow_index_1 on follow(degree);
rebuild edge index follow_index_1;

```

Windows git bash设置字体大小

代码块

```
echo > ~/.minttyrc <<EOF
Font=Consolas
FontHeight=14
EOF
```

搜狗输入法

代码块

```
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。
vim ~/.config/sogoupinyin/conf/env.ini

```

```
在该配置文件中添加一行
DefaultSymbol=0

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';

select datid, datname,
    pid,
    usesysid,
    usename,
    client_addr,
    client_port,
    query_start,
    state
    --,query
FROM pg_stat_activity;
```

代码块

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

```

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

代码块

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

Jenkins

代码块

```
FROM ubuntu:16.04
MAINTAINER james@example.com
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 -
COPY gpg /tmp/gpg
RUN cat /tmp/gpg | apt-key add -; rm /tmp/gpg
RUN apt-get clean && rm -rf /var/lib/apt/lists/*
#RUN echo deb https://apt.dockerproject.org/repo ubuntu-trusty main > /etc/apt/sources.list.d/docker.list
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 iptables ca-certificates git-core
RUN apt-get install -qqy --allow-unauthenticated docker-ce
#RUN apt-get install -qqy --allow-unauthenticated openjdk-9-jdk
RUN add-apt-repository -y ppa:openjdk-r/ppa
RUN apt-get update -qq && apt-get install -qqy openjdk-17-jdk

ENV JENKINS_HOME /opt/jenkins/data
#ENV JENKINS_MIRROR http://mirrors.jenkins-ci.org

RUN mkdir -p $JENKINS_HOME/plugins
#RUN curl -sf -o /opt/jenkins/jenkins.war -L $JENKINS_MIRROR/war-stable/latest/jenkins.war
COPY jenkins.war /opt/jenkins/jenkins.war

#RUN for plugin in chucknorris greenballs scm-api git-client git ws-cleanup ;\
#    do curl -sf -o $JENKINS_HOME/plugins/${plugin}.hpi \
#       -L $JENKINS_MIRROR/plugins/${plugin}/latest/${plugin}.hpi ; done
COPY ./*.hpi $JENKINS_HOME/plugins/

ADD ./dockerjenkins.sh /usr/local/bin/dockerjenkins.sh
RUN sed -i 's/^docker daemon/dockerd/' /usr/local/bin/dockerjenkins.sh
RUN chmod +x /usr/local/bin/dockerjenkins.sh

#VOLUME /var/lib/docker

EXPOSE 8080

# docker daemon & => dockerd
ENTRYPOINT [ "/usr/local/bin/dockerjenkins.sh" ]


$ sudo mkdir -p /var/jenkins_home
$ cd /var/jenkins_home
$ sudo chown -R 1000 /var/jenkins_home

docker run --privileged -d \
-p 8080:8080 \
-v /var/jenkins_home:/opt/jenkins/data \
-v /var/run/docker.sock:/var/run/docker.sock \
--name jenkins \
lindyang/jenkins

sudo sed -i 's/https:/http:/' /var/lib/docker/volumes/data/_data/hudson.model.UpdateCenter.xml;
# http://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json

# https://baijiahao.baidu.com/s?id=1742735186119625775&wfr=spider&for=pc
# https://www.cnblogs.com/yshc/p/10621224.html

https://updates.jenkins.io/download/plugins/skip-certificate-check/1.1/skip-certificate-check.hpi

- Dashboard/Manage Jenkins/插件管理/advanced
- https://www.cnblogs.com/fengwenqian/p/13534786.html

http://updates.jenkins.io/update-center.json;
https://gitee.com/Errorcode500/docker-jenkins-sample.git

/opt/jenkins/data/jobs/${JOB_NAME}/workspace

cat > Dockerfile <<EOF
FROM ubuntu:16.04
RUN echo "while true; do date; echo $WORKSPACE; sleep 1; done" > /usr/bin/run.sh
CMD ["/bin/bash", "/usr/bin/run.sh"]
EOF
docker build -t loop_test_$OS .
docker run -d --name loop_test.$OS loop_test_$OS
docker logs loop_test.$OS
docker stop loop_test.$OS
docker rm loop_test.$OS
docker rmi loop_test_$OS

```

TProv

代码块

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

生成证书

代码块

```
openssl genpkey -algorithm RSA -out private.key -pkeyopt rsa_keygen_bits:2048;

openssl req -new -key private.key -out request.csr -subj \
"/C=CN/ST=Beijing/L=Beijing/O=WebRay/OU=IT/CN=cacheproxy.com/emailAddress=ridward@qq.com"

openssl x509 -req -days 36135 -in request.csr -signkey private.key -out certificate.crt;

openssl pkey -in private.key -text -noout;
openssl req -in request.csr -text -noout;
openssl x509 -in certificate.crt -text -noout;
```

升级系统

代码块

```
上传包到 172.21.15.38
/var/log/webraydb/Dump/NJ/WebRay/1cf935508eb43c32a229826cf8c1be59

在要升级的系统上以 admin 执行
# patchup/patchall
patchup -u 1cf935508eb43c32a229826cf8c1be59 -p E9DCCB48 \
-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

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

Rbenv

代码块

```

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

FROM ubuntu:20.04
RUN apt-get -yqq update
RUN DEBIAN_FRONTEND=noninteractive TZ=Asia/Shanghai apt-get -y install tzdata
RUN apt-get -yqq install build-essential nodejs
RUN apt-get -yqq install git wget libffi-dev zlib1g-dev libyaml-dev
RUN rm /bin/sh && ln -s /bin/bash /bin/sh
#RUN wget -q -O ~/.rbenv/cache/ruby-3.4.1.tar.gz -c https://ftp.ruby-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
RUN git clone --depth=1 https://github.com/rbenv/ruby-build.git /root/.rbenv/plugins/ruby-build
RUN /root/.rbenv/plugins/ruby-build/install.sh
ENV PATH /root/.rbenv/bin:/root/.rbenv/shims:$PATH
RUN echo 'eval "$(rbenv init -)"' >> /root/.bashrc
RUN mkdir -p /root/.rbenv/cache
COPY ruby-3.4.1.tar.gz /root/.rbenv/cache/
RUN rbenv install 3.4.1
RUN rbenv global 3.4.1
RUN gem sources --remove https://rubygems.org/ --add https://mirrors.tuna.tsinghua.edu.cn/rubygems/;
RUN gem update --system
RUN gem install ffi -v 1.17.1
#RUN gem install -s https://gems.ruby-china.com/ jekyll -v 2.5.3
RUN gem install jekyll -v 2.5.3
VOLUME /data
VOLUME /var/www/html
WORKDIR /data

ENTRYPOINT ["jekyll", "build", "--destination=/var/www/html"]

# gem install faraday-retry
```

Uwsig

代码块

```
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实战》

代码块

```
https://wangwei1237.github.io/Kubernetes-in-Action-Second-Edition/

ip link set docker0 up;
DOCKER_HOST;
redis-server --protected-mode no;
docker-compose logs/ps/stop/start/rm/kill;

curl --unix-socket /var/run/docker.sock http://docker/info

docker -H tcp://127.0.0.1:2375 images;
```

交叉编译

代码块

```
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


```