» yandex_compute_disk

Get information about a Yandex Compute disk. For more information, see the official documentation.

» Example Usage

```
data "yandex_compute_disk" "my_disk" {
  disk_id = "some_disk_id"
}

resource "yandex_compute_instance" "default" {
  ...

  secondary_disk {
    disk_id = "${data.yandex_compute_disk.my_disk.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `disk_id` - (Optional) The ID of a specific disk.
- `name` - (Optional) Name of the disk.

NOTE: One of `disk_id` or `name` should be specified.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `description` - Optional description of this disk.
- `folder_id` - ID of the folder that the disk belongs to.
- `zone` - ID of the zone where the disk resides.
- `size` - Size of the disk, specified in Gb.
- `image_id` - ID of the source image that was used to create this disk.
- `snapshot_id` - Source snapshot that was used to create this disk.
- `type` - Type of the disk.
- `status` - Status of the disk.
- `labels` - Map of labels applied to this disk.
- `product_ids` - License IDs that indicate which licenses are attached to this disk.

- `instance_ids` - IDs of instances to which this disk is attached.
- `created_at` - Disk creation timestamp.

» yandex_compute_image

Get information about a Yandex Compute image. For more information, see the official documentation.

» Example Usage

```
data "yandex_compute_image" "my_image" {
  family = "ubuntu-1804-lts"
}

resource "yandex_compute_instance" "default" {
  ...

  boot_disk {
    initialize_params {
      image_id = "${data.yandex_compute_image.my_image.id}"
    }
  }
}
```

» Argument Reference

The following arguments are supported:

- `image_id` - (Optional) The ID of a specific image.
- `family` - (Optional) The family name of an image. Used to search the latest image in a family.
- `name` - (Optional) The name of the image.

NOTE: Either `image_id`, `family` or `name` must be specified.

- `folder_id` - (Optional) Folder that the resource belongs to. If a value is not provided, the default provider folder is used.

NOTE: If you specify `family` without `folder_id` then lookup takes place in the 'standard-images' folder.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `description` - An optional description of this image.
- `family` - The OS family name of the image.
- `min_disk_size` - Minimum size of the disk which is created from this image.
- `size` - The size of the image, specified in Gb.
- `status` - The status of the image.
- `product_ids` - License IDs that indicate which licenses are attached to this image.
- `os_type` - Operating system type that the image contains.
- `labels` - A map of labels applied to this image.
- `created_at` - Image creation timestamp.

» yandex_compute_instance

Get information about a Yandex Compute instance. For more information, see the official documentation.

» Example Usage

```
data "yandex_compute_instance" "my_instance" {
  instance_id = "some_instance_id"
}

output "instance_external_ip" {
  value = "${data.yandex_compute_instance.my_instance.network_interface.0.nat_ip_address}"
}
```

» Argument Reference

The following arguments are supported:

- `instance_id` - (Optional) The ID of a specific instance.
- `name` - (Optional) Name of the instance.

NOTE: One of `instance_id` or `name` should be specified.

» Attributes Reference

- `description` - Description of the instance.
- `folder_id` - ID of the folder that the instance belongs to.
- `fqdn` - FQDN of the instance.
- `zone` - Availability zone where the instance resides.
- `labels` - A set of key/value label pairs to assign to the instance.
- `metadata` - Metadata key/value pairs to make available from within the instance.
- `platform_id` - Type of virtual machine to create. Default is 'standard-v1'.
- `status` - Status of the instance.
- `resources.0.memory` - Memory size allocated for the instance.
- `resources.0.cores` - Number of CPU cores allocated for the instance.
- `resources.0.core_fraction` - Baseline performance for a core, set as a percent.
- `boot_disk` - The boot disk for the instance. Structure is documented below.
- `network_interface` - The networks attached to the instance. Structure is documented below.
- `network_interface.0.ip_address` - An internal IP address of the instance, either manually or dynamically assigned.
- `network_interface.0.nat_ip_address` - An assigned external IP address if the instance has NAT enabled.
- `secondary_disk` - List of secondary disks attached to the instance. Structure is documented below.
- `scheduling_policy` - Scheduling policy configuration. The structure is documented below.
- `service_account_id` - ID of the service account authorized for this instance.
- `created_at` - Instance creation timestamp.

The `boot_disk` block supports:

- `auto_delete` - Whether the disk is auto-deleted when the instance is deleted. The default value is false.
- `device_name` - Name that can be used to access an attached disk under `/dev/disk/by-id/`.
- `mode` - Access to the disk resource. By default a disk is attached in `READ_WRITE` mode.
- `disk_id` - ID of the attached disk.
- `initialize_params` - Parameters used for creating a disk alongside the instance. The structure is documented below.

The `initialize_params` block supports:

- `name` - Name of the boot disk.

- `description` - Description of the boot disk.
- `size` - Size of the disk in GB.
- `type` - Disk type.
- `image_id` - A disk image to initialize this disk from.
- `snapshot_id` - A snapshot to initialize this disk from.

The `network_interface` block supports:

- `index` - The index of the network interface, generated by the server.
- `mac_address` - MAC address that is assigned to the network interface.
- `ip_address` - The private IP address to assign to the instance. If empty, the address is automatically assigned from the specified subnet.
- `subnet_id` - ID of the subnet to attach this interface to. The subnet must reside in the same zone where this instance was created.
- `nat` - Assigned for the instance's public address that is used to access the internet over NAT.
- `nat_ip_address` - Public IP address of the instance.
- `nat_ip_version` - IP version for the public address.

The `secondary_disk` block supports:

- `auto_delete` - Specifies whether the disk is auto-deleted when the instance is deleted.
- `device_name` - This value can be used to reference the device from within the instance for mounting, resizing, and so on.
- `mode` - Access to the Disk resource. By default, a disk is attached in `READ_WRITE` mode.
- `disk_id` - ID of the disk that is attached to the instance.

The `scheduling_policy` block supports:

- `preemptible` - (Optional) Specifies if the instance is preemptible. Defaults to false.

» yandex_compute_snapshot

Get information about a Yandex Compute snapshot. For more information, see the official documentation.

» Example Usage

