» nsxt_policy_realization_info

This data source provides information about the realization of a policy resource on NSX manager. This data source will wait until realization is determined as either success or error. It is recommended to use this data source if further configuration depends on resource realization.

» Example Usage

```
data "nsxt_policy_tier1_gateway" "tier1_gw" {
  display_name = "tier1_gw"
}

data "nsxt_policy_realization_info" "info" {
  path = data.nsxt_policy_tier1_gateway.tier1_gw.path
  entity_type = "RealizedLogicalRouter"
}
```

» Argument Reference

- `path` - (Required) The policy path of the resource.
- `entity_type` - (Optional) The entity type of realized resource. If not set, on of the realized resources of the policy resource will be retrieved.

» Attributes Reference

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

- `state` - The realization state of the resource: "REALIZED", "UNKNOWN", "UNREALIZED" or "ERROR".
- `realized_id` - The id of the realized object.

» nsxt_policy_segment_realization

This data source provides information about the realization of a policy segment or policy vlan segment on hypervisor. This data source will wait until realization is complete with either success, partial success or error. It is recommended to use this data source in conjunction with vsphere provider, in order to ensure segment is realized on hypervisor before VM is created on same network.

» Example Usage

```
resource "nsxt_policy_segment" "s1" {
  display_name        = "segment1"
  transport_zone_path = data.nsxt_transport_zone.tz1.path
}

data "nsxt_policy_segment_realization" "info" {
  path = data.nsxt_policy_segment.s1.path
}
```

» Argument Reference

- `path` - (Required) The policy path of the segment.

» Attributes Reference

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

- `state` - The realization state of the resource: `success`, `partial_success`, `orphaned`, `failed` or `error`.

» nsxt_policy_edge_cluster

This data source provides information about policy edge clusters configured in NSX.

» Example Usage

```
data "nsxt_policy_edge_cluster" "ec" {
  display_name = "ec"
}
```

» Argument Reference

- `id` - (Optional) The ID of the edge cluster to retrieve.
- `display_name` - (Optional) The Display Name prefix of the edge cluster to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_edge_node

This data source provides information about policy edge nodes configured in NSX.

» Example Usage

```
data "nsxt_policy_edge_cluster" "ec" {
  display_name = "ec"
}

data "nsxt_policy_edge_node" "node1" {
  edge_cluster_path = data.nsxt_policy_edge_cluster.ec.path
  member_index      = 0
}
```

» Argument Reference

- `edge_cluster_path` - (Required) The path of edge cluster where to which this node belongs.
- `id` - (Optional) The ID of the edge node to retrieve.
- `display_name` - (Optional) The Display Name prefix of the edge node to retrieve.
- `member_index` - (Optional) Member index of the node in edge cluster.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_tier0_gateway

This data source provides information about policy Tier-0 gateways configured in NSX.

» Example Usage

```
data "nsxt_policy_tier0_gateway" "tier0_gw_gateway" {
  display_name = "tier0-gw"
}
```

» Argument Reference

- `id` - (Optional) The ID of Tier-0 gateway to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Tier-0 gateway to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `edge_cluster_path` - The path of the Edge cluster where this Tier-0 gateway is placed.
- `path` - The NSX path of the policy resource.

» nsxt_policy_tier1_gateway

This data source provides information about policy Tier-1s configured in NSX.

» Example Usage

```
data "nsxt_policy_tier1_gateway" "tier1_router" {
  display_name = "tier1_gw"
}
```

» Argument Reference

- id - (Optional) The ID of Tier-1 gateway to retrieve.
- display_name - (Optional) The Display Name prefix of the Tier-1 gateway to retrieve.

» Attributes Reference

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

- description - The description of the resource.
- edge_cluster_path - The path of the Edge cluster where this Tier-1 gateway is placed.
- path - The NSX path of the policy resource.

» nsxt_policy_service

This data source provides information about policy services configured in NSX.

» Example Usage

```
data "nsxt_policy_service" "dns_service" {
  display_name = "DNS"
}
```

» Argument Reference

- id - (Optional) The ID of service to retrieve.
- display_name - (Optional) The Display Name prefix of the service to retrieve.

» Attributes Reference

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

- description - The description of the resource.
- path - The NSX path of the policy resource.

» nsxt_policy_ip_discovery_profile

This data source provides information about policy IpDiscoveryProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_ip_discovery_profile" "test" {
  display_name = "ip-discovery-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of IpDiscoveryProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the IpDiscoveryProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_spoofguard_profile

This data source provides information about policy Spoofguard Profile configured on NSX.

» Example Usage

```
data "nsxt_policy_spoofguard_profile" "test" {
  display_name = "spoofguard-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of SpoofGuardProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the SpoofGuardProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_qos_profile

This data source provides information about policy QosProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_qos_profile" "test" {
  display_name = "qos-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of QosProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the QosProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_ipv6_ndra_profile

This data source provides information about policy Ipv6NdraProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_ipv6_ndra_profile" "test" {
  display_name = "ipv6-ndra-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Ipv6NdraProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Ipv6NdraProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_ipv6_dad_profile

This data source provides information about policy Ipv6DadProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_ipv6_dad_profile" "test" {
  display_name = "ipv6-dad-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Ipv6DadProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Ipv6DadProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_gateway_qos_profile

This data source provides information about policy GatewayQosProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_gateway_qos_profile" "test" {
  display_name = "gateway-qos-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of GatewayQosProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the GatewayQosProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_segment_security_profile

This data source provides information about policy SegmentSecurityProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_segment_security_profile" "test" {
  display_name = "segment-security-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of SegmentSecurityProfile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the SegmentSecurityProfile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_mac_discovery_profile

This data source provides information about policy MacDiscoveryProfile configured in NSX.

» Example Usage

```
data "nsxt_policy_mac_discovery_profile" "test" {
  display_name = "mac-discovery-profile1"
}
```

» Argument Reference

- id - (Optional) The ID of MacDiscoveryProfile to retrieve.
- display_name - (Optional) The Display Name prefix of the MacDiscoveryProfile to retrieve.

» Attributes Reference

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

- description - The description of the resource.
- path - The NSX path of the policy resource.

» nsxt_policy_vm

This data source provides information about Policy based Virtual Machine (VM) configured in NSX and allows look-up of the VM by display_name or the BIOS, external or instance ID exposed on the VM resource.

» Example Usage

```
data "nsxt_policy_vm" "nsxt_vm1" {
  display_name = "nsxt-virtualmachine1"
}
```

» Argument Reference

- display_name - (Optional) The Display Name prefix of the Virtual Machine to retrieve.
- external_id - (Optional) The external ID of the Virtual Machine.
- bios_id - (Optional) The BIOS UUID of the Virtual Machine.
- instance_id - (Optional) The instance UUID of the Virtual Machine.

» Attributes Reference

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

- description - The description of the Virtual Machine.

» nsxt_policy_certificate

This data source provides information about Service Certificate configured in NSX.

» Example Usage

```
data "nsxt_policy_certificate" "test" {
  display_name = "certificate1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Certificate to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Certificate to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_vni_pool

This data source provides information about policy VNI Pools configured in NSX. This data source is supported with NSX 3.0.0 onwards.

» Example Usage

```
data "nsxt_policy_vni_pool" "test" {
  display_name = "vnipool1"
}
```

» Argument Reference

- `id` - (Optional) The ID of VNI Pool Config to retrieve.
- `display_name` - (Optional) The Display Name prefix of the VNI Pool Config to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.
- `start` - The start range of VNI Pool.
- `end` - The end range of VNI Pool.

» nsxt_policy_transport_zone

This data source provides information about Policy based Transport Zones (TZ) configured in NSX. A Transport Zone defines the scope to which a network can extend in NSX. For example an overlay based Transport Zone is associated with both hypervisors and logical switches and defines which hypervisors will be able to serve the defined logical switch. Virtual machines on the hypervisor associated with a Transport Zone can be attached to logical switches in that same Transport Zone.

» Example Usage

```
data "nsxt_policy_transport_zone" "overlay_transport_zone" {
  display_name = "1-transportzone-87"
}
```

```
data "nsxt_policy_transport_zone" "vlan_transport_zone" {
  transport_type = "VLAN_BACKED"
  is_default     = true
}
```

» Argument Reference

- `id` - (Optional) The ID of Transport Zone to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Transport Zone to retrieve.

- `transport_type` - (Optional) Transport type of requested Transport Zone, one of `OVERLAY_STANDARD`, `OVERLAY_ENS`, `VLAN_BACKED` and `UNKNOWN`.
- `is_default` - (Optional) May be set together with `transport_type` in order to retrieve default Transport Zone for for this transport type.

» Attributes Reference

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

- `description` - The description of the Transport Zone.
- `is_default` - A boolean flag indicating if this Transport Zone is the default.
- `transport_type` - The transport type of this transport zone.
- `path` - The NSX path of the policy resource.

» nsxt_policy_ip_block

This data source provides information about policy IP Blocks configured in NSX.

» Example Usage

```
data "nsxt_policy_ip_block" "test" {
  display_name = "ipblock1"
}
```

» Argument Reference

- `id` - (Optional) The ID of IP Block Config to retrieve.
- `display_name` - (Optional) The Display Name prefix of the IP Block Config to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_ip_pool

This data source provides information about policy IP Pools configured in NSX.

» Example Usage

```
data "nsxt_policy_ip_pool" "test" {
  display_name = "ippool1"
}
```

» Argument Reference

- `id` - (Optional) The ID of IP Pool Config to retrieve.
- `display_name` - (Optional) The Display Name prefix of the IP Pool Config to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_lb_app_profile

This data source provides information about policy Load Balancer Application Profile configured in NSX.

» Example Usage

```
data "nsxt_policy_lb_app_profile" "test" {
  type         = "TCP"
  display_name = "my-tcp-profile"
}
```

» Argument Reference

- `id` - (Optional) The ID of Profile to retrieve.
- `type` - (Optional) Type of Profile to retrieve, one of `HTTP`, `TCP`, `UDP`, `ANY`.

- display_name - (Optional) The Display Name prefix of the Profile to retrieve.

» Attributes Reference

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

- description - The description of the resource.
- path - The NSX path of the policy resource.

» nsxt_policy_lb_client_ssl_profile

This data source provides information about policy Client SSL Profile for Load Balancer configured in NSX.

» Example Usage

```
data "nsxt_policy_lb_client_ssl_profile" "test" {
  display_name = "myprofile"
}
```

» Argument Reference

- id - (Optional) The ID of Profile to retrieve.
- display_name - (Optional) The Display Name prefix of the Profile to retrieve.

» Attributes Reference

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

- description - The description of the resource.
- path - The NSX path of the policy resource.

» nsxt_policy_lb_server_ssl_profile

This data source provides information about policy Server SSL Profile for Load Balancer configured in NSX.

» Example Usage

```
data "nsxt_policy_lb_server_ssl_profile" "test" {
  display_name = "myprofile"
}
```

» Argument Reference

- `id` - (Optional) The ID of Profile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Profile to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_lb_monitor

This data source provides information about policy Load Balancer Monitor configured in NSX.

» Example Usage

```
data "nsxt_policy_lb_monitor" "test" {
  type         = "TCP"
  display_name = "my-tcp-monitor"
}
```

» Argument Reference

- `id` - (Optional) The ID of Monitor to retrieve.
- `type` - (Optional) Type of Monitor to retrieve, one of `HTTP`, `HTTPS`, `TCP`, `UDP`, `ICMP`, `PASSIVE`, `ANY`.
- `display_name` - (Optional) The Display Name prefix of Monitor to retrieve.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_lb_persistence_profile

This data source provides information about policy Load Balancer Persistence Profiles configured in NSX.

» Example Usage

```
data "nsxt_policy_lb_persistence_profile" "test" {
  display_name = "policy-lb-persistence-profile1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Load Balanacer Persistence Profile to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Load Balancer Persistence Profile to retrieve.
- `type` - (Optional) The Load Balancer Persistence Profile type. One of `ANY`, `SOURCE_IP`, `COOKIE` or `GENERIC`. Defaults to `ANY`.

» Attributes Reference

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

- `description` - The description of the resource.
- `path` - The NSX path of the policy resource.

» nsxt_policy_tier0_gateway

This resource provides a method for the management of a Tier-0 gateway.

» Example Usage

```
resource "nsxt_policy_tier0_gateway" "tier0_gw" {
  description               = "Tier-0 provisioned by Terraform"
  display_name              = "Tier0-gw1"
  nsx_id                    = "predefined_id"
  failover_mode             = "PREEMPTIVE"
  default_rule_logging      = false
  enable_firewall           = true
  force_whitelisting        = true
  ha_mode                   = "ACTIVE_STANDBY"
  internal_transit_subnets  = ["102.64.0.0/16"]
  transit_subnets           = ["101.64.0.0/16"]
  edge_cluster_path         = data.nsxt_policy_edge_cluster.EC.path

  bgp_config {
      local_as_num    = "60000"
      multipath_relax = false

      route_aggregation {
        prefix = "12.10.10.0/24"
      }

      route_aggregation {
        prefix = "12.11.10.0/24"
      }
  }

  vrf_config {
    gateway_path       = data.nsxt_policy_tier0_gateway.vrf.path
    route_distinuisher = "62000:10"
    evpn_transit_vni   = 76001
    route_target {
      auto_mode      = false
      import_targets = ["62000:2"]
      export_targets = ["62000:3", "10.2.2.0:3"]
    }
  }

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- display_name - (Required) Display name of the resource.
- description - (Optional) Description of the resource.
- tag - (Optional) A list of scope + tag pairs to associate with this Tier-0 gateway.
- nsx_id - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- edge_cluster_path - (Optional) The path of the edge cluster where the Tier-0 is placed. Must be specified when bgp_config is enabled.
- failover_mode - (Optional) This failover mode determines, whether the preferred service router instance for given logical router will preempt the peer. Accepted values are PREEMPTIVE/NON_PREEMPTIVE.
- default_rule_logging - (Optional) Boolean flag indicating if the default rule logging will be enabled or not. The default value is false.
- enable_firewall - (Optional) Boolean flag indicating if the edge firewall will be enabled or not. The default value is true.
- force_whitelisting - (Optional) Boolean flag indicating if white-listing will be forced or not. The default value is false.
- ipv6_ndra_profile_path - (Optional) Policy path to IPv6 NDRA profile.
- ipv6_dad_profile_path - (Optional) Policy path to IPv6 DAD profile.
- ha_mode - (Optional) High-availability Mode for Tier-0. Valid values are ACTIVE_ACTIVE and ACTIVE_STANDBY.
- internal_transit_subnets - (Optional) Internal transit subnets in CIDR format. At most 1 CIDR.
- transit_subnets - (Optional) Transit subnets in CIDR format.
- dhcp_config_path - (Optional) Policy path to DHCP server or relay configuration to use for this gateway.
- bgp_config - (Optional) The BGP configuration for the Tier-0 gateway. When enabled a valid edge_cluster_path must be set on the Tier-0 gateway.
  - tag - (Optional) A list of scope + tag pairs to associate with this Tier-0 gateway's BGP configuration.
  - ecmp - (Optional) A boolean flag to enable/disable ECMP. Default is true.
  - enabled - (Optional) A boolean flag to enable/disable BGP. Default is true.
  - inter_sr_ibgp - (Optional) A boolean flag to enable/disable inter SR IBGP configuration. Default is true.
  - local_as_num - (Optional) BGP AS number in ASPLAIN/ASDOT Format. Default is 65000.
  - multipath_relax - (Optional) A boolean flag to enable/disable multipath relax for BGP. Default is true.
  - graceful_restart_mode - (Optional) Setting to control BGP grace-

  ful restart mode, one of `DISABLE`, `GR_AND_HELPER`, `HELPER_ONLY`.
  - `graceful_restart_timer` - (Optional) BGP graceful restart timer. Default is `180`.
  - `graceful_restart_stale_route_timer` - (Optional) BGP stale route timer. Default is `600`.
  - `route_aggregation`- (Optional) Zero or more route aggregations for BGP.
  - `prefix` - (Required) CIDR of aggregate address.
  - `summary_only` - (Optional) A boolean flag to enable/disable summarized route info. Default is `true`.
- `vrf_config` - (Optional) VRF config for VRF Tier0. This clause is supported with NSX 3.0.0 onwards.
  - `gateway_path` - (Required) Default Tier0 path. Cannot be modified after realization.
  - `evpn_transit_vni` - (Optional) L3 VNI associated with the VRF for overlay traffic. VNI must be unique and belong to configured VNI pool.
  - `route_distinguisher` - (Optional) Route distinguisher. Format: : or :.
  - `route_target` - (Optional) Only one target is supported.
  - `auto_mode` - (Optional) When true, import and export targets should not be specified.
  - `address_family` - (Optional) Address family, currently only `L2VPN_EVPN` is supported, which is the default.
  - `import_targets` - (Optional) List of import route targets. Format: :.
  - `export_targets` - (Optional) List of export route targets. Format: :.

» Attributes Reference

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

- `id` - ID of the Tier-0 gateway.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.
- `bgp_config` - The following attributes are exported for `bgp_config`:
  - `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
  - `path` - The NSX path of the policy resource.

» Importing

An existing policy Tier-0 gateway can be imported into this resource, via the following command:

```
terraform import nsxt_policy_tier0_gateway.tier0_gw ID
```

The above command imports the policy Tier-0 gateway named `tier0_gw` with the NSX Policy ID `ID`.

» nsxt_policy_tier0_gateway

This resource provides a method for the management of a Tier-0 gateway Interface. Note that edge cluster must be configured on Tier-0 Gateway in order to configure interfaces on it.

» Example Usage

```
data "nsxt_policy_tier0_gateway" "gw1" {
  display_name = "gw1"
}

data "nsxt_policy_ipv6_ndra_profile" "slaac" {
  display_name = "slaac"
}

resource "nsxt_policy_vlan_segment" "segment0" {
  display_name = "segment0"
  vlan_ids     = [12]
}

resource "nsxt_policy_tier0_gateway_interface" "if1" {
  display_name           = "segment0_interface"
  description            = "connection to segment0"
  type                   = "SERVICE"
  gateway_path           = data.nsxt_policy_tier0_gateway.gw1.path
  segment_path           = nsxt_policy_vlan_segment.segment0.path
  subnets                = ["12.12.2.13/24"]
  mtu                    = 1500
  ipv6_ndra_profile_path = data.nsxt_policy_ipv6_ndra_profile.slaac.path
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- `type` - (Optional) Type of this interface, one of `SERVICE`, `EXTERNAL`, `LOOPBACK`. Default is `EXTERNAL`
- `gateway_path` - (Required) Policy path for the Tier-0 Gateway.
- `segment_path` - (Optional) Policy path for segment to be connected with this Tier1 Gateway. This argemnt is required for interfaces of type `SERVICE` and `EXTERNAL`.
- `subnets` - (Required) list of Ip Addresses/Prefixes in CIDR format, to be associated with this interface.
- `edge_node_path` - (Optional) Path of edge node for this interface, relevant for interfaces of type `EXTERNAL`.
- `mtu` - (Optional) Maximum Transmission Unit for this interface.
- `ipv6_ndra_profile_path` - (Optional) IPv6 NDRA profile to be associated with this interface.
- `enable_pim` - (Optional) Flag to enable Protocol Independent Multicast, relevant only for interfaces of type `EXTERNAL`. This attribute is supported with NSX 3.0.0 onwards.
- `access_vlan_id`- (Optional) Access VLAN ID, relevant only for VRF interfaces. This attribute is supported with NSX 3.0.0 onwards.
- `urpf_mode` - (Optional) Unicast Reverse Path Forwarding mode, one of `NONE`, `STRICT`. Default is `STRICT`. This attribute is supported with NSX 3.0.0 onwards.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy Tier-0 Gateway Interface can be imported into this resource, via the following command:

```
terraform import nsxt_policy_tier0_gateway_interface.interface1 GW-ID/LOCALE-SERVICE-ID/ID
```

The above command imports the policy Tier-0 gateway interface named `interface1` with the NSX Policy ID `ID` on Tier0 Gateway `GW-ID`, under locale service `LOCALE-SERVICE-ID`.

» nsxt_policy_tier1_gateway

This resource provides a method for the management of a Tier-1 gateway. A Tier-1 gateway is often used for tenants, users and applications. There can be many Tier-1 gateways connected to a common Tier-0 provider gateway.

» Example Usage

```
data "nsxt_policy_tier0_gateway" "T0" {
  display_name = "T0"
}

data "nsxt_policy_edge_cluster" "EC" {
  display_name = "EC"
}

resource "nsxt_policy_tier1_gateway" "tier1_gw" {
  description               = "Tier-1 provisioned by Terraform"
  display_name              = "Tier1-gw1"
  nsx_id                    = "predefined_id"
  edge_cluster_path         = data.nsxt_policy_edge_cluster.EC.path
  failover_mode             = "PREEMPTIVE"
  default_rule_logging      = "false"
  enable_firewall           = "true"
  enable_standby_relocation = "false"
  force_whitelisting        = "true"
  tier0_path                = data.nsxt_policy_tier0_gateway.T0.path
  route_advertisement_types = ["TIER1_STATIC_ROUTES", "TIER1_CONNECTED"]
  pool_allocation           = "ROUTING"

  tag {
    scope = "color"
    tag   = "blue"
  }

  route_advertisement_rule {
    name                      = "rule1"
    action                    = "DENY"
    subnets                   = ["20.0.0.0/24", "21.0.0.0/24"]
    prefix_operator           = "GE"
    route_advertisement_types = ["TIER1_CONNECTED"]
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Tier-1 gateway.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- `edge_cluster_path` - (Optional) The path of the edge cluster where the Tier-1 is placed.
- `failover_mode` - (Optional) This failover mode determines, whether the preferred service router instance for given logical router will preempt the peer. Accepted values are PREEMPTIVE/NON_PREEMPTIVE.
- `default_rule_logging` - (Optional) Boolean flag indicating if the default rule logging will be enabled or not. The default value is false.
- `enable_firewall` - (Optional) Boolean flag indicating if the edge firewall will be enabled or not. The default value is true.
- `enable_standby_relocation` - (Optional) Boolean flag indicating if the standby relocation will be enabled or not. The default value is false.
- `force_whitelisting` - (Optional) Boolean flag indicating if white-listing will be forced or not. The default value is false.
- `tier0_path` - (Optional) The path of the connected Tier0.
- `route_advertisement_types` - (Optional) Enable different types of route advertisements: TIER1_STATIC_ROUTES, TIER1_CONNECTED, TIER1_NAT, TIER1_LB_VIP, TIER1_LB_SNAT, TIER1_DNS_FORWARDER_IP, TIER1_IPSEC_LOCAL_ENDPOINT.
- `ipv6_ndra_profile_path` - (Optional) Policy path to IPv6 NDRA profile.
- `ipv6_dad_profile_path` - (Optional) Policy path to IPv6 DAD profile.
- `dhcp_config_path` - (Optional) Policy path to DHCP server or relay configuration to use for this gateway.
- `pool_allocation` - (Optional) Size of edge node allocation at for routing and load balancer service to meet performance and scalability requirements, one of `ROUTING`, `LB_SMALL`, `LB_MEDIUM`, `LB_LARGE`, `LB_XLARGE`. Default is `ROUTING`. Changing this attribute would force new resource.
- `route_advertisement_rule` - (Optional) List of rules for routes advertisement:
  - `name` - (Required) The name of the rule.

