Ignite Documentation

Ignite development team

Dec 10, 2019

TABLE OF CONTENTS:

In this folder you can read more about how to use Ignite, and how it works:

CHAPTER

ONE

IGNITE INSTALLATION GUIDE

This guide describes the installation and uninstallation process of Ignite.

1.1 System requirements

Ignite runs on most Intel, AMD or ARM (AArch64) based `linux/amd64` systems with `KVM` support. See the full CPU support table in *dependencies.md* for more information.

See *cloudprovider.md* for guidance on running Ignite on various cloud providers and suitable instances that you could use.

NOTE: You do **not** need to install any "traditional" QEMU/KVM packages, as long as there is virtualization support in the CPU and kernel it works.

See *dependencies.md* for needed dependencies.

1.1.1 Checking for KVM support

Please read *dependencies.md* for the full reference, but if you quickly want to check if your CPU and kernel supports virtualization, run these commands:

```
$ lscpu | grep Virtualization
Virtualization:      VT-x

$ lsmod | grep kvm
kvm_intel             200704  0
kvm                   593920  1 kvm_intel
```

Alternatively, on Ubuntu-like systems there's a tool called `kvm-ok` in the `cpu-checker` package. Check for KVM support using `kvm-ok`:

```
$ sudo apt-get update && sudo apt-get install -y cpu-checker
...
$ kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

With this kind of output, you're ready to go!

1.2 Installing dependencies

Ignite has a few dependencies (read more in this *doc*). Install them on Ubuntu/CentOS like this:(Ignite does not depend on docker package version. If you already installed docker-ce, you don't need to replace it to docker.io.)

Ubuntu:

```
apt-get update && apt-get install -y --no-install-recommends dmsetup openssh-client␣
↪git binutils
which containerd || apt-get install -y --no-install-recommends containerd
    # Install containerd if it's not present -- prevents breaking docker-ce␣
↪installations
```

CentOS:

```
yum install -y e2fsprogs openssh-clients git
which containerd || ( yum-config-manager --add-repo https://download.docker.com/linux/
↪centos/docker-ce.repo && yum install -y containerd.io )
    # Install containerd if it's not present
```

1.2.1 CNI Plugins

Install the CNI binaries like this:

```
export CNI_VERSION=v0.8.2
export ARCH=$([ $(uname -m) = "x86_64" ] && echo amd64 || echo arm64)
mkdir -p /opt/cni/bin
curl -sSL https://github.com/containernetworking/plugins/releases/download/${CNI_
↪VERSION}/cni-plugins-linux-${ARCH}-${CNI_VERSION}.tgz | tar -xz -C /opt/cni/bin
```

Note that the SSH and Git packages are optional; they are only needed if you use the `ignite ssh` and/or `ignite gitops` commands.

1.3 Downloading the binary

Ignite is a currently a single binary application. To install it, download the binary from the GitHub releases page, save it as `/usr/local/bin/ignite` and make it executable.

To install Ignite from the command line, follow these steps:

```
export VERSION=v0.6.2
export GOARCH=$(go env GOARCH 2>/dev/null || echo "amd64")

for binary in ignite ignited; do
    echo "Installing ${binary}..."
    curl -sfLo ${binary} https://github.com/weaveworks/ignite/releases/download/$
↪{VERSION}/${binary}-${GOARCH}
    chmod +x ${binary}
    sudo mv ${binary} /usr/local/bin
done
```

Ignite uses semantic versioning, select the version to be installed by changing the `VERSION` environment variable.

1.4 Verifying the installation

If the installation was successful, the `ignite` command should now be available:

```
$ ignite version
Ignite version: version.Info{Major:"0", Minor:"6", GitVersion:"v0.6.2", GitCommit:"...
↪", GitTreeState:"clean", BuildDate:"...", GoVersion:"...", Compiler:"gc", Platform:
↪"linux/amd64"}
Firecracker version: v0.18.0
Runtime: containerd
```

Now you can continue with the *Getting Started Walkthrough*.

1.5 Removing the installation

To completely remove the Ignite installation, execute the following as root:

```
# Force-remove all running VMs
ignite rm -f $(ignite ps -aq)
# Remove the data directory
rm -r /var/lib/firecracker
# Remove the ignite and ignited binaries
rm /usr/local/bin/ignite{,d}
```

CHAPTER

TWO

REQUIREMENTS AND DEPENDENCIES

2.1 Virtualization features

Firecracker by design only supports emulating 4 devices:

- `virtio-block`
- `virtio-net`
- a serial console
- a 1-button keyboard used only to stop the microVM (invoked with `reboot`)

Everything apart from above, is not supported, and out of scope.

2.2 Host Requirements

- A host running Linux 4.14 or newer
- `sysctl net.ipv4.ip_forward=1`
- loaded kernel loop module: `modprobe -v loop`
- Optional: `sysctl net.bridge.bridge-nf-call-iptables=0`
- One of the following CPUs:

2.3 Guest Requirements

- A Linux kernel 4.14 or newer
- Kernel config:
  - `CONFIG_VIRTIO_BLK=y` (mandatory)
  - `CONFIG_VIRTIO_NET=y` (mandatory)
  - `CONFIG_KEYBOARD_ATKBD=y` (optional but recommended)
  - `CONFIG_SERIO_I8042=y` (optional but recommended)

2.4 Ignite on-host dependencies

Ignite shells out to a few dependencies on the host. With time, we aim to eliminate as many of these as possible.

2.4.1 Container Runtime

- `containerd` for managing the containers Ignite uses (default, preferred)
  - Ubuntu package: `containerd`
  - CentOS package: `containerd.io`
    - From docker's repositories: `yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo`
- `docker` for managing the containers Ignite uses (also installs `containerd` automatically)
  - Ubuntu package: `docker.io`
  - CentOS package: `docker`

2.4.2 CNI plugins

```
export CNI_VERSION=v0.8.2
export ARCH=$([ $(uname -m) = "x86_64" ] && echo amd64 || echo arm64)
curl -sSL https://github.com/containernetworking/plugins/releases/download/${CNI_
↪VERSION}/cni-plugins-linux-${ARCH}-${CNI_VERSION}.tgz | tar -xz -C /opt/cni/bin
```

2.4.3 Other Binaries

- `mount` & `umount` for mounting and unmounting block devices
  - Ubuntu package: `mount` (installed by default)
  - CentOS package: `util-linux` (installed by default)
- `tar` for extracting files from the docker image onto the filesystem
  - Ubuntu package: `tar` (installed by default)
  - CentOS package: `tar` (installed by default)
- `mkfs.ext4` for formatting a block device with a ext4 filesystem
  - Ubuntu package: `e2fsprogs` (installed by default)
  - CentOS package: `e2fsprogs`
- `e2fsck` & `resize2fs` for cleaning and resizing the ext4 filesystems
  - Ubuntu package: `e2fsprogs` (installed by default)
  - CentOS package: `e2fsprogs`
- `strings` for detecting the kernel version
  - Ubuntu package: `binutils`
  - CentOS package: `binutils` (installed by default)
- `dmsetup` for managing device mapper snapshots and overlays

  - Ubuntu package: `dmsetup`
  - CentOS package: `device-mapper` (installed by default)
- `ssh` for SSH-ing into the VM (optional, for `ignite ssh` only)
  - Ubuntu package: `openssh-client`
  - CentOS package: `openssh-clients`
- `git` for the GitOps mode of Ignite (optional, for `ignite gitops` only)
  - Ubuntu package: `git`
  - CentOS package: `git`

CHAPTER

THREE

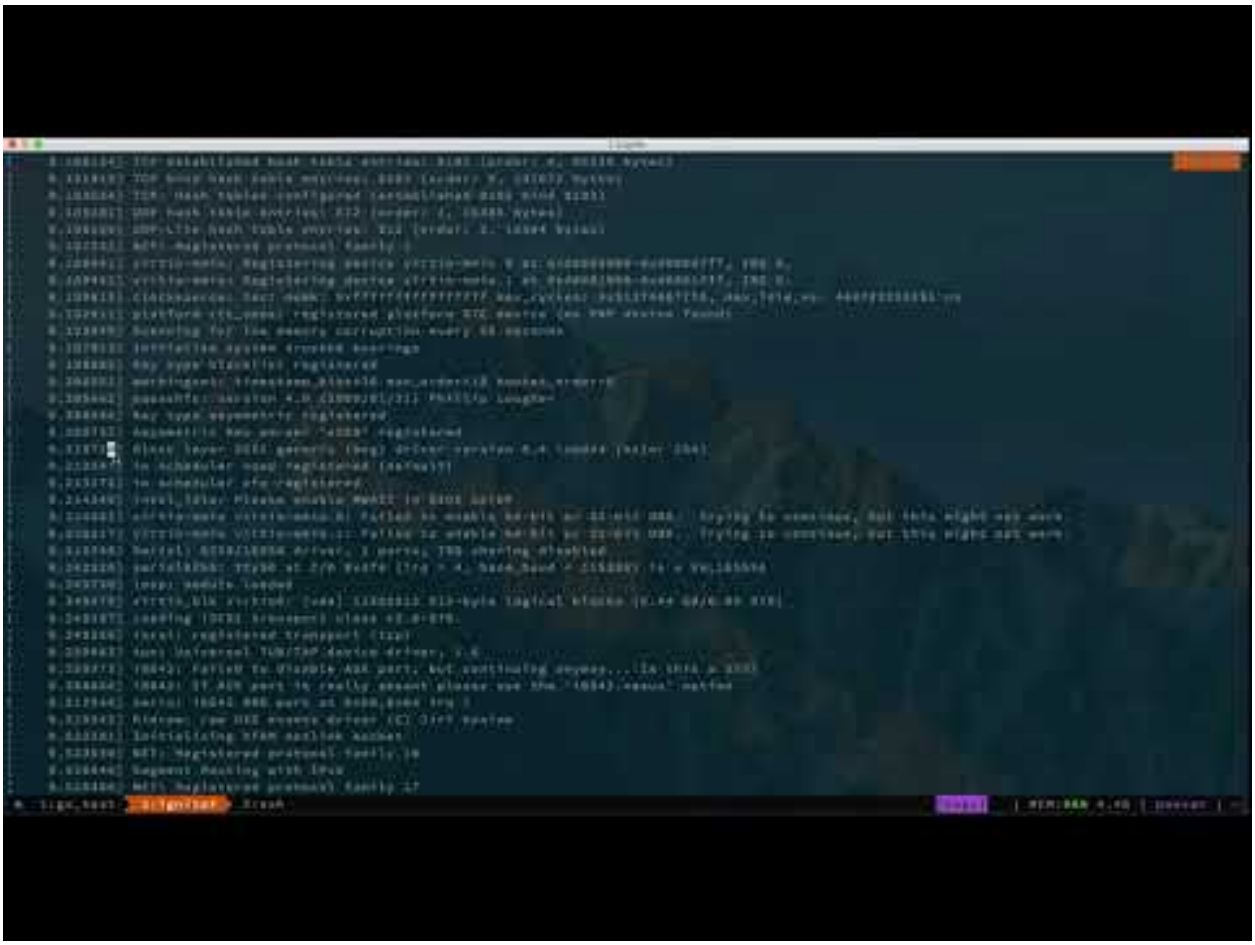
HOW TO USE IGNITE TO RUN VMS

Ignite is a containerized Firecracker microVM administration tool. It runs and manages virtual machines in separate containers using Firecracker.

This is a quick guide on how to get started with Ignite. The guide will cover the following topics in order:

Keep in mind that all Ignite commands require `root` for now. This will change later.

Here's a demo that shows the topics above in action with ignite running on Amazon EC2 i3.metal instance which satisfies the /dev/kvm dependency.



Alternatively, you might want to check out the TGIK deep-dive session from Joe Beda on what Ignite is and how it works:

3.1 Importing a VM base image

A VM base image (or just `image`) is an OCI container, which contains a filesystem and has an init system installed. Ignite currently supports importing `images` from Docker images, for which it has the following command:

```
# ignite image import <identifier>
```

The identifier can either be the UID of the image in Docker, or its name. If using the name without specifying a tag, `:latest` is automatically appended. Ignite can also match a prefix of the given name/UID in any command provided that it's unique, so you can e.g. enter just the three first letters of a name if they are unique to a single resource.

Go ahead and import the `weaveworks/ignite-ubuntu` Docker image. If it isn't present locally, Ignite will pull it for you:

```
# ignite image import weaveworks/ignite-ubuntu
...
INFO[0002] Created image with ID "cae0ac317cca74ba" and name "weaveworks/ignite-
↪ubuntu:latest"
```

Now the `weaveworks/ignite-ubuntu` image is imported and ready for VM use.

3.2 Creating a new VM based on the imported image

The `images` are read-only references of what every VM based on them should contain. To create a functional `VM`, Ignite uses `device mapper` to overlay a writable snapshot on top of the `image`. All changes to the `VM` will be saved in the snapshot.

Let's create a new VM with some options:

```
# ignite create weaveworks/ignite-ubuntu \
  --name my-vm \
  --cpus 2 \
  --memory 1GB \
  --size 6GB \
  --ssh
...
INFO[0001] Created VM with ID "3c5fa9a18682741f" and name "my-vm"
```

3.2.1 Options for VM generation

The previous example tells Ignite to create a `VM` with the name `my-vm` and that it should have 2 CPU cores, 1 GB of RAM, a writable snapshot size of 6 GB and have SSH access enabled.

The snapshot stores a delta compared to the base `image`, so a `--size` of "6GB" enables storing 6 Gigabytes of data changes (addition or removal).

The `--ssh` flag generates a new private/public key pair for the `VM` and exports the public key it into the `VM`. This is used for `ignite ssh <identifier>` later.

All available options can be listed with `ignite create --help`.

3.3 Starting a VM

Starting a created `VM` is very straight forward:

```
# ignite start my-vm
```

The `VM` will be matched by its name or ID (useful if there are similarly named `VMs`).

If no error occured, your `VM` is now running.

3.4 Inspecting VMs and their resources

Ignite currently manages three kinds of resources: `images`, `kernels` and `VMs`. The `kernels` are quite transparent, and get automatically imported from the docker image `weaveworks/ignite-kernel:4.19.47` by default (overridable during `create`).

To list the available `kernels`, enter:

```
# ignite kernels
KERNEL ID               NAME                                    CREATED SIZE    ␣
↪VERSION
aefb459546315344        weaveworks/ignite-kernel:4.19.47        61m ago 49.0 MB 4.19.
↪47
```

To list the imported `images`, enter:

```
# ignite images
IMAGE ID                NAME                                CREATED SIZE
cae0ac317cca74ba        weaveworks/ignite-ubuntu:latest 82m ago 268.9 MB
```

And to list the running `VMs`, enter:

```
# ignite ps
VM ID                   IMAGE                               KERNEL
↪           CREATED SIZE    CPUS    MEMORY          STATE   IPS             PORTS
↪NAME
3c5fa9a18682741f        weaveworks/ignite-ubuntu:latest weaveworks/ignite-kernel:4.19.
↪47         63m ago 4.0 GB  2       1.0 GB          Running 172.17.0.3
↪my-vm
```

To list all `VMs` instead of just running ones, add the `-a` flag to `ps`.

3.5 Accessing a VM

Ignite has two ways to access a CLI in a `VM`, the first option is to attach to the `VM's` TTY and the other is to SSH into the `VM`.

3.5.1 Attaching to the TTY

To attach to the running `VM's` TTY, enter:

```
# ignite attach my-vm
3c5fa9a18682741f
<enter>
Ubuntu 18.04.2 LTS 3c5fa9a18682741f ttyS0

3c5fa9a18682741f login:
```

If nothing is displayed, hit Enter to re-display the login prompt. Login using the credentials set in the `image` (usually `root` with password `root`).

To detach from the TTY, enter the key combination **^P^Q** (Ctrl + P + Q):

```
root@3c5fa9a18682741f:~# <^P^Q>
INFO[0001] Detached
$
```

3.6 SSH into the VM

NOTE: SSH works only if the `--ssh` flag is specified during `create`. Otherwise there are no public keys imported into the `VM` and most `images` have password-based root logins disabled for security reasons.

To SSH into a `VM`, enter:

```
# ignite ssh my-vm
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.19.47 x86_64)
...
root@3c5fa9a18682741f:~#
```

To exit SSH, just quit the shell process with `exit`.

NOTE: Each SSH access spawns its own session, but TTY access via `attach` is **shared**, every attached user operates the same terminal.

3.7 All in one

Ignite has a shorthand for performing `image import`, `create`, `start` and possibly also `attach` all in one command:

```
# ignite run weaveworks/ignite-ubuntu \
  --name another-vm \
  --cpus 2 \
  --memory 1GB \
  --size 6GB \
  --ssh \
  --interactive
```

This imports the given `image`, creates a new `VM` from it, starts the `VM` and attaches to the `VM's` TTY.

`run` accepts all the flags for `image import`, `create` and `start`. Using the `--interactive` flag of `start`, an `attach` is performed right after the `VM` has been started.

3.8 Stopping a VM

Ignite `VMs` can be stopped three ways:

1. By running: `# ignite stop my-vm`
2. By running: `# ignite kill my-vm`
3. By issuing the `reboot` command inside the VM

If the `VM's kernel` has support for Firecracker's virtual keyboard, `stop` will issue Ctrl + Alt + Del to gracefully shut down the `VM`. It will wait 20 seconds for Firecracker to exit, after which the `VM` will be forcibly killed.

`kill` is an alias for `stop -f`, which force-kills the `VM`. **WARNING:** The `VM` is given no time to close open resources, so this might lead to data loss or filesystem corruption.

Issuing `reboot` inside the `VM` is the recommended way to stop a `VM` that doesn't support Firecracker's virtual keyboard. By *rebooting* the `VM` Firecracker *shuts itself down gracefully*.

NOTE: Do *not* enter `shutdown` or `halt` inside the `VM`, this will result in Firecracker hanging.

3.9 Removing a VM

To remove `VMs` in Ignite, use the following command:

```
# ignite rm my-vm
```

The VM needs to not be running for this to succeed. Using the `--force` flag (`-f` for short) a running VM can also be removed, it will be killed before removal. This force option is also able to stop and remove zombie VMs.

