JDCloud

JDCloud provider helps managing resources on JDCloud (https://www.jdcloud.com/en/). Before you start with this plugin, you have to provide a pair of access key and secret to identify yourself.

> Navigation bar on the left gives you a brief mind on how to manage resources on JDCloud that is currently available.

Authentication

Credential consists of your key pairs and the region id, which is used for authentication. Currently you can set up your credential in two ways:

- Simply write them in your configuration file
- Set them as environment variables

Write them in your configuration file

For example, a credential can look like this. Place this at the beginning of `jdcloud.tf`

```
provider "jdcloud" {
  access_key = "your_access_key"
  secret_key = "your_secret_key"
  region     = "cn-north-1"
}
```

Set as environment variable

Or you can set them as environment variable via command line

```
$ export access_key="your_access_key"
$ export secret_key="your_secret_key"
$ export region="cn-north-1"
```

And leave the provider field blank in configuration file. Terraform will load them automatically.

```
provider "jdcloud" {

}
```

jdcloud_availability_group

Provides a JDCloud Availability Group

Example Usage

```
resource "jdcloud_availability_group" "ag_01" {
  availability_group_name = "example_ag_name"
  az = ["cn-north-1a","cn-north-1b"]
  instance_template_id = "example_template_id"
  description  = "This is an example description"
  ag_type = "docker"
}
```

Argument Reference

The following arguments are supported

- `availability_group_name` - (Required) : A string to name this availability group
- `az` - (Required) : Az is a slice consists of strings. All azs has to be in same region, e.g. cn-north-1
- `instance_template_id` - (Required) : A string. All instances within this Ag are created under a certain template,specify this template with its id
- `ag_type` - (Optional) : A string. It decides which kind of 'instance' you are expecting, this fields can be `docker` or `kvm`, by default it is set to `kvm`
- `description` - (Optional) : Describe this Ag, if needed.

Attributes Reference

The following attributes are exported:

- `id` - The id of this Ag, can be used to reference this availability group.

jdcloud_disk_attachment

After a disk has been created, you probably need to attach it to an instance

Example Usage

```
resource "jdcloud_disk_attachment" "disk-attachment-example"{
  instance_id = "i-example"
  disk_id = "vol-example"
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required): The id of target instance
- `disk_id` - (Required): The id of disk
- `auto_delete` - (Optional): If this field is set to true, disk will be deleted after it has been detached from instance
- `device_name` - (Optional) : Specify the logical attachment point , for example, attachment point can be "vba" "vbc" etc. Just to make sure this point is available with no other device using it.

jdcloud_disk

Provides a JDCloud disk

Currently disks paid by "prepaid_by_duration" cannot be deleted before they are expired

Example Usage

```
resource "jdcloud_disk" "example" {
  az           = "cn-north-1a"
  name         = "example_disk"
  description  = "test"
  disk_type    = "ssd"
  disk_size_gb = 60
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required): A string to name this cloud disk
- `disk_size_gb` - (Required): The volume of this disk. For a "ssd" disk, the volume varies from 20Gb-2000Gb, for a "prenium-hdd" disk, the volume varies from 20Gb to 3000Gb.
- `disk_type` - (Required): Can be "ssd" or "prenium-hdd"
- `az` - (Required): The place that this disk will be locate at.
- `client_token` - (Optional): Idempotent check parameter. If you have no idea about this parameter, just leave it blank and a default one will be generated.
- `multi-attachable` - (Optional): Determine if this disk can be attached to several instance at the same time.
- `description` - (Optional): Describe this disk
- `snapshot_id` - (Optional): If you would like to create a disk from an existing snapshot, fill in the id here
- `charge_mode` - (Optional): Candidate payment method lists as following :
  - "prepaid_by_duration" : Pay before using at a planned interval
  - "postpaid_by_usage" : Pay after using, price will be determined according to disk specs and time
  - "postpaid_by_duration":Pay after using, price will be determined according to disk specs and time
- `charge_duration` - (Optional): Used only when charge_mode is prepaid_by_duration, can be "month" ,"year", default : "month"
- `charge_unit` - (Optional): Used only when charge_mode is prepaid_by_duration, specifies how long you would like to buy. When charge_duration is "month", charge_unit varies from 1 to 9, when duration is "year", charge_unit varies from 1 to 3.