```
data "yandex_compute_snapshot" "my_snapshot" {
  snapshot_id = "some_snapshot_id"
}

resource "yandex_compute_instance" "default" {
```

```
  ...

  boot_disk {
    initialize_params {
      snapshot_id = "${data.yandex_compute_snapshot.my_snapshot.id}"
    }
  }
}
```

» Argument Reference

The following arguments are supported:

- `snapshot_id` - (Optional) The ID of a specific snapshot.
- `name` - (Optional) The name of the snapshot.

NOTE: One of `snapshot_id` or `name` should be specified.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `description` - An optional description of this snapshot.
- `folder_id` - ID of the folder that the snapshot belongs to.
- `storage_size` - The size of the snapshot, specified in Gb.
- `status` - The status of the snapshot.
- `disk_size` - Minimum required size of the disk which is created from this snapshot.
- `source_disk_id` - ID of the source disk.
- `labels` - A map of labels applied to this snapshot.
- `product_ids` - License IDs that indicate which licenses are attached to this snapshot.
- `created_at` - Snapshot creation timestamp.

» yandex_iam_policy

Generates an IAM policy document that may be referenced by and applied to other Yandex.Cloud Platform resources, such as the `yandex_resourcemanager_folder` resource.

```
data "yandex_iam_policy" "admin" {
  binding {
    role = "admin"
```

```
    members = [
      "userAccount:user_id_1"
    ]
  }

  binding {
    role = "viewer"

    members = [
      "userAccount:user_id_2"
    ]
  }
}
```

This data source is used to define IAM policies to apply to other resources. Currently, defining a policy through a data source and referencing that policy from another resource is the only way to apply an IAM policy to a resource.

» Argument Reference

The following arguments are supported:

- `binding` (Required) - A nested configuration block (described below) that defines a binding to be included in the policy document. Multiple `binding` arguments are supported.

Each policy document configuration must have one or more `binding` blocks. Each block accepts the following arguments:

- `role` (Required) - The role/permission that will be granted to the members. See the IAM Roles documentation for a complete list of roles.
- `members` (Required) - An array of identities that will be granted the privilege in the `role`. Each entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Attributes Reference

The following attribute is exported:

- `policy_data` - The above bindings serialized in a format suitable for referencing from a resource that supports IAM.

» yandex_iam_role

Generates an IAM role document that may be referenced by and applied to other Yandex.Cloud Platform resources, such as the `yandex_resourcemanager_folder` resource. For more information, see the official documentation.

```
data "yandex_iam_role" "admin" {
  binding {
    role = "admin"

    members = [
      "userAccount:user_id_1"
    ]
  }
}
```

This data source is used to define IAM roles in order to apply them to other resources. Currently, defining a role through a data source and referencing that role from another resource is the only way to apply an IAM role to a resource.

» Argument Reference

The following arguments are supported:

- `binding` (Required) - A nested configuration block (described below) that defines a binding to be included in the policy document. Multiple `binding` arguments are supported.

Each role document configuration must have one or more `binding` blocks. Each block accepts the following arguments:

- `role` (Required) - The role/permission that will be granted to the members. See the IAM Roles documentation for a complete list of roles.
- `members` (Required) - An array of identities that will be granted the privilege in the `role`. Each entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Attributes Reference

The following attribute is exported:

- `role_data` - The above bindings serialized in a format suitable for referencing from a resource that supports IAM.

» yandex_iam_service_account

Get information about a Yandex IAM service account. For more information about accounts, see Yandex.Cloud IAM users.

```
data "yandex_iam_service_account" "builder" {
  service_account_id = "sa_id"
}
```

» Argument reference

- `name` - Name of the service account. Can be updated without creating a new resource.
- `description` - Description of the service account.
- `folder_id` - ID of the folder that the service account will be created in. If omitted, the provider folder configuration is used by default.

» yandex_iam_user

Get information about a Yandex IAM user account. For more information about accounts, see Yandex.Cloud IAM users

```
data "yandex_iam_user" "admin" {
  login = "my-yandex-login"
}
```

This data source is used to define IAM Users that can be used by other resources.

» Argument Reference

The following arguments are supported:

- `login` (Optional) - Login name used to sign in to Yandex Passport.
- `user_id` (Optional) - User ID used to manage IAM access bindings.

NOTE: Either `login` or `user_id` must be specified.

» Attributes Reference

The following attribute is exported:

- `user_id` - ID of IAM user account.
- `login` - Login name of IAM user account.

- `default_email` - Email address of user account.

» yandex_resourcemanager_cloud

Use this data source to get cloud details. For more information, see Cloud.

» Example Usage

```
data "yandex_resourcemanager_cloud" "foo" {
  name = "foo-cloud"
}

output "cloud_create_timestamp" {
  value = "${data.yandex_resourcemanager_cloud.foo.created_at}"
}
```

» Argument Reference

The following arguments are supported:

- `cloud_id` - (Optional) ID of the cloud.
- `name` - (Optional) Name of the cloud.

NOTE: Either `cloud_id` or `name` must be specified.

» Attributes Reference

The following attributes are returned:

- `name` - Name of the cloud.
- `description` - Description of the cloud.
- `created_at` - Cloud creation timestamp.

» yandex_resourcemanager_folder

Use this data source to get information about a Yandex Resource Manager Folder. For more information, see the official documentation.

