

Packet Provider

The Packet provider is used to interact with the resources supported by Packet. 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 Packet Provider
provider "packet" {
  auth_token = "${var.auth_token}"
}

# Create a project
resource "packet_project" "cool_project" {
  name           = "My First Terraform Project"
  payment_method = "PAYMENT_METHOD_ID"          # Only required for a non-default payment method
}

# Create a device and add it to tf_project_1
resource "packet_device" "web1" {
  hostname       = "tf.coreos2"
  plan           = "baremetal_1"
  facility       = "ewr1"
  operating_system = "coreos_stable"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.cool_project.id}"
}
```

Argument Reference

The following arguments are supported:

- `auth_token` - (Required) This is your Packet API Auth token. This can also be specified with the `PACKET_AUTH_TOKEN` shell environment variable.

packet_operating_system

Use this data source to get Packet Operating System image.

Example Usage

```
data "packet_operating_system" "example" {
  name           = "Container Linux"
  distro         = "coreos"
  version        = "alpha"
  provisionable_on = "baremetal_1"
}

resource "packet_device" "server" {
  hostname     = "tf.coreos2"
  plan         = "baremetal_1"
  facility     = "ewr1"
  operating_system = "${data.packet_operating_system.example.id}"
  billing_cycle = "hourly"
  project_id   = "${packet_project.cool_project.id}"
}
```

Argument Reference

- `distro` - (Optional) Name of the OS distribution.
- `name` - (Optional) Name or part of the name of the distribution. Case insensitive.
- `provisionable_on` - (Optional) Plan name.
- `version` - (Optional) Version of the distribution

Attributes Reference

- `id` - Operating system slug
- `slug` - Operating system slug (same as `id`)

packet_precreated_ip_block

Use this data source to get CIDR expression for precreated IPv6 and IPv4 blocks in Packet. You can then use the cidrsubnet TF builtin function to derive subnets.

Example Usage

```
# Create project, device in it, and then assign /64 subnet from precreated block
# to the new device

resource "packet_project" "test" {
  name = "testpro"
}

resource "packet_device" "web1" {
  hostname      = "tftest"
  plan          = "baremetal_0"
  facility      = "ewr1"
  operating_system = "ubuntu_16_04"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.test.id}"
}

# we have to make the datasource depend on the device. Here I do it implicitly
# with the project_id param, because an explicit "depends_on" attribute in
# a datasource taints the state:
# https://github.com/hashicorp/terraform/issues/11806
data "packet_precreated_ip_block" "test" {
  facility      = "ewr1"
  project_id    = "${packet_device.test.project_id}"
  address_family = 6
  public        = true
}

# The precreated IPv6 blocks are /56, so to get /64, we specify 8 more bits for network.
# The cidrsubnet interpolation will pick second /64 subnet from the precreated block.

resource "packet_ip_attachment" "from_ipv6_block" {
  device_id = "${packet_device.web1.id}"
  cidr_notation = "${cidrsubnet(data.packet_precreated_ip_block.test.cidr_notation,8,2)}"
}
```

Argument Reference

- `project_id` - (Required) ID of the project where the searched block should be.
- `address_family` - (Required) 4 or 6, depending on which block you are looking for.
- `public` - (Required) Whether to look for public or private block.
- `facility` - (Required) Facility of the searched block.

Attributes Reference

- `cidr_notation` - CIDR notation of the looked up block.

packet_operating_system

Use this data source to get Packet Spot Market Price.

Example Usage

```
data "packet_spot_market_price" "example" {  
  facility = "ewr1"  
  plan     = "baremetal_1"  
}
```

Argument Reference

- `facility` - (Required) Name of the facility.
- `plan` - (Required) Name of the plan.

Attributes Reference

- `price` - Current spot market price for given plan in given facility.

packet_device

Provides a Packet device resource. This can be used to create, modify, and delete devices.

Note: All arguments including the `root_password` and `user_data` will be stored in the raw state as plain-text. Read more about sensitive data in state (</docs/state/sensitive-data.html>).

Example Usage

