一次"诡异"的 Ansible 密码问题排查,最后的"真相"竟是这样 腦 廊



背景

在做大批量运维的时候,DBA 需要掌握和使用 ansible。今天有同事使用 ansible 遇到了一个奇怪现象,和我交流了一下,我发现这个现象很奇怪,也很有趣,我认为是个 BUG,于是排查,之后有了这篇文章。

现象

同事使用的是 ansible2.7.8,我使用的是最新版 python3.11+ ansible2.14,我们都复现了这个现象。这个问题简而言之是,使用比较特殊的密码组合作为主机密码,ansible 在使用上会遇到问题,导致正确的密码而无法连接。

以下是我的环境:

			129
角色	hostname	IP	使用的密码
ansible server	192-168-199-121	192.168.199.121	不涉及
有问题的受控机	192-168-199-99	192.168.199.99	1#fander
没问题的受控机	192-168-199-131	192.168.199.131	Root-123

同事疑问1--特殊密码, ansible无法连接会报错



分类列表	
#原创	43篇
# 活动	3篇
# 技术干货	333篇
精品课程 ❤ 3对1教学服务	免费资料>



近期文章

1.YK人工智能(六)——万字长文学会...

同事设置了一个形如"1#xxxxxx"的密码,使用 ssh root@192.168.199.99 连接是完全没有问题的,但使用 ansible 连接则会报错,"Invalid/incorrect password: Permission denied, pleas e try again." 无法连接。这个密码经过我的研究,是有规律,1位到任意长度的数字 + '#' + 任意长度字符,ansible 会报错。

也就是, 以下密码会报错

https://blog.51cto.com/u_15576159/5868439

- 1#fander
- 12#fander

以下密码不会报错

- 1a#fander
- 1@fander

同事疑问2——此特殊密码,ansible无法连接会报错,但有时却能连接不报错 排查过程

1. ansible -vvvvv 排查

-v 参数可以让 ansible 输出 debug 日志, v 越多越详细。当然了,可能并不需要五个 v, 我不知道要打多少个 v 时,我就把 v 打满。(mysqlbinlog 我打 3个 v)

```
(192.168.199.99) ESTABLISH SSH CONNECTION FOR USER: root
(192.168.199.99) SSH: ansible.cfg set ssh_args: (-(-0)(ControlMaster-auto)(-0)(ControlPersist=66s)
(192.168.199.99) SSH: ANSIBLE_RENOTE_USER/remote_user/ansible_user/user/-u set: (-0)(User-"root")
(192.168.199.99) SSH: SSH: Set ssh_common_args: ()
(192.168.199.99) SSH: SEEC sshpass =d10 ssh -vvv -C -O ControlMaster-auto =O ControlPersist=60s =O StrictMostKeyChecking=no =O 'User="root"' -O ConnectTimeout=10 =O 'ControlPath="/root/.ansible/cp/e2f9f7759b"' 192.168.199.99 '/bin/sh -c '''' echo ~root && sleep 0''''''
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: Reading configuration data /etc/ssh/ssh_common_args: ()
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: Reading configuration data /etc/ssh/ssh_common_args: ()
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: Reading configuration data /etc/ssh/ssh_common_args: ()
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: Reading configuration data /etc/ssh/ssh_common_args: ()
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: Reading configuration data /etc/ssh/ssh_common_args: ()
(192.168.199.99) (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips = 26 Jan 2017\r\ndebug1: dentity file /root/.ssh/file or directory\r\ndebug1: dentity
```

