

Enhancing FreeBSD Test Suite Parallelism

FreeBSD jail in jail after years of Linux docker in docker



OpenFest 2024

Igor Ostapenko

igoro@FreeBSD.org
pm@igoro.pro



> whoami

- A software engineer since the 90's
- Held various roles: Software Architect, Technology Advocate & Coach, DevOps, Tech Lead, Team Lead, Project Manager, CTO
- Worked with organizations of different sizes, from startups to international corporations like Nokia Networks
- At different levels, from assembly lang and reverse engineering up to consumer-facing stand-alone, mobile, and web apps

- This talk is about one of my FreeBSD contributions

Why do we test?

- Software development. Software development never changes.

Why do we test?

- Software development. Software development never changes.
- It's still done by us, human beings. Defects are inevitable.

Why do we test?

- Software development. Software development never changes.
- It's still done by us, human beings. Defects are inevitable.
- There are at least two options:
 - End users are going to face those defects
 - Or we hunt the bugs before releasing

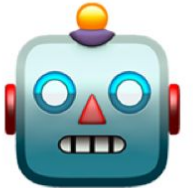
Why do we test?

- Software development. Software development never changes.
- It's still done by us, human beings. Defects are inevitable.
- There are at least two options:
 - End users are going to face those defects
 - Or we hunt the bugs before releasing
- \$\$\$ usually is a decision maker

Why do we test?

- Software development. Software development never changes.
- It's still done by us, human beings. Defects are inevitable.
- There are at least two options:
 - End users are going to face those defects
 - Or we hunt the bugs before releasing
- \$\$\$ usually is a decision maker

We can fix
you



The FreeBSD Test Suite

The FreeBSD Test Suite

- Unit test level

The FreeBSD Test Suite

- Unit test level
- Integration/system test level

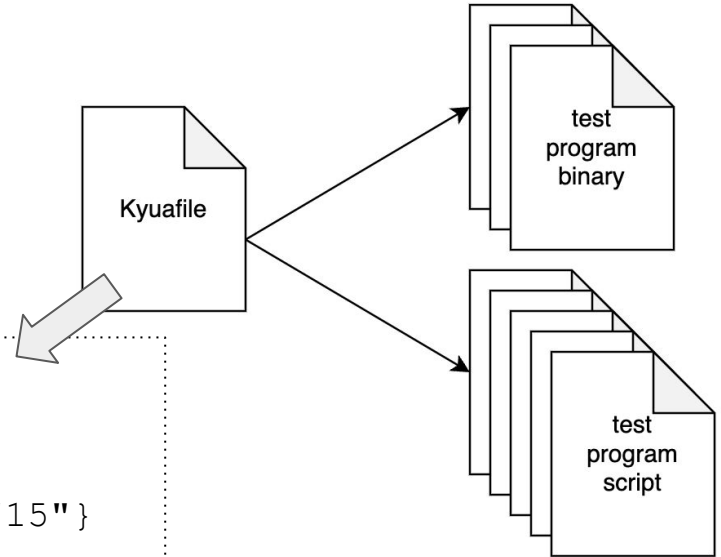
The FreeBSD Test Suite

- Unit test level
- Integration/system test level
- Organized with Kyua

The FreeBSD Test Suite

- Unit test level
- Integration/system test level
- Organized with Kyua
- A Kyuafile lists the test programs

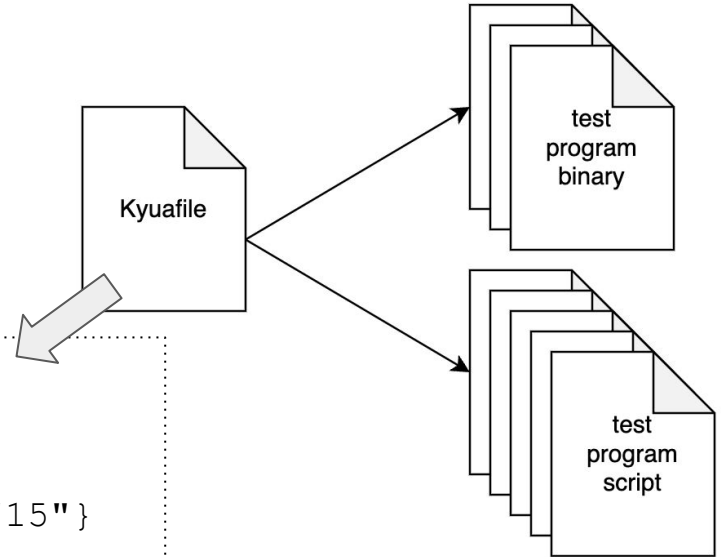
```
syntax(2)
test_suite("FreeBSD")
test_program{name="arp", is_exclusive="true"}
atf_test_program{name="ptrace_test", timeout="15"}
```



The FreeBSD Test Suite

- Unit test level
- Integration/system test level
- Organized with Kyua
- A Kyuafile lists the test programs
- `> kyua test`

