Ignition Provider

The Ignition provider is used to generate Ignition (https://coreos.com/ignition/docs/latest/) configuration files. *Ignition* is the provisioning utility used by CoreOS (https://coreos.com/) Linux.

The ignition provider is what we call a *logical provider* and doesn't manage any *physical* resources. It generates configurations files to be used by other resources.

Use the navigation to the left to read about the available resources.

Ignition versions

The current Ignition version supported by this provider is the `2.1.0`. For older versions you should use previous releases (https://github.com/terraform-providers/terraform-provider-ignition/releases) of this provider:

- terraform-provider-ignition `<= 0.2.0` - ignition `2.0.0`
- terraform-provider-ignition `1.0.0 =>` - ignition `2.1.0`

Example Usage

This config will write a single service unit (shown below) with the contents of an example service. This unit will be enabled as a dependency of multi-user.target and therefore start on boot

```
# Systemd unit data resource containing the unit definition
data "ignition_systemd_unit" "example" {
  name = "example.service"
  content = "[Service]\nType=oneshot\nExecStart=/usr/bin/echo Hello World\n\n[Install]\nWantedBy=multi-us
er.target"
}

# Ingnition config include the previous defined systemd unit data resource
data "ignition_config" "example" {
  systemd = [
    "${data.ignition_systemd_unit.example.id}",
  ]
}

# Create a CoreOS server using the Igntion config.
resource "aws_instance" "web" {
  # ...

  user_data = "${data.ignition_config.example.rendered}"
}
```

ignition_config

Renders an ignition configuration as JSON. It contains all the disks, partitions, arrays, filesystems, files, users, groups and units.

Example Usage

```
data "ignition_config" "example" {
    systemd = [
        "${data.ignition_systemd_unit.example.id}",
    ]
}
```

Argument Reference

The following arguments are supported:

- `disks` - (Optional) The list of disks to be configured and their options.
- `arrays` - (Optional) The list of RAID arrays to be configured.
- `filesystems` - (Optional) The list of filesystems to be configured and/or used in the `ignition_file`, `ignition_directory`, and `ignition_link` resources.
- `files` - (Optional) The list of files to be written.
- `directories` - (Optional) The list of directories to be created.
- `links` - (Optional) The list of links to be created.
- `systemd` - (Optional) The list of systemd units. Describes the desired state of the systemd units.
- `networkd` - (Optional) The list of networkd units. Describes the desired state of the networkd files.
- `users` - (Optional) The list of accounts to be added.
- `groups` - (Optional) The list of groups to be added.
- `append` - (Optional) Any number of blocks with the configs to be appended to the current config.
- `replace` - (Optional) A block with config that will replace the current.

The `append` and `replace` blocks supports:

- `source` - (Required) The URL of the config. Supported schemes are http, https, tftp, s3, and data. When using http, it is advisable to use the verification option to ensure the contents haven't been modified.
- `verification` - (Optional) The hash of the config, in the form *<type>-<value>* where type is sha512.

Attributes Reference

The following attributes are exported:

- `rendered` - The final rendered template.

ignition_directory

Describes a directory to be created in a particular filesystem.

Example Usage

```
data "ignition_directory" "folder" {
    filesystem = "foo"
    path = "/folder"
}
```

Argument Reference

The following arguments are supported:

- `filesystem` - (Required) The internal identifier of the filesystem. This matches the last filesystem with the given identifier. This should be a valid name from a *ignition_filesystem* resource.
- `path` - (Required) The absolute path to the directory.
- `mode` - (Optional) The directory's permission mode. Note that the mode can be specified as a decimal value (i.e. 0755 -> 493) or an octal value(i.e 0755).
- `uid` - (Optional) The user ID of the owner.
- `gid` - (Optional) The group ID of the owner.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_disk

Describes the desired state of a system's disk.

Example Usage

```
data "ignition_disk" "foo" {
    device = "/dev/sda"
    partition {
        start = 2048
        size = 196037632
    }
}
```

Argument Reference

The following arguments are supported:

- `device` - (Required) The absolute path to the device. Devices are typically referenced by the */dev/disk/by-**symlinks.
- `wipe_table` - (Optional) Whether or not the partition tables shall be wiped. When true, the partition tables are erased before any further manipulation. Otherwise, the existing entries are left intact.
- `partition` - (Optional) The list of partitions and their configuration for this particular disk..