Attributes Reference

The following attributes are exported:

- `id` - The id of this disk, can be used to attach/detach from an instance, look like vol-xxxx

Import

Existing disk object can be imported to Terraform state by specifying the disk id:

```
terraform import jdcloud_disk.example vol-abc12345678
```

jdcloud_eip_association

After you have created an EIP, you can attach this EIP with an instance

Example Usage

```
resource "jdcloud_eip_association" "eip-association-example"{
  instance_id = "i-example"
  elastic_ip_id = "fip-example"
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required) : The id of instance
- `elastic_ip_id` - (Required): The id of EIP

jdcloud_eip

Provides a JDCloud elastic IP address

Example Usage

```
resource "jdcloud_eip" "example"{
  eip_provider = "bgp"
  bandwidth_mbps = 1
}
```

Argument Reference

The following arguments are supported:

- `eip_provider` - (Required): Name of your ip service provider, can be bgp or no_bgp, according to the region this instance locates at:
  - cn-north-1 : bgp
  - cn-south-1 : bgp or no_bgp
  - cn-east-1 : bgp or no_bgp
  - cn-east-2 : bgp
- `bandwidth_mbps` - (Required): Specify the bandwidth of your public ip, varify from 1 to 20
- `elastic_ip_address` - (Optional): Specify the IP address you would like to see, or a default IP address will be generated

Attributes Reference

The following attributes are exported:

- `id` : The id of this elastic IP, can be used to attach/detach from private IP address

Import

Existing EIP can be imported to Terraform state by specifying the EIP id:

```
terraform import jdcloud_eip.example fip-abc123456
```

layout: "jdcloud" page_title: "JDCloud Ag Instance" sidebar_current: "docs-jdcloud-ag-instance" description: |-

Creates instance inside an availability group

jdcloud_ag_instance

You can not only creates instance by `jdcloud_instance` , but also through an availability group with a instance template. All instance details required are all specified inside this template. All you need is just to give it a **unique** name.

Example Usage

```
resource "jdcloud_instance_ag_instance" "ag_instance" {
  "availability_group_id" = "ag-example"
  "instances" = [{
    "instance_name" = "xiaohan"
  },{
    "instance_name" = "xiaohan2"
  }]
}
```

Argument Reference

The following arguments are supported:

- `availability_group_id` - (Required): Which availability group you would like to use
- `instances` - (Required): A lists consists of the following element
    - `instance_name` - (Required): The name of this instance. **Must be unique**

Attributes Reference

The following attributes are exported:

- `instance_id` - The id of each instance inside this ag, will be used during `ResourceUpdate` and `ResourceDelete`

jdcloud_instance

Provides a JDCloud ECS instance

Currently instance paid by "prepaid_by_duration" cannot be deleted before they are expired

Example Usage

```
resource "jdcloud_instance" "vm-1" {
  az = "cn-north-1a"
  instance_name = "my-vm-1"
  instance_type = "c.n1.large"
  image_id = "bba85cab-dfdc-4359-9218-7a2de429dd80"
  password = "example_password"
  subnet_id = "example_subnetid"
  network_interface_name = "example_ni_name"
  primary_ip = "172.16.0.13"
  secondary_ips = ["172.16.0.14","172.16.0.15"]
  secondary_ip_count   = 2
  security_group_ids = ["example_SgId"]
  sanity_check = 1

  elastic_ip_bandwidth_mbps = 10
  elastic_ip_provider = "bgp"

  system_disk = {
    disk_category = "local"
    device_name = "vda"
    disk_size_gb =  40
  }

  data_disk = {
    disk_category = "local"
    auto_delete = true
    device_name = "vdb"
  }

  data_disk = {
    disk_category = "cloud"
    device_name = "vdc"
    disk_type = "ssd"
    disk_name = "exampleDisk"
    disk_size_gb = 50
    az = "cn-north-1a"

