Cobbler Provider

The Cobbler provider is used to interact with a locally installed Cobbler (http://cobbler.github.io) service. The provider needs to be configured with the proper credentials before it can be used.

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

Example Usage

```
# Configure the Cobbler provider
provider "cobbler" {
  username = "${var.cobbler_username}"
  password = "${var.cobbler_password}"
  url      = "${var.cobbler_url}"
}

# Create a Cobbler Distro
resource "cobbler_distro" "ubuntu-1404-x86_64" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `username` - (Required) The username to the Cobbler service. This can also be specified with the `COBBLER_USERNAME` shell environment variable.
- `password` - (Required) The password to the Cobbler service. This can also be specified with the `COBBLER_PASSWORD` shell environment variable.
- `url` - (Required) The url to the Cobbler service. This can also be specified with the `COBBLER_URL` shell environment variable.
- `insecure` - (Optional) Ignore SSL certificate warnings and errors. This can also be specified with the `COBBLER_INSECURE` shell environment variable.
- `cacert_file` - (Optional) The path or contents of an SSL CA certificate. This can also be specified with the `COBBLER_CACERT_FILE` shell environment variable.

cobbler_distro

Manages a distribution within Cobbler.

Example Usage

```
resource "cobbler_distro" "ubuntu-1404-x86_64" {
  name       = "foo"
  breed      = "ubuntu"
  os_version = "trusty"
  arch       = "x86_64"
  kernel     = "/var/www/cobbler/ks_mirror/Ubuntu-14.04/install/netboot/ubuntu-installer/amd64/linux"
  initrd     = "/var/www/cobbler/ks_mirror/Ubuntu-14.04/install/netboot/ubuntu-installer/amd64/initrd.gz"
}
```

Argument Reference

The following arguments are supported:

- `arch` - (Required) The architecture of the distro. Valid options are: i386, x86_64, ia64, ppc, ppc64, s390, arm.
- `breed` - (Required) The "breed" of distribution. Valid options are: redhat, fedora, centos, scientific linux, suse, debian, and ubuntu. These choices may vary depending on the version of Cobbler in use.
- `boot_files` - (Optional) Files copied into tftpboot beyond the kernel/initrd.
- `comment` - (Optional) Free form text description.
- `fetchable_files` - (Optional) Templates for tftp or wget.
- `kernel` - (Required) Absolute path to kernel on filesystem. This must already exist prior to creating the distro.
- `kernel_options` - (Optional) Kernel options to use with the kernel.
- `kernel_options_post` - (Optional) Post install Kernel options to use with the kernel after installation.
- `initrd` - (Required) Absolute path to initrd on filesystem. This must already exist prior to creating the distro.
- `mgmt_classes` - (Optional) Management classes for external config management.
- `name` - (Required) A name for the distro.
- `os_version` - (Required) The version of the distro you are creating. This varies with the version of Cobbler you are using. An updated signature list may need to be obtained in order to support a newer version. Example: `trusty`.
- `owners` - (Optional) Owners list for authz_ownership.
- `redhat_management_key` - (Optional) Red Hat Management key.
- `redhat_management_server` - (Optional) Red Hat Management server.
- `template_files` - (Optional) File mappings for built-in config management.

Attributes Reference

All of the above Optional attributes are also exported.

Notes

The path to the `kernel` and `initrd` files must exist before creating a Distro. Usually this involves running `cobbler import ...` prior to creating the Distro.

cobbler_kickstart_file

Manages a Kickstart File within Cobbler.

Example Usage

```
resource "cobbler_kickstart_file" "my_kickstart" {
  name = "/var/lib/cobbler/kickstarts/my_kickstart.ks"
  body = "<content of kickstart file>"
}
```

Argument Reference

The following arguments are supported:

- `body` - (Required) The body of the kickstart file.
- `name` - (Required) The name of the kickstart file. This must be the full path, including `/var/lib/cobbler/kickstarts`.

cobbler_profile

Manages a Profile within Cobbler.

Example Usage