如图红圈和横线,这两处很关键。

红圈告诉我,我在执行 ansible 时,其实底层调用的是 sshpass 和 ssh。

横线这一处,报有个文件不存在,我对比了执行正常连接的机器(192.168.199.131),是不会报这个的,所以此处属于异常,需要排查。

这些日志输出其实一团糟,可以粘贴到 notepad++, 用以下方法美化输出。

粘贴到 notepad++ 只有两行,通过替换 \r\n 为换行符即可。

首先, 我先把 \r\n 替换为 fanderissb

然后, 再以扩展模式, 替换回来

2.《哪吒2》破百亿,中小企业也能"逆… 3.c#开发windows应用程序几个小技巧



现在就比较好阅读了,这一处就是不正常的。

```
H new 32□
      EXEC sshpass -d10 ssh -vvv -C -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=no -o 'User="root": -o ConnectTimeout=10 -o 'Co
      <192.168.199.99> (5, b'', b'OpenSSH_7.4pl, OpenSSL-1.0.2k-fips 26 Jan 2017
      debugl: Reading configuration data /etc/ssh/ssh_config
      debugl: /etc/ssh/ssh_config line 58: Applying options for *
      debugl - auto-mux - Trying existing master
      debug. : Control socket "/root/.ansible/cp/e2f9f7759b" does not exist
      debug2: resolving "192.168.199.99" port 22
      debug2: ssh connect direct: needpriv 0
      debugl: Connecting to 192.168.199.99 [192.168.199.99] port 22.
      debug2: fd 3 setting O NONBLOCK
      debugl: fd 3 clearing O NONBLOCK
      debugl: Connection established.
      debug3: timeout: 9997 ms remain after connect
     debugl: permanently_set_uid: 0/0
debugl: key_load_public: No such file or directory
     debugl: identity file /root/.ssh/id_rsa type -1
      debugl: key_load_public: No such file or directory
      debugl: identity file /root/.ssh/id_rsa-cert type -1
      debugl: key_load_public: No such file or directory
      debugl: identity file /root/.ssh/id_dsa type -1
      debugl: key_load_public: No such file or directory
      debugl: identity file /root/.ssh/id_dsa-cert type -1
      debugl: key_load_public: No such file or directory
  24 debugl: identity file /root/.ssh/id_ecdsa type -1
     debugl: key_load_public: No such file or directory
     debugl: identity file /root/.ssh/id_ecdsa-cert type -1
      debugl: key_load_public: No such file or directory
  28 debugl: identity file /root/.ssh/id_ed25519 type -1
  debugl: key_load_public: No-such-file-or-directory
debugl: identity_file-/root/.ssh/id_ed25519-cert-type--1
     debugl: Enabling compatibility mode for protocol 2.0
      debugl: Local version string SSH-2.0-OpenSSH_7.4
                                                             version OpenSSH_7.4
      debugl: Remote protocol version 2.0. remote softwa
     debugl: match: OpenSSH 7.4 pat OpenSSH* compat 0x04000000
  35 debug2: fd 3 setting O NONBLOCK
     debugl: Authenticating to 192.168.199.99:22 as \'root\'
  37 debug3: hostkeys_foreach: reading file "/root/.ssh/known_hosts"
  debug3: record_hostkey: found key type ECDSA in file /root/.ssh/known_hosts:2
  39 debug3: load_hostkeys: loaded 1 keys from 192.168.199.99
  40 debug3: order_hostkeyalgs: prefer hostkeyalgs: ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@openssh.com,ecdsa-sha2-nistp5
  41 debug3: send packet: type 20
  42 debug1: SSH2_MSG_KEXINIT sent
  43 debug3: receive packet: type 20
  44 debug1: SSH2_MSG_KEXINIT received
  45 debug2: local client KEXINIT proposal
  46 debug2: KEX algorithms: curve25519-sha256,curve25519-sha2569libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-gro
  47 debug2: host key algorithms: ecdsa-sha2-nistp256-cert-v01@openssh.com, ecdsa-sha2-nistp384-cert-v01@openssh.com, ecdsa-sha2-nistp521-cert-v01@openssh
  48 debug2: ciphers ctos: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,aes128-cbc,aes1
  49 debug2: ciphers stoc: chacha20-poly1305@openssh.com, aes128-ctr, aes192-ctr, aes256-ctr, aes128-gcm@openssh.com, aes256-gcm@openssh.com, aes128-cbc, aes1
  debug: MACs ctos: umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@opens
  61 debug2: MACs stoc: umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@opens
  52 debug2: compression ctos: zlib@openssh.com, zlib, none
  53 debug2: compression stoc: zlib@openssh.com,zlib,none
                                                                                                        (一) 芬达的数据库学习笔记
  54 debug2: languages ctos:
  55 debug2: languages stoc:
  56 debug2: first_kex_follows 0
```