```
# ignite rm -f my-vm
```

3.10 Removing other resources

To remove an `image`, run:

```
# ignite rmi weaveworks/ignite-ubuntu
```

And to remove a `kernel`, run:

```
# ignite rmk weaveworks/ignite-kernel:4.19.47
```

NOTE: To fully uninstall all Ignite data, remove the data directory at `/var/lib/firecracker`. Remember to stop all running VMs before doing this.

CHAPTER

FOUR

RUN IGNITE VMS DECLARATIVELY

Flags can be convenient for simple use, but have many limitations. For more advanced use cases, and to allow GitOps flows, there is another way: telling Ignite what to do *declaratively*, using a file containing an API object.

The first commands to support this feature are `ignite run` and `ignite create`. Here's an example API object file contents:

```
apiVersion: ignite.weave.works/v1alpha2
kind: VM
metadata:
  name: my-vm
spec:
  image:
    oci: weaveworks/ignite-ubuntu
  cpus: 2
  diskSize: 3GB
  memory: 800MB
```

This API object specifies a need for 2 vCPUs, 800 MB of RAM and 3 GB of disk space.

We can tell Ignite to make this happen by simply running:

```
$ ignite run --config my-vm.yaml
INFO[0001] Created VM with ID "e04128e6f96176a8" and name "my-vm"
INFO[0002] Networking is handled by "cni"
INFO[0002] Started Firecracker VM "e04128e6f96176a8" in a container with ID "ignite-
↪e04128e6f96176a8"
```

The full reference format for the `VM` kind is as follows:

```
apiVersion: ignite.weave.works/v1alpha2
kind: VM
metadata:
  # Automatically set when the object is created
  created: [RFC3339-formatted date]
  # Required, the name of the VM
  name: [string]
  # Optional, autogenerated if not specified
  uid: [16-char hex UID]
  # Optional, a string-string map with label keys and values
  labels:
    foo: bar
  # Optional, a string-string map with annotation keys and values
  annotations:
    foo: bar
```

(continues on next page)

(continued from previous page)

```
spec:
  # Optional, how many vCPUs should be allocated for the VM
  # Default: 1
  cpus: [uint64]
  # Optional, how much RAM should be allocated for the VM
  # Default: 512MB
  memory: [size]
  # Optional, how much free writable space the VM should have at runtime
  # Default: 4GB
  diskSize: [size]

  image:
    # Required, what OCI image to use as the VM's rootfs
    # For example: weaveworks/ignite-ubuntu:latest
    oci: [OCI image reference]
  kernel:
    # Optional, the kernel command line for the VM
    # Default: "console=ttyS0 reboot=k panic=1 pci=off ip=dhcp"
    cmdLine: [string]
    # Required, what OCI image to get the kernel binary (and optionally modules) from
    # Default: weaveworks/ignite-kernel:4.19.47
    oci:  [OCI image reference]

  network:
    # Optional, an array of port mappings that map ports bound to the VM to the host
    # Default: unset, no port mappings
    ports:
    # This example maps UDP port 0.0.0.0:6443 inside the VM to 10.0.0.2:443 on the␣
↪physical host
    - hostPort: 433
      vmPort: 6443
      # Optional, specify an address to bind to on the host
      # Default: 0.0.0.0, any address
      bindAddress: 10.0.0.2
      # Optional, specify a protocol for the port mapping (tcp or udp)
      # Default: tcp
      protocol: udp

  storage:
    # Optional, an array of mountPath and name pairs,
    # set the mount points for named volumes inside the VM.
    # Names must match configured named volumes.
    # Default: unset, no mount points
    volumeMounts:
    - mountPath: /mnt
      name: volume0
    # Optional, an array of blockDevice and name pairs,
    # expose block devices on the host inside the VM.
    # The blockDevice path must point to a block device formatted
    # with a filesystem providing an UUID (such as ext4 or xfs).
    # Default: unset, no volume forwarding
    volumes:
    - blockDevice:
        path: /dev/sdb1
      name: volume0

  # Optional, an array of files/directories to copy into the VM on creation
```

(continues on next page)

(continued from previous page)

```
  # Default: unset, nothing will be copied
  copyFiles:
  # This example copies a Kubernetes KubeConfig file from /etc/kubernetes/admin.conf
  # on the host to /home/user/.kube/config inside the VM
  - hostPath: /etc/kubernetes/admin.conf
    vmPath: /home/user/.kube/config

  # Optional, provides automation to easily access your VM with the "ignite ssh"␣
↪command
  # If "ssh: true" is set, Ignite will generate an SSH key and copy the
  # public key into the VM. This allows for automatic "ignite ssh" logins.
  # Alternatively: specify a path to a public key to put in /root/.ssh/authorized_
↪keys in the VM.
  # Default: unset, no actions regarding SSH automation
  ssh: [true, or public key path]
```

You can find the full API reference in the api/ subfolder of the project.

CHAPTER

FIVE

IGNITE - THE GITOPS VM

Ignite is a "GitOps-first" project, GitOps is supported out of the box using the `ignited gitops` command. Previously this was integrated as `ignite gitops`, but this functionality has now moved to `ignited`, Ignite's upcoming daemon binary.

In Git you declaratively store the desired state of a set of VMs you want to manage. `ignited gitops` reconciles the state from Git, and applies the desired changes as state is updated in the repo. It also commits and pushes any local changes/additions to the managed VMs back to the repository.

This can then be automated, tracked for correctness, and managed at scale - just some of the benefits of GitOps.

The workflow is simply this:

- Run `ignited gitops [repo]`, where repo is an **SSH url** to your Git repo
- Create a file with the VM specification, specifying how much vCPUs, RAM, disk, etc. you'd like for the VM
- Run `git push` and see your VM start on the host

See it in action! (Note: The screencast is from an older version which differs somewhat)

asciicast

5.1 Try it out

Go ahead and create a Git repository.

NOTE: You need an SSH key for **root** that has push access to your repository. `ignited` will commit and push changes back to the repository using the default key for it. To edit your root's git configuration, run `sudo gitconfig --global --edit`. The root requirement will be removed in a future release.

Here's a sample configuration you can push to it (my-vm.yaml):

```
apiVersion: ignite.weave.works/v1alpha2
kind: VM
metadata:
  name: my-vm
  uid: 599615df99804ae8
spec:
  image:
    oci: weaveworks/ignite-ubuntu
  cpus: 2
  diskSize: 3GB
  memory: 800MB
  ssh: true
```

(continues on next page)

(continued from previous page)

```
status:
  running: true
```

For a more complete example repository configuration, see luxas/ignite-gitops

After you have installed Ignite, you can do the following:

```
ignited gitops git@github.com:<user>/<repository>.git
```

NOTE: HTTPS doesn't preserve authentication information for `ignited` to push changes, you need to set up SSH authentication and use the SSH clone URL for now.

Ignite will now search that repo for suitable JSON/YAML files, and apply their state locally. You should see `my-vm` starting up in `ignite ps`. To enter the VM, run `ignite ssh my-vm`.

Please refer to docs/declarative-config.md for the full API reference.

CHAPTER

SIX

NETWORKING

Ignite uses network plugins to manage VM networking.

The default plugin is CNI; and the default CNI network is automatically put in `/etc/cni/net.d/10-ignite.conflist` if `/etc/cni/net.d` is empty. In order to switch to some other CNI plugin, remove `/etc/cni/net.d/10-ignite.conflist`, and install e.g. *Weave Net* like below.

The legacy `docker-bridge` networking provider is also supported, but deprecated.

To select the network plugin, use the `--network-plugin` flag for `ignite` and `ignited`:

```
ignite --network-plugin cni <command>
ignited --network-plugin docker-bridge <command>
```

6.1 Comparison

6.1.1 The default CNI network

Automatically installed to `/etc/cni/net.d/10-ignite.conflist` unless you have populated `/etc/cni/net.d` with something else.

Pros:

- **Kubernetes-compatible**: You can use the same overlay networks as you use with Kubernetes, and hence get your VMs on the same network as your containers.
- **Port mapping support**: This mode supports port mappings from the VM to the host

Cons:

- **No multi-node support**: The IP is local (in the `172.18.0.0/16` range), and hence other computers can't connect to your VM's IP address.

6.1.2 An third-party CNI plugin

For example, *Weave Net*, or an other Kubernetes and/or CNI-implementation.

Pros:

- **Multi-node support**: CNI implementations can often route packets between multiple physical hosts. External computers can access the VM's IP.
- **Kubernetes-compatible**: You can use the same overlay networks as you use with Kubernetes, and hence get your VMs on the same network as your containers.

- **Port mapping support**: This mode supports port mappings from the VM to the host

Cons:

- **More software needed**: There's now one extra piece of software to configure and manage.

6.1.3 docker-bridge

Pros:

- **Quick start**: If you're running `docker`, you can get up and running without installing extra software
- **Port mapping support**: This mode supports port mappings from the VM to the host

Cons:

- **docker-dependent**: By design, this mode is can only be used with docker, and is hence not portable across container runtimes.
- **No multi-node support**: The IP is local (in the `172.17.0.0/16` range), and hence other computers can't connect to your VM's IP address.

6.2 Multi-node networking with Weave Net

To use e.g. Ignite together with Weave Net, run this on all physical machines that need to connect to the overlay network:

```
# Remove Ignite's default CNI network if it exists
rm -rf /etc/cni/net.d/10-ignite.conflist

# This tries to autodetect the primary IP address of this machine
# Ref: https://stackoverflow.com/questions/13322485/how-to-get-the-primary-ip-address-
↪of-the-local-machine-on-linux-and-macos
export PRIMARY_IP_ADDRESS=$(ip -o route get 1.1.1.1 | cut -d' ' -f7)
# A space-separated list of all the peers in the overlay network
export KUBE_PEERS="${PRIMARY_IP_ADDRESS}"
# Start Weave Net in a container
docker run -d \
  --privileged \
  --net host \
  --pid host \
  --restart always \
  -e HOSTNAME="$(hostname)" \
  -e KUBE_PEERS="${KUBE_PEERS}" \
  -v /var/lib/weave:/weavedb \
  -v /opt:/host/opt \
  -v /home:/host/home \
  -v /etc:/host/etc \
  -v /var/lib/dbus:/host/var/lib/dbus \
  -v /lib/modules:/lib/modules \
  -v /run/xtables.lock:/run/xtables.lock \
  --entrypoint /home/weave/launch.sh \
  weaveworks/weave-kube:2.5.2
```

If you're running Kubernetes on the physical machine you want to use for Ignite VMs, it should work out of the box, as the CNI implementation is most probably already running in a `DaemonSet` on that machine.

CHAPTER

SEVEN

MONITOR IGNITE WITH PROMETHEUS

Every Ignite VM container exposes Prometheus metrics about itself.

These metrics are available with the following command:

```
VM_NAME="my-vm"
VM_ID=$(ignite inspect vm ${VM_NAME} | jq -r .metadata.uid)
curl --unix-socket /var/lib/firecracker/vm/${VM_ID}/prometheus.sock http:/metrics
```

This will report metrics for the `ignite-spawn` component, managing the Firecracker daemon inside of the container. If you want to see how much overhead `ignite-spawn` and `firecracker` have combined for running a VM, you can check it with `docker stats`:

```
$ VM_NAME="my-vm"
$ VM_ID=$(ignite inspect vm ${VM_NAME} | jq -r .metadata.uid)
$ docker stats ignite-${VM_ID}
CONTAINER ID        NAME                      CPU %               MEM USAGE / LIMIT
↪   MEM %               NET I/O             BLOCK I/O           PIDS
29259f616d9c        ignite-cc82b4424244b3e4   3.12%               9.145MiB / 15.52GiB
↪   0.06%               5.35kB / 2.5kB      324MB / 4.1kB       15
$ docker top ignite-${VM_ID}
UID                 PID                 PPID                C                   STIME
↪               TTY                 TIME                CMD
root                28693               28666               0                   14:11
↪               pts/0               00:00:00            /usr/local/bin/ignite-spawn
↪cc82b4424244b3e4
root                28785               28693               1                   14:11
↪               pts/0               00:00:01            firecracker --api-sock /tmp/
↪firecracker.sock
```

CHAPTER

EIGHT

RUN A SET OF IGNITE VMS WITH FOOTLOOSE

If you think of Firecracker as `runc`, you can think of Ignite as `docker`. Is there then any `docker-compose` thing for Ignite?

Yes, Footloose. With Footloose, you can run containers as Virtual Machines in two modes, either using `docker` or `ignite`.

`docker` isolation with Footloose is good for environments where KVM is not enabled (e.g. public CI providers, Macs, etc.), and `ignite` isolation is good when you want to run real VMs.

8.1 Installation

Install Footloose 0.6.0 or higher like this:

```
export VERSION=0.6.0
curl -sLo footloose https://github.com/weaveworks/footloose/releases/download/$
↪{VERSION}/footloose-${VERSION}-linux-x86_64
chmod +x footloose
sudo mv footloose /usr/local/bin/
```

8.2 Get Started

This how you can have Footloose invoke Ignite in a *declaratively* manner, using a file containing an API object.

An example file as follows:

```
# footloose.yaml
cluster:
  name: cluster
  privateKey: cluster-key
machines:
- count: 3
  spec:
    image: weaveworks/ignite-ubuntu:latest
    name: vm%d
    portMappings:
    - containerPort: 22
    # This is by default "docker". However, just set this to "ignite" and it'll work␣
↪with Ignite :)
    backend: ignite
    # Optional configuration parameters for ignite:
```

(continues on next page)

(continued from previous page)

```
    ignite:
      cpus: 2
      memory: 1GB
      diskSize: 5GB
      kernel: "weaveworks/ignite-kernel:4.19.47"
```

This Footloose API object specifies an Ignite VM with 2 vCPUs, 1GB of RAM, `weaveworks/ignite-kernel:4.19.47` kernel and 5GB of disk.

Given that you have Footloose and *Ignite* installed, and the above file created as `footloose.yaml` in the current directory, you can run

```
$ footloose create
INFO[0000] Docker Image: weaveworks/ignite-ubuntu:latest present locally
INFO[0000] Creating machine: cluster-vm0 ...
INFO[0002] Creating machine: cluster-vm1 ...
INFO[0004] Creating machine: cluster-vm2 ...
```

SSH into the VM:

```
$ footloose ssh vm0
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.19.47 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@a07c6f1782c70136:~#
```

Run the following to stop the VMs:

```
$ footloose stop
INFO[0000] Stopping machine: cluster-vm0 ...
INFO[0002] Stopping machine: cluster-vm1 ...
INFO[0004] Stopping machine: cluster-vm2 ...
```

For more information check out the Footloose README.

CHAPTER

NINE

AWESOME-IGNITE

An awesome list of interesting projects, blog posts and other resources using, mentioning or buidling on top of Ignite.

If you have created a blog post or similar resource related to Ignite, please send a PR to this list!

9.1 Blog Posts

- Fire Up Your VMs with Weave Ignite by Alexis Richardson, 10th of July, 2019
- Weave Ignite et Weave Footloose dans Scaleway: quand les machines virtuelles se prennent pour des containers et inversement by Karim, 30th of June, 2019
- Weave didn't start the fire, but anyway…here's Ignite by Julia Schmidt, 11st of July, 2019
- Das Projekt Ignite soll Container und VMs zusammenbringen by Matthias Parbel, 11st of July, 2019
- Ignite VM using Firecracker by Arun Gupta, 13th of July, 2019
- Ignite on Google Cloud by Saiyam Pathak, 13rd of July, 2019
- The clearest sign of AWS' open source success wasn't built by Amazon by Matt Asay, 16th of July, 2019
- Weaveworks Releases Ignite, AWS Firecracker-Powered Software for Running Containers as VMs by Richard Seroter, 18th of July, 2019
- Ignite - container workload in real virtual machines by Felix Wiedmann, 20th of July, 2019
- Weave Ignite Brings a Container GitOps Experience to VMs by Mike Melanson, 25th of July, 2019

9.2 YouTube Videos

- Weave Ignite - Firecracker Micro VM manager demo by Paavan Mistry, 10th of July, 2019
- TGI Kubernetes 082: Weaveworks Ignite by Joe Beda, 12nd of July, 2019

9.3 Online Events

- WOUG: Introducing Ignite - the GitOps VM with Lucas Käldström & Radu Weiss by Tamao Nakahara, 17th of July, 2019

9.4 Screencasts

- Get started with Ignite
- Spawn multiple Ignite VMs with Footloose
- Set up a HA Kubernetes cluster in Ignite VMs

9.5 Projects

- Virtual Kubelet provider for Ignite by Chanwit Kaewkasi
- ignite-ci: Ignite and Ansible by Jean-Alexis Lauricella
- LF Edge Virtualization Engine; possible user

CHAPTER

TEN

CLOUD PROVIDER INSTANCES WITH KVM SUPPORT

If you intend to use a cloud provider to test Ignite, you can use the instructions below to provision an instance that satisfies the KVM system requirements described in the *installation guide*.

10.1 Amazon Web Services

Amazon EC2 bare metal instances provide direct access to the Intel® Xeon® Scalable processor and memory resources of the underlying server. These instances are ideal for workloads that require access to the hardware feature set (such as Intel® VT-x), for applications that need to run in non-virtualized environments for licensing or support requirements, or for customers who wish to use their own hypervisor.

Here's a list of instances with KVM support, with pricing (as of July 2019), to help you test Ignite. All the instances listed below are EBS-optimized, with 25 Gigabit available network performance and IPv6 support.

Use the AWS console to launch one of these instances and connect to your instance using SSH. Then, follow the instructions in the *installation guide*.

10.2 Google Cloud

Source: https://blog.kubernauts.io/ignite-on-google-cloud-5d5228a5ffec

Use Google compute from a custom KVM image so that Ignite can be installed and run easily.

- Login to Google Cloud Console
- Open Google Cloud Shell
- Run the following command to create custom images with KVM enabled

