PostFinance

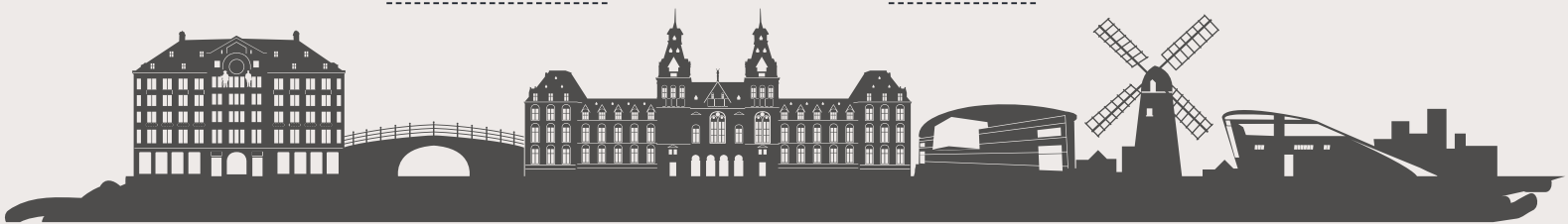


TOPF

Open-Sourcing PostFinance's Tool to Migrate and Manage Talos Linux Clusters

Clément Nussbaumer clement.n8r.ch

Sebastian Stephan stephan.li



Kubernetes at PostFinance

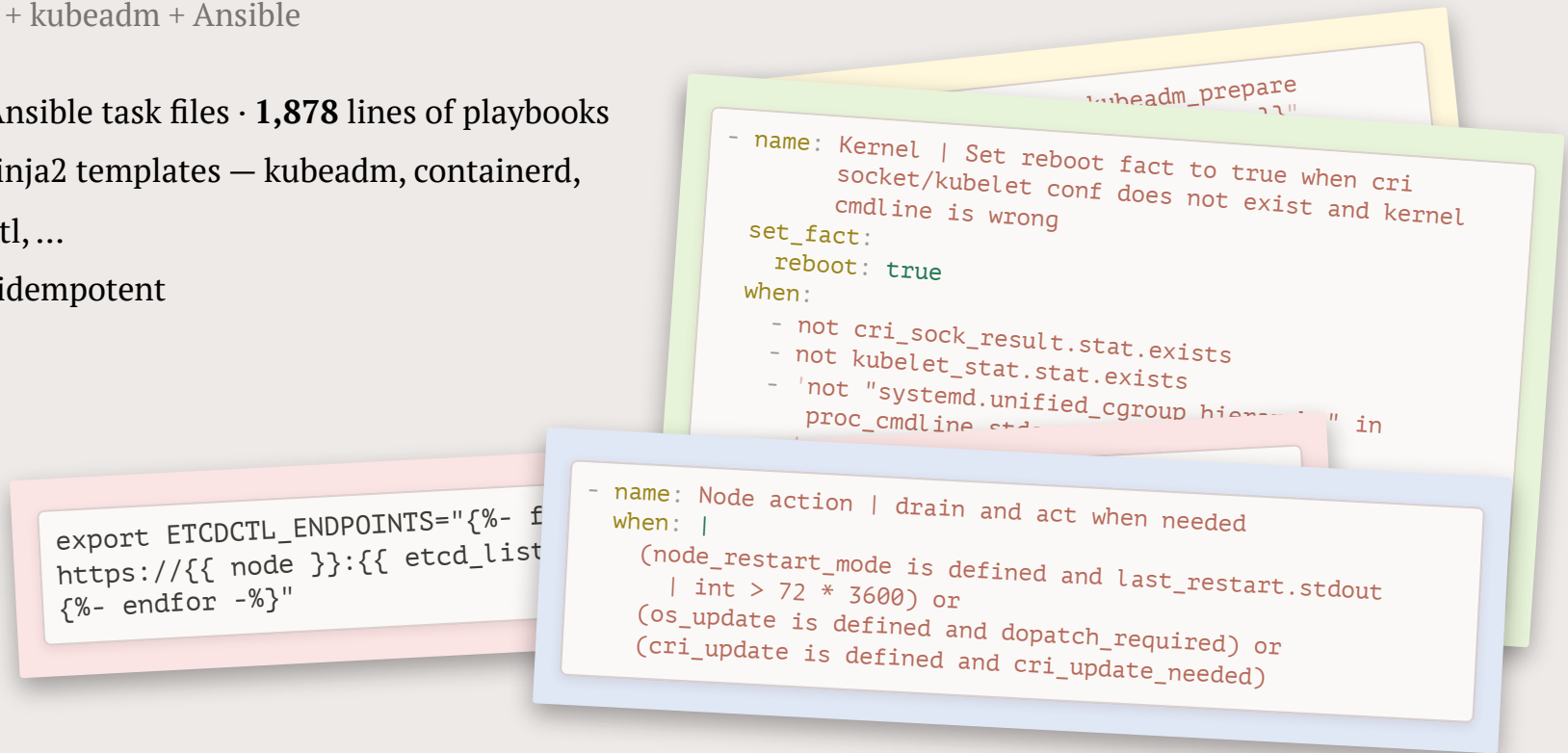
- Systemic Swiss bank
- ~35 **Talos Linux** Kubernetes clusters
- ~55 TiB Memory, largest cluster 133 Nodes
- Air-gapped environment
- 2 on-prem datacenters - VSphere Virtualization
- Chaos Monkey on all clusters 🐒 (postfinance/chaosmonkey)
- Oldest cluster:

```
k get ns kube-system
NAME          STATUS   AGE
kube-system   Active   7y83d
```