经过研究,这些 debug 日志其实来源于我前面红圈的命令行 sshpass xxx 的输出。

```
|root@192-168-199-121 - | # sshpass -d10 ssh -vvv -C -o ControlPaster-auto -o ControlPersist=60s -o StrictHostKeyChecking-no -o 'User-"root"' -o ConnectTimeout=10 -o 'ControlPath-"/root/.ansible/cp/e2f97f759b" 192.168.199.99 '/bin/sh -c ''''''echo -root && sleep 0'''''''

OpenSSH_7.4p1, OpenSSL 1.0.2k-fips 26 Jan 2017

debugl: Reading configuration data /etc/ssh/ssh_config

debugl: control sockat "/root/.ansible/cp/e2f9f7f59b" does not exist

debugl: control sockat "/root/.ansible/cp/e2f9f7f59b" does not exist

debugl: connecting to 192.168.199.99 port 22

debug2: ssh_connect_direct: needpriv 0

debug1: fd 3 setting 0 MOMBLOCK

debug1: fd 3 setting 0 MOMBLOCK

debug1: fd 3 clearing 0 MOMBLOCK

debug1: fornection established.

debug3: timeout: 9997 ms remain after connect

debug1: identity file /root/.ssh/id_rsa type -1

debug1: identity file /root/.ssh/id_rsa type -1

debug1: identity file /root/.ssh/id_sa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa type -1

debug1: key_load_public: No such file or directory

debug1: identity file /root/.ssh/id_dsa-cert type -1

debug1: key_load_public: No such file or directory
```

查阅资料和整理,原理大概是这样的:

- ansible 获取变量 ansible_ssh_pass 的值 ->
 - 。 存到一个数字为10的文件描述符(sshpass的那个 -d10 参数的意思) ->
 - sshpass 程序读取和传输给 ssh (ssh 连接远程服务器不是要输入密码吗? sshpass就是模拟 我们手敲密码给 ssh 的) -> 芬达的数据库学习笔记
 - ssh 到远端服务器,执行 "echo ~root && sleep 0"

正确连接是长这个样子的,命令结果是输出"/root\n",前面的 0 是命令 \$? 的返回码,0 代表执行正常。

而我们密码有问题的服务器返回码是 5, 输出是空

```
<192.168.199.99> SSH: Set ssh_extra_args: ()
<192.168.199.99> SSH: Set ssh_extra_args: ()
<192.168.199.99> SSH: found only ControlPersist; added ControlPath: (-o)(ControlPath="/rook.192.168.199.99> SSH: EXEC sshpass -d10 ssh -vvv -C -o ControlMaster=auto -o ControlPersistible/cp/e2f9f7759b"' 192.168.199.99 '/bin/sh -c '"'"'echo ~root && sleep 0'"'"''
<192.168.199.99> (5, b'', b'OpenSSH_7.4p1, OpenSSL 1.0.2k-fips 26 Jan 2017\r\ndebug1: Reads for *\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: Control socket "/root/.ansitect_direct: needpriv 0\r\ndebug1: Connecting to 192.168.199.9
***This is a control of the control of t
```

```
1. sshpass -d10 ssh -vvv -C -o Cnotallow=auto -o Cnotallow=60s \
2. -o StrictHostKeyChecking=no -o 'User="root"' -o Cnotallow=10 \
3. -o 'Cnotallow="/root/.ansible/cp/e2f9f7759b"' 192.168.199.99 \
4. '/bin/sh -c '"'"'echo ~root && sleep 0'"'"''
```

在这句命令中,Cnotallow=60s 是 ssh 的参数,代表建立一个长链接,保持 60 秒,这个长链接就是建立在 Cnotallow=/root/.ansible/cp/e2f9f7759b 的路径下。