```
# Create a device and add it to cool_project
resource "packet_device" "web1" {
  hostname      = "tf.coreos2"
  plan          = "baremetal_1"
  facility      = "ewr1"
  operating_system = "coreos_stable"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.cool_project.id}"
}
```

```
# Same as above, but boot via iPXE initially, using the Ignition Provider for provisioning
resource "packet_device" "pxe1" {
  hostname      = "tf.coreos2-pxe"
  plan          = "baremetal_1"
  facility      = "ewr1"
  operating_system = "custom_ipxe"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.cool_project.id}"
  ipxe_script_url = "https://rawgit.com/cloudnativelabs/pxe/master/packet/coreos-stable-packet.ipxe"
  always_pxe     = "false"
  user_data      = "${data.ignition_config.example.rendered}"
}
```

```
# Deploy device on next-available reserved hardware and do custom partitioning.
resource "packet_device" "web1" {
  hostname      = "tftest"
  plan          = "baremetal_0"
  facility      = "sjc1"
  operating_system = "ubuntu_16_04"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.cool_project.id}"
  hardware_reservation_id = "next-available"
  storage = <<EOS
{
  "disks": [
    {
      "device": "/dev/sda",
      "wipeTable": true,
      "partitions": [
        {
          "label": "BIOS",
          "number": 1,
          "size": 4096
        },
        {
          "label": "SWAP"

```

```

        "label": "SWAP",
        "number": 2,
        "size": "3993600"
    },
    {
        "label": "ROOT",
        "number": 3,
        "size": 0
    }
]
}
],
"filesystems": [
    {
        "mount": {
            "device": "/dev/sda3",
            "format": "ext4",
            "point": "/",
            "create": {
                "options": [
                    "-L",
                    "ROOT"
                ]
            }
        }
    },
    {
        "mount": {
            "device": "/dev/sda2",
            "format": "swap",
            "point": "none",
            "create": {
                "options": [
                    "-L",
                    "SWAP"
                ]
            }
        }
    }
]
}
EOS
}

```

Argument Reference

The following arguments are supported:

- `hostname` - (Required) The device name
- `project_id` - (Required) The id of the project in which to create the device
- `operating_system` - (Required) The operating system slug. To find the slug, or visit Operating Systems API docs (<https://www.packet.net/developers/api/#operatingsystems>), set your API auth token in the top of the page and see JSON from the API response.
- `facility` - (Required) The facility in which to create the device. To find the facility code, visit Facilities API docs (<https://www.packet.net/developers/api/#facilities>), set your API auth token in the top of the page and see JSON from the API response.

- `plan` - (Required) The device plan slug. To find the plan slug, visit Device plans API docs (<https://www.packet.net/developers/api/#plans>), set your auth token in the top of the page and see JSON from the API response.
- `billing_cycle` - (Required) monthly or hourly
- `user_data` (Optional) - A string of the desired User Data for the device.
- `public_ipv4_subnet_size` (Optional) - Size of allocated subnet, more information is in the Custom Subnet Size (<https://help.packet.net/article/55-custom-subnet-size>) doc.
- `ipxe_script_url` (Optional) - URL pointing to a hosted iPXE script. More information is in the Custom iPXE (<https://help.packet.net/article/26-custom-ipxe>) doc.
- `always_pxe` (Optional) - If true, a device with OS `custom_ipxe` will continue to boot via iPXE on reboots.
- `hardware_reservation_id` (Optional) - The id of hardware reservation where you want this device deployed, or next-available if you want to pick your next available reservation automatically.
- `storage` (Optional) - JSON for custom partitioning. Only usable on reserved hardware. More information in in the Custom Partitioning and RAID (<https://help.packet.net/article/61-custom-partitioning-raid>) doc.
- `tags` - Tags attached to the device
- `description` - Description string for the device

Attributes Reference

The following attributes are exported:

- `id` - The ID of the device
- `hostname` - The hostname of the device
- `project_id` - The ID of the project the device belongs to
- `facility` - The facility the device is in
- `plan` - The hardware config of the device
- `network` - The device's private and public IP (v4 and v6) network details
- `access_public_ipv6` - The ipv6 maintenance IP assigned to the device
- `access_public_ipv4` - The ipv4 maintenance IP assigned to the device
- `access_private_ipv4` - The ipv4 private IP assigned to the device
- `locked` - Whether the device is locked
- `billing_cycle` - The billing cycle of the device (monthly or hourly)
- `operating_system` - The operating system running on the device
- `state` - The status of the device
- `created` - The timestamp for when the device was created

- `updated` - The timestamp for the last time the device was updated
- `tags` - Tags attached to the device
- `description` - Description string for the device
- `hardware_reservation_id` - The id of hardware reservation which this device occupies
- `root_password` - Root password to the server (disabled after 24 hours)

packet_ip_attachment

Provides a resource to attach elastic IP subnets to devices.

To attach an IP subnet from a reserved block to a provisioned device, you must derive a subnet CIDR belonging to one of your reserved blocks in the same project and facility as the target device.

For example, you have reserved IPv4 address block 147.229.10.152/30, you can choose to assign either the whole block as one subnet to a device; or 2 subnets with CIDRs 147.229.10.152/31 and 147.229.10.154/31; or 4 subnets with mask prefix length 32. More about the elastic IP subnets is here (<https://help.packet.net/article/54-elastic-ips>).

Device and reserved block must be in the same facility.

Example Usage