Starting Point

Debian + kubeadm + Ansible

- **32** Ansible task files · **1,878** lines of playbooks
- **20** Jinja2 templates — kubeadm, containerd, sysctl, …
- not idempotent

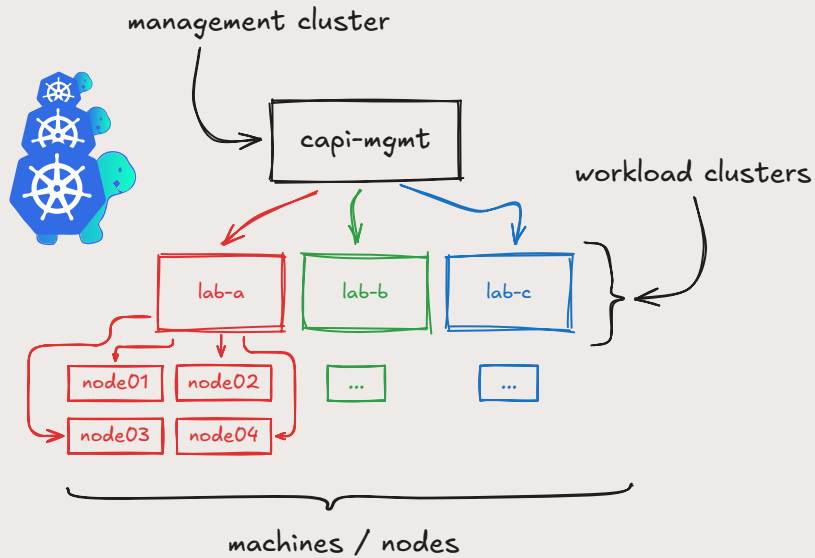


Our initial plan: ClusterAPI + Talos

- Declarative cluster lifecycle
- CRDs for clusters/nodes

but...

- Siderolabs shifted away from CAPI[1]
- Kubernetes to manage Kubernetes
- CAPI complexity vs simplicity goal



1. https://www.siderolabs.com/blog/talos-linux-capi-community-maintenance

Our Options

```
- name: Apply Talos machine config
  ansible.builtin.command: |
    talosctl apply -f {{ config }}
```

Talhelper

?