```
gcloud compute images create nested-virt \
  --source-image-project=ubuntu-os-cloud \
  --source-image-family=ubuntu-1604-lts \
  --licenses="https://www.googleapis.com/compute/v1/projects/vm-options/global/
↪licenses/enable-vmx"
```

- Create an instance with the custom image created

10.3 Microsoft Azure

Azure supports nested virtualizations on Dv3 and Ev3 series virtual machines with 4 or more vCPUs. The smallest sizes supporting Ignite are listed below for development purposes. Larger and more specialized SKUs also support

virtualization. There is no special configuration required. Follow the standard Ignite installation instructions for Ubuntu.

10.4 Packet

As a bare metal provider Packet naturally supports virtualization. For development purposes, using a t1.small.x86 with Ubuntu 18.04 or 19.04 works well.

CHAPTER

ELEVEN

DEVELOPER DOCUMENTATION

Ignite is a Go project using well-known libraries like:

- github.com/spf13/cobra
- github.com/spf13/pflag
- k8s.io/apimachinery
- sigs.k8s.io/yaml
- github.com/firecracker-microvm/firecracker-go-sdk

and so on.

It uses Go modules as the vendoring mechanism.

11.1 Build from source

The only build requirement is Docker.

To build `ignite`, `ignited` and `ignite-spawn` for all supported architectures, run:

```
make build-all
```

To only build for a specific architecture, append the architecture to the command:

```
make build-all-amd64
make build-all-arm64
```

11.2 Pre-commit tidying

Before committing, please run this make target to (re)generate autogenerated content and tidy your local environment:

```
make autogen tidy
```

11.3 Building reference OS images

```
make -C images WHAT=ubuntu
make -C images WHAT=centos
```

11.4 Generic instructions on releasing a version

- Fix all issues in the `v0.X.Y` milestone
- Create a `v0.X.Y` tracking issue (like https://github.com/weaveworks/ignite/issues/379)
- Update documentation for the latest version (like https://github.com/weaveworks/ignite/pull/378)
- Make sure your git remote `upstream` points to `git@github.com:weaveworks/ignite.git`, and `origin` to `git@github.com:<user>/ignite.git`
- Get a Github API token with `repo` access, and put it in `bin/gren_token` for automatic release note generation
- Make sure you have access to push to the `weaveworks/ignite*` repositories to Docker Hub

11.5 Releasing a minor version

- A minor version is done based off the `master` branch
- Set the environment variable to tell what minor version to release: `export MINOR=X`
  - If this is a prerelease, set e.g. `export EXTRA=-alpha.1`, `export EXTRA=-beta.1`, or `export EXTRA=-rc.1`
- The script to run is `hack/minor-release.sh all`. It will:
  - Tidy your environment by running `make tidy autogen graph` and doing a commit
  - Autogenerate the changelog, provisionally using GREN. The script will wait for you to open an editor and manually fixup `docs/releases/v0.X.0.md`. Then proceed with `Y`, which will create the commit to be tagged `v0.X.0`
  - Create the `v0.X.0` tag using `git tag`
  - Build the release binaries to `bin/releases/v0.X.0` and push the `weaveworks/ignite:v0.X.0` images and manifest list to Docker Hub
  - Push the tag, and latest commits to the `master` and newly-created `release-0.X` branch

11.6 Releasing a patch version

- A patch version is done based off the `release-0.X` branch
  - **Note:** Before running the release, `git cherrypick` relevant commits into the release branch
- Set the environment variables to tell what patch version to release: `export MINOR=X` and `export PATCH=Y`
  - If this is a prerelease, set e.g. `export EXTRA=-alpha.1`, `export EXTRA=-beta.1`, or `export EXTRA=-rc.1`
- The script to run is `hack/patch-release.sh all`
  - Tidy your environment by running `make tidy autogen graph` and doing a commit
  - Autogenerate the changelog, provisionally using GREN. The script will wait for you to open an editor and manually fixup `docs/releases/v0.X.Y.md`. Then proceed with `Y`, which will create the commit to be tagged `v0.X.Y`

  - Create the `v0.X.Y` tag using `git tag`
  - Build the release binaries to `bin/releases/v0.X.Y` and push the `weaveworks/ignite:v0.X.Y` images and manifest list to Docker Hub
  - Push the tag, and latest commits to the `release-0.X` branch

11.7 Publishing a release

- Go to `Project Releases` in the Github UI, and select `Draft a new release`
- Select the tag you've just created `v0.X.Y` from either the `master` (minor releases) or `release-0.X` branch
- Let the release title be `v0.X.Y`
- Paste the content in `docs/releases/v0.X.Y.md` in the release description, and add installing guidelines as per the earlier releases
- Upload the binaries in `bin/releases/v0.X.Y` named as `{ignite,ignited}-{amd64,arm64}`
- Click `Publish Release` and announce to everyone you know!

CHAPTER

TWELVE

ROADMAP

This is a provisional roadmap for what we'd like to do in the coming releases.

- Split the `ignite` CLI into a client-server model (using e.g. `gRPC`)
  - The CLI should only be a thin wrapper that talks to `ignited`
  - With this `ignite` can be run without root
  - `ignited` will be run with `root` privileges, or in a container with capabilities specifically set
- Provide deb/rpm packages for an easier installation
- Add Virtual Kubelet support to `ignited`
  - `ignited` will register as a Virtual Kubelet in the target Kubernetes cluster
  - The `VM` API type will be register as a `CustomResourceDefinition`
- Use device-mapper Thin Provisioning for layering image -> kernel -> resize -> writable overlay
  - We might be able to utilize/vendor in containerd's devicemapper snapshotter
- Define what's in and out of scope for Ignite clearly, e.g.
  - Supporting to restart VMs or not
  - Supporting multiple network interfaces or not
  - Create one architecture diagram more and a design document
- Parallelized internal architecture for better performance
- Generate OpenAPI documentation and specifications
- Add support for CSI volumes

CHAPTER

THIRTEEN

FAQ

A collection of Frequently Asked Questions about Ignite:

13.1 Q: Can I use Ignite as CRI runtime for kubelet?

No, you can't. Ignite isn't designed for running containers, hence it cannot work as a CRI runtime.

Ignite runs VMs instead. In the future, we envision Ignite to (maybe) be able to run VMs (not containers) based off Kubernetes Pods using some special annotations. This would however (most likely) be done as a containerd plugin (lower in the stack than CRI).

13.2 Q: What is the difference between {Kata Containers, gVisor, fc-containerd} and Ignite?

Kata Containers, gVisor, and firecracker-containerd run *containers*, and Ignite runs *VMs*.

Kata can integrate with Firecracker, but the value add there is more isolation, as the container is spawned inside of a minimal Firecracker VM.

firecracker-containerd enables you to do the same as Kata; add isolation for a container; but this time in a bit more lightweight manner, as a containerd plugin.

gVisor acts as a gatekeeper between your application in a container, and the kernel. gVisor emulates the kernel syscalls, and based on if they are "safe" or not, passes them though to the underlying kernel, or performs a similar operation. gVisor's value-add is the same as the above, added isolation for containers.

Ignite however, uses the rootfs from an OCI image, and runs that content as a real VM. Inside of the Firecracker VM spawned, there are no extra containers running (unless the user installs a container runtime).

13.3 Q: Why does Ignite require KVM?

Firecracker is a KVM implementation, and uses KVM to manage and virtualize the VM.

13.4 Q: Why does Ignite require to be root?

In order to prepare the VM filesystem, Ignite needs to create a file containing an ext4 filesystem for Firecracker to boot. In order to populate this filesystem in the file-based block device, Ignite needs to temporarily `mount` the filesystem, and copy the desired root filesystem source contents in. `mount` requires the UID to be 0 (root).

We hope to remove this requirement from the Ignite CLI in the future #24, #33. However, some part of Ignite (although hidden) will always need to execute as root due to the need to `mount`.

13.5 Q: Can I run Ignite on a Mac?

No. Firecracker requires KVM, as per the above, a feature that is not available on MacOS. Technically, you could spin up a VM running Linux inside of a Mac, and inside of that Linux VM, with nested virtualization enabled, run Ignite. However, that might defeat the purpose of Ignite on Mac in the first place.

13.6 Q: Why is Docker (containers) needed/used?

Docker, currently the only available container runtime usable by Ignite, is used for a couple of reasons:

1. **Running long-lived processes**: At the very early Ignite PoC stage, we tried to run the Firecracker process under `systemd`, but this was in many ways suboptimal. Attaching to the serial console, fetching logs, and 2. and 3. were very hard to achieve. Also, we'd need to somehow install the Firecracker binary on host. Packaging everything in a container, and running the Firecracker process in that container was a natural fit.
2. **Sandboxing the Firecracker process**: Firecracker should not be run on host without sandboxing, as per their security model. Firecracker provides the jailer binary to do sandboxing/isolation from the host for the Firecracker process, but a container does this equally well, if not better.
3. **Container Networking**: Using containers, we already know what IP to give the VM. We can integrate with e.g. the default docker bridge, docker's `libnetwork` in general, or CNI. This reduces the amount of scope and work needed by Ignite, and keeps our implementation lean. It also directly makes Ignite usable alongside normal containers, e.g. on a host running Kubernetes Pods.
4. **OCI compliant operations**: Using an existing container runtime, we do not need to implement everything from the OCI spec ourselves. Instead, we can re-use functionality from the runtime, e.g. `pull`, `create`, and `export`.

All in all, we do not want to reinvent the wheel. We reuse what we can from existing proven container tools.

13.7 Q: How does my filesystem in a Docker image end up in a Firecracker VM?

In short, we `pull` an OCI image using the container runtime (Docker for now), `create` a new container using this image, and finally `export` the rootfs of that created container to a tar file. This tar file is then extracted into the mount point of an ext4-formatted block device file of the OCI image's size. The kernel OCI image is similarly copied into the rootfs of the container. Lastly, Ignite modifies some well-known files like `/etc/hosts` and `/etc/resolv.conf` for the VM to work as you would expect it to.

13.8 Q: How does networking work as there are both containers and VMs?

First, Ignite spawns a container using the runtime. In this container, one Ignite component, `ignite-spawn`, is running. `ignite-spawn` loops the network interfaces inside of the container, and looks for a valid one to use for the VM. It removes the IP address from the container, and remembers it for later.

Next, `ignite-spawn` creates a `tap` device which Firecracker will use, and bridges the `tap` device with the existing `veth` interface created by the container runtime. With these two interfaces bridged, all information routed to the container, will end up in the VM's `tap` interface.

Lastly, `ignite-spawn` spawns the Firecracker process, which starts the VM. The VM is started with the `ip=dhcp` kernel argument, which makes the kernel automatically do a DHCP request for an IP. The kernel asks for an IP to use, and `ignite-spawn` responds with the IP the container initially had.

13.9 Q: Where does Ignite originate from?

As per the announcement blog post: https://www.weave.works/blog/fire-up-your-vms-with-weave-ignite

> Ignite is a clean room implementation of a project Lucas prototyped while on army service.
>
> Lucas Käldström (@luxas) is a Kubernetes SIG Lead and Top CNCF Ambassador 2017, and is a long-standing member of the Weaveworks family since graduating from High School (story here). As a young Finnish citizen, Lucas had to do his mandatory Military Service for around a year.
>
> Naturally for Lucas, he started evangelising Kubernetes within the military, and got assigned programming tasks. Security and resource consumption are critical army concerns, so Lucas and a colleague, Dennis Marttinen (@twelho), decided to experiment with Firecracker, creating an elementary version of Ignite. On leaving the army they were granted permission to work on an open source rewrite, working with Weaveworks.

CHAPTER

FOURTEEN

IGNITE API TYPES

The Ignite API types use the `k8s.io/apimachinery` framework, and similar conventions as the Kubernetes API types. That said, the Ignite API types don't (at the moment at least) have any connection or depend on Kubernetes itself in any way.

The API types are used for two primary things:

- Letting end users **:doc:'declaratively specify advanced configuration <../declarative-config>'_** (e.g. `ignite create --config`)
- Persisting internal data on disk in a standardized way.

Please refer to the (autogenerated) docs for the API types below:

14.1 v1alpha1

```
import "github.com/weaveworks/ignite/pkg/apis/ignite/v1alpha1"
```

- *Overview*
- *Index*

14.1.1 Overview

+k8s:deepcopy-gen=package +k8s:defaulter-gen=TypeMeta +k8s:openapi-gen=true +k8s:conversion-gen=github.com/weaveworks/ignite/pkg/apis/ignite

14.1.2 Index

- *Constants*
- *Variables*
- *func Convert_ignite_ImageSpec_To_v1alpha1_ImageSpec(in *ignite.ImageSpec, out *ImageSpec, s conversion.Scope) error*
- *func Convert_ignite_KernelSpec_To_v1alpha1_KernelSpec(in *ignite.KernelSpec, out *KernelSpec, s conversion.Scope) error*
- *func Convert_ignite_OCIImageSource_To_v1alpha1_OCIImageSource(in *ignite.OCIImageSource, out *OCIImageSource, s conversion.Scope) error*
- *func Convert_ignite_OCI_To_v1alpha1_OCIClaim(in *meta.OCIImageRef, out *OCIImageClaim) error*

- *func Convert_ignite_VMImageSpec_To_v1alpha1_VMImageSpec(in *ignite.VMImageSpec, out *VMImageSpec, s conversion.Scope) error*
- *func Convert_ignite_VMKernelSpec_To_v1alpha1_VMKernelSpec(in *ignite.VMKernelSpec, out *VMKernelSpec, s conversion.Scope) error*
- *func Convert_ignite_VMSpec_To_v1alpha1_VMSpec(in *ignite.VMSpec, out *VMSpec, s conversion.Scope) error*
- *func Convert_ignite_VMStatus_To_v1alpha1_VMStatus(in *ignite.VMStatus, out *VMStatus, s conversion.Scope) error*
- *func Convert_v1alpha1_ImageSpec_To_ignite_ImageSpec(in *ImageSpec, out *ignite.ImageSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_KernelSpec_To_ignite_KernelSpec(in *KernelSpec, out *ignite.KernelSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_OCIClaim_To_ignite_OCI(in *OCIImageClaim, out *meta.OCIImageRef) error*
- *func Convert_v1alpha1_OCIImageSource_To_ignite_OCIImageSource(in *OCIImageSource, out *ignite.OCIImageSource, s conversion.Scope) (err error)*
- *func Convert_v1alpha1_VMImageSpec_To_ignite_VMImageSpec(in *VMImageSpec, out *ignite.VMImageSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_VMKernelSpec_To_ignite_VMKernelSpec(in *VMKernelSpec, out *ignite.VMKernelSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_VMNetworkSpec_To_ignite_VMNetworkSpec(in *VMNetworkSpec, out *ignite.VMNetworkSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_VMSpec_To_ignite_VMSpec(in *VMSpec, out *ignite.VMSpec, s conversion.Scope) error*
- *func Convert_v1alpha1_VMStatus_To_ignite_VMStatus(in *VMStatus, out *ignite.VMStatus, s conversion.Scope) error*
- *func SetDefaults_OCIImageClaim(obj *OCIImageClaim)*
- *func SetDefaults_PoolSpec(obj *PoolSpec)*
- *func SetDefaults_VMKernelSpec(obj *VMKernelSpec)*
- *func SetDefaults_VMNetworkSpec(obj *VMNetworkSpec)*
- *func SetDefaults_VMSpec(obj *VMSpec)*
- *func SetDefaults_VMStatus(obj *VMStatus)*
- *type FileMapping*
- *type Image*
- *type ImageSourceType*
- *type ImageSpec*
- *type ImageStatus*
- *type Kernel*
- *type KernelSpec*
- *type KernelStatus*
- *type NetworkMode*

    - *func (nm NetworkMode) String() string*
- *type OCIImageClaim*
- *type OCIImageSource*
- *type Pool*
- *type PoolDevice*
- *type PoolDeviceType*
- *type PoolSpec*
- *type PoolStatus*
- *type SSH*
    - *func (s *SSH) MarshalJSON() ([]byte, error)*
    - *func (s *SSH) UnmarshalJSON(b []byte) error*
- *type VM*
- *type VMImageSpec*
- *type VMKernelSpec*
- *type VMNetworkSpec*
- *type VMSpec*
- *type VMState*
- *type VMStatus*

Package files

conversion.go defaults.go doc.go json.go register.go types.go

14.1.3 Constants

```
const (
    KindImage  runtime.Kind = "Image"
    KindKernel runtime.Kind = "Kernel"
    KindVM     runtime.Kind = "VM"
)
```

```
const (
    // GroupName is the group name use in this package
    GroupName = "ignite.weave.works"
)
```

14.1.4 Variables

```
var (
    // SchemeBuilder the schema builder
    SchemeBuilder = runtime.NewSchemeBuilder(
        addKnownTypes,
```

(continues on next page)

(continued from previous page)

```
        addDefaultingFuncs,
    )

    AddToScheme = localSchemeBuilder.AddToScheme
)
```

```
var SchemeGroupVersion = schema.GroupVersion{
    Group:   GroupName,
    Version: "v1alpha1",
}
```

SchemeGroupVersion is group version used to register these objects

14.1.5 func Convert_ignite_ImageSpec_To_v1alpha1_ImageSpec