```
# Get folder by ID
data "yandex_resourcemanager_folder" "my_folder_1" {
  folder_id = "folder_id_number_1"
}
```

```
# Search by fields
data "yandex_resourcemanager_folder" "my_folder_2" {
  folder_id = "folder_id_number_2"
}

output "my_folder_1_name" {
  value = "${data.yandex_resourcemanager_folder.my_folder_1.name}"
}

output "my_folder_2_cloud_id" {
  value = "${data.yandex_resourcemanager_folder.my_folder_2.cloud_id}"
}
```

» Argument Reference

The following arguments are supported:

- `folder_id` (Optional) - ID of the folder.
- `name` (Optional) - Name of the Folder.

NOTE: Either `folder_id` or `name` must be specified.

» Attributes Reference

The following attributes are exported:

- `description` - Description of the folder.
- `cloud_id` - ID of the cloud that contains the folder.
- `status` - Current status of the folder.
- `labels` - A map of labels applied to this folder.
- `created_at` - Folder creation timestamp.

» yandex_vpc_network

Get information about a Yandex VPC network. For more information, see Yandex.Cloud VPC.

```
data "yandex_vpc_network" "admin" {
  network_id = "my-network-id"
}
```

This data source is used to define VPC Networks that can be used by other resources.

» Argument Reference

The following arguments are supported:

- `network_id` (Optional) - ID of the network.
- `name` (Optional) - Name of the network.

NOTE: One of `network_id` or `name` should be specified.

» Attributes Reference

The following attribute is exported:

- `description` - Description of the network.
- `folder_id` - ID of the folder that the resource belongs to.
- `labels` - Labels assigned to this network.
- `created_at` - Creation timestamp of this network.

» yandex_vpc_route_table

Get information about a Yandex VPC route table. For more information, see Yandex.Cloud VPC.

```
data "yandex_vpc_route_table" "this" {
  route_table_id = "my-rt-id"
}
```

This data source is used to define VPC Route Table that can be used by other resources.

» Argument Reference

The following arguments are supported:

- `route_table_id` (Optional) - Route table ID.
- `name` - (Optional) - Name of the route table.

NOTE: One of `route_table_id` or `name` should be specified.

» Attributes Reference

The following attribute is exported:

- `description` - Description of the route table.

- `folder_id` - ID of the folder that the resource belongs to.
- `network_id` - ID of the network this route table belongs to.
- `labels` - Labels to assign to this route table.
- `static_route` - List of static route records of the route table. Structure is documented below.
- `created_at` - Creation timestamp of this route table.

The `static_route` block supports:

- `destination_prefix` - Route prefix in CIDR notation.
- `next_hop_address` - Address of the next hop.

» yandex_vpc_subnet

Get information about a Yandex VPC subnet. For more information, see Yandex.Cloud VPC.

```
data "yandex_vpc_subnet" "admin" {
  subnet_id = "my-subnet-id"
}
```

This data source is used to define VPC Subnets that can be used by other resources.

» Argument Reference

The following arguments are supported:

- `subnet_id` (Optional) - Subnet ID.
- `name` - (Optional) - Name of the subnet.

NOTE: One of `subnet_id` or `name` should be specified.

» Attributes Reference

The following attribute is exported:

- `description` - Description of the subnet.
- `folder_id` - ID of the folder that the resource belongs to.
- `network_id` - ID of the network this subnet belongs to.
- `labels` - Labels to assign to this subnet.
- `zone` - Name of the availability zone for this subnet.
- `route_table_id` - ID of the route table to assign to this subnet.
- `v4_cidr_blocks` - The blocks of internal IPv4 addresses owned by this subnet.

- `v6_cidr_blocks` - The blocks of internal IPv6 addresses owned by this subnet.
- `created_at` - Creation timestamp of this subnet.

Note: `v6_cidr_blocks` attribute is currently not supported. It will be available in the future.

» yandex_compute_disk

Persistent disks are used for data storage and function similarly to physical hard and solid state drives.

A disk can be attached or detached from the virtual machine and can be located locally. A disk can be moved between virtual machines within the same availability zone. Each disk can be attached to only one virtual machine at a time.

For more information about disks in Yandex.Cloud, see:

- Documentation
- How-to Guides
  - Attach and detach a disk
  - Backup operation

» Example Usage

```
resource "yandex_compute_disk" "default" {
  name     = "disk"
  type     = "network-nvme"
  zone     = "ru-central1-a"
  image_id = "ubuntu-16.04-v20180727"

  labels {
    environment = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Optional) Name of the disk. Provide this property when you create a resource.