    auto_delete = true
    disk_name = "vm1-datadisk-1"
    description = "test"
  }
}
```

Argument Reference

The following arguments are supported:

- `az` - (Required) The available zone this ECS instance locates at
- `instance_name` - (Required) Instance name is a string consists of no more than 32 characters, available characters contains:
  - Chinese characters
  - alphanumeric characters
  - "_" and "-" (Underline and hyphen)
- `instance_type` - (Required) Less than 32 characters, available instance type (https://docs.jdcloud.com/cn/virtual-machines/instance-type-family)
- `images_id` - (Required) Image id used to create this ECS instance, can be public image , private image and cloud market place image.
- `subnet_id` - (Required) The id of a VPC subnet. ECS instance created will be in this VPC
- `system_disk` - (Required) The parameter of your system_disk contains:
  - `disk_category` - (Required): can be local or cloud. Especially when the region of this instance is cn-north-1. Only local disk is available. For other regions, both local and cloud are fine.
  - `disk_size_gb` - (Required) : The volume of your disk size, for a local system disk locates at cn-north-1, the volume will be fixed to 40Gb
  - `device_name` - (Required) : Specify the logical attachment point , for example, attachment point can be "vba" "vbc" etc. Just to make sure this point is available with no other device using it.
- `data_disk` - (Optional) : Similar to system disk. You can also create number of data disks together with your ECS instance.
  - `disk_category` - (Required): A string , can be "local" or "cloud".
  - `device_name` - (Required) : Specify the logical attachment point , for example, attachment point can be "vba" "vbc" etc. Just to make sure this point is available with no other device using it.
  - `disk_type` - (Required) : Type of this disk, "ssd" or "prenium-hdd".
  - `disk_size_gb` - (Required) : The volume of your disk size, for "ssd", volume varies from 20Gb to 1000 Gb. For "prenium-hdd" disk, volume varies from 20Gb to 3000Gb
  - `disk_name` - (Required): A string used to name this disk
  - `az` - (Required): The place this disk will be locate at
  - `auto_delete` - (Optional) : Bool value. If this value is set to "true", disk will be deleted when it is detached from instance.
  - `snapshot_id` - (Optional) :Fill in if you would like to create this disk from a snapshot.
  - `description` - (Optional) : Description of this disk
- `description` - (Optional) Description of this ECS instance

- `password` - (Optional) If password of this instance is not set. A default password will be sent to you by email and SMS
- `key_names` - (Optional) Name of the key pair used to login to instance. Look like `${jdcloud_key_pairs.key-1.key_name}`
- `primary_ip` - (Optional) You can specify an public IP address for this instance. If not specified, default public ip address will be generated and assigned.
- `elastic_ip_bandwidth` - (Optional) Specify the bandwidth of your public ip.
- `elastic_ip_provider` - (Optional) Name of your ip service provider, can be bgp or no_bgp, according to the region this instance locates at:
    - cn-north-1 : bgp
    - cn-south-1 : bgp or no_bgp
    - cn-east-1 : bgp or no_bgp
    - cn-east-2 : bgp
- `security_group_id` - (Optional) A list of security group ids to associate with
- `network_interface_name` - (Optional) The id of a network interface, each ECS comes with a elastic network interface. You can leave it as a default name or specify a name you would like to see.
- `sanity_check` - (Optional) : Idempotent check for this network interface, if you have no idea what this parameter is about, just leave it blank.
- `secondary_ips` - (Optional) A list of private ips. These private ips will be associated with the primary network interface on this instance
- `secondary_ip_count` - (Optional) Besides specifying some private ips. By specifying this , a number of private ips will be generated and associated with the network interface.

Attribute Reference

The following attributes are exported:

- `id` - The id of this instance, can be used to attach disk, network interface.
- `disk_id` - Ids of data disk, can be used to detach certain cloud disk.

jdcloud_instance_template

Instances can be built from images and templates, this resources helps you to create an `instance template. Instance templates` can useful when using `Availability-Group`

Example Usage