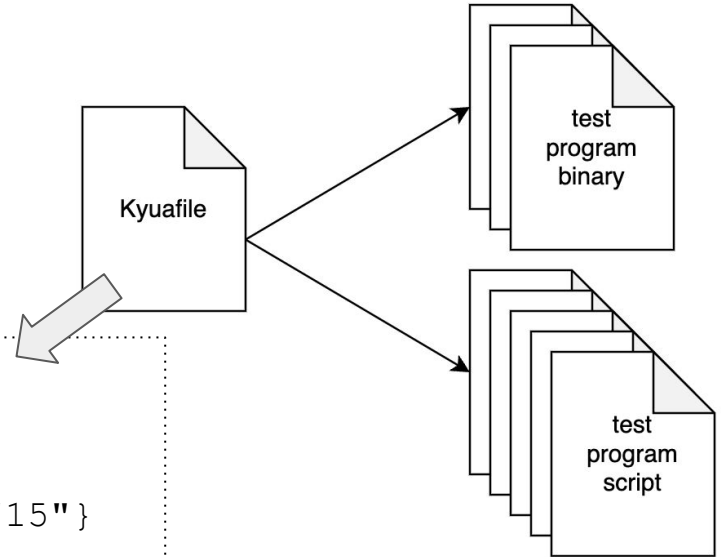
```
syntax(2)
test_suite("FreeBSD")
test_program{name="arp", is_exclusive="true"}
atf_test_program{name="ptrace_test", timeout="15"}
```



The FreeBSD Test Suite

- Unit test level
- Integration/system test level
- Organized with Kyua
- A Kyuafile lists the test programs
- `> kyua test`
- `> kyua -v parallelism=8 test`

```
syntax(2)
test_suite("FreeBSD")
test_program{name="arp", is_exclusive="true"}
atf_test_program{name="ptrace_test", timeout="15"}
```



And there are FreeBSD Jails

Containerization

- There are many visions, but we just want to isolate the things

Containerization

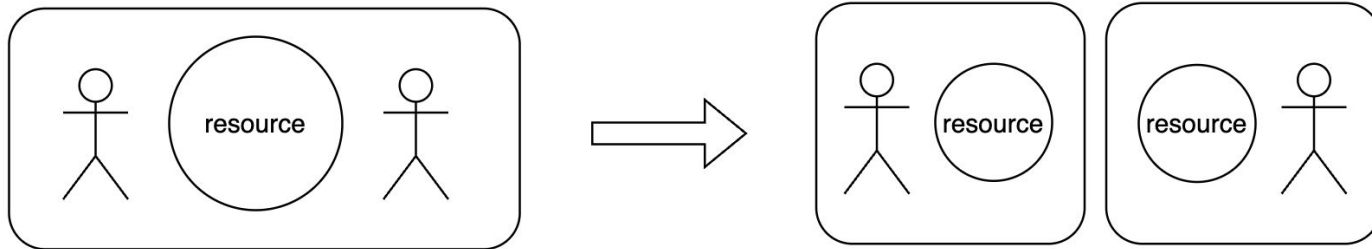
- There are many visions, but we just want to isolate the things
- Initially, it was about security

Containerization

- There are many visions, but we just want to isolate the things
- Initially, it was about security
- Ended up as the new ways of development, delivery, and operations

Containerization

- There are many visions, but we just want to isolate the things
- Initially, it was about security
- Ended up as the new ways of development, delivery, and operations



Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers to Solaris 10, later evolved into Solaris Zones

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers to Solaris 10, later evolved into Solaris Zones
- 2007 – HP released Secure Resource Partitions for HP-UX, later renamed to HP-UX Containers

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers to Solaris 10, later evolved into Solaris Zones
- 2007 – HP released Secure Resource Partitions for HP-UX, later renamed to HP-UX Containers
- 2008 – Linux 2.6.24 introduced **cgroups**

Containerization renaissance

- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers to Solaris 10, later evolved into Solaris Zones
- 2007 – HP released Secure Resource Partitions for HP-UX, later renamed to HP-UX Containers
- 2008 – Linux 2.6.24 introduced **cgroups**
- 2013 – Linux 3.8 introduced **user namespaces**