- `description` - (Optional) Description of the disk. Provide this property when you create a resource.
- `folder_id` - (Optional) The ID of the folder that the disk belongs to. If it is not provided, the default provider folder is used.
- `labels` - (Optional) Labels to assign to this disk. A list of key/value pairs.
- `zone` - (Optional) Availability zone where the disk will reside.
- `size` - (Optional) Size of the persistent disk, specified in GB. You can specify this field when creating a persistent disk using the `image_id` or `snapshot_id` parameter, or specify it alone to create an empty persistent disk. If you specify this field along with `image_id` or `snapshot_id`, the size value must not be less than the size of the source image or the size of the snapshot.
- `type` - (Optional) Type of disk to create. Provide this when creating a disk. One of `network-hdd` (default) or `network-nvme`.
- `image_id` - (Optional) The source image to use for disk creation.
- `snapshot_id` - (Optional) The source snapshot to use for disk creation.

NOTE: Only one of `image_id` or `snapshot_id` can be specified.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `status` - The status of the disk.
- `created_at` - Creation timestamp of the disk.

» Timeouts

This resource provides the following configuration options for timeouts:

- `create` - Default is 5 minutes.
- `update` - Default is 5 minutes.
- `delete` - Default is 5 minutes.

» Import

A disk can be imported using any of these accepted formats:

```
$ terraform import yandex_compute_disk.default disk_id
```

» yandex_compute_image

Creates a virtual machine image resource for the Yandex Compute Cloud service from an existing tarball. For more information, see the official documentation.

» Example Usage

```
resource "yandex_compute_image" "foo-image" {
  name       = "my-custom-image"
  source_url = "https://storage.yandexcloud.net/lucky-images/kube-it.img"
}

resource "yandex_compute_instance" "vm" {
  name = "vm-from-custom-image"

  # ...

  boot_disk {
    initialize_params {
      image_id = "${yandex_compute_image.foo-image.id}"
    }
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Optional) Name of the disk.
- `description` - (Optional) An optional description of the image. Provide this property when you create a resource.
- `folder_id` - (Optional) The ID of the folder that the resource belongs to. If it is not provided, the default provider folder is used.
- `labels` - (Optional) A set of key/value label pairs to assign to the image.
- `family` - (Optional) The name of the image family to which this image belongs.
- `min_disk_size` - (Optional) Minimum size in GB of the disk that will be created from this image.
- `os_type` - (Optional) Operating system type that is contained in the image. Possible values: ”LINUX”, ”WINDOWS”.

- `source_family` - (Optional) The name of the family to use as the source of the new image. The ID of the latest image is taken from the "standard-images" folder. Changing the family forces a new resource to be created.
- `source_image` - (Optional) The ID of an existing image to use as the source of the image. Changing this ID forces a new resource to be created.
- `source_snapshot` - (Optional) The ID of a snapshot to use as the source of the image. Changing this ID forces a new resource to be created.
- `source_disk` - (Optional) The ID of a disk to use as the source of the image. Changing this ID forces a new resource to be created.
- `source_url` - (Optional) The URL to use as the source of the image. Changing this URL forces a new resource to be created.
- `product_ids` - (Optional) License IDs that indicate which licenses are attached to this image.

NOTE: One of `source_family`, `source_image`, `source_snapshot`, `source_disk` or `source_url` must be specified.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `size` - The size of the image, specified in GB.
- `status` - The status of the image.
- `created_at` - Creation timestamp of the image.

» Timeouts

`yandex_compute_image` provides the following configuration options for timeouts:

- `create` - Default 5 minutes
- `update` - Default 5 minutes
- `delete` - Default 5 minutes

» Import

A VM image can be imported using the `id` of the resource, e.g.

```
$ terraform import yandex_compute_image.web-image image_id
```

» yandex_compute_instance

A VM instance resource. For more information, see the official documentation.

» Example Usage

```
resource "yandex_compute_instance" "default" {
  name        = "test"
  platform_id = "standard-v1"
  zone        = "ru-central1-a"

  resources {
    cores  = 2
    memory = 4
  }

  boot_disk {
    initialize_params {
      image_id = "image_id"
    }
  }

  network_interface {
    subnet_id = "${yandex_vpc_subnet.foo.id}"
  }

  metadata {
    foo      = "bar"
    ssh-keys = "ubuntu:${file("~/.ssh/id_rsa.pub")}"
  }
}