Omni

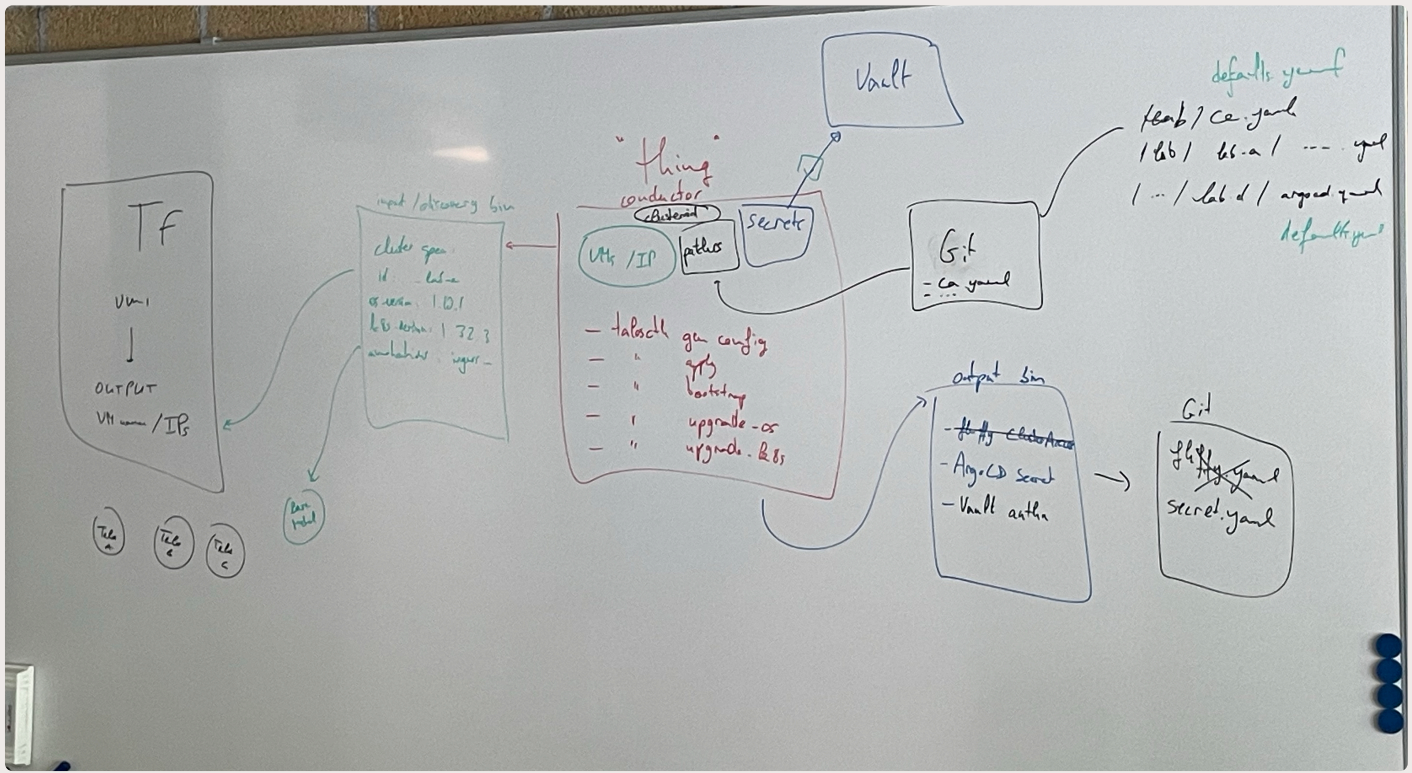
manual — config generator — turnkey

> **Note**
>
> This project is now archived and abandoned. I suggest people who depend on this tool to migrate to other similar tools like topf or talstomize



Archived on **Aug 26th, 2026**

Thing



The original whiteboard sketch of what would become TOPF

TOPF

Talos Orchestrator by PostFinance

Not an operator.
Not a controller.
A CLI tool.

*Topf is also German for a **(plant) pot***



Hello World

```
├── all/
│   └── • disk.yaml
└── • topf.yaml
```

Patches are just files with some Talos config, placed in `all/`.

• topf.yaml

```yaml
clusterName: my-cluster
clusterEndpoint: https://192.168.1.11:6443
nodes:
  - host: node1
    ip: 192.168.1.11
    role: control-plane
```

• all/disk.yaml — (could be any name)

```yaml
# docs.siderolabs.com/talos/v1.13/reference/configuratior
machine:
  install:
    disk: /dev/sda
```

```
$ topf apply --auto-bootstrap
Do you want to apply the above changes to node1 (Mode: NO_REBOOT)? [y/n]: y
13:58:21 INFO applied machine config command=apply node=node1 mode=NO_REBOOT
13:59:11 INFO etcd bootstrap completed successfully command=apply
```

Patches per Role

```
├── all/
│   └── disk.yaml
├── control-plane/
│   └── ● vip.yaml
├── worker/
│   └── ● kernel-params.yaml
└── topf.yaml
```

● control-plane/vip.yaml

```yaml
apiVersion: v1alpha1
kind: Layer2VIPConfig
name: 192.168.1.100
link: eth0
```

● worker/kernel-params.yaml

```yaml
machine:
  sysctls:
    vm.max_map_count: "1048576"
```

Patches in `control-plane/` and `worker/` folders only apply to nodes with that role.

Why not just `talosctl`?