```
resource "jdcloud_instance_template" "instance_template" {
  "template_name" = "<Name it as you like>"
  "password" = "<Give it a password>"
  "instance_type" = "g.n2.medium"
  "image_id" = "img-example"
  "bandwidth" = 5
  "description" = "GrandJDcloud"
  "ip_service_provider" = "BGP"
  "charge_mode" = "bandwith"
  "subnet_id" = "subnet-exmaple"
  "security_group_ids" = [" sg-example"]
  "system_disk" = {
    "disk_category" = "local"
    "disk_type" = ""
  }
  "data_disks" = {
    disk_category = "cloud"
  }
}
```

Argument Reference

The following arguments are supported

- `template_name` - (Required) : A string, name your instance template
- `password` - (Optional) : String. Once this filed is set. All instance created from this template will use this password
- `key_names` - (Optional) Name of the key pair used to login to instance. You can create a template with password or ssh keys, but not both.
- `instance_type` - (Required) : Specs of this Instance. More available instance type lists Here (https://docs.jdcloud.com/virtual-machines/instance-type-family)
- `image_id` - (Required) : A string, which image you would like to use, usually Ubuntu image or Golang images (https://market.jdcloud.com/#/) are good choices
- `ElasticIP` - (Optional) : If you would like a public IP, fill in here
    - `ip_service_provider` - (Optional): BGP or NonBGP. Principles of them are the same as creating instance, if you are not sure, leave it blank
    - `charge_mode` - (Optional): Candidates are bandwith and flow. By default its `bandwith`
    - `bandwidth` - (Required) : Integer, ranges from 1 to 200

- `subnet_id` - (Required) : This field determines which `vpc` and `subnet` instances will be
- `security_group_ids` - (Required) : Slices consists of strings. It states the security-groups on this instance
- `description` - (Optional) : Describe it, Just like other resources.
- `system_disk` - (Required) : Parameters for system_disk contains
  - `disk_category` - (Required): can be local or cloud. Especially when the region of this instance is cn-north-1. Only local disk is available. For other regions, both local and cloud are fine.
  - `disk_size_gb` - (Required) : The volume of your disk size, for a local system disk locates at cn-north-1, the volume will be fixed to 40Gb
  - `device_name` - (Required) : Specify the logical attachment point , for example, attachment point can be "vba" "vbc" etc. Just to make sure this point is available with no other device using it.
- `data_disks` - (Optional) :
  - `disk_category` - (Required): A string , can be "local" or "cloud".
  - `device_name` - (Required) : Specify the logical attachment point , for example, attachment point can be "vba" "vbc" etc. Just to make sure this point is available with no other device using it.
  - `disk_type` - (Required) : Type of this disk, "ssd" or "prenium-hdd".
  - `disk_size_gb` - (Required) : The volume of your disk size, for "ssd", volume varies from 20Gb to 1000 Gb. For "prenium-hdd" disk, volume varies from 20Gb to 3000Gb
  - `disk_name` - (Required): A string used to name this disk
  - `az` - (Required): The place this disk will be locate at
  - `auto_delete` - (Optional) : Bool value. If this value is set to "true", disk will be deleted when it is detached from instance.
  - `snapshot_id` - (Optional) :Fill in if you would like to create this disk from a snapshot.
  - `description` - (Optional) : Description of this disk

Attribute Reference

The following attributes are exported:

- `id` - The id of this instance template

jdcloud_key_pairs

key_pairs can be useful when trying to login to instance. This function helps to create key_pairs

Example Usage

```
resource "jdcloud_key_pairs" "keypairs_1" {
  key_name   = "JDCLOUD-KEY-PAIR"
  public_key = "ssh-rsa <Your Public Key> rsa-key-jddevelop"
}
```

Argument Reference

The following arguments are supported:

- `key_name` - (Required): Name of your key pairs
- `public_key` - (Optional): There are two ways to upload keys. The trivial one is to upload your public_key.
- `key_file` - (Optional): The other way is to provide your key_file and the `private_key` will be returned

Attributes Reference

The following attributes are exported:

- `key_finger_print` : The finger print of this key pairs
- `private_key` : When you try to generate keys, private key will be returned from server as response, stored locally.

jdcloud_network_acl

This resource create a network acl

Example Usage

```
resource "jdcloud_network_acl" "acl-test" {
  vpc_id = "vpc-exmaple",
  name = "example-name",
  description = "example-descrption-to-this-acl",
}
```

Argument Reference

The following arguments are supported

- `vpc_id` - (Required) : Network acls are supposed to exists under a vpc, fill the id of this vpc here
- `name` - (Required) : Name it
- `description` - (Optional) : A string, give it a description if needed

Attributes Reference

The following attributes are exported:

- `id` - The id of this Network Acl, which can be used to reference this acl.

Import

Existing route table can be imported to Terraform state by specifying the network acl id:

```
terraform import jdcloud_network_acl.example acl-abc123456
```

jdcloud_network_interface_attachment

After you have created network interface, you can attach this network interface to an instance.

Example Usage

```
resource "jdcloud_network_interface_attachment" "attachment-example"{
  instance_id = "i-example"
  network_interface_id = "port-example"
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required) : The id of instance you would like to associate with
- `network_interface_id` - (Required): ****The id of network interface you would like to associate
- `auto_delete` - (Optional): If this field is set to true, network interface will be deleted automatically after detaching from instance

jdcloud_network_interface

Provides a JDCloud network interface

Example Usage

```
resource "jdcloud_network_interface" "example-ineterface" {
  subnet_id = "subnet-example"
  network_interface_name = "example"
}
```