```
func Convert_ignite_ImageSpec_To_v1alpha1_ImageSpec(in *ignite.ImageSpec, out␣
↪*ImageSpec, s conversion.Scope) error
```

Convert_ignite_ImageSpec_To_v1alpha1_ImageSpec calls the autogenerated conversion function along with custom conversion logic

14.1.6 func Convert_ignite_KernelSpec_To_v1alpha1_KernelSpec

```
func Convert_ignite_KernelSpec_To_v1alpha1_KernelSpec(in *ignite.KernelSpec, out␣
↪*KernelSpec, s conversion.Scope) error
```

Convert_ignite_KernelSpec_To_v1alpha1_KernelSpec calls the autogenerated conversion function along with custom conversion logic

14.1.7 func Convert_ignite_OCIImageSource_To_v1alpha1_OCIImageSource

```
func Convert_ignite_OCIImageSource_To_v1alpha1_OCIImageSource(in *ignite.
↪OCIImageSource, out *OCIImageSource, s conversion.Scope) error
```

Convert_ignite_OCIImageSource_To_v1alpha1_OCIImageSource calls the autogenerated conversion function along with custom conversion logic

14.1.8 func Convert_ignite_OCI_To_v1alpha1_OCIClaim

```
func Convert_ignite_OCI_To_v1alpha1_OCIClaim(in *meta.OCIImageRef, out␣
↪*OCIImageClaim) error
```

14.1.9 func Convert_ignite_VMImageSpec_To_v1alpha1_VMImageSpec

```
func Convert_ignite_VMImageSpec_To_v1alpha1_VMImageSpec(in *ignite.VMImageSpec, out␣
↪*VMImageSpec, s conversion.Scope) error
```

Convert_ignite_VMImageSpec_To_v1alpha1_VMImageSpec calls the autogenerated conversion function along with custom conversion logic

14.1.10 func Convert_ignite_VMKernelSpec_To_v1alpha1_VMKernelSpec

```
func Convert_ignite_VMKernelSpec_To_v1alpha1_VMKernelSpec(in *ignite.VMKernelSpec,␣
↪out *VMKernelSpec, s conversion.Scope) error
```

Convert_ignite_VMKernelSpec_To_v1alpha1_VMKernelSpec calls the autogenerated conversion function along with custom conversion logic

14.1.11 func Convert_ignite_VMSpec_To_v1alpha1_VMSpec

```
func Convert_ignite_VMSpec_To_v1alpha1_VMSpec(in *ignite.VMSpec, out *VMSpec, s␣
↪conversion.Scope) error
```

Convert_ignite_VMSpec_To_v1alpha1_VMSpec calls the autogenerated conversion function along with custom conversion logic

14.1.12 func Convert_ignite_VMStatus_To_v1alpha1_VMStatus

```
func Convert_ignite_VMStatus_To_v1alpha1_VMStatus(in *ignite.VMStatus, out *VMStatus,␣
↪s conversion.Scope) error
```

Convert_ignite_VMStatus_To_v1alpha1_VMStatus calls the autogenerated conversion function along with custom conversion logic

14.1.13 func Convert_v1alpha1_ImageSpec_To_ignite_ImageSpec

```
func Convert_v1alpha1_ImageSpec_To_ignite_ImageSpec(in *ImageSpec, out *ignite.
↪ImageSpec, s conversion.Scope) error
```

Convert_v1alpha1_ImageSpec_To_ignite_ImageSpec calls the autogenerated conversion function along with custom conversion logic

14.1.14 func Convert_v1alpha1_KernelSpec_To_ignite_KernelSpec

```
func Convert_v1alpha1_KernelSpec_To_ignite_KernelSpec(in *KernelSpec, out *ignite.
↪KernelSpec, s conversion.Scope) error
```

Convert_v1alpha1_KernelSpec_To_ignite_KernelSpec calls the autogenerated conversion function along with custom conversion logic

14.1.15 func Convert_v1alpha1_OCIClaim_To_ignite_OCI

```
func Convert_v1alpha1_OCIClaim_To_ignite_OCI(in *OCIImageClaim, out *meta.
↪OCIImageRef) error
```

14.1.16 func Convert_v1alpha1_OCIImageSource_To_ignite_OCIImageSource

```
func Convert_v1alpha1_OCIImageSource_To_ignite_OCIImageSource(in *OCIImageSource, out␣
↪*ignite.OCIImageSource, s conversion.Scope) (err error)
```

Convert_v1alpha1_OCIImageSource_To_ignite_OCIImageSource calls the autogenerated conversion function along with custom conversion logic

14.1.17 func Convert_v1alpha1_VMImageSpec_To_ignite_VMImageSpec

```
func Convert_v1alpha1_VMImageSpec_To_ignite_VMImageSpec(in *VMImageSpec, out *ignite.
↪VMImageSpec, s conversion.Scope) error
```

Convert_v1alpha1_VMImageSpec_To_ignite_VMImageSpec calls the autogenerated conversion function along with custom conversion logic

14.1.18 func Convert_v1alpha1_VMKernelSpec_To_ignite_VMKernelSpec

```
func Convert_v1alpha1_VMKernelSpec_To_ignite_VMKernelSpec(in *VMKernelSpec, out␣
↪*ignite.VMKernelSpec, s conversion.Scope) error
```

Convert_v1alpha1_VMKernelSpec_To_ignite_VMKernelSpec calls the autogenerated conversion function along with custom conversion logic

14.1.19 func Convert_v1alpha1_VMNetworkSpec_To_ignite_VMNetworkSpec

```
func Convert_v1alpha1_VMNetworkSpec_To_ignite_VMNetworkSpec(in *VMNetworkSpec, out␣
↪*ignite.VMNetworkSpec, s conversion.Scope) error
```

Convert_v1alpha1_VMNetworkSpec_To_ignite_VMNetworkSpec calls the autogenerated conversion function along with custom conversion logic

14.1.20 func Convert_v1alpha1_VMSpec_To_ignite_VMSpec

```
func Convert_v1alpha1_VMSpec_To_ignite_VMSpec(in *VMSpec, out *ignite.VMSpec, s␣
↪conversion.Scope) error
```

Convert_ignite_VMSpec_To_v1alpha1_VMSpec calls the autogenerated conversion function along with custom conversion logic

14.1.21 func Convert_v1alpha1_VMStatus_To_ignite_VMStatus

```
func Convert_v1alpha1_VMStatus_To_ignite_VMStatus(in *VMStatus, out *ignite.VMStatus,␣
↪s conversion.Scope) error
```

Convert_v1alpha1_VMStatus_To_ignite_VMStatus calls the autogenerated conversion function along with custom conversion logic

14.1.22 func SetDefaults_OCIImageClaim

```
func SetDefaults_OCIImageClaim(obj *OCIImageClaim)
```

14.1.23 func SetDefaults_PoolSpec

```
func SetDefaults_PoolSpec(obj *PoolSpec)
```

14.1.24 func SetDefaults_VMKernelSpec

```
func SetDefaults_VMKernelSpec(obj *VMKernelSpec)
```

14.1.25 func SetDefaults_VMNetworkSpec

```
func SetDefaults_VMNetworkSpec(obj *VMNetworkSpec)
```

14.1.26 func SetDefaults_VMSpec

```
func SetDefaults_VMSpec(obj *VMSpec)
```

14.1.27 func SetDefaults_VMStatus

```
func SetDefaults_VMStatus(obj *VMStatus)
```

14.1.28 type FileMapping

```
type FileMapping struct {
    HostPath string `json:"hostPath"`
    VMPath   string `json:"vmPath"`
}
```

FileMapping defines mappings between files on the host and VM

14.1.29 type Image

```
type Image struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`
```

(continues on next page)

(continued from previous page)

```
    Spec   ImageSpec   `json:"spec"`
    Status ImageStatus `json:"status"`
}
```

Image represents a cached OCI image ready to be used with Ignite +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.1.30 type ImageSourceType

```
type ImageSourceType string
```

ImageSourceType is an enum of different supported Image Source Types

```
const (
    // ImageSourceTypeDocker defines that the image is imported from Docker
    ImageSourceTypeDocker ImageSourceType = "Docker"
)
```

14.1.31 type ImageSpec

```
type ImageSpec struct {
    OCIClaim OCIImageClaim `json:"ociClaim"`
}
```

ImageSpec declares what the image contains

14.1.32 type ImageStatus

```
type ImageStatus struct {
    // OCISource contains the information about how this OCI image was imported
    OCISource OCIImageSource `json:"ociSource"`
}
```

ImageStatus defines the status of the image

14.1.33 type Kernel

```
type Kernel struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`

    Spec   KernelSpec   `json:"spec"`
    Status KernelStatus `json:"status"`
}
```

Kernel is a serializable object that caches information about imported kernels This file is stored in /var/lib/firecracker/kernels/{oci-image-digest}/metadata.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.1.34 type KernelSpec

```
type KernelSpec struct {
    OCIClaim OCIImageClaim `json:"ociClaim"`
}
```

KernelSpec describes the properties of a kernel

14.1.35 type KernelStatus

```
type KernelStatus struct {
    Version   string         `json:"version"`
    OCISource OCIImageSource `json:"ociSource"`
}
```

KernelStatus describes the status of a kernel

14.1.36 type NetworkMode

```
type NetworkMode string
```

NetworkMode defines different states a VM can be in

```
const (
    // NetworkModeCNI specifies the network mode where CNI is used
    NetworkModeCNI NetworkMode = "cni"
    // NetworkModeDockerBridge specifies the default docker bridge network is used
    NetworkModeDockerBridge NetworkMode = "docker-bridge"
)
```

func (NetworkMode) String

```
func (nm NetworkMode) String() string
```

14.1.37 type OCIImageClaim

```
type OCIImageClaim struct {
    // Type defines how the image should be imported
    Type ImageSourceType `json:"type"`
    // Ref defines the reference to use when talking to the backend.
    // This is most commonly the image name, followed by a tag.
    // Other supported ways are $registry/$user/$image@sha256:$digest
    // This ref is also used as ObjectMeta.Name for kinds Images and Kernels
    Ref meta.OCIImageRef `json:"ref"`
}
```

OCIImageClaim defines a claim for importing an OCI image

14.1.38 type OCIImageSource

```
type OCIImageSource struct {
    // ID defines the source's ID (e.g. the Docker image ID)
    ID string `json:"id"`
    // Size defines the size of the source in bytes
    Size meta.Size `json:"size"`
    // RepoDigests defines the image name as it was when pulled
    // from a repository, and the digest of the image
    // The format is $registry/$user/$image@sha256:$digest
    // This field is unpopulated if the image used as the source
    // has never been pushed to or pulled from a registry
    RepoDigests []string `json:"repoDigests,omitempty"`
}
```

OCIImageSource specifies how the OCI image was imported. It is the status variant of OCIImageClaim

14.1.39 type Pool

```
type Pool struct {
    runtime.TypeMeta `json:",inline"`

    Spec   PoolSpec   `json:"spec"`
    Status PoolStatus `json:"status"`
}
```

Pool defines device mapper pool database This file is managed by the snapshotter part of Ignite, and the file (existing as a singleton) is present at /var/lib/firecracker/snapshotter/pool.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.1.40 type PoolDevice

```
type PoolDevice struct {
    Size   meta.Size `json:"size"`
    Parent meta.DMID `json:"parent"`
    // Type specifies the type of the contents of the device
    Type PoolDeviceType `json:"type"`
    // MetadataPath points to the JSON/YAML file with metadata about this device
    // This is most often of the format /var/lib/firecracker/{type}/{id}/metadata.json
    MetadataPath string `json:"metadataPath"`
}
```

PoolDevice defines one device in the pool

14.1.41 type PoolDeviceType

```
type PoolDeviceType string
```

```
const (
    PoolDeviceTypeImage  PoolDeviceType = "Image"
    PoolDeviceTypeResize PoolDeviceType = "Resize"
    PoolDeviceTypeKernel PoolDeviceType = "Kernel"
    PoolDeviceTypeVM     PoolDeviceType = "VM"
)
```

14.1.42 type PoolSpec

```
type PoolSpec struct {
    // MetadataSize specifies the size of the pool's metadata
    MetadataSize meta.Size `json:"metadataSize"`
    // DataSize specifies the size of the pool's data
    DataSize meta.Size `json:"dataSize"`
    // AllocationSize specifies the smallest size that can be allocated at a time
    AllocationSize meta.Size `json:"allocationSize"`
    // MetadataPath points to the file where device mapper stores all metadata␣
↪information
    // Defaults to constants.SNAPSHOTTER_METADATA_PATH
    MetadataPath string `json:"metadataPath"`
    // DataPath points to the backing physical device or sparse file (to be loop␣
↪mounted) for the pool
    // Defaults to constants.SNAPSHOTTER_DATA_PATH
    DataPath string `json:"dataPath"`
}
```

PoolSpec defines the Pool's specification

14.1.43 type PoolStatus

```
type PoolStatus struct {
    // The Devices array needs to contain pointers to accommodate "holes" in the␣
↪mapping
    // Where devices have been deleted, the pointer is nil
    Devices []*PoolDevice `json:"devices"`
}
```

PoolStatus defines the Pool's current status

14.1.44 type SSH

```
type SSH struct {
    Generate  bool   `json:"-"`
    PublicKey string `json:"-"`
}
```

SSH specifies different ways to connect via SSH to the VM SSH uses a custom marshaller/unmarshaller. If generate is true, it marshals to true (a JSON bool). If PublicKey is set, it marshals to that string.

func (*SSH) MarshalJSON

```
func (s *SSH) MarshalJSON() ([]byte, error)
```

func (*SSH) UnmarshalJSON

```
func (s *SSH) UnmarshalJSON(b []byte) error
```

14.1.45 type VM

```
type VM struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`

    Spec   VMSpec   `json:"spec"`
    Status VMStatus `json:"status"`
}
```

VM represents a virtual machine run by Firecracker These files are stored in /var/lib/firecracker/vm/{vm-id}/metadata.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.1.46 type VMImageSpec

```
type VMImageSpec struct {
    OCIClaim OCIImageClaim `json:"ociClaim"`
}
```

14.1.47 type VMKernelSpec

```
type VMKernelSpec struct {
    OCIClaim OCIImageClaim `json:"ociClaim"`
    CmdLine  string        `json:"cmdLine,omitempty"`
}
```

14.1.48 type VMNetworkSpec

```
type VMNetworkSpec struct {
    Mode  NetworkMode       `json:"mode"`
    Ports meta.PortMappings `json:"ports,omitempty"`
}
```

14.1.49 type VMSpec

```
type VMSpec struct {
    Image    VMImageSpec    `json:"image"`
    Kernel   VMKernelSpec   `json:"kernel"`
    CPUs     uint64         `json:"cpus"`
    Memory   meta.Size      `json:"memory"`
    DiskSize meta.Size      `json:"diskSize"`
    Network  VMNetworkSpec `json:"network"`

    // This will be done at either "ignite start" or "ignite create" time
    // TODO: We might revisit this later
    CopyFiles []FileMapping `json:"copyFiles,omitempty"`
    // SSH specifies how the SSH setup should be done
    // nil here means "don't do anything special"
    // If SSH.Generate is set, Ignite will generate a new SSH key and copy it in to␣
↪authorized_keys in the VM
    // Specifying a path in SSH.Generate means "use this public key"
    // If SSH.PublicKey is set, this struct will marshal as a string using that path
    // If SSH.Generate is set, this struct will marshal as a bool => true
    SSH *SSH `json:"ssh,omitempty"`
}
```

VMSpec describes the configuration of a VM

14.1.50 type VMState

```
type VMState string
```

VMState defines different states a VM can be in

```
const (
    VMStateCreated VMState = "Created"
    VMStateRunning VMState = "Running"
    VMStateStopped VMState = "Stopped"
)
```

14.1.51 type VMStatus

```
type VMStatus struct {
    State       VMState          `json:"state"`
    IPAddresses meta.IPAddresses `json:"ipAddresses,omitempty"`
    Image       OCIImageSource   `json:"image"`
    Kernel      OCIImageSource   `json:"kernel"`
}
```

VMStatus defines the status of a VM

Generated by godoc2md

14.2 v1alpha2

```
import "github.com/weaveworks/ignite/pkg/apis/ignite/v1alpha2"
```

- *Overview*
- *Index*

14.2.1 Overview

+k8s:deepcopy-gen=package +k8s:defaulter-gen=TypeMeta +k8s:openapi-gen=true +k8s:conversion-gen=github.com/weaveworks/ignite/pkg/apis/ignite

14.2.2 Index

- *type VMKernelSpec*
- *type VMNetworkSpec*
- *type VMSpec*
- *type VMStatus*
- *type VMStorageSpec*
- *type Volume*
- *type VolumeMount*

Package files

defaults.go doc.go json.go register.go types.go

14.2.3 Constants

```
const (
    KindImage  runtime.Kind = "Image"
    KindKernel runtime.Kind = "Kernel"
    KindVM     runtime.Kind = "VM"
)
```

```
const (
    // GroupName is the group name use in this package
    GroupName = "ignite.weave.works"
)
```

14.2.4 Variables

```
var (
    // SchemeBuilder the schema builder
    SchemeBuilder = runtime.NewSchemeBuilder(
        addKnownTypes,
        addDefaultingFuncs,
    )

    AddToScheme = localSchemeBuilder.AddToScheme
)
```

```
var SchemeGroupVersion = schema.GroupVersion{
    Group:   GroupName,
    Version: "v1alpha2",
}
```

SchemeGroupVersion is group version used to register these objects

14.2.5 func SetDefaults_PoolSpec

```
func SetDefaults_PoolSpec(obj *PoolSpec)
```

14.2.6 func SetDefaults_VMKernelSpec

```
func SetDefaults_VMKernelSpec(obj *VMKernelSpec)
```

14.2.7 func SetDefaults_VMSpec