The `partition` block supports:

- `label` - (Optional) The PARTLABEL for the partition.
- `number` - (Optional) The partition number, which dictates it's position in the partition table (one-indexed). If zero, use the next available partition slot.
- `size` - (Optional) The size of the partition (in sectors). If zero, the partition will fill the remainder of the disk.
- `start` - (Optional) The start of the partition (in sectors). If zero, the partition will be positioned at the earliest available part of the disk.
- `type_guid` - (Optional) The GPT partition type GUID (http://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs). If omitted, the default will be *0FC63DAF-8483-4772-8E79-3D69D8477DE4* (Linux filesystem data).

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_file

Describes a file to be written in a particular filesystem.

Example Usage

File with inline content:

```
data "ignition_file" "hello" {
    filesystem = "foo"
    path = "/hello.txt"
    content {
        content = "Hello World!"
    }
}
```

File with remote content:

```
data "ignition_file" "hello" {
    filesystem = "qux"
    path = "/hello.txt"
    source {
        source = "http://example.com/hello.txt.gz"
        compression = "gzip"
        verification = "sha512-0123456789abcdef0123456789...456789abcdef"
    }
}
```

Argument Reference

The following arguments are supported:

- `filesystem` - (Required) The internal identifier of the filesystem. This matches the last filesystem with the given identifier. This should be a valid name from a *ignition_filesystem* resource.
- `path` - (Required) The absolute path to the file.
- `content` - (Optional) Block to provide the file content inline.
- `source` - (Optional) Block to retrieve the file content from a remote location.

  Note: `content` and `source` are mutually exclusive.
- `mode` - (Optional) The file's permission mode. The mode must be properly specified as a decimal value (i.e. 0644 -> 420).
- `uid` - (Optional) The user ID of the owner.
- `gid` - (Optional) The group ID of the owner.

The `content` block supports:

- `mime` - (Required) MIME format of the content (default *text/plain*).
- `content` - (Required) Content of the file.

The `source` block supports:

- `source` - (Required) The URL of the file contents. Supported schemes are http, https, tftp, s3, and data (https://tools.ietf.org/html/rfc2397). When using http, it is advisable to use the verification option to ensure the contents haven't been modified.
- `compression` - (Optional) The type of compression used on the contents (null or gzip). Compression cannot be used with S3.
- `verification` - (Optional) The hash of the config, in the form *<type>-<value>* where type is sha512.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_filesystem

Describes the desired state of a the system's filesystems to be configured and/or used with the *ignition_file* resource.

Example Usage

```
data "ignition_filesystem" "foo" {
    name = "root"
    mount {
        device = "/dev/disk/by-label/ROOT"
        format = "xfs"
        create = true
        options = ["-L", "ROOT"]
    }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Optional) The identifier for the filesystem, internal to Ignition. This is only required if the filesystem needs to be referenced in the a *ignition_files* resource.
- `mount` - (Optional) Contains the set of mount and formatting options for the filesystem. A non-null entry indicates that the filesystem should be mounted before it is used by Ignition.
- `path` - (Optional) The mount-point of the filesystem. A non-null entry indicates that the filesystem has already been mounted by the system at the specified path. This is really only useful for */sysroot*.

The `mount` block supports:

- `device` - (Required) The absolute path to the device. Devices are typically referenced by the */dev/disk/by-**` symlinks.
- `format` - (Required) The filesystem format (ext4, btrfs, xfs, vfat, or swap).
- `wipe_filesystem` - (Optional) Whether or not to wipe the device before filesystem creation.
- `label` - (Optional) The label of the filesystem.
- `uuid` - (Optional) The uuid of the filesystem.
- `options` - (Optional) Any additional options to be passed to the format-specific mkfs utility.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_group

Describes the desired group additions to the passwd database.

Example Usage

```
data "ignition_group" "foo" {
    name = "foo"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The groupname for the account.
- `password_hash` - (Optional) The encrypted password for the account.
- `gid` - (Optional) The group ID of the new account.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_link

Describes a link to be created in a particular filesystem.