Containerization renaissance

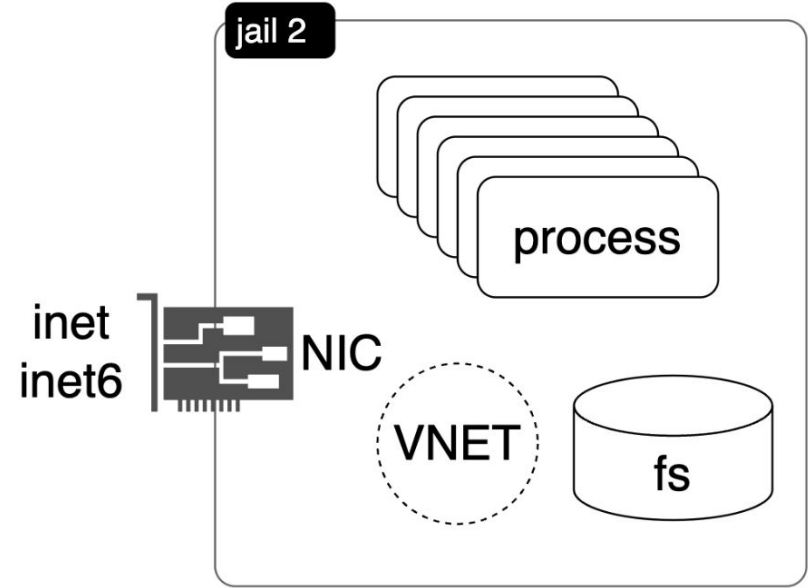
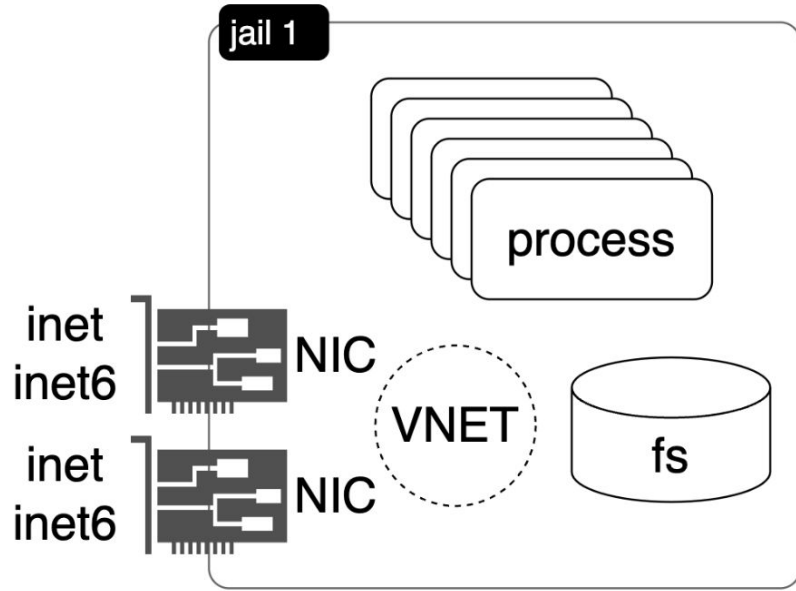
- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control, process domains
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers to Solaris 10, later evolved into Solaris Zones
- 2007 – HP released Secure Resource Partitions for HP-UX, later renamed to HP-UX Containers
- 2008 – Linux 2.6.24 introduced **cgroups**
- 2013 – Linux 3.8 introduced **user namespaces**
- 2013, a month later – the release of **Docker**

Containerization renaissance

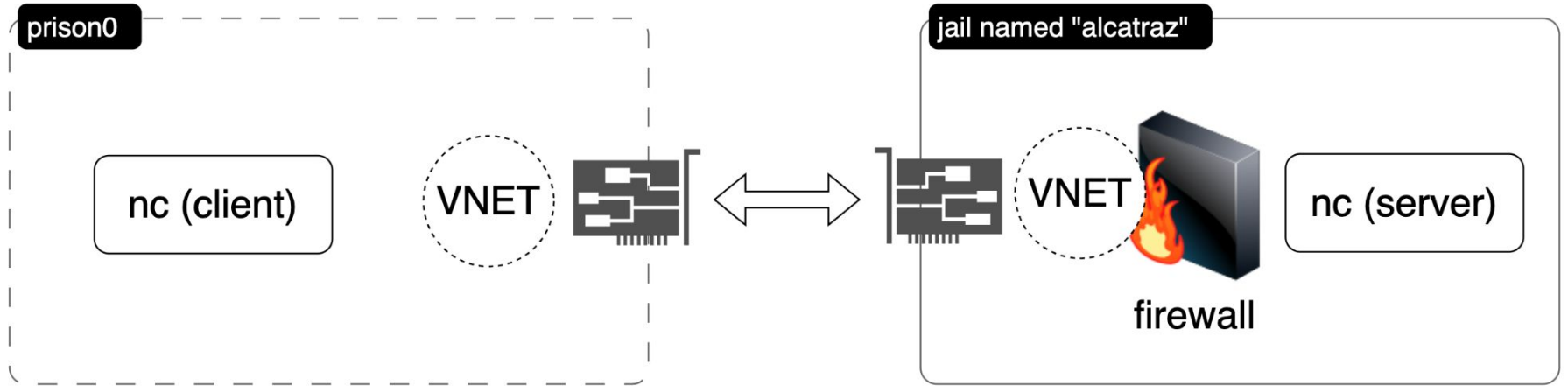
- 1979 – **chroot** was added to Unix Version 7 (considered as a real start point)
- 1980s and 1990s – mandatory access control
- 2000 – FreeBSD 4.0 introduced **jails**
- 2004 – Sun added Solaris Containers (later renamed to Solaris Zones)
- 2007 – HP released Secure Resource Containers (later renamed to HP-UX Containers)
- 2008 – Linux 2.6.24 introduced **cgroups**
- 2013 – Linux 3.8 introduced **user namespaces**
- 2013, a month later – the release of **Docker**