```
func SetDefaults_VMSpec(obj *VMSpec)
```

14.2.8 type BlockDeviceVolume

```
type BlockDeviceVolume struct {
    Path string `json:"path"`
}
```

BlockDeviceVolume defines a block device on the host

14.2.9 type FileMapping

```
type FileMapping struct {
    HostPath string `json:"hostPath"`
    VMPath   string `json:"vmPath"`
}
```

FileMapping defines mappings between files on the host and VM

14.2.10 type Image

```
type Image struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`

    Spec   ImageSpec   `json:"spec"`
    Status ImageStatus `json:"status"`
}
```

Image represents a cached OCI image ready to be used with Ignite +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.2.11 type ImageSpec

```
type ImageSpec struct {
    OCI meta.OCIImageRef `json:"oci"`
}
```

ImageSpec declares what the image contains

14.2.12 type ImageStatus

```
type ImageStatus struct {
    // OCISource contains the information about how this OCI image was imported
    OCISource OCIImageSource `json:"ociSource"`
}
```

ImageStatus defines the status of the image

14.2.13 type Kernel

```
type Kernel struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`

    Spec   KernelSpec   `json:"spec"`
    Status KernelStatus `json:"status"`
}
```

Kernel is a serializable object that caches information about imported kernels This file is stored in /var/lib/firecracker/kernels/{oci-image-digest}/metadata.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.2.14 type KernelSpec

```
type KernelSpec struct {
    OCI meta.OCIImageRef `json:"oci"`
}
```

KernelSpec describes the properties of a kernel

14.2.15 type KernelStatus

```
type KernelStatus struct {
    Version   string         `json:"version"`
    OCISource OCIImageSource `json:"ociSource"`
}
```

KernelStatus describes the status of a kernel

14.2.16 type OCIImageSource

```
type OCIImageSource struct {
    // ID defines the source's content ID (e.g. the canonical OCI path or Docker␣
↪image ID)
    ID *meta.OCIContentID `json:"id"`
    // Size defines the size of the source in bytes
    Size meta.Size `json:"size"`
}
```

OCIImageSource specifies how the OCI image was imported. It is the status variant of OCIImageClaim

14.2.17 type Pool

```
type Pool struct {
    runtime.TypeMeta `json:",inline"`

    Spec   PoolSpec   `json:"spec"`
    Status PoolStatus `json:"status"`
}
```

Pool defines device mapper pool database This file is managed by the snapshotter part of Ignite, and the file (existing as a singleton) is present at /var/lib/firecracker/snapshotter/pool.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.2.18 type PoolDevice

```
type PoolDevice struct {
    Size   meta.Size `json:"size"`
    Parent meta.DMID `json:"parent"`
    // Type specifies the type of the contents of the device
    Type PoolDeviceType `json:"type"`
    // MetadataPath points to the JSON/YAML file with metadata about this device
    // This is most often of the format /var/lib/firecracker/{type}/{id}/metadata.json
    MetadataPath string `json:"metadataPath"`
}
```

PoolDevice defines one device in the pool

14.2.19 type PoolDeviceType

```
type PoolDeviceType string
```

```
const (
    PoolDeviceTypeImage  PoolDeviceType = "Image"
    PoolDeviceTypeResize PoolDeviceType = "Resize"
    PoolDeviceTypeKernel PoolDeviceType = "Kernel"
    PoolDeviceTypeVM     PoolDeviceType = "VM"
)
```

14.2.20 type PoolSpec

```
type PoolSpec struct {
    // MetadataSize specifies the size of the pool's metadata
    MetadataSize meta.Size `json:"metadataSize"`
    // DataSize specifies the size of the pool's data
    DataSize meta.Size `json:"dataSize"`
    // AllocationSize specifies the smallest size that can be allocated at a time
    AllocationSize meta.Size `json:"allocationSize"`
    // MetadataPath points to the file where device mapper stores all metadata␣
↪information
    // Defaults to constants.SNAPSHOTTER_METADATA_PATH
    MetadataPath string `json:"metadataPath"`
    // DataPath points to the backing physical device or sparse file (to be loop␣
↪mounted) for the pool
    // Defaults to constants.SNAPSHOTTER_DATA_PATH
    DataPath string `json:"dataPath"`
}
```

PoolSpec defines the Pool's specification

14.2.21 type PoolStatus

```
type PoolStatus struct {
    // The Devices array needs to contain pointers to accommodate "holes" in the␣
↪mapping
    // Where devices have been deleted, the pointer is nil
    Devices []*PoolDevice `json:"devices"`
}
```

PoolStatus defines the Pool's current status

14.2.22 type Runtime

```
type Runtime struct {
    ID string `json:"id"`
}
```

Runtime specifies the VM's runtime information

14.2.23 type SSH

```
type SSH struct {
    Generate  bool   `json:"-"`
    PublicKey string `json:"-"`
}
```

SSH specifies different ways to connect via SSH to the VM SSH uses a custom marshaller/unmarshaller. If generate is true, it marshals to true (a JSON bool). If PublicKey is set, it marshals to that string.

func (*SSH) MarshalJSON

```
func (s *SSH) MarshalJSON() ([]byte, error)
```

func (*SSH) UnmarshalJSON

```
func (s *SSH) UnmarshalJSON(b []byte) error
```

14.2.24 type VM

```
type VM struct {
    runtime.TypeMeta `json:",inline"`
    // runtime.ObjectMeta is also embedded into the struct, and defines the human-
↪readable name, and the machine-readable ID
    // Name is available at the .metadata.name JSON path
    // ID is available at the .metadata.uid JSON path (the Go type is k8s.io/
↪apimachinery/pkg/types.UID, which is only a typed string)
    runtime.ObjectMeta `json:"metadata"`

    Spec   VMSpec   `json:"spec"`
    Status VMStatus `json:"status"`
}
```

VM represents a virtual machine run by Firecracker These files are stored in /var/lib/firecracker/vm/{vm-id}/metadata.json +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

14.2.25 type VMImageSpec

```
type VMImageSpec struct {
    OCI meta.OCIImageRef `json:"oci"`
}
```

14.2.26 type VMKernelSpec

```
type VMKernelSpec struct {
    OCI     meta.OCIImageRef `json:"oci"`
    CmdLine string           `json:"cmdLine,omitempty"`
}
```

14.2.27 type VMNetworkSpec

```
type VMNetworkSpec struct {
    Ports meta.PortMappings `json:"ports,omitempty"`
}
```

14.2.28 type VMSpec

```
type VMSpec struct {
    Image    VMImageSpec  `json:"image"`
    Kernel   VMKernelSpec `json:"kernel"`
    CPUs     uint64       `json:"cpus"`
    Memory   meta.Size    `json:"memory"`
    DiskSize meta.Size    `json:"diskSize"`
    // TODO: Implement working omitempty without pointers for the following entries
    // Currently both will show in the JSON output as empty arrays. Making them
    // pointers requires plenty of nil checks (as their contents are accessed␣
↪directly)
    // and is very risky for stability. APIMachinery potentially has a solution.
    Network VMNetworkSpec `json:"network,omitempty"`
    Storage VMStorageSpec `json:"storage,omitempty"`
    // This will be done at either "ignite start" or "ignite create" time
    // TODO: We might revisit this later
    CopyFiles []FileMapping `json:"copyFiles,omitempty"`
    // SSH specifies how the SSH setup should be done
    // nil here means "don't do anything special"
    // If SSH.Generate is set, Ignite will generate a new SSH key and copy it in to␣
↪authorized_keys in the VM
    // Specifying a path in SSH.Generate means "use this public key"
    // If SSH.PublicKey is set, this struct will marshal as a string using that path
    // If SSH.Generate is set, this struct will marshal as a bool => true
    SSH *SSH `json:"ssh,omitempty"`
}
```

VMSpec describes the configuration of a VM

14.2.29 type VMStatus

```
type VMStatus struct {
    Running     bool             `json:"running"`
    Runtime     *Runtime         `json:"runtime,omitempty"`
    StartTime   *runtime.Time    `json:"startTime,omitempty"`
    IPAddresses meta.IPAddresses `json:"ipAddresses,omitempty"`
    Image       OCIImageSource   `json:"image"`
    Kernel      OCIImageSource   `json:"kernel"`
}
```

VMStatus defines the status of a VM

14.2.30 type VMStorageSpec

```
type VMStorageSpec struct {
    Volumes      []Volume      `json:"volumes,omitempty"`
    VolumeMounts []VolumeMount `json:"volumeMounts,omitempty"`
}
```

VMStorageSpec defines the VM's Volumes and VolumeMounts

14.2.31 type Volume

```
type Volume struct {
    Name        string              `json:"name"`
    BlockDevice *BlockDeviceVolume `json:"blockDevice,omitempty"`
}
```

Volume defines named storage volume

14.2.32 type VolumeMount

```
type VolumeMount struct {
    Name      string `json:"name"`
    MountPath string `json:"mountPath"`
}
```

VolumeMount defines the mount point for a named volume inside a VM

Generated by godoc2md

14.3 v1alpha1

```
import "github.com/weaveworks/ignite/pkg/apis/meta/v1alpha1"
```

- *Overview*
- *Index*

14.3.1 Overview

+k8s:deepcopy-gen=package +k8s:openapi-gen=true

14.3.2 Index

- *Variables*
- *type DMID*
  - *func NewDMID(i int) DMID*
  - *func NewPoolDMID() DMID*
  - *func (d *DMID) Index() int*
  - *func (d *DMID) Pool() bool*
  - *func (d DMID) String() string*
- *type IPAddresses*
  - *func (i IPAddresses) String() string*
- *type OCIContentID*
  - *func ParseOCIContentID(str string) (*OCIContentID, error)*

  - *func (o *OCIContentID) Digest() digest.Digest*
  - *func (o *OCIContentID) Local() bool*
  - *func (o *OCIContentID) MarshalJSON() ([]byte, error)*
  - *func (o *OCIContentID) RepoDigest() (n reference.Named)*
  - *func (o *OCIContentID) SchemeString() string*
  - *func (o *OCIContentID) String() string*
  - *func (o *OCIContentID) UnmarshalJSON(b []byte) (err error)*
- *type OCIImageRef*
  - *func NewOCIImageRef(imageStr string) (o OCIImageRef, err error)*
  - *func (i OCIImageRef) IsUnset() bool*
  - *func (i OCIImageRef) MarshalJSON() ([]byte, error)*
  - *func (i OCIImageRef) Normalized() string*
  - *func (i OCIImageRef) Ref() reference.NamedTagged*
  - *func (i OCIImageRef) String() string*
  - *func (i *OCIImageRef) UnmarshalJSON(b []byte) (err error)*
- *type PortMapping*
  - *func (p PortMapping) String() string*
- *type PortMappings*
  - *func ParsePortMappings(input []string) (PortMappings, error)*
  - *func (p PortMappings) String() string*
- *type Protocol*
  - *func (p Protocol) String() string*
  - *func (p *Protocol) UnmarshalJSON(b []byte) (err error)*
- *type Size*
  - *func NewSizeFromBytes(bytes uint64) Size*
  - *func NewSizeFromSectors(sectors uint64) Size*
  - *func NewSizeFromString(str string) (Size, error)*
  - *func (s Size) Add(other Size) Size*
  - *func (s *Size) MarshalJSON() ([]byte, error)*
  - *func (s Size) Max(other Size) Size*
  - *func (s Size) Min(other Size) Size*
  - *func (s Size) Sectors() uint64*
  - *func (s Size) String() string*
  - *func (s *Size) UnmarshalJSON(b []byte) error*

Package files

dmid.go doc.go image.go net.go size.go

14.3.3 Variables

```
var EmptySize = NewSizeFromBytes(0)
```

14.3.4 type DMID

```
type DMID struct {
    // contains filtered or unexported fields
}
```

DMID specifies the format for device mapper IDs

func NewDMID

```
func NewDMID(i int) DMID
```

func NewPoolDMID

```
func NewPoolDMID() DMID
```

func (*DMID) Index

```
func (d *DMID) Index() int
```

func (*DMID) Pool

```
func (d *DMID) Pool() bool
```

func (DMID) String

```
func (d DMID) String() string
```

14.3.5 type IPAddresses

```
type IPAddresses []net.IP
```

IPAddresses represents a list of VM IP addresses

func (IPAddresses) String

```
func (i IPAddresses) String() string
```

14.3.6 type OCIContentID

```
type OCIContentID struct {
    // contains filtered or unexported fields
}
```

func ParseOCIContentID

```
func ParseOCIContentID(str string) (*OCIContentID, error)
```

ParseOCIContentID takes in a string to parse into an *OCIContentID If given a local Docker SHA like "sha256:3285f65b2651c68b5316e7a1fbabd30b5ae47914ac5791ac4bb9d59d029b924b", it will be parsed into the local format, encoded as "docker://". Given a full repo digest, such as "weaveworks/ignite-ubuntu@sha256:3285f65b2651c68b5316e7a1fbabd30b5ae47914ac5791ac4bb9d59d029b924b", it will be parsed into the OCI registry format, encoded as "oci://@".

func (*OCIContentID) Digest

```
func (o *OCIContentID) Digest() digest.Digest
```

Digest gets the digest of the content ID

func (*OCIContentID) Local

```
func (o *OCIContentID) Local() bool
```

Local returns true if the image has no repoName, i.e. it's not available from a registry

func (*OCIContentID) MarshalJSON

```
func (o *OCIContentID) MarshalJSON() ([]byte, error)
```

func (*OCIContentID) RepoDigest

```
func (o *OCIContentID) RepoDigest() (n reference.Named)
```

RepoDigest returns a repo digest based on the OCIContentID if it is not local

func (*OCIContentID) SchemeString

```
func (o *OCIContentID) SchemeString() string
```

Scheme returns the string representation with the scheme prefix

func (*OCIContentID) String

```
func (o *OCIContentID) String() string
```

String returns the string representation for either format

func (*OCIContentID) UnmarshalJSON

```
func (o *OCIContentID) UnmarshalJSON(b []byte) (err error)
```

14.3.7 type OCIImageRef

```
type OCIImageRef struct {
    // contains filtered or unexported fields
}
```

OCIImageRef is a struct containing a names and tagged reference by which an OCI runtime can identify an image to retrieve.

func NewOCIImageRef

```
func NewOCIImageRef(imageStr string) (o OCIImageRef, err error)
```

NewOCIImageRef parses and normalizes a reference to an OCI (docker) image.

func (OCIImageRef) IsUnset

```
func (i OCIImageRef) IsUnset() bool
```

func (OCIImageRef) MarshalJSON

```
func (i OCIImageRef) MarshalJSON() ([]byte, error)
```

func (OCIImageRef) Normalized

```
func (i OCIImageRef) Normalized() string
```

Normalized returns the normalized reference, e.g. “docker.io/weaveworks/ignite-ubuntu:latest”

func (OCIImageRef) Ref

```
func (i OCIImageRef) Ref() reference.NamedTagged
```

Ref parses the internal strings to a reference.NamedTagged

func (OCIImageRef) String

```
func (i OCIImageRef) String() string
```

String returns the familiar form of the reference, e.g. “weaveworks/ignite-ubuntu:latest”

func (*OCIImageRef) UnmarshalJSON

```
func (i *OCIImageRef) UnmarshalJSON(b []byte) (err error)
```

14.3.8 type PortMapping

```
type PortMapping struct {
    BindAddress net.IP    `json:"bindAddress,omitempty"`
    HostPort    uint64    `json:"hostPort"`
    VMPort      uint64    `json:"vmPort"`
    Protocol    Protocol `json:"protocol,omitempty"`
}
```

PortMapping defines a port mapping between the VM and the host

func (PortMapping) String

```
func (p PortMapping) String() string
```

14.3.9 type PortMappings

```
type PortMappings []PortMapping
```

PortMappings represents a list of port mappings

func ParsePortMappings

```
func ParsePortMappings(input []string) (PortMappings, error)
```

func (PortMappings) String

```
func (p PortMappings) String() string
```

14.3.10 type Protocol

```
type Protocol string
```

Protocol specifies a network port protocol

```
const (
    ProtocolTCP Protocol = "tcp"
    ProtocolUDP Protocol = "udp"
)
```

func (Protocol) String

```
func (p Protocol) String() string
```

func (*Protocol) UnmarshalJSON

```
func (p *Protocol) UnmarshalJSON(b []byte) (err error)
```

14.3.11 type Size

```
type Size struct {
    datasize.ByteSize
}
```

Size specifies a common unit for data sizes

func NewSizeFromBytes

```
func NewSizeFromBytes(bytes uint64) Size
```

func NewSizeFromSectors

```
func NewSizeFromSectors(sectors uint64) Size
```

func NewSizeFromString

```
func NewSizeFromString(str string) (Size, error)
```

func (Size) Add

```
func (s Size) Add(other Size) Size
```

Add returns a copy, does not modify the receiver

func (*Size) MarshalJSON

```
func (s *Size) MarshalJSON() ([]byte, error)
```

func (Size) Max

```
func (s Size) Max(other Size) Size
```

func (Size) Min

```
func (s Size) Min(other Size) Size
```

func (Size) Sectors

```
func (s Size) Sectors() uint64
```

func (Size) String

```
func (s Size) String() string
```

Override ByteSize's default string implementation which results in .HR() without spaces

func (*Size) UnmarshalJSON

```
func (s *Size) UnmarshalJSON(b []byte) error
```

Generated by godoc2md

CHAPTER

FIFTEEN

IGNITE

ignite: easily run Firecracker VMs

15.1 ignite

ignite: easily run Firecracker VMs

15.1.1 Synopsis

Ignite is a containerized Firecracker microVM administration tool. It can build VM images, spin VMs up/down and manage multiple VMs efficiently.

Administration is divided into three subcommands: image Manage base images for VMs kernel Manage VM kernels vm Manage VMs

Ignite also supports the same commands as the Docker CLI. Combining an Image and a Kernel gives you a runnable VM.

Example usage:

```
$ ignite run weaveworks/ignite-ubuntu \
        --cpus 2 \
        --memory 2GB \
        --ssh \
        --name my-vm
$ ignite images
$ ignite kernels
$ ignite ps
$ ignite logs my-vm
$ ignite ssh my-vm
```

15.1.2 Options

```
  -h, --help                    help for ignite
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.1.3 SEE ALSO

- ignite attach - Attach to a running VM
- ignite completion - Output bash completion for ignite to stdout
- ignite create - Create a new VM without starting it
- ignite exec - execute a command in a running VM
- ignite image - Manage base images for VMs
- ignite inspect - Inspect an Ignite Object
- ignite kernel - Manage VM kernels
- ignite kill - Kill running VMs
- ignite logs - Get the logs for a running VM
- ignite ps - List running VMs
- ignite rm - Remove VMs
- ignite rmi - Remove VM base images
- ignite rmk - Remove kernels
- ignite run - Create a new VM and start it
- ignite ssh - SSH into a running vm
- ignite start - Start a VM
- ignite stop - Stop running VMs
- ignite version - Print the version of ignite
- ignite vm - Manage VMs

15.2 ignite attach

Attach to a running VM

15.2.1 Synopsis

Connect the current terminal to the running VM's TTY. To detach from the VM's TTY, type ^P^Q (Ctrl + P + Q). The given VM is matched by prefix based on its ID and name.

```
ignite attach <vm> [flags]
```

15.2.2 Options

```
  -h, --help   help for attach
```

15.2.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.2.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.3 ignite completion

Output bash completion for ignite to stdout

15.3.1 Synopsis

In order to start using the auto-completion, run:

```
. <(ignite completion)
```

To configure your bash shell to load completions for each session, run:

```
echo '. <(ignite completion)' >> ~/.bashrc
```

```
ignite completion [flags]
```

15.3.2 Options

```
  -h, --help   help for completion
```

15.3.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.3.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.4 ignite create

Create a new VM without starting it

15.4.1 Synopsis

Create a new VM by combining the given image with a kernel. If no kernel is given using the kernel flag (-k, –kernel-image), use the default kernel (weaveworks/ignite-kernel:4.19.47).

