Overlord:
Deploy Jails as Fast as You Code

BY JESÚS DANIEL COLMENARES OVIEDO

When AppJail, a BSD-3 licensed open source framework entirely written in `sh(1)` and C to create isolated, portable and easy to deploy environments using FreeBSD jails that behave like an application, was created, my intention was to test ports so as not to mess up my main environment. Today AppJail is more than just a script for testing ports, it is highly flexible and has some very useful automation features.

After AppJail reached stability and had been used on a variety of systems, I realized that just deploying a jail for every service I wanted was not feasible, especially when more and more services need to be deployed. Thus Director was born.

AppJail Director is a tool for running multi-jail environments on AppJail using a simple YAML specification. A Director file is used to define how one or more jails that make up your application are configured. Once you have a Director file, you can create and start your application with a single command: `appjail-director up`.

AppJail is more than just a script for testing ports.

Director is the first attempt to follow the "everything is code" philosophy related to AppJail. Director organizes jails into projects, so you create a project with one or more jails declaratively, and Director takes into account any changes you have made to that file or a related file (such as the Makejail used). If Director has seen a change, it doesn't hesitate to destroy your jail to recreate it. This sounds a bit crazy, but is best explained in *The Ephemeral Concept*:

> Director treats each jail as ephemeral. This does not mean that your jails will not persist after you stop them or restart your system, what it means is that Director assumes that it is safe to destroy the jails since you have clearly separated the data that should be persisted from the data considered ephemeral.

There are more details in the `appjail-ephemeral(7)` man page, but the principle is the same as above.

Director by itself does not deploy jails, it needs instructions that perform configuration, package installation, etc., so it heavily exploits a feature of AppJail called Makejails, a simple text file that automates the steps of creating a jail. There are many created in the Centralized Repository, but nothing prevents you from using your own repository to host your Makejails.

Both AppJail and Director simplify my life a lot, however you have to deal with a problem that neither AppJail nor Director solve, which is orchestrating jails on many servers. AppJail and Director combined with SSH may be workable for a few servers, but when you have more and more, this is just painful. Thus Overlord was born.

Overlord is a fast, distributed orchestrator for FreeBSD jails oriented to GitOps. You define a file with the service intended to run on your cluster and deployment takes seconds to minutes.

Fortunately for Overlord (and for me), AppJail and Director are mature, so it's a smart move to reuse those extensively tested tools and combine them with Overlord. This is what I have done and borrowing the same philosophy from Director that "everything is code" is why the orchestrator is easy to use. Another decision I have made is that everything in Overlord is asynchronous. Deployments can take a long time when the service is huge, but even when the deployment only takes a little while, it's much better to send instructions declaratively and let Overlord handle our work. In this article, in more detail, we'll look at many of the things I've said here.

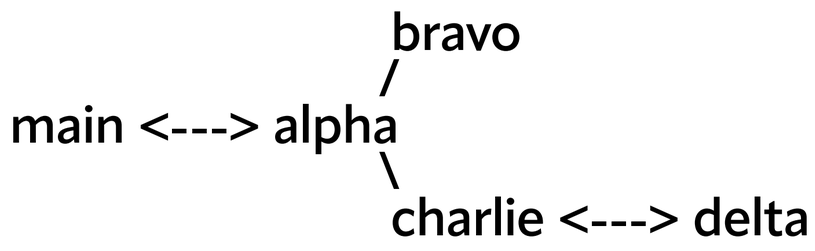
Architecture

The Overlord architecture is described as a tree chain architecture. Each Overlord instance running the API server can be configured to group other chains. Each member can be further configured to group more chains together. However, although this step can be performed almost infinitely, doing this without caution can introduce latency, so it is important to know how you plan to organize your servers.

Fortunately for Overlord, AppJail and Director are mature

The reason for choosing this architecture is because it is very simple and scales very well, so sharing resources among many servers is just sticking each chain together forming a cluster to deploy projects.

This architecture also abstracts the way projects are deployed. A user who wants to deploy a project does not need to know the endpoint of each chain, only the first one (also known as the root chain) is enough. This is because each chain is tagged with an arbitrary string, so a user only needs to specify in his deployment file the endpoint of the root chain, the access token and the labels. Although labels are subjective, this can represent a desire. For example, we can label servers with the string vm-only for those servers that have the ability to deploy virtual machines or db-only for databases, it is really very arbitrary.



Assume that only charlie and delta have the `db-only` label. To deploy projects to the API servers with the specified labels, the client must make an HTTP request to main, specifying the chain `alpha.charlie` and `alpha.charlie.delta`. This is done transparently and does not require user intervention.

main . alpha . charlie
&
main . alpha . charlie . delta

Smart Timeouts

What happens if a chain is down? If the root chain is down, we can do nothing, although we can specify more than one (although in the rest of this document we will only use one). However, if a chain after the root chain is down something interesting happens.

When a chain positioned after the root chain is down, the chain that detects the error can blacklist the failed chain for a while. The blacklist is used to not display failed chains, although it is not forbidden to attempt to connect to a failed chain through another chain, this is because successfully connecting to a blacklisted chain automatically re-enables it.

Each blacklisted chain has an assigned time to remain on the blacklist. After a period of time, the chain is removed from the blacklist, however, if the chain continues to fail, it is added back to the blacklist but for a longer time. This time increment has a limit so as not to disable a chain forever.

Each blacklisted chain has an assigned time to remain on the blacklist.

Doing the above has the good effect that an HTTP request can decrease the time to complete since there is no need to connect to a failed chain.

Deploying Projects

The best way to demonstrate Overlord is to deploy a small project.

Note that a project can have more than one jail, however, in the following example only one jail is needed.

filebrowser.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
projectName: filebrowser
projectFile: |
  options:
    - virtualnet: ':<random> default'
    - nat:
  services:
    filebrowser:
      makejail: 'gh+AppJail-makejails/filebrowser'
```

```
    volumes:
      - db: filebrowser-db
      - log: filebrowser-log
      - www: filebrowser-www
    start-environment:
      - FB_NOAUTH: 1
    arguments:
      - filebrowser_tag: 14.2
    options:
      - expose: '8432:8080 ext_if:tailscale0 on_if:tailscale0'
  default_volume_type: '<volumefs>'
  volumes:
    db:
      device: /var/appjail-volumes/filebrowser/db
    log:
      device: /var/appjail-volumes/filebrowser/log
    www:
      device: /var/appjail-volumes/filebrowser/www
```

You have noticed that I'm not specifying the access token and entry point explicitly, but through environment variables, which are loaded through the `.env` file:

.env:

```
ENTRYPOINT=http://127.0.0.1:8888
TOKEN=<access token>
```

And now another question: how is the access token generated? This is easy, the token is generated from the machine running the Overlord instance you want to contact, however only who has the privileges to access the secret key (which is generated pseudo-randomly by default) can generate tokens.

```
# OVERLORD_CONFIG=/usr/local/etc/overlord.yml overlord gen-token
```

The next step is simply to apply the deployment file.

```
$ overlord apply -f filebrowser.yml
```

If there is no output, everything is fine, however this does not mean that the project is deployed. When a deployment file is applied and contains the specification to deploy a project (in the above case a **directorProject**), it is queued waiting for its turn. However, since there are no other projects currently running, our project will be deployed as fast as possible.

