Ignition and Butane

OKD Foundations Video Series

Agenda

- What is Ignition?
- What is Butane?
- Managing configs with the Machine Config Operator
- Using Butane to write MachineConfigs

What is Ignition?

Ignition

- **Automate** provisioning to configure a node
  - Runs **exactly once**, on **first boot**, during the initramfs stage
- Any logic for machine lifetime is **encoded** in the config
  - Very easy to automatically **re-provision** nodes
- **Same starting point** whether on bare metal or cloud
  - Use Ignition **everywhere** as opposed to kickstart for bare metal and cloud-init for cloud

Ignition configs

- **Declarative** JSON documents provided via user data
- Can write files and systemd units, create users and groups, create partition disks, format filesystems, etc.
- **If provisioning fails, the boot fails** (no half provisioned systems)
- Ignition configs are **machine-friendly** (JSON)

```
{
 "ignition": {
  "config": {},
  "timeouts": {},
  "version": "3.0.0"
 },
 "passwd": {
  "users": [
   {
    "name": "core",
    "passwordHash":
      "$6$43y3tkl...",
    "sshAuthorizedKeys": [
     "ssh-ed25519 ..."
    ]
   }
  ]
 },
 "storage": {
  ...
 },
 "systemd": {
  ...
 }
}
```

Who uses Ignition?

- **Fedora CoreOS (FCOS) in OKD**
- **CentOS Stream CoreOS (SCOS) in OKD-SCOS**
- RHEL CoreOS (RHCOS) in OpenShift
- Fedora IoT and RHEL for Edge
- Flatcar Container Linux
- openSUSE MicroOS

What is Butane?

Butane configs

- **Butane** is a configuration transpiler
- **Converts** Butane configs to Ignition configs or MachineConfigs
- Butane configs are **Human friendly** (YAML)
- Ignition semantics, plus **sugar** for common operations
- Transpiler catches common errors at **build time**

```
variant: fcos
version: 1.5.0
passwd:
 users:
   - name: core
     ssh_authorized_keys:
       - ssh-ed25519 ...
systemd:
 units:
   - name: myscript.service
     enabled: false
     contents: |
       ...
storage:
 files:
   - path: /etc/chrony.conf
     overwrite: yes
     mode: 0644
     contents:
       local: chrony.conf
   - path: /etc/containers/...
     contents:
       local: foo.container
```

Example: Encrypted storage via LUKS

- Unlock via TPM2 or a Tang server (via Clevis)
- Includes support for the **root partition**

```
# LUKS for / using TPM2
variant: fcos
version: 1.5.0
boot_device:
  luks:
    tpm2: true
```

```
# LUKS for another device
variant: fcos
version: 1.5.0
storage:
  luks:
    - name: data
      device: /dev/vdb
      clevis:
        tpm2: true
  filesystems:
    - path: /var/lib/data
      device: /dev/mapper/data
      format: xfs
      label: DATA
      with_mount_unit: true
```

https://docs.fedoraproject.org/en-US/fedora-coreos/storage/#_encrypted_storage_luks

Example: RAID support

- Setup any RAID level (0, 1, 5, etc.) on first boot **via Ignition**
- Mirrors EFI System Partition (ESP) & BIOS bootloader
- Side effect: ESP no longer mounted (empty /boot/efi)

```
# Mirror boot disk with RAID1
variant: fcos
version: 1.5.0
boot_device:
  mirror:
    devices:
      - /dev/sda
      - /dev/sdb
```

```
# Move / to RAID0
variant: fcos
version: 1.5.0
storage:
  raid:
    - name: myroot
      level: raid0
      devices:
        - /dev/disk/by-id/virtio-disk1
        - /dev/disk/by-id/virtio-disk2
  filesystems:
    - device: /dev/md/myroot
      format: xfs
      wipe_filesystem: true
      label: root
```

https://docs.fedoraproject.org/en-US/fedora-coreos/storage/#_reconfiguring_the_root_filesystem

Managing node configs with the Machine Config Operator (MCO)

Machine Config Operator

- In OKD, changes are managed by **operators**
- The **Machine Config Operator** (MCO) manages changes to node configurations which are stored as **MachineConfigs**
- Those configs are provided to **new nodes** as Ignition configs
- **Existing nodes** are updated in-place

MachineConfigs in OKD

- The initial **MachineConfigs** are generated by the OKD installer
- They are then translated to **Ignition configs** for the bootstrap, control-plane and worker nodes
- Users can **customize** their OKD installation by providing their own MachineConfigs
- Users can write Butane configs, which are then validated and translated into **MachineConfigs**
- Butane config ⇨ MachineConfig ⇨ Ignition config

Using Butane to write MachineConfigs

Example: Encrypted, RAID storage via LUKS

$ cat worker-storage.bu

```
variant: openshift
version: 4.13.0
metadata:
  name: worker-storage
  labels: machineconfiguration.openshift.io/role: worker
boot_device:
  luks:
    tpm2: true
  mirror:
    devices:
      - /dev/sda
      - /dev/sdb
```

$ butane worker-storage.bu -o <installation_directory>/openshift/99-worker-storage.yaml

https://docs.okd.io/4.13/installing/install_config/installing-customizing.html#installation-special-config-mirrored-disk_installing-customizing

Example: Configuring chrony time service

$ cat worker-chrony.bu

```
variant: openshift
version: 4.13.0
metadata:
  name: worker-storage
  labels: machineconfiguration.openshift.io/role: worker
storage:
  files:
  - path: /etc/chrony.conf
    mode: 0644
    overwrite: true
    contents:
      inline: |
        pool 0.rhel.pool.ntp.org iburst
        driftfile /var/lib/chrony/drift
        makestep 1.0 3
        rtcsync
        logdir /var/log/chrony
```

$ butane worker-chrony.bu -o 99-worker-chrony.yaml
$ oc apply -f ./99-worker-chrony.yaml

https://docs.okd.io/4.13/post_installation_configuration/machine-configuration-tasks.html

Get involved!

OKD: https://www.okd.io/

Fedora CoreOS:

https://fedoraproject.org/coreos/