```
# Reserve /30 block of max 2 public IPv4 addresses in Parsippany, NJ (ewr1) for myproject
resource "packet_reserved_ip_block" "myblock" {
  project_id = "${packet_project.myproject.id}"
  facility   = "ewr1"
  quantity   = 2
}

# Assign /32 subnet (single address) from reserved block to a device
resource "packet_ip_attachment" "first_address_assignment" {
  device_id = "${packet_device.mydevice.id}"
  # following interpolation will result to sth like "147.229.10.152/32"
  cidr_notation = "${cidrhost(packet_reserved_ip_block.myblock.cidr_notation,0)}/32"
}
```

Argument Reference

The following arguments are supported:

- `device_id` - (Required) ID of device to which to assign the subnet
- `cidr_notation` - (Required) CIDR notation of subnet from block reserved in the same project and facility as the device

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the assignment
- `device_id` - ID of device to which subnet is assigned
- `cidr_notation` - Assigned subnet in CIDR notation, e.g. "147.229.15.30/31"
- `gateway` - IP address of gateway for the subnet

- `network` - Subnet network address
- `netmask` - Subnet mask in decimal notation, e.g. "255.255.255.0"
- `cidr` - length of CIDR prefix of the subnet as integer
- `address_family` - Address family as integer (4 or 6)
- `public` - boolean flag whether subnet is reachable from the Internet

packet_organization

Provides a resource to manage organization resource in Packet.

Example Usage

```
# Create a new Project
resource "packet_organization" "tf_organization_1" {
  name = "foobar"
  description = "quux"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the Organization.
- `description` - Description string.
- `website` - Website link.
- `twitter` - Twitter handle.
- `logo` - Logo URL.

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the organization.
- `name` - The name of the Organization.
- `description` - Description string.
- `website` - Website link.
- `twitter` - Twitter handle.
- `logo` - Logo URL.

packet_project

Provides a Packet Project resource to allow you manage devices in your projects.