```
$ overlord get-info -f filebrowser.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
```

```
projects:
  filebrowser:
    state: DONE
    last_log: 2025-04-22_17h57m45s
    locked: False
    services:
      - {'name': 'filebrowser', 'status': 0, 'jail': 'e969b06736'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
    last_update: 7 minutes and 42.41 seconds
    job_id: 14
    restarted: False
    labels:
     error: False
     message: None
```

Metadata

Metadata is used to create small files (such as configuration files) that can be used when deploying projects or VMs. While a git hosting such as GitLab, GitHub, Gitea, etc., is very useful in combination with Makejails, you can use Metadata instead of relying on a git hosting to further configure the service or VM you are deploying.

The other advantage of metadata is that it can be shared between different deployments. For example, by deploying virtual machines that share the same **`sshd_config(5)`** and **`authorized_keys`** files.

tor.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
projectName: tor
projectFile: |
  options:
    - virtualnet: ':<random> address:10.0.0.50 default'
    - nat:
  services:
    tor:
      makejail: !ENV '${OVERLORD_METADATA}/tor.makejail'
      volumes:
        - data: ‘/var/db/tor’
```

```
  volumes:
    data:
      device: '/var/appjail-volumes/tor/data'
```

metadata.yml:

```
kind: metadata
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
metadata:
  tor.makejail: |
    OPTION start
    OPTION overwrite=force

    INCLUDE gh+DtxdF/efficient-makejail

    PKG tor

    CMD echo "SocksPort 0.0.0.0:9050" > /usr/local/etc/tor/torrc
    CMD echo "HTTPTunnelPort 0.0.0.0:9080" >> /usr/local/etc/tor/torrc

    SERVICE tor oneenable
    SERVICE tor start
```

.env:

```
ENTRYPOINT=http://127.0.0.1:8888
TOKEN=<access token>
```

There is no difference between deploying a project and the metadata from the user's point of view. However, metadata is not queued, it is simply written (asynchronously) to disk.

```
$ overlord apply -f metadata.yml
$ overlord apply -f tor.yml
$ overlord get-info -f metadata.yml -t metadata
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  metadata:
    tor.makejail: |
```

```
      OPTION start
      OPTION overwrite=force

      INCLUDE gh+DtxdF/efficient-makejail

      PKG tor

      CMD echo "SocksPort 0.0.0.0:9050" > /usr/local/etc/tor/torrc
      CMD echo "HTTPTunnelPort 0.0.0.0:9080" >> /usr/local/etc/tor/torrc

      SERVICE tor oneenable
      SERVICE tor start
$ overlord get-info -f tor.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  projects:
    tor:
      state: UNFINISHED
      last_log: 2025-04-22_18h40m30s
      locked: True
      services:
        - {'name': 'tor', 'status': 0, 'jail': '7ce0dfdcef'}
      up:
        operation: RUNNING
        last_update: 38.01 seconds
        job_id: 16
```

Deploying FreeBSD VMs

Overlord can deploy virtual machines thanks to the great vm-bhyve project. A virtual machine isolates many parts that a jail cannot, with the overhead that such a thing implies, however that overhead is not a problem depending on what you are doing.

This deployment works as follows: A director file is created (overlord does this internally), which is used to further create a jail that represents the environment that must have vm-bhyve installed, must be configured to use the firewall (one supported by FreeBSD) and must be configured with the bridge used by the VMs. This sounds really complicated, but there is a Makejail that does it, so take a look at it for details. The Makejail mentioned above creates the environment with vm-bhyve-devel installed, configures pf(4) and creates a bridge with an assigned IPv4 (`192.168.8.1/24`), so we must assign our VM an IPv4 between that range. `pf(4)` is not configured to isolate further connections, so an application inside the VM can "escape" to other services, which may or may not be desirable depending on what it is doing.

vm.yml:

```
kind: vmJail
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - !ENV '${DST}'
vmName: !ENV '${VM}'
makejail: 'gh+DtxdF/vm-makejail'
template:
  loader: 'bhyveload'
  cpu: !ENV '${CPU}'
  memory: !ENV '${MEM}'
  network0_type: 'virtio-net'
  network0_switch: 'public'
  wired_memory: 'YES'
diskLayout:
  driver: 'nvme'
  size: !ENV '${DISK}'
  from:
    type: 'components'
  components:
    - base.txz
    - kernel.txz
  osArch: amd64
  osVersion: !ENV '${VERSION}-RELEASE'
disk:
  scheme: 'gpt'
  partitions:
    - type: 'freebsd-boot'
      size: '512k'
      alignment: '1m'
    - type: 'freebsd-swap'
      size: !ENV '${SWAP}'
      alignment: '1m'
    - type: 'freebsd-ufs'
      alignment: '1m'
      format:
        flags: '-Uj'
  bootcode:
    bootcode: '/boot/pmbr'
    partcode: '/boot/gptboot'
    index: 1
fstab:
```

```
    - device: '/dev/nda0p3'
      mountpoint: '/'
      type: 'ufs'
      options: 'rw,sync'
      dump: 1
      pass: 1
    - device: '/dev/nda0p2'
      mountpoint: 'none'
      type: 'swap'
      options: 'sw'
      dump: 0
      pass: 0
script-environment:
  - HOSTNAME: !ENV '${HOSTNAME}'
script: |
  set -xe
  set -o pipefail

  . "/metadata/environment"

  sysrc -f /mnt/etc/rc.conf ifconfig_vtnet0="inet 192.168.8.2/24"
  sysrc -f /mnt/etc/rc.conf defaultrouter="192.168.8.1"
  sysrc -f /mnt/etc/rc.conf fsck_y_enable="YES"
  sysrc -f /mnt/etc/rc.conf clear_tmp_enable="YES"
  sysrc -f /mnt/etc/rc.conf dumpdev="NO"
  sysrc -f /mnt/etc/rc.conf moused_nondefault_enable="NO"
  sysrc -f /mnt/etc/rc.conf hostname="${HOSTNAME}"

  if [ -f "/metadata/resolv.conf" ]; then
    cp -a /metadata/resolv.conf /mnt/etc/resolv.conf
  fi

  if [ -f "/metadata/loader.conf" ]; then
    cp /metadata/loader.conf /mnt/boot/loader.conf
  fi

  if [ -f "/metadata/zerotier_network" ]; then
    pkg -c /mnt install -y zerotier

    zerotier_network='head -1 -- "/metadata/zerotier_network"'

    cat << EOF > /mnt/etc/rc.local
    while :; do
      if ! /usr/local/bin/zerotier-cli join ${zerotier_network}; then
        sleep 1
        continue
      fi
```

```
    break
  done

  rm -f /etc/rc.local
EOF

  sysrc -f /mnt/etc/rc.conf zerotier_enable="YES"
elif [ -f "/metadata/ts_auth_key" ]; then
  pkg -c /mnt install -y tailscale

    ts_auth_key='head -1 -- "/metadata/ts_auth_key"'

    echo "/usr/local/bin/tailscale up --accept-dns=false --auth-key=\"${ts_auth_key}\" &&
rm -f /etc/rc.local" > /mnt/etc/rc.local

    sysrc -f /mnt/etc/rc.conf tailscaled_enable="YES"
  fi

  if [ -f "/metadata/timezone" ]; then
    timezone='head -1 -- "/metadata/timezone"'

    ln -fs "/usr/share/zoneinfo/${timezone}" /mnt/etc/localtime
  fi

  if [ -f "/metadata/sshd_config" ]; then
    sysrc -f /mnt/etc/rc.conf sshd_enable="YES"
    cp /metadata/sshd_config /mnt/etc/ssh/sshd_config
  fi

  if [ -f "/metadata/ssh_key" ]; then
    cp /metadata/ssh_key /mnt/etc/ssh/authorized_keys
  fi

  if [ -f "/metadata/sysctl.conf" ]; then
    cp /metadata/sysctl.conf /mnt/etc/sysctl.conf
  fi

  if [ -f "/metadata/pkg.conf" ]; then
    mkdir -p /mnt/usr/local/etc/pkg/repos
    cp /metadata/pkg.conf /mnt/usr/local/etc/pkg/repos/Latest.conf
  fi
metadata:
  - resolv.conf
  - loader.conf
  - timezone
  - sshd_config
```

```
  - ssh_key
  - sysctl.conf
  - pkg.conf
  - ts_auth_key
```

