Running and Distributing FreeBSD Containers

Yan Ka, Chiu, EuroBSDcon 2023

Overview

- Quick introduction to container / OCI
- FreeBSD quirks and features
- Xc features and demo
- Future work

Open Container Initiative (OCI)

- Open standard for OS level virtualization
- Defines a number of specifications
  - Runtime Specification
  - Image Specification
  - Distribution Specification

FreeBSD Jail / Container Ecosystem

- Lots of toolings (AppJail, Bastille, Iocage, …)
- Mostly creating stateful Jails
- Some are modern container like (pot) but not OCI compatible
- Only a few are both OCI compatible
    - FreeBSD port of podman
    - xc 👋

Why not port “podman”, “Docker”, etc…

Why invent another wheel

- At time time a FreeBSD podman port was not a thing
- REALLY want something play well and feel native to FreeBSD
- Improve on OCI image specification short-comings
- OCI is great, but I wanted more from FreeBSD containers

Why another Jail manager

- Need something for container workflow (ephemeral Jails)
- Need something to overcome the “distribution” problem
- Need something to play well with FreeBSD features (that’s why we run FreeBSD)

FreeBSD “quirks”

Device nodes

- Many features require access to devices
  - bhyve
  - nmdm
  - tuntap
- Require “something” to dynamically generate Devfs rulesets
  - But not generating harmful ones (e.g. add path nda* unhide)

Special consideration

- VNET / non-VNET Jails
- Linux Jails
- Jailed ZFS
- DTrace
- ~~Configure network interface / routing table without “ifconfig” in Jail~~
- ~~Null mount on file~~

What is xc

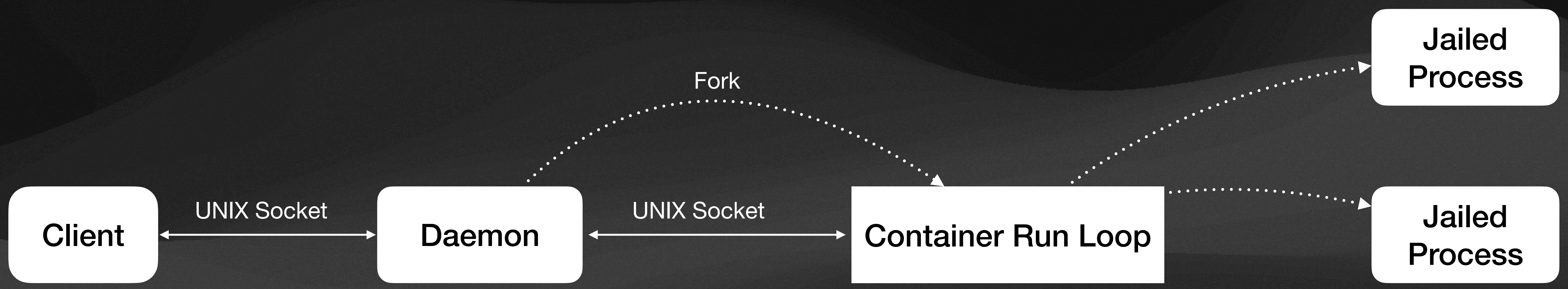
- Container Runtime for FreeBSD
- Optimized for FreeBSD features
- Written 100% in Rust
- Strong focus on rigorousness to reduce user error
- Utilize industry standard (OCI Distribution Specification) for Image Distribution
- “Self Documenting” container images

Features

Not a Docker clone

- Utilize OCI image registry for distribution (AzureCR, DockerHub, AWS ECR..)
- Flexible networking
- Support both VNET/non-VNET containers
- Pre-Instantiation sanity checks
- Volume Hints
- Dynamic devfs rules allocation/generation (e.g. for block device, bhyve, etc…)
- Support Jail/Unjail ZFS datasets (e.g. for poudriere)
- Support running some Linux Docker/OCI Containers unmodified
- DTrace/USDT support on both the Runtime and Containers

Architecture



Using xc

Images

- Pull from Image Registries
- Convert a Jail (e.g. Bastille Jails) to container image
- Build using “Jailfile”

Networking

- Optional
- Synchronized with <xc:network:$NETWORK_NAME> pf tables
- Handle “Which interface the IP address should add to” for non VNET Jails
- Handle “Which interface is the bridge for the new epair” for VNET Jails
- Optionally handles automatic address allocation

Demo:
DockerHub & Linux Container

DTrace Support

- Allowing tracing per container (Jail)
- Wrapper around DWatch
- Enable valuable per-container performance/behavioural insight

DTrace Support - USDT
What is USDT?