  - `action` - (Required) Action to advertise filtered routes to the connected Tier0 gateway. PERMIT (which is the default): Enables the advertisement, DENY: Disables the advertisement.
  - `subnets` - (Required) list of network CIDRs to be routed.
  - `prefix_operator` - (Optional) Prefix operator to apply on subnets. GE prefix operator (which is the default|) filters all the routes having network subset of any of the networks configured in Advertise rule. EQ prefix operator filter all the routes having network equal to any of the network configured in Advertise rule.The name of the rule.
- `route_advertisement_types` - (Optional) List of desired types of route advertisements, supported values: `TIER1_STATIC_ROUTES`, `TIER1_CONNECTED`, `TIER1_NAT`, `TIER1_LB_VIP`, `TIER1_LB_SNAT`, `TIER1_DNS_FORWARDER_IP`, `TIER1_IPSEC_LOCAL_ENDPOINT`.
- `ingress_qos_profile_path` - (Optional) QoS Profile path for ingress traffic on link connected to Tier0 gateway.
- `egress_qos_profile_path` - (Optional) QoS Profile path for egress traffic on link connected to Tier0 gateway.

» Attributes Reference

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

- `id` - ID of the Tier-1 gateway.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy Tier-1 gateway can be imported into this resource, via the following command:

```
terraform import nsxt_policy_tier1_gateway.tier1_gw ID
```

The above command imports the policy Tier-1 gateway named `tier1_gw` with the NSX Policy ID `ID`.

» nsxt_policy_tier1_gateway

This resource provides a method for the management of a Tier-1 gateway Interface. Note that edge cluster must be configured on Tier-1 Gateway in order to configure interfaces on it.

» Example Usage

```
data "nsxt_policy_tier1_gateway" "gw1" {
  display_name = "gw1"
}

data "nsxt_policy_ipv6_ndra_profile" "slaac" {
  display_name = "slaac"
}

resource "nsxt_policy_vlan_segment" "segment1" {
  display_name = "segment1"
  vlan_ids     = [12]
}

resource "nsxt_policy_tier1_gateway_interface" "if1" {
  display_name           = "segment1_interface"
  description            = "connection to segment1"
  gateway_path           = data.nsxt_policy_tier1_gateway.gw1.path
  segment_path           = nsxt_policy_vlan_segment.segment1.path
  subnets                = ["12.12.2.13/24"]
  mtu                    = 1500
  ipv6_ndra_profile_path = data.nsxt_policy_ipv6_ndra_profile.slaac.path
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- `gateway_path` - (Required) Policy path for the Tier-1 Gateway.
- `segment_path` - (Required) Policy path for segment to be connected with this Tier1 Gateway.
- `subnets` - (Required) list of Ip Addresses/Prefixes in CIDR format, to be associated with this interface.
- `mtu` - (Optional) Maximum Transmission Unit for this interface.
- `ipv6_ndra_profile_path` - (Optional) IPv6 NDRA profile to be associated with this interface.
- `urpf_mode` - (Optional) Unicast Reverse Path Forwarding mode, one of `NONE`, `STRICT`. Default is `STRICT`. This attribute is supported with NSX 3.0.0 onwards.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy Tier-1 Gateway Interface can be imported into this resource, via the following command:

```
terraform import nsxt_policy_tier1_gateway_interface.interface1 GW-ID/LOCALE-SERVICE-ID/ID
```

The above command imports the policy Tier-1 gateway interface named `interface1` with the NSX Policy ID `ID` on Tier1 Gateway `GW-ID`, under locale service `LOCALE-SERVICE-ID`.

» nsxt_policy_service

This resource provides a way to configure a networking and security service which can be used within NSX Policy.

» Example Usage

```
resource "nsxt_policy_service" "service_icmp" {
  description  = "ICMP service provisioned by Terraform"
  display_name = "S1"

  icmp_entry {
    display_name = "ICMP-entry"
    description  = "ICMP v4 entry"
    protocol     = "ICMPv4"
    icmp_code    = "1"
    icmp_type    = "3"
  }

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

```
resource "nsxt_policy_service" "service_l4port" {
  description  = "L4 ports service provisioned by Terraform"
  display_name = "S1"

  l4_port_set_entry {
    display_name      = "TCP80"
    description       = "TCP port 80 entry"
    protocol          = "TCP"
    destination_ports = [ "80" ]
  }

  tag {
    scope = "color"
    tag   = "pink"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource. The service must contain at least 1 entry (of at least one of the types), and possibly more.
- `icmp_entry` - (Optional) Set of ICMP type service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
  - `protocol` - (Required) Version of ICMP protocol ICMPv4 or ICMPv6.
  - `icmp_code` - (Optional) ICMP message code.
  - `icmp_type` - (Optional) ICMP message type.
- `l4_port_set_entry` - (Optional) Set of L4 ports set service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
  - `protocol` - (Required) L4 protocol. Accepted values - 'TCP' or 'UDP'.
  - `destination_ports` - (Optional) Set of destination ports.
  - `source_ports` - (Optional) Set of source ports.

- `igmp_entry` - (Optional) Set of IGMP type service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
- `ether_type_entry` - (Optional) Set of Ether type service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
  - `ether_type` - (Required) Type of the encapsulated protocol.
- `ip_protocol_entry` - (Optional) Set of IP Protocol type service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
  - `protocol` - (Required) IP protocol number.
- `algorithm_entry` - (Optional) Set of Algorithm type service entries. Each with the following attributes:
  - `display_name` - (Optional) Display name of the service entry.
  - `description` - (Optional) Description of the service entry.
  - `destination_port` - (Required) a single destination port.
  - `source_ports` - (Optional) Set of source ports/ranges.
  - `algorithm` - (Required) Algorithm one of ”ORACLE_TNS”, ”FTP”, ”SUN_RPC_TCP”, ”SUN_RPC_UDP”, ”MS_RPC_TCP”, ”MS_RPC_UDP”, ”NBNS_BROADCAST”, ”NBDG_BROADCAST”, ”TFTP”

» Attributes Reference

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

- `id` - ID of the service.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing service can be imported into this resource, via the following command:

```
terraform import nsxt_policy_service.service_icmp ID
```

The above service imports the service named `service_icmp` with the NSX ID `ID`.

» nsxt_policy_group

This resource provides a method for the management of an inventory Group and its members. Groups as often used as sources and destinations, as well as in the Applied To field, in firewall rules.

» Example Usage

```
resource "nsxt_policy_group" "group1" {
    display_name = "tf-group1"
    description  = "Terraform provisioned Group"

    criteria {
        condition {
            key         = "Name"
            member_type = "VirtualMachine"
            operator    = "STARTSWITH"
            value       = "public"
        }
        condition {
            key         = "OSName"
            member_type = "VirtualMachine"
            operator    = "CONTAINS"
            value       = "Ubuntu"
        }
    }

    conjunction {
        operator = "OR"
    }

    criteria {
        ipaddress_expression {
            ip_addresses = ["211.1.1.1", "212.1.1.1", "192.168.1.1-192.168.1.100"]
        }
    }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.

- domain - (Optional) The domain to use for the Group. This domain must already exist.
- tag - (Optional) A list of scope + tag pairs to associate with this Group.
- nsx_id - (Optional) The NSX ID of this resource. If set, this ID will be used to create the group resource.
- criteria - (Optional) A repeatable block to specify criteria for members of this Group. If more than 1 criteria block is specified, it must be separated by a conjunction. In a criteria block the following membership selection expressions can be used:
  - ipaddress_expression - (Optional) An expression block to specify individual IP Addresses, ranges of IP Addresses or subnets for this Group.
  - ip_addresses - (Required for a ipaddress_expression) This list can consist of a single IP address, IP address range or a subnet. Its type can be of either IPv4 or IPv6. Both IPv4 and IPv6 addresses within one expression is not allowed.
  - path_expression - (Optional) An expression block to specify direct group members by policy path.
  - member_paths - (Required for a path_expression) List of policy paths for direct members for this Group (such as Segments, Segment ports, Groups etc).
  - condition (Optional) A repeatable condition block to select this Group's members. When multiple condition blocks are used in a single criteria they form a nested expression that's implicitly ANDed together and each nested condition must used the same member_type.
  - key (Required for a condition) Specifies the attribute to query. Must be one of: Tag, ComputerName, OSName or Name. For a member_type other than VirtualMachine, only the Tag key is supported.
  - member_type (Required for a condition) Specifies the type of resource to query. Must be one of: IPSet, LogicalPort, LogicalSwitch, Segment, SegmentPort or VirtualMachine.
  - operator (Required for a condition) Specifies the query operator to use. Must be one of: CONTAINS, ENDSWITH, EQUALS, NOTEQUALS or STARTSWITH.
  - value (Required for a condition) User specified string value to use in the query. For Tag criteria, use 'scope|value' notation if you wish to specify scope in criteria.
- conjunction (Required for multiple criteria) When specifying multiple criteria, a conjunction is used to specify if the criteria should selected using AND or OR.
  - operator (Required for conjunction) The operator to use. Must be one of AND or OR. If AND is used, then the criteria block before/after must be of the same type and if using condition then also must use the same member_type.

» Attributes Reference

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

- `id` - ID of the Group.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy Group can be imported into this resource, via the following command:

```
terraform import nsxt_policy_group.group1 ID
```

The above command imports the policy Group named `group` with the NSX Policy ID `ID`.

If the Group to import isn't in the `default` domain, the domain name can be added to the `ID` before a slash.

For example to import a Group with `ID` in the `MyDomain` domain:

```
terraform import nsxt_policy_group.group1 MyDomain/ID
```

» nsxt_policy_security_policy

This resource provides a method for the management of Security Policy and rules under it.

» Example Usage

```
resource "nsxt_policy_security_policy" "policy1" {
    display_name = "policy1"
    description  = "Terraform provisioned Security Policy"
    category     = "Application"
    locked       = false
    stateful     = true
    tcp_strict   = false
    scope        = [nsxt_policy_group.pets.path]

    rule {
      display_name       = "block_icmp"
      destination_groups = [nsxt_policy_group.cats.path, nsxt_policy_group.dogs.path]
      action             = "DROP"
      services           = [nsxt_policy_service.icmp.path]
      logged             = true
    }

    rule {
      display_name     = "allow_udp"
      source_groups    = [nsxt_policy_group.fish.path]
      sources_excluded = true
      scope            = [nsxt_policy_group.aquarium.path]
      action           = "ALLOW"
      services         = [nsxt_policy_service.udp.path]
      logged           = true
      disabled         = true
      notes            = "Disabled by starfish for debugging"
    }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `domain` - (Optional) The domain to use for the resource. This domain must already exist.
- `tag` - (Optional) A list of scope + tag pairs to associate with this policy.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `category` - (Required) Category of this policy, one of `Ethernet`, `Emergency`, `Infrastructure`, `Environment`, `Application`.
- `comments` - (Optional) Comments for security policy lock/unlock.
- `locked` - (Optional) Indicates whether a security policy should be locked. If locked by a user, no other user would be able to modify this policy.
- `scope` - (Optional) The list of policy object paths where the rules in this policy will get applied.
- `sequence_number` - (Optional) This field is used to resolve conflicts between security policies across domains.
- `stateful` - (Optional) If true, state of the network connects are tracked and a stateful packet inspection is performed. Default is true.
- `tcp_strict` - (Optional) Ensures that a 3 way TCP handshake is done before the data packets are sent. Default is false.
- `rule` - (Optional) A repeatable block to specify rules for the Security Policy. Each rule includes the following fields:
  - `display_name` - (Required) Display name of the resource.

  - `description` - (Optional) Description of the resource.
  - `action` - (Optional) Rule action, one of `ALLOW`, `DROP`, `REJECT`. Default is `ALLOW`.
  - `destination_groups` - (Optional) Set of group paths that serve as destination for this rule.
  - `source_groups` - (Optional) Set of group paths that serve as source for this rule.
  - `destinations_excluded` - (Optional) Negation of destination groups.
  - `sources_excluded` - (Optional) Negation of source groups.
  - `direction` - (Optional) Traffic direction, one of `IN`, `OUT` or `IN_OUT`. Default is `IN_OUT`.
  - `disabled` - (Optional) Flag to disable this rule. Default is false.
  - `ip_version` - (Optional) Version of IP protocol, one of `IPV4`, `IPV6`, `IPV4_IPV6`. Default is `IPV4_IPV6`.
  - `logged` - (Optional) Flag to enable packet logging. Default is false.
  - `notes` - (Optional) Additional notes on changes.
  - `profiles` - (Optional) Set of profile paths relevant for this rule.
  - `scope` - (Optional) Set of policy object paths where the rule is applied.
  - `services` - (Optional) Set of service paths to match.
  - `tag` - (Optional) A list of scope + tag pairs to associate with this rule.

» Attributes Reference

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

- `id` - ID of the Security Policy.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.
- `rule`:
  - `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
  - `path` - The NSX path of the policy resource.
  - `sequence_number` - Sequence number of the this rule, is defined by order of rules in the list.
  - `rule_id` - Unique positive number that is assigned by the system and is useful for debugging.

» Importing

An existing security policy can be imported into this resource, via the following command:

```
terraform import nsxt_policy_security_policy.policy1 domain/ID
```

The above command imports the security policy named `policy1` under NSX domain `domain` with the NSX Policy ID `ID`.

» nsxt_policy_gateway_policy

This resource provides a method for the management of a Gateway Policy and its Rules.

» Example Usage

```
resource "nsxt_policy_gateway_policy" "test" {
  display_name    = "tf-gw-policy"
  description     = "Terraform provisioned Gateway Policy"
  category        = "LocalGatewayRules"
  locked          = false
  sequence_number = 3
  stateful        = true
  tcp_strict      = false

  tag {
    scope = "color"
    tag   = "orange"
  }

  rule {
    display_name          = "rule1"
    destination_groups    = [nsxt_policy_group.group1.path, nsxt_policy_group.group2.path]
    disabled              = true
    action                = "DROP"
    logged                = true
    scope                 = [nsxt_policy_tier1_gateway.policygateway.path]
  }

}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `category` - (Required) The category to use for priority of this Gateway Policy. Must be one of: `Emergency`, `SystemRules`, `SharedPreRules`, `LocalGatewayRules`, `AutoServiceRules` and `Default`.
- `description` - (Optional) Description of the resource.
- `domain` - (Optional) The domain to use for the Gateway Policy. This domain must already exist.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Gateway Policy.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the Gateway Policy resource.
- `comments` - (Optional) Comments for this Gateway Policy including lock/unlock comments.
- `locked` - (Optional) A boolean value indicating if the policy is locked. If locked, no other users can update the resource.
- `sequence_number` - (Optional) An int value used to resolve conflicts between security policies across domains
- `stateful` - (Optional) A boolean value to indicate if this Policy is stateful. When it is stateful, the state of the network connects are tracked and a stateful packet inspection is performed.
- `tcp_strict` - (Optional) A boolean value to enable/disable a 3 way TCP handshake is done before the data packets are sent.
- `rule` (Optional) A repeatable block to specify rules for the Gateway Policy. Each rule includes the following fields:
  - `display_name` - (Required) Display name of the resource.
  - `description` - (Optional) Description of the resource.
  - `destination_groups` - (Optional) A list of destination group paths to use for the policy.
  - `destinations_excluded` - (Optional) A boolean value indicating negation of destination groups.
  - `direction` - (Optional) The traffic direction for the policy. Must be one of: `IN`, `OUT` or `IN_OUT`. Defaults to `IN_OUT`.
  - `disabled` - (Optional) A boolean value to indicate the rule is disabled. Defaults to `false`.
  - `ip_version` - (Optional) The IP Protocol for the rule. Must be one of: `IPV4`, `IPV6` or `IPV4_IPV6`. Defaults to `IPV4_IPV6`.
  - `logged` - (Optional) A boolean flag to enable packet logging.
  - `notes` - (Optional) Text for additional notes on changes for the rule.
  - `profiles` - (Optional) A list of profiles for the rule.
  - `scope` - (Required) List of policy paths where the rule is applied.
  - `services` - (Optional) List of services to match.
  - `source_groups` - (Optional) A list of source group paths to use for

    the policy.
  - `source_excluded` - (Optional) A boolean value indicating negation of source groups.
  - `tag` - (Optional) A list of scope + tag pairs to associate with this Rule.
  - `action` - (Optional) The action for the Rule. Must be one of: `ALLOW`, `DROP` or `REJECT`. Defaults to `ALLOW`.

» Attributes Reference

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

- `id` - ID of the Secuirty Policy.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.
- `rule`:
  - `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
  - `path` - The NSX path of the policy resource.
  - `sequence_number` - Sequence number of the this rule, is defined by order of rules in the list.
  - `rule_id` - Unique positive number that is assigned by the system and is useful for debugging.

» Importing

An existing Gateway Policy can be imported into this resource, via the following command:

```
terraform import nsxt_policy_gateway_policy.gwpolicy1 ID
```

The above command imports the policy Gateway Policy named `gwpolicy1` with the NSX Policy id `ID`.

If the Policy to import isn't in the `default` domain, the domain name can be added to the `ID` before a slash.

For example to import a Group with `ID` in the `MyDomain` domain:

```
terraform import nsxt_policy_gateway_policy.gwpolicy1 MyDomain/ID
```

» nsxt_policy_static_route

This resource provides a method for the management of a Static Route.

» Example Usage

```
resource "nsxt_policy_static_route" "route1" {
  display_name = "sroute"
  gateway_path = nsxt_policy_tier0_gateway.tier0_gw.path
  network      = "13.1.1.0/24"

  next_hop {
    admin_distance = "2"
    ip_address     = "11.10.10.1"
  }

  next_hop {
    admin_distance = "4"
    ip_address = "12.10.10.1"
  }

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Tier-0 gateway.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- `network` - (Required) The network address in CIDR format for the route.
- `gateway_path` (Required) The NSX Policy path to the Tier0 or Tier1 Gateway for this Static Route.
- `next_hop` - (Required) One or more next hops for the static route.
  - `admin_distance` - (Optional) The cost associated with the next hop. Valid values are 1 - 255 and the default is 1.
  - `ip_address` - (Required) The gateway address of the next hop.
  - `interface` - (Optional) The policy path to the interface associated with the static route.

» Attributes Reference

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

- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy Static Route can be imported into this resource, via the following command:

```
terraform import nsxt_policy_static_route.route1 GWID/ID
```

The above command imports the policy Static Route named `route1` for the NSX Tier0 or Tier1 Gateway `GWID` with the NSX Policy ID `ID`.

» nsxt_policy_nat_rule

This resource provides a method for the management of a NAT Rule.

» Example Usage

```
resource "nsxt_policy_nat_rule" "dnat1" {
  display_name         = "dnat_rule1"
  action               = "DNAT"
  source_networks      = ["9.1.1.1", "9.2.1.1"]
  destination_networks = ["11.1.1.1"]
  translated_networks  = ["10.1.1.1"]
  gateway_path         = nsxt_policy_tier1_gateway.t1gateway.path
  logging              = false
  firewall_match       = "MATCH_INTERNAL_ADDRESS"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this NAT Rule.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the policy resource.
- `gateway_path` - (Required) The NSX Policy path to the Tier0 or Tier1 Gateway for this NAT Rule.
- `action` - (Required) The action for the NAT Rule. One of `SNAT`, `DNAT`, `REFLEXIVE`, `NO_SNAT`, `NO_DNAT`, `NAT64`.
- `destination_networks` - (Optional) A list of destination network IP addresses or CIDR.
- `enabled` - (Optional) Enable/disable the Rule. Defaults to `true`.
- `firewall_match` - (Optional) Firewall match flag. One of `MATCH_EXTERNAL_ADDRESS`, `MATCH_INTERNAL_ADDRESS`, `BYPASS`.
- `logging` - (Optional) Enable/disable rule logging. Defaults to `false`.
- `rule_priority` - (Optional) The priority of the rule. Valid values between 0 to 2147483647. Defaults to `100`.
- `service` - (Optional) Policy path of Service on which the NAT rule will be applied.
- `source_networks` - (Optional) A list of source network IP addresses or CIDR.
- `translated_networks` - (Optional) A list of translated network IP addresses or CIDR.
- `translated_ports` - (Optional) Port number or port range. For use with `DNAT` action only.
- `scope` - (Optional) A list of paths to interfaces and/or labels where the NAT Rule is enforced.

» Attributes Reference

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

- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing policy NAT Rule can be imported into this resource, via the following command:

```
terraform import nsxt_policy_nat_rule.rule1 GWID/ID
```

The above command imports the policy NAT Rule named `rule1` for the NSX Tier0 or Tier1 Gateway `GWID` with the NSX Policy ID `ID`.

» nsxt_policy_ip_block

This resource provides a means to configure IP Blocks in NSX Policy.

» Example Usage

```
resource "nsxt_policy_ip_block" "block1" {
  display_name = "ip-block1"
  cidr         = "192.168.1.0/24"

  tag {
    scope = "color"
    tag   = "blue"
  }

  tag {
    scope = "env"
    tag   = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) The display name for the IP Block.
- `description` - (Optional) Description of the resource.
- `cidr` - (Required) Network address and the prefix length which will be associated with a layer-2 broadcast domain.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP Block.

» Attributes Reference

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

- `id` - ID of the IP Block.

- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the resource.

» Importing

An existing IP Block can be imported into this resource, via the following command:

```
terraform import nsxt_policy_ip_block.block1 ID
```

The above would import NSX IP Block as a resource named `block1` with the NSX id `ID`, where `ID` is NSX ID of the IP Block.

» nsxt_policy_ip_pool

This resource provides a means to configure IP Pools in NSX Policy.

» Example Usage

```
resource "nsxt_policy_ip_pool" "pool1" {
  display_name = "ip-pool1"

  tag {
    scope = "color"
    tag   = "blue"
  }

  tag {
    scope = "env"
    tag   = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) The display name for the IP Pool.
- `description` - (Optional) Description of the resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP Pool.

» Attributes Reference

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

- `id` - ID of the IP Pool.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the resource.

» Importing

An existing IP Pool can be imported into this resource, via the following command:

```
terraform import nsxt_policy_ip_pool.pool1 ID
```

The above would import NSX IP Pool as a resource named `pool1` with the NSX ID `ID`, where `ID` is NSX ID of the IP Pool.

» nsxt_policy_ip_pool_block_subnet

This resource provides a means to configure IP Pool Block Subnets in NSX Policy.

» Example Usage

```
resource "nsxt_policy_ip_pool_block_subnet" "block_subnet1" {
  display_name        = "block-subnet1"
  pool_path           = nsxt_policy_ip_pool.pool1.path
  block_path          = nsxt_policy_ip_block.block1.path
  size                = 8
  auto_assign_gateway = false

  tag {
    scope = "color"
    tag   = "blue"
  }

  tag {
    scope = "env"
    tag   = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) The display name for the Block Subnet.
- `pool_path` - (Required) The Policy path to the IP Pool for this Block Subnet.
- `block_path` - (Required) The Policy path to the IP Block for this Block Subnet.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `size` - (Required) The size of this Block Subnet. Must be a power of 2
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Block Subnet.
- `auto_assign_gateway - (Optional)` A boolean flag to toggle auto-assignment of the Gateway IP for this Subnet

» Attributes Reference

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

- `id` - ID of this Block Subnet.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the resource.

» Importing

An existing Block can be imported into this resource, via the following command:

```
terraform import nsxt_policy_ip_pool_block_subnet.block_subnet1 pool-id/subnet-id
```

The above would import NSX Block Subnet as a resource named `block_subnet1` with the NSX ID `subnet-id` in the IP Pool `pool-id`, where `subnet-id` is NSX ID of Block Subnet and `pool-id` is the IP Pool ID the Subnet is in.

» nsxt_policy_ip_pool_static_subnet

This resource provides a means to configure IP Pool Static Subnets in NSX Policy.

» Example Usage

```
resource "nsxt_policy_ip_pool_static_subnet" "static_subnet1" {
  display_name        = "static-subnet1"
  pool_path           = nsxt_policy_ip_pool.pool1.path
  cidr                = "12.12.12.0/24"
  gateway             = "12.12.12.1"

  allocation_range {
    start = "12.12.12.10"
    end   = "12.12.12.20"
  }
  allocation_range {
      start = "12.12.12.100"
      end   = "12.12.12.120"
    }

  tag {
    scope = "color"
    tag   = "blue"
  }

  tag {
    scope = "env"
    tag   = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) The display name for the Static Subnet.
- `pool_path` - (Required) The Policy path to the IP Pool for this Static Subnet.
- `cidr` - (Required) The network CIDR
- `allocation_range` - (Required) One or more IP allocation ranges for the Subnet.
  - `start` - (Required) The start IP address for the allocation range.
  - `end` - (Required) The end IP address for the allocation range.
- `description` - (Optional) Description of the resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Static Subnet.