Argument Reference

The following arguments are supported:

- `subnet_id` - (Required) : Subnet is a string which represents the created network interface belong to
- `network_interface_name` - (Required) : The name of this network interface
- `description` - (Optional) : Describe this network interface
- `az` - (Optional) : The place this network interface locates at
- `primary_ip_address` - (Optional) : This is a string of an IP address, specify or leave it blank and a default ip address will be generated and assigned.
- `sanity_check` - (Optional) : Idempotent check for this network interface, if you have no idea what this parameter is about, just leave it blank.
- `secondary_ip_addresses` - (Optional) : This is a list of private ip addresses, by the time you create a network interface, you can also specify some private ip addresses with it.
- `secondary_ip_count` - (Optional) : If you just want some certain amount of private ip addresses, but not care about what they actually are. Fill in the amount of ips, system will generate them for you.
- `security_groups` - (Optional) : A list of security group ids you would like to associate this network interface with.

Attribute Reference

The following attributes are exported:

- `id` : The id of this network interface, can be used to attach/detach from an instance

jdcloud_network_security_group

Provides a JDCloud network security group

Example Usage

```
resource "jdcloud_network_security_group" "sg-example" {
  network_security_group_name = "sg-example"
  vpc_id = "vpc-example"
}
```

Argument Reference

The following arguments are supported:

- `network_security_group_name` - (Required): Name this security group
- `vpc_id` - (Required) : Security group is supposed to exist under a vpc, give the id this vpc
- `description` - (Optional): Describe this security group

Attribute Reference

The following attributes are exported:

- `security_group_id` : A string used to identify this security group, needed when attaching/detaching from a network interface

Import

Existing security group can be imported to Terraform state by specifying the its id.

```
terraform import jdcloud_network_security_group.example sg-abc123456
```

jdcloud_network_security_group_rules

Provides a JDCloud network security group rules. After you have created security group, this resource helps to assign certain rules with it.

Example Usage

The following arguments are supported:

```
resource "jdcloud_network_security_group_rules" "sg-rule-example" {
  security_group_id = "sg-example"
  security_group_rules = [{
    address_prefix = "0.0.0.0/0"
    direction      = "0"
    from_port      = "10"
    protocol       = "6"
    to_port        = "20"
  }]
}
```

Argument Reference

The following arguments are supported:

- `security_group_id` - (Required) : The security group that these rules associate with
- `security_group_rules` - (Required) : A list of rule specs where rule spec are defined as following
  - `address_prefix` - (Required) : Address prefix look like `10.0.0.0/16` ,which represents , IPs with such form will apply this rule. So this is also a filter for the oncoming IPs. Especially, `0.0.0.0/0` means all IPs apply this rule.
  - `direction` - (Required) : Data can come into VPC , or get out of VPC, use "direction" to indicate which direction of data apply this rule, candidate value contains: **0**-represents data coming in, **1**-represents data going out
  - `protocol` - (Required) : Which type of protocol apply this rule, candidate protocol contains, this value can be, candidate value contains: **300**-represents all protocol apply this rule, **6**-TCP apply this rule, **17**-UDP apply this rule, **1**-ICMP apply this rule.
  - `from_port` - (Optional) : Restriction on starting port, if protocol is transport layer protocol , default value for this field is 1. If not a transport layer protocol , this field will be fixed to 0
  - `to_port` - (Optional) : Restriction on ending port, if protocol is transport layer protocol , default value for this field is 65535. If not a transport layer protocol , this field will be fixed to 0
  - `description` - (Optional) : Describe this security group rule

Attribute Reference

The following attributes are exported:

- `rule_id` : Each rule has its own id for attaching/detaching purpose.

jdcloud_oss_bucket

Provides a JDCloud OSS bucket

Example Usage

```
resource "jdcloud_oss_bucket" "oss-example" {
  bucket_name = "oss-example"
}
```

Argument Reference

The following arguments are supported:

- `bucket_name` - (Required) : Name this bucket
- `acl` - (Optional) : This field specify the privilege on this bucket, candidate options are as following:
  - `private` : Owner has full control to this bucket
  - `public-read` : Owner has full control, other people can read from this but no writing is allowed
  - `public-read-write` : Everyone can read/write from this bucket

jdcloud_oss_bucket_upload

After the OSS bucket has been created, you can upload file to this OSS bucket

Example Usage

```
resource "jdcloud_oss_bucket_upload" "example" {
  bucket_name = "bucket_example"
  file_name = "/path/to/file"
}
```