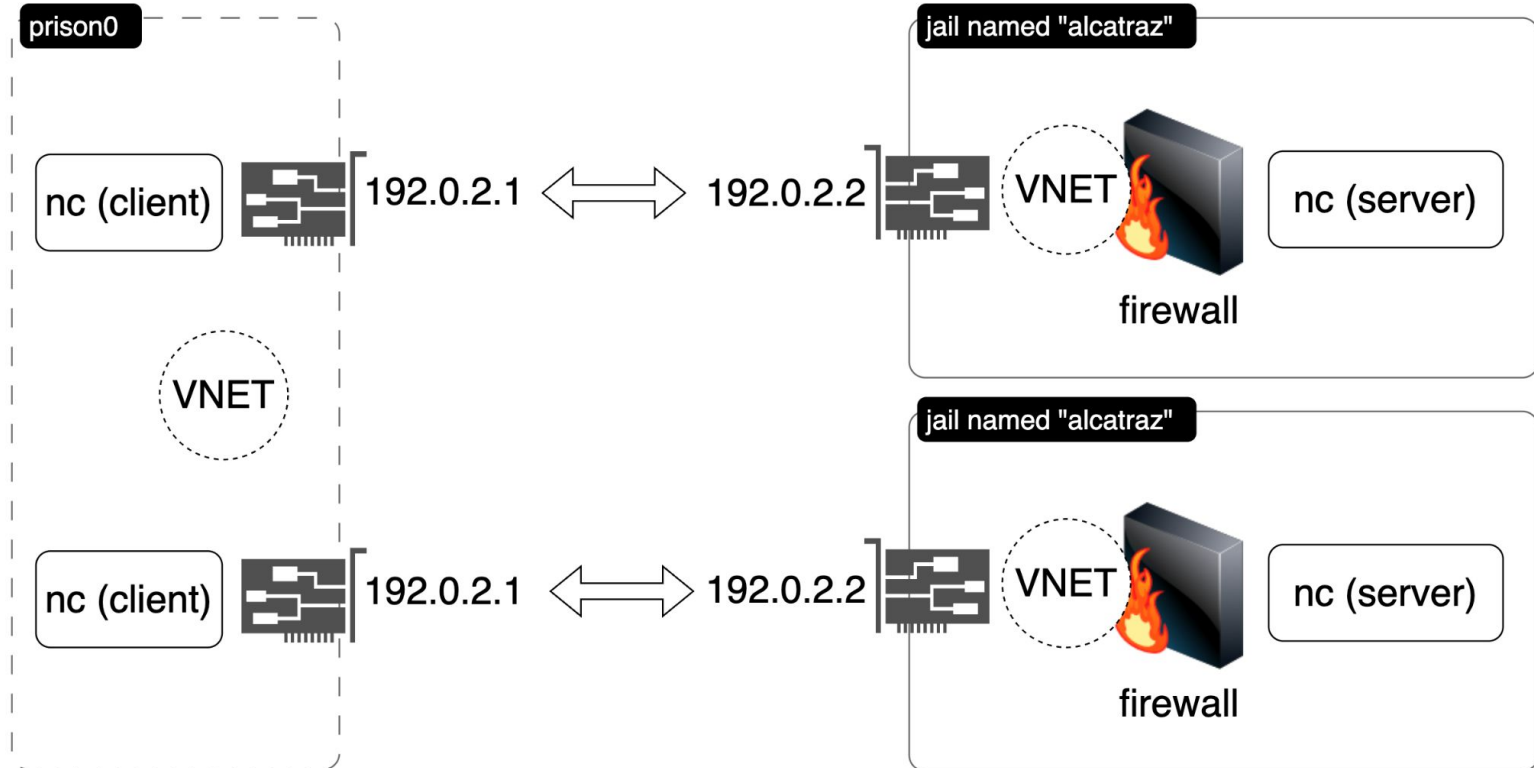
Virtual Machine as a metaphor



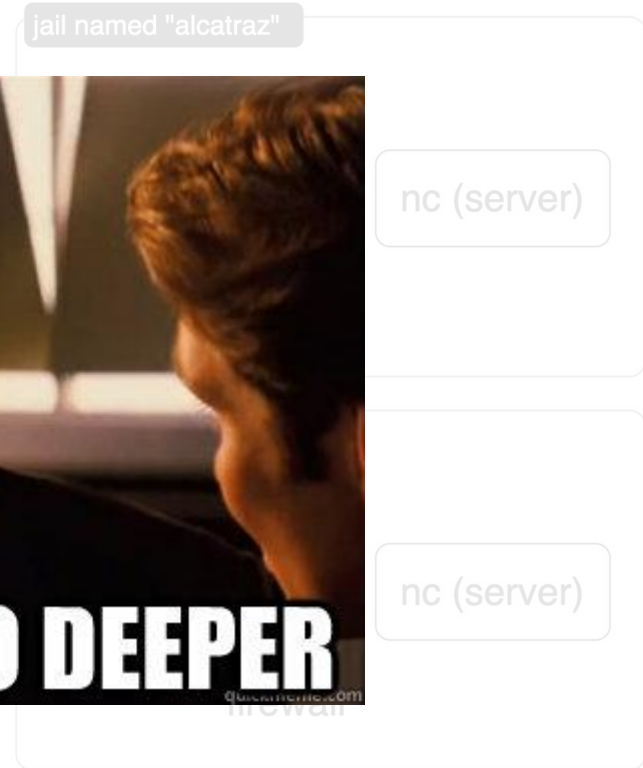
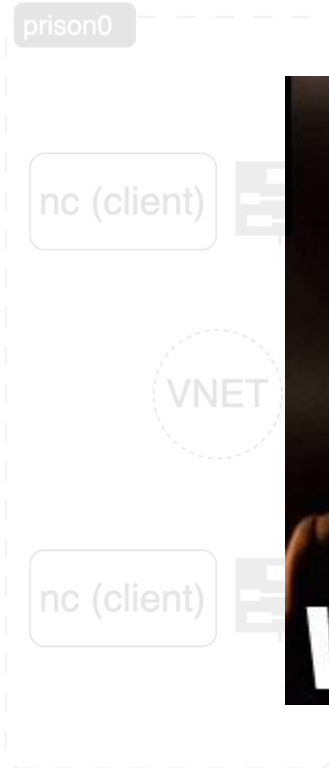
Testing of FreeBSD network modules



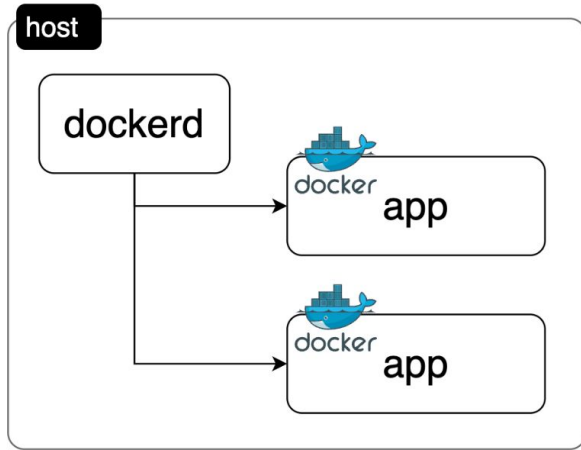
Such tests are mutually exclusive



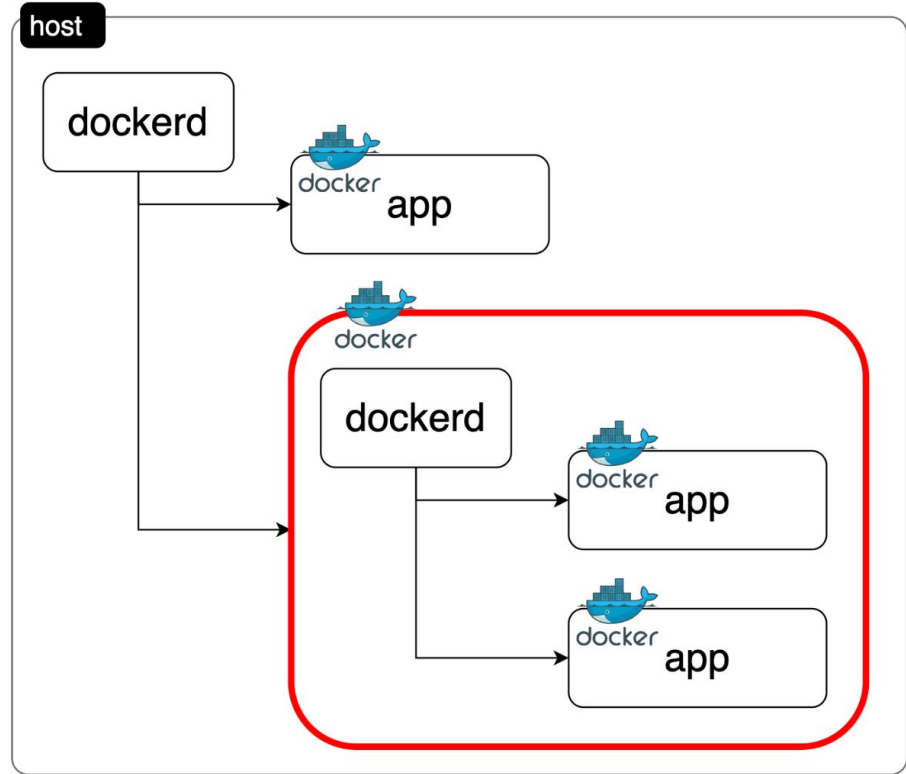
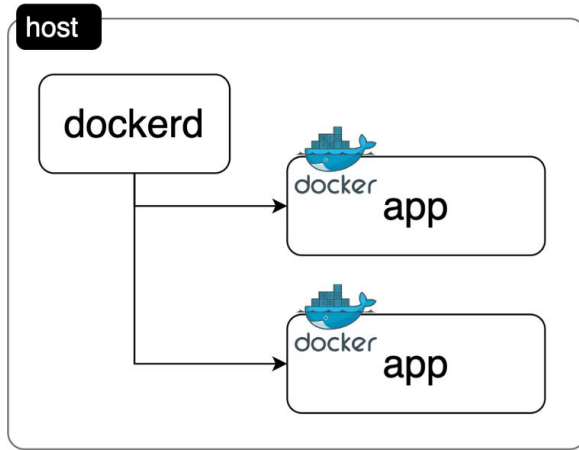
Such tests are mutually exclusive



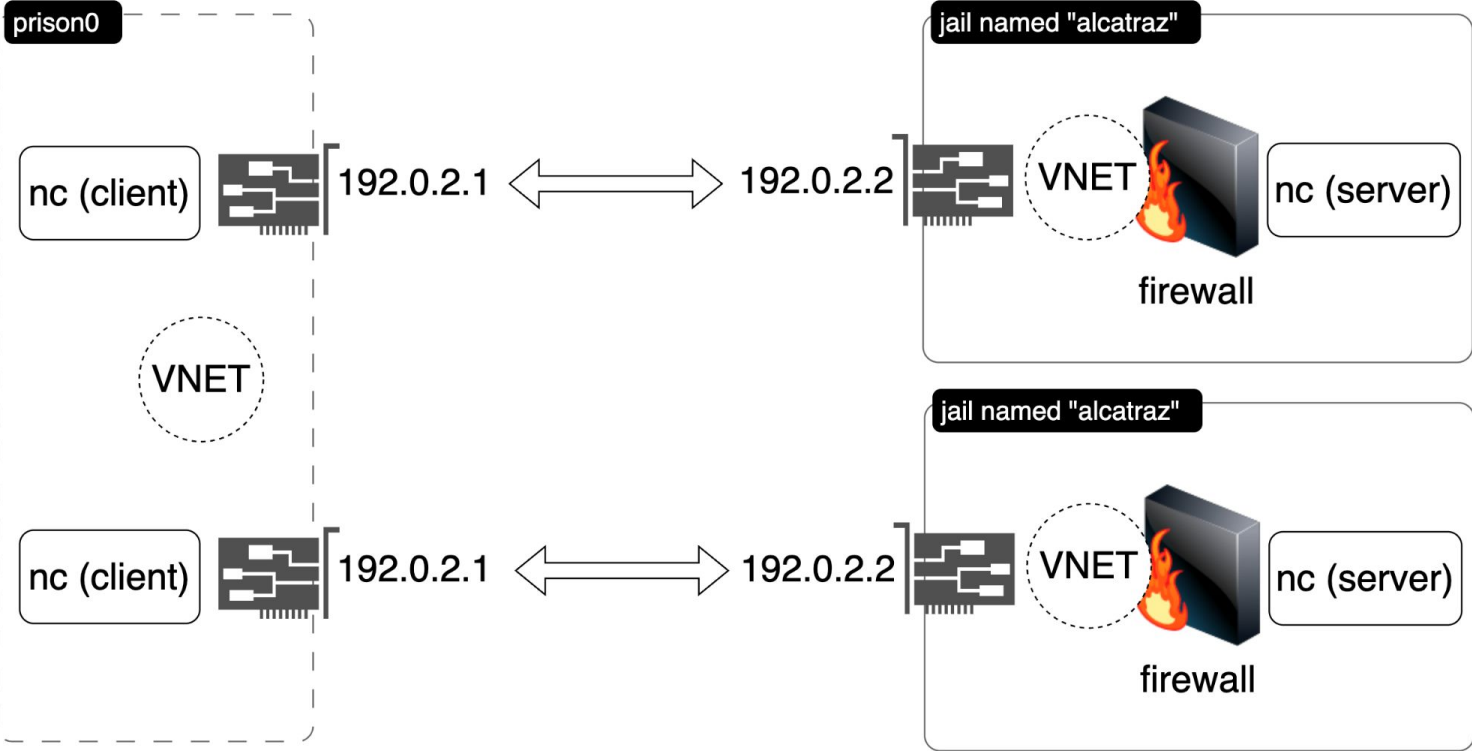
Docker in Docker (dind) – containerized CI workflows



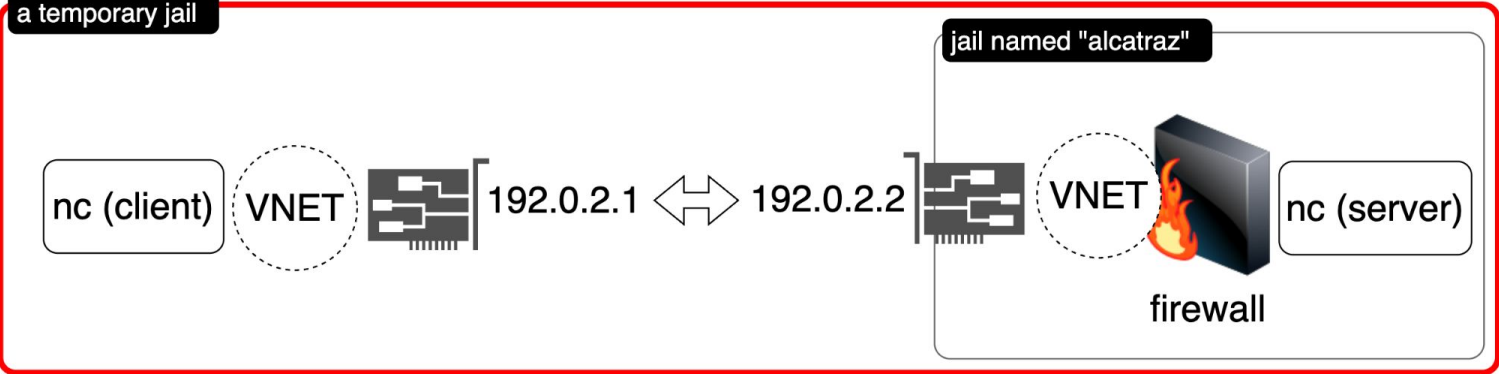
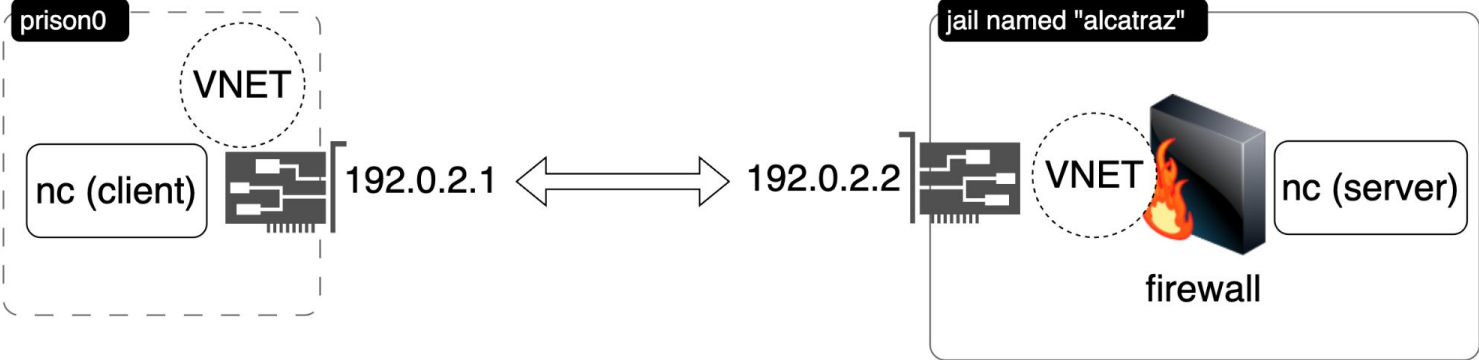
Docker in Docker (dind) – containerized CI workflows



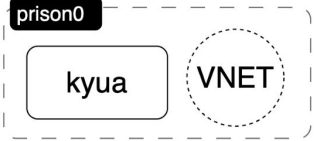
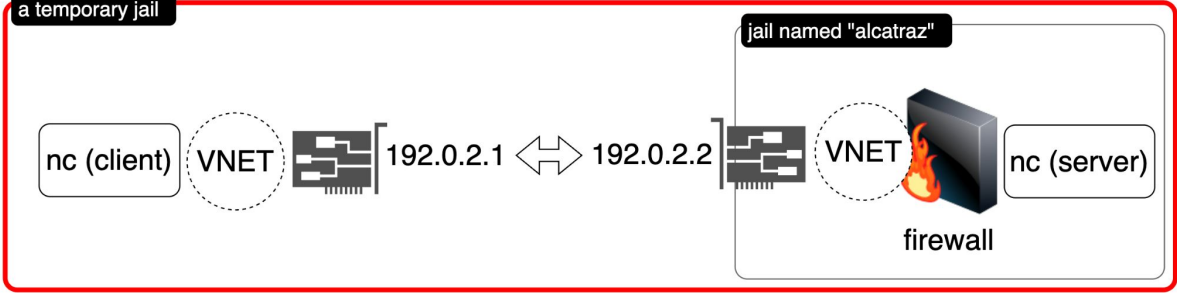
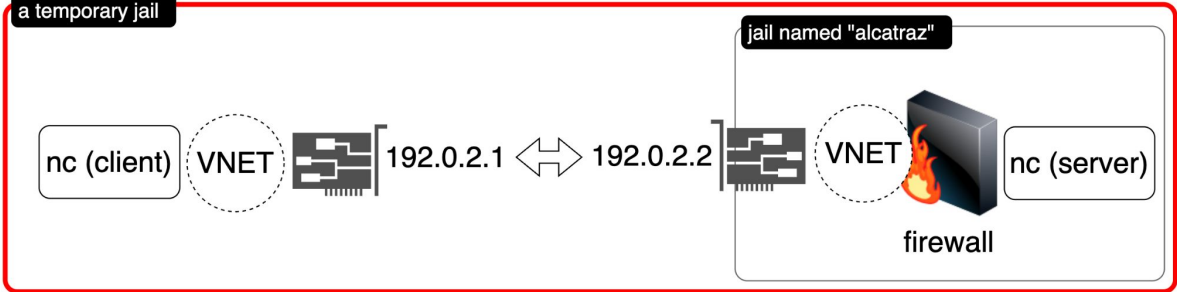
FreeBSD Jail in Jail