metadata.yml:

```
kind: metadata
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - vm-only
metadata:
  ts_auth_key: '<tailscale auth key>'
  resolv.conf: |
    nameserver 192.168.1.107
  timezone: 'America/Caracas'
  loader.conf: |
    nvme_load="YES"
    if_bridge_load="YES"
    bridgestp_load="YES"
    if_wg_load="YES"
    kern.racct.enable=1
ssh_key: '<SSH public key>'
sshd_config: |
  # Ports
  Port 22

  # Authentication
  PubkeyAuthentication yes
  AuthenticationMethods publickey
  PermitRootLogin prohibit-password
  PrintMotd no

  # Forwarding
  X11Forwarding no
  AllowAgentForwarding yes

  # Connection checks
  ClientAliveCountMax 3
  ClientAliveInterval 15

  # Compression
  Compression no
```

```
    # Limits
    LoginGraceTime 40

    # Public keys
    AuthorizedKeysFile      /etc/ssh/authorized_keys

    # SFTP
    Subsystem sftp internal-sftp
  sysctl.conf: |
    # A bit of hardening
    security.bsd.see_other_uids=0
    security.bsd.see_other_gids=0
    security.bsd.see_jail_proc=0
    kern.randompid=1

    # Allow packet filtering in if_bridge(4)
    net.link.bridge.pfil_member=1
    net.link.bridge.pfil_bridge=1
  pkg.conf: |
    FreeBSD: {
      url: "pkg+http://pkg.FreeBSD.org/${ABI}/latest",
      mirror_type: "srv",
      signature_type: "fingerprints",
      fingerprints: "/usr/share/keys/pkg",
      enabled: yes
    }
```

.profile-vmtest.env:

```
ENTRYPOINT=http://127.0.0.1:8888
TOKEN=<access token>
VM=vmtest
CPU=1
MEM=256M
DISK=10G
VERSION=14.2
SWAP=1G
HOSTNAME=vmtest
DST=provider
```

Instead of copy and paste the deployment file each time you want to deploy a virtual machine, it is preferable to create several environment (or profile-like) files.

```
$ overlord -e .profile-vmtest.env apply -f metadata.yml
$ overlord -e .profile-vmtest.env apply -f vm.yml
$ overlord -e .profile-vmtest.env get-info -f vm.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
```

```
  labels:
    - all
    - provider
    - vm-only
  projects:
    vmtest:
      state: UNFINISHED
      last_log: 2025-04-22_20h19m34s
      locked: True
      services:
        - {'name': 'vm', 'status': 0, 'jail': 'vmtest'}
      up:
        operation: RUNNING
        last_update: 58.85 seconds
        job_id: 17
```

Depending on the type of installation, this may take some time. In the above case we chose to install FreeBSD from its components, so if the server does not have them yet or if it has them but they change remotely (for example: modification time), Overlord will proceed to download them.

```
$ overlord -e .profile-vmtest.env get-info -f vm.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - provider
    - vm-only
  projects:
    vmtest:
      state: DONE
      last_log: 2025-04-22_20h19m34s
      locked: False
      services:
        - {‘name’: 'vm', 'status': 0, 'jail': 'vmtest'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 6 minutes and 10.02 seconds
        job_id: 17
        restarted: False
$ overlord -e .profile-vmtest.env get-info -f vm.yml -t vm --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
```

```
  labels:
    - all
    - provider
    - vm-only
  projects:
    vmtest:
      virtual-machines:
          operation: COMPLETED
          output: |
            md0 created
            md0p1 added
            md0p2 added
            md0p3 added
            /dev/md0p3: 9214.0MB (18870272 sectors) block size 32768, fragment size 4096
               using 15 cylinder groups of 625.22MB, 20007 blks, 80128 inodes.
               with soft updates
            super-block backups (for fsck_ffs -b #) at:
             192, 1280640, 2561088, 3841536, 5121984, 6402432, 7682880, 8963328, 10243776,
             11524224, 12804672, 14085120, 15365568, 16646016, 17926464
            Using inode 4 in cg 0 for 75497472 byte journal
            bootcode written to md0
            partcode written to md0p1
            ifconfig_vtnet0:  -> inet 192.168.8.2/24
            defaultrouter: NO -> 192.168.8.1
            fsck_y_enable: NO -> YES
            clear_tmp_enable: NO -> YES
            dumpdev: NO -> NO
            moused_nondefault_enable: YES -> NO
            hostname:  -> vmtest
            [vmtest.appjail] Installing pkg-2.1.0...
            [vmtest.appjail] Extracting pkg-2.1.0: .......... done
            Updating FreeBSD repository catalogue...
            [vmtest.appjail] Fetching meta.conf: . done
            [vmtest.appjail] Fetching data.pkg: .......... done
            Processing entries: .......... done
            FreeBSD repository update completed. 35950 packages processed.
            All repositories are up to date.
            The following 2 package(s) will be affected (of 0 checked):

            New packages to be INSTALLED:
               ca_root_nss: 3.108
               tailscale: 1.82.5

            Number of packages to be installed: 2

            The process will require 35 MiB more space.
            11 MiB to be downloaded.
```

```
[vmtest.appjail] [1/2] Fetching tailscale-1.82.5.pkg: .......... done
[vmtest.appjail] [2/2] Fetching ca_root_nss-3.108.pkg: .......... done
Checking integrity... done (0 conflicting)
[vmtest.appjail] [1/2] Installing ca_root_nss-3.108...
[vmtest.appjail] [1/2] Extracting ca_root_nss-3.108: ....... done
Scanning /usr/share/certs/untrusted for certificates...
Scanning /usr/share/certs/trusted for certificates...
Scanning /usr/local/share/certs for certificates...
[vmtest.appjail] [2/2] Installing tailscale-1.82.5...
[vmtest.appjail] [2/2] Extracting tailscale-1.82.5: ...... done
=====
Message from ca_root_nss-3.108:

--
FreeBSD does not, and can not warrant that the certification authorities
whose certificates are included in this package have in any way been
audited for trustworthiness or RFC 3647 compliance.

Assessment and verification of trust is the complete responsibility of
the system administrator.

This package installs symlinks to support root certificate discovery
for software that either uses other cryptographic libraries than
OpenSSL, or use OpenSSL but do not follow recommended practice.

If you prefer to do this manually, replace the following symlinks with
either an empty file or your site-local certificate bundle.

  * /etc/ssl/cert.pem
  * /usr/local/etc/ssl/cert.pem
  * /usr/local/openssl/cert.pem
tailscaled_enable:  -> YES
sshd_enable: NO -> YES
vm_list:  -> vmtest
Starting vmtest
  * found guest in /vm/vmtest
  * booting...
newfs: soft updates journaling set
 + set -o pipefail
 + . /metadata/environment
 + export 'HOSTNAME=vmtest'
 + sysrc -f /mnt/etc/rc.conf 'ifconfig_vtnet0=inet 192.168.8.2/24'
 + sysrc -f /mnt/etc/rc.conf 'defaultrouter=192.168.8.1'
 + sysrc -f /mnt/etc/rc.conf 'fsck_y_enable=YES'
 + sysrc -f /mnt/etc/rc.conf 'clear_tmp_enable=YES'
 + sysrc -f /mnt/etc/rc.conf 'dumpdev=NO'
 + sysrc -f /mnt/etc/rc.conf 'moused_nondefault_enable=NO'
```

```
+ sysrc -f /mnt/etc/rc.conf 'hostname=vmtest'
+ [ -f /metadata/resolv.conf ]
+ cp -a /metadata/resolv.conf /mnt/etc/resolv.conf
+ [ -f /metadata/loader.conf ]
+ cp /metadata/loader.conf /mnt/boot/loader.conf
+ [ -f /metadata/zerotier_network ]
+ [ -f /metadata/ts_auth_key ]
+ pkg -c /mnt install -y tailscale
+ head -1 -- /metadata/ts_auth_key
+ ts_auth_key=[REDACTED]
+ echo '/usr/local/bin/tailscale up --accept-dns=false --auth-key="[REDACTED]"
&& rm -f /etc/rc.local'
+ sysrc -f /mnt/etc/rc.conf 'tailscaled_enable=YES'
+ [ -f /metadata/timezone ]
+ head -1 -- /metadata/timezone
+ timezone=America/Caracas
+ ln -fs /usr/share/zoneinfo/America/Caracas /mnt/etc/localtime
+ [ -f /metadata/sshd_config ]
+ sysrc -f /mnt/etc/rc.conf 'sshd_enable=YES'
+ cp /metadata/sshd_config /mnt/etc/ssh/sshd_config
+ [ -f /metadata/ssh_key ]
+ cp /metadata/ssh_key /mnt/etc/ssh/authorized_keys
+ [ -f /metadata/sysctl.conf ]
+ cp /metadata/sysctl.conf /mnt/etc/sysctl.conf
+ [ -f /metadata/pkg.conf ]
+ mkdir -p /mnt/usr/local/etc/pkg/repos
+ cp /metadata/pkg.conf /mnt/usr/local/etc/pkg/repos/Latest.conf
last_update: 5 minutes and 12.6 seconds
job_id: 17
```