- `dns_nameservers` - (Optional) A list of up to 3 DNS nameservers for the Subnet.
- `dns_suffix` - (Optional) The DNS suffix for the Subnet.
- `gateway` - (Optional) The gateway IP for the Subnet.

» Attributes Reference

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

- `id` - ID of this Static Subnet.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the resource.

» Importing

An existing Static can be imported into this resource, via the following command:

```
terraform import nsxt_policy_ip_pool_static_subnet.static_subnet1 pool-id/subnet-id
```

The above would import NSX Static Subnet as a resource named `static_subnet1` with the NSX ID `subnet-id` in the IP Pool `pool-id`, where `subnet-id` is ID of Static Subnet and `pool-id` is the IP Pool ID the Subnet is in.

» nsxt_policy_ip_address_allocation

This resource provides a method for the management of a IP Address Allocations. Note that IP Address Allocations cannot be updated once created and changing any attributes of an existing allocation will re-create it.

» Example Usage

```
resource "nsxt_policy_ip_address_allocation" "test" {
  display_name  = "test"
  description   = "Terraform provisioned IpAddressAllocation"
  pool_path     = nsxt_policy_ip_pool.pool1.path
  allocation_ip = "12.12.12.12"
}
```

» Argument Reference

The following arguments are supported:

- display_name - (Required) Display name of the resource.
- description - (Optional) Description of the resource.
- tag - (Optional) A list of scope + tag pairs to associate with this resource.
- nsx_id - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- allocation_ip - (Optional) The IP Address to allocate. If unspecified any free IP in the pool will be allocated.
- pool_path - (Required) The policy path to the IP Pool for this Allocation.

» Attributes Reference

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

- id - ID of the Allocation.
- revision - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- path - The NSX path of the policy resource.
- allocation_ip - If the allocation_ip is not specified in the resource, any free IP is allocated and its value is exported on this attribute.

» Importing

An existing IP Allocation can be imported into this resource, via the following command:

```
terraform import nsxt_policy_ip_address_allocation.test POOL-ID/ID
```

The above command imports IpAddressAllocation named test with the NSX IpAddressAllocation ID ID in IP Pool POOL-ID.

» nsxt_policy_bgp_neighbor

This resource provides a method for the management of a BGP Neighbor.

» Example Usage

```
resource "nsxt_policy_bgp_neighbor" "test" {
  display_name          = "tfbpg"
  description           = "Terraform provisioned BgpNeighborConfig"
  bgp_path              = nsxt_policy_tier0_gateway.testresource.bgp_config.0.path
  allow_as_in           = true
  graceful_restart_mode = "HELPER_ONLY"
  hold_down_time        = 300
  keep_alive_time       = 200
  neighbor_address      = "12.12.11.23"
  password              = "passw0rd"
  remote_as_num         = "60000"
  source_addresses      = ["12.3.40.251"]
  depends_on            = [nsxt_policy_tier0_gateway_interface.my_gateway_interface]

  bfd_config {
    enabled  = true
    interval = 1000
    multiple = 4
  }

  route_filtering {
    address_family = "IPV4"
    maximum_routes = 20
  }
}
```

NOTE: If bgp neighbor configuration depends on gateway interface, please add `depends_on` clause in `nsxt_policy_bgp_neighbor` resource in order to ensure correct order of creation/deletion.

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `bgp_path` - (Required) The policy path to the BGP configuration for this neighbor.
- `allow_as_in` - (Optional) Flag to enable allowas_in option for BGP neighbor. Defaults to `false`.
- `graceful_restart_mode` - (Optional) BGP Graceful Restart Configuration Mode. One of `DISABLE`, `GR_AND_HELPER` or `HELPER_ONLY`.
- `hold_down_time` - (Optional) Wait time in seconds before declaring peer dead. Defaults to `180`.
- `keep_alive_time` - (Optional) Interval between keep alive messages sent to peer. Defaults to `60`.

- `maximum_hop_limit` - (Optional) Maximum number of hops allowed to reach BGP neighbor. Defaults to `1`.
- `neighbor_address` - (Required) Neighbor IP Address.
- `password` - (Optional) Password for BGP neighbor authentication. Set to the empty string to clear out the password.
- `remote_as_num` - (Required) 4 Byte ASN of the neighbor in ASPLAIN Format.
- `source_addresses` - (Optional) A list of up to 8 source IP Addresses for BGP peering.
- `bfd_config` - (Optional) The BFD configuration.
  - `enabled` - (Optional) A boolean flag to enable/disable BFD. Defaults to `false`.
  - `interval` - (Optional) Time interval between heartbeat packets in milliseconds. Defaults to `500`.
  - `multiple` - (Optional) Number of times heartbeat packet is missed before BFD declares the neighbor is down. Defaults to `3`.
- `route_filtering` - (Optional) Up to 2 route filters for the neighbor. Note that prior to NSX version 3.0.0, only 1 element is supported.
  - `address_family` - (Required) Address family type. Must be one of `EVPN`, `IPV4` or `IPV6`. Note the `EVPN` property is only available starting with NSX version 3.0.0.
  - `enabled`- (Optional) A boolean flag to enable/disable address family. Defaults to `false`.
  - `in_route_filter`- (Optional) Path of prefix-list or route map to filter routes for IN direction.
  - `out_route_filter`- (Optional) Path of prefix-list or route map to filter routes for OUT direction.
  - `maximum_routes` - (Optional) Maximum number of routes for the address family. Note this property is only available starting with NSX version 3.0.0.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing BGP Neighbor can be imported into this resource, via the following command:

```
terraform import nsxt_policy_bgp_neighbor.test T0_ID/LOCALE_SERVICE_ID/NEIGHBOR_ID
```

The above command imports BGP Neighbor named `test` with the NSX BGP Neighbor ID `NEIGHBOR_ID` from the Tier-0 `T0_ID` and Locale Service `LOCALE_SERVICE_ID`.

» nsxt_policy_dhcp_relay

This resource provides a method for the management of a Dhcp Relay.

» Example Usage

```
resource "nsxt_policy_dhcp_relay" "test" {
    display_name      = "test"
    description       = "Terraform provisioned Dhcp Relay"
    server_addresses = ["10.0.0.2", "7001::2"]
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `server_addresses` - (Required) List of DHCP server addresses.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing object can be imported into this resource, via the following command:

```
terraform import nsxt_policy_dhcp_relay.test ID
```

The above command imports Dhcp Relay named `test` with the NSX Dhcp Relay ID `ID`.

» nsxt_policy_segment

This resource provides a method for the management of Segments.

» Example Usage

```
resource "nsxt_policy_segment" "segment1" {
    display_name      = "segment1"
    description       = "Terraform provisioned Segment"
    connectivity_path = nsxt_policy_tier1_gateway.mygateway.path

    subnet {
      cidr        = "12.12.2.1/24"
      dhcp_ranges = ["12.12.2.100-12.12.2.160"]

      dhcp_v4_config {
        server_address = "12.12.2.2/24"
        lease_time     = 36000

        dhcp_option_121 {
          network  = "6.6.6.0/24"
          next_hop = "1.1.1.21"
        }

        dhcp_generic_option {
          code = "119"
          values = ["abc"]
        }
      }
    }

    advanced_config {
      connectivity = "OFF"
      local_egress = true
    }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this policy.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `connectivity_path` - (Optional) Policy path to the connecting Tier-0 or Tier-1.
- `domain_name`- (Optional) DNS domain names.
- `overlay_id` - (Optional) Overlay connectivity ID for this Segment.
- `transport_zone_path` - (Required) Policy path to the Overlay transport zone. This property is required if more than one overlay transport zone is defined, and none is marked as default.
- `dhcp_config_path` - (Optional) Policy path to DHCP server or relay configuration to use for subnets configured on this segment. This attribute is supported with NSX 3.0.0 onwards.
- `subnet` - (Required) Subnet configuration block.
  - `cidr` - (Required) Gateway IP address CIDR. This argument can not be changed if DHCP is enabled for the subnet.
  - `dhcp_ranges` - (Optional) List of DHCP address ranges for dynamic IP allocation.
  - `dhcp_v4_config` - (Optional) DHCPv4 config for IPv4 subnet. This clause is supported with NSX 3.0.0 onwards.
  - `server_address` - (Optional) IP address of the DHCP server in CIDR format. This attribute is required if segment has provided dhcp_config_path and it represents a DHCP server config.
  - `dns_servers` - (Optional) List of IP addresses of DNS servers for the subnet.
  - `lease_time` - (Optional) DHCP lease time in seconds.
  - `dhcp_option_121` - (Optional) DHCP classless static routes.
    - `network` - (Required) Destination in cidr format.
    - `next_hop` - (Required) IP address of next hop.
  - `dhcp_generic_option` - (Optional) Generic DHCP options.
    - `code` - (Required) DHCP option code. Valid values are from 0 to 255.
    - `values` - (Required) List of DHCP option values.
  - `dhcp_v6_config` - (Optional) DHCPv6 config for IPv6 subnet. This clause is supported with NSX 3.0.0 onwards.
  - `server_address` - (Optional) IP address of the DHCP server in CIDR format. This attribute is required if segment has provided dhcp_config_path and it represents a DHCP server config.
  - `dns_servers` - (Optional) List of IP addresses of DNS servers for the subnet.

    - `lease_time` - (Optional) DHCP lease time in seconds.
    - `preferred_time` - (Optional) The time interval in seconds, in which the prefix is advertised as preferred.
    - `domain_names` - (Optional) List of domain names for this subnet.
    - `excluded_range` - (Optional) List of excluded address ranges to define dynamic ip allocation ranges.
        - `start` - (Required) IPv6 address that marks beginning of the range.
        - `end` - (Required) IPv6 address that marks end of the range.
    - `sntp_servers` - (Optional) IPv6 address of SNTP servers for the subnet.
- `l2_extension` - (Optional) Configuration for extending Segment through L2 VPN.
    - `l2vpn_paths` - (Optional) Policy paths of associated L2 VPN sessions.
    - `tunnel_id` - (Optional) The Tunnel ID that's a int value between 1 and 4093.
- `advanced_config` - (Optional) Advanced Segment configuration.
    - `address_pool_paths` - (Optional) List of Policy path to IP address pools.
    - `connectivity` - (Optional) Connectivity configuration to manually connect (ON) or disconnect (OFF).
    - `hybrid` - (Optional) Boolean flag to identify a hybrid logical switch.
    - `local_egress` - (Optional) Boolean flag to enable local egress.
    - `uplink_teaming_policy` - (Optional) The name of the switching uplink teaming policy for the bridge endpoint. This name corresponds to one of the switching uplink teaming policy names listed in the transport zone.

» Attributes Reference

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

- `id` - ID of the Secuirty Policy.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.
- In the `subnet`:
    - `network` The network CIDR for the subnet.

» Importing

An existing segment can be imported into this resource, via the following command:

```
terraform import nsxt_policy_segment.segment1 ID
```

The above command imports the segment named `segment1` with the NSX Segment ID `ID`.

» nsxt_policy_vlan_segment

This resource provides a method for the management of VLAN backed Segments.

» Example Usage

```
resource "nsxt_policy_vlan_segment" "vlansegment1" {
  display_name        = "vlansegment1"
  description         = "Terraform provisioned VLAN Segment"
  transport_zone_path = data.nsxt_policy_transport_zone.vlantz.path
  domain_name         = "tftest2.org"
  vlan_ids            = ["101", "102"]

  subnet {
    cidr        = "12.12.2.1/24"
    dhcp_ranges = ["12.12.2.100-12.12.2.160"]

    dhcp_v4_config {
      server_address = "12.12.2.2/24"
      lease_time     = 36000

      dhcp_option_121 {
        network  = "6.6.6.0/24"
        next_hop = "1.1.1.21"
      }

      dhcp_generic_option {
        code = "119"
        values = ["abc"]
      }
    }
  }

  advanced_config {
    connectivity = "OFF"
    local_egress = true
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this policy.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `domain_name`- (Optional) DNS domain names.
- `transport_zone_path` - (Required) Policy path to the VLAN backed transport zone.
- `vlan_ids` - (Optional) VLAN IDs for VLAN backed Segment.
- `dhcp_config_path` - (Optional) Policy path to DHCP server or relay configuration to use for subnets configured on this segment. This attribute is supported with NSX 3.0.0 onwards.
- `subnet` - (Required) Subnet configuration block.
  - `cidr` - (Required) Gateway IP address CIDR.
  - `dhcp_ranges` - (Optional) List of DHCP address ranges for dynamic IP allocation.
  - `dhcp_v4_config` - (Optional) DHCPv4 config for IPv4 subnet. This attribute is supported with NSX 3.0.0 onwards.
  - `server_address -` (Optional) IP address of the DHCP server in CIDR format. This attribute is required if segment has provided dhcp_config_path and it represents a DHCP server config.
  - `dns_servers` - (Optional) List of IP addresses of DNS servers for the subnet.
  - `lease_time` - (Optional) DHCP lease time in seconds.
  - `dhcp_option_121` - (Optional) DHCP classless static routes.
    - `network` - (Required) Destination in cidr format.
    - `next_hop` - (Required) IP address of next hop.
  - `dhcp_generic_option` - (Optional) Generic DHCP options.
    - `code` - (Required) DHCP option code. Valid values are from 0 to 255.
    - `values` - (Required) List of DHCP option values.
  - `dhcp_v6_config` - (Optional) DHCPv6 config for IPv6 subnet. This attribute is supported with NSX 3.0.0 onwards.
  - `server_address -` (Optional) IP address of the DHCP server in CIDR format. This attribute is required if segment has provided dhcp_config_path and it represents a DHCP server config.
  - `dns_servers` - (Optional) List of IP addresses of DNS servers for the subnet.
  - `lease_time` - (Optional) DHCP lease time in seconds.
  - `preferred_time` - (Optional) The time interval in seconds, in which the prefix is advertised as preferred.
  - `domain_names` - (Optional) List of domain names for this subnet.

    - `excluded_range` - (Optional) List of excluded address ranges to define dynamic ip allocation ranges.
        - `start` - (Required) IPv6 address that marks beginning of the range.
        - `end` - (Required) IPv6 address that marks end of the range.
    - `sntp_servers` - (Optional) IPv6 address of SNTP servers for the subnet.
- `l2_extension` - (Optional) Configuration for extending Segment through L2 VPN.
    - `l2vpn_paths` - (Optional) Policy paths of associated L2 VPN sessions.
    - `tunnel_id` - (Optional) The Tunnel ID that's a int value between 1 and 4093.
- `advanced_config` - (Optional) Advanced Segment configuration.
    - `address_pool_paths` - (Optional) List of Policy path to IP address pools.
    - `connectivity` - (Optional) Connectivity configuration to manually connect (ON) or disconnect (OFF).
    - `hybrid` - (Optional) Boolean flag to identify a hybrid logical switch.
    - `local_egress` - (Optional) Boolean flag to enable local egress.
    - `uplink_teaming_policy` - (Optional) The name of the switching uplink teaming policy for the bridge endpoint. This name corresponds to one of the switching uplink teaming policy names listed in the transport zone.

» Attributes Reference

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

- `id` - ID of the Secuirty Policy.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing segment can be imported into this resource, via the following command:

```
terraform import nsxt_policy_vlan_segment.segment1 ID
```

The above command imports the VLAN backed segment named `segment1` with the NSX Segment ID `ID`.

» nsxt_policy_vm_tags

This resource provides a means to configure tags that are applied to objects such as Virtual Machines. A Virtual Machine is not directly managed by NSX however, NSX allows attachment of tags to a virtual machine. This tagging enables tag based grouping of objects. Deletion of `nsxt_policy_vm_tags` resource will remove all tags from the Virtual Machine and is equivalent to update operation with empty tag set.

» Example Usage

```
resource "nsxt_policy_vm_tags" "vm1_tags" {
  instance_id = "${vsphere_virtual_machine.vm1.id}"

  tag {
    scope = "color"
    tag   = "blue"
  }

  tag {
    scope = "env"
    tag   = "test"
  }
}
```

» Argument Reference

The following arguments are supported:

- `instance_id` - (Required) ID of the Virtual Machine. Can be the instance UUID or BIOS UUID.
- `tag` - (Optional) A list of scope + tag pairs to associate with this Virtual Machine.

» Importing

An existing Tags collection can be imported into this resource, via the following command:

```
terraform import nsxt_policy_vm_tags.vm1_tags ID
```

The above would import NSX Virtual Machine tags as a resource named `vm1_tags` with the NSX ID `ID`, where ID is external ID of the Virtual Machine.

» nsxt_policy_dhcp_server

This resource provides a method for the management of a DHCP Server configurations. This resource is supported with NSX 3.0.0 onwards.

» Example Usage

```
resource "nsxt_policy_dhcp_server" "test" {
  display_name      = "test"
  description       = "Terraform provisioned DhcpServerConfig"
  edge_cluster_path = data.nsxt_policy_edge_cluster.ec1.path
  lease_time        = 200
  server_addresses  = ["110.64.0.1/16", "2001::1234:abcd:ffff:c0a8:101/64"]
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `edge_cluster_path` - (Optional) The Policy path to the edge cluster for this DHCP Server.
- `lease_time` - (Optional) IP address lease time in seconds. Valid values from `60` to `4294967295`. Default is `86400`.
- `preferred_edge_paths` - (Optional) Policy paths to edge nodes. The first edge node is assigned as active edge, and second one as standby edge.
- `server_addresses` - (Optional) DHCP server address in CIDR format. At most 2 supported; one IPv4 and one IPv6 address. Server address can also be specified on segment subnet level.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing DHCP Server can be imported into this resource, via the following command:

```
terraform import nsxt_policy_dhcp_server.dhcp1 ID
```

The above command imports a DHCP Server named `dhcp1` with the NSX DHCP Server ID `ID`.

» nsxt_policy_lb_pool

This resource provides a method for the management of a LBPool.

» Example Usage

```
resource "nsxt_policy_lb_pool" "test" {
    display_name         = "test"
    description          = "Terraform provisioned LB Pool"
    algorithm            = "IP_HASH"
    min_active_members   = 2
    active_monitor_path  = "/infra/lb-monitor-profiles/default-icmp-lb-monitor"
    passive_monitor_path = "/infra/lb-monitor-profiles/default-passive-lb-monitor"
    member {
      admin_state                = "ENABLED"
      backup_member              = false
      display_name               = "member1"
      ip_address                 = "5.5.5.5"
      max_concurrent_connections = 12
      port                       = "77"
      weight                     = 1
    }
    snat {
       type = "AUTOMAP"
    }
    tcp_multiplexing_enabled = true
    tcp_multiplexing_number  = 8
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.

- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `algorithm` - (Optional) Load balancing algorithm, one of `ROUND_ROBIN`, `WEIGHTED_ROUND_ROBIN`, `LEAST_CONNECTION`, `WEIGHTED_LEAST_CONNECTION`, `IP_HASH`. Default is `ROUND_ROBIN`.
- `member_group` - (Optional) Grouping specification for pool members. When `member_group` is set, `member` should not be specified.
  - `group_path` - (Required) Path for policy group.
  - `allow_ipv4` - (Optional) Use IPv4 addresses from the grouping object, default is `true`.
  - `allow_ipv6` - (Optional) Use IPv6 addresses from the grouping object, default is `true`
  - `max_ip_list_size` - (Optional) Maximum number of IPs to use from the grouping object.
  - `port` - (Optional) If port is specified, all connections will be redirected to this port.
- `member`- (Optional) Members of the pool. When `member' is set,`member_group‘ should not be specified.
  - `ip_address` - (Required) Member IP address.
  - `admin_state` - (Optional) One of `ENABLED`, `DISABLED`, `GRACEFUL_DISABLED`. Default is `ENABLED`.
  - `backup_member` - (Optional) Whether this member is a backup member.
  - `display_name` - (Optional) Display name of the member.
  - `max_concurrent_connections` - (Optional) To ensure members are not overloaded, connections to a member can be capped by this setting.
  - `port` - (Optional) If port is specified, all connections will be redirected to this port.
  - `weight` - (Optional) Pool member weight is used for WEIGHTED algorithms.
- `min_active_members` - (Optional) A pool is considered active if there are at least certain minimum number of members.
- `active_monitor_path` - (Optional) Active monitor to be associated with this pool.
- `passive_monitor_path` - (Optional) Passive monitor to be associated with this pool.
- `snat` - (Optional) Source NAT may be required to ensure traffic from the server destined to the client is received by the load balancer.
  - `type` - (Optional) SNAT type, one of ’AUTOMAP`,`DISABLED`,`IPPOOL`. Default is`AUTOMAP‘.
  - `ip_pool_addresses` - (Optional) List of IP ranges or IP CIDRs to use for IPPOOL SNAT type.
- `tcp_multiplexing_enabled` - (Optional) Enable TCP multiplexing

within the pool.
- `tcp_multiplexing_number` - (Optional) The maximum number of TCP connections per pool that are idly kept alive for sending future client requests.

» Attributes Reference

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

- `id` - ID of the Secuirty Policy.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing pool can be imported into this resource, via the following command:

```
terraform import nsxt_policy_lb_pool.test ID
```

The above command imports LBPool named `test` with the NSX LBPool ID `ID`.

» nsxt_policy_lb_service

This resource provides a method for the management of a Load Balancer Service.

» Example Usage

```
data "nsxt_policy_tier1_gateway" "test" {
  display_name = "test"
}


resource "nsxt_policy_lb_service" "test" {
  display_name      = "test"
  description       = "Terraform provisioned Service"
  connectivity_path = data.nsxt_policy_tier1_gateway.test.path
  size = "SMALL"
  enabled = true
  error_log_level = "ERROR"
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `size` - (Optional) Load Balancer Service size, one of `SMALL`, `MEDIUM`, `LARGE`, `XLARGE`, `DLB`. Default is `SMALL`.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.
- `connectivity_path` - (Optional) Tier1 Gateway where this service is instantiated. In future, other objects will be supported.
- `enabled` - (Optional) Flag to enable the service.
- `error_log_level` - (Optional) Log level for the service, one of `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`, `ALERT`, `EMERGENCY`. Default is `INFO`.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing service can be imported into this resource, via the following command:

```
terraform import nsxt_policy_lb_service.test ID
```

The above command imports LBService named `test` with the NSX Load Balancer Service ID `ID`.

» nsxt_policy_lb_virtual_server

This resource provides a method for the management of a Load Balancer Virtual Server.