Example Usage

```
# Create a new Project
resource "packet_project" "tf_project_1" {
  name      = "Terraform Fun"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the Project on Packet.net
- `payment_method_id` - The UUID of payment method for this project. If you keep it empty, Packet API will pick your default Payment Method.
- `organization_id` - The UUID of Organization under which you want to create the project. If you leave it out, the project will be create under your the default Organization of your account.

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the project
- `payment_method_id` - The UUID of payment method for this project.
- `organization_id` - The UUID of this project's parent organization.
- `created` - The timestamp for when the Project was created
- `updated` - The timestamp for the last time the Project was updated

packet_reserved_ip_block

Provides a resource to create and manage blocks of reserved IP addresses in a project.

When user provision first device in a facility, Packet automatically allocates IPv6/56 and private IPv4/25 blocks. The new device then gets IPv6 and private IPv4 addresses from those block. It also gets a public IPv4/31 address. Every new device in the project and facility will automatically get IPv6 and private IPv4 addresses from pre-allocated i blocks. The IPv6 and private IPv4 blocks can't be created, only imported.

It is only possible to create public IPv4 blocks, with masks from /24 (256 addresses) to /32 (1 address).

Once IP block is allocated or imported, an address from it can be assigned to device with the `packet_ip_attachment` resource.

Example Usage

```
# Allocate /30 block of max 2 public IPv4 addresses in Parsippany, NJ (ewr1) for myproject

resource "packet_reserved_ip_block" "two_elastic_addresses" {
  project_id = "${packet_project.myproject.id}"
  facility   = "ewr1"
  quantity   = 2
}
```

Argument Reference

The following arguments are supported:

- `facility` - (Required) The facility where to allocate the address block
- `project_id` - (Required) The packet project ID where to allocate the address block
- `quantity` - (Required) The number of allocated /32 addresses, a power of 2

Attributes Reference

The following attributes are exported:

- `facility` - The facility where the addresses are
- `project_id` - To which project the addresses belong
- `quantity` - Number of /32 addresses in the block
- `id` - The unique ID of the block
- `cidr_notation` - Address and mask in CIDR notation, e.g. "147.229.15.30/31"
- `network` - Network IP address portion of the block specification

- netmask - Mask in decimal notation, e.g. "255.255.255.0"
- cidr - length of CIDR prefix of the block as integer
- address_family - Address family as integer (4 or 6)
- public - boolean flag whether addresses from a block are public

Idempotent reference to a first /32 address from a reserved block might look like

"\${cidrhost(packet_reserved_ip_block.test.cidr_notation,0)}/32".

packet_spot_market_request

Provides a Packet Spot Market Request resource to allow you to manage spot market requests on your account.

<https://help.packet.net/en-us/article/20-spot-market> (<https://help.packet.net/en-us/article/20-spot-market>)

Example Usage

```
# Create a spot market request
resource "packet_spot_market_request" "req" {
  project_id      = "${packet_project.cool_project.id}"
  "max_bid_price" = 0.03
  "facilities"    = ["ewr1"]
  "devices_min"  = 1
  "devices_max"  = 1

  "instance_parameters" {
    "hostname"      = "testspot"
    "billing_cycle" = "hourly"
    "operating_system" = "coreos_stable"
    "plan"          = "baremetal_0"
  }
}
```

Argument Reference

The following arguments are supported:

- `devices_max` - (Required) Maximum number devices to be created
- `devices_min` - (Required) Minimum number devices to be created
- `max_bid_price` - (Required) Maximum price user is willing to pay per hour per device
- `facilities` - (Required) Facility IDs where devices should be created
- `instance_parameters` - (Required) Device parameters. See device resource for details
- `project_id` - (Required) Project ID
- `wait_for_devices` - (Optional) On resource creation - wait until all desired devices are active, on resource destruction - wait until devices are removed

Timeouts

The `timeouts` block allows you to specify timeouts (<https://www.terraform.io/docs/configuration/resources.html#timeouts>) for certain actions:

- `create` - (Defaults to 60 mins) Used when creating the Spot Market Request and `wait_for_devices == true`
- `delete` - (Defaults to 60 mins) Used when destroying the Spot Market Request and `wait_for_devices == true`

Attributes Reference

The following attributes are exported:

- id - The ID of the Spot Market Request

packet_ssh_key

Provides a Packet SSH key resource to allow you manage SSH keys on your account. All SSH keys on your account are loaded on all new devices, they do not have to be explicitly declared on device creation.

Example Usage