As I'm using tailscale and the VM above is about to be configured to join my tailnet, after a while it should appear in the node list:

```
$ tailscale status
...
100.124.236.28  vmtest               REDACTED@    freebsd -
$ ssh root@100.124.236.28
The authenticity of host '100.124.236.28 (100.124.236.28)' can't be established.
ED25519 key fingerprint is SHA256:Oc61mU8erpgS2evkwL9WhOOl4Ze94sSNfhImLy3b4UQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '100.124.236.28' (ED25519) to the list of known hosts.
root@vmtest:~ #
```

Service Discovery

A service is deployed, however depending on how many servers you have in your cluster the service endpoint can be a difficult part to build. You know the port and external inter-

face used and certainly the IP address is easy to get, but it is much easier to use DNS, which is its primary purpose. The service can be deployed on different servers but its endpoint is always the same.

SkyDNS is an older but powerful protocol that, in combination with Etcd, can provide easy service discovery. Overlord can be configured to use both Etcd and SkyDNS. Let's deploy our Etcd cluster.

etcd.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
    - r2
    - centralita
projectName: etcd-cluster
projectFile: |
  options:
    - alias:
    - ip4_inherit:
  services:
    etcd:
      makejail: 'gh+AppJail-makejails/etcd'
      arguments:
        - etcd_tag: '14.2-34'
      volumes:
        - data: etcd-data
      start-environment:
        - ETCD_NAME: !ENV '${NAME}'
        - ETCD_ADVERTISE_CLIENT_URLS: !ENV 'http://${HOSTIP}:2379'
        - ETCD_LISTEN_CLIENT_URLS: !ENV 'http://${HOSTIP}:2379'
        - ETCD_LISTEN_PEER_URLS: !ENV 'http://${HOSTIP}:2380'
        - ETCD_INITIAL_ADVERTISE_PEER_URLS: !ENV 'http://${HOSTIP}:2380'
        - ETCD_INITIAL_CLUSTER_TOKEN: 'etcd-demo-cluster'
        - ETCD_INITIAL_CLUSTER: !ENV '${CLUSTER}'
        - ETCD_INITIAL_CLUSTER_STATE: 'new'
        - ETCD_HEARTBEAT_INTERVAL: '5000'
        - ETCD_ELECTION_TIMEOUT: '50000'
        - ETCD_LOG_LEVEL: 'error'
  default_volume_type: '<volumefs>'
  volumes:
    data:
      device: /var/appjail-volumes/etcd-cluster/data
environment:
```

```
  CLUSTER: 'etcd0=http://100.65.139.52:2380,etcd1=http://100.109.0.125:2380,etc-
d2=http://100.96.18.2:2380'
labelsEnvironment:
  desktop:
    NAME: 'etcd0'
    HOSTIP: '100.65.139.52'
  r2:
    NAME: 'etcd1'
    HOSTIP: '100.109.0.125'
  centralita:
    NAME: 'etcd2'
    HOSTIP: '100.96.18.2'
```

Profit!

```
$ overlord apply -f etcd.yml
$ overlord get-info -f etcd.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
  - vm-only
projects:
  etcd-cluster:
    state: DONE
    last_log: 2025-04-23_02h28m36s
    locked: False
    services:
      - {'name': 'etcd', 'status': 0, 'jail': 'f094a31c46'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 8 minutes and 11.51 seconds
      job_id: 20
      restarted: False
      labels:
       error: False
       message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
```

```
    - centralita
    - services
  projects:
    etcd-cluster:
      state: DONE
      last_log: 2025-04-23_02h28m37s
      locked: False
      services:
        - {'name': 'etcd', 'status': 0, 'jail': '1ff836df47'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 5 minutes and 37.82 seconds
        job_id: 2
        restarted: False
        labels:
         error: False
         message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: r2
  labels:
    - all
    - r2
    - services
  projects:
    etcd-cluster:
      state: DONE
      last_log: 2025-04-23_02h28m38s
      locked: False
      services:
        - {'name': 'etcd', 'status': 0, 'jail': '756ae9d5ca'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 5 minutes and 5.04 seconds
        job_id: 1
        restarted: False
        labels:
         error: False
         message: None
```

Once the deployment is done, it is time to configure each Overlord instance with the fol-

lowing parameters.

/usr/local/etc/overlord.yml:

```
etcd:
  100.65.139.52: {}
  100.109.0.125: {}
  100.96.18.2: {}
```

Remember to restart the Overlord processes for the changes to take effect.

