Immutable Infrastructure with Flatcar Container Linux

FOSDEM’22 Infra Management devroom | 2022-02-05
Hi, I'm Kai

Kai Lüke
Software Engineer, Microsoft
Working on
Flatcar Container Linux

Github: pothos
Email: kailuke@microsoft.com
Immutable Infrastructure
»Immutable Infrastructure«

- Paradigm to reprovision servers instead of managing their configuration after provisioning

- Pros:
  - Reproducible and consistent configuration, e.g., matching a git repository

- Cons:
  - Reprovisioning takes longer
  - Data gets lost (local application data, logs, SSH host keys, ...)
  - New IP address depending on cloud environment
Flatcar Container Linux
(Fork/Continuation of CoreOS Container Linux)
Why Flatcar Container Linux?

- **Minimal distribution for containers**
  - Reduced dependencies
  - Less base software to manage
  - Reduced attack surface area

- **Secure, immutable file system**
  - Read-only /usr partition
  - No package installation or modification of base OS files
  - Removes entire category of security threats (e.g., runc vulnerability CVE-2019-5736)

- **Automated, streamlined updates**
  - Easily apply all latest security patches
  - Atomic updates and rollbacks
  - Co-ordinated with Kubernetes control plane (update operator)

- **Declarative provisioning**
  - First boot setup from declarative configuration
  - Immutable infrastructure (no custom per-node changes during production)
  - Repeatable deployment
Ignition Config

- JSON format
- Declaration of files, systemd units, networks, users, filesystems, and partitions
- Referencing data from external resources
- Applied from initramfs (first-boot flag file for GRUB sets kernel parameter)
- Contrast to cloud-init which runs after the initramfs, and on every boot
Container Linux Config (CLC)

- Friendlier YAML format with extras (octal permissions, variables for metadata)
- Transpiled to Ignition JSON through transpiler “ct”
  
  ```
  cat cl.yaml | docker run --rm -i quay.io/coreos/ct:latest-dev > ignition.json
  ./flatcar_production_qemu.sh  -i ignition.json
  ```

- Spec: [flatcar.org/docs/latest/provisioning/config-transpiler/configuration/](https://flatcar.org/docs/latest/provisioning/config-transpiler/configuration/)


Container Linux Config Example

```yaml
storage:
  files:
    - path: /etc/some.conf
      filesystem: root
      mode: 0644
      contents:
        inline: |
          A=a
          B=b

Or with remote instead of 'inline' content:

remote:
  url: ...
```
Terraform
Terraform and Ignition

- Ignition config is set through instance user-data attribute (no need for the SSH provisioner)
- `terraform-ct-provider` to transpilde CLC to Ignition, often combined with the template-provider
- Or: `terraform-ignition-provider` (1.x) to assemble Ignition JSON from HCL
Terraform Example

resource "digitalocean_droplet" "machine" {
  for_each = toset(var.machines)
  image    = digitalocean_custom_image.flatcar.id
  user_data = data.ct_config.machine-ignition[each.key].rendered
}
data "ct_config" "machine-ignition" {
  for_each = toset(var.machines)
  content  = data.template_file.machine-cl-config[each.key].rendered
}
data "template_file" "machine-cl-config" {
  for_each = toset(var.machines)
  template = file("${path.module}/cl/machine-${each.key}.yamltmpl")
  vars     = { something = var.something }
Configuration Changes and Instance Lifecycle
Instance Lifecycle with Replacement

- Instance replacement for user-data changes can be disruptive: downtime, data transfer needed, slow, maybe IP address changes, too, etc.

- Workarounds: create_before_destroy, backups or only using external storage, last resort: delaying replacement with ignore_changes (→ config drift)
Instance Lifecycle without Replacement?

- Ansible: Flatcar bootstrap with pypy in home folder
- Not really immutable infra without reprovisioning as old files may be lingering around → config drift
- Also, half Ignition, half Ansible gets messy
- Can't we just somehow rerun Ignition?
Instance reprovisioning with Ignition

- touch /boot/flatcar/first_boot is not enough:
  - Must remove old versions of config files
  - Must remove /etc/machine-id to trigger systemd first-boot semantics for preset evaluation
- Big hammer: Reformat rootfs through Ignition (use other disks for persistent data)
Reformat with Ignition

CLC snippet:

```yaml
filesystems:
  - name: root
    mount:
      device: /dev/disk/by-label/ROOT
      format: ext4
      wipe_filesystem: true
      label: ROOT
```

Quite fast, preserves the IP address but still loses most local data...
More problems: User-data Updates

- Terraform cloud providers and cloud APIs in general make it hard to update user data in-place
- Workaround: Indirection through cloud bucket/blob storage like S3/GCS, let instance user data point to the storage object (stable reference), and update the config in the storage object
Ignition Config Indirection

- Ignition Config
  - Replace Directive (use config from URL)

Stable URL

Storage Object
  - Final Ignition Config

Config Updates
Ignition Config in Storage Object

Point to storage URL:

```
ignition: → config: → replace: → source: → URL
```

Terraform Example:

```
resource ... {
    user_data = "{
        "ignition": {
            "version": "2.3.0",
            "config": {
                "replace": {
                    "source": "s3://${aws_s3_bucket_object.object.bucket}/${aws_s3_bucket_object.object.id}"
                }
            }
        }
    }
}
```

```
resource "aws_s3_bucket_object" "object" {
    content = data.ct_config.machine-ignition.rendered
}
```
Trigger Reboot and Ignition Rerun

Terraform null resource:

```hcl
resource "null_resource" "reboot-when-ignition-changes" {
  triggers = {
    config = data.ct_config.machine-ignition.rendered
  }
  depends_on = [aws_s3_bucket_object.object]
  provisioner "local-exec" {
    command = "[... SSH cmd to create first_boot file and reboot ...]"
  }
}
```

Not the nicest workaround, some git-ops like daemon on the instance to check the storage object can work, too
Idea: Teach Ignition to Preserve State

- Instead of discarding the whole rootfs, let's improve Ignition to be able to selectively keep wanted files.
- A Draft PR implements this, e.g.:

```
"cleanExcept":
  ["/etc/ssh/ssh_host_.*", "/var/log", ...]
```
- Specify app data or container image folders.
- The machine ID can be preserved with `systemd.machine_id=...` in `grub.cfg`.
Proof-of-Concept Demo

- Using the qemu helper script instead of Terraform: asciinema.org/a/462614
Results with the Proof-of-Concept

- Fast reprovisioning, preserves IP address and all local data (SSH host keys, system logs, application data, as needed)
- Declarative config management without drift
- Since only SSH is needed it's even viable for bare metal lacking IPMI automation (place/update config.ign file on OEM partition)
- Some workarounds were needed, though
Summary

- Immutable Infra possible even for stateful systems
- Flatcar Container Linux already simplifies OS maintenance through immutable A/B updates
- Choose your strategy for user-data config changes
- Terraform examples on GitHub: flatcar-linux/flatcar-terraform
Thank you!

Kai Lüke

Github: pothos
Email: kailuke@microsoft.com

Flatcar Container Linux

Website: flatcar.org

GitHub Repos: flatcar-linux

Terraform Examples: flatcar-linux/flatcar-terraform