» Example Usage

```
resource "nsxt_policy_lb_virtual_server" "test" {
  display_name               = "test"
  description                = "Terraform provisioned Virtual Server"
  access_log_enabled         = true
  application_profile_path   = data.nsxt_policy_lb_app_profile.tcp.path
  enabled                    = true
  ip_address                 = "10.10.10.21"
  ports                      = ["443"]
  default_pool_member_ports  = ["80"]
  service_path               = nsxt_policy_lb_service.app1.path
  max_concurrent_connections = 6
  max_new_connection_rate    = 20
  pool_path                  = nsxt_policy_lb_pool.pool1.path
  sorry_pool_path            = nsxt_policy_lb_pool.sorry_pool.path

  client_ssl {
    client_auth              = "REQUIRED"
    default_certificate_path = data.nsxt_policy_certificate.cert1.path
    ca_paths                 = [data.nsxt_policy_certificate.lb_ca.path]
    certificate_chain_depth  = 3
    ssl_profile_path         = data.nsxt_policy_lb_client_ssl_profile.lb_profile.path
  }

  server_ssl {
    server_auth             = "REQUIRED"
    client_certificate_path = data.nsxt_policy_certificate.client_ca.path
    certificate_chain_depth = 3
    ssl_profile_path        = data.nsxt_policy_lb_server_ssl_profile.lb_profile.path
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Required) Display name of the resource.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this resource.
- `nsx_id` - (Optional) The NSX ID of this resource. If set, this ID will be used to create the resource.

- `application_profile_path` - (Required) Application profile path for this virtual server.
- `access_log_enabled` - (Optional) If set, all connections/requests sent to the virtual server are logged to access log.
- `default_pool_member_ports` - (Optional) Default pool member ports to use when member port is not defined on the pool.
- `enabled` - (Optional) Flag to enable this Virtual Server.
- `ip_address` - (Required) Virtual Server IP address.
- `ports` - (Required) Virtual Server Ports.
- `persistence_profile_path` - (Optional) Path to persistance profile allowing related client connections to be sent to the same backend server.
- `service_path` - (Optional) Virtual Server can be associated with Load Balancer Service.
- `max_concurrent_connections` - (Optional) To ensure one virtual server does not over consume resources, connections to Virtual Server can be capped.
- `max_new_connection_rate` - (Optional) To ensure one virtual server does not over consume resources, connections to a member can be rate limited.
- `pool_path` - (Optional)Path for Load Balancer Pool.
- `sorry_pool_path` - (Optional) When load balancer can not select server in pool, the request would be served by sorry pool
- `server_ssl` - (Optional)
  - `server_auth` - (Optional) Server Authentication Mode, one of `REQUIRED`, `IGNORE`, `AUTO_APPLY`. Default is `AUTO_APPLY`.
  - `certificate_chain_depth` - (Optional) Allowed certificate chain depth.
  - `client_certificate_path` - (Optional) Client certificat path for client authentication against the server.
  - `ca_paths` - (Optional) If server auth type is REQUIRED, client certificate must be signed by one Certificate Authorities provided here.
  - `crl_paths` - (Optional) Certificate Revocation Lists can be specified to disallow compromised certificate.
  - `ssl_profile_path` - (Optional) Server SSL profile path.
- `client_ssl` - (Optional)
  - `client_auth` - (Optional) Client Authentication Mode, one of `REQUIRED`, `IGNORE`. Default is `IGNORE`.
  - `certificate_chain_depth` - (Optional) Allowed certificate chain depth.

  - `ca_paths` - (Optional) If client auth type is REQUIRED, client certificate must be signed by one Certificate Authorities provided here.
  - `crl_paths` - (Optional) Certificate Revocation Lists can be specified to disallow compromised client certificate.
  - `default_certificate_path` - (Optional) Default Certificate Path. Must be specified if client_auth is set to `REQUIRED`.
  - `sni_paths` - (Optional) This setting allows multiple certificates(for different hostnames) to be bound to the same virtual server.
  - `ssl_profile_path` - (Optional) Client SSL profile path.

- `log_significant_event_only` - (Optional) If true, significant events are logged in access log. This flag is supported since NSX 3.0.0.

- `access_list_control` - (Optional) Specifies the access list control to define how to filter client connections.

  - `action` - (Required) Action for connections matching the grouping object.
  - `group_path` - (Required) The path of grouping object which defines the IP addresses or ranges to match client IP.
  - `enabled` - (Optional) Indicates whether to enable access list control option. Default is true.

» Attributes Reference

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

- `id` - ID of the resource.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `path` - The NSX path of the policy resource.

» Importing

An existing Virtual Server can be imported into this resource, via the following command:

```
terraform import nsxt_policy_lb_virtual_server.test ID
```

The above command imports Load Balancer Virtual Server named `test` with the NSX Load Balancer Virtual Server ID `ID`.

» nsxt_edge_cluster

This data source provides information about Edge clusters configured in NSX. An Edge cluster is a collection of Edge nodes which can be deployed as either VM form-factor or bare-metal form-factor machines for connectivity between overlay logical switches and non-NSX underlay networking for north/south layer 2 or layer 3 connectivity. Each T0 router will be placed on one ore more Edge nodes in an Edge cluster therefore this data source is needed for the creation of T0 logical routers.

» Example Usage

```
data "nsxt_edge_cluster" "edge_cluster1" {
  display_name = "edgecluster"
}
```

» Argument Reference

- `id` - (Optional) The ID of Edge Cluster to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Edge Cluster to retrieve.

» Attributes Reference

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

- `description` - The description of the edge cluster.
- `deployment_type` - This field could show deployment_type of members. It would return UNKNOWN if there is no members, and return VIRTUAL_MACHINE|PHYSICAL_MACHINE if all Edge members are VIRTUAL_MACHINE|PHYSICAL_MACHINE.
- `member_node_type` - An Edge cluster is homogeneous collection of NSX transport nodes used for north/south connectivity between NSX logical networking and physical networking. Hence all transport nodes of the cluster must be of same type. This field shows the type of transport node,

» nsxt_logical_tier0_router

This data source provides information about logical Tier 0 routers configured in NSX. A Tier 0 router is used to connect NSX networking with traditional physical networking. Tier 0 routers are placed on an Edge cluster and will exist on one or more Edge node depending on deployment settings (i.e. active/active or active/passive). A Tier 0 router forwards layer 3 IP packets and typically peers with a traditional physical router using BGP or can use static routing.

» Example Usage

```
data "nsxt_logical_tier0_router" "tier0_router" {
  display_name = "PLR1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Logical Tier 0 Router to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Logical Tier 0 Router to retrieve.

» Attributes Reference

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

- `description` - The description of the logical Tier 0 router.
- `edge_cluster_id` - The id of the Edge cluster where this logical router is placed.
- `high_availability_mode` - The high availability mode of this logical router.

» nsxt_logical_tier1_router

This data source provides information about logical Tier 1 routers configured in NSX.

» Example Usage

```
data "nsxt_logical_tier1_router" "tier1_router" {
  display_name = "router1"
}
```

» Argument Reference

- `id` - (Optional) The ID of Logical Tier 1 Router to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Logical Tier 1 Router to retrieve.

» Attributes Reference

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

- `description` - The description of the logical Tier 0 router.
- `edge_cluster_id` - The id of the Edge cluster where this logical router is placed.

» nsxt_ns_group

This data source provides information about a network and security (NS) group in NSX. A NS group is used to group other objects into collections for application of other settings.

» Example Usage

```
data "nsxt_ns_group" "ns_group_1" {
  display_name = "test group"
}
```

» Argument Reference

- `id` - (Optional) The ID of NS group to retrieve
- `display_name` - (Optional) The Display Name of the NS group to retrieve.

» Attributes Reference

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

- `description` - The description of the NS group.

» nsxt_ns_service

This data source provides information about a network and security (NS) service configured in NSX. NS services are either factory defined in NSX or can be defined by the NSX administrator. They provide a convenience name for a port/protocol pair that is often used in fire walling or load balancing.

» Example Usage

```
data "nsxt_ns_service" "ns_service_dns" {
  display_name = "DNS"
}
```

» Argument Reference

- `id` - (Optional) The ID of NS service to retrieve
- `display_name` - (Optional) The Display Name of the NS service to retrieve.

» Attributes Reference

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

- `description` - The description of the NS service.

» nsxt_mac_pool

This data source provides information about a MAC pool configured in NSX.

» Example Usage

```
data "nsxt_mac_pool" "mac_pool" {
  display_name = "DefaultMacPool"
}
```

» Argument Reference

- `id` - (Optional) The ID of MAC pool to retrieve

- display_name - (Optional) The Display Name of the MAC pool to retrieve.

» Attributes Reference

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

- description - The description of the MAC pool.

» nsxt_switching_profile

The switching profile data source provides information about switching profiles configured in NSX. A switching profile is a template that defines the settings of one or more logical switches. There can be both factory default and user defined switching profiles. One example of a switching profile is a quality of service (QoS) profile which defines the QoS settings of all switches that use the defined switch profile.

» Example Usage

```
data "nsxt_switching_profile" "qos_profile" {
  display_name = "qos-profile"
}
```

» Argument Reference

- id - (Optional) The ID of Switching Profile to retrieve.
- display_name - (Optional) The Display Name of the Switching Profile to retrieve.

» Attributes Reference

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

- resource_type - The resource type representing the specific type of this switching profile.
- description - The description of the switching profile.

» nsxt_transport_zone

This data source provides information about Transport Zones (TZ) configured in NSX. A Transport Zone defines the scope to which a network can extend in NSX. For example an overlay based Transport Zone is associated with both hypervisors and logical switches and defines which hypervisors will be able to serve the defined logical switch. Virtual machines on the hypervisor associated with a Transport Zone can be attached to logical switches in that same Transport Zone.

» Example Usage

```
data "nsxt_transport_zone" "overlay_transport_zone" {
  display_name = "1-transportzone-87"
}
```

» Argument Reference

- `id` - (Optional) The ID of Transport Zone to retrieve.
- `display_name` - (Optional) The Display Name prefix of the Transport Zone to retrieve.

» Attributes Reference

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

- `description` - The description of the Transport Zone.
- `host_switch_name` - The name of the N-VDS (host switch) on all Transport Nodes in this Transport Zone that will be used to run NSX network traffic.
- `transport_type` - The transport type of this transport zone (OVERLAY or VLAN).

» nsxt_certificate

This data source provides information about various types of certificates imported into NSX trust management.

» Example Usage

```
data "nsxt_certificate" "CA" {
  display_name = "ca-cert"
}
```

» Argument Reference

- `id` - (Optional) The ID of Certificate to retrieve.
- `display_name` - (Optional) The Display Name of the Certificate to retrieve.

» Attributes Reference

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

- `description` - The description of the Certificate.

» nsxt_ip_pool

This data source provides information about a IP pool configured in NSX.

» Example Usage

```
data "nsxt_ip_pool" "ip_pool" {
  display_name = "DefaultIpPool"
}
```

» Argument Reference

- `id` - (Optional) The ID of IP pool to retrieve
- `display_name` - (Optional) The Display Name of the IP pool to retrieve.

» Attributes Reference

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

- `description` - The description of the IP pool.

» nsxt_firewall_section

This data source provides information about firewall section in NSX.

» Example Usage

```
data "nsxt_firewall_section" "block_all" {
  display_name = "block all"
}
```

» Argument Reference

- `id` - (Optional) The ID of resource to retrieve
- `display_name` - (Optional) The Display Name of resource to retrieve.

» Attributes Reference

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

- `description` - The description of resource.

» nsxt_dhcp_relay_profile

This resource can be used to configure a NSX DHCP relay profile on the NSX manager. A DHCP relay profile is a type of template that can be used to define a remote DHCP server where DHCP packets can be relayed for DHCP requests of machines attached to NSX logical topologies. The DHCP relay profile can be used in a DHCP relay service and later consumed by a router downlink port. Currently the DHCP relay is not supported for logical routers link ports on Tier0 or Tier1.

» Example Usage

```
resource "nsxt_dhcp_relay_profile" "dr_profile" {
  description  = "DRP provisioned by Terraform"
  display_name = "DRP"

  tag {
    scope = "color"
    tag   = "red"
  }

  server_addresses = ["1.1.1.1"]
}

resource "nsxt_dhcp_relay_service" "dr_service" {
  display_name          = "DRS"
  dhcp_relay_profile_id = "${nsxt_dhcp_relay_profile.dr_profile.id}"
}

resource "nsxt_logical_router_downlink_port" "router_downlink" {
  display_name                  = "logical_router_downlink_port"
  linked_logical_switch_port_id = "${nsxt_logical_port.port1.id}"
  logical_router_id             = "${nsxt_logical_tier1_router.rtr1.id}"

  subnet {
    ip_addresses  = ["8.0.0.1"]
    prefix_length = 24
  }

  service_binding {
    target_id   = "${nsxt_dhcp_relay_service.dr_service.id}"
    target_type = "LogicalService"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this DHCP relay profile.
- `server_addresses` - (Required) IP addresses of the DHCP relay servers. Maximum allowed amount is 2.

» Attributes Reference

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

- `id` - ID of the DHCP relay profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing DHCP Relay profile can be imported into this resource, via the following command:

```
terraform import nsxt_dhcp_relay_profile.dr_profile UUID
```

The above command imports the DHCP relay profile named `dr_profile` with the NSX id `UUID`.

» nsxt_dhcp_relay_service

This resource provides a way to configure the DHCP relay service on the NSX manager. The DHCP relay service uses a DHCP relay profile and later consumed by a router downlink port to provide DHCP addresses to virtual machines connected to a logical switch. Currently the DHCP relay is not supported for logical routers link ports on Tier0 or Tier1.

» Example Usage

```
resource "nsxt_dhcp_relay_profile" "dr_profile" {
  description  = "DRP provisioned by Terraform"
  display_name = "DRP"

  tag {
    scope = "color"
    tag   = "red"
  }

  server_addresses = ["1.1.1.1"]
}

resource "nsxt_dhcp_relay_service" "dr_service" {
  display_name          = "DRS"
  dhcp_relay_profile_id = "${nsxt_dhcp_relay_profile.dr_profile.id}"
}

resource "nsxt_logical_router_downlink_port" "router_downlink" {
  display_name                  = "logical_router_downlink_port"
  linked_logical_switch_port_id = "${nsxt_logical_port.port1.id}"
  logical_router_id             = "${nsxt_logical_tier1_router.rtr1.id}"

  subnet {
    ip_addresses  = ["8.0.0.1"]
    prefix_length = 24
  }

  service_binding {
    target_id   = "${nsxt_dhcp_relay_service.dr_service.id}"
    target_type = "LogicalService"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this dhcp_relay_service.
- `dhcp_relay_profile_id` - (Required) DHCP relay profile referenced by the DHCP relay service.

» Attributes Reference

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

- `id` - ID of the DHCP relay service.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing DHCP Relay service can be imported into this resource, via the following command:

```
terraform import nsxt_dhcp_relay_service.dr_service UUID
```

The above command imports the DHCP relay service named `dr_service` with the NSX id `UUID`.

» nsxt_dhcp_server_profile

Provides a resource to configure DHCP server profile on NSX-T manager

» Example Usage

```
data "nsxt_edge_cluster" "edge_cluster1" {
  display_name = "edgecluster"
}

resource "nsxt_dhcp_server_profile" "dhcp_profile" {
  description                 = "dhcp_profile provisioned by Terraform"
  display_name                = "dhcp_profile"
  edge_cluster_id             = "${data.nsxt_edge_cluster.edge_cluster1.id}"
  edge_cluster_member_indexes = [0, 1]

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `edge_cluster_id` - (Required) Edge cluster uuid.
- `edge_cluster_member_indexes` - (Optional) Up to 2 edge nodes from the given cluster. If none is provided, the NSX will auto-select two edge-nodes from the given edge cluster. If user provides only one edge node, there will be no HA support.
- `tag` - (Optional) A list of scope + tag pairs to associate with this DHCP profile.

» Attributes Reference

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

- `id` - ID of the DHCP server profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing DHCP profile can be imported into this resource, via the following command:

```
terraform import nsxt_dhcp_server_profile.dhcp_profile UUID
```

The above would import the DHCP server profile named `dhcp_profile` with the nsx id `UUID`

» nsxt_firewall_section

This resource provides a way to configure a firewall section on the NSX manager. A firewall section is a collection of firewall rules that are grouped together. Order of firewall sections can be controlled with 'insert_before' attribute.

» Example Usage

```
data "nsxt_firewall_section" "block_all" {
  display_name = "BLOCK"
}

resource "nsxt_firewall_section" "firewall_sect" {
  description  = "FS provisioned by Terraform"
  display_name = "FS"

  tag {
    scope = "color"
    tag   = "blue"
  }

  applied_to {
    target_type = "NSGroup"
    target_id   = "${nsxt_ns_group.group1.id}"
  }

  section_type  = "LAYER3"
  stateful      = true
  insert_before = "${data.nsxt_firewall_section.block_all.id}"

  rule {
    display_name          = "out_rule"
    description           = "Out going rule"
    action                = "ALLOW"
    logged                = true
    ip_protocol           = "IPV4"
    direction             = "OUT"
    destinations_excluded = "false"
    sources_excluded      = "true"

    source {
      target_type = "LogicalSwitch"
      target_id   = "${nsxt_logical_switch.switch1.id}"
    }

    destination {
      target_type = "LogicalSwitch"
      target_id   = "${nsxt_logical_switch.switch2.id}"
    }
  }

  rule {
    display_name = "in_rule"
    description  = "In going rule"
    action       = "DROP"
    logged       = true
    ip_protocol  = "IPV4"
    direction    = "IN"

    service {
      target_type = "NSService"
      target_id   = "e8d59e13-484b-4825-ae3b-4c11f83249d9"
    }

    service {
      target_type = "NSService"
      target_id   = "${nsxt_l4_port_set_ns_service.http.id}"
    }
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this firewall section. Defaults to ID if not set.
- `description` - (Optional) Description of this firewall section.
- `tag` - (Optional) A list of scope + tag pairs to associate with this firewall

section.

- `applied_to` - (Optional) List of objects where the rules in this section will be enforced. This will take precedence over rule level applied_to. [Supported target types: ”LogicalPort”, ”LogicalSwitch”, ”NSGroup”, ”LogicalRouter”]
- `section_type` - (Required) Type of the rules which a section can contain. Either LAYER2 or LAYER3. Only homogeneous sections are supported.
- `stateful` - (Required) Stateful or Stateless nature of firewall section is enforced on all rules inside the section. Layer3 sections can be stateful or stateless. Layer2 sections can only be stateless.
- `insert_before` - (Optional) Firewall section id that should come immediately after this one. It is user responsibility to use this attribute in consistent manner (for example, if same value would be set in two separate sections, the outcome would depend on order of creation). Changing this attribute would force recreation of the firewall section.
- `rule` - (Optional) A list of rules to be applied in this section. each rule has the following arguments:
  - `display_name` - (Optional) The display name of this rule. Defaults to ID if not set.
  - `description` - (Optional) Description of this rule.
  - `action` - (Required) Action enforced on the packets which matches the firewall rule. [Allowed values: ”ALLOW”, ”DROP”, ”REJECT”]
  - `applied_to` - (Optional) List of objects where rule will be enforced. The section level field overrides this one. Null will be treated as any. [Supported target types: ”LogicalPort”, ”LogicalSwitch”, ”NSGroup”, ”LogicalRouterPort”]
  - `destination` - (Optional) List of the destinations. Null will be treated as any. [Allowed target types: ”IPSet”, ”LogicalPort”, ”LogicalSwitch”, ”NSGroup”, ”MACSet” (depending on the section type)]
  - `destinations_excluded` - (Optional) When this boolean flag is set to true, the rule destinations will be negated.
  - `direction` - (Optional) Rule direction in case of stateless firewall rules. This will only considered if section level parameter is set to stateless. Default to IN_OUT if not specified. [Allowed values: ”IN”, ”OUT”, ”IN_OUT”]
  - `disabled` - (Optional) Flag to disable rule. Disabled will only be persisted but never provisioned/realized.
  - `ip_protocol` - (Optional) Type of IP packet that should be matched while enforcing the rule. [allowed values: ”IPV4”, ”IPV6”, ”IPV4_IPV6”]
  - `logged` - (Optional) Flag to enable packet logging. Default is disabled.
  - `notes` - (Optional) User notes specific to the rule.
  - `rule_tag` - (Optional) User level field which will be printed in CLI and packet logs.
  - `service` - (Optional) List of the services. Null will be treated as any.

  [Allowed target types: "NSService", "NSServiceGroup"]
  - source - (Optional) List of sources. Null will be treated as any. [Allowed target types: "IPSet", "LogicalPort", "LogicalSwitch", "NSGroup", "MACSet" (depending on the section type)]
  - sources_excluded - (Optional) When this boolean flag is set to true, the rule sources will be negated.

» Attributes Reference

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

- id - ID of the firewall section.
- revision - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- is_default - A boolean flag which reflects whether a firewall section is default section or not. Each Layer 3 and Layer 2 section will have at least and at most one default section.

» Importing

An existing Firewall section can be imported into this resource, via the following command:

```
terraform import nsxt_firewall_section.firewall_sect UUID
```

The above command imports the firewall section named firewall_sect with the NSX id UUID.

» nsxt_ip_block

Provides a resource to configure IP block on NSX-T manager

» Example Usage

```
resource "nsxt_ip_block" "ip_block" {
  description  = "ip_block provisioned by Terraform"
  display_name = "ip_block"
  cidr         = "2.1.1.0/24"

  tag {
    scope = "color"
    tag   = "red"
  }
}

resource "nsxt_ip_block_subnet" "ip_block_subnet" {
  description = "ip_block_subnet"
  block_id    = "${nsxt_ip_block.ip_block.id}"
  size        = 16
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `cidr` - (Required) Represents network address and the prefix length which will be associated with a layer-2 broadcast domain.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP block.

» Attributes Reference

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

- `id` - ID of the IP block.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IP block can be imported into this resource, via the following command:

```
terraform import nsxt_ip_block.ip_block UUID
```

The above would import the IP block named `ip_block` with the nsx id `UUID`

» nsxt__ip__block__subnet

Provides a resource to configure IP block subnet on NSX-T manager

» Example Usage

```
resource "nsxt_ip_block" "ip_block" {
  display_name = "block1"
  cidr         = "55.0.0.0/24"
}

resource "nsxt_ip_block_subnet" "ip_block_subnet" {
  description  = "ip_block_subnet provisioned by Terraform"
  display_name = "ip_block_subnet"
  block_id     = "${nsxt_ip_block.ip_block.id}"
  size         = 16

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `block_id` - (Required) Block id for which the subnet is created.
- `size` - (Required) Represents the size or number of IP addresses in the subnet.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP block subnet.

» Attributes Reference

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

- `id` - ID of the IP block subnet.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `allocation_range` - A collection of IPv4 IP ranges used for IP allocation.
- `cidr` - Represents the size or number of IP addresses in the subnet. All subnets of the same block must have the same size, which must be a power of 2.

» Importing

An existing IP block subnet can be imported into this resource, via the following command:

```
terraform import nsxt_ip_block_subnet.ip_block_subnet UUID
```

The above would import the IP block subnet named `ip_block_subnet` with the nsx id `UUID`

» nsxt_ip_pool

Provides a resource to configure IP pool on NSX-T manager

» Example Usage

```
resource "nsxt_ip_pool" "ip_pool" {
  description = "ip_pool provisioned by Terraform"
  display_name = "ip_pool"

  tag {
    scope = "color"
    tag   = "red"
  }

  subnet {
    allocation_ranges = ["2.1.1.1-2.1.1.11", "2.1.1.21-2.1.1.100"]
    cidr              = "2.1.1.0/24"
    gateway_ip        = "2.1.1.12"
    dns_suffix        = "abc"
    dns_nameservers   = ["33.33.33.33"]
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP pool.

- subnet - (Optional) Subnets can be IPv4 or IPv6 and they should not overlap. The maximum number will not exceed 5 subnets. Each subnet has the following arguments:
  - allocation_ranges - (Required) A collection of IPv4 Pool Ranges
  - cidr - (Required) Network address and the prefix length which will be associated with a layer-2 broadcast domainIPv4 Pool Ranges
  - dns_nameservers - (Optional) A collection of up to 3 DNS servers for the subnet
  - dns_suffix - (Optional) The DNS suffix for the DNS server
  - gateway_ip - (Optional) The default gateway address on a layer-3 router

» Attributes Reference

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

- id - ID of the IP pool.
- revision - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IP pool can be imported into this resource, via the following command:

```
terraform import nsxt_ip_pool.ip_pool UUID
```

The above would import the IP pool named ip_pool with the nsx id UUID

» nsxt_ip_pool_allocation_ip_address

Provides a resource to allocate an IP address from an IP pool on NSX-T manager

» Example Usage

```
data "nsxt_ip_pool" "ip_pool" {
  display_name = "DefaultIpPool"
}

resource "nsxt_ip_pool_allocation_ip_address" "pool_ip_address" {
  ip_pool_id = "${data.nsxt_ip_pool.ip_pool.id}"
}
```

» Argument Reference

The following arguments are supported:

- `ip_pool_id` - (Required) Ip Pool ID from which the IP address will be allocated.

» Attributes Reference

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

- `id` - ID of the IP pool allocation IP address (currently identical to `allocation_ip`).
- `allocation_ip` - Allocation IP address.

» nsxt_ip_set

This resources provides a way to configure an IP set in NSX. An IP set is a collection of IP addresses. It is often used in the configuration of the NSX firewall.

» Example Usage

```
resource "nsxt_ip_set" "ip_set1" {
  description  = "IS provisioned by Terraform"
  display_name = "IS"

  tag {
    scope = "color"
    tag   = "blue"
  }

  ip_addresses = ["1.1.1.1", "2.2.2.2"]
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP set.