```
# Create a new SSH key
resource "packet_ssh_key" "key1" {
  name      = "terraform-1"
  public_key = "${file("/home/terraform/.ssh/id_rsa.pub")}"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the SSH key for identification
- `public_key` - (Required) The public key. If this is a file, it can be read using the file interpolation function

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the key
- `name` - The name of the SSH key
- `public_key` - The text of the public key
- `fingerprint` - The fingerprint of the SSH key
- `created` - The timestamp for when the SSH key was created
- `updated` - The timestamp for the last time the SSH key was updated

packet_volume

Provides a Packet Block Storage Volume resource to allow you to manage block volumes on your account. Once created by Terraform, they must then be attached and mounted using the `api` and `packet_block_attach` and `packet_block_detach` scripts.

Example Usage

```
# Create a new block volume
resource "packet_volume" "volume1" {
  description      = "terraform-volume-1"
  facility         = "ewr1"
  project_id       = "${packet_project.cool_project.id}"
  plan             = "storage_1"
  size            = 100
  billing_cycle    = "hourly"

  snapshot_policies = {
    snapshot_frequency = "1day"

    snapshot_count = 7
  }

  snapshot_policies = {
    snapshot_frequency = "1month"

    snapshot_count = 6
  }
}
```

Argument Reference

The following arguments are supported:

- `plan` - (Required) The service plan slug of the volume
- `facility` - (Required) The facility to create the volume in
- `project_id` - (Required) The packet project ID to deploy the volume in
- `size` - (Required) The size in GB to make the volume
- `billing_cycle` - The billing cycle, defaults to "hourly"
- `description` - Optional description for the volume
- `snapshot_policies` - Optional list of snapshot policies
- `locked` - Lock or unlock the volume

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the volume
- `name` - The name of the volume
- `description` - The description of the volume
- `size` - The size in GB of the volume
- `plan` - Performance plan the volume is on
- `billing_cycle` - The billing cycle, defaults to hourly
- `facility` - The facility slug the volume resides in
- `state` - The state of the volume
- `locked` - Whether the volume is locked or not
- `project_id` - The project id the volume is in
- `attachments` - A list of attachments, each with it's own `href` attribute
- `created` - The timestamp for when the volume was created
- `updated` - The timestamp for the last time the volume was updated

packet_volume_attachment

Provides attachment of Packet Block Storage Volume to Devices.

Device and volume must be in the same location (facility).

Once attached by Terraform, they must then be mounted using the `packet_block_attach` and `packet_block_detach` scripts.

Example Usage

```
resource "packet_project" "test_project" {
  name = "test-project"
}

resource "packet_device" "test_device_va" {
  hostname      = "terraform-test-device-va"
  plan          = "baremetal_0"
  facility      = "ewr1"
  operating_system = "ubuntu_16_04"
  billing_cycle  = "hourly"
  project_id     = "${packet_project.test_project.id}"
}

resource "packet_volume" "test_volume_va" {
  plan = "storage_1"
  billing_cycle = "hourly"
  size = 100
  project_id = "${packet_project.test_project.id}"
  facility = "ewr1"
  snapshot_policies = { snapshot_frequency = "1day", snapshot_count = 7 }
}

resource "packet_volume_attachment" "test_volume_attachment" {
  device_id = "${packet_device.test_device_va.id}"
  volume_id = "${packet_volume.test_volume_va.id}"
}
```

Argument Reference

The following arguments are supported:

- `volume_id` - (Required) The ID of the volume to attach
- `device_id` - (Required) The ID of the device to which the volume should be attached

Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the volume attachment