Possible, but not as nice

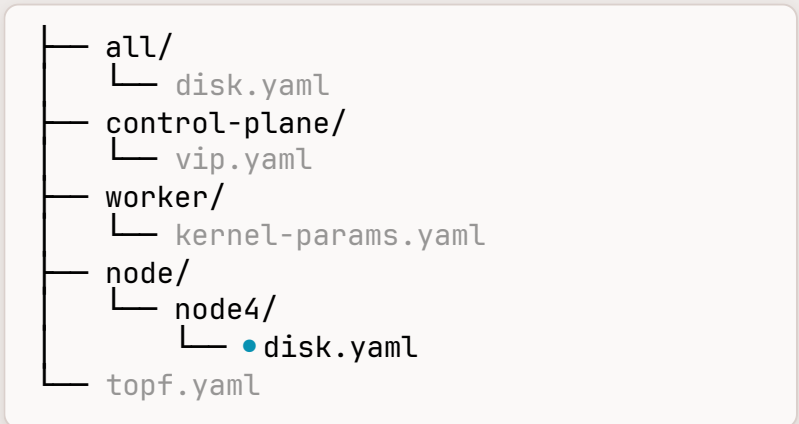
```
$ talosctl gen secrets -o secrets.yaml

$ talosctl gen config my-cluster https://192.168.1.100:6443 \
    --with-secrets secrets.yaml \
    --config-patch @disk.yaml \
    --config-patch-control-plane @vip.yaml \
    --config-patch-worker @kernel-params.yaml \
    --output-dir ./generated

$ talosctl apply-config --insecure -n 192.168.1.11 -f ./generated/controlplane.yaml
$ talosctl apply-config --insecure -n 192.168.1.12 -f ./generated/controlplane.yaml
$ talosctl apply-config --insecure -n 192.168.1.13 -f ./generated/controlplane.yaml
$ talosctl apply-config --insecure -n 192.168.1.14 -f ./generated/worker.yaml
$ talosctl apply-config --insecure -n 192.168.1.15 -f ./generated/worker.yaml

$ talosctl bootstrap -n 192.168.1.11 --talosconfig ./generated/talosconfig
```

Node Specific Patches



- `node/node4/disk.yaml`

```yaml
# node4 has different hardware → override
machine:
  install:
    disk: /dev/nvme0n1
```

node4 gets patches from `all/`, `worker/` and `nodes/node4/`.

Patches are overlayed from least specific to specific, so things can be overridden.

Templating

```
├── all/
│   └── • hostname.yaml.tpl
└── • topf.yaml
```

Use any data from `topf.yaml`.

• `all/hostname.yaml.tpl`

```
apiVersion: v1alpha1
kind: HostnameConfig
hostname: {{ .Node.Host }}
```

using any data from...

• `topf.yaml`

```
...
nodes:
  - host: node1 # <-- for example from here
    ip: 192.168.1.11
    role: control-plane
```

Patches using go templating must end in `.tpl`.

Templating: Node Data

```
├── all/
│   ├── • disk.yaml.tpl
├── worker/
│   ├── • gpu.yaml.tpl
└── • topf.yaml
```

Add arbitrary data to your nodes.

• `topf.yaml`

```yaml
nodes:
  - host: node4
    data: # <-- arbitrary k/v data
      disk: /dev/sda
      gpu: true
```

• `all/disk.yaml.tpl`

```
machine:
  install:
    disk: {{ .Node.Data.disk }}
```

• `worker/gpu.yaml.tpl`

```
{{ if index .Node.Data "gpu" }}
machine:
  kernel:
    modules:
      - name: nvidia
{{ end }}
```

Upgrading Talos

```
├── all/
│   └── • disk.yaml
└── • topf.yaml
```

Track the desired Talos version in `topf.yaml`, use `topf upgrade` to upgrade.

• topf.yaml

```yaml
clusterName: my-cluster
clusterEndpoint: https://192.168.1.11:6443
talosVersion: 1.13.8
nodes:
  - host: node1
    ip: 192.168.1.11
    role: control-plane
```

```
$ topf upgrade
level=INFO msg="upgrade required" version_actual=1.13.7 version_desired=1.13.8
Do you want to upgrade node node1 with installer factory.talos.dev..:v1.13.8? [y/n]: y
level=INFO msg="upgrade artifacts installed"
level=INFO msg="draining kubernetes node"
level=INFO msg="reboot initiated"
level=INFO msg="machine ready, waiting for stabilization..."
level=INFO msg="kubernetes node uncordoned"
```

Secrets Management

```
├── all/
│   └── • wireguard.yaml
├── secrets.yaml ⟵ auto generated
└── topf.yaml
```

- SOPS[1] support.
- Vals[2] support.
- Custom logic: `secretsProvider`
- Secrets are redacted in dry-run outputs
- Secrets everywhere