- `ip_addresses` - (Optional) IP addresses.

» Attributes Reference

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

- `id` - ID of the IP set.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IP set can be imported into this resource, via the following command:

```
terraform import nsxt_ip_set.ip_set1 UUID
```

The above command imports the IP set named `ip_set1` with the NSX id `UUID`.

» nsxt_lb_cookie_persistence_profile

Provides a resource to configure lb cookie persistence profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_cookie_persistence_profile" "lb_cookie_persistence_profile" {
  description        = "lb_cookie_persistence_profile provisioned by Terraform"
  display_name       = "lb_cookie_persistence_profile"
  cookie_name        = "my_cookie"
  persistence_shared = "false"
  cookie_fallback    = "false"
  cookie_garble      = "false"
  cookie_mode        = "INSERT"

  insert_mode_params {
    cookie_domain      = ".example2.com"
    cookie_path        = "/subfolder"
    cookie_expiry_type = "SESSION_COOKIE_TIME"
    max_idle_time      = "1000"
    max_life_time      = "2000"
  }
  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `cookie_mode` - (Optional) The cookie persistence mode. Accepted values: PREFIX, REWRITE and INSERT which is the default.
- `cookie_name` - (Required) cookie name.
- `persistence_shared` - (Optional) A boolean flag which reflects whether the cookie persistence is private or shared. When false (which is the default value), the cookie persistence is private to each virtual server and is qualified by the pool. If set to true, in cookie insert mode, cookie persistence could be shared across multiple virtual servers that are bound to the same pools.
- `cookie_fallback` - (Optional) A boolean flag which reflects whether once the server points by this cookie is down, a new server is selected, or the requests will be rejected.
- `cookie_garble` - (Optional) A boolean flag which reflects whether the cookie value (server IP and port) would be encrypted or in plain text.
- `insert_mode_params` - (Optional) Additional parameters for the INSERT cookie mode:
  - `cookie_domain` - (Optional) HTTP cookie domain (for INSERT mode only).
  - `cookie_path` - (Optional) HTTP cookie path (for INSERT mode only).
  - `cookie_expiry_type` - (Optional) Type of cookie expiration timing (for INSERT mode only). Accepted values: SESSION_COOKIE_TIME for session cookie time setting and PERSISTENCE_COOKIE_TIME for persistence cookie time setting.
  - `max_idle_time` - (Required if cookie_expiry_type is set) Maximum interval the cookie is valid for from the last time it was seen in a request.
  - `max_life_time` - (Required for INSERT mode with SESSION_COOKIE_TIME expiration) Maximum interval the cookie is valid for from the first time the cookie was seen in a request.

- `tag` - (Optional) A list of scope + tag pairs to associate with this lb cookie persistence profile.

» Attributes Reference

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

- `id` - ID of the lb cookie persistence profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb cookie persistence profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_cookie_persistence_profile.lb_cookie_persistence_profile UUID
```

The above would import the lb cookie persistence profile named `lb_cookie_persistence_profile` with the nsx id `UUID`

» nsxt_lb_source_ip_persistence_profile

Provides a resource to configure lb source ip persistence profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_source_ip_persistence_profile" "lb_source_ip_persistence_profile" {
  description = "lb_source_ip_persistence_profile provisioned by Terraform"
  display_name = "lb_source_ip_persistence_profile"
  persistence_shared       = "true"
  ha_persistence_mirroring = "true"
  purge_when_full          = "true"
  timeout                  = "100"

  tag {
    scope = "color"
    tag   = "red"
  }

}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb source ip persistence profile.
- `persistence_shared` - (Optional) A boolean flag which reflects whether the cookie persistence is private or shared.
- `ha_persistence_mirroring` - (Optional) A boolean flag which reflects whether persistence entries will be synchronized to the HA peer.
- `timeout` - (Optional) Persistence expiration time in seconds, counted from the time all the connections are completed. Defaults to 300 seconds.
- `purge_when_full` - (Optional) A boolean flag which reflects whether entries will be purged when the persistence table is full. Defaults to true.

» Attributes Reference

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

- `id` - ID of the lb source ip persistence profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb source ip persistence profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_source_ip_persistence_profile.lb_source_ip_persistence_profile UUID
```

The above would import the lb source ip persistence profile named `lb_source_ip_persistence_profile` with the nsx id `UUID`

» nsxt_lb_pool

Provides a resource to configure lb pool on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_icmp_monitor" "lb_icmp_monitor" {
  display_name = "lb_icmp_monitor"
  fall_count   = 3
  interval     = 5
}

resource "nsxt_lb_passive_monitor" "lb_passive_monitor" {
  display_name = "lb_passive_monitor"
  max_fails    = 3
  timeout      = 10
}

resource "nsxt_lb_pool" "lb_pool" {
  description              = "lb_pool provisioned by Terraform"
  display_name             = "lb_pool"
  algorithm                = "WEIGHTED_ROUND_ROBIN"
  min_active_members       = 1
  tcp_multiplexing_enabled = false
  tcp_multiplexing_number  = 3
  active_monitor_id        = "${nsxt_lb_icmp_monitor.lb_icmp_monitor.id}"
  passive_monitor_id       = "${nsxt_lb_passive_monitor.lb_passive_monitor.id}"

  member {
    admin_state                = "ENABLED"
    backup_member              = "false"
    display_name               = "1st-member"
    ip_address                 = "1.1.1.1"
    max_concurrent_connections = "1"
    port                       = "87"
    weight                     = "1"
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}

resource "nsxt_lb_pool" "lb_pool_with_dynamic_membership" {
  description              = "lb_pool provisioned by Terraform"
  display_name             = "dynamic_lb_pool"
  algorithm                = "LEAST_CONNECTION"
  min_active_members       = 1
  tcp_multiplexing_enabled = false
  tcp_multiplexing_number  = 3
  active_monitor_id        = "${nsxt_lb_icmp_monitor.lb_icmp_monitor.id}"
  passive_monitor_id       = "${nsxt_lb_passive_monitor.lb_passive_monitor.id}"

  snat_translation {
    type          = "SNAT_IP_POOL"
    ip            = "1.1.1.1"
  }

  member_group {
    ip_version_filter   = "IPV4"
    limit_ip_list_size  = true
    max_ip_list_size    = "4"
    port                = "80"

    grouping_object {
      target_type = "NSGroup"
      target_id   = "${nsxt_ns_group.group1.id}"
      }
    }
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `active_monitor_id` - (Optional) Active health monitor Id. If one is not set, the active healthchecks will be disabled.
- `algorithm` - (Optional) Load balancing algorithm controls how the incoming connections are distributed among the members. Supported algorithms are: ROUND_ROBIN, WEIGHTED_ROUND_ROBIN, LEAST_CONNECTION, WEIGHTED_LEAST_CONNECTION, IP_HASH.
- `member` - (Optional) Server pool consists of one or more pool members. Each pool member is identified, typically, by an IP address and a port.

Each member has the following arguments:

  - admin_state - (Optional) Pool member admin state. Possible values: ENABLED, DISABLED and GRACEFUL_DISABLED
  - backup_member - (Optional) A boolean flag which reflects whether this is a backup pool member. Backup servers are typically configured with a sorry page indicating to the user that the application is currently unavailable. While the pool is active (a specified minimum number of pool members are active) BACKUP members are skipped during server selection. When the pool is inactive, incoming connections are sent to only the BACKUP member(s).
  - display_name - (Optional) The display name of this resource. pool member name.
  - ip_address - (Required) Pool member IP address.
  - max_concurrent_connections - (Optional) To ensure members are not overloaded, connections to a member can be capped by the load balancer. When a member reaches this limit, it is skipped during server selection. If it is not specified, it means that connections are unlimited.
  - port - (Optional) If port is specified, all connections will be sent to this port. Only single port is supported. If unset, the same port the client connected to will be used, it could be overrode by default_pool_member_port setting in virtual server. The port should not specified for port range case.
  - weight - (Optional) Pool member weight is used for WEIGHTED_ROUND_ROBIN balancing algorithm. The weight value would be ignored in other algorithms.

- member_group - (Optional) Dynamic pool members for the loadbalancing pool. When member group is defined, members setting should not be specified. The member_group has the following arguments:
  - grouping_object - (Required) Grouping object of type NSGroup which will be used as dynamic pool members. The IP list of the grouping object would be used as pool member IP setting.
  - ip_version_filter - (Optional) Ip version filter is used to filter IPv4 or IPv6 addresses from the grouping object. If the filter is not specified, both IPv4 and IPv6 addresses would be used as server IPs. Supported filtering is ”IPV4” and ”IPV6” (”IPV4” is the default one)
  - limit_ip_list_size - (Optional) Limits the max number of pool members. If false, allows the dynamic pool to grow up to the load balancer max pool member capacity.
  - max_ip_list_size - (Optional) Should only be specified if limit_ip_list_size is set to true. Limits the max number of pool members to the specified value.
  - port - (Optional) If port is specified, all connections will be sent to this port. If unset, the same port the client connected to will be used, it could be overridden by default_pool_member_ports setting in virtual server. The port should not specified for multiple ports

case.

- `min_active_members` - (Optional) The minimum number of members for the pool to be considered active. This value is 1 by default.
- `passive_monitor_id` - (Optional) Passive health monitor Id. If one is not set, the passive healthchecks will be disabled.
- 'snat_translation - (Optional) SNAT translation configuration for the pool.
  - `type` - (Optional) Type of SNAT performed to ensure reverse traffic from the server can be received and processed by the loadbalancer. Supported types are: SNAT_AUTO_MAP, SNAT_IP_POOL and TRANSPARENT
  - `ip` - (Required for snat_translation of type SNAT_IP_POOL) Ip address or Ip range for SNAT of type SNAT_IP_POOL.
- `tcp_multiplexing_enabled` - (Optional) TCP multiplexing allows the same TCP connection between load balancer and the backend server to be used for sending multiple client requests from different client TCP connections. Disabled by default.
- `tcp_multiplexing_number` - (Optional) The maximum number of TCP connections per pool that are idly kept alive for sending future client requests. The default value for this is 6.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb pool.

» Attributes Reference

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

- `id` - ID of the lb pool.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb pool can be imported into this resource, via the following command:

```
terraform import nsxt_lb_pool.lb_pool UUID
```

The above would import the lb pool named `lb_pool` with the nsx id `UUID`

» nsxt_lb_http_virtual_server

Provides a resource to configure lb http or https virtual server on NSX-T manager

» Example Usage

```
resource "nsxt_lb_http_application_profile" "http_xff" {
  x_forwarded_for = "INSERT"
}

resource "nsxt_lb_cookie_persistence_profile" "session_persistence" {
  cookie_name = "SESSION"
}

resource "nsxt_lb_pool" "pool1" {
  algorithm = "LEAST_CONNECTION"
  member {
    ip_address = "3.0.0.1"
    port       = "443"
  }
  member {
    ip_address = "3.0.0.2"
    port       = "443"
  }
}

resource "nsxt_lb_pool" "sorry_pool" {
  member {
    ip_address = "3.0.0.15"
    port       = "443"
  }
}

resource "nsxt_lb_http_request_rewrite_rule" "redirect_post" {
  match_strategy = "ALL"
  method_condition {
    method = "POST"
  }

  uri_rewrite_action {
    uri = "/sorry_page.html"
  }
}

resource "nsxt_lb_client_ssl_profile" "ssl1" {
  prefer_server_ciphers = true
}

resource "nsxt_lb_server_ssl_profile" "ssl1" {
  session_cache_enabled = false
}

resource "nsxt_lb_http_virtual_server" "lb_virtual_server" {
  description                = "lb_virtual_server provisioned by terraform"
  display_name               = "virtual server 1"
  access_log_enabled         = true
  application_profile_id     = "${nsxt_lb_http_application_profile.http_xff.id}"
  enabled                    = true
  ip_address                 = "10.0.0.2"
  port                       = "443"
  default_pool_member_port   = "8888"
  max_concurrent_connections = 50
  max_new_connection_rate    = 20
  persistence_profile_id     = "${nsxt_lb_cookie_persistence_profile.session_persistence.id
  pool_id                    = "${nsxt_lb_pool.pool1.id}"
  sorry_pool_id              = "${nsxt_lb_pool.sorry_pool.id}"
  rule_ids                   = ["${nsxt_lb_http_request_rewrite_rule.redirect_post.id}"]

  client_ssl {
    client_ssl_profile_id   = "${nsxt_lb_client_ssl_profile.ssl1.id}"
    default_certificate_id  = "${data.nsxt_certificate.cert1.id}"
    certificate_chain_depth = 2
    client_auth             = true
    ca_ids                  = ["${data.nsxt_certificate.ca.id}"]
    crl_ids                 = ["${data.nsxt_certificate.crl.id}"]
    sni_certificate_ids     = ["${data.nsxt_certificate.sni.id}"]
  }

  server_ssl {
    server_ssl_profile_id   = "${nsxt_lb_server_ssl_profile.ssl1.id}"
    client_certificate_id   = "${data.nsxt_certificate.client.id}"
    certificate_chain_depth = 2
    server_auth             = true
    ca_ids                  = ["${data.nsxt_certificate.server_ca.id}"]
    crl_ids                 = ["${data.nsxt_certificate.crl.id}"]
  }

  tag {
    scope = "color"
    tag   = "green"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `enabled` - (Optional) Whether the virtual server is enabled. Default is true.
- `ip_address` - (Required) Virtual server IP address.
- `port` - (Required) Virtual server port.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb http virtual server.
- `access_log_enabled` - (Optional) Whether access log is enabled. Default is false.
- `application_profile_id` - (Required) The application profile defines the application protocol characteristics.
- `default_pool_member_port` - (Optional) Default pool member port.
- `max_concurrent_connections` - (Optional) To ensure one virtual server does not over consume resources, affecting other applications hosted on the same LBS, connections to a virtual server can be capped. If it is not specified, it means that connections are unlimited.
- `max_new_connection_rate` - (Optional) To ensure one virtual server does not over consume resources, connections to a member can be rate limited. If it is not specified, it means that connection rate is unlimited.
- `persistence_profile_id` - (Optional) Persistence profile is used to allow related client connections to be sent to the same backend server.
- `pool_id` - (Optional) Pool of backend servers. Server pool consists of one or more servers, also referred to as pool members, that are similarly configured and are running the same application.
- `sorry_pool_id` - (Optional) When load balancer can not select a backend server to serve the request in default pool or pool in rules, the request would be served by sorry server pool.
- `rule_ids` - (Optional) List of load balancer rules that provide customization of load balancing behavior using match/action rules.
- `client_ssl` - (Optional) Client side SSL customization.
  - `client_ssl_profile_id` - (Required) Id of client SSL profile that defines reusable properties.

  - `default_certificate_id` - (Required) Id of certificate that will be used if the server does not host multiple hostnames on the same IP address or if the client does not support SNI extension.
  - `certificate_chain_depth` - (Optional) Allowed depth of certificate chain. Default is 3.
  - `client_auth` - (Optional) Whether client authentication is mandatory. Default is false.
  - `ca_ids` - (Optional) List of CA certificate ids for client authentication.
  - `crl_ids` - (Optional) List of CRL certificate ids for client authentication.
  - `sni_certificate_ids` - (Optional) List of certificates to serve different hostnames.

- `server_ssl` - (Optional) Server side SSL customization.

  - `server_ssl_profile_id` - (Required) Id of server SSL profile that defines reusable properties.
  - `server_auth` - (Optional) Whether server authentication is needed. Default is False. If true, ca_ids should be provided.
  - `certificate_chain_depth` - (Optional) Allowed depth of certificate chain. Default is 3.
  - `client_certificate_id` - (Optional) Whether server authentication is required. Default is false.
  - `ca_ids` - (Optional) List of CA certificate ids for server authentication.
  - `crl_ids` - (Optional) List of CRL certificate ids for server authentication.

» Attributes Reference

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

- `id` - ID of the lb http virtual server.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb http virtual server can be imported into this resource, via the following command:

```
terraform import nsxt_lb_http_virtual_server.lb_http_virtual_server UUID
```

The above would import the lb http virtual server named `lb_http_virtual_server` with the nsx id `UUID`

» nsxt_lb_tcp_virtual_server

Provides a resource to configure lb tcp virtual server on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_fast_tcp_application_profile" "timeout_60" {
  close_timeout = 60
  idle_timeout  = 60
}

resource "nsxt_lb_source_ip_persistence_profile" "ip_profile" {
  display_name = "source1"
}

resource "nsxt_lb_pool" "pool1" {
  algorithm = "LEAST_CONNECTION"
  member {
    ip_address = "3.0.0.1"
    port       = "443"
  }
  member {
    ip_address = "3.0.0.2"
    port       = "443"
  }
}

resource "nsxt_lb_pool" "sorry_pool" {
  member {
    ip_address = "3.0.0.15"
    port       = "443"
  }
}

resource "nsxt_lb_tcp_virtual_server" "lb_virtual_server" {
  description                = "lb_virtual_server provisioned by terraform"
  display_name               = "virtual server 1"
  access_log_enabled         = true
  application_profile_id     = "${nsxt_lb_fast_tcp_application_profile.timeout_60.id}"
  enabled                    = true
  ip_address                 = "10.0.0.2"
  ports                      = ["443"]
  default_pool_member_ports  = ["8888"]
  max_concurrent_connections = 50
  max_new_connection_rate    = 20
  persistence_profile_id     = "${nsxt_lb_source_ip_persistence_profile.ip_profile.id}"
  pool_id                    = "${nsxt_lb_pool.pool1.id}"
  sorry_pool_id              = "${nsxt_lb_pool.sorry_pool.id}"

  tag {
    scope = "color"
    tag   = "green"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `enabled` - (Optional) Whether the virtual server is enabled. Default is true.
- `ip_address` - (Required) Virtual server IP address.
- `ports` - (Required) List of virtual server ports.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb tcp virtual server.
- `access_log_enabled` - (Optional) Whether access log is enabled. Default is false.
- `application_profile_id` - (Required) The application profile defines the application protocol characteristics.
- `default_pool_member_ports` - (Optional) List of default pool member ports.
- `max_concurrent_connections` - (Optional) To ensure one virtual server does not over consume resources, affecting other applications hosted on the same LBS, connections to a virtual server can be capped. If it is not specified, it means that connections are unlimited.
- `max_new_connection_rate` - (Optional) To ensure one virtual server does not over consume resources, connections to a member can be rate limited. If it is not specified, it means that connection rate is unlimited.
- `persistence_profile_id` - (Optional) Persistence profile is used to allow related client connections to be sent to the same backend server. Only source ip persistance profile is accepted.
- `pool_id` - (Optional) Pool of backend servers. Server pool consists of one or more servers, also referred to as pool members, that are similarly configured and are running the same application.
- `sorry_pool_id` - (Optional) When load balancer can not select a backend

server to serve the request in default pool or pool in rules, the request would be served by sorry server pool.

» Attributes Reference

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

- `id` - ID of the lb tcp virtual server.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb tcp virtual server can be imported into this resource, via the following command:

```
terraform import nsxt_lb_tcp_virtual_server.lb_tcp_virtual_server UUID
```

The above would import the lb tcp virtual server named `lb_tcp_virtual_server` with the nsx id `UUID`

» nsxt_lb_udp_virtual_server

Provides a resource to configure lb udp virtual server on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_fast_udp_application_profile" "timeout_60" {
  idle_timeout  = 60
}

resource "nsxt_lb_source_ip_persistence_profile" "ip_profile" {
  display_name = "source1"
}

resource "nsxt_lb_pool" "pool1" {
  algorithm = "LEAST_CONNECTION"
  member {
    ip_address = "3.0.0.1"
    port       = "443"
  }
  member {
    ip_address = "3.0.0.2"
    port       = "443"
  }
}

resource "nsxt_lb_pool" "sorry_pool" {
  member {
    ip_address = "3.0.0.15"
    port       = "443"
  }
}

resource "nsxt_lb_udp_virtual_server" "lb_virtual_server" {
  description                = "lb_virtual_server provisioned by terraform"
  display_name               = "virtual server 1"
  access_log_enabled         = true
  application_profile_id     = "${nsxt_lb_fast_udp_application_profile.timeout_60.id}"
  enabled                    = true
  ip_address                 = "10.0.0.2"
  ports                      = ["443"]
  default_pool_member_ports  = ["8888"]
  max_concurrent_connections = 50
  max_new_connection_rate    = 20
  persistence_profile_id     = "${nsxt_lb_source_ip_persistence_profile.ip_profile.id}"
  pool_id                    = "${nsxt_lb_pool.pool1.id}"
  sorry_pool_id              = "${nsxt_lb_pool.sorry_pool.id}"

  tag {
    scope = "color"
    tag   = "green"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `enabled` - (Optional) Whether the virtual server is enabled. Default is true.
- `ip_address` - (Required) Virtual server IP address.
- `ports` - (Required) List of virtual server port.

- `tag` - (Optional) A list of scope + tag pairs to associate with this lb udp virtual server.
- `access_log_enabled` - (Optional) Whether access log is enabled. Default is false.
- `application_profile_id` - (Required) The application profile defines the application protocol characteristics.
- `default_pool_member_ports` - (Optional) List of default pool member ports.
- `max_concurrent_connections` - (Optional) To ensure one virtual server does not over consume resources, affecting other applications hosted on the same LBS, connections to a virtual server can be capped. If it is not specified, it means that connections are unlimited.
- `max_new_connection_rate` - (Optional) To ensure one virtual server does not over consume resources, connections to a member can be rate limited. If it is not specified, it means that connection rate is unlimited.
- `persistence_profile_id` - (Optional) Persistence profile is used to allow related client connections to be sent to the same backend server. Only source ip persistence profile is accepted.
- `pool_id` - (Optional) Pool of backend servers. Server pool consists of one or more servers, also referred to as pool members, that are similarly configured and are running the same application.
- `sorry_pool_id` - (Optional) When load balancer can not select a backend server to serve the request in default pool or pool in rules, the request would be served by sorry server pool.

» Attributes Reference

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

- `id` - ID of the lb udp virtual server.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb udp virtual server can be imported into this resource, via the following command:

```
terraform import nsxt_lb_udp_virtual_server.lb_udp_virtual_server UUID
```

The above would import the lb udp virtual server named `lb_udp_virtual_server` with the nsx id `UUID`

» nsxt_lb_service

Provides a resource to configure lb service on NSX-T manager. Note that lb service needs to be attached to Tier-1 router that satisfies following preconditions: * It needs to reside on edge cluster * It needs to be condigured with either uplink port or centralized service port

In order to enforce correct order of create/delete, it is recommended to add depends_on clause to lb service.

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
data "nsxt_edge_cluster" "EC" {
  display_name = "%s"
}

data "nsxt_logical_tier0_router" "test" {
  display_name = "%s"
}

resource "nsxt_logical_router_link_port_on_tier0" "test" {
  display_name      = "port_on_tier0"
  logical_router_id = "${data.nsxt_logical_tier0_router.test.id}"
}

resource "nsxt_logical_tier1_router" "test" {
  display_name    = "test"
  edge_cluster_id = "${data.nsxt_edge_cluster.EC.id}"
}

resource "nsxt_logical_router_link_port_on_tier1" "test" {
  logical_router_id             = "${nsxt_logical_tier1_router.test.id}"
  linked_logical_router_port_id = "${nsxt_logical_router_link_port_on_tier0.test.id}"
}

resource "nsxt_lb_service" "lb_service" {
  description  = "lb_service provisioned by Terraform"
  display_name = "lb_service"

  tag {
    scope = "color"
    tag   = "red"
  }
  enabled           = true
  logical_router_id = "${nsxt_logical_tier1_router.test.id}"
  error_log_level   = "INFO"
  size              = "MEDIUM"

  depends_on        = ["nsxt_logical_router_link_port_on_tier1.test"]
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb service.
- `logical_router_id` - (Required) Tier1 logical router this service is attached to. Note that this router needs to have edge cluster configured, and have an uplink port or CSP (centralized service port).
- `enabled` - (Optional) whether the load balancer service is enabled.
- `error_log_level` - (Optional) Load balancer engine writes information about encountered issues of different severity levels to the error log. This setting is used to define the severity level of the error log.
- `size` - (Required) Size of load balancer service. Accepted values are SMALL/MEDIUM/LARGE.
- `virtual_server_ids` - (Optional) Virtual servers associated with this Load Balancer.

» Attributes Reference

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

- `id` - ID of the lb_service.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb service can be imported into this resource, via the following command:

```
terraform import nsxt_lb_service.lb_service UUID
```

The above would import the lb service named `lb_service` with the nsx id `UUID`

» nsxt_lb_http_forwarding_rule

Provides a resource to configure lb http forwarding rule on NSX-T manager. This rule will be executed when HTTP request message is forwarded by load balancer.

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usages

This example represents a superset of all possible action and conditions (and thus doesn't make much sense). More specific examples are provided below.