- Customized probes defined in application
- Allow to trace application specific probe points
- Implemented in lots of software stacks
  - Erlang BEAM
  - Ruby

DTrace Support - USDT

- Support applications running in containers to register USDT probes
- The Runtime daemon itself also containers a number of USDT probes

Demo: Simple Erlang Container

Devfs ruleset management

- Container image can specify additional rules required
- Runtime automatically generates ruleset on demand, reuse identical ruleset
- Prompt user the generated devfs rules if required
- User can accept, or abort before the Jail created

Demo:
Diskless, networkless BHyve

Environment Variable Guarding

Environment Variable Guarding

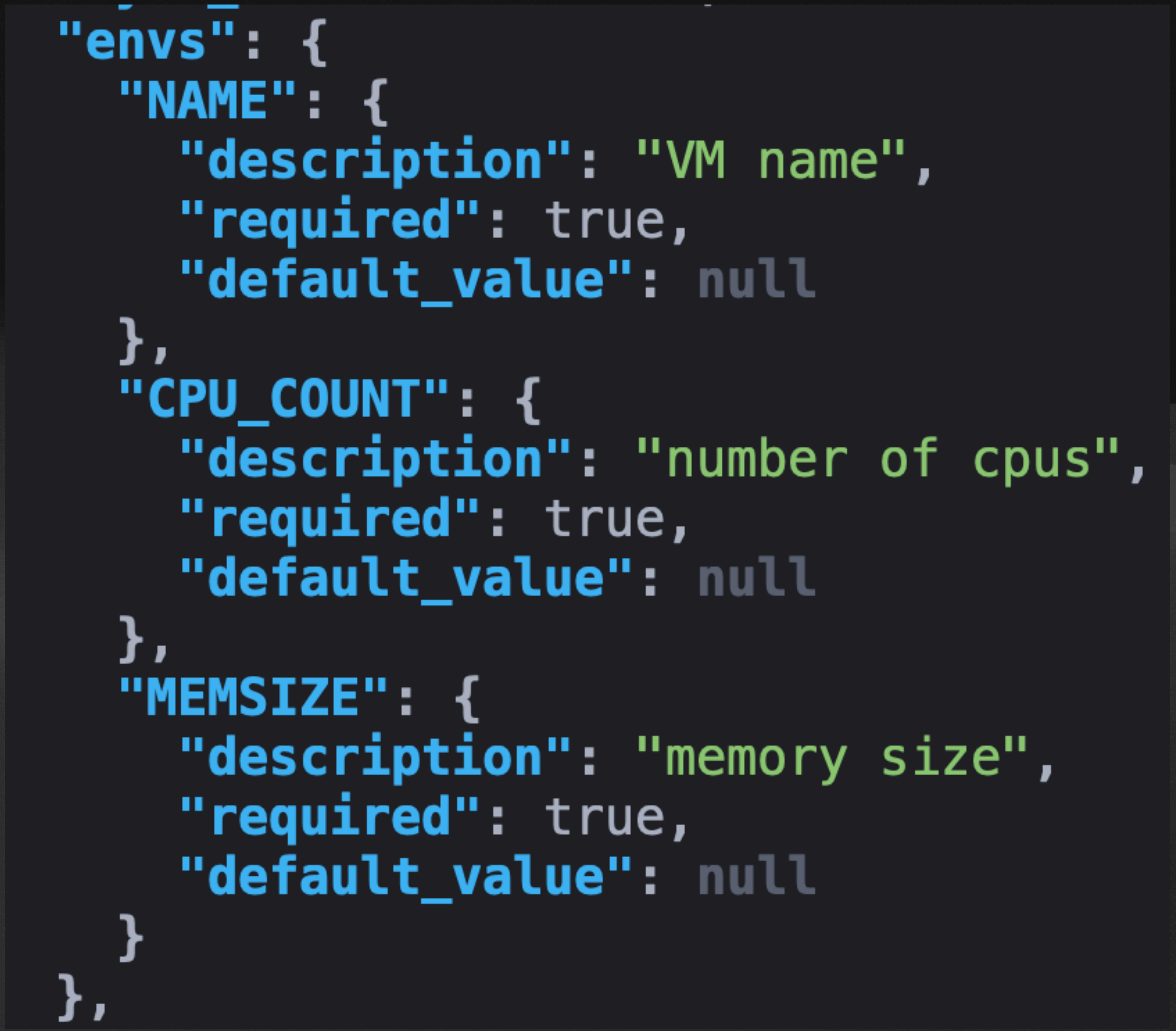
“Traditional” container

- Satisfiability check (if exists) often considered part of the “business logic”
  - Container must be created and run for validity checks - Expensive
  - No guarantee of such check even exists
  - No knowledge of required variables without consult external documentation/trial

Environment Variable Guarding

“xc” container image