```
resource "cobbler_profile" "my_profile" {
  name   = "/var/lib/cobbler/snippets/my_snippet"
  distro = "ubuntu-1404-x86_64"
}
```

Argument Reference

The following arguments are supported:

- `boot_files` - (Optional) Files copied into tftpboot beyond the kernel/initrd.
- `comment` - (Optional) Free form text description.
- `parent` - (Optional) The parent this profile inherits settings from.
- `server` - (Optional) The server-override for the profile.
- `distro` - (Optional) Parent distribution.
- `enable_gpxe` - (Optional) Use gPXE instead of PXELINUX for advanced booting options.
- `enable_menu` - (Optional) Enable a boot menu.
- `fetchable_files` - (Optional) Templates for tftp or wget.
- `kernel_options` - (Optional) Kernel options for the profile.
- `kernel_options_post` - (Optional) Post install kernel options.
- `kickstart` - (Optional) The kickstart file to use.
- `ks_meta` - (Optional) Kickstart metadata.
- `mgmt_classes` - (Optional) For external configuration management.
- `mgmt_parameters` - (Optional) Parameters which will be handed to your management application (Must be a valid YAML dictionary).
- `name_servers_search` - (Optional) Name server search settings.
- `name_servers` - (Optional) Name servers.
- `name` - (Required) The name of the profile.
- `owners` - (Optional) Owners list for authz_ownership.
- `proxy` - (Optional) Proxy URL.

- `redhat_management_key` - (Optional) Red Hat Management Key.
- `redhat_management_server` - (Optional) RedHat Management Server.
- `repos` - (Optional) Repos to auto-assign to this profile.
- `template_files` - (Optional) File mappings for built-in config management.
- `template_remote_kickstarts` - (Optional) remote kickstart templates.
- `virt_auto_boot` - (Optional) Auto boot virtual machines.
- `virt_bridge` - (Optional) The bridge for virtual machines.
- `virt_cpus` - (Optional) The number of virtual CPUs.
- `virt_file_size` - (Optional) The virtual machine file size.
- `virt_path` - (Optional) The virtual machine path.
- `virt_ram` - (Optional) The amount of RAM for the virtual machine.
- `virt_type` - (Optional) The type of virtual machine. Valid options are: xenpv, xenfv, qemu, kvm, vmware, openvz.
- `virt_disk_driver` - (Optional) The virtual machine disk driver.

Attributes Reference

All of the above Optional attributes are also exported.

cobbler_repo

Manages a repo within Cobbler.

Example Usage

```
resource "cobbler_repo" "my_repo" {
  name           = "my_repo"
  breed          = "apt"
  arch           = "x86_64"
  apt_components = ["main"]
  apt_dists      = ["trusty"]
  mirror         = "http://us.archive.ubuntu.com/ubuntu/"
}
```

Argument Reference

The following arguments are supported:

- `apt_components` - (Optional) List of Apt components such as main, restricted, universe. Applicable to apt breeds only.
- `apt_dists` - (Optional) List of Apt distribution names such as trusty, trusty-updates. Applicable to apt breeds only.
- `arch` - (Optional) The architecture of the repo. Valid options are: i386, x86_64, ia64, ppc, ppc64, s390, arm.
- `breed` - (Required) The "breed" of distribution. Valid options are: rsync, rhn, yum, apt, and wget. These choices may vary depending on the version of Cobbler in use.
- `comment` - (Optional) Free form text description.
- `createrepo_flags` - (Optional) Flags to use with `createrepo`.
- `environment` - (Optional) Environment variables to use during repo command execution.
- `keep_updated` - (Optional) Update the repo upon Cobbler sync. Valid values are true or false.
- `mirror` - (Required) Address of the repo to mirror.
- `mirror_locally` - (Required) Whether to copy the files locally or just references to the external files. Valid values are true or false.
- `name` - (Required) A name for the repo.
- `owners` - (Optional) List of Owners for authz_ownership.
- `proxy` - (Optional) Proxy to use for downloading the repo. This argument does not work on older versions of Cobbler.
- `rpm_list` - (Optional) List of specific RPMs to mirror.

Attributes Reference

All of the above Optional attributes are also exported.

cobbler_snippet

Manages a Snippet within Cobbler.

Example Usage

```
resource "cobbler_snippet" "my_snippet" {
  name = "/var/lib/cobbler/snippets/my_snippet"
  body = "<content of snippet>"
}
```

Argument Reference

The following arguments are supported:

- `body` - (Required) The body of the snippet.
- `name` - (Required) The name of the snippet. This must be the full path, including `/var/lib/cobbler/snippets`.

cobbler_system

Manages a System within Cobbler.

Example Usage