```
resource "nsxt_lb_http_forwarding_rule" "lb_rule" {
  description = "lb_rule provisioned by Terraform"
  display_name = "lb_rule"
  match_strategy = "ANY"

  tag {
    scope = "color"
    tag   = "red"
  }

  body_condition {
    value          = "XXX"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  header_condition {
    name       = "header1"
    value      = "bad"
    match_type = "EQUALS"
    inverse    = true
  }

  cookie_condition {
    name           = "name"
    value          = "cookie1"
    match_type     = "STARTS_WITH"
    case_sensitive = true
  }

  cookie_condition {
    name           = "name"
    value          = "cookie2"
    match_type     = "STARTS_WITH"
    case_sensitive = true
  }

  method_condition {
    method = "HEAD"
  }

  version_condition {
    version = "HTTP_VERSION_1_0"
    inverse = true
  }

  uri_condition {
    uri        = "/index.html"
    match_type = "EQUALS"
  }

  ip_condition {
    source_address = "1.1.1.1"
  }

  tcp_condition {
    source_port = 7887
  }

  http_reject_action {
    reply_status  = "500"
    reply_message = "rejected"
  }

  http_redirect_action {
    redirect_status = "200"
    redirect_url    = "/abc.com"
  }

  select_pool_action {
    pool_id  = "${nsxt_lb_pool.pool.id}"
  }
}
```

The following rule will match if header X-FORWARDED-FOR does not start with ”192.168”, request method is GET and URI contains ”books”:

```
resource "nsxt_lb_http_forwarding_rule" "lb_rule1" {
  match_strategy = "ALL"
  header_condition {
    name       = "X-FORWARDED-FOR"
    value      = "192.168"
    match_type = "STARTS_WITH"
    inverse    = true
  }

  method_condition {
    method = "GET"
  }

  uri_condition {
    uri        = "books"
    match_type = "CONTAINS"
  }

  http_reject_action {
    reply_status  = "500"
    reply_message = "rejected"
  }
}
```

The following rule will match if header X-TEST contains ”apples” or ”pears”, regardless of the case:

```
resource "nsxt_lb_http_forwarding_rule" "lb_rule1" {
  match_strategy = "ANY"

  header_condition {
    name           = "X-TEST"
    value          = "apples"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  header_condition {
    name           = "X-TEST"
    value          = "pears"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  select_pool_action {
    pool_id  = "${nsxt_lb_pool.pool.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb rule.
- `match_strategy` - (Required) Strategy to define how load balancer rule is considered a match when multiple match conditions are specified in one rule. If set to ALL, then load balancer rule is considered a match only if all the conditions match. If set to ANY, then load balancer rule is considered a match if any one of the conditions match.
- `body_condition` - (Optional) Set of match conditions used to match http request body:
  - `value` - (Required) The value to look for in the body.
  - `match_type` - (Required) Defines how value field is used to match the body of HTTP requests. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `header_condition` - (Optional) Set of match conditions used to match http request header:
  - `name` - (Required) The name of HTTP header to match.
  - `value` - (Required) The value of HTTP header to match.
  - `match_type` - (Required) Defines how value field is used to match the header value of HTTP requests. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX. Header name field does not support match types.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `cookie_condition` - (Optional) Set of match conditions used to match http request cookie:
  - `name` - (Required) The name of cookie to match.
  - `value` - (Required) The value of cookie to match.
  - `match_type` - (Required) Defines how value field is used to match the cookie. Accepted values are STARTS_WITH, ENDS_WITH,

CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `method_condition` - (Optional) Set of match conditions used to match http request method:
  - `method` - (Required) One of GET, HEAD, POST, PUT, OPTIONS.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `version_condition` - (Optional) Match condition used to match http version of the request:
  - `version` - (Required) One of HTTP_VERSION_1_0, HTTP_VERSION_1_1.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `ip_condition` - (Optional) Set of match conditions used to match IP header values of HTTP request:
  - `source_address` - (Required) The value source IP address to match.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `uri_condition` - (Optional) Set of match conditions used to match http request URI:
  - `uri` - (Required) The value of URI to match.
  - `match_type` - (Required) Defines how value field is used to match the URI. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `http_reject_action` - (At least one action is required) Set of http reject actions to be executed when load balancer rule matches:
  - `reply_status` - (Required) The HTTP reply status.
  - `reply_message` - (Required) The HTTP reply message.

- `http_redirect_action` - (At least one action is required) Set of http redirect actions to be executed when load balancer rule matches:
  - `redirect_status` - (Required) The HTTP reply status.
  - `redirect_url` - (Required) The URL to redirect to.

- `select_pool_action` - (At least one action is required) Set of pool selection actions to be executed when load balancer rule matches:
  - `pool_id` - (Required) The loadbalancer pool the request will be forwarded to.

» Attributes Reference

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

- `id` - ID of the lb rule.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb rule can be imported into this resource, via the following command: } }

```
terraform import nsxt_lb_http_forwarding_rule.lb_rule UUID
```

The above would import the lb rule named `lb_rule` with the nsx id `UUID`

» nsxt_lb_http_request_rewrite_rule

Provides a resource to configure lb http request rewrite rule on NSX-T manager. This rule will be executed when HTTP request message is received by load balancer.

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usages

This example represents a superset of all possible action and conditions (and thus doesn't make much sense). More specific examples are provided below.

```
resource "nsxt_lb_http_request_rewrite_rule" "lb_rule" {
  description = "lb_rule provisioned by Terraform"
  display_name = "lb_rule"
  match_strategy = "ANY"

  tag {
    scope = "color"
    tag   = "red"
```

```
}

body_condition {
  value      = "XXX"
  match_type = "CONTAINS"
  case_sensitive = false
}

header_condition {
  name       = "header1"
  value      = "bad"
  match_type = "EQUALS"
  inverse    = true
}

cookie_condition {
  name           = "name"
  value          = "cookie1"
  match_type     = "STARTS_WITH"
  case_sensitive = true
}

cookie_condition {
  name           = "name"
  value          = "cookie2"
  match_type     = "STARTS_WITH"
  case_sensitive = true
}

method_condition {
  method = "HEAD"
}

version_condition {
  version = "HTTP_VERSION_1_0"
  inverse = true
}

uri_condition {
  uri        = "/index.html"
  match_type = "EQUALS"
}

uri_arguments_condition {
  uri_arguments = "delete"
  match_type    = "CONTAINS"
```

```
    inverse       = true
  }

  ip_condition {
    source_address = "1.1.1.1"
  }

  tcp_condition {
    source_port = 7887
  }

  header_rewrite_action {
    name  = "header1"
    value = "value2"
  }

  uri_rewrite_action {
    uri           = "new.html"
    uri_arguments = "redirect=true"
  }
}
```

The following rule will match if header X-FORWARDED-FOR does not start with ”192.168”, request method is GET and URI contains ”books”:

```
resource "nsxt_lb_http_request_rewrite_rule" "lb_rule1" {
  match_strategy = "ALL"

  header_condition {
    name       = "X-FORWARDED-FOR"
    value      = "192.168"
    match_type = "STARTS_WITH"
    inverse    = true
  }

  method_condition {
    method = "GET"
  }

  uri_condition {
    uri        = "books"
    match_type = "CONTAINS"
  }

  header_rewrite_action {
    name  = "header1"
    value = "value2"
  }
}
```

The following rule will match if header X-TEST contains ”apples” or ”pears”, regardless of the case:

```
resource "nsxt_lb_http_request_rewrite_rule" "lb_rule1" {
  match_strategy = "ANY"

  header_condition {
    name           = "X-TEST"
    value          = "apples"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  header_condition {
    name           = "X-TEST"
    value          = "pears"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  header_rewrite_action {
    name  = "header1"
    value = "value2"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb rule.
- `match_strategy` - (Required) Strategy to define how load balancer rule is considered a match when multiple match conditions are specified in one rule. If set to ALL, then load balancer rule is considered a match only if all the conditions match. If set to ANY, then load balancer rule is considered a match if any one of the conditions match.
- `body_condition` - (Optional) Set of match conditions used to match http request body:

  - `value` - (Required) The value to look for in the body.
  - `match_type` - (Required) Defines how value field is used to match the body of HTTP requests. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `header_condition` - (Optional) Set of match conditions used to match http request header:

  - `name` - (Required) The name of HTTP header to match.
  - `value` - (Required) The value of HTTP header to match.
  - `match_type` - (Required) Defines how value field is used to match the header value of HTTP requests. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX. Header name field does not support match types.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `cookie_condition` - (Optional) Set of match conditions used to match http request cookie:

  - `name` - (Required) The name of cookie to match.
  - `value` - (Required) The value of cookie to match.
  - `match_type` - (Required) Defines how value field is used to match the cookie. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `method_condition` - (Optional) Set of match conditions used to match http request method:

  - `method` - (Required) One of GET, HEAD, POST, PUT, OPTIONS.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `version_condition` - (Optional) Match condition used to match http version of the request:

  - `version` - (Required) One of HTTP_VERSION_1_0, HTTP_VERSION_1_1.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `uri_condition` - (Optional) Set of match conditions used to match http request URI:
  - `uri` - (Required) The value of URI to match.
  - `match_type` - (Required) Defines how value field is used to match the URI. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `uri_arguments_condition` - (Optional) Set of match conditions used to match http request URI arguments (query string):
  - `uri_arguments` - (Required) Query string of URI, typically contains key value pairs.
  - `match_type` - (Required) Defines how value field is used to match the URI. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `ip_condition` - (Optional) Set of match conditions used to match IP header values of HTTP request:
  - `source_address` - (Required) The value source IP address to match.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `header_rewrite_action` - (At least one action is required) Set of header rewrite actions to be executed when load balancer rule matches:
  - `name` - (Required) The name of HTTP header to be rewritten.
  - `value` - (Required) The new value of HTTP header.
- `uri_rewrite_action` - (At least one action is required) Set of URI rewrite actions to be executed when load balancer rule matches:
  - `uri` - (Required) The new URI for the HTTP request.
  - `uri_arguments` - (Required) The new URI arguments(query string) for the HTTP request.

» Attributes Reference

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

- `id` - ID of the lb rule.

- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb rule can be imported into this resource, via the following command: } }

```
terraform import nsxt_lb_http_request_rewrite_rule.lb_rule UUID
```

The above would import the lb rule named `lb_rule` with the nsx id `UUID`

» nsxt_lb_http_response_rewrite_rule

Provides a resource to configure lb http response rewrite rule on NSX-T manager. This rule will be executed when HTTP response message is received by load balancer.

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usages

This example represents a superset of all possible conditions (and thus doesn't make much sense). More specific examples are provided below.

```
resource "nsxt_lb_http_response_rewrite_rule" "lb_rule" {
  description = "lb_rule provisioned by Terraform"
  display_name = "lb_rule"
  match_strategy = "ALL"

  tag {
    scope = "color"
    tag   = "blue"
  }

  request_header_condition {
    name       = "header1"
    value      = "bad"
    match_type = "EQUALS"
    inverse    = true
  }

  response_header_condition {
    name       = "header1"
    value      = "good"
    match_type = "EQUALS"
    inverse    = false
  }

  cookie_condition {
    name           = "name1"
    value          = "cookie1"
    match_type     = "STARTS_WITH"
    case_sensitive = true
  }

  cookie_condition {
    name           = "name2"
    value          = "cookie2"
    match_type     = "STARTS_WITH"
    case_sensitive = true
  }

  method_condition {
    method = "HEAD"
  }

  version_condition {
    version = "HTTP_VERSION_1_1"
    inverse = true
  }

  uri_condition {
    uri        = "/index.html"
    match_type = "EQUALS"
  }

  uri_arguments_condition {
    uri_arguments = "delete"
    match_type    = "CONTAINS"
    inverse       = true
  }

  ip_condition {
    source_address = "1.1.1.1"
  }

  tcp_condition {
    source_port = 7887
  }
```

```
  header_rewrite_action {
    name  = "header1"
    value = "even better"
  }
}
```

The following rule will match if request header X-FORWARDED-FOR does not start with ”192.168”, request method is GET and response content is json:

```
resource "nsxt_lb_http_response_rewrite_rule" "lb_rule1" {
  match_strategy = "ALL"

  request_header_condition {
    name       = "X-FORWARDED-FOR"
    value      = "192.168"
    match_type = "STARTS_WITH"
    inverse    = true
  }

  response_header_condition {
    name       = "Content-Type"
    value      = "/json"
    match_type = "CONTAINS"
    inverse    = false
  }

  method_condition {
    method = "GET"
  }

  header_rewrite_action {
    name  = "header1"
    value = "value2"
  }
}
```

The following rule will match if response header X-TEST contains ”apples” or ”pears”, regardless of the case:

```
resource "nsxt_lb_http_response_rewrite_rule" "lb_rule1" {
  match_strategy = "ANY"

  response_header_condition {
    name           = "X-TEST"
    value          = "apples"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  response_header_condition {
    name           = "X-TEST"
    value          = "pears"
    match_type     = "CONTAINS"
    case_sensitive = false
  }

  header_rewrite_action {
    name  = "header1"
    value = "value2"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb rule.
- `match_strategy` - (Required) Strategy to define how load balancer rule is considered a match when multiple match conditions are specified in one rule. If set to ALL, then load balancer rule is considered a match only if all the conditions match. If set to ANY, then load balancer rule is considered a match if any one of the conditions match.
- `request_header_condition` - (Optional) Set of match conditions used to match http request header:
  - `name` - (Required) The name of HTTP header to match.
  - `value` - (Required) The value of HTTP header to match.
  - `match_type` - (Required) Defines how value field is used to match the header value of HTTP request. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX. Header name field does not support match types.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `response_header_condition` - (Optional) Set of match conditions used to match http response header:

  - name - (Required) The name of HTTP header to match.
  - value - (Required) The value of HTTP header to match.
  - match_type - (Required) Defines how value field is used to match the header value of HTTP response. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX. Header name field does not support match types.
  - case_sensitive - (Optional) If true, case is significant in the match. Default is true.
  - inverse - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- cookie_condition - (Optional) Set of match conditions used to match http request cookie:

  - name - (Required) The name of cookie to match.
  - value - (Required) The value of cookie to match.
  - match_type - (Required) Defines how value field is used to match the cookie. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - case_sensitive - (Optional) If true, case is significant in the match. Default is true.
  - inverse - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- method_condition - (Optional) Set of match conditions used to match http request method:

  - method - (Required) One of GET, HEAD, POST, PUT, OPTIONS.
  - inverse - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- version_condition - (Optional) Match condition used to match http version of the request:

  - version - (Required) One of HTTP_VERSION_1_0, HTTP_VERSION_1_1.
  - inverse - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- uri_condition - (Optional) Set of match conditions used to match http request URI:

  - uri - (Required) The value of URI to match.
  - match_type - (Required) Defines how value field is used to match the URI. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - case_sensitive - (Optional) If true, case is significant in the match. Default is true.
  - inverse - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.

- `uri_arguments_condition` - (Optional) Set of match conditions used to match http request URI arguments (query string):
  - `uri_arguments` - (Required) Query string of URI, typically contains key value pairs.
  - `match_type` - (Required) Defines how value field is used to match the URI. Accepted values are STARTS_WITH, ENDS_WITH, CONTAINS, EQUALS, REGEX.
  - `case_sensitive` - (Optional) If true, case is significant in the match. Default is true.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `ip_condition` - (Optional) Set of match conditions used to match IP header values of HTTP message:
  - `source_address` - (Required) The value source IP address to match.
  - `inverse` - (Optional) A flag to indicate whether reverse the match result of this condition. Default is false.
- `header_rewrite_action` - (Required) Set of header rewrite actions to be executed on the outgoing response when load balancer rule matches:
  - `name` - (Required) The name of HTTP header to be rewritten.
  - `value` - (Required) The new value of HTTP header.

» Attributes Reference

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

- `id` - ID of the lb rule.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb rule can be imported into this resource, via the following command: } }

```
terraform import nsxt_lb_http_response_rewrite_rule.lb_rule UUID
```

The above would import the lb rule named `lb_rule` with the nsx id `UUID`

» nsxt_lb_client_ssl_profile

Provides a resource to configure lb client ssl profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_client_ssl_profile" "lb_client_ssl_profile" {
  description           = "lb_client_ssl_profile provisioned by Terraform"
  display_name          = "lb_client_ssl_profile"
  protocols             = ["TLS_V1_2"]
  ciphers               = ["TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256", "TLS_ECDHE_RSA_WITH_AES_
  prefer_server_ciphers = true
  session_cache_enabled = true
  session_cache_timeout = 200

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb client ssl profile.
- `prefer_server_ciphers` - (Optional) During SSL handshake as part of the SSL client Hello client sends an ordered list of ciphers that it can support (or prefers) and typically server selects the first one from the top of that list it can also support. For Perfect Forward Secrecy(PFS), server could override the client's preference. Defaults to false.
- `ciphers` - (Optional) supported SSL cipher list to client side. The supported ciphers can contain: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_CB
  TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_256_C
  TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_S
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_RSA_WITH_AES_128_CBC_SHA25
  TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA256, TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SH
  TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_12

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_128_CBC
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDH_ECDSA_WITH_AES_256_
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDH_RSA_WITH_AES_128_CB
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDH_RSA_WITH_AES_128_GCM_
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDH_RSA_WITH_AES_256_GCM_

- `prefer_server_ciphers` - (Optional) During SSL handshake as part of the SSL client Hello client sends an ordered list of ciphers that it can support (or prefers) and typically server selects the first one from the top of that list it can also support. For Perfect Forward Secrecy(PFS), server could override the client's preference. Defaults to false.
- `protocols` - (Optional) SSL versions TLS_V1_1 and TLS_V1_2 are supported and enabled by default. SSL_V2, SSL_V3, and TLS_V1 are supported, but disabled by default.
- `session_cache_enabled` - (Optional) SSL session caching allows SSL client and server to reuse previously negotiated security parameters avoiding the expensive public key operation during handshake. Defaults to true.
- `session_cache_timeout` - (Optional) Session cache timeout specifies how long the SSL session parameters are held on to and can be reused. Default value is 300.

» Attributes Reference

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

- `id` - ID of the lb client ssl profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `is_secure` - This flag is set to true when all the ciphers and protocols are secure. It is set to false when one of the ciphers or protocols is insecure.

» Importing

An existing lb client ssl profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_client_ssl_profile.lb_client_ssl_profile UUID
```

The above would import the lb client ssl profile named `lb_client_ssl_profile` with the nsx id `UUID`

» nsxt_lb_server_ssl_profile

Provides a resource to configure lb server ssl profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_server_ssl_profile" "lb_server_ssl_profile" {
  description           = "lb_server_ssl_profile provisioned by Terraform"
  display_name          = "lb_server_ssl_profile"
  protocols             = ["TLS_V1_2"]
  ciphers               = ["TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256", "TLS_ECDHE_RSA_WITH_AES_
  session_cache_enabled = true

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb server ssl profile.
- `ciphers` - (Optional) supported SSL cipher list to client side. The supported ciphers can contain: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
  TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_CB
  TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_256_CB
  TLS_ECDH_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_CBC_SHA,
  TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA,
  TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_S
  TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384, TLS_RSA_WITH_AES_128_CBC_SHA256
  TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA256,
  TLS_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
  TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128
  TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384, TLS_ECDHE_ECDSA_WITH_AES_256
  TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDH_ECDSA_WITH_AES_128_CBC
  TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDH_ECDSA_WITH_AES_256_

TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDH_RSA_WITH_AES_128_CB
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDH_RSA_WITH_AES_128_GCM_
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384, TLS_ECDH_RSA_WITH_AES_256_GCM_

- `prefer_server_ciphers` - (Optional) During SSL handshake as part of the SSL client Hello client sends an ordered list of ciphers that it can support (or prefers) and typically server selects the first one from the top of that list it can also support. For Perfect Forward Secrecy(PFS), server could override the client's preference. Defaults to false.
- `protocols` - (Optional) SSL versions TLS_V1_1 and TLS_V1_2 are supported and enabled by default. SSL_V2, SSL_V3, and TLS_V1 are supported, but disabled by default.
- `session_cache_enabled` - (Optional) SSL session caching allows SSL server and server to reuse previously negotiated security parameters avoiding the expensive public key operation during handshake. Defaults to true.

» Attributes Reference

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

- `id` - ID of the lb server ssl profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `is_secure` - This flag is set to true when all the ciphers and protocols are secure. It is set to false when one of the ciphers or protocols is insecure.

» Importing

An existing lb server ssl profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_server_ssl_profile.lb_server_ssl_profile UUID
```

The above would import the lb server ssl profile named `lb_server_ssl_profile` with the nsx id `UUID`

» nsxt_lb_fast_tcp_application_profile

Provides a resource to configure LB fast TCP application profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_fast_tcp_application_profile" "lb_fast_tcp_profile" {
  description       = "lb_fast_tcp_application_profile provisioned by Terraform"
  display_name      = "lb_fast_tcp_application_profile"
  close_timeout     = "8"
  idle_timeout      = "1800"
  ha_flow_mirroring = "false"

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `close_timeout` - (Optional) Timeout in seconds to specify how long a closed TCP connection should be kept for this application before cleaning up the connection. Value can range between 1-60, with a default of 8 seconds.
- `idle_timeout` - (Optional) Timeout in seconds to specify how long an idle TCP connection in ESTABLISHED state should be kept for this application before cleaning up. The default value will be 1800 seconds
- `ha_flow_mirroring` - (Optional) A boolean flag which reflects whether flow mirroring is enabled, and all the flows to the bounded virtual server are mirrored to the standby node. By default this is disabled.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb fast tcp profile.