Various configuration options can be set during creation by using the flags for this command.

If the name flag (-n, –name) is not specified, the VM is given a random name. Using the copy files flag (-f, –copy-files), additional files/directories can be added to the VM during creation with the syntax /host/path:/vm/path.

Example usage: $ ignite create weaveworks/ignite-ubuntu –name my-vm –cpus 2 –ssh –memory 2GB –size 6GB

```
ignite create <OCI image> [flags]
```

15.4.2 Options

```
      --config string            Specify a path to a file with the API resources you␣
↪want to pass
  -f, --copy-files strings       Copy files/directories from the host to the created␣
↪VM
      --cpus uint                VM vCPU count, 1 or even numbers between 1 and 32␣
↪(default 1)
  -h, --help                     help for create
      --kernel-args string       Set the command line for the kernel (default␣
↪"console=ttyS0 reboot=k panic=1 pci=off ip=dhcp")
  -k, --kernel-image oci-image   Specify an OCI image containing the kernel at /boot/␣
↪vmlinux and optionally, modules (default weaveworks/ignite-kernel:4.19.47)
      --memory size              Amount of RAM to allocate for the VM (default 512.0␣
↪MB)
  -n, --name string              Specify the name
  -p, --ports strings            Map host ports to VM ports
  -s, --size size                VM filesystem size, for example 5GB or 2048MB␣
↪(default 4.0 GB)
      --ssh[=<path>]             Enable SSH for the VM. If <path> is given, it will␣
↪be imported as the public key. If just '--ssh' is specified, a new keypair will be␣
↪generated. (default is unset, which disables SSH access to the VM)
  -v, --volumes volume           Expose block devices from the host inside the VM
```

15.4.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.4.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.5 ignite exec

execute a command in a running VM

15.5.1 Synopsis

Execute a command in a running VM using SSH and the private key created for it during generation. If no private key was created or wanting to use a different identity file, use the identity file flag (-i, –identity) to override the used identity file. The given VM is matched by prefix based on its ID and name.

```
ignite exec <vm> <command...> [flags]
```

15.5.2 Options

```
  -h, --help              help for exec
  -i, --identity string   Override the vm's default identity file
  -t, --timeout uint32    Timeout waiting for connection in seconds (default 10)
```

15.5.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.5.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.6 ignite gitops

Run the GitOps feature of Ignite

15.6.1 Synopsis

Run Ignite in GitOps mode watching the given repository. The repository needs to be publicly cloneable. Ignite will watch for changes in the master branch by default, overridable with the branch flag (-b, –branch). If any new/changed

VM specification files are found in the repo (in JSON/YAML format), their configuration will automatically be declaratively applied.

To quit GitOps mode, use (Ctrl + C).

```
ignite gitops <repo-url> [flags]
```

15.6.2 Options

```
  -b, --branch string   What branch to sync (default "master")
  -h, --help            help for gitops
  -p, --paths strings   What subdirectories to care about. Default the whole␣
↪repository
```

15.6.3 Options inherited from parent commands

```
      --log-level loglevel   Specify the loglevel for the program (default info)
  -q, --quiet                The quiet mode allows for machine-parsable output, by␣
↪printing only IDs
```

15.6.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.7 ignite image

Manage base images for VMs

15.7.1 Synopsis

Groups together functionality for managing VM base images. Calling this command alone lists all available images.

```
ignite image [flags]
```

15.7.2 Options

```
  -h, --help   help for image
```

15.7.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
```

(continues on next page)

(continued from previous page)

```
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.7.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs
- ignite image import - Import a new base image for VMs
- ignite image ls - List available VM base images
- ignite image rm - Remove VM base images

15.8 ignite image import

Import a new base image for VMs

15.8.1 Synopsis

Import an OCI image as a base image for VMs, takes in a Docker image identifier. This importing is done automatically when the "run" or "create" commands are run. The import step is essentially a cache for images to be used later when running VMs.

```
ignite image import <OCI image> [flags]
```

15.8.2 Options

```
  -h, --help   help for import
```

15.8.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.8.4 SEE ALSO

- ignite image - Manage base images for VMs

15.9 ignite image ls

List available VM base images

15.9.1 Synopsis

List all available VM base images. Outputs the same as the parent command.

```
ignite image ls [flags]
```

15.9.2 Options

```
  -h, --help   help for ls
```

15.9.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.9.4 SEE ALSO

- ignite image - Manage base images for VMs

15.10 ignite image rm

Remove VM base images

15.10.1 Synopsis

Remove one or multiple VM base images. Images are matched by prefix based on their ID and name. To remove multiple images, chain the matches separated by spaces. The force flag (-f, –force) kills and removes any running VMs using the image.

```
ignite image rm <image>... [flags]
```

15.10.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rm
```

15.10.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.10.4 SEE ALSO

- ignite image - Manage base images for VMs

15.11 ignite inspect

Inspect an Ignite Object

15.11.1 Synopsis

Retrieve information about the given object of the given kind. The kind can be "image", "kernel" or "vm". The object is matched by prefix based on its ID and name. Outputs JSON by default, can be overridden with the output flag (-o, –output).

```
ignite inspect <kind> <object> [flags]
```

15.11.2 Options

```
  -h, --help            help for inspect
  -o, --output string   Output the object in the specified format (default "json")
```

15.11.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.11.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.12 ignite kernel

Manage VM kernels

15.12.1 Synopsis

Groups together functionality for managing VM kernels. Calling this command alone lists all available kernels.

```
ignite kernel [flags]
```

15.12.2 Options

```
  -h, --help   help for kernel
```

15.12.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.12.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs
- ignite kernel import - Import a kernel image from an OCI image
- ignite kernel ls - List available VM kernels
- ignite kernel rm - Remove kernels

15.13 ignite kernel import

Import a kernel image from an OCI image

15.13.1 Synopsis

Import an OCI image as a kernel image for VMs, takes in a Docker image identifier. This importing is done automatically when the "run" or "create" commands are run. The import step is essentially a cache for images to be used later when running VMs.

```
ignite kernel import <OCI image> [flags]
```

15.13.2 Options

```
  -h, --help   help for import
```

15.13.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.13.4 SEE ALSO

- ignite kernel - Manage VM kernels

15.14 ignite kernel ls

List available VM kernels

15.14.1 Synopsis

List all available VM kernels. Outputs the same as the parent command.

```
ignite kernel ls [flags]
```

15.14.2 Options

```
  -h, --help   help for ls
```

15.14.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.14.4 SEE ALSO

- ignite kernel - Manage VM kernels

15.15 ignite kernel rm

Remove kernels

15.15.1 Synopsis

Remove one or multiple VM kernels. Kernels are matched by prefix based on their ID and name. To remove multiple kernels, chain the matches separated by spaces. The force flag (-f, –force) kills and removes any running VMs using the kernel.

```
ignite kernel rm <kernel>... [flags]
```

15.15.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rm
```

15.15.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.15.4 SEE ALSO

- ignite kernel - Manage VM kernels

15.16 ignite kill

Kill running VMs

15.16.1 Synopsis

Kill (force stop) one or multiple VMs. The VMs are matched by prefix based on their ID and name. To kill multiple VMs, chain the matches separated by spaces.

```
ignite kill <vm>... [flags]
```

15.16.2 Options

```
  -h, --help   help for kill
```

15.16.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.16.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.17 ignite logs

Get the logs for a running VM

15.17.1 Synopsis

Show the logs for the given VM. The VM needs to be running (its backing container needs to exist). The VM is matched by prefix based on its ID and name.

```
ignite logs <vm> [flags]
```

15.17.2 Options

```
  -h, --help   help for logs
```

15.17.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.17.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.18 ignite ps

List running VMs

15.18.1 Synopsis

List all running VMs. By specifying the all flag (-a, –all), also list VMs that are not currently running.

```
ignite ps [flags]
```

15.18.2 Options

```
  -a, --all    Show all VMs, not just running ones
  -h, --help   help for ps
```

15.18.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.18.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.19 ignite rm

Remove VMs

15.19.1 Synopsis

Remove one or multiple VMs. The VMs are matched by prefix based on their ID and name. To remove multiple VMs, chain the matches separated by spaces. The force flag (-f, –force) kills running VMs before removal instead of throwing an error.

```
ignite rm <vm>... [flags]
```

15.19.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rm
```

15.19.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.19.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.20 ignite rmi

Remove VM base images

15.20.1 Synopsis

Remove one or multiple VM base images. Images are matched by prefix based on their ID and name. To remove multiple images, chain the matches separated by spaces. The force flag (-f, –force) kills and removes any running VMs using the image.

```
ignite rmi <image> [flags]
```

15.20.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rmi
```

15.20.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.20.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.21 ignite rmk

Remove kernels

15.21.1 Synopsis

Remove one or multiple VM kernels. Kernels are matched by prefix based on their ID and name. To remove multiple kernels, chain the matches separated by spaces. The force flag (-f, –force) kills and removes any running VMs using the kernel.

```
ignite rmk <kernel> [flags]
```

15.21.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rmk
```

15.21.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.21.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.22 ignite run

Create a new VM and start it

15.22.1 Synopsis

Create and start a new VM immediately. The image (and kernel) is matched by prefix based on its ID and name. This command accepts all flags used to create and start a VM. The interactive flag (-i, –interactive) can be specified to immediately attach to the started VM after creation.

Example usage: $ ignite run weaveworks/ignite-ubuntu –interactive –name my-vm –cpus 2 –ssh –memory 2GB –size 10G

```
ignite run <OCI image> [flags]
```

15.22.2 Options

```
      --config string            Specify a path to a file with the API resources you
↪want to pass
  -f, --copy-files strings       Copy files/directories from the host to the created
↪VM
      --cpus uint                VM vCPU count, 1 or even numbers between 1 and 32
↪(default 1)
  -d, --debug                    Debug mode, keep container after VM shutdown
  -h, --help                     help for run
  -i, --interactive              Attach to the VM after starting
      --kernel-args string       Set the command line for the kernel (default
↪"console=ttyS0 reboot=k panic=1 pci=off ip=dhcp")
  -k, --kernel-image oci-image   Specify an OCI image containing the kernel at /boot/
↪vmlinux and optionally, modules (default weaveworks/ignite-kernel:4.19.47)
      --memory size              Amount of RAM to allocate for the VM (default 512.0
↪MB)
  -n, --name string              Specify the name
  -p, --ports strings            Map host ports to VM ports
  -s, --size size                VM filesystem size, for example 5GB or 2048MB
↪(default 4.0 GB)
      --ssh[=<path>]             Enable SSH for the VM. If <path> is given, it will
↪be imported as the public key. If just '--ssh' is specified, a new keypair will be
↪generated. (default is unset, which disables SSH access to the VM)
  -v, --volumes volume           Expose block devices from the host inside the VM
```

15.22.3 Options inherited from parent commands

```
      --log-level loglevel       Specify the loglevel for the program (default info)
      --network-plugin plugin    Network plugin to use. Available options are: [cni
↪docker-bridge] (default cni)
  -q, --quiet                    The quiet mode allows for machine-parsable output by
↪printing only IDs
      --runtime runtime          Container runtime to use. Available options are:
↪[docker containerd] (default containerd)
```

15.22.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.23 ignite ssh

SSH into a running vm

15.23.1 Synopsis

SSH into the running VM using the private key created for it during generation. If no private key was created or wanting to use a different identity file, use the identity file flag (-i, –identity) to override the used identity file. The given VM is matched by prefix based on its ID and name.

```
ignite ssh <vm> [flags]
```

15.23.2 Options

```
  -h, --help              help for ssh
  -i, --identity string   Override the vm's default identity file
  -t, --timeout uint32    Timeout waiting for connection in seconds (default 10)
```

15.23.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.23.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.24 ignite start

Start a VM

15.24.1 Synopsis

Start the given VM. The VM is matched by prefix based on its ID and name. If the interactive flag (-i, –interactive) is specified, attach to the VM after starting.

```
ignite start <vm> [flags]
```

15.24.2 Options

```
  -d, --debug         Debug mode, keep container after VM shutdown
  -h, --help          help for start
  -i, --interactive   Attach to the VM after starting
```

15.24.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.24.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.25 ignite stop

Stop running VMs

15.25.1 Synopsis

Stop one or multiple VMs. The VMs are matched by prefix based on their ID and name. To stop multiple VMs, chain the matches separated by spaces. The force flag (-f, –force) kills VMs instead of trying to stop them gracefully.

The VMs are given a 20 second grace period to shut down before they will be forcibly killed.

```
ignite stop <vm>... [flags]
```

15.25.2 Options

```
  -f, --force-kill   Force kill the VM
  -h, --help         help for stop
```

15.25.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.25.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.26 ignite version

Print the version of ignite

15.26.1 Synopsis

Print the version of ignite

```
ignite version [flags]
```

15.26.2 Options

```
  -h, --help            help for version
  -o, --output string   Output format; available options are 'yaml', 'json' and 'short
↪'
```

15.26.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.26.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs

15.27 ignite vm

Manage VMs

15.27.1 Synopsis

Groups together functionality for managing VMs.

```
ignite vm [flags]
```

15.27.2 Options

```
  -h, --help   help for vm
```

15.27.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.27.4 SEE ALSO

- ignite - ignite: easily run Firecracker VMs
- ignite vm attach - Attach to a running VM
- ignite vm create - Create a new VM without starting it
- ignite vm kill - Kill running VMs
- ignite vm logs - Get the logs for a running VM
- ignite vm ps - List running VMs
- ignite vm rm - Remove VMs
- ignite vm run - Create a new VM and start it
- ignite vm ssh - SSH into a running vm
- ignite vm start - Start a VM
- ignite vm stop - Stop running VMs

15.28 ignite vm attach

Attach to a running VM

15.28.1 Synopsis

Connect the current terminal to the running VM's TTY. To detach from the VM's TTY, type ^P^Q (Ctrl + P + Q). The given VM is matched by prefix based on its ID and name.

```
ignite vm attach <vm> [flags]
```

15.28.2 Options

```
  -h, --help   help for attach
```

15.28.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.28.4 SEE ALSO

- ignite vm - Manage VMs

15.29 ignite vm create

Create a new VM without starting it

15.29.1 Synopsis

Create a new VM by combining the given image with a kernel. If no kernel is given using the kernel flag (-k, –kernel-image), use the default kernel (weaveworks/ignite-kernel:4.19.47).

Various configuration options can be set during creation by using the flags for this command.

If the name flag (-n, –name) is not specified, the VM is given a random name. Using the copy files flag (-f, –copy-files), additional files/directories can be added to the VM during creation with the syntax /host/path:/vm/path.

Example usage: $ ignite create weaveworks/ignite-ubuntu –name my-vm –cpus 2 –ssh –memory 2GB –size 6GB

```
ignite vm create <OCI image> [flags]
```

15.29.2 Options

```
      --config string           Specify a path to a file with the API resources you␣
↪want to pass
  -f, --copy-files strings      Copy files/directories from the host to the created␣
↪VM
      --cpus uint               VM vCPU count, 1 or even numbers between 1 and 32␣
↪(default 1)
  -h, --help                    help for create
      --kernel-args string      Set the command line for the kernel (default␣
↪"console=ttyS0 reboot=k panic=1 pci=off ip=dhcp")
  -k, --kernel-image oci-image  Specify an OCI image containing the kernel at /boot/␣
↪vmlinux and optionally, modules (default weaveworks/ignite-kernel:4.19.47)
      --memory size             Amount of RAM to allocate for the VM (default 512.0␣
↪MB)
  -n, --name string             Specify the name
  -p, --ports strings           Map host ports to VM ports
  -s, --size size               VM filesystem size, for example 5GB or 2048MB␣
↪(default 4.0 GB)
```

(continues on next page)

(continued from previous page)

```
      --ssh[=<path>]              Enable SSH for the VM. If <path> is given, it will␣
↪be imported as the public key. If just '--ssh' is specified, a new keypair will be␣
↪generated. (default is unset, which disables SSH access to the VM)
  -v, --volumes volume            Expose block devices from the host inside the VM
```

15.29.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.29.4 SEE ALSO

- ignite vm - Manage VMs

15.30 ignite vm kill

Kill running VMs

15.30.1 Synopsis

Kill (force stop) one or multiple VMs. The VMs are matched by prefix based on their ID and name. To kill multiple VMs, chain the matches separated by spaces.