```
supervisorctl restart overlord:
```

The next service we have to deploy is CoreDNS. Thanks to it we can use SkyDNS through the Etcd plugin.

coredns.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
    - r2
projectName: dns-server
projectFile: |
  options:
    - alias:
    - ip4_inherit:
  services:
    coredns:
      makejail: !ENV '${OVERLORD_METADATA}/coredns.makejail'
```

metadata.yml:

```
kind: metadata
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
    - r2
metadata:
  Corefile: |
    .:53 {
      bind tailscale0
      log
```

```
    errors
    forward . 208.67.222.222 208.67.220.220
    etcd overlord.lan. {
      endpoint http://100.65.139.52:2379 http://100.109.0.125:2379 http://100.96.18.2:2379
    }
    hosts /etc/hosts namespace.lan.
    cache 30
  }
coredns.hosts: |
  100.65.139.52    controller.namespace.lan
  100.96.18.2      centralita.namespace.lan
  100.127.18.7     fbsd4dev.namespace.lan
  100.123.177.93   provider.namespace.lan
  100.109.0.125   r2.namespace.lan
  172.16.0.3      cicd.namespace.lan
coredns.makejail: |
  OPTION start
  OPTION overwrite=force
  OPTION healthcheck="health_cmd:jail:service coredns status" "recover_cmd:jail:service
coredns restart"

  INCLUDE gh+DtxdF/efficient-makejail

  CMD mkdir -p /usr/local/etc/pkg/repos
  COPY ${OVERLORD_METADATA}/coredns.pkg.conf /usr/local/etc/pkg/repos/Latest.conf

  PKG coredns

  CMD mkdir -p /usr/local/etc/coredns
  COPY ${OVERLORD_METADATA}/Corefile /usr/local/etc/coredns/Corefile

  COPY ${OVERLORD_METADATA}/coredns.hosts /etc/hosts

  SYSRC coredns_enable=YES
  SERVICE coredns start
coredns.pkg.conf: |
  FreeBSD: {
    url: "pkg+https://pkg.FreeBSD.org/${ABI}/latest",
    mirror_type: "srv",
    signature_type: "fingerprints",
    fingerprints: "/usr/share/keys/pkg",
    enabled: yes
  }
```

As you can see in the CoreDNS configuration file, the **`overlord.lan`**. zone is assumed but by default Overlord only uses `.` which does not make sense for this context, so proceed to configure Overlord with this in mind and deploy CoreDNS.

/usr/local/etc/overlord.yml:

```
skydns:
  zone: 'overlord.lan.'
```

Note: Remember to restart the Overlord processes for the changes to take effect.

```
$ overlord apply -f metadata.yml
$ overlord apply -f coredns.yml
$ overlord get-info -f coredns.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  projects:
    dns-server:
      state: DONE
      last_log: 2025-04-23_13h32m49s
      locked: False
      services:
        - {'name': 'coredns', 'status': 0, 'jail': '8106aaca6d'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 2 minutes and 30.14 seconds
        job_id: 25
        restarted: False
        labels:
         error: False
         message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: r2
  labels:
    - all
    - r2
    - services
  projects:
    dns-server:
      state: DONE
      last_log: 2025-04-23_13h32m54s
      locked: False
      services:
        - {'name': 'coredns', 'status': 0, 'jail': '9516eb48aa'}
```

```
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 3 minutes and 26.9 seconds
      job_id: 4
      restarted: False
      labels:
       error: False
       message: None
```

Our Etcd cluster is up and running and our DNS servers are up and running. Clients should be configured to resolve DNS hostnames through those DNS servers, so configure them in their `resolv.conf(5)` or similar.

/etc/resolv.conf:

```
nameserver 100.65.139.52
nameserver 100.109.0.125
```

Our Frankenstein is alive! So the next step is to deploy a service and test if all parts are working as expected.

homebox.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - centralita
projectName: homebox
projectFile: |
  options:
    - virtualnet: ':<random> default'
    - nat:
  services:
    homebox:
      makejail: gh+AppJail-makejails/homebox
      options:
        - expose: '8666:7745 ext_if:tailscale0 on_if:tailscale0'
        - label: 'overlord.skydns:1'
        - label: 'overlord.skydns.group:homebox'
        - label: 'overlord.skydns.interface:tailscale0'
      volumes:
        - data: homebox-data
```

```
      arguments:
        - homebox_tag: 14.2
  default_volume_type: '<volumefs>'
  volumes:
    data:
      device: /var/appjail-volumes/homebox/data
```

So simple. Overlord intercepts the labels that we define in our Director file and based on that it creates the DNS record.

```
$ overlord apply -f homebox.yml
$ overlord get-info -f homebox.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
  projects:
    homebox:
      state: DONE
      last_log: 2025-04-23_15h44m38s
      locked: False
      services:
        - {'name': 'homebox', 'status': 0, 'jail': '1f97e32f36'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 4 minutes and 4.1 seconds
        job_id: 6
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             homebox:
               error: False
               message: None
          skydns:
           services:
             homebox:
               error: False
               message: (project:homebox, service:homebox, records:[address:True,ptr:None,s-
rv:None] records has been updated.
```

Finally, our endpoint is http://homebox.overlord.lan:8666/

Load Balancing

An interesting fact about SkyDNS is that multiple domains are grouped, so if we deploy a service on multiple servers and they use the same group, a DNS request will return three A records (in the case of IPv4), or what amounts to three IPv4 addresses.

hello-http.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - services
projectName: hello-http
projectFile: |
  options:
    - virtualnet: ':<random> default'
    - nat:
  services:
    darkhttpd:
      makejail: 'gh+DtxdF/hello-http-makejail'
      options:
        - expose: '9128:80 ext_if:tailscale0 on_if:tailscale0'
        - label: 'appjail.dns.alt-name:hello-http'
        - label: 'overlord.skydns:1'
        - label: 'overlord.skydns.group:hello-http'
        - label: 'overlord.skydns.interface:tailscale0'
      arguments:
        - darkhttpd_tag: 14.2
```

A nice side effect of this is that services are load-balanced in a round-robin fashion, although this is entirely client-dependent, but most modern ones do it.

```
$ overlord apply -f hello-http.yml
$ overlord get-info -f hello-http.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
entrypoint: main
chain: None
labels:
  - all
  - desktop
  - services
  - vm-only
projects:
  hello-http:
    state: DONE
```

```
    last_log: 2025-04-23_16h26m08s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '7c2225c5fe'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 2 minutes and 43.3 seconds
    job_id: 28
    restarted: False
    labels:
     error: False
     message: None
     load-balancer:
       services:
         darkhttpd:
           error: False
           message: None
     skydns:
       services:
         darkhttpd:
           error: False
           message: (project:hello-http, service:darkhttpd, records:[ad-
dress:True,ptr:None,srv:None] records has been updated.
    datacenter: http://127.0.0.1:8888
      entrypoint: main
      chain: centralita
      labels:
        - all
        - centralita
        - services
      projects:
        hello-http:
          state: DONE
          last_log: 2025-04-23_16h26m09s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '3822f65e97'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 2 minutes and 18.56 seconds
      job_id: 13
```

```
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             darkhttpd:
               error: False
               message: None
         skydns:
           services:
             darkhttpd:
               error: False
               message: (project:hello-http, service:darkhttpd, records:[ad-
dress:True,ptr:None,srv:None] records has been updated.
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: r2
  labels:
    - all
    - r2
    - services
  projects:
  hello-http:
    state: DONE
    last_log: 2025-04-23_16h26m10s
    locked: False
    services:
      - {‘name’: ‘darkhttpd’, ‘status’: 0, ‘jail’: ‘0e0e64eb3c’}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {‘errlevel’: 0, ‘message’: None, ‘failed’: []}
      last_update: 51.17 seconds
      job_id: 8
      restarted: False
      labels:
       error: False
       message: None
       load-balancer:
         services:
           darkhttpd:
             error: False
             message: None
       skydns:
         services:
```

```
              darkhttpd:
                error: False
                message: (project:hello-http, service:darkhttpd, records:[ad-
dress:True,ptr:None,srv:None] records has been updated.
$ host -t A hello-http.overlord.lan
hello-http.overlord.lan has address 100.65.139.52
hello-http.overlord.lan has address 100.109.0.125
hello-http.overlord.lan has address 100.96.18.2
$ curl http://hello-http.overlord.lan:9128/
curl http://hello-http.overlord.lan:9128
Hello, world!
UUID: 472ffbdb-9472-4aa2-95ff-39f4bde214df
$ curl http://hello-http.overlord.lan:9128/
Hello, world!
UUID: 7db3b268-87bb-4e81-8be3-e888378fa13b
```

However, I know that in most cases a more complex configuration is needed. Worse, as noted above this is client-dependent, so it may or may not fit your intent.