» Attributes Reference

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

- `id` - ID of the lb fast tcp profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb fast tcp profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_fast_tcp_application_profile.lb_fast_tcp_profile UUID
```

The above would import the LB fast TCP application profile named `lb_fast_tcp_profile` with the nsx id `UUID`

» nsxt_lb_fast_udp_application_profile

Provides a resource to configure LB fast UDP application profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_fast_udp_application_profile" "lb_fast_udp_profile" {
  description       = "lb_fast_udp_application_profile provisioned by Terraform"
  display_name      = "lb_fast_udp_application_profile"
  idle_timeout      = "1800"
  ha_flow_mirroring = "false"

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `idle_timeout` - (Optional) Timeout in seconds to specify how long an idle UDP connection in ESTABLISHED state should be kept for this application before cleaning up. The default value will be 300 seconds
- `ha_flow_mirroring` - (Optional) A boolean flag which reflects whether flow mirroring is enabled, and all the flows to the bounded virtual server are mirrored to the standby node. By default this is disabled.

- `tag` - (Optional) A list of scope + tag pairs to associate with this lb fast udp profile.

» Attributes Reference

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

- `id` - ID of the lb fast udp profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb fast udp profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_fast_udp_application_profile.lb_fast_udp_profile UUID
```

The above would import the LB fast UDP application profile named `lb_fast_udp_profile` with the nsx id `UUID`

» nsxt_lb_http_application_profile

Provides a resource to configure LB HTTP application profile on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_http_application_profile" "lb_http_application_profile" {
  description            = "lb_http_application_profile provisioned by Terraform"
  display_name           = "lb_http_application_profile"
  http_redirect_to       = "http://www.example.com"
  http_redirect_to_https = "false"
  idle_timeout           = "15"
  request_body_size      = "100"
  request_header_size    = "1024"
  response_timeout       = "60"
  x_forwarded_for        = "INSERT"
  ntlm                   = "true"

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `http_redirect_to` - (Optional) A URL that incoming requests for that virtual server can be temporarily redirected to, If a website is temporarily down or has moved. When set, http_redirect_to_https should be false.
- `http_redirect_to_https` - (Optional) A boolean flag which reflects whether the client will automatically be redirected to use SSL. When true, the http_redirect_to should not be specified.
- `idle_timeout` - (Optional) Timeout in seconds to specify how long an HTTP application can remain idle. Defaults to 15 seconds.
- `ntlm` - (Optional) A boolean flag which reflects whether NTLM challenge/response methodology will be used over HTTP. Can be set to true only if http_redirect_to_https is false.
- `request_body_size` - (Optional) Maximum request body size in bytes. If it is not specified, it means that request body size is unlimited.
- `request_header_size` - (Optional) Maximum request header size in bytes. Requests with larger header size will be processed as best effort whereas a request with header below this specified size is guaranteed to be processed. Defaults to 1024 bytes.
- `response_timeout` - (Optional) Number of seconds waiting for the server response before the connection is closed. Defaults to 60 seconds.
- `x_forwarded_for` - (Optional) When this value is set, the x_forwarded_for header in the incoming request will be inserted or replaced. Supported values are ”INSERT” and ”REPLACE”.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb http profile.

» Attributes Reference

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

- `id` - ID of the lb http application profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb http profile can be imported into this resource, via the following command:

```
terraform import nsxt_lb_http_application_profile.lb_http_application_profile UUID
```

The above would import the LB HTTP application profile named `lb_http_application_profile` with the nsx id `UUID`

» nsxt_logical_dhcp_server

Provides a resource to configure logical DHCP server on NSX-T manager

» Example Usage

```
data "nsxt_edge_cluster" "edgecluster" {
  display_name = "edgecluster1"
}

resource "nsxt_dhcp_server_profile" "serverprofile" {
  edge_cluster_id = "${data.nsxt_edge_cluster.edgecluster.id}"
}

resource "nsxt_logical_dhcp_server" "logical_dhcp_server" {
  display_name     = "logical_dhcp_server"
  description      = "logical_dhcp_server provisioned by Terraform"
  dhcp_profile_id  = "${nsxt_dhcp_server_profile.PRF.id}"
  dhcp_server_ip   = "1.1.1.10/24"
  gateway_ip       = "1.1.1.20"
  domain_name      = "abc.com"
  dns_name_servers = ["5.5.5.5"]

  dhcp_option_121 {
    network  = "6.6.6.0/24"
    next_hop = "1.1.1.21"
  }

  dhcp_generic_option {
    code = "119"
    values = ["abc"]
  }

  tag {
    scope = "color"
    tag   = "red"
  }

}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `dhcp_profile_id` - (Required) DHCP profile uuid.
- `dhcp_server_ip` - (Required) DHCP server IP in cidr format.
- `gateway_ip` - (Optional) Gateway IP.
- `domain_name` - (Optional) Domain name.
- `dns_name_servers` - (Optional) DNS IPs.
- `dhcp_option_121` - (Optional) DHCP classless static routes.
  - `network` - (Required) Destination in cidr format.
  - `next_hop` - (Required) IP address of next hop.
- `dhcp_generic_option` - (Optional) Generic DHCP options.
  - `code` - (Required) DHCP option code. Valid values are from 0 to 255.
  - `values` - (Required) List of DHCP option values.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical DHCP server.

» Attributes Reference

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

- `id` - ID of the logical DHCP server.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `attached_logical_port_id` - ID of the attached logical port.

» Importing

An existing logical DHCP server can be imported into this resource, via the following command:

```
terraform import nsxt_logical_dhcp_server.logical_dhcp_server UUID
```

The above would import the logical DHCP server named `logical_dhcp_server` with the nsx id `UUID`

» nsxt_dhcp_server_ip_pool

Provides a resource to configure IP Pool for logical DHCP server on NSX-T manager

» Example Usage

```
data "nsxt_edge_cluster" "edgecluster" {
  display_name = "edgecluster1"
}

resource "nsxt_dhcp_server_profile" "serverprofile" {
  edge_cluster_id = "${data.nsxt_edge_cluster.edgecluster.id}"
}

resource "nsxt_logical_dhcp_server" "logical_dhcp_server" {
  display_name     = "logical_dhcp_server"
  dhcp_profile_id  = "${nsxt_dhcp_server_profile.PRF.id}"
  dhcp_server_ip   = "1.1.1.10/24"
  gateway_ip       = "1.1.1.20"
}

resource "nsxt_dhcp_server_ip_pool" "dhcp_ip_pool" {
  display_name           = "ip pool"
  description            = "ip pool"
  logical_dhcp_server_id = "${nsxt_logical_dhcp_server.logical_dhcp_server.id}"
  gateway_ip             = "1.1.1.21"
  lease_time             = 1296000
  error_threshold        = 98
  warning_threshold      = 70

  ip_range {
    start = "1.1.1.40"
    end   = "1.1.1.60"
  }

  dhcp_option_121 {
    network  = "5.5.5.0/24"
    next_hop = "1.1.1.21"
  }
  dhcp_generic_option {
    code   = "119"
    values = ["abc"]
  }

  tag {
    scope = "color"
    tag   = "red"
  }

}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `logical_dhcp_server_id` - (Required) DHCP server uuid. Changing this would force new pool to be created.
- `gateway_ip` - (Optional) Gateway IP.
- `ip_range` - (Required) IP Ranges to be used within this pool.
  - `start` - (Required) IP address that indicates range start.
  - `end` - (Required) IP address that indicates range end.
- `lease_time` - (Optional) Lease time in seconds. Default is 86400.
- `error_threshold` - (Optional) Error threshold in percent. Valid values are from 80 to 100, default is 100.
- `warning_threshold` - (Optional) Warning threshold in percent. Valid values are from 50 to 80, default is 80.
- `dhcp_option_121` - (Optional) DHCP classless static routes. If specified, overrides DHCP server settings.
  - `network` - (Required) Destination in cidr format.
  - `next_hop` - (Required) IP address of next hop.
- `dhcp_generic_option` - (Optional) Generic DHCP options. If specified, overrides DHCP server settings.
  - `code` - (Required) DHCP option code. Valid values are from 0 to 255.
  - `values` - (Required) List of DHCP option values.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical DHCP server.

» Attributes Reference

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

- `id` - ID of the DHCP server IP pool.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing DHCP server IP Pool can be imported into this resource, via the following command:

```
terraform import nsxt_dhcp_server_ip_pool.ip_pool DHCP_SERVER_UUID POOL_UUID
```

The above would import the IP pool named `ip pool` for dhcp server with nsx ID `DHCP_SERVER_UUID` and pool nsx id `POOL_UUID`

» nsxt_logical_dhcp_port

This resource provides a resource to configure a logical port on a logical switch, and attach it to a DHCP server.

» Example Usage

```
resource "nsxt_logical_dhcp_server" "logical_dhcp_server" {
  display_name    = "logical_dhcp_server"
  dhcp_profile_id = "${nsxt_dhcp_server_profile.PRF.id}"
  dhcp_server_ip  = "1.1.1.10/24"
  gateway_ip      = "1.1.1.20"
}

resource "nsxt_logical_switch" "switch" {
  display_name      = "LS1"
  admin_state       = "UP"
  transport_zone_id = "${data.nsxt_transport_zone.transport_zone.id}"
}

resource "nsxt_logical_dhcp_port" "dhcp_port" {
  admin_state       = "UP"
  description       = "LP1 provisioned by Terraform"
  display_name      = "LP1"
  logical_switch_id = "${nsxt_logical_switch.switch.id}"
  dhcp_server_id    = "${nsxt_logical_dhcp_server.logical_dhcp_server.id}"
  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `logical_switch_id` - (Required) Logical switch ID for the logical port.
- `dhcp_server_id` - (Required) Logical DHCP server ID for the logical port.
- `admin_state` - (Optional) Admin state for the logical port. Accepted values - 'UP' or 'DOWN'. The default value is 'UP'.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical port.

» Attributes Reference

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

- `id` - ID of the logical DHCP port.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing DHCP Logical Port can be imported into this resource, via the following command:

```
terraform import nsxt_logical_dhcp_port.dhcp_port UUID
```

The above command imports the logical DHCP port named `dhcp_port` with the NSX id `UUID`.

» nsxt_logical_port

This resource provides a resource to configure a logical port on a logical switch in the NSX system. Like physical switches a logical switch can have one or more ports which can be connected to virtual machines or logical routers.

» Example Usage

```
resource "nsxt_logical_port" "logical_port" {
  admin_state       = "UP"
  description       = "LP1 provisioned by Terraform"
  display_name      = "LP1"
  logical_switch_id = "${nsxt_logical_switch.switch1.id}"

  tag {
    scope = "color"
    tag   = "blue"
  }

  switching_profile_id {
    key   = "${data.nsxt_switching_profile.qos_profile.resource_type}"
    value = "${data.nsxt_switching_profile.qos_profile.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `logical_switch_id` - (Required) Logical switch ID for the logical port.
- `admin_state` - (Optional) Admin state for the logical port. Accepted values - 'UP' or 'DOWN'. The default value is 'UP'.
- `switching_profile_id` - (Optional) List of IDs of switching profiles (of various types) to be associated with this switch. Default switching profiles will be used if not specified.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical port.

» Attributes Reference

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

- `id` - ID of the logical port.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing Logical Port can be imported into this resource, via the following command:

```
terraform import nsxt_logical_port.logical_port UUID
```

The above command imports the logical port named `logical_port` with the NSX id `UUID`.

» nsxt_logical_router_centralized_service_port

This resource provides a means to define a centralized service port on a logical router to connect a logical tier0 or tier1 router to a logical switch. This allows the router to be used for E-W load balancing

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_logical_router_centralized_service_port" "cs_port" {
  description                   = "Centralized service port provisioned by Terraform"
  display_name                  = "CSP1"
  logical_router_id             = "${nsxt_logical_tier1_router.rtr1.id}"
  linked_logical_switch_port_id = "${nsxt_logical_port.logical_port1.id}"
  ip_address                    = "1.1.0.1/24"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `logical_router_id` - (Required) Identifier for logical Tier-0 or Tier-1 router on which this port is created
- `linked_logical_switch_port_id` - (Required) Identifier for port on logical switch to connect to
- `ip_address` - (Required) Logical router port subnet (ip_address / prefix length)

- `urpf_mode` - (Optional) Unicast Reverse Path Forwarding mode. Accepted values are ”NONE” and ”STRICT” which is the default value.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this port.

» Attributes Reference

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

- `id` - ID of the logical router centralized service port.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing logical router centralized service port can be imported into this resource, via the following command:

```
terraform import nsxt_logical_router_centralized_service_port.cs_port UUID
```

The above command imports the logical router centralized service port named `cs_port` with the NSX id `UUID`.

» nsxt_logical_router_downlink_port

This resource provides a means to define a downlink port on a logical router to connect a logical tier1 router to a logical switch. The result of this is to provide a default gateway to virtual machines running on the logical switch.

» Example Usage

```
resource "nsxt_logical_router_downlink_port" "downlink_port" {
  description                   = "DP1 provisioned by Terraform"
  display_name                  = "DP1"
  logical_router_id             = "${nsxt_logical_tier1_router.rtr1.id}"
  linked_logical_switch_port_id = "${nsxt_logical_port.logical_port1.id}"
  ip_address                    = "1.1.0.1/24"

  service_binding {
    target_id   = "${nsxt_dhcp_relay_service.dr_service.id}"
    target_type = "LogicalService"
  }
  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `logical_router_id` - (Required) Identifier for logical Tier-1 router on which this port is created
- `linked_logical_switch_port_id` - (Required) Identifier for port on logical switch to connect to
- `ip_address` - (Required) Logical router port subnet (ip_address / prefix length)
- `urpf_mode` - (Optional) Unicast Reverse Path Forwarding mode. Accepted values are ”NONE” and ”STRICT” which is the default value.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this port.
- `service_binding` - (Optional) A list of services for this port. Currently only ”LogicalService” is supported as a target_type, and a DHCP relay service ID as target_id

» Attributes Reference

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

- `id` - ID of the logical router downlink port.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `mac_address` - The MAC address assigned to this port

» Importing

An existing logical router downlink port can be imported into this resource, via the following command:

```
terraform import nsxt_logical_router_downlink_port.downlink_port UUID
```

The above command imports the logical router downlink port named `downlink_port` with the NSX id `UUID`.

» nsxt_logical_router_link_port_on_tier0

This resource provides the ability to configure a logical router link port on a tier 0 logical router. This port can then be used to connect the tier 0 logical router to another logical router.

» Example Usage

```
resource "nsxt_logical_router_link_port_on_tier0" "link_port_tier0" {
  description       = "TIER0_PORT1 provisioned by Terraform"
  display_name      = "TIER0_PORT1"
  logical_router_id = "${data.nsxt_logical_tier0_router.rtr1.id}"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `logical_router_id` - (Required) Identifier for logical Tier0 router on which this port is created.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this port.

» Attributes Reference

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

- `id` - ID of the logical router link port.
- `linked_logical_switch_port_id` - Identifier for port on logical router to connect to.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing logical router link port on Tier-0 can be imported into this resource, via the following command:

```
terraform import nsxt_logical_router_link_port_on_tier0.link_port_tier0 UUID
```

The above command imports the logical router link port on the tier 0 logical router named `link_port_tier0` with the NSX id `UUID`.

» nsxt_logical_router_link_port_on_tier1

This resource provides the ability to configure a logical router link port on a tier 1 logical router. This port can then be used to connect the tier 1 logical router to another logical router.

» Example Usage

```
resource "nsxt_logical_router_link_port_on_tier1" "link_port_tier1" {
  description                   = "TIER1_PORT1 provisioned by Terraform"
  display_name                  = "TIER1_PORT1"
  logical_router_id             = "${nsxt_logical_tier1_router.rtr1.id}"
  linked_logical_router_port_id = "${nsxt_logical_router_link_port_on_tier0.link_port_tier0

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `logical_router_id` - (Required) Identifier for logical tier-1 router on which this port is created.
- `linked_logical_switch_port_id` - (Required) Identifier for port on logical Tier-0 router to connect to.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this port.

» Attributes Reference

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

- `id` - ID of the logical router link port.

- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing logical router link port on Tier-1 can be imported into this resource, via the following command:

```
terraform import nsxt_logical_router_link_port_on_tier1.link_port_tier1 UUID
```

The above command imports the logical router link port on the tier 1 router named `link_port_tier1` with the NSX id `UUID`.

» nsxt_logical_switch

This resource provides a method to create overlay logical switch in NSX. Virtual machines can then be connected to the appropriate logical switch for the desired topology and network connectivity.

» Example Usage

```
resource "nsxt_logical_switch" "switch1" {
  admin_state       = "UP"
  description       = "LS1 provisioned by Terraform"
  display_name      = "LS1"
  transport_zone_id = "${data.nsxt_transport_zone.transport_zone.id}"
  replication_mode  = "MTEP"

  tag {
    scope = "color"
    tag   = "blue"
  }

  address_binding {
    ip_address = "2.2.2.2"
    mac_address = "00:11:22:33:44:55"
  }

  switching_profile_id {
    key   = "${data.nsxt_switching_profile.qos_profiles.resource_type}"
    value = "${data.nsxt_switching_profile.qos_profiles.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `transport_zone_id` - (Required) Transport Zone ID for the logical switch.
- `admin_state` - (Optional) Admin state for the logical switch. Accepted values - 'UP' or 'DOWN'. The default value is 'UP'.
- `replication_mode` - (Optional) Replication mode of the Logical Switch. Accepted values - 'MTEP' (Hierarchical Two-Tier replication) and 'SOURCE' (Head Replication), with 'MTEP' being the default value. Applies to overlay logical switches.
- `switching_profile_id` - (Optional) List of IDs of switching profiles (of various types) to be associated with this switch. Default switching profiles will be used if not specified.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `ip_pool_id` - (Optional) Ip Pool ID to be associated with the logical switch.
- `mac_pool_id` - (Optional) Mac Pool ID to be associated with the logical switch.
- `address_binding` - (Optional) A list address bindings for this logical switch
  - `ip_address` - (Required) IP Address
  - `mac_address` - (Required) MAC Address
  - `vlan` - (Optional) Vlan
- `vlan` - (Deprecated, Optional) Vlan for vlan logical switch. This attribute is deprecated, please use nsxt_vlan_logical_switch resource to manage vlan logical switches.
- `vni` - (Optional, Readonly) Vni for the logical switch.
- `address_binding` - (Optional) List of Address Bindings for the logical switch. This setting allows to provide bindings between IP address, mac Address and vlan.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical switch.

» Attributes Reference

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

- `id` - ID of the logical switch.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing X can be imported into this resource, via the following command:

```
terraform import nsxt_logical_switch.switch1 UUID
```

The above command imports the logical switch named `switch1` with the NSX id `UUID`.

» nsxt_vlan_logical_switch

This resource provides a method to create vlan logical switch in NSX. Virtual machines can then be connected to the appropriate logical switch for the desired topology and network connectivity.

» Example Usage

```
resource "nsxt_vlan_logical_switch" "switch1" {
  admin_state       = "UP"
  description       = "LS1 provisioned by Terraform"
  display_name      = "LS1"
  transport_zone_id = "${data.nsxt_transport_zone.vlan_transport_zone.id}"
  vlan              = 2

  tag {
    scope = "color"
    tag   = "blue"
  }

  switching_profile_id {
    key   = "${data.nsxt_switching_profile.qos_profiles.resource_type}"
    value = "${data.nsxt_switching_profile.qos_profiles.id}"
  }
}
```

» Argument Reference

The following arguments are supported:

- `transport_zone_id` - (Required) Transport Zone ID for the logical switch.
- `admin_state` - (Optional) Admin state for the logical switch. Accepted values - 'UP' or 'DOWN'. The default value is 'UP'.
- `vlan` - (Required) Vlan for the logical switch.

- `switching_profile_id` - (Optional) List of IDs of switching profiles (of various types) to be associated with this switch. Default switching profiles will be used if not specified.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `ip_pool_id` - (Optional) Ip Pool ID to be associated with the logical switch.
- `mac_pool_id` - (Optional) Mac Pool ID to be associated with the logical switch.
- `address_binding` - (Optional) List of Address Bindings for the logical switch. This setting allows to provide bindings between IP address, mac Address and vlan.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical switch.

» Attributes Reference

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

- `id` - ID of the logical switch.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing X can be imported into this resource, via the following command:

```
terraform import nsxt_vlan_logical_switch.switch1 UUID
```

The above command imports the logical switch named `switch1` with the NSX id `UUID`.

» nsxt_logical_tier0_router

This resource provides a method for the management of a tier 0 logical router.

» Example Usage

```
resource "nsxt_logical_tier0_router" "tier0_router" {
  display_name           = "RTR"
  description            = "ACTIVE-STANDBY Tier0 router provisioned by Terraform"
  high_availability_mode = "ACTIVE_STANDBY"
  edge_cluster_id        = "${data.nsxt_edge_cluster.edge_cluster.id}"
  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `edge_cluster_id` - (Required) Edge Cluster ID for the logical Tier0 router. Changing this setting on existing router will re-create the router.
- `failover_mode` - (Optional) Failover mode which determines whether the preferred service router instance for given logical router will preempt the peer. Accepted values are PREEMPTIVE/NON_PREEMPTIVE. This setting is relevant only for ACTIVE_STANDBY high availability mode.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical Tier0 router.
- `high_availability_mode` - (Optional) High availability mode "ACTIVE_ACTIVE"/"ACTIVE_STANDBY". Changing this setting on existing router will re-create the router.

» Attributes Reference

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

- `id` - ID of the logical Tier0 router.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `firewall_sections` - (Optional) The list of firewall sections for this router

» Importing

An existing logical tier0 router can be imported into this resource, via the following command:

```
terraform import nsxt_logical_tier0_router.tier0_router UUID
```

The above command imports the logical tier 0 router named `tier0_router` with the NSX id `UUID`.

» nsxt_logical_tier1_router

This resource provides a method for the management of a tier 1 logical router. A tier 1 logical router is often used for tenants, users and applications. There can be many tier 1 logical routers connected to a common tier 0 provider router.

» Example Usage

```
resource "nsxt_logical_tier1_router" "tier1_router" {
  description                 = "RTR1 provisioned by Terraform"
  display_name                = "RTR1"
  failover_mode               = "PREEMPTIVE"
  edge_cluster_id             = "${data.nsxt_edge_cluster.edge_cluster.id}"
  enable_router_advertisement = true
  advertise_connected_routes  = false
  advertise_static_routes     = true
  advertise_nat_routes        = true
  advertise_lb_vip_routes     = true
  advertise_lb_snat_ip_routes = false

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `edge_cluster_id` - (Optional) Edge Cluster ID for the logical Tier1 router.
- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of the resource.
- `tag` - (Optional) A list of scope + tag pairs to associate with this logical Tier1 router.
- `failover_mode` - (Optional) This failover mode determines, whether the preferred service router instance for given logical router will preempt the peer. Note - It can be specified if and only if logical router is ACTIVE_STANDBY and NON_PREEMPTIVE mode is supported only for a Tier1 logical router. For ACTIVE_ACTIVE logical routers, this field must not be populated
- `enable_router_advertisement` - (Optional) Enable the router advertisement

- `advertise_connected_routes` - (Optional) Enable the router advertisement for all NSX connected routes
- `advertise_static_routes` - (Optional) Enable the router advertisement for static routes
- `advertise_nat_routes` - (Optional) Enable the router advertisement for NAT routes
- `advertise_lb_vip_routes` - (Optional) Enable the router advertisement for LB VIP routes
- `advertise_lb_snat_ip_routes` - (Optional) Enable the router advertisement for LB SNAT IP routes

» Attributes Reference

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

- `id` - ID of the logical Tier1 router.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `advertise_config_revision` - Indicates current revision number of the advertisement configuration object as seen by NSX-T API server. This attribute can be useful for debugging.
- `firewall_sections` - (Optional) The list of firewall sections for this router

» Importing

An existing logical tier1 router can be imported into this resource, via the following command:

```
terraform import nsxt_logical_tier1_router.tier1_router UUID
```

The above command imports the logical tier 1 router named `tier1_router` with the NSX id `UUID`.

» nsxt_nat_rule

This resource provides a means to configure a NAT rule in NSX. NAT provides network address translation between one IP address space and another IP address space. NAT rules can be destination NAT or source NAT rules.