实际上,就是这个 ControlPersist ,可以解答同事疑问2——此特殊密码,ansible无法连接会报错,但有时却能连接。

为了方便解释,我把 ansible 服务器的这个 ControlPersist 调大到 600 秒。

```
    [root@192-168-199-121 ~]# cat /etc/ansible/ansible.cfg
    [defaults]
    host_key_checking = False
    [ssh_connection]
    ssh_args = -C -o Cnotallow=auto -o Cnotallow=600
```

首先, 我登录远端服务器, 先把密码改回常规密码。

- [root@192-168-199-99 ~]# echo "Root-123"|passwd --stdin root
 Changing password for user root.
 passwd: all authentication tokens updated successfully.
- 接着, 我配置好 ansible 服务器的 hosts 文件。

```
    [root@192-168-199-121 ~]# cat /etc/ansible/hosts
    [fander_vm]
    192.168.199.99 ansible_user=root ansible_ssh_pass="Root-123"
```

然后, 开始测试。完全没有问题, 能连通。

执行 ps -ef 能观察到,我们第一次连接完后,ssh 并没有断开,有一个背景执行的长链接,他实际上是一个多路复用的 socket 连接,后续我们再连远端服务器时就是复用他,不需要重新验证密码。

```
1. [root@192-168-199-121 ~]# ps -eflgrep ssh

2. root 860 1 0 17:18 ? 00:00:00 /usr/sbin/sshd

3. root 945 860 0 17:18 ? 00:00:04 sshd: root@pts/d

4. root 9220 1 0 22:40 ? 00:00:00 ssh: /root/.ans/sincot 9267 1009 0 22:41 pts/0 00:00:00 grep --color=au/
```

这个时候, 我把远端的服务器密码修改为有问题的密码

```
    [root@192-168-199-99 ~]# echo "1#fander"|passwd --stdin root
    Changing password for user root.
    passwd: all authentication tokens updated successfully.
```

此时, 我的 ansible 服务器的 hosts 配置里仍然用的旧密码

```
    [root@192-168-199-121 ~]# cat /etc/ansible/hosts
    [fander_vm]
    192.168.199.99 ansible_user=root ansible_ssh_pass="Root-123"
```

密码是错误的,那么我还能连吗?答案是——能。这就是连接复用,不需要重新验证密码,直接复用前面的 socket 连接。

所以, 这时你通过 ansible, 密码乱输或者不输密码都能连。

```
1. [root@192-168-199-121 ~]# cat /etc/ansible/hosts
2. [fander_vm]
3. 192.168.199.99 ansible_user=root #我这里直接去掉了密码
4. 5. [root@192-168-199-121 ~]# date;ansible 192.168.199.99 -m shell -a
7. Sun Nov 13 22:49:15 CST 2022 #我复用这个链接的时间
8. 192.168.199.99 | CHANGED | rc=0 >> anaconda-ks.cfg
```

那么这个长链接是创建后的 600 秒自动销毁吗? (提醒, 前面我修改的Cnotallow=600s)

答案——否。因为我在创建连接后,中途复用过这个连接通道,时间是 22:49:15, 所以他消失时间不是创建时间 22:40:28 + 10 分钟, 而是 22:49:15 + 10分钟, 也就是 22:59:15。

```
[root@192-168-199-121 cp]# pwd
    /root/.ansible/cp
4.
     [root@192-168-199-121 cp]# stat e2f9f7759b;date
       File: 'e2f9f7759b'
6.
      Size: 0
                              Blocks: 0 IO Block: 4096 soc
7.
    Device: fd00h/64768d Inode: 68415916 Links: 1
    Access: (0600/srw-----) Uid: ( 0/ root) Gid: ( 0/
9.
    Access: 2022-11-13 22:40:28.680577986 +0800
10.
    Modify: 2022-11-13 22:40:28.615577988 +0800
11.
     Change: 2022-11-13 22:40:28.615577988 +0800
12.
     Birth: -
13.
     Sun Nov 13 22:59:00 CST 2022
14.
15.
16.
     [root@192-168-199-121 cp]# stat e2f9f7759b;date
17.
     stat: cannot stat 'e2f9f7759b': No such file or directory
18.
     Sun Nov 13 22:59:30 CST 2022
19.
21.
     # 超过22:59:15, socket消失了。
```

那同事的疑问2,就可以解释了,有问题的密码依然不能通过 ansible 连接,能连接的假象是因为曾经用正确的密码建立过长链接(这个是 ssh 的参数功能,不是 ansible),后面连接时复用了此连接,没有使用密码认证,所以也就不会报错了。待 ControlPersist 超时后,so cket 销毁,有问题的密码连接就开始报错了。

我们继续排查同事的疑问1。

- ansible 获取变量 ansible_ssh_pass 的值 -> 4
 - 。 存到一个数字为10的文件描述符(sshpass的那个 -d10 参数的意思) -> 3
 - sshpass 程序读取和传输给 ssh (ssh 连接远程服务器不是要输入密码吗? sshpass就是模拟 我们手敲密码给 ssh 的) -> 2
 - ssh 到远端服务器,执行 "echo ~root && sleep 0" 1

根据我前面整理的原理,我标记1、2、3、4数字的这几步,我按这个顺序从下往上开始—— 排除。

其实,本来应该从最顶上的4开始排查的,但4需要阅读源码,所以我从简单的1开始排查。

2. 排查 ssh

首先,我测试标记为 1 的步骤,手敲这个密码,ssh 是否认正常。结果是连接正常。

```
[root@192-168-199-121 ~]# ssh root@192.168.199.99
root@192.168.199.99's password:手敲 1#fander
Last login: Sun Nov 13 23:13:41 2022 from 192.168.199.121
[root@192-168-199-99 ~]#
```

3. 排查 sshpass

然后,测试标记为 2 的步骤,测试 sshpass 传输密码是否正常的。结果也是连接正常。

```
1. [root@192-168-199-99 ~]# sshpass -p "1#fander" ssh -C -o Cnotallo
2. -o StrictHostKeyChecking=no -o 'User="root"' -o Cnotallow=10 \
    -o 'Cnotallow="/root/.ansible/cp/e2f9f7759b"' 192.168.199.99 \
    '/bin/sh -c '"'"'echo ~root && sleep 0'"'"''
5. /root
```

这里,我调整了原命令,因为原命令用的是 sshpass -d10,这个文件描述符文件我不知道如何制造,所以我改为用 sshpass -p 来测试。我去掉了 -vvv ,因为如果一切正常,我不需要刷屏的 debug 日志。

4. 使用 paramiko 连接方式辅助排查

标记为 3 的步骤,我不知道如何测试,但我知道 ansible 除了 ssh 连接,还有一种叫 param iko 的连接方式,他是旧版 ansible 的默认连接方式,他比较低效,他不使用 ControlPersis t ,也就不会建立 Control socket ,而之前我们连接报错时,日志的关键信息就是 Control socket "/root/.ansible/cp/e2f9f7759b" does not exist

当然了,他也不会使用 sshpass 和不会把密码写入那个数字为 10 的文件描述符。所以,如果我能在 paramiko 的连接方式能复现报错,那么就和标记为 3 的步骤无关。

```
    [root@192-168-199-121 inventory]# cat /etc/ansible/ansible.cfg
    [defaults]
    host_key_checking = False
    callback_whitelist = timer
    transport = paramiko
```

结果是,我使用 paramiko 的连接方式,也能复现连接报错。

他这个 python 抛出异常非常好,终于让我知道为什么密码会错误了,原来传进去的密码不是"1#fander",而是"1"。也就是问题肯定不在标记为 3 的步骤,而是标记为 4 的步骤。

根据报错,我在 auth_handler.py 文件里打了两个 print。

```
m.add_string(self.auth_method)

if self.auth_method == "password":

m.add_boolean(False)

print(self.password)

password = b(self.password)

print(password)

m.add_string(password)

m.add_string(password)

print(password)

m.add_string(password)

print(password)

m.add_string(password)
```

我们再看看输出。

那么,标记为 1、2、3 的步骤我们都排除了,问题出在标记为 4 的步骤,也就是现在的问题是:

为什么 ansible 传入给 sshpass 和 ssh 的密码不正确,应该传入"1#fander",但最终 传入"1"?

5. 排查 ansible -k

我们再来做个测试,不使用 ansible hosts 文件传递密码,使用 -k 参数手敲密码。发现一切正常。

6. 水落石出,是 ansible 的 hosts 设置问题

那问题完全能定位出来了,这个疑似 bug,不是 ssh 也不是 sshpass 的问题,而是 ansible 的问题。并且,我们能确定,这个和 ansible 读取解析 /etc/ansible/hosts 文件有关。

经过我查阅资料,ansible 读取解析 /etc/ansible/hosts 相关的代码在这个路径下, ini.py 文件。

```
    cd /usr/local/python3/lib/python3.11/site-packages/
    cd ansible_core-2.14.0-py3.11.egg/ansible/plugins/inventory
    less ini.py
```

通过阅读注释,发现这个不是 bug,而是官方知道的问题,所以属于一个坑。

```
EXAMPLES = ''' fmt: ini
33
34
     # Example 1
35
      [web]
     hostl
36
      host2 ansible port=222 # defined inline, interpreted as an integer
37
38
39
40
      http_port=8080 # all members of 'web' will inherit these
41
      myvar=23 # defined in a :vars section, interpreted as a string
42
      [web:children] # child groups will automatically add their hosts to parent group
43
44
      apache
45
      nginx
46
47
      [apache]
48
      tomcatl
      tomcat2 myvar=34 # host specific vars override group vars
49
      tomcat3 mysecret="'03#pa33wOrd'" # proper quoting to prevent value change
50
51
52
      [nginx]
53
      jenkinsl
54
55
      [nginx:vars]
      has java = True # vars in child groups override same in parent
56
57
58
      [all:vars]
```

最后

我经常阅读源码都是通过阅读注释就解决的,这很有趣,适合我这种萌新 coder。现在是 20 22年11月13日的23:56分,由于能力和时间的关系,我就写到这里了。大家应该看懂了 解决办法,请大家避免这个坑,这个坑不单止针对密码,在 hosts 文件的所有变量设置都应 该这么做(上图横线)。有兴趣深入研究的同学可以继续看看源代码。

来源:本文转自公众号芬达的数据库学习笔记, ② 点击查看原文。



上一篇: 运维必知必会的 Kubectl 命令总... 下一篇: 11个步骤完美排查服务器是否被...



提问和评论都可以, 用心的回复会被更多人看到

评论

相关文章

Ansible Playbook: 批量管理的艺术

本次课程深入探讨了Ansible中Playbook的概念和应用。Playbook作为Ansible的核心功能之一, ...

Ansible Playbook Ad-hoc 批量管理 远程主机管理

记录一次接口无法使用的问题排查

所有接口无法使用, postman测试nginx 返回504服务器cpu, 内存正常原因: php-fpm进程数太低...

php

Python 判断for循环最后一次

1. 简介在 Python 编程中,判断 for 循环是否到达最后一次迭代是一个常见需求。通过不同的方...

迭代器 嵌套

PTA|指针进阶(一)字符定位(最后一次找到的字符)

本题要求定义一个函数,在字符串中查找字符,并定位在最后一次找到的位置。函数接口定义: ...

字符串 查找字符 调用函数

一次诡异的磁盘空间占用问题排查

大半夜接到线上一服务器磁盘占用率超过90%的短信,需要立即处理。一般这种情况都是线上异...

运维 linux shell 重启 服务器

TiDB 集群一次诡异的写入慢问题排查经历

作者: mydb 靳献旗 | 汽车之家 DBA 1.背景 最近处理了一个 TiDB 集群写入慢的问题,虽然问...

服务器 系统日志 数据

一次"诡异"的mysql死锁问题

https://blog.51cto.com/u_15576159/5868439

下面是mysql error.log中有关死锁部分的信息: 2020-02-11T15:09:21.788043+08:00 2341233 [...

mysql deadlock transaction lock rollback

一次诡异的ssh远程时断时续问题

一大早,公司砸开了锅,纷纷说公司所有的服务器都出现登录出现延迟。我连忙远程了一下,用...

ssh 远程连接 linux 服务器 时

一次诡异的线上数据库的死锁问题排查过程

前几天,线上发生了一次数据库死锁问题,这一问题前前后后排查了比较久的时间,这个过程中... _{死锁}

MySQL案例: 一次诡异的Aborted connection错误排查

简介前段时间,研究怎么去提升数据库安全,例如禁止执行不带条件的update操作,于是就想到...
Java

一次诡异的磁盘IO使用率高排查

问题描述前段时间,一个Mysql数据库服务器磁盘IO写操作非常高,经过排查,发现是mysql线...
Java

一次"诡异的"QPS翻倍

在做MySQL升级从5.0.49升级到5.1.68之后,发现QPS翻倍了;在review了所有域的应用之后,...

mysql query cache

一次诡异的502错误

环境描述: nginx前段, proxy_pass到后端程序故障描述: 同样的请求,参数不同,一个可以正...

502 nginx upstream

一次内存泄露问题的排查

系统对外提供的Solr查询接口,在来自外部调用的压力加大之后,就会出现solr查询报Read Ti...

Java 职场 随笔 休闲 Java语言

生产环境一次诡异的NPE问题,反转了4次

前言公司为了保证系统的稳定性,加了很多监控,比如:接口响应时间、cpu使用率、内存使用...

java bug rabbitmq zookeeper 接口

记一次CPU飙升的问题排查

1.背景通过公司监控工具监控,发现公司某个应用cpu利用率达到120%,也就是说这个应用自...

linux 运维 运维开发 故障排查 java

一次calico起不来的问题排查

1. 故障描述在原来的集群上纳管一台主机10.248.60.113,由于10.248.60.113主机kubelet的配...

linux docker 运维 devops 容器

记录一次Region is Unavailable问题的排查

作者: tidb菜鸟一只 现象在我一次对数据库sysbench压测的时候,对数据库准备压测的数据时...

tidb tikv https docs grafana

一次缓存性能问题排查

一次缓存性能问题排查实战分享

性能 软件测试

记一次单机系统的性能优化:最后竟是 TCP 的锅

前言这篇文章的主题是记录一次 Python 程序的性能优化, 在优化的过程中遇到的问题, 以及如...

数据库 linux java mysql python

一次诡异的调优

最近碰到的一个Java应用,费了半天劲还是没定位到是哪儿的问。发上来给大家看看,给点建议...

用例 数据库 执行时间

jquery 拖拽悬浮

现在的网站...很多模块化的东西..都希望实现可视化的操作..主要原因是为了提高用户体验.增强...

jquery 拖拽悬浮 jquery function html stylesheet

如何查看uboot支持的nfs版本

-Boot环境变量的解释说明环 境 变 量解 释 说 明bootdelay定义执行自动启动的等候秒数baudrat...

如何查看uboot支持的nfs版本 服务端 网络文件系统 地址解析协议

二级python操作题怎么运行

一、交换两个变量的值。 在Python里面交换变量的值,不需要像C语言一下用异或操作,并且...

二级python操作题怎么运行 python 开发语言 python编程开发 python入门

java lib conf启动

由于Java是支持单继承的(接口除外),所以我们普遍启动线程的方式都是实现Runnable接口并重...

java lib conf启动 java启动线程用什么方法 System ide 公众号

linux怎么更新nginx

前言:显卡硬件是英维达p2000的。安装的uos系统,但是显卡驱动自带的是一个很难用的开源...

linux怎么更新nginx nvidia闭源 UOS Linux 显卡驱动 重启

51CTO 博客 技术成就考想

友情链接 关于我们

鸿蒙开发者社区 51CTO学堂 官方博客 全部文章 热门标签 班级博客

Copyright © 2005-2025 **51CTO.COM** 版权所有京ICP证060544号 **51CTO 软考资讯** 了解我们 网站地图 意见反馈