```
ignite vm kill <vm>... [flags]
```

15.30.2 Options

```
  -h, --help   help for kill
```

15.30.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.30.4 SEE ALSO

- ignite vm - Manage VMs

15.31 ignite vm logs

Get the logs for a running VM

15.31.1 Synopsis

Show the logs for the given VM. The VM needs to be running (its backing container needs to exist). The VM is matched by prefix based on its ID and name.

```
ignite vm logs <vm> [flags]
```

15.31.2 Options

```
  -h, --help   help for logs
```

15.31.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.31.4 SEE ALSO

- ignite vm - Manage VMs

15.32 ignite vm ps

List running VMs

15.32.1 Synopsis

List all running VMs. By specifying the all flag (-a, –all), also list VMs that are not currently running.

```
ignite vm ps [flags]
```

15.32.2 Options

```
  -a, --all    Show all VMs, not just running ones
  -h, --help   help for ps
```

15.32.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.32.4 SEE ALSO

- ignite vm - Manage VMs

15.33 ignite vm rm

Remove VMs

15.33.1 Synopsis

Remove one or multiple VMs. The VMs are matched by prefix based on their ID and name. To remove multiple VMs, chain the matches separated by spaces. The force flag (-f, –force) kills running VMs before removal instead of throwing an error.

```
ignite vm rm <vm>... [flags]
```

15.33.2 Options

```
  -f, --force   Force this operation. Warning, use of this mode may have unintended␣
↪consequences.
  -h, --help    help for rm
```

15.33.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.33.4 SEE ALSO

- ignite vm - Manage VMs

15.34 ignite vm run

Create a new VM and start it

15.34.1 Synopsis

Create and start a new VM immediately. The image (and kernel) is matched by prefix based on its ID and name. This command accepts all flags used to create and start a VM. The interactive flag (-i, –interactive) can be specified to immediately attach to the started VM after creation.

Example usage: $ ignite run weaveworks/ignite-ubuntu –interactive –name my-vm –cpus 2 –ssh –memory 2GB –size 10G

```
ignite vm run <OCI image> [flags]
```

15.34.2 Options

```
      --config string            Specify a path to a file with the API resources you␣
↪want to pass
  -f, --copy-files strings       Copy files/directories from the host to the created␣
↪VM
      --cpus uint                VM vCPU count, 1 or even numbers between 1 and 32␣
↪(default 1)
  -d, --debug                    Debug mode, keep container after VM shutdown
  -h, --help                     help for run
  -i, --interactive              Attach to the VM after starting
      --kernel-args string       Set the command line for the kernel (default
↪"console=ttyS0 reboot=k panic=1 pci=off ip=dhcp")
  -k, --kernel-image oci-image   Specify an OCI image containing the kernel at /boot/
↪vmlinux and optionally, modules (default weaveworks/ignite-kernel:4.19.47)
      --memory size              Amount of RAM to allocate for the VM (default 512.0␣
↪MB)
  -n, --name string              Specify the name
  -p, --ports strings            Map host ports to VM ports
  -s, --size size                VM filesystem size, for example 5GB or 2048MB␣
↪(default 4.0 GB)
      --ssh[=<path>]             Enable SSH for the VM. If <path> is given, it will␣
↪be imported as the public key. If just '--ssh' is specified, a new keypair will be␣
↪generated. (default is unset, which disables SSH access to the VM)
  -v, --volumes volume           Expose block devices from the host inside the VM
```

15.34.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
```

(continues on next page)

(continued from previous page)

```
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.34.4 SEE ALSO

- ignite vm - Manage VMs

15.35 ignite vm ssh

SSH into a running vm

15.35.1 Synopsis

SSH into the running VM using the private key created for it during generation. If no private key was created or wanting to use a different identity file, use the identity file flag (-i, –identity) to override the used identity file. The given VM is matched by prefix based on its ID and name.

```
ignite vm ssh <vm> [flags]
```

15.35.2 Options

```
  -h, --help              help for ssh
  -i, --identity string   Override the vm's default identity file
  -t, --timeout uint32    Timeout waiting for connection in seconds (default 10)
```

15.35.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.35.4 SEE ALSO

- ignite vm - Manage VMs

15.36 ignite vm start

Start a VM

15.36.1 Synopsis

Start the given VM. The VM is matched by prefix based on its ID and name. If the interactive flag (-i, –interactive) is specified, attach to the VM after starting.

```
ignite vm start <vm> [flags]
```

15.36.2 Options

```
  -d, --debug         Debug mode, keep container after VM shutdown
  -h, --help          help for start
  -i, --interactive   Attach to the VM after starting
```

15.36.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.36.4 SEE ALSO

- ignite vm - Manage VMs

15.37 ignite vm stop

Stop running VMs

15.37.1 Synopsis

Stop one or multiple VMs. The VMs are matched by prefix based on their ID and name. To stop multiple VMs, chain the matches separated by spaces. The force flag (-f, –force) kills VMs instead of trying to stop them gracefully.

The VMs are given a 20 second grace period to shut down before they will be forcibly killed.

```
ignite vm stop <vm>... [flags]
```

15.37.2 Options

```
  -f, --force-kill   Force kill the VM
  -h, --help         help for stop
```

15.37.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
  -q, --quiet                   The quiet mode allows for machine-parsable output by␣
↪printing only IDs
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

15.37.4 SEE ALSO

- ignite vm - Manage VMs

CHAPTER

SIXTEEN

IGNITED

ignited: run Firecracker VMs declaratively through a manifest directory or Git

16.1 ignited

ignited: run Firecracker VMs declaratively through a manifest directory or Git

16.1.1 Synopsis

Ignite is a containerized Firecracker microVM administration tool. It can build VM images, spin VMs up/down and manage multiple VMs efficiently.

TODO: ignited documentation

16.1.2 Options

```
  -h, --help                     help for ignited
      --log-level loglevel       Specify the loglevel for the program (default info)
      --network-plugin plugin    Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
      --runtime runtime          Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

16.1.3 SEE ALSO

- ignited completion - Output bash completion for ignited to stdout
- ignited daemon - Operates in daemon mode and watches /etc/firecracker/manifests for VM specifications to run.
- ignited gitops - Run the GitOps feature of Ignite
- ignited version - Print the version of ignite

16.2 ignited completion

Output bash completion for ignited to stdout

16.2.1 Synopsis

In order to start using the auto-completion, run:

```
. <(ignited completion)
```

To configure your bash shell to load completions for each session, run:

```
echo '. <(ignited completion)' >> ~/.bashrc
```

```
ignited completion [flags]
```

16.2.2 Options

```
  -h, --help   help for completion
```

16.2.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

16.2.4 SEE ALSO

- ignited - ignited: run Firecracker VMs declaratively through a manifest directory or Git

16.3 ignited daemon

Operates in daemon mode and watches /etc/firecracker/manifests for VM specifications to run.

16.3.1 Synopsis

Operates in daemon mode and watches /etc/firecracker/manifests for VM specifications to run.

```
ignited daemon [flags]
```

16.3.2 Options

```
  -h, --help   help for daemon
```

16.3.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

16.3.4 SEE ALSO

- ignited - ignited: run Firecracker VMs declaratively through a manifest directory or Git

16.4 ignited gitops

Run the GitOps feature of Ignite

16.4.1 Synopsis

Run Ignite in GitOps mode watching the given repository. The repository needs to be publicly cloneable. Ignite will watch for changes in the master branch by default, overridable with the branch flag (-b, –branch). If any new/changed VM specification files are found in the repo (in JSON/YAML format), their configuration will automatically be declaratively applied.

To quit GitOps mode, use (Ctrl + C).

```
ignited gitops <repo-url> [flags]
```

16.4.2 Options

```
  -b, --branch string   What branch to sync (default "master")
  -h, --help            help for gitops
  -p, --paths strings   What subdirectories to care about. Default the whole␣
↪repository
```

16.4.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

16.4.4 SEE ALSO

- ignited - ignited: run Firecracker VMs declaratively through a manifest directory or Git

16.5 ignited version

Print the version of ignite

16.5.1 Synopsis

Print the version of ignite

```
ignited version [flags]
```

16.5.2 Options

```
  -h, --help            help for version
  -o, --output string   Output format; available options are 'yaml', 'json' and 'short
↪'
```

16.5.3 Options inherited from parent commands

```
      --log-level loglevel      Specify the loglevel for the program (default info)
      --network-plugin plugin   Network plugin to use. Available options are: [cni␣
↪docker-bridge] (default cni)
      --runtime runtime         Container runtime to use. Available options are:␣
↪[docker containerd] (default containerd)
```

16.5.4 SEE ALSO

- ignited - ignited: run Firecracker VMs declaratively through a manifest directory or Git

CHAPTER

SEVENTEEN

V0.1.0

This is the first, proof-of-concept version of Ignite. It has all the essential features, and a pretty complete implementation of the docker UX.

CHAPTER

EIGHTEEN

V0.2.0

Major release with significant improvements

- Ignite is now using `devicemapper` under the hood, for overlay snapshots for filesystem writes, allowing for image reuse, efficient use of space and way faster builds!
- Added sample Ubuntu 18.04 and CentOS 7 OS images & a 4.19 kernel build
- Automatic network configuration, now the OS image doesn't need to enable DHCP, as that is done in the kernel
- Automatically populate `/etc/hosts` and `/etc/resolv.conf`, too
- Add an option to bind a port exposed by the VM to a host port (`ignite run -p 80:80`)
- Add an option for modifying the kernel command line (`ignite run --kernel-args`)
- Add an option to copy files from the host into the VM (`ignite run --copy-files`)
- Add an option to specify the amount of cores, RAM, and overlay size (`ignite run --cpus 2 --memory 1024 --size 4GB`)
- Removed the need for the Ignite container to run with `--privileged`
- Allow for force-deletions of images, kernels and vms.
- Added documentation.
- Moved repo from `luxas/ignite` to `weaveworks/ignite`

CHAPTER

NINETEEN

V0.3.0

Major release with significant UX and internal improvements:

- There is no longer a difference between an Ignite image and an OCI image, this is now the same thing.
  - Ignite operates on OCI images directly, for both OS images and kernels. The kernel is expected to be coupled with the image given to `ignite run`, in `/boot/vmlinux`.
- It is now possible to do `ignite run [OCI image]` directly, and everything (e.g. pulling the image) is handled automatically. e.g. `ignite run -i weaveworks/ignite-ubuntu`.
- Now `ignite images` shows OCI images that are cached and ready to use, and `ignite kernels` the kernels already imported from base images.
- Added an example usage guide for running a Kubernetes cluster in HA mode using kubeadm and Ignite.
- Removed `ignite build`, and `ignite image/kernel import`; as these are no longer needed
- Importing an image from a tROADMAPar file is no longer possible, package the contents in an OCI image instead
- Added a new command `ignite ssh [vm]` and flag: `ignite run --ssh`. This allows for automatic SSH logins.
- Now Ignite logs user-friendly messages by default. To get machine-readable output, use the `--quiet` flag.
- Ignite now requires the user to be `root`. This will be revisited later, when the architecture has changed.
- The command outputs and structure is now more user-friendly.
- Fixed several bugs both under the hood, and user-affecting ones

CHAPTER

TWENTY

V0.4.0

The first release candidate for Ignite's biggest release yet!

There are many significant changes compared to before:

20.1 New Features

- Make base and kernel OCI images composable for a VM. You can now choose what kernel to combine with what base image freely https://github.com/weaveworks/ignite/pull/105
- Add the GitOps mode for Ignite using `ignite gitops` https://github.com/weaveworks/ignite/pull/100
  - Documentation: https://github.com/weaveworks/ignite/blob/master/gitops
- Make it possible to run `ignite create` and `ignite run` declaratively https://github.com/weaveworks/ignite/commit/57333646b52a0e1e3a725340e994b2749b39e5bd
  - Documentation: https://github.com/weaveworks/ignite/blob/master/docs/declarative-config.md
- Added Prometheus metrics for `ignite-spawn` https://github.com/weaveworks/ignite/commit/94abc529972873db3fa3ee954099a4f62d67b6f3
  - Documentation: https://github.com/weaveworks/ignite/blob/master/docs/prometheus.md
- Implemented CNI support https://github.com/weaveworks/ignite/commit/a8897532f9f6a8f5c40025f0f93ab2d24f2c7cd3

20.2 API Machinery

- Added the `ignite.weave.works/v1alpha1` API group with the Ignite API types https://github.com/weaveworks/ignite/commit/ca1edc8e7a61b950811c6145ba2ad53f8cdc2a04
  - API reference: https://github.com/weaveworks/ignite/blob/master/api/ignite.md
  - This API version will not change in a future version. When improvements are made, it will be to `v1alpha2` etc.
- Add a meta API package containing supporting but generic API types for Ignite https://github.com/weaveworks/ignite/commit/09d51abd409ee361e93884baae24ffc92cde63a9
  - API reference: https://github.com/weaveworks/ignite/blob/master/api/meta.md
- Create composable interfaces for the internal API machinery: `Client` -> `Cache` -> `Storage` -> `RawStorage` -> `Serializer` https://github.com/weaveworks/ignite/pull/93 https://github.com/weaveworks/ignite/pull/96 https://github.com/weaveworks/ignite/pull/99

- The API Machinery used in Ignite is partly based on the Kubernetes API machinery (`k8s.io/apimachinery`), and hence follows some of the same patterns

20.3 New Commands

- Add the `ignite inspect` command https://github.com/weaveworks/ignite/pull/107
- Add the `ignite gitops` command https://github.com/weaveworks/ignite/pull/100

20.4 Documentation

- Add user-facing documentation and guides https://github.com/weaveworks/ignite/pull/113
  - See: https://github.com/weaveworks/ignite/tree/master/docs
- Generate OpenAPI specifications https://github.com/weaveworks/ignite/commit/f1c5bfd473799f712c4c1d8fb276426780c1bf01
  - See: https://github.com/weaveworks/ignite/blob/master/api/openapi/openapi_generated.go
- Add API type documentation https://github.com/weaveworks/ignite/commit/218c94723f836b8e2cb82886b8664544933ea605
  - See: https://github.com/weaveworks/ignite/blob/master/api
- Added architecture diagram https://github.com/weaveworks/ignite/commit/da53f9fc2f5790edacb5d1b541dd4da8a6089673
  - See: https://github.com/weaveworks/ignite/blob/master/docs/architecture.png
- Added graph of module dependencies https://github.com/weaveworks/ignite/commit/be7cc088c671c5728155fb146367a67d4ada4ea6
  - See: https://github.com/weaveworks/ignite/blob/master/docs/dependencies.svg

20.5 Updated Images

20.5.1 Base Images

- `weaveworks/ignite-ubuntu:v0.4.0`: https://github.com/weaveworks/ignite/blob/master/images/ubuntu/Dockerfile
- `weaveworks/ignite-centos:v0.4.0`: https://github.com/weaveworks/ignite/blob/master/images/centos/Dockerfile
- `weaveworks/ignite-amazonlinux:v0.4.0`: https://github.com/weaveworks/ignite/blob/master/images/amazonlinux/Dockerfile
- `weaveworks/ignite-alpine:v0.4.0`: https://github.com/weaveworks/ignite/blob/master/images/alpine/Dockerfile

20.5.2 Kernel Images

- `weaveworks/ignite-kernel:4.14.123`: https://github.com/weaveworks/ignite/blob/master/images/kernel/Dockerfile
- `weaveworks/ignite-kernel:4.19.47` (default): https://github.com/weaveworks/ignite/blob/master/images/kernel/Dockerfile
- `weaveworks/ignite-amazon-kernel:v0.4.0` (using `4.14.55`): https://github.com/weaveworks/ignite/blob/master/images/amazon-kernel/Dockerfile

20.6 Internal Improvements

- A significant refactor of the whole application has been made to support the new API machinery
- Add structured logging https://github.com/weaveworks/ignite/pull/110
- Factor out `ignite-spawn` into its own binary running in the container https://github.com/weaveworks/ignite/commit/0a1965e7203877c591dc79504ce257a57fd00480
- Upgraded the Firecracker version to v0.17.0 https://github.com/weaveworks/ignite/commit/41e3595b9e8d35c24e8cd97037cc1c7045779ee9
- Set Go version to 1.12.6 https://github.com/weaveworks/ignite/commit/d00cce7d2b09e97f8d515c4a6161b11fc6c61a2c

20.7 Trying it out / Next Steps!

In short:

```
export VERSION=v0.4.0
curl -Lo ignite <https://github.com/weaveworks/ignite/releases/download/${VERSION}/
↪ignite
chmod +x ignite
sudo mv ignite /usr/local/bin
```

A longer installation guide is available here: https://github.com/weaveworks/ignite/blob/master/docs/installation.md

CHAPTER

TWENTYONE

V0.4.1

The first patch release for the `v0.4.x` release stream. If you want to go and look at the new and changed stuff in v0.4.0, see here.

This release, we had an **amazing** amount of 9 PRs from 6 community contributors, in 48 hours after launch :tada:! We hope to see this trend continue, all help is very welcome to this community-driven project!