» Example Usage

```
resource "nsxt_nat_rule" "rule1" {
  logical_router_id         = "${nsxt_logical_tier1_router.rtr1.id}"
  description               = "NR provisioned by Terraform"
  display_name              = "NR"
  action                    = "SNAT"
  enabled                   = true
  logging                   = true
  nat_pass                  = false
  translated_network        = "4.4.0.0/24"
  match_destination_network = "3.3.3.0/24"
  match_source_network      = "5.5.5.0/24"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `logical_router_id` - (Required) ID of the logical router.
- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this NAT rule.
- `action` - (Required) NAT rule action type. Valid actions are: SNAT, DNAT, NO_NAT and REFLEXIVE. All rules in a logical router are either stateless or stateful. Mix is not supported. SNAT and DNAT are stateful, and can NOT be supported when the logical router is running at active-active HA mode. The REFLEXIVE action is stateless. The NO_NAT action has no translated_fields, only match fields.
- `enabled` - (Optional) enable/disable the rule.
- `logging` - (Optional) enable/disable the logging of rule.
- `match_destination_network` - (Required for action=DNAT, not allowed for action=REFLEXIVE) IP Address | CIDR. Omitting this field implies Any.
- `match_source_network` - (Required for action=NO_NAT or REFLEXIVE, Optional for the other actions) IP Address | CIDR. Omitting this field implies Any.
- `nat_pass` - (Optional) Enable/disable to bypass following firewall stage. The default is true, meaning that the following firewall stage will be skipped. Please note, if action is NO_NAT, then nat_pass must be set to true or omitted.
- `translated_network` - (Required for action=DNAT or SNAT) IP Address

| IP Range | CIDR.
- `translated_ports` - (Optional) port number or port range. Allowed only when action=DNAT.
- `rule_priority` - The priority of the rule which is ascending, valid range [0-2147483647]. If multiple rules have the same priority, evaluation sequence is undefined.

» Attributes Reference

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

- `id` - ID of the NAT rule.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing NAT rule can be imported into this resource, via the following command:

```
terraform import nsxt_nat_rule.rule1 logical-router-uuid/nat-rule-num
```

The above command imports the NAT rule named `rule1` with the number id `nat-rule-num` that belongs to the tier 1 logical router with the NSX id `logical-router-uuid`.

» nsxt_ns_group

This resource provides a method to create and manage a network and security (NS) group in NSX. A NS group is used to group other objects into collections for application of other settings.

» Example Usage

```
resource "nsxt_ns_group" "group2" {
  description  = "NG provisioned by Terraform"
  display_name = "NG"

  member {
    target_type = "NSGroup"
    value       = "${nsxt_ns_group.group1.id}"
  }
  membership_criteria {
    target_type = "LogicalPort"
    scope       = "XXX"
    tag         = "YYY"
  }

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this NS group.
- `member` - (Optional) Reference to the direct/static members of the NSGroup. Can be ID based expressions only. VirtualMachine cannot be added as a static member.
  - `target_type` - (Required) Static member type, one of: NSGroup, IPSet, LogicalPort, LogicalSwitch, MACSet
  - `value` - (Required) Member ID
- `membership_criteria` - (Optional) List of tag or ID expressions which define the membership criteria for this NSGroup. An object must satisfy at least one of these expressions to qualify as a member of this group.
  - `target_type` - (Required) Dynamic member type, one of: LogicalPort, LogicalSwitch, VirtualMachine.
  - `scope` - (Optional) Tag scope for matching dynamic members.
  - `tag` - (Optional) Tag value for matching dynamic members.

» Attributes Reference

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

- `id` - ID of the NS group.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing networking and security group can be imported into this resource, via the following command:

```
terraform import nsxt_ns_group.group2 UUID
```

The above command imports the networking and security group named `group2` with the NSX id `UUID`.

» nsxt_static_route

This resource provides a means to configure static routes in NSX to determine where IP traffic is routed.

» Example Usage

```
resource "nsxt_static_route" "static_route" {
  description       = "SR provisioned by Terraform"
  display_name      = "SR"
  logical_router_id = "${nsxt_logical_tier1_router.router1.id}"
  network           = "4.4.4.0/24"

  next_hop {
    ip_address              = "8.0.0.10"
    administrative_distance = "1"
    logical_router_port_id  = "${nsxt_logical_router_downlink_port.downlink_port.id}"
  }

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this static route.

- `logical_router_id` - (Required) Logical router id.
- `network` - (Required) CIDR.
- `next_hop` - (Required) List of Next Hops, each with those arguments:
  - `administrative_distance` - (Optional) Administrative Distance for the next hop IP.
  - `ip_address` - (Optional) Next Hop IP.
  - `logical_router_port_id` - (Optional) Reference of logical router port to be used for next hop.

» Attributes Reference

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

- `id` - ID of the static route.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `next_hop` additional arguments:
  - `bfd_enabled` - Status of bfd for this next hop where bfd_enabled = true indicate bfd is enabled for this next hop and bfd_enabled = false indicate bfd peer is disabled or not configured for this next hop.
  - `blackhole_action` - Action to be taken on matching packets for NULL routes.

» Importing

An existing static route can be imported into this resource, via the following command:

```
terraform import nsxt_static_route.static_route logical-router-uuid/static-route-num
```

The above command imports the static route named `static_route` with the number `static-route-num` that belongs to the tier 1 logical router with the NSX id `logical-router-uuid`.

» nsxt_vm_tags

This resource provides a means to configure tags that are applied to objects such as virtual machines. A virtual machine is not directly managed by NSX however, NSX allows attachment of tags to a virtual machine. This tagging enables tag based grouping of objects. Deletion of `nsxt_vm_tags` resource will remove all tags from the virtual machine and is equivalent to update operation with empty tag set.

» Example Usage

```
resource "nsxt_vm_tags" "vm1_tags" {
  instance_id = "${vsphere_virtual_machine.vm1.id}"

  tag {
    scope = "color"
    tag   = "blue"
  }

  logical_port_tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `instance_id` - (Required) BIOS Id of the Virtual Machine.
- `tag` - (Optional) A list of scope + tag pairs to associate with this VM.
- `logical_port_tag` - (Optional) A list of scope + tag pairs to associate with all logical ports that are automatically created for this VM.

» Importing

An existing Tags collection can be imported into this resource, via the following command:

```
terraform import nsxt_vm_tags.vm1_tags id
```

The above would import NSX virtual machine tags as a resource named `vm1_tags` with the NSX id `id`, where id is external ID (not the BIOS id) of the virtual machine.

» nsxt_algorithm_type_ns_service

This resource provides a way to configure a networking and security service which can be used with the NSX firewall. A networking and security service is an object that contains the TCP/UDP algorithm, source ports and destination ports in a single entity.

» Example Usage

```
resource "nsxt_algorithm_type_ns_service" "ns_service_alg" {
  description       = "S1 provisioned by Terraform"
  display_name      = "S1"
  algorithm         = "FTP"
  destination_port  = "21"
  source_ports      = ["9001-9003"]

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description.
- `destination_port` - (Required) a single destination port.
- `source_ports` - (Optional) Set of source ports/ranges.
- `algorithm` - (Required) Algorithm one of ”ORACLE_TNS”, ”FTP”, ”SUN_RPC_TCP”, ”SUN_RPC_UDP”, ”MS_RPC_TCP”, ”MS_RPC_UDP”, ”NBNS_BROADCAST”, ”NBDG_BROADCAST”, ”TFTP”
- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can’t be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing Algorithm type NS service can be imported into this resource, via the following command:

```
terraform import nsxt_algorithm_type_ns_service.ns_service_alg UUID
```

The above command imports the algorithm based networking and security service named `ns_service_alg` with the NSX id `UUID`.

» nsxt_ether_type_ns_service

This resource provides a way to configure a networking and security service which can be used within NSX. This specific service is for the layer 2 Ethernet protocol.

» Example Usage

```
resource "nsxt_ether_type_ns_service" "etns" {
  description  = "S1 provisioned by Terraform"
  display_name = "S1"
  ether_type   = "1536"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description.
- `ether_type` - (Required) Type of the encapsulated protocol.
- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can't be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing Ethernet type NS service can be imported into this resource, via the following command:

```
terraform import nsxt_ether_type_ns_service.etns UUID
```

The above command imports the ethernet type networking and security service named `etns` with the NSX id `UUID`.

» nsxt_icmp_type_ns_service

This resource provides a way to configure a networking and security service which can be used within NSX. This specific service is for the ICMP protocol.

» Example Usage

```
resource "nsxt_icmp_type_ns_service" "ns_service_icmp" {
  description  = "S1 provisioned by Terraform"
  display_name = "S1"
  protocol     = "ICMPv4"
  icmp_type    = "5"
  icmp_code    = "1"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description.
- `protocol` - (Required) Version of ICMP protocol ICMPv4 or ICMPv6.
- `icmp_type` - (Optional) ICMP message type.
- `icmp_code` - (Optional) ICMP message code
- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can't be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing ICMP type NS Service can be imported into this resource, via the following command:

```
terraform import nsxt_icmp_type_ns_service.x id
```

The above service imports the ICMP type network and security service named `x` with the NSX id `id`.

» nsxt_igmp_type_ns_service

This resource provides a way to configure a networking and security service which can be used within NSX. This specific service is for the IGMP protocol.

» Example Usage

```
resource "nsxt_igmp_type_ns_service" "ns_service_igmp" {
  description  = "S1 provisioned by Terraform"
  display_name = "S1"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description.

- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can't be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IGMP type NS Service can be imported into this resource, via the following command:

```
terraform import nsxt_igmp_type_ns_service.ns_service_igmp UUID
```

The above command imports the IGMP based networking and security service named `ns_service_igmp` with the NSX id `UUID`.

» nsxt_ip_protocol_ns_service

This resource provides a way to configure a networking and security service which can be used within NSX. This specific service is for the IP protocol.

» Example Usage

```
resource "nsxt_ip_protocol_ns_service" "ns_service_ip" {
  description  = "S1 provisioned by Terraform"
  display_name = "S1"
  protocol     = "10"

  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description.
- `protocol` - (Required) IP protocol number (0-255)
- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can't be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IP protocol NS service can be imported into this resource, via the following command:

```
terraform import nsxt_ip_protocol_ns_service.ns_service_ip UUID
```

The above command imports the IP protocol based networking and security service named `ns_service_ip` with the NSX id `UUID`.

» nsxt_l4_port_set_ns_service

This resource provides a way to configure a networking and security service which can be used within NSX. This specific service is for configuration of layer 4 ports.

» Example Usage

```
resource "nsxt_l4_port_set_ns_service" "ns_service_l4" {
  description       = "S1 provisioned by Terraform"
  display_name      = "S1"
  protocol          = "TCP"
  destination_ports = ["73", "8080", "81"]
  tag {
    scope = "color"
    tag   = "blue"
  }
}
```

» Argument Reference

The following arguments are supported:

- `display_name` - (Optional) Display name, defaults to ID if not set.
- `description` - (Optional) Description of this resource.
- `destination_ports` - (Optional) Set of destination ports.
- `source_ports` - (Optional) Set of source ports.
- `protocol` - (Required) L4 protocol. Accepted values - 'TCP' or 'UDP'.
- `tag` - (Optional) A list of scope + tag pairs to associate with this service.

» Attributes Reference

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

- `id` - ID of the NS service.
- `default_service` - The default NSServices are created in the system by default. These NSServices can't be modified/deleted.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing L4 port set NS service can be imported into this resource, via the following command:

```
terraform import nsxt_l4_port_set_ns_service.ns_service_l4 UUID
```

The above command imports the layer 4 port based networking and security service named `ns_service_l4` with the NSX id `UUID`.

» nsxt_ns_service_group

Provides a resource to configure NS service group on NSX-T manager

» Example Usage

```
data "nsxt_ns_service" "dns" {
  display_name = "DNS"
}

resource "nsxt_ip_protocol_ns_service" "prot17" {
  display_name = "ip_prot"
  protocol     = "17"
}

resource "nsxt_ns_service_group" "ns_service_group" {
  description  = "ns_service_group provisioned by Terraform"
  display_name = "ns_service_group"
  members      = ["${nsxt_ip_protocol_ns_service.prot17.id}", "${data.nsxt_ns_service.dns.id

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this NS service group.
- `members` - (Required) List of NSServices IDs that can be added as members to an NSServiceGroup. All members should be of the same L2 type: Ethernet, or Non Ethernet.

» Attributes Reference

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

- `id` - ID of the NS service group.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing ns service group can be imported into this resource, via the following command:

```
terraform import nsxt_ns_service_group.ns_service_group UUID
```

The above would import the NS service group named `ns_service_group` with the nsx id `UUID`

» nsxt_lb_icmp_monitor

Provides a resource to configure lb icmp monitor on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_icmp_monitor" "lb_icmp_monitor" {
  description  = "lb_icmp_monitor provisioned by Terraform"
  display_name = "lb_icmp_monitor"
  fall_count   = 3
  interval     = 5
  monitor_port = 7887
  rise_count   = 3
  timeout      = 10
  data_length  = 56

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb icmp monitor.

- `fall_count` - (Optional) Number of consecutive checks must fail before marking it down.
- `interval` - (Optional) The frequency at which the system issues the monitor check (in seconds).
- `monitor_port` - (Optional) If the monitor port is specified, it would override pool member port setting for healthcheck. Port range is not supported.
- `rise_count` - (Optional) Number of consecutive checks must pass before marking it up.
- `timeout` - (Optional) Number of seconds the target has in which to respond to the monitor request.
- `data_length` - (Optional) The data size (in bytes) of the ICMP healthcheck packet.

» Attributes Reference

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

- `id` - ID of the lb_icmp_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb icmp monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_icmp_monitor.lb_icmp_monitor UUID
```

The above would import the lb icmp monitor named `lb_icmp_monitor` with the nsx id `UUID`

» nsxt_lb_tcp_monitor

Provides a resource to configure lb tcp monitor on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_tcp_monitor" "lb_tcp_monitor" {
  description  = "lb_tcp_monitor provisioned by Terraform"
  display_name = "lb_tcp_monitor"
  fall_count   = 3
  interval     = 5
  monitor_port = 7887
  rise_count   = 3
  timeout      = 10

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb tcp monitor.
- `fall_count` - (Optional) Number of consecutive checks must fail before marking it down.
- `interval` - (Optional) The frequency at which the system issues the monitor check (in seconds).
- `monitor_port` - (Optional) If the monitor port is specified, it would override pool member port setting for healthcheck. Port range is not supported.
- `rise_count` - (Optional) Number of consecutive checks must pass before marking it up.
- `timeout` - (Optional) Number of seconds the target has in which to respond to the monitor request.
- `receive` - (Optional) Expected data, if specified, can be anywhere in the response and it has to be a string, regular expressions are not supported.
- `send` - (Optional) Payload to send out to the monitored server. If both send and receive are not specified, then just a TCP connection is established (3-way handshake) to validate server is healthy, no data is sent.

» Attributes Reference

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

- `id` - ID of the lb_tcp_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb tcp monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_tcp_monitor.lb_tcp_monitor UUID
```

The above would import the lb tcp monitor named `lb_tcp_monitor` with the nsx id `UUID`

» nsxt_lb_udp_monitor

Provides a resource to configure lb udp monitor on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_udp_monitor" "lb_udp_monitor" {
  description  = "lb_udp_monitor provisioned by Terraform"
  display_name = "lb_udp_monitor"
  fall_count   = 3
  interval     = 5
  monitor_port = 7887
  rise_count   = 3
  timeout      = 10
  send         = "hi"
  receive      = "hello"

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb udp monitor.

- `fall_count` - (Optional) Number of consecutive checks must fail before marking it down.
- `interval` - (Optional) The frequency at which the system issues the monitor check (in seconds).
- `monitor_port` - (Optional) If the monitor port is specified, it would override pool member port setting for healthcheck. Port range is not supported.
- `rise_count` - (Optional) Number of consecutive checks must pass before marking it up.
- `timeout` - (Optional) Number of seconds the target has in which to respond to the monitor request.
- `receive` - (Required) Expected data, if specified, can be anywhere in the response and it has to be a string, regular expressions are not supported.
- `send` - (Required) Payload to send out to the monitored server.

» Attributes Reference

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

- `id` - ID of the lb_udp_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb udp monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_udp_monitor.lb_udp_monitor UUID
```

The above would import the lb udp monitor named `lb_udp_monitor` with the nsx id `UUID`

» nsxt_lb_http_monitor

Provides a resource to configure lb http monitor on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_http_monitor" "lb_http_monitor" {
  description           = "lb_http_monitor provisioned by Terraform"
  display_name          = "lb_http_monitor"
  fall_count            = 2
  interval              = 5
  monitor_port          = 8080
  rise_count            = 5
  timeout               = 10
  request_body          = "ping"
  request_method        = "HEAD"
  request_url           = "/index.html"
  request_version       = "HTTP_VERSION_1_1"
  response_body         = "pong"
  response_status_codes = [200, 304]

  tag {
    scope = "color"
    tag   = "red"
  }

  request_header {
    name  = "X-healthcheck"
    value = "NSX"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb http monitor.
- `fall_count` - (Optional) Number of consecutive checks that must fail before marking it down.
- `interval` - (Optional) The frequency at which the system issues the monitor check (in seconds).
- `monitor_port` - (Optional) If the monitor port is specified, it would override pool member port setting for healthcheck. A port range is not supported.
- `rise_count` - (Optional) Number of consecutive checks that must pass before marking it up.
- `timeout` - (Optional) Number of seconds the target has to respond to the monitor request.
- `request_body` - (Optional) String to send as HTTP health check request

body. Valid only for certain HTTP methods like POST.

- `request_header` - (Optional) HTTP request headers.
- `request_method` - (Optional) Health check method for HTTP monitor type. Valid values are GET, HEAD, PUT, POST and OPTIONS.
- `request_url` - (Optional) URL used for HTTP monitor.
- `request_version` - (Optional) HTTP request version. Valid values are HTTP_VERSION_1_0 and HTTP_VERSION_1_1.
- `response_body` - (Optional) If response body is specified, healthcheck HTTP response body is matched against the specified string and server is considered healthy only if there is a match (regular expressions not supported). If response body string is not specified, HTTP healthcheck is considered successful if the HTTP response status code is among configured values.
- `response_status_codes` - (Optional) HTTP response status code should be a valid HTTP status code.

» Attributes Reference

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

- `id` - ID of the lb_http_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb http monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_http_monitor.lb_http_monitor UUID
```

The above would import the lb http monitor named `lb_http_monitor` with the nsx id `UUID`

» nsxt_lb_https_monitor

Provides a resource to configure lb https monitor on NSX-T manager

» Example Usage

```
data "nsxt_certificate" "client" {
  display_name = "client-1"
}
```

```
data "nsxt_certificate" "CA" {
  display_name = "ca-1"
}

resource "nsxt_lb_https_monitor" "lb_https_monitor" {
  description             = "lb_https_monitor provisioned by Terraform"
  display_name            = "lb_https_monitor"
  fall_count              = 2
  interval                = 5
  monitor_port            = 8080
  rise_count              = 5
  timeout                 = 10
  certificate_chain_depth = 2
  ciphers                 = ["TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256", "TLS_ECDHE_RSA_WITH_AE
  client_certificate_id   = "${data.nsxt_certificate.client.id}"
  protocols               = ["TLS_V1_2"]
  request_body            = "ping"
  request_method          = "HEAD"
  request_url             = "/index.html"
  request_version         = "HTTP_VERSION_1_1"
  response_body           = "pong"
  response_status_codes   = [200, 304]
  server_auth             = "REQUIRED"
  server_auth_ca_ids      = ["${data.nsxt_certificate.CA.id}"]
  server_auth_crl_ids     = ["78ba3814-bfe1-45e5-89d3-46862bed7896"]

  request_header {
    name  = "X-healthcheck"
    value = "NSX"
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.

- `tag` - (Optional) A list of scope + tag pairs to associate with this lb https monitor.
- `fall_count` - (Optional) Number of consecutive checks that must fail before marking it down.
- `interval` - (Optional) The frequency at which the system issues the monitor check (in seconds).
- `monitor_port` - (Optional) If the monitor port is specified, it would override pool member port setting for healthcheck. A port range is not supported.
- `rise_count` - (Optional) Number of consecutive checks that must pass before marking it up.
- `timeout` - (Optional) Number of seconds the target has to respond to the monitor request.
- `certificate_chain_depth` - (Optional) Authentication depth is used to set the verification depth in the server certificates chain.
- `ciphers` - (Optional) List of supported SSL ciphers.
- `client_certificate_id` - (Optional) Client certificate can be specified to support client authentication.
- `protocols` - (Optional) SSL versions TLS1.1 and TLS1.2 are supported and enabled by default. SSLv2, SSLv3, and TLS1.0 are supported, but disabled by default.
- `request_body` - (Optional) String to send as HTTP health check request body. Valid only for certain HTTP methods like POST.
- `request_header` - (Optional) HTTP request headers.
- `request_method` - (Optional) Health check method for HTTP monitor type. Valid values are GET, HEAD, PUT, POST and OPTIONS.
- `request_url` - (Optional) URL used for HTTP monitor.
- `request_version` - (Optional) HTTP request version. Valid values are HTTP_VERSION_1_0 and HTTP_VERSION_1_1.
- `response_body` - (Optional) If response body is specified, healthcheck HTTP response body is matched against the specified string and server is considered healthy only if there is a match (regular expressions not supported). If response body string is not specified, HTTP healthcheck is considered successful if the HTTP response status code is among configured values.
- `response_status_codes` - (Optional) HTTP response status code should be a valid HTTP status code.
- `server_auth` - (Optional) Server authentication mode - REQUIRED or IGNORE.
- `server_auth_ca_ids` - (Optional) If server auth type is REQUIRED, server certificate must be signed by one of the trusted Certificate Authorities (CAs), also referred to as root CAs, whose self signed certificates are specified.
- `server_auth_crl_ids` - (Optional) A Certificate Revocation List (CRL) can be specified in the server-side SSL profile binding to disallow compromised server certificates.

» Attributes Reference

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

- `id` - ID of the lb_https_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.
- `is_secure` - This flag is set to true when all the ciphers and protocols are secure. It is set to false when one of the ciphers or protocols is insecure.

» Importing

An existing lb https monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_https_monitor.lb_https_monitor UUID
```

The above would import the lb https monitor named `lb_https_monitor` with the nsx id `UUID`

» nsxt_lb_passive_monitor

Provides a resource to configure lb passive monitor on NSX-T manager

NOTE: This resource requires NSX version 2.3 or higher.

» Example Usage

```
resource "nsxt_lb_passive_monitor" "lb_passive_monitor" {
  description  = "lb_passive_monitor provisioned by Terraform"
  display_name = "lb_passive_monitor"
  max_fails    = 3
  timeout      = 10

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this lb passive monitor.
- `max_fails` - (Optional) When consecutive failures reach this value, the member is considered temporarily unavailable for a configurable period.
- `timeout` - (Optional) After this timeout period, the member is probed again.

» Attributes Reference

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

- `id` - ID of the lb_passive_monitor.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing lb passive monitor can be imported into this resource, via the following command:

```
terraform import nsxt_lb_passive_monitor.lb_passive_monitor UUID
```

The above would import the lb passive monitor named `lb_passive_monitor` with the nsx id `UUID`

» nsxt_ip_discovery_switching_profile

Provides a resource to configure IP discovery switching profile on NSX-T manager

» Example Usage

```
resource "nsxt_ip_discovery_switching_profile" "ip_discovery_switching_profile" {
  description           = "ip_discovery_switching_profile provisioned by Terraform"
  display_name          = "ip_discovery_switching_profile"
  vm_tools_enabled      = "false"
  arp_snooping_enabled  = "true"
  dhcp_snooping_enabled = "false"
  arp_bindings_limit    = "1"
  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this IP discovery switching profile.
- `arp_snooping_enabled` - (Optional) A boolean flag iIndicates whether ARP snooping is enabled.
- `vm_tools_enabled` - (Optional) A boolean flag iIndicates whether VM tools will be enabled. This option is only supported on ESX where vm-tools is installed.
- `dhcp_snooping_enabled` - (Optional) A boolean flag iIndicates whether DHCP snooping is enabled.
- `arp_bindings_limit` - (Optional) Limit for the amount of ARP bindings.

» Attributes Reference

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

- `id` - ID of the IP discovery switching profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing IