Een Small Alpine Linux Docker-afbeelding op IOx configureren

Inhoud

Inleiding

Dit document beschrijft het installatieproces om op Docker gebaseerde toepassingen te maken, in te voeren en te beheren op Cisco IOx-compatibele apparaten.

Voorwaarden

Vereisten

Er zijn geen specifieke vereisten van toepassing op dit document.

Gebruikte componenten

De informatie in dit document is gebaseerd op de volgende software- en hardware-versies:

- IOx-compatibel apparaat dat voor IOx is geconfigureerd:
  IP-adres ingesteldGuest Operating System (GOS) en Cisco Application Framework (CAF)Network-adresomzetting (NAT) ingesteld voor toegang tot CAF (poort 8443)NAT geconfigureerd voor toegang tot GOS-shelf (poort 222)
- Linux-host (een minimale CentOS 7-installatie wordt voor dit artikel gebruikt)
- IOx-clientinstallatiebestanden die kunnen worden gedownload van:
  https://software.cisco.com/download/release.html?mdfid=286306005&softwareid=286306762

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u de potentiële impact van elke opdracht begrijpen.

Achtergrondinformatie

IOx kan verschillende soorten pakketten onderbrengen, hoofdzakelijk Java, Python, LXC, Virtual

Machine (VM) enz., en kan ook Docker-containers gebruiken. Cisco biedt een basisbeeld en een volledige DOCKER-hub aan: https://devhub.cisco.com/artifactory/webapp/#/artifacts/browse/tree/General/iox-docker die kan worden gebruikt om Docker-containers te bouwen.

Hier is een stap-voor-stap gids over hoe je een eenvoudige Docker-container kunt bouwen met het gebruik van Alpine Linux. Alpine Linux is een kleine Linux-afbeelding (ongeveer 5MB), die vaak wordt gebruikt als basis voor Docker-containers. In dit artikel, start u met een geconfigureerd IOx-apparaat, een lege CentOS 7 Linux-machine en u bouwt een kleine Python-webserver, verpakt het in een Docker-container en implementeert dat op een IOx-apparaat.

Configureren

1. Installeer en bereid IOx-client op de Linux-host.

IOx-client is het gereedschap dat toepassingen kan verpakken en met het IOx-Geschikt apparaat kan communiceren om IOx-toepassingen te beheren.

Nadat u het ioxclient-installatiepakket hebt gedownload, kunt u het als volgt installeren:

```
[jedepuyd@db ~]$ ll ioxclient_1.3.0.0_linux_amd64.tar.gz
-rw-r--r--. 1 jedepuyd jedepuyd 4668259 Jun 22 09:19 ioxclient_1.3.0.0_linux_amd64.tar.gz

[jedepuyd@db ~]$ tar -xvzf ioxclient_1.3.0.0_linux_amd64.tar.gz
ioxclient_1.3.0.0_linux_amd64/ioxclient
ioxclient_1.3.0.0_linux_amd64/README.md

[jedepuyd@db ~]$ ./ioxclient_1.3.0.0_linux_amd64/ioxclient --version
Config file not found : /home/jedepuyd/.ioxclientcfg.yaml
Creating one time configuration..
Your / your organization's name : Cisco
Your / your organization's URL : www.cisco.com
Your IOx platform's IP address[127.0.0.1] : 10.48.43.197
Your IOx platform's port number[8443] :
Authorized user name[root] : admin
Password for admin :
Local repository path on IOx platform[/software/downloads]:
URL Scheme (http/https) [https]:
API Prefix[/iox/api/v2/hosting/]:
Your IOx platform's SSH Port[2222]:
Activating Profile default
Saving current configuration
ioxclient version 1.3.0.0

[jedepuyd@db ~]$ ./ioxclient_1.3.0.0_linux_amd64/ioxclient --version
ioxclient version 1.3.0.0
```

Zoals u kunt zien, kan bij de eerste lancering van IOx client een profiel worden gegenereerd voor het IOx-apparaat dat u kunt beheren met IOx-client. Wilt u dit later doen of als u de instellingen wilt toevoegen of wijzigen, dan kunt u deze opdracht later uitvoeren: **profielen van ioxclients maken**

2. Installeer en bereid Docker op de Linux-host.

Docker wordt gebruikt om een container te bouwen en de uitvoering van onze steekproeftoepassing te testen.

De installatiestappen om Docker te installeren zijn sterk afhankelijk van de Linux-OS waarop u het installeert. Voor dit artikel kunt u CentOS 7 gebruiken. Voor installatie-instructies voor verschillende distributies raadpleegt u: https://docs.docker.com/engine/installation/.

Installatievereisten:

```
[jedepuyd@db ~]$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
...
Complete!
```

Docker-rapport toevoegen:

```
[jedepuyd@db ~]$ sudo yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo
Loaded plugins: fastestmirror
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to
/etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

Installeer Docker (accepteer de GPG-sleutelverificatie tijdens installatie):

```
[jedepuyd@db ~]$ sudo yum install docker-ce
...
Complete!
```

Docker starten

```
[jedepuyd@db ~]$ sudo systemctl start docker

[jedepuyd@db iox_docker_pythonweb]$ vi Dockerfile
[jedepuyd@db iox_docker_pythonweb]$ cat Dockerfile
FROM alpine:3.3

RUN apk add --no-cache python
COPY webserver.py /webserver.py
```

U kunt Docker als een normale gebruiker benaderen/uitvoeren door deze gebruiker aan de Docker-groep toe te voegen en het lidmaatschap van de groep op te frissen:

```
[jedepuyd@db ~]$ sudo usermod -a -G docker jedepuyd
[jedepuyd@db ~]$ newgrp docker
```

Inloggen op Docker Hub:

Docker Hub bevat het alpiene basebeeld dat u kunt gebruiken. Voor het geval u nog geen Docker-ID hebt, dient u zich te registreren op: https://hub.docker.com/.

```
[jedepuyd@db ~]$ docker login
Log in with your Docker ID to push and pull images from Docker Hub. If you do not have a Docker
ID, head over to https://hub.docker.com to create one.
Username: jensdepuydt
Password:
Login Succeeded
```

3. Maak de Python-webserver.

Nu de voorbereiding is voltooid, kunt u beginnen met het bouwen van de eigenlijke toepassing die kan worden uitgevoerd op het IOx-enabled-apparaat.

```
[jedepuyd@db ~]$ mkdir iox_docker_pythonweb
[jedepuyd@db ~]$ cd iox_docker_pythonweb/
[jedepuyd@db iox_docker_pythonweb]$ vi webserver.py
[jedepuyd@db iox_docker_pythonweb]$ cat webserver.py
#!/usr/bin/env python
from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer
import SocketServer
import os

class S(BaseHTTPRequestHandler):
    def _set_headers(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def do_GET(self):
        self._set_headers()
        self.wfile.write("<html><body><h1>IOX python webserver</h1></body></html>")

def run(server_class=HTTPServer, handler_class=S, port=80):
    server_address = ('', port)
    httpd = server_class(server_address, handler_class)
    print 'Starting webserver...'
    log_file_dir = os.getenv("CAF_APP_LOG_DIR", "/tmp")
    log_file_path = os.path.join(log_file_dir, "webserver.log")
    logf = open(log_file_path, 'w')
    logf.write('Starting webserver....\n')
    logf.close()

    httpd.serve_forever()

if __name__ == "__main__":
    from sys import argv

    if len(argv) == 2:
        run(port=int(argv[1]))
    else:
        run()
```

Deze code is een zeer minimale Python-webserver, die je maakt in webserver.py. De webserver retourneert de IOx python webserver zodra er een GET is aangevraagd. De poort waarop de webserver start kan poort 80 zijn of het eerste argument dat aan webserver.py gegeven wordt.

Deze code bevat in de uitvoerfunctie ook een schrijfbestand naar een logbestand. Het logbestand is beschikbaar voor overleg met de IOx-client of de plaatselijke beheerder.

4. Maak de Dockerfile en Docker-container.

Nu u de applicatie (webserver.py) hebt die in uw container moet lopen, is het tijd om de Docker-container te bouwen. Een container wordt gedefinieerd in een Dockerfile:

```
[jedepuyd@db iox_docker_pythonweb]$ vi Dockerfile
[jedepuyd@db iox_docker_pythonweb]$ cat Dockerfile
FROM alpine:3.3

RUN apk add --no-cache python
```

```
COPY webserver.py /webserver.py
```

Zoals u kunt zien, wordt het Dockerfile ook eenvoudig gehouden. Als u begint met het beeld op de basis van de Alpen, installeert u Python en kopieert u uw webserver.py naar de achterzijde van de container.

Zodra u uw Dockerfile klaar hebt, kunt u de Docker-container bouwen:

```
jedepuyd@db iox_docker_pythonweb]$ docker build -t ioxpythonweb:1.0 .
Sending build context to Docker daemon 3.584 kB
Step 1/3 : FROM alpine:3.3
3.3: Pulling from library/alpine
10462c29356c: Pull complete
Digest: sha256:9825fd1a7e8d5feb52a2f7b40c9c4653d477b797f9ddc05b9c2bc043016d4819
Status: Downloaded newer image for alpine:3.3
 ---> 461b3f7c318a
Step 2/3 : RUN apk add --no-cache python
 ---> Running in b057a8183250
fetch http://dl-cdn.alpinelinux.org/alpine/v3.3/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.3/community/x86_64/APKINDEX.tar.gz
(1/10) Installing libbz2 (1.0.6-r4)
(2/10) Installing expat (2.1.1-r1)
(3/10) Installing libffi (3.2.1-r2)
(4/10) Installing gdbm (1.11-r1)
(5/10) Installing ncurses-terminfo-base (6.0-r6)
(6/10) Installing ncurses-terminfo (6.0-r6)
(7/10) Installing ncurses-libs (6.0-r6)
(8/10) Installing readline (6.3.008-r4)
(9/10) Installing sqlite-libs (3.9.2-r0)
(10/10) Installing python (2.7.12-r0)
Executing busybox-1.24.2-r1.trigger
OK: 51 MiB in 21 packages
 ---> 81e98c806ee9
Removing intermediate container b057a8183250
Step 3/3 : COPY webserver.py /webserver.py
 ---> c9b7474b12b2
Removing intermediate container 4705922100e6
Successfully built c9b7474b12b2

[jedepuyd@db iox_docker_pythonweb]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ioxpythonweb        1.0                 c9b7474b12b2        11 seconds ago      43.4 MB
alpine              3.3                 461b3f7c318a        2 days ago          4.81 MB
```

De opdracht Geavanceerd maken van docker downloads van de basisafbeelding en installeert Python- en afhankelijkheden, zoals u in het DOCKbestandsbestand hebt gevraagd. De laatste opdracht is voor verificatie.

5. Test de aangelegde Docker-container.

Deze stap is optioneel, maar het is goed om te controleren of uw net gebouwde Docker container klaar is om te werken zoals verwacht.

```
[jedepuyd@db iox_docker_pythonweb]$ docker run -ti ioxpythonweb:1.0
/ # python /webserver.py 9000 &
/ # Starting webserver...

/ # netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
```

```
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      7/python
/ # exit
```

Zoals je kunt zien in de output van netstat, nadat je de webserver.py start, luistert het op poort 9000.

6. Maak het IOx-pakket met de Docker-container.

Nu je de functionaliteit van je webserver in de container hebt geverifieerd, is het tijd om het IOx-pakket voor te bereiden en te bouwen voor implementatie. Aangezien Dockerfile instructies geeft om een Docker-container te bouwen, bevat het pakket.yaml instructies voor IOx-client om uw IOx-pakket te bouwen.