- Image config contains specification of each environment variable
- Enable runtime to check for invalid configuration
- Provide useful feedback
- Extendable

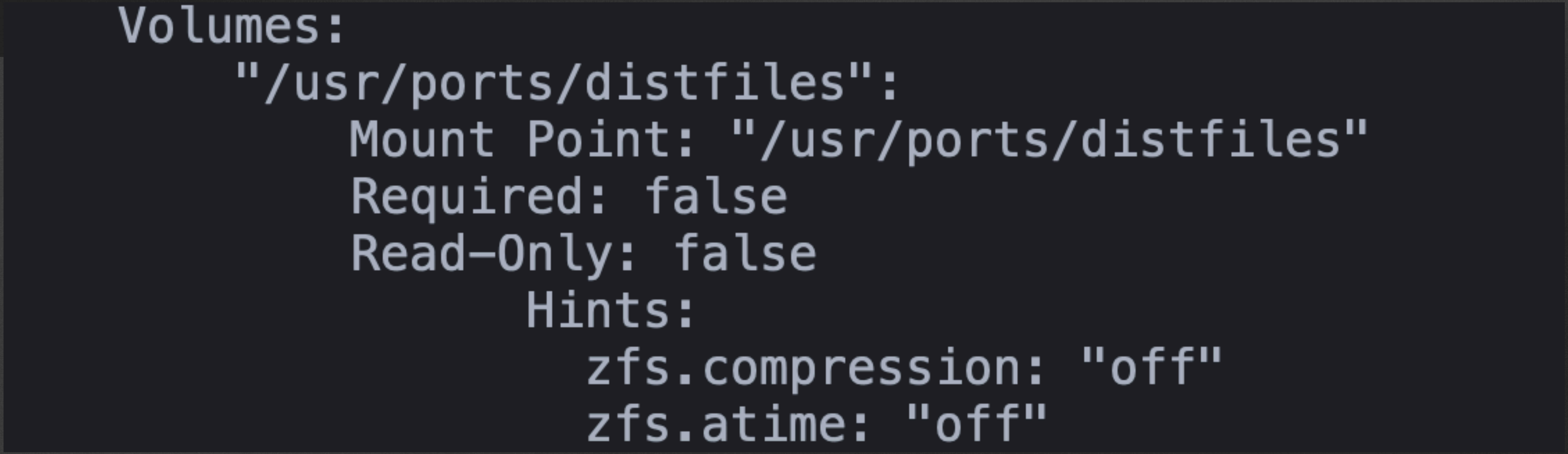


```
"envs": {
  "NAME": {
    "description": "VM name",
    "required": true,
    "default_value": null
  },
  "CPU_COUNT": {
    "description": "number of cpus",
    "required": true,
    "default_value": null
  },
  "MEMSIZE": {
    "description": "memory size",
    "required": true,
    "default_value": null
  }
},
```

ZFS

Volume Hints

- Allow developer to specify recommended ZFS properties for application volumes
- User can create volume base on the application specific purpose



```
    Volumes:
        "/usr/ports/distfiles":
            Mount Point: "/usr/ports/distfiles"
            Required: false
            Read-Only: false
                  Hints:
                    zfs.compression: "off"
                    zfs.atime: "off"
```

Jailed ZFS

- Allows Jails to manage ZFS datasets
- Useful for ZFS related applications
- Runtime keep tracks of allocation
- Poudriere

Demo: Poudriere

Future/Ongoing Work