resource "yandex_vpc_network" "foo" {}

resource "yandex_vpc_subnet" "foo" {
  zone       = "ru-central1-a"
  network_id = "${yandex_vpc_network.foo.id}"
}
```

» Argument Reference

The following arguments are supported:

- `resources` - (Required) Compute resources that are allocated for the instance. The structure is documented below.
- `boot_disk` - (Required) The boot disk for the instance. The structure is documented below.
- `network_interface` - (Required) Networks to attach to the instance. This can be specified multiple times. The structure is documented below.

- `name` - (Optional) Resource name.
- `description` - (Optional) Description of the instance.
- `folder_id` - (Optional) The ID of the folder that the resource belongs to. If it is not provided, the default provider folder is used.
- `labels` - (Optional) A set of key/value label pairs to assign to the instance.
- `zone` - (Optional) The availability zone where the virtual machine will be created. If it is not provided, the default provider folder is used.
- `hostname` - (Optional) Host name for the instance. This field is used to generate the instance `fqdn` value. The host name must be unique within the network and region. If not specified, the host name will be equal to `id` of the instance and `fqdn` will be `<id>.auto.internal`. Otherwise FQDN will be `<hostname>.<region_id>.internal`.
- `metadata` - (Optional) Metadata key/value pairs to make available from within the instance.
- `platform_id` - (Optional) The type of virtual machine to create. The default is 'standard-v1'.
- `secondary_disk` - (Optional) A list of disks to attach to the instance. The structure is documented below. **Note**: The `allow_stopping_for_update` property must be set to true in order to update this structure.
- `scheduling_policy` - (Optional) Scheduling policy configuration. The structure is documented below.
- `service_account_id` - (Optional) ID of the service account authorized for this instance.
- `allow_stopping_for_update` - (Optional) If true, allows Terraform to stop the instance in order to update its properties. If you try to update a property that requires stopping the instance without setting this field, the update will fail.

The `resources` block supports:

- `cores` - (Required) CPU cores for the instance.

- `memory` - (Required) Memory size in GB.
- `core_fraction` - (Optional) If provided, specifies baseline performance for a core as a percent.

The `boot_disk` block supports:

- `auto_delete` - (Optional) Defines whether the disk will be auto-deleted when the instance is deleted. The default value is `True`.
- `device_name` - (Optional) Name that can be used to access an attached disk.
- `mode` - (Optional) Type of access to the disk resource. By default, a disk is attached in `READ_WRITE` mode.
- `disk_id` - (Optional) The ID of the existing disk (such as those managed by `yandex_compute_disk`) to attach as a boot disk.
- `initialize_params` - (Optional) Parameters for a new disk that will be created alongside the new instance. Either `initialize_params` or `disk_id` must be set. The structure is documented below.

NOTE: Either `initialize_params` or `disk_id` must be specified.

The `initialize_params` block supports:

- `name` - (Optional) Name of the boot disk.
- `description` - (Optional) Description of the boot disk.
- `size` - (Optional) Size of the disk in GB.
- `type` - (Optional) Disk type.
- `image_id` - (Optional) A disk image to initialize this disk from.
- `snapshot_id` - (Optional) A snapshot to initialize this disk from.

NOTE: Either `image_id` or `snapshot_id` must be specified.

The `network_interface` block supports:

- `subnet_id` - (Required) ID of the subnet to attach this interface to. The subnet must exist in the same zone where this instance will be created.
- `ip_address` - (Optional) The private IP address to assign to the instance. If empty, the address will be automatically assigned from the specified subnet.
- `ipv6` - (Optional) If true, allocate an IPv6 address for the interface. The address will be automatically assigned from the specified subnet.
- `ipv6_address` - (Optional) The private IPv6 address to assign to the instance.

- `nat` - (Optional) Provide a public address, for instance, to access the internet over NAT.

The `secondary_disk` block supports:

- `disk_id` - (Required) ID of the disk that is attached to the instance.
- `auto_delete` - (Optional) Whether the disk is auto-deleted when the instance is deleted. The default value is false.
- `device_name` - (Optional) Name that can be used to access an attached disk under `/dev/disk/by-id/`.
- `mode` - (Optional) Type of access to the disk resource. By default, a disk is attached in `READ_WRITE` mode.

The `scheduling_policy` block supports:

- `preemptible` - (Optional) Specifies if the instance is preemptible. Defaults to false.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `fqdn` - The fully qualified DNS name of this instance.
- `network_interface.0.address` - The internal IP address of the instance.
- `network_interface.0.nat_ip_address` - The external IP address of the instance.
- `status` - The status of this instance.
- `created_at` - Creation timestamp of the instance.

» Import

Instances can be imported using the `ID` of an instance, e.g.

```
$ terraform import yandex_compute_instance.default instance_id
```

» yandex_compute_snapshot

Creates a new snapshot of a disk. For more information, see the official documentation.

» Example Usage

```
resource "yandex_compute_snapshot" "default" {
  name           = "test-snapshot"
  source_disk_id = "test_disk_id"

  labels {
    my-label = "my-label-value"
  }
}
```

» Argument Reference

The following arguments are supported:

- `source_disk_id` - (Required) ID of the disk to create a snapshot from.

- `name` - (Optional) A name for the resource.
- `description` - (Optional) Description of the resource.
- `folder_id` - (Optional) The ID of the folder that the resource belongs to. If it is not provided, the default provider folder is used.
- `labels` - (Optional) A set of key/value label pairs to assign to the snapshot.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `disk_size` - Size of the disk when the snapshot was created, specified in GB.
- `storage_size` - Size of the snapshot, specified in GB.
- `created_at` - Creation timestamp of the snapshot.

» yandex_iam_service_account

Allows management of a Yandex.Cloud IAM service account. To assign roles and permissions, use the yandex_iam_service_account_iam_binding, yandex_iam_service_account_iam_member and yandex_iam_service_account_iam_policy resources.

» Example Usage

This snippet creates a service account.

```
resource "yandex_iam_service_account" "sa" {
  name        = "VM Manager"
  description = "service account to manage VMs"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Optional) Name of the service account. Can be updated without creating a new resource.
- `description` - (Optional) Description of the service account.
- `folder_id` - (Optional) ID of the folder that the service account will be created in. Defaults to the provider folder configuration.

» Import

Service accounts can be imported using their IDs, e.g.

```
$ terraform import yandex_iam_service_account.my_sa service_account_id
```

» IAM policy for a service account

When managing IAM roles, you can treat a service account either as a resource or as an identity. This resource is used to add IAM policy bindings to a service account resource to configure permissions that define who can edit the service account.

There are three different resources that help you manage your IAM policy for a service account. Each of these resources is used for a different use case:

- yandex_iam_service_account_iam_policy: Authoritative. Sets the IAM policy for the service account and replaces any existing policy already attached.
- yandex_iam_service_account_iam_binding: Authoritative for a given role. Updates the IAM policy to grant a role to a list of members. Other roles within the IAM policy for the service account are preserved.
- yandex_iam_service_account_iam_member: Non-authoritative. Updates the IAM policy to grant a role to a new member. Other members for the role of the service account are preserved.