```
resource "cobbler_system" "my_system" {
  name         = "my_system"
  profile      = "${cobbler_profile.my_profile.name}"
  name_servers = ["8.8.8.8", "8.8.4.4"]
  comment      = "I'm a system"

  interface {
    name        = "eth0"
    mac_address = "aa:bb:cc:dd:ee:ff"
    static      = true
    ip_address  = "1.2.3.4"
    netmask     = "255.255.255.0"
  }

  interface {
    name        = "eth1"
    mac_address = "aa:bb:cc:dd:ee:fa"
    static      = true
    ip_address  = "1.2.3.5"
    netmask     = "255.255.255.0"
  }
}
```

Argument Reference

The following arguments are supported:

- `boot_files` - (Optional) TFTP boot files copied into tftpboot.
- `comment` - (Optional) Free form text description
- `enable_gpxe` - (Optional) Use gPXE instead of PXELINUX.
- `fetchable_files` - (Optional) Templates for tftp or wget.
- `gateway` - (Optional) Network gateway.
- `hostname` - (Optional) Hostname of the system.
- `image` - (Optional) Parent image (if no profile is used).
- `interface` - (Optional)
- `ipv6_default_device` - (Optional) IPv6 default device.
- `kernel_options` - (Optional) Kernel options. ex: selinux=permissive.
- `kernel_options_post` - (Optional) Kernel options (post install).

- `kickstart` - (Optional) Path to kickstart template.
- `ks_meta` - (Optional) Kickstart metadata.
- `ldap_enabled` - (Optional) Configure LDAP at next config update.
- `ldap_type` - (Optional) LDAP management type.
- `mgmt_classes` - (Optional) Management classes for external config management.
- `mgmt_parameters` - (Optional) Parameters which will be handed to your management application. Must be a valid YAML dictionary.
- `monit_enabled` - (Optional) Configure monit on this machine at next config update.
- `name_servers_search` - (Optional) Name servers search path.
- `name_servers` - (Optional) Name servers.
- `name` - (Required) The name of the system.
- `netboot_enabled` - (Optional) (re)Install this machine at next boot.
- `owners` - (Optional) Owners list for authz_ownership.
- `power_address` - (Optional) Power management address.
- `power_id` - (Optional) Usually a plug number or blade name if power type requires it.
- `power_pass` - (Optional) Power management password.
- `power_type` - (Optional) Power management type.
- `power_user` - (Optional) Power management user.
- `profile` - (Required) Parent profile.
- `proxy` - (Optional) Proxy URL.
- `redhat_management_key` - (Optional) Red Hat management key.
- `redhat_management_server` - (Optional) Red Hat management server.
- `status` - (Optional) System status (development, testing, acceptance, production).
- `template_files` - (Optional) File mappings for built-in configuration management.
- `template_remote_kickstarts` - (Optional) template remote kickstarts.
- `virt_auto_boot` - (Optional) Auto boot the VM.
- `virt_cpus` - (Optional) Number of virtual CPUs in the VM.
- `virt_disk_driver` - (Optional) The on-disk format for the virtualization disk.
- `virt_file_size` - (Optional) Virt file size.
- `virt_path` - (Optional) Path to the VM.
- `virt_pxe_boot` - (Optional) Use PXE to build this VM?

- `virt_ram` - (Optional) The amount of RAM for the VM.
- `virt_type` - (Optional) Virtualization technology to use: xenpv, xenfv, qemu, kvm, vmware, openvz.

The `interface` block supports:

- `name` - (Required) The device name of the interface. ex: eth0.
- `cnames` - (Optional) Canonical name records.
- `dhcp_tag` - (Optional) DHCP tag.
- `dns_name` - (Optional) DNS name.
- `bonding_opts` - (Optional) Options for bonded interfaces.
- `bridge_opts` - (Optional) Options for bridge interfaces.
- `gateway` - (Optional) Per-interface gateway.
- `interface_type` - (Optional) The type of interface: na, master, slave, bond, bond_slave, bridge, bridge_slave, bonded_bridge_slave.
- `interface_master` - (Optional) The master interface when slave.
- `ip_address` - (Optional) The IP address of the interface.
- `ipv6_address` - (Optional) The IPv6 address of the interface.
- `ipv6_mtu` - (Optional) The MTU of the IPv6 address.
- `ipv6_static_routes` - (Optional) Static routes for the IPv6 interface.
- `ipv6_default_gateway` - (Optional) The default gateawy for the IPv6 address / interface.
- `mac_address` - (Optional) The MAC address of the interface.
- `management` - (Optional) Whether this interface is a management interface.
- `netmask` - (Optional) The IPv4 netmask of the interface.
- `static` - (Optional) Whether the interface should be static or DHCP.
- `static_routes` - (Optional) Static routes for the interface.
- `virt_bridge` - (Optional) The virtual bridge to attach to.

Attribute Reference

All optional attributes listed above are also exported.