Example Usage

```
data "ignition_link" "symlink" {
    filesystem = "foo"
    path = "/symlink"
    target = "/foo"
}
```

Argument Reference

The following arguments are supported:

- `filesystem` - (Required) The internal identifier of the filesystem. This matches the last filesystem with the given identifier. This should be a valid name from a *ignition_filesystem* resource.
- `path` - (Required) The absolute path to the link.
- `target` - (Required) The target path of the link.
- `hard` - (Optional) A symbolic link is created if this is false, a hard one if this is true.
- `uid` - (Optional) The user ID of the owner.
- `gid` - (Optional) The group ID of the owner.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_networkd_unit

Describes the desired state of the networkd units.

Example Usage

```
data "ignition_networkd_unit" "example" {
    name = "00-eth0.network"
    content = "[Match]\nName=eth0\n\n[Network]\nAddress=10.0.1.7"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the file. This must be suffixed with a valid unit type (e.g. *00-eth0.network*).
- `content` - (Required) The contents of the networkd file.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_raid

Describes the desired state of the system's RAID.

Example Usage

```
data "ignition_raid" "md" {
    name = "data"
    level = "stripe"
    devices = [
        "/dev/disk/by-partlabel/raid.1.1",
        "/dev/disk/by-partlabel/raid.1.2"
    ]
}

data "ignition_disk" "disk1" {
    device = "/dev/sdb"
    wipe_table = true
    partition {
        label = "raid.1.1"
        number = 1
        size = 20480
        start = 0
    }
}

data "ignition_disk" "disk2" {
    device = "/dev/sdc"
    wipe_table = true
    partition {
        label = "raid.1.2"
        number = 1
        size = 20480
        start = 0
    }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name to use for the resulting md device.
- `level` - (Required) The redundancy level of the array (e.g. linear, raid1, raid5, etc.).
- `devices` - (Required) The list of devices (referenced by their absolute path) in the array.
- `spares` - (Optional) The number of spares (if applicable) in the array.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*

ignition_systemd_unit

Describes the desired state of the systemd units.

Example Usage

```
data "ignition_systemd_unit" "example" {
    name = "example.service"
    content = "[Service]\nType=oneshot\nExecStart=/usr/bin/echo Hello World\n\n[Install]\nWantedBy=multi-user.target"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the unit. This must be suffixed with a valid unit type (e.g. *thing.service*).
- `enabled` - (Optional) Whether or not the service shall be enabled. When true, the service is enabled. In order for this to have any effect, the unit must have an install section. (default true)
- `mask` - (Optional) Whether or not the service shall be masked. When true, the service is masked by symlinking it to */dev/null*.
- `content` - (Optional) The contents of the unit.
- `dropin` - (Optional) The list of drop-ins for the unit.

The `dropin` block supports:

- `name` - (Required) The name of the drop-in. This must be suffixed with *.conf*.
- `content` - (Optional) The contents of the drop-in.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.

ignition_user

Describes the desired user additions to the passwd database.

Example Usage

```
data "ignition_user" "foo" {
    name = "foo"
    home_dir = "/home/foo/"
    shell = "/bin/bash"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The username for the account.
- `password_hash` - (Optional) The encrypted password for the account.
- `ssh_authorized_keys` - (Optional) A list of SSH keys to be added to the user's authorized_keys.
- `uid` - (Optional) The user ID of the new account.
- `gecos` - (Optional) The GECOS field of the new account.
- `home_dir` - (Optional) The home directory of the new account.
- `no_create_home` - (Optional) Whether or not to create the user's home directory.
- `primary_group` - (Optional) The name or ID of the primary group of the new account.
- `groups` - (Optional) The list of supplementary groups of the new account.
- `no_user_group` - (Optional) Whether or not to create a group with the same name as the user.
- `no_log_init` - (Optional) Whether or not to add the user to the lastlog and faillog databases.
- `shell` - (Optional) The login shell of the new account.
- `system` - (Optional) Whether or not to make the account a system account. This only has an effect if the account doesn't exist yet.

Attributes Reference

The following attributes are exported:

- `id` - ID used to reference this resource in *ignition_config*.