Note: `yandex_iam_service_account_iam_policy` **cannot** be used in conjunction with `yandex_iam_service_account_iam_binding` and `yandex_iam_service_account_iam_member` or they will conflict over what your policy should be.

Note: `yandex_iam_service_account_iam_binding` resources **can be** used in conjunction with `yandex_iam_service_account_iam_member` resources **only if** they do not grant privileges to the same role.

» yandex_service_account_iam_binding

```
resource "yandex_iam_service_account_iam_binding" "admin-account-iam" {
  service_account_id = "your-service-account-id"
  role               = "admin"

  members = [
    "userAccount:foo_user_id",
  ]
}
```

» Argument Reference

The following arguments are supported:

- `service_account_id` - (Required) The service account ID to apply a binding to.
- `role` - (Required) The role that should be applied. Only one `yandex_iam_service_account_iam_binding` can be used per role.
- `members` - (Required) Identities that will be granted the privilege in `role`. Each entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

Service account IAM binding resources can be imported using the service account ID and role.

```
$ terraform import yandex_iam_service_account_iam_binding.admin-account-iam "service_account
```

» IAM policy for a service account

When managing IAM roles, you can treat a service account either as a resource or as an identity. This resource is used to add IAM policy bindings to a service account resource to configure permissions that define who can edit the service account.

There are three different resources that help you manage your IAM policy for a service account. Each of these resources is used for a different use case:

- yandex_iam_service_account_iam_policy: Authoritative. Sets the IAM policy for the service account and replaces any existing policy already attached.
- yandex_iam_service_account_iam_binding: Authoritative for a given role. Updates the IAM policy to grant a role to a list of members. Other roles within the IAM policy for the service account are preserved.
- yandex_iam_service_account_iam_member: Non-authoritative. Updates the IAM policy to grant a role to a new member. Other members for the role of the service account are preserved.

Note: `yandex_iam_service_account_iam_policy` **cannot** be used in conjunction with `yandex_iam_service_account_iam_binding` and `yandex_iam_service_account_iam_member` or they will conflict over what your policy should be.

Note: `yandex_iam_service_account_iam_binding` resources **can be** used in conjunction with `yandex_iam_service_account_iam_member` resources **only if** they do not grant privileges to the same role.

» yandex_service_account_iam_member

```
resource "yandex_iam_service_account_iam_member" "admin-account-iam" {
  service_account_id = "your-service-account-id"
  role               = "admin"
  member             = "userAccount:bar_user_id"
}
```

» Argument Reference

The following arguments are supported:

- `service_account_id` - (Required) The service account ID to apply a policy to.
- `role` - (Required) The role that should be applied. Only one `yandex_iam_service_account_iam_binding` can be used per role.

- `member` - (Required) Identity that will be granted the privilege in `role`. Entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

Service account IAM member resources can be imported using the service account ID, role and member.

```
$ terraform import yandex_iam_service_account_iam_member.admin-account-iam "service_account_
```

» IAM policy for a service account

When managing IAM roles, you can treat a service account either as a resource or as an identity. This resource is used to add IAM policy bindings to a service account resource to configure permissions that define who can edit the service account.

There are three different resources that help you manage your IAM policy for a service account. Each of these resources is used for a different use case:

- yandex_iam_service_account_iam_policy: Authoritative. Sets the IAM policy for the service account and replaces any existing policy already attached.
- yandex_iam_service_account_iam_binding: Authoritative for a given role. Updates the IAM policy to grant a role to a list of members. Other roles within the IAM policy for the service account are preserved.
- yandex_iam_service_account_iam_member: Non-authoritative. Updates the IAM policy to grant a role to a new member. Other members for the role of the service account are preserved.

Note: `yandex_iam_service_account_iam_policy` **cannot** be used in conjunction with `yandex_iam_service_account_iam_binding` and `yandex_iam_service_account_iam_member` or they will conflict over what your policy should be.

Note: `yandex_iam_service_account_iam_binding` resources **can be** used in conjunction with `yandex_iam_service_account_iam_member` resources **only if** they do not grant privileges to the same role.

» yandex_service_account_iam_policy

```
data "yandex_iam_policy" "admin" {
  binding {
    role = "admin"

    members = [
      "userAccount:foobar_user_id",
    ]
  }
}

resource "yandex_iam_service_account_iam_policy" "admin-account-iam" {
  service_account_id = "your-service-account-id"
  policy_data        = "${data.yandex_iam_policy.admin.policy_data}"
}
```

» Argument Reference

The following arguments are supported:

- `service_account_id` - (Required) The service account ID to apply a policy to.
- `members` - (Required) Identities that will be granted the privilege in `role`. Each entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.
- `role` - (Required) The role that should be applied. Only one `yandex_iam_service_account_iam_binding` can be used per role.
- `policy_data` - (Required only by `yandex_iam_service_account_iam_policy`) The policy data generated by a `yandex_iam_policy` data source.

» Import

Service account IAM policy resources can be imported using the service account ID.

```
$ terraform import yandex_iam_service_account_iam_policy.admin-account-iam service_account_i
```

» yandex_iam_service_account_static_access_key

Allows management of a Yandex.Cloud IAM service account static access keys. Generated pair of keys are used to access Yandex Object Storage on behalf of service account.