Fortunately, Overlord comes with an integration with HAProxy, or more specifically with Data Plane API, so your configuration can be as complex as you need.

haproxy.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
    - r2
projectName: load-balancer
projectFile: |
  options:
    - alias:
    - ip4_inherit:
  services:
    haproxy:
      makejail: !ENV '${OVERLORD_METADATA}/haproxy.makejail'
      arguments:
        - haproxy_tag: 14.2-dataplaneapi
      options:
        - label: 'overlord.skydns:1'
        - label: 'overlord.skydns.group:revproxy'
        - label: 'overlord.skydns.interface:tailscale0'
```

metadata.yml:

```
kind: metadata
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
    - r2
metadata:
  haproxy.makejail: |
    ARG haproxy_tag=13.5
    ARG haproxy_ajspec=gh+AppJail-makejails/haproxy

    OPTION start
    OPTION overwrite=force
    OPTION copydir=${OVERLORD_METADATA}
    OPTION file=/haproxy.conf

    FROM --entrypoint "${haproxy_ajspec}" haproxy:${haproxy_tag}

    INCLUDE gh+DtxdF/efficient-makejail

    SYSRC haproxy_enable=YES
    SYSRC haproxy_config=/haproxy.conf

    SERVICE haproxy start

    STOP

    STAGE start

    WORKDIR /dataplaneapi

    RUN daemon \
            -r \
            -t "Data Plane API" \
            -P .master \
            -p .pid \
            -o .log \
             ./dataplaneapi \
                 -f /usr/local/etc/dataplaneapi.yml \
                 --host=0.0.0.0 \
                 --port=5555 \
                 --spoe-dir=/usr/local/etc/haproxy/spoe \
```

```
                --haproxy-bin=/usr/local/sbin/haproxy \
                --reload-cmd="service haproxy reload" \
                --restart-cmd="service haproxy restart" \
                --status-cmd="service haproxy status" \
                --maps-dir=/usr/local/etc/haproxy/maps \
                --config-file=/haproxy.conf \
                --ssl-certs-dir=/usr/local/etc/haproxy/ssl \
                --general-storage-dir=/usr/local/etc/haproxy/general \
                --dataplane-storage-dir=/usr/local/etc/haproxy/dataplane \
                --log-to=file \
                --userlist=dataplaneapi
haproxy.conf: |
  userlist dataplaneapi
    user admin insecure-password cuwBvS5XMphtCNuC

  global
    daemon
    log 127.0.0.1:514 local0
    log-tag HAProxy

  defaults
    mode http
    log global
    option httplog
    timeout client 30s
    timeout server 50s
    timeout connect 10s
    timeout http-request 10s

frontend web
  bind :80
  default_backend web

backend web
  option httpchk HEAD /
  balance roundrobin
```

We are about to deploy HAProxy / Data Plane API on two servers. The reason for doing this is to avoid SPOF (single point of failure), at least if one instance of HAProxy / Data Plane API goes down at any time, the other will rescue us. However, Overlord can only point to one instance of Data Plane API, so if we use two (as below) servers, we need to specify one instance on one and a different instance on the other. As you can see in the Director file, we have used SkyDNS, so that clients can use the **`revproxy.overlord.lan`** domain instead of each individual IP address, with the advantage that even if one instance of HAProxy / Data Plane API is down, we have the other and the client can make requests to the other.

```
$ overlord apply -f metadata.yml
$ overlord apply -f haproxy.yml
$ overlord get-info -f haproxy.yml -t projects --filter-per-project
```

```
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  projects:
    load-balancer:
      state: DONE
      last_log: 2025-04-23_17h04m01s
      locked: False
      services:
        - {‘name’: ‘haproxy’, ‘status’: 0, ‘jail’: ‘8d92fc6d2d’}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {‘errlevel’: 0, ‘message’: None, ‘failed’: []}
        last_update: 2 minutes and 20.12 seconds
        job_id: 30
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             haproxy:
               error: False
               message: None
         skydns:
           services:
             haproxy:
               error: False
               message: (project:load-balancer, service:haproxy, records:[ad-
dress:True,ptr:None,srv:None] records has been updated.
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: r2
  labels:
    - all
    - r2
    - services
  projects:
    load-balancer:
      state: DONE
```

```
    last_log: 2025-04-23_17h04m02s
    locked: False
    services:
      - {'name': 'haproxy', 'status': 0, 'jail': '05c589c8a1'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 2 minutes and 53.27 seconds
    job_id: 10
    restarted: False
    labels:
     error: False
     message: None
     load-balancer:
       services:
         haproxy:
           error: False
           message: None
     skydns:
       services:
         haproxy:
                 error: False
                 message: (project:load-balancer, service:haproxy, records:[ad-
dress:True,ptr:None,srv:None] records has been updated.
```

/usr/local/etc/overlord.yml (centralita)

```
dataplaneapi:
  auth:
    username: 'admin'
    password: 'cuwBvS5XMphtCNuC'
  entrypoint: 'http://100.65.139.52:5555'
```

/usr/local/etc/overlord.yml (provider):

```
dataplaneapi:
  auth:
    username: 'admin'
    password: 'cuwBvS5XMphtCNuC'
  entrypoint: 'http://100.109.0.125:5555'
```

hello-http.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
```

```
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - centralita
    - provider
projectName: hello-http
projectFile: |
  options:
    - virtualnet: ':<random> default'
    - nat:
  services:
    darkhttpd:
      makejail: 'gh+DtxdF/hello-http-makejail'
      options:
        - expose: '9128:80 ext_if:tailscale0 on_if:tailscale0'
        - label: 'overlord.load-balancer:1'
        - label: 'overlord.load-balancer.backend:web'
        - label: 'overlord.load-balancer.interface:tailscale0'
        - label: 'overlord.load-balancer.interface.port:9128'
        - label: 'overlord.load-balancer.set.check:"enabled"'
      arguments:
        - darkhttpd_tag: 14.2
```

Profit!

```
$ overlord apply -f hello-http.yml
$ overlord get-info -f hello-http.yml -t projects --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: provider
  labels:
    - all
    - provider
    - vm-only
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_17h57m16s
      locked: False
      services:
        - {'name': 'darkhttpd', 'status': 0, 'jail': '79f16243de'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 1 minute and 22.1 seconds
        job_id: 1
```

```
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             darkhttpd:
               error: False
               message: (project:hello-http, service:darkhttpd, backend:web, serverid:-
fa8f94b1-6b2b-4cb4-a808-e9da46014c86, code:202, transaction_id:8fcf5d67-12df-4fcd-a2e3-
1f5f18fe1844, commit:1) server has been successfully added.
         skydns:
           services:
             darkhttpd:
               error: False
               message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
projects:
  hello-http:
    state: DONE
    last_log: 2025-04-23_17h57m16s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '52dfa071cb'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 1 minute and 19.53 seconds
      job_id: 15
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             darkhttpd:
               error: False
               message: (project:hello-http, service:darkhttpd, backend:web, serverid:f-
4b9e170-67bb-403e-88da-112c55b45fce, code:202, transaction_id:0fa886c8-68a6-4716-aa6b-
```

```
824aa3e776ad, commit:1) server has been successfully added.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
$ curl http://revproxy.overlord.lan
Hello, world!
UUID: 8579af73-7d11-40b3-8444-6dac62e34b8e
$ curl http://revproxy.overlord.lan
Hello, world!
UUID: e463b1d5-13eb-4f04-9b0a-caf4339a8058
```