21.1 New Features / UX Improvements

- It is now possible to access and talk to the Firecracker socket, and metrics/logs FIFOs. (#132, @patrobinson)
- Verify that the VM *actually* did start before `ignite start/run` reports success (#139, @twelho)
- Provide better UX and error messages for the `ignite ssh` command (#149, @twelho)

21.2 Bugfixes

- Set the `-F` flag to `mkfs.ext4`, as it is required on RHEL platforms. (#131, @junaid18183)
- Generate RSA keys instead of ED25519 on FIPS machines. (#136, @junaid18183)
- Make the filtering framework respect exact matches (#138, @twelho)
- Don't fail although there are inactive network interfaces in the container (#146, @luxas)

21.3 Docs improvements

- Add a next action link for better developer-workflow in `installation.md` (#118, @alexellis)
- Error out if the Ignite binary download fails (#120, @alexellis)
- Improve wording in the `README.md` (#125, @seeekr)
- Fix link to the `prometheus.md` doc (#126, @webwurst)
- Fix typo in `docs/README.md` (#128, @andrelop)
- Added demo video to docs/usage.md (#140, @paavan98pm)
- Add docs on how to check for KVM support in the CPU/kernel (#145, @luxas)

21.4 Trying it out / Next Steps!

In short:

```
export VERSION=v0.4.1
curl -fLo ignite https://github.com/weaveworks/ignite/releases/download/${VERSION}/
↪ignite
chmod +x ignite
sudo mv ignite /usr/local/bin
```

A more throughout installation guide is available here: https://github.com/weaveworks/ignite/blob/master/docs/installation.md

CHAPTER

TWENTYTWO

V0.4.2

The second patch release for the `v0.4.x` release stream. If you want to have a look, here are changes for versions v0.4.0 and v0.4.1.

In this release, we had 5 PRs from 3 community contributors, thank you for your amazing work :tada:! We hope to see this trend continue, all help is very welcome to this community-driven project!

22.1 New Features / UX Improvements

- Flannel is now usable with the Ignite kernel thanks to adding the VXLAN kernel module (#154, @curx)
- HAProxy checking for Kubernetes API `/healthz` endpoint (#156, @curx)
- Allow unmarshaling unquoted UIDs from JSON for convenience (#178, @twelho)
- The VM images based on `Ubuntu`, `CentOS` and `Amazon Linux` can now be built on top of a specific release (#193, @twelho)

22.2 Bugfixes

- Import only `/boot` and `/lib` from kernel OCI images, don't overwrite e.g. `/etc/resolv.conf` (#168, @twelho)
- The creation timestamp can now be omitted from specification files, it will be added automatically (#174, @twelho)
- List all VMs instead of just running ones when calling `ignite vm ls/list` (#179, @twelho)
- More robust kernel version checking if e.g. the `strings` binary is not available (#189, @twelho)

22.3 Docs improvements

- Fix formatting in `README.md` (#166, @sftim)
- Fix link to `CentOS` image in `README.md` (#161, @akshaychhajed)
- Added `loop` kernel module dependency to `docs/dependencies.md` (#155, @curx)
- Clarify usage on `Ubuntu` and `CentOS`, embed links to Joe Beda's TGIK recording (#175, @luxas)
- Added a brand new `FAQ`! Check it out at *FAQ.md*! (#197, @luxas)

22.4 Trying it out / Next Steps!

In short:

```
export VERSION=v0.4.2
curl -fLo ignite https://github.com/weaveworks/ignite/releases/download/${VERSION}/
↪ignite
chmod +x ignite
sudo mv ignite /usr/local/bin
```

A more throughout installation guide is available here: https://github.com/weaveworks/ignite/blob/master/docs/installation.md

CHAPTER

TWENTYTHREE

V0.5.0

Released: 13/08/2019

This release consists of **54** noteworthy PRs from 12 contributors. We had **14** contributions from 8 external contributors, thanks!

The main themes of this release has been:

- **Persistent Storage:** Block Device support added as the first external volume type
- **Improved API:** We're continuously improving the API; this release contains `ignite.weave.works/v1alpha2` (still backwards-compatible with `v1alpha1`)
- **Read-write GitOps:** In GitOps mode, Ignite now also pushes the actual state in `.status` back to the repo
- **Refactoring towards a client-server model:** We're now shipping `ignited` that holds the reconciling GitOps and Manifest Directory modes
- **Multi-platform:** We're now shipping ARM 64-bit binaries that you can use on e.g. Packet (and eventually, Raspberry Pi 4!)

Also, our documentation is now available at **https://ignite.readthedocs.org**. Check that site out whenever you need some information, or open an issue :)

23.1 New Features

- Support external volumes (block devices) in Ignite VMs (#275, @twelho)
- Add ARM64 support (#173, @luxas)
- Add new binary: `ignited` (#264, @luxas)
- Add new command: `ignite exec` (#232, @BenTheElder)
- Add Manifest Directory support (like kubelet's Static Pods) (#234, @twelho)
- Support directories as well with the `--copy-files` flag (#271, @twelho)
- Implement read-write GitOps (#241, @twelho)

23.2 API Changes

- Remove `.spec.network.mode`; use a global `--network-plugin` flag instead (#319, @luxas)
- Rename `.spec.image.ociClaim.ref` to `.spec.image.oci` for simplicity (#311, @twelho)
- Redesign OCI image status: Display the image's exact repository digest (#307, @twelho)

- Add `.status.runtime.id` the VM container's ID (#294, @twelho)
- Support configuring `BindAddress` and `Protocol` for a `PortMapping` (#299, @twelho)
- Add `vm.status.startTime` to track the VM's uptime externally (#296, @twelho)
- Replace `vm.status.state` with `vm.status.running` (#292, @twelho)
- Add the initial v1alpha2 API types (#250, @twelho)

23.3 Enhancements

- Refactor: Use the `netlink` library instead of exec'ing out to `ip` (#279, @alexeldeib)
- Improve the CNI implementation, and documentation (#308, @luxas)
- Enable testing in CI, fix the Makefile and tidy (#280, @luxas)
- Automatically generate the release notes (#283, @luxas)
- Structured logging across the application; add logging support to `ignite-spawn` (#247, @twelho)
- Extract watcher, batcher and monitor into pkg/util (#245, @luxas)
- Robust recursive FileWatcher support using `notify` (#265, @twelho)
- Document developer meetings (#272, @dholbach)
- Create/use a runtime interface instead of direct calls to Docker (#211, @twelho)
- Add structured validation for the API types (#216, @luxas)
- Add Strategic Merge Patch support to the storage (#225, @luxas)
- Improve vulnerability scanning of Docker image (#239, @DieterReuter)
- CNI networking cleanup support, Docker client robustness improvements (#111, @twelho)
- Support checksum-based Cache invalidation, improve cache's object handling (#227, @twelho)
- Rename GitStorage into ManifestStorage (#226, @luxas)
- Client and Storage rework: Recognize multiple API groups (#221, @luxas)
- Create internal API types, and use them (#215, @luxas)

23.4 Bug Fixes

- Fix `ignite rm -f` for a running VM without `--debug` (#320, @twelho)
- Ensure the directory for `godoc2md` (#231, @BenTheElder)
- Run `gofmt` first after generating code (#236, @BenTheElder)
- Fix image root permissions (#249, @praseodym)
- Separate graph generation from `make tidy`, add make target docs (#233, @twelho)

23.5 Documentation

- Documentation updates for v0.5.0 (#324, @twelho)
- docs: packet and azure (#330, @alexeldeib)
- add simple CODEOWNERS file (#331, @dholbach)
- add logo to docs (#326, @dholbach)
- Document cloud provider instances with KVM support (#222, @paavan98pm)
- Add Ignite + Footloose documentation (#313, @robertojrojas)
- Add a Read The Docs website: `ignite.readthedocs.org` (#246, @dholbach)
- Documentation updates and clarifications for the New Storage implementation (#242, @twelho)
- Index awesome doc (#276, @dholbach)
- Update docs links (#268, @dholbach)
- Add Google Group to docs for Calendar and permissions (#248, @stealthybox)
- Docs fix: Remove duplicate bracket (#212, @silenceshell)
- Docs fix: Update the command for deleting all VMs (#201, @curx)
- Docs fix: Duplicate bracket (#218, @silenceshell)
- Docs fix: ID is in `.metadata.uid`, not `.metadata.name` (#219, @silenceshell)
- Add an awesome-ignite list for ignite (#270, @luxas)
- Changed –kernel to –kernel-image for accuracy (#217, @paavan98pm)

CHAPTER

TWENTYFOUR

V0.5.1

Released: 16/08/2019

The first patch release in the `v0.5.X` series. Contains some much needed UX improvements, go ahead and try it out!

24.1 Enhancements

- Make `ignite daemon` handle file moves without re-creating and support multiple active moves at once (#341, @twelho)
- Fix `GOHOSTARCH` propagation in the Makefile, tag development image for the host architecture only (#340, @twelho)
- Fix `ignite-spawn`'s VM metadata formatting when performing a cleanup (#336, @twelho)
- Automatically optimize the size of imported images, support importing large images (#335, @twelho)

24.2 Documentation

- Change Read the Docs links to point to the stable branch in main README (#338, @twelho)

CHAPTER

TWENTYFIVE

V0.5.2

Released: 26/08/2019

This is the second patch release in the `v0.5.X` series, containing one bug fix needed for integrating well with Footloose.

25.1 Bug Fixes

- Fix Docker client port mappings by actually exposing them after binding (#350, @twelho)

CHAPTER

TWENTYSIX

V0.6.0

Released: 30/08/2019

Welcome to the `v0.6.0` release, consisting of major underlying improvements, and a more efficient runtime.

This release consists of **25** noteworthy PRs from 4 contributors; although v0.5.0 was released just two weeks ago! We had **5** contributions from 2 external contributors, thanks!

The main themes of this release has been:

- **containerd** is now used as the default container runtime for higher security and speed, and less resource usage
  - This means that Ignite doesn't depend on `docker` anymore!
- **CNI** is now the default networking plugin; by default the `bridge` and `portmap` plugins are used
  - You can still use your third-party CNI implementation of choice, see the networking doc
- **GitOps Toolkit** refactor is complete; now everything you need to create your Git-backed application is available at **https://github.com/weaveworks/gitops-toolkit**
  - Ignite is using this toolkit internally to perform its GitOps capabilities, now you can easily use this functionality, too!
- **Bugfixes and usability improvements** all around the place

Also, our documentation is now available at **https://ignite.readthedocs.org**. Check that site out whenever you need some information, or open an issue :)

26.1 Deprecations

- As per v0.5.0, the `v1alpha2` API version is the default. Going forward, the `v1alpha1` API version is deprecated, and will be removed in a future release.

26.2 New Features

- Make containerd the default runtime and CNI the default network plugin (#371, @twelho)
- Implement the containerd runtime for Ignite (#337, @twelho)
- Add a default CNI `bridge` and `portmap` network for Ignite (#370, @twelho)
- Implement hostPort support with CNI (#375, @luxas)
- Add openSUSE images (#357, @aojea)

26.3 Enhancements

- Implement cleanup of CNI networks using the default bridge (#376, @luxas)
- containerd backend improvements (#368, @twelho)
- Implement runtime selection, only load necessary providers (#366, @twelho)
- Split packages so we can extract `gitops-toolkit` (#347, @luxas)
- Switch to using `weaveworks/gitops-toolkit` (#359, @luxas)
- Switch imports to utilize `gitops-toolkit` (#354, @luxas)
- Simplify the CNI code by vendoring `github.com/containerd/go-cni` (#349, @luxas)
- FileWatcher: Support internal moves without re-creating and multiple active moves at once (#341, @twelho)
- Fix GOHOSTARCH propagation, tag development image for the host architecture only (#340, @twelho)
- Fix ignite-spawn's formatting when performing cleanup on VM metadata (#336, @twelho)
- Automatically optimize the size of an imported image (#335, @twelho)
- Add shell autocompletion for `ignited` (#363, @silenceshell)

26.4 Bug Fixes

- Add `err` as a param for `log.Errorf` (#367, @silenceshell)
- Fix an issue in the GitDirectory loop when trying to commit without any actual changes (#369, @silenceshell)
- GitOps: only change the VM state if it differs from the current one (#374, @twelho)
- Move VM network removal to logically correct place (#373, @twelho)
- Fix Docker client port mappings by actually exposing them after binding (#350, @twelho)

26.5 Documentation

- Update the docs for v0.6.0 (#378, @luxas)
- Docs: Bump latest Ignite version to v0.5.1 (#362, @silenceshell)
- Change Read the Docs links to point to the stable branch in main README (#338, @twelho)

CHAPTER

TWENTYSEVEN

V0.6.1

Released: 02/10/2019

We're excited to release `v0.6.1` with usability improvements and lots of bug fixes :)

This release consists of **32** noteworthy PRs from 6 contributors over the past month.We had **7** contributions from 4 external contributors. Thanks so much!

Ignite should now work with most installations of containerd – even those that are installed underneath upstream docker.Care has been taken with our installation instructions to ensure we are not breaking users docker installations.We've also implemented a graceful fallback to older containerd-shim versions and now support containerd-shim-runc-v2.

This release also contains numerous fixes that make the CNI network plugin work much more reliably.Connections to the internet from vm's using CNI should now work on most machines by default.Please see the following user-facing change.

27.1 Default CNI Network Change

The default `cni0` bridge has changed to a new `ignite0` bridge introduced by the #460 bugfix. This comes with a new subnet as well.We did this because the default CNI config shipped in `v0.6.0` was a non-working configuration for most users.You may continue to use the default CNI configuration. Nothing will change automatically.

If you are using your own CNI configuration, this does not affect you.

To migrate your running CNI networked vm's to the new default subnet, you can:

1. install this new ignite version
2. stop the relevant vm's
3. delete the CNI network
4. restart them

Example:

```
# first, upgrade to ignite v0.6.1

# list all vm's on the default 172.18.0.0/16 CNI network
sudo bin/ignite vm ls | grep '\b172.18.[0-9][0-9]*.[0-9][0-9]*\b'
# stop the listed vm's with the appropriate runtime
sudo bin/ignite my-containerd-vm
sudo bin/ignite my-docker-vm --runtime docker

# remove the old CNI network config
```

(continues on next page)

(continued from previous page)

```
sudo rm -rf rm /etc/cni/net.d/
# optional: remove the old bridge
sudo ifconfig cni0 down
sudo ip link delete cni0

# restart your vm's
sudo bin/ignite my-containerd-vm
sudo bin/ignite my-docker-vm --runtime docker
# Your vm's will now have addresses configured in the 10.61.0.0/16 subnet.
# If they did not have internet connectivity before, they now should.
```

27.2 Enhancements

- wait for SSH when starting a VM (#429, @chanwit)
- skip root requirement for ignite version (#409, @chanwit)
- show runtime name when ignite version (#405, @chanwit)
- improve preflight check and add a containerd test case for ignite run (#416, @chanwit)
- Detect available containerd-shim versions defaulting to legacy linux runtime (#411, @stealthybox)
- log runtime during ignite run (#388, @silenceshell)
- preflight before start operation (#360, @najeal)

27.3 Bug Fixes

- Change default CNI network name, bridge name, and subnet #460, @stealthybox
- Chain firewall plugin to fix routing for default CNI bridge #442, @stealthybox
- Teardown IPMasq rules for all actual configured bridges instead of using the hardcoded default string (#461, @stealthybox)
- Fix containerd resolv.conf + DHCP behavior (#441, @stealthybox)
- Make getIPChains more precise and less failure-prone (#426, @stealthybox)
- quick fix typo umount as unmount in preflight check (#415, @chanwit)
- fix possible dm snapshot leaks (#381, @chanwit)
- make rm command more robust with addition check (#413, @chanwit)

27.4 Documentation

- Make documented install safer for docker-ce users (#454, @stealthybox)
- improve rm docs (#444, @chanwit)
- add installation notes about docker (#397, @kobayashi)
- use image weaveworks/ubuntu instead of centos:7 in command examples (#387, @silenceshell)

- default cni network does not support multi-node (#385, @silenceshell)

27.5 Dependencies

- update containerd to 1.3.0 and golang to 1.12.10 (#464, @chanwit)
- update firecracker to v0.18.0 (#414, @chanwit)
- Bump indirect dependency on klog #453, @stealthybox
- On release, use tidy-in-docker to prevent module differences from differing versions of go (#433, @stealthybox)

27.6 Development

- Add an e2e for ignite run #412, @chanwit
- Store e2e command output /w errors + remove variable sleep #425, @stealthybox
- e2e docker+cni and curl google.com #422, @stealthybox
- Load all images into containerd (#435, @stealthybox)
- specify the minimum version of make (#389, @silenceshell)
- Update Makefile for containerd with overridable commands (#417, @stealthybox)

27.7 Governance

- Update CODEOWNERS (#420, @stealthybox)
- Switch maintainers (#398, @luxas)

CHAPTER

TWENTYEIGHT

V0.6.2

Released: 08/10/2019

This is the second patch release in the `v0.6.X` series, containing bugfixes: It moves the blocking SSH wait for vm's run with `--ssh` to using the actual protocol. It also fixes a locale bug with `resize2fs` parsing that used to occur when using `zh_CN.utf8`.

28.1 Bug Fixes

- Breakout and test `resize2fs` parsing with localized strings + fix for `zh_CN.utf8` (#473, @stealthybox)
- Refactor SSH wait (#474, @stealthybox)
- Use SSH Dial to check if SSH service is really running (#469, @chanwit)

28.2 Documentation

- Ensure CNI bin dir exists before installing (#471, @stealthybox)
- Bump docs install version to v0.6.2 (#475, @stealthybox)

CHAPTER

TWENTYNINE

V0.6.3

Released: 10/12/2019

This is the third patch release in the `v0.6.x` series, containing 1 security bug fix.

29.1 Security Bug Fixes

- Patch CVE-2019-18960 – Use Firecracker v0.18.1 for ignite 0.6.x (#499, @stealthybox)