Before use keys do not forget to assign a proper role to a service account.

» Example Usage

This snippet creates a service account static access key.

```
resource "yandex_iam_service_account_static_access_key" "sa-static-key" {
  service_account_id = "some_sa_id"
  description        = "static access key for object storage"
}
```

» Argument Reference

The following arguments are supported:

- `service_account_id` - (Required) ID of the service account which is used to get a static key.

- `description` - (Optional) The description of the service account static key.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `access_key` - ID of the static access key.
- `secret_key` - Private part of generated static access key.
- `created_at` - Creation timestamp of the static access key.

» yandex_resourcemanager_cloud_iam_binding

Allows creation and management of a single binding within IAM policy for an existing Yandex Resource Manager cloud.

» Example Usage

```
data "yandex_resourcemanager_cloud" "project1" {
  name = "Project 1"
}

resource "yandex_resourcemanager_cloud_iam_binding" "admin" {
  cloud_id = "${data.yandex_resourcemanager_cloud.project1.id}"

  role = "editor"

  members = [
    "userAccount:some_user_id",
  ]
}
```

» Argument Reference

The following arguments are supported:

- `cloud_id` - (Required) ID of the cloud to attach the policy to.
- `role` - (Required) The role that should be assigned. Only one `yandex_resourcemanager_cloud_iam_binding` can be used per role.
- `members` - (Required) An array of identities that will be granted the privilege in the `role`. Each entry can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

IAM binding imports use space-delimited identifiers; first the resource in question and then the role. These bindings can be imported using the `cloud_id` and role, e.g.

```
$ terraform import yandex_resourcemanager_cloud_iam_binding.viewer "cloud_id viewer"
```

» yandex_resourcemanager_cloud_iam_member

Allows creation and management of a single member for a single binding within the IAM policy for an existing Yandex Resource Manager cloud.

Note: Roles controlled by `yandex_resourcemanager_cloud_iam_binding` should not be assigned using `yandex_resourcemanager_cloud_iam_member`.

» Example Usage

```
data "yandex_resourcemanager_cloud" "department1" {
  name = "Department 1"
}

resource "yandex_resourcemanager_cloud_iam_member" "admin" {
  cloud_id = "${data.yandex_resourcemanager_cloud.department1.id}"
  role     = "editor"
  member   = "userAccount:user_id"
}
```

» Argument Reference

The following arguments are supported:

- `cloud_id` - (Required) ID of the cloud to attach a policy to.
- `role` - (Required) The role that should be assigned.
- `member` - (Required) The identity that will be granted the privilege that is specified in the `role` field. This field can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

IAM member imports use space-delimited identifiers; the resource in question, the role, and the account. This member resource can be imported using the `cloud id`, role, and account, e.g.

```
$ terraform import yandex_resourcemanager_cloud_iam_member.my_project "cloud_id viewer foo@e
```

» yandex_resourcemanager_folder_iam_binding

Allows creation and management of a single binding within IAM policy for an existing Yandex Resource Manager folder.

Note: This resource *must not* be used in conjunction with `yandex_resourcemanager_folder_iam_policy` or they will conflict over what your policy should be.

» Example Usage

```
data "yandex_resourcemanager_folder" "project1" {
  folder_id = "some_folder_id"
}

resource "yandex_resourcemanager_folder_iam_binding" "admin" {
  folder_id = "${data.yandex_resourcemanager_folder.project1.id}"

  role = "editor"

  members = [
    "userAccount:some_user_id",
  ]
}
```

» Argument Reference

The following arguments are supported:

- `folder_id` - (Required) ID of the folder to attach a policy to.
- `role` - (Required) The role that should be assigned. Only one `yandex_resourcemanager_folder_iam_binding` can be used per role.
- `members` - (Required) An array of identities that will be granted the privilege that is specified in the `role` field. Each entry can have one of the following values:
  - **userAccount:{user_id}**: An email address that represents a specific Yandex account. For example, ivan@yandex.ru or joe@example.com.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

IAM binding imports use space-delimited identifiers; first the resource in question and then the role. These bindings can be imported using the `folder_id` and role, e.g.

```
$ terraform import yandex_resourcemanager_folder_iam_binding.viewer "folder_id viewer"
```

» yandex_resourcemanager_folder_iam_member

Allows creation and management of a single member for a single binding within the IAM policy for an existing Yandex Resource Manager folder.

Note: This resource *must not* be used in conjunction with `yandex_resourcemanager_folder_iam_policy` or they will conflict over what your policy should be. Similarly, roles controlled by `yandex_resourcemanager_folder_iam_binding` should not be assigned using `yandex_resourcemanager_folder_iam_member`.

» Example Usage

```
data "yandex_resourcemanager_folder" "department1" {
  folder_id = "some_folder_id"
}

resource "yandex_resourcemanager_folder_iam_member" "admin" {
  folder_id = "${data.yandex_resourcemanager.department1.name}"

  role   = "editor"
  member = "userAccount:user_id"
}
```

» Argument Reference

The following arguments are supported:

- `folder_id` - (Required) ID of the folder to attach a policy to.
- `role` - (Required) The role that should be assigned.
- `member` - (Required) The identity that will be granted the privilege that is specified in the `role` field. This field can have one of the following values:
  - **userAccount:{user_id}**: A unique user ID that represents a specific Yandex account.
  - **serviceAccount:{service_account_id}**: A unique service account ID.