Horizontal Autoscaling

Even when there are hundreds of servers, deploying projects is an easy task, however the problem with this approach is that we are wasting resources and the clients probably only use less than 5% of the resources of our cluster, or on the contrary, you can deploy your project on a few servers which you think is enough until you realize at some point that this is not enough, even worse, some servers can be down at any time for any reason. This is what Overlord autoscaling can solve.

hello-http.yml:

```
kind: directorProject
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - desktop
projectName: hello-http
projectFile: |
  options:
    - virtualnet: ':<random> default'
    - nat:
  services:
    darkhttpd:
      makejail: 'gh+DtxdF/hello-http-makejail'
      options:
        - expose: '9128:80 ext_if:tailscale0 on_if:tailscale0'
        - label: 'overlord.load-balancer:1'
        - label: 'overlord.load-balancer.backend:web'
        - label: 'overlord.load-balancer.interface:tailscale0'
        - label: 'overlord.load-balancer.interface.port:9128'
        - label: 'overlord.load-balancer.set.check:"enabled"'
      arguments:
        - darkhttpd_tag: 14.2
```

```
autoScale:
  replicas:
    min: 3
  labels:
    - services
    - provider
```

As you have probably noticed, we have specified two types of labels. This is the subtle difference between autoscaling and non-autoscaling deployments. Unlike non-autoscaling deployments, the labels in **`deployIn.labels`** are for using matching servers for autoscaling and monitoring, or in other words, the servers that match the specified labels (in this case **`desktop`**) are responsible for deployment, monitoring and, if necessary, redeployment. On the other side, the servers matching the labels in **`autoScale.labels`** (in this case **`services`** and **`provider`**) are for deploying the project as non-autoscaling deployments. We have specified that the project will have at least three replicas. There are other things we can specify like **`rctl(8)`** rules, but for simplicity this is sufficient.

```
$ overlord apply -f hello-http.yml
$ overlord get-info -f hello-http.yml -t projects --filter-per-project --use-autoscale-la-
bels
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_19h36m17s
      locked: False
      services:
        - {‘name’: ‘darkhttpd’, ‘status’: 0, ‘jail’: ‘0524bcf91b’}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {‘errlevel’: 0, ‘message’: None, ‘failed’: []}
      last_update: 58.24 seconds
      job_id: 31
      restarted: False
      labels:
       error: False
       message: None
       load-balancer:
         services:
```

```
              darkhttpd:
                error: False
                message: (project:hello-http, service:darkhttpd, backend:web, serve-
rid:0d67d160-61af-4810-b277-5fb9e20da8eb, code:202, transaction_id:baa5b939-f724-4bd3-9d65-
2ef769def3f5, commit:1) server has been successfully added.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: provider
  labels:
    - all
    - provider
    - vm-only
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_20h00m11s
      locked: False
      services:
  - {'name': 'darkhttpd', 'status': 0, 'jail': '2c2d22d2a5'}
up:
  operation: COMPLETED
  output:
   rc: 0
   stdout: {'errlevel': 0, 'message': None, 'failed': []}
  last_update: 4 minutes and 46.3 seconds
  job_id: 6
  restarted: False
  labels:
   error: False
   message: None
   load-balancer:
     services:
              darkhttpd:
                error: False
                message: (project:hello-http, service:darkhttpd, backend:web, serverid:-
fa8f94b1-6b2b-4cb4-a808-e9da46014c86, code:202, transaction_id:6792e6fe-a
778-44a7-b23a-1b2c23fe5904, commit:1) server has been successfully updated.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
```

```
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_20h04m25s
      locked: False
      services:
        - {‘name’: ‘darkhttpd’, ‘status’: 0, ‘jail’: ‘a6549318ce’}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {‘errlevel’: 0, ‘message’: None, ‘failed’: []}
        last_update: 33.34 seconds
        job_id: 21
        restarted: False
        labels:
          error: False
          message: None
          load-balancer:
            services:
              darkhttpd:
                error: False
                message: (project:hello-http, service:darkhttpd, backend:web, serverid:f-
4b9e170-67bb-403e-88da-112c55b45fce, code:202, transaction_id:00e632ce-c215-4784-9e61-
9507d914ba6a, commit:1) server has been successfully updated.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
$ overlord get-info -f hello-http.yml -t autoscale --filter-per-project
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
    - vm-only
  projects:
```

```
    hello-http:
      autoScale:
        last_update: 7.65 seconds
        operation: COMPLETED
        output:
         message: None
$ curl http://revproxy.overlord.lan
Hello, world!
UUID: 08951a86-2aef-4e85-9bfc-7fe68b5cc62d
$ curl http://revproxy.overlord.lan
Hello, world!
UUID: 5a06a89d-6109-438e-bc04-1ef739473994
```

Suppose the service in **provider** is down for any reason.

```
$ overlord get-info -f hello-http.yml -t projects --filter-per-project --use-autoscale-la-
bels
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: None
  labels:
    - all
    - desktop
    - services
  - vm-only
projects:
  hello-http:
    state: DONE
    last_log: 2025-04-23_19h36m17s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '0524bcf91b'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 13 minutes and 37.64 seconds
      job_id: 32
      restarted: False
      labels:
       error: False
       message: None
       load-balancer:
         services:
           darkhttpd:
             error: False
             message: (project:hello-http, service:darkhttpd, backend:web, serve-
```

```
rid:0d67d160-61af-4810-b277-5fb9e20da8eb, code:202, transaction_id:a2ba93d7-6ce6-4a36-aab2-
09be13a00c17, commit:1) server has been successfully updated.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: provider
  labels:
    - all
    - provider
    - vm-only
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_20h11m58s
      locked: False
      services:
        - {'name': 'darkhttpd', 'status': 66, 'jail': '2c2d22d2a5'}
      up:
        operation: COMPLETED
        output:
         rc: 0
         stdout: {'errlevel': 0, 'message': None, 'failed': []}
        last_update: 1 minute and 13.9 seconds
        job_id: 7
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             darkhttpd:
               error: False
               message: (project:hello-http, service:darkhttpd, backend:web, serverid:-
fa8f94b1-6b2b-4cb4-a808-e9da46014c86, code:202, transaction_id:b19e7997-871c-4293-a8a9-
51ce03f2bbaa, commit:1) server has been successfully updated.
          skydns:
            services:
              darkhttpd:
                error: False
                message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
```

```
  labels:
    - all
    - centralita
    - services
  projects:
    hello-http:
      state: DONE
      last_log: 2025-04-23_20h04m25s
      locked: False
      services:
    - {‘name’: ‘darkhttpd’, ‘status’: 0, ‘jail’: ‘a6549318ce’}
  up:
    operation: COMPLETED
    output:
     rc: 0
     stdout: {‘errlevel’: 0, ‘message’: None, ‘failed’: []}
    last_update: 8 minutes and 29.3 seconds
    job_id: 21
    restarted: False
    labels:
     error: False
     message: None
     load-balancer:
            services:
              darkhttpd:
                error: False
                message: (project:hello-http, service:darkhttpd, backend:web, serverid:f-
4b9e170-67bb-403e-88da-112c55b45fce, code:202, transaction_id:00e632ce-c215-4784-9e61-
9507d914ba6a, commit:1) server has been successfully updated.
         skydns:
            services:
              darkhttpd:
                error: False
                message: None
```