Argument Reference

The following arguments are supported:

- `bucket_name` - (Required): Bucket name you would like to upload file
- `file_name` - (Required) : Location to the file you would like to upload. For example , this field can be `/home/DevOps/HelloWorld.cpp` In order to avoid unable to find your file, absolute path to this file is recommended to use.

jdcloud_rds_account

Set up users for each RDS instance

Example Usage

```
resource "jdcloud_rds_account" "account_example"{
  instance_id = "mysql-example"
  username = "example"
  password = "example2018"
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required): Fill the id of instance you would like to set up account on
- `username` - (Required): Your username. Naming rules: Letters both in upper case and lower case and English underline "_", no more than 16 characters
- `password` - (Required): Password must contain and only supports letters both in upper case and lower case as well as figures, no less than 8 characters and no more than 16 characters

jdcloud_rds_database

After a RDS instance has been created, you probably need to create some databases on this instance

```
resource "jdcloud_rds_database" "db-example"{
  instance_id = "mysql-example"
  db_name = " example"
  character_set = "utf8"
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required) : This field specifies which RDS instance you are using
- `db_name` - (Required) : Name your database. Each database should have a unique name
- `character_set` - (Required) : Candidate character set contains: Utf-8 / SQL_Latin1_General_CP1_CI_AS . etc

jdcloud_rds_instance

Provides a JDCloud RDS instance.

Example Usage

```
resource "jdcloud_rds_instance" "rds_example"{
  instance_name = "example"
  engine = "MySQL"
  engine_version = "5.7"
  instance_class = "db.mysql.s1.micro"
  instance_storage_gb = "20"
  az = "cn-north-1a"
  vpc_id = "vpc-example"
  subnet_id = "subnet-example"
  charge_mode = "postpaid_by_usage"
}
```

Argument Reference

The following arguments are supported:

- `instance_name` - (Required) : Name this RDS instance. Restriction on instance_name lists following
  - Chinese characters and alphanumeric characters
  - "_" and "-" (Underline and hyphen)
  - No less than 2 characters and no more than 32 characters
- `engine` - (Required) : Candidate database engine type lists following
  - MySQL
  - Percona
  - MariaDB
  - SQL-Server
- `engine_version` - (Required) : Select engine version for your cloud database
  - MySQL : 5.6 or 5.7
  - Percona : Only 5.7 available
  - MariaDB : Only 10.2 available
  - SQL-Server : 2008R2 / 2012 / 2014 / 2016
- `instance_class` - (Required) : Each RDS instance is indeed an ECS instance. So you also have to choose its specification, for example, core numbers and memory
- `instance_storage_gb` - (Required) : Storage size of this RDS instance
- `az` - (Required) : The place that this RDS instance locates at

- `vpc_id` - (Required) : Each instance is supposed to exists under a subnet as well as a vpc, fill in the id of the vpc in this field.
- `subnet_id` - (Required) : Each instance is supposed to exists under a subnet as well as a vpc, fill in the id of subnet in this field.
- `charge_mode` - (Required) : Charge mode can be
  - prepaid_by_duration: This means you would like to pay for a planned term before using this instance. Especially, you can not delete a RDS instance of "prepaid_by_duration" type before they expired. Each account can have at most 5 RDS instance
  - postpaid_by_duration: This means that you would like to pay for a unplanned term after using this instance
  - postpaid_by_usage: This means you would like to pay after usage according to the instance spec.
- `charge_unit` - (Optional) : Used only when charge mode is "prepaid_by_duration", can be "month" or "year", by default this value is "month"
- `charge_duration` - (Optional) : Used only when charge_mode is prepaid_by_duration, specifies how long you would like to buy. When charge_duration is "month", charge_unit varies from 1 to 9, when duration is "year", charge_unit varies from 1 to 3.

Attribute Reference

The following attributes are exported:

- `id` : The id of this RDS instance, can be used to reference this instance.

jdcloud_rds_privilege

Assign privileges to each account

Example Usage

```
resource "jdcloud_rds_privilege" "pri-example" {
  instance_id = "mysql-example"
  username = "example"
  account_privilege = [

    {db_name = "db1",privilege = "rw"},
    {db_name = "db2",privilege = "ro"},

  ]
}
```

Argument Reference

The following arguments are supported:

- `instance_id` - (Required) : This field specifies which RDS instance you are using
- `username` - (Required) : The name of account
- `account_privilege` - (Required) : This is a list of rule specs where each rule spec contains:
  - `db_name` - (Required) : Specifies the database
  - `privilege` - (Required) : You can choose "rw" - This account is allowed to both read and write from this database. "ro"-This account is only allowed to read from this accoun

jdcloud_route_table

Provides a JDCloud route table

Example Usage

```
resource "jdcloud_route_table" "example" {
  vpc_id = "vpc-example"
  route_table_name = "example"
}
```

Argument Reference

The following arguments are supported:

- `vpc_id` - (Required) : Route table is supposed to exists under a VPC. Fill the id of this VPC here
- `route_table_name` - (Required) : Name this route table
- `description` - (Optional) : Describe it

Attribute Reference

The following attributes are exported:

- `id` : The id of this route table , used to attach/detach certain route table rules

Import

Existing route table can be imported to Terraform state by specifying the route table id:

```
terraform import jdcloud_route_table.example rtb-abc12345678
```

jdcloud_route_table_rules

After a route table is created, you probably need to add some routing rules to this route table

Example Usage

```
resource "jdcloud_route_table_rules" "rule-example"{
  route_table_id = "rtb-example"
  rule_specs = [

    {
      next_hop_type = "internet"
      next_hop_id   = "internet"
      address_prefix= "10.0.0.0/16"
    },

    {
      next_hop_type = "instance"
      next_hop_id   = "i-example"
      address_prefix= "10.0.2.0/16"
    }]

}
```

Argument Reference

The following arguments are supported:

- `route_table_id` - (Required): Determine which route table you would like to add rules on
- `rule_specs` - (Required): A list of rule specs where each spec is specified as following:
  - `next_hop_type` - (Required): After routing, which location you would like to go, candidates contains: **instance**-route to an instance, **internet**-route to public internet, **vpc_peering**-used only in vpc peering connection, **bgw**-route to boarder gateway.
  - `next_hop_id` - (Required): The id of destination,it can be : **instance**-If you would like to a certain instance, leave its instance_id here, **internet**-If you would like to route to internet , fill "internet" here.
  - `address_prefix` - (Required): look like `10.0.2.0/16` .Address with such prefix will apply this routing rule
  - `priority` - (Optional): Specify of the priority of this rule, value varies from 1 to 255, default value is 100 if you leave it blank

Attribute Reference

The following attributes are exported:

- `id` : id of each rule can be used to attach/detach from this route table

jdcloud_subnet

Provides a JDCloud subnet

Example Usage

```
resource "jdcloud_subnet" "subnet-exmaple"{
    vpc_id = "vpc-example"
    cidr_block = "10.0.128.0/24"
    subnet_name = "example"
}
```

Argument Reference

The following arguments are supported:

- `subnet_name` - (Required) : Name this subnet. Naming rules are : Can't be blank and only supports for Chinese, numbers, capital and lowercase letters, English underline “_” and hyphen “-”. No more than 32 characters
- `vpc_id` - (Required) : Subnets are supposed to exists under a vpc , fill the id of vpc here
- `cidr_block` - (Required) : The address interval this subnet contains. Especially the cidr_block of this subnet can not have overlap with other subnet in this VPC
- `description` - (Optional) : Describe this subnet

Attribute Reference

The following attributes are exported:

- `id` : id of this subnet, used to reference this subnet

Import

Existing route table can be imported to Terraform state by specifying the route table id:

```
terraform import jdcloud_route_table.example rtb-abc12345678
```

jdcloud_vpc

Provides a JDCloud VPC

Example Usage

```
resource "jdcloud_vpc" "vpc-example"{
    vpc_name = "example"
    cidr_block = "172.16.0.0/19"
}
```

Argument reference

The following arguments are supported:

- `vpc_name` - (Required) : The name of this VPC
- `cidr_block` - (Required) : Each VPC contains an IP addresses interval. For example, a VPC with cidr_bock look like `172.16.0.0/19` . IPs of subnet within this VPC are subset of `172.16.0.0/19` .
- `description` - (Optional) : Describe this VPC

Attribute Reference

The following attributes are exported:

- `id` - : The id of this vpc, used to specify this VPC

Import

Existing VPC can be imported to Terraform state by specifying the id of this VPC.

```
terraform import jdcloud_vpc.example vpc-example
```