```
jedepuyd@db iox_docker_pythonweb]$ vi package.yaml
[jedepuyd@db iox_docker_pythonweb]$ cat package.yaml
descriptor-schema-version: "2.2"

info:
  name: "iox_docker_pythonweb"
  description: "simple docker python webserver on port 9000"
  version: "1.0"
  author-link: "http://www.cisco.com"
  author-name: "Jens Depuydt"

app:
  cpuarch: "x86_64"
  type: docker
  resources:
    profile: c1.small
    network:
      -
        interface-name: eth0
        ports:
            tcp: [9000]

  startup:
    rootfs: rootfs.tar
    target: ["python","/webserver.py","9000"]
```

Zie voor meer informatie over de inhoud van de verpakking.yaml: https://developer.cisco.com/media/iox-dev-guide-3-10-16/concepts/package_descriptor/.

Nadat u het Package.yaml hebt gemaakt, kunt u het IOx-pakket beginnen te bouwen.

Eerste stap is het exporteren van de wortel FS van de Docker-afbeelding:

```
[jedepuyd@db iox_docker_pythonweb]$ docker save -o rootfs.tar ioxpythonweb:1.0
```

Daarna kunt u het pakket maken.tar:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient package .
Currently active profile:  default
Command Name: package
Checking if package descriptor file is present.
Validating descriptor file /home/jedepuyd/iox_docker_pythonweb/package.yaml with package schema
definitions
Parsing descriptor file.
Found schema version  2.2
Loading schema file for version  2.2
```

```
Validating package descriptor file..
File /home/jedepuyd/iox_docker_pythonweb/package.yaml is valid under schema version 2.2
Created Staging directory at :  /tmp/700740789
Copying contents to staging directory
Checking for application runtime type
Couldn't detect application runtime type
Creating an inner envelope for application artifacts
Generated  /tmp/700740789/artifacts.tar.gz
Calculating SHA1 checksum for package contents..
Parsing Package Metadata file :  /tmp/700740789/.package.metadata
Wrote package metadata file :  /tmp/700740789/.package.metadata
Root Directory :  /tmp/700740789
Output file:  /tmp/335805072
Path:  .package.metadata
SHA1 : 55614e72481a64726914b89801a3276a855c728a
Path:  artifacts.tar.gz
SHA1 : 816c7bbfd8ae76af451642e652bad5cf9592370c
Path:  package.yaml
SHA1 : ae75859909f6ea6947f599fd77a3f8f04fda0709
Generated package manifest at  package.mf
Generating IOx Package..
Package generated at /home/jedepuyd/iox_docker_pythonweb/package.tar
```

Het resultaat van de bouw is een IOx pakket (Package.tar), dat de container van Docker bevat, klaar om op IOx te worden opgesteld.

> Opmerking: IOxclient kan de docker opslagopdracht ook in één stap uitvoeren. Op CentOS resulteert dit in de export naar de standaard rootfs.img in plaats van rootfs.tar, wat problemen oplevert in een later stadium van het proces. De enige stap die moet worden gemaakt, kan worden gedaan met behulp van: IOx client docker Packet IOxpythonweb:1.0.

8. De verpakking op het IOx-apparaat implementeren, activeren en starten.

Laatste stappen zijn om het IOx-pakket op het IOx-apparaat in te voeren, het te activeren en te starten. Deze stappen kunnen worden uitgevoerd met behulp van IOx-client, Local Manager of Fog Network Director. Voor dit artikel kunt u IOx-client gebruiken.

Om het pakket op het IOx-apparaat in te zetten, gebruikt u de naam python_web:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app install
python_web package.tar
Currently active profile:  default
Command Name: application-install
Installation Successful. App is available at:
https://10.48.43.197:8443/iox/api/v2/hosting/apps/python_web
Successfully deployed
```

Voordat u de toepassing kunt activeren, moet u bepalen hoe de netwerkconfiguratie moet worden geconfigureerd. Om dit te doen, moet u een JSON-bestand maken. Wanneer u het programma activeert, kan het aan de activeringsaanvraag worden toegevoegd.

```
[jedepuyd@db iox_docker_pythonweb]$ vi activate.json
[jedepuyd@db iox_docker_pythonweb]$ cat activate.json
{
    "resources": {
        "profile": "c1.small",
        "network": [{"interface-name": "eth0", "network-name": "iox-nat0","port_map": {"mode":
"1to1"},"ports":{"tcp":9000}}]
```

```
    }
}
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app activate
python_web --payload activate.json
Currently active profile :  default
Command Name: application-activate
Payload file : activate.json. Will pass it as application/json in request body..
App python_web is Activated
```

Laatste actie hier is om de toepassing te starten die u zojuist hebt ingezet en geactiveerd:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app start
python_web
Currently active profile :  default
Command Name: application-start
App python_web is Started
```

Aangezien u uw IOx-toepassing hebt ingesteld om te luisteren op poort 9000 voor tor HTTP-verzoeken, moet u die poort nog steeds doorsturen van uw IOx-apparaat naar de container aangezien de container achter NAT ligt. Voer dit op Cisco IOS® uit om dit te doen.

```
BRU-IOT-809-1#sh iox host list det | i IPV4
    IPV4 Address of Host:       192.168.1.2
BRU-IOT-809-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
BRU-IOT-809-1(config)#ip nat inside source static tcp 192.168.1.2 9000 interface
GigabitEthernet0 9000
BRU-IOT-809-1(config)#exit
```

De eerste opdracht maakt een lijst van het interne IP-adres van GOS (verantwoordelijk voor start/stop/run de IOx-containers).

De tweede opdracht vormt een statische poort voorwaarts voor poort 9000 op de Gi0 interface van de IOS-zijde naar GOS. Indien uw apparaat via een L2 poort is aangesloten (wat waarschijnlijk de case is op IR829), moet u de Gi0 interface vervangen door het juiste VLAN dat de ip nationaal buitenstatement heeft geconfigureerd.

Verifiëren

Gebruik dit gedeelte om te bevestigen dat de configuratie correct werkt.

Om te controleren of de webserver goed werkt en reageert, kunt u met deze opdracht proberen de webserver te benaderen.

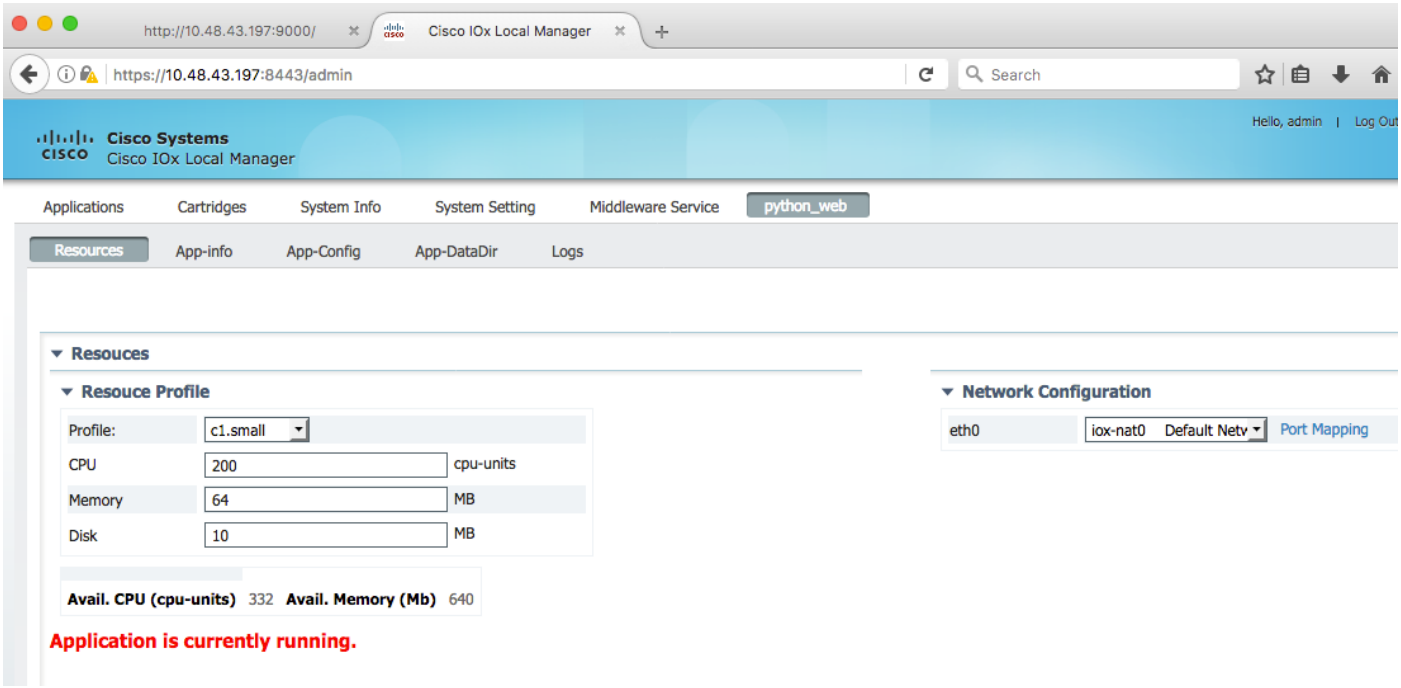
```
[jedepuyd@db iox_docker_pythonweb]$ curl http://10.48.43.197:9000/
<html><body><h1>IOX python webserver</h1></body></html>
```

Of, van een echte browser zoals getoond in de afbeelding.

U kunt de toepassingsstatus ook controleren via de IOxclient-CLI:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app status
python_web
Currently active profile :  default
Command Name: application-status
Saving current configuration
App python_web is RUNNING
```

en u kunt de toepassingsstatus ook controleren vanuit de Local Manager GUI zoals in de afbeelding weergegeven.



Zo bekijkt u het logbestand waarnaar u in webserver.py schrijft:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app logs info
python_web
Currently active profile :  default
Command Name: application-logs-info

Log file information for :  python_web
Size_bytes : 711
Download_link : /admin/download/logs?filename=python_web-watchDog.log
Timestamp : Thu Jun 22 08:21:18 2017
Filename : watchDog.log

Size_bytes : 23
Download_link : /admin/download/logs?filename=python_web-webserver.log
```

```
Timestamp : Thu Jun 22 08:21:23 2017
Filename : webserver.log

Size_bytes : 2220
Download_link : /admin/download/logs?filename=python_web-container_log_python_web.log
Timestamp : Thu Jun 22 08:21:09 2017
Filename : container_log_python_web.log
```

Problemen oplossen

Deze sectie verschaft informatie die u kunt gebruiken om problemen met uw configuratie op te lossen.

Om de toepassing en/of de container te kunnen oplossen is de makkelijkste manier om met de console van de toepassing te verbinden die loopt:

```
[jedepuyd@db iox_docker_pythonweb]$ ../ioxclient_1.3.0.0_linux_amd64/ioxclient app console
python_web
Currently active profile:  default
Command Name: application-console
Console setup is complete..
Running command: [ssh -p 2222 -i python_web.pem appconsole@10.48.43.197]
The authenticity of host '[10.48.43.197]:2222 ([10.48.43.197]:2222)' can't be established.
ECDSA key fingerprint is 1d:e4:1e:e1:99:8b:1d:d5:ca:43:69:6a:a3:20:6d:56.
Are you sure you want to continue connecting (yes/no)? yes
/ # netstat -tlpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:9000            0.0.0.0:*               LISTEN      19/python
/ # ps aux | grep python
   19 root       0:00 python /webserver.py 9000
```