Without any intervention, let’s see the magic.

```
datacenter: http://127.0.0.1:8888
entrypoint: main
chain: None
labels:
  - all
  - desktop
  - services
  - vm-only
projects:
  hello-http:
    state: DONE
```

```
    last_log: 2025-04-23_19h36m17s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '0524bcf91b'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 14 minutes and 30.7 seconds
      job_id: 32
      restarted: False
      labels:
       error: False
       message: None
       load-balancer:
         services:
           darkhttpd:
             error: False
             message: (project:hello-http, service:darkhttpd, backend:web, serve-
rid:0d67d160-61af-4810-b277-5fb9e20da8eb, code:202, transaction_id:a2ba93d7-6ce6-4a36-aab2-
09be13a00c17, commit:1) server has been successfully updated.
       skydns:
         services:
           darkhttpd:
             error: False
             message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: provider
  labels:
    - all
    - provider
    - vm-only
  projects:
    hello-http:
      state: DONE
    last_log: 2025-04-23_20h13m37s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': '2c2d22d2a5'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 27.47 seconds
```

```
        job_id: 8
        restarted: False
        labels:
         error: False
         message: None
         load-balancer:
           services:
             darkhttpd:
               error: False
               message: (project:hello-http, service:darkhttpd, backend:web, serverid:-
fa8f94b1-6b2b-4cb4-a808-e9da46014c86, code:202, transaction_id:98ca3a6a-65e4-450b-a4c3-
4f135e36be37, commit:1) server has been successfully updated.
         skydns:
           services:
             darkhttpd:
               error: False
               message: None
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
projects:
  hello-http:
    state: DONE
    last_log: 2025-04-23_20h04m25s
    locked: False
    services:
      - {'name': 'darkhttpd', 'status': 0, 'jail': 'a6549318ce'}
    up:
      operation: COMPLETED
      output:
       rc: 0
       stdout: {'errlevel': 0, 'message': None, 'failed': []}
      last_update: 9 minutes and 22.37 seconds
      job_id: 21
      restarted: False
      labels:
       error: False
       message: None
       load-balancer:
         services:
           darkhttpd:
             error: False
             message: (project:hello-http, service:darkhttpd, backend:web, serverid:f-
```

```
4b9e170-67bb-403e-88da-112c55b45fce, code:202, transaction_id:00e632ce-c215-4784-9e61-
9507d914ba6a, commit:1) server has been successfully updated.
        skydns:
          services:
            darkhttpd:
              error: False
              message: None
```

The service is alive again.

Info, Metrics, and more...

Thanks to AppJail, a lot of information can be obtained from jails. Overlord has a special deployment called **`readOnly`** that can be perfectly combined with the **`get-info`** command.

info.yml:

```
kind: readOnly
datacenters:
  main:
    entrypoint: !ENV '${ENTRYPOINT}'
    access_token: !ENV '${TOKEN}'
deployIn:
  labels:
    - all
$ overlord get-info -f info.yml -t projects --filter adguardhome
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
  projects:
    adguardhome:
      state: DONE
      last_log: 2025-04-07_17h32m40s
      locked: False
      services:
        - {'name': 'server', 'status': 0, 'jail': '2a67806954'}
$ overlord get-info -f info.yml -t jails --filter 2a67806954
datacenter: http://127.0.0.1:8888
  entrypoint: main
  chain: centralita
  labels:
    - all
    - centralita
    - services
  jails:
    2a67806954:
```

```
      stats:
        cputime: 91
        datasize: 8400896 (8.01 MiB)
        stacksize: 0 (0 bytes)
        coredumpsize: 0 (0 bytes)
        memoryuse: 75104256 (71.62 MiB)
        memorylocked: 0 (0 bytes)
        maxproc: 4
        openfiles: 296
        vmemoryuse: 1367982080 (1.27 GiB)
        pseudoterminals: 0
        swapuse: 0 (0 bytes)
        nthr: 13
        msgqqueued: 0
        msgqsize: 0
        nmsgq: 0
        nsem: 0
        nsemop: 0
        nshm: 0
        shmsize: 0 (0 bytes)
        wallclock: 363548
        pcpu: 0
        readbps: 0 (0 bytes)
        writebps: 0 (0 bytes)
        readiops: 0
        writeiops: 0
      info:
        name: 2a67806954
        network_ip4: 10.0.0.3
        ports: 53/tcp,53/udp,53/tcp,53/udp
        status: UP
        type: thin
        version: 14.2-RELEASE
      cpuset: 0, 1
      expose:
        - {‘enabled’: ‘1’, ‘name’: None, ‘network_name’: ‘ajnet’, ‘hport’: ‘53’, ‘jport’:
‘53’, ‘protocol’: ‘udp’, ‘ext_if’: ‘tailscale0’, ‘on_if’: ‘tailscale0’, ‘nro’: ‘3’}
        - {‘enabled’: ‘1’, ‘name’: None, ‘network_name’: ‘ajnet’, ‘hport’: ‘53’, ‘jport’:
‘53’, ‘protocol’: ‘tcp’, ‘ext_if’: ‘jext’, ‘on_if’: ‘jext’, ‘nro’: ‘0’}
        - {‘enabled’: ‘1’, ‘name’: None, ‘network_name’: ‘ajnet’, ‘hport’: ‘53’, ‘jport’:
‘53’, ‘protocol’: ‘tcp’, ‘ext_if’: ‘tailscale0’, ‘on_if’: ‘tailscale0’, ‘nro’: ‘2’}
        - {‘enabled’: ‘1’, ‘name’: None, ‘network_name’: ‘ajnet’, ‘hport’: ‘53’, ‘jport’:
‘53’, ‘protocol’: ‘udp’, ‘ext_if’: ‘jext’, ‘on_if’: ‘jext’, ‘nro’: ‘1’}
      fstab:
        - {‘enabled’: ‘1’, ‘name’: None, ‘device’: ‘/var/appjail-volumes/adguardhome/db’,
‘mountpoint’: ‘adguardhome-db’, ‘type’: ‘<volumefs>’, ‘options’: ‘rw’, ‘dump’: ‘0’, ‘pass’:
None, ‘nro’: ‘0’}
```

```
      labels:
        - {‘value’: ‘1’, ‘name’: ‘overlord.skydns’}
        - {‘value’: ‘adguardhome’, ‘name’: ‘appjail.dns.alt-name’}
        - {‘value’: ‘tailscale0’, ‘name’: ‘overlord.skydns.interface’}
        - {‘value’: ‘adguardhome’, ‘name’: ‘overlord.skydns.group’}
      nat:
        - {‘rule’: ‘nat on "jext" from 10.0.0.3 to any -> ("jext:0")’, ‘network’: ‘ajnet’}
      volumes:
        - {‘mountpoint’: ‘usr/local/etc/AdGuardHome.yaml’, ‘type’: ‘<pseudofs>’, ‘uid’:
None, ‘gid’: None, ‘perm’: ‘644’, ‘name’: ‘adguardhome-conf’}
        - {‘mountpoint’: ‘/var/db/adguardhome’, ‘type’: ‘<pseudofs>’, ‘uid’: None, ‘gid’:
None, ‘perm’: ‘750’, ‘name’: ‘adguardhome-db’}
```

Future Work

There are more things Overlord can do for you than this document provides, see the Wiki for more examples.

Overlord is a recent project, there is a lot of room for improvement and future features will be added to improve its usability. If you would like to support the project, please consider donating.

JESÚS DANIEL COLMENARES OVIEDO is a sysadmin, developer and ports committer and is the creator of AppJail and all the projects related to it (Director, Makejails, Reproduce, Overlord, etc.).