• wireguard.yaml (Vals ref)

```yaml
kind: WireguardConfig
name: wg.int
privateKey: ref+gitlab://gitlab.com/projects/42/privkey
```



1. https://github.com/getsops/sops
2. https://github.com/helmfile/vals



Live Demo

github.com/postfinance/topf

Migration

1. match config
2. import PKI material
3. add control-plane nodes (one by one)
4. add worker nodes
5. remove old kubeadm nodes



Day 2000 Migration.pdf

YouTube recording

Nodes Provider

Resolve node lists and secrets at runtime from external sources.

instead of a static list…

```yaml
# topf.yaml
nodes:
  - host: node1
    ip: 192.168.1.11
    role: control-plane
```

…point at a binary:

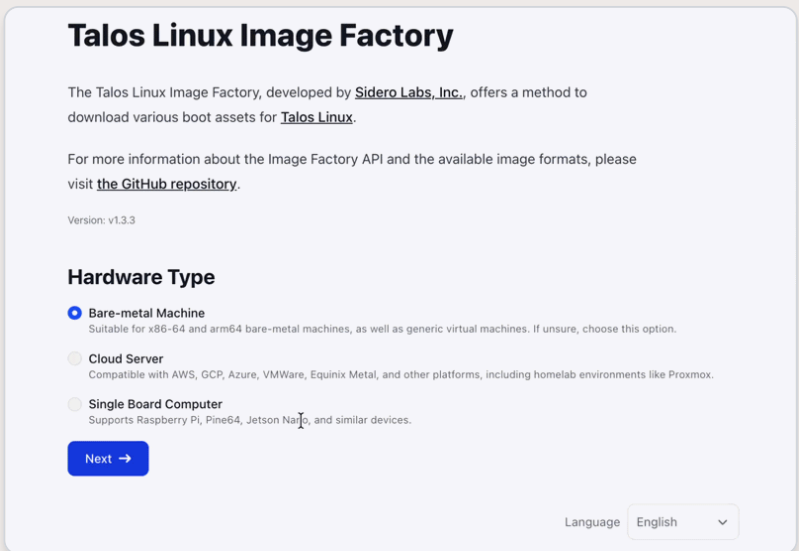
```yaml
# topf.yaml
nodesProvider: ./nodes-from-terraform.sh
```

e.g. read nodes from a Terraform output, cloud API, or inventory

Schematic Resolution

stop juggling Talos Factory [1] image hashes

Before — the Factory UI



After — a manifest in Git

- `manifest.yaml.tpl`

```yaml
customization:
  extraKernelArgs:
    - ip={{ .Node.IP }}::10.0.0.1:255.255.255.0::eth0
  systemExtensions:
    officialExtensions:
      - siderolabs/vmtoolsd-guest-agent
```

- `topf.yaml`

```yaml
schematicId: "@manifest.yaml.tpl"
```

1. https://factory.talos.dev/

TOPF in GitLab CI

one pipeline per cluster — GitOps for day-2 ops

```yaml
# .gitlab-ci.yml
default:
  before_script:
    - git clone --branch "$CONFIG_REF" "https://$CONFIG_REPO" .

dry-run:
  script: topf apply --dry-run

apply:
  script: topf apply --confirm=false
  when: manual

upgrade:
  script: topf upgrade --confirm=false
  when: manual
```

```
$ topf apply --dry-run

  node2  (Mode: NO_REBOOT)
  machine:
    install:
-     disk: /dev/sda
+     disk: /dev/nvme0n1
    kernel:
      modules:
+       - name: nvidia

  1 node changed — exit code 2
```

Give it a Try

- Kubernetes with Talos is so easy, you might not need a managed Service.
- Play around with Talos and TOPF in your Homelab!





PostFinance



Thank you!

postfinance/topf talos.dev

clement.n8r.ch stephan.li

Questions?

Multiple Nodes

```
├── all/
│   └── • disk.yaml
└── • topf.yaml
```

Run `topf apply` to join those nodes.

• topf.yaml

```yaml
clusterName: my-cluster
clusterEndpoint: https://192.168.1.11:6443
nodes:
  - host: node1
    ip: 192.168.1.11
    role: control-plane
  - host: node2
    ip: 192.168.1.12
    role: control-plane
  - host: node3
    ip: 192.168.1.13
    role: control-plane
  - host: node4
    ip: 192.168.1.14
    role: worker
  - host: node5
    ip: 192.168.1.15
    role: worker
```