FreeBSD Jail in Jail



FreeBSD Jail in Jail



Kyua's execution environment concept

- A test case may ask for a specific **execenv**

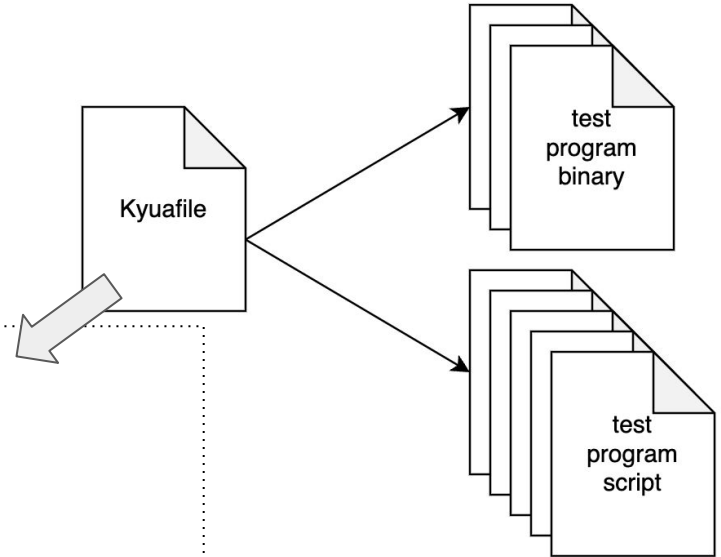
Kyua's execution environment concept

- A test case may ask for a specific **execenv**
- The old way is named **host**

Kyua's execution environment concept

- A test case may ask for a specific **execenv**
- The old way is named **host**
- The first extra execution environment is **jail**

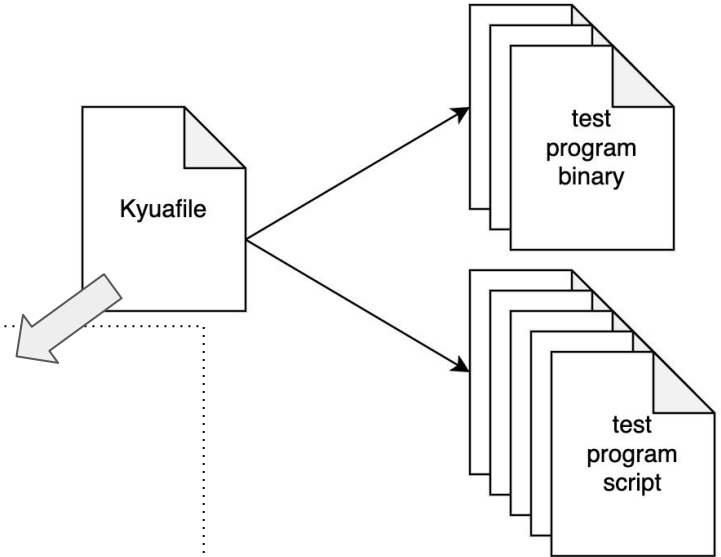
```
syntax(2)
test_suite("FreeBSD")
atf_test_program{name="t1", timeout="15"}
atf_test_program{name="t2", execenv="jail"}
atf_test_program{name="t3", execenv="jail",
execenv_jail_params="vnet allow.raw_sockets"}
```



Kyua's execution environment concept

- A test case may ask for a specific **execenv**
- The old way is named **host**
- The first extra execution environment is **jail**
- `> kyua -v parallelism=8 test`

```
syntax(2)
test_suite("FreeBSD")
atf_test_program{name="t1", timeout="15"}
atf_test_program{name="t2", execenv="jail"}
atf_test_program{name="t3", execenv="jail",
                  execenv_jail_params="vnet allow.raw_sockets"}
```



The outcome

- Switching PF (Packet Filter) firewall tests to this feature was very easy:

```
-TEST_METADATA+= is_exclusive=true
```

```
+TEST_METADATA+= execenv="jail"
```

```
+TEST_METADATA+= execenv_jail_params="vnet allow.raw_sockets"
```

- It depends, but there are reports that it takes several minutes instead of half an hour using the same environment

97%

Developer satisfaction


The takeaways

- FreeBSD Jails can be thought of as similar to Linux Docker containers
 - a good enough, though not perfect, analogy


The takeaways

- FreeBSD Jails can be thought of as similar to Linux Docker containers
 - a good enough, though not perfect, analogy
- Hierarchical jails can be leveraged similar ways as Docker-in-Docker


The takeaways

- FreeBSD Jails can be thought of as similar to Linux Docker containers
 - a good enough, though not perfect, analogy
- Hierarchical jails can be leveraged similar ways as Docker-in-Docker
- By the way, OCI  FreeBSD

The takeaways

- FreeBSD Jails can be thought of as similar to Linux Docker containers
 - a good enough, though not perfect, analogy
- Hierarchical jails can be leveraged similar ways as Docker-in-Docker
- By the way, OCI  FreeBSD
- The FreeBSD Test Suite is constantly growing and improving
 - New features on the userland may help on the kernel side

The takeaways

- FreeBSD Jails can be thought of as similar to Linux Docker containers
 - a good enough, though not perfect, analogy
- Hierarchical jails can be leveraged similar ways as Docker-in-Docker
- By the way, OCI  FreeBSD
- The FreeBSD Test Suite is constantly growing and improving
 - New features on the userland may help on the kernel side
- Probably, FreeBSD PF has the best test coverage among open source firewalls

Thank you



OpenFest 2024

Igor Ostapenko

igoro@FreeBSD.org
pm@igoro.pro