» Import

IAM member imports use space-delimited identifiers; the resource in question, the role, and the account. This member resource can be imported using the `folder id`, role, and account, e.g.

```
$ terraform import yandex_resourcemanager_folder_iam_member.my_project "folder_id viewer foo
```

» yandex_folder_iam_policy

Allows creation and management of the IAM policy for an existing Yandex Resource Manager folder.

» Example Usage

```
data "yandex_resourcemanager_folder" "project1" {
  folder_id = "my_folder_id"
}

data "yandex_iam_policy" "admin" {
  binding {
    role = "editor"

    members = [
      "userAccount:some_user_id",
    ]
  }
}

resource "yandex_resourcemanager_iam_policy" "folder_admin_policy" {
  folder_id   = "${data.yandex_folder.project1.id}"
  policy_data = "${data.yandex_iam_policy.admin.policy_data}"
}
```

» Argument Reference

The following arguments are supported:

- `folder_id` - (Required) ID of the folder that the policy is attached to.
- `policy_data` - (Required) The `yandex_iam_policy` data source that represents the IAM policy that will be applied to the folder. This policy overrides any existing policy applied to the folder.

» yandex_vpc_network

Manages a network within the Yandex.Cloud. For more information, see the official documentation.

- How-to Guides
  - Cloud Networking

  - VPC Addressing

» Example Usage

```
resource "yandex_vpc_network" "default" {
  name = "foobar"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Optional) Name of the network. Provided by the client when the network is created.
- `description` - (Optional) An optional description of this resource. Provide this property when you create the resource.
- `folder_id` - (Optional) ID of the folder that the resource belongs to. If it is not provided, the default provider folder is used.
- `labels` - (Optional) Labels to apply to this network. A list of key/value pairs.

» Attributes Reference

In addition to the arguments listed above, the following computed attributes are exported:

- `created_at` - Creation timestamp of the key.

» Import

A network can be imported using the `id` of the resource, e.g.

```
$ terraform import yandex_vpc_network.default network_id
```

» yandex_vpc_route_table

Manages a route table within the Yandex.Cloud. For more information, see the official documentation.

- How-to Guides
  - Cloud Networking

» Example Usage

```
resource "yandex_vpc_network" "lab-net" {
  name = "lab-network"
}

resource "yandex_vpc_route_table" "lab-rt-a" {
  network_id = "${yandex_vpc_network.lab-net.id}"

  static_route = {
    destination_prefix = "10.2.0.0/16"
    next_hop_address   = "172.16.10.10"
  }
}
```

» Argument Reference

The following arguments are supported:

- `network_id` - (Required) ID of the network this route table belongs to.

- `name` - (Optional) Name of the route table. Provided by the client when the route table is created.
- `description` - (Optional) An optional description of the route table. Provide this property when you create the resource.
- `folder_id` - (Optional) The ID of the folder to which the resource belongs. If omitted, the provider folder is used.
- `labels` - (Optional) Labels to assign to this route table. A list of key/value pairs.
- `static_route` - (Optional) A list of static route records for the route table. The structure is documented below.

The `static_route` block supports:

- `destination_prefix` - Route prefix in CIDR notation.
- `next_hop_address` - Address of the next hop.

» Attributes Reference

- `created_at` - Creation timestamp of the route table.

» Import

A route table can be imported using the `id` of the resource, e.g.:

```
$ terraform import yandex_vpc_route_table.default route_table_id
```

» yandex_vpc_subnet

Manages a subnet within the Yandex.Cloud. For more information, see the official documentation.

- How-to Guides
  - Cloud Networking
  - VPC Addressing

» Example Usage

```
resource "yandex_vpc_network" "lab-net" {
  name = "lab-network"
}

resource "yandex_vpc_subnet" "lab-subnet-a" {
  v4_cidr_blocks = ["10.2.0.0/16"]
  zone           = "ru-central1-a"
  network_id     = "${yandex_vpc_network.lab-net.id}"
}
```

» Argument Reference

The following arguments are supported:

- `network_id` - (Required) ID of the network this subnet belongs to. Only networks that are in the distributed mode can have subnets.
- `v4_cidr_blocks` - (Required) A list of blocks of internal IPv4 addresses that are owned by this subnet. Provide this property when you create the subnet. For example, 10.0.0.0/22 or 192.168.0.0/16. Blocks of addresses must be unique and non-overlapping within a network. Minimum subnet size is /28, and maximum subnet size is /16. Only IPv4 is supported.
- `zone` - (Required) Name of the Yandex.Cloud zone for this subnet.

- `name` - (Optional) Name of the subnet. Provided by the client when the subnet is created.

- `description` - (Optional) An optional description of the subnet. Provide this property when you create the resource.
- `folder_id` - (Optional) The ID of the folder to which the resource belongs. If omitted, the provider folder is used.
- `labels` - (Optional) Labels to assign to this subnet. A list of key/value pairs.
- `route_table_id` - (Optional) The ID of the route table to assign to this subnet. Assigned route table should belong to the same network as this subnet.
- `v6_cidr_blocks` - (Optional) An optional list of blocks of IPv6 addresses that are owned by this subnet.

Note: The `v6_cidr_blocks` attribute is currently not supported. It will be available in the future.

» Timeouts

This resource provides the following configuration options for timeouts:

- `create` - Default is 3 minute.
- `update` - Default is 3 minute.
- `delete` - Default is 3 minute.

» Import

A subnet can be imported using the `id` of the resource, e.g.:

```
$ terraform import yandex_vpc_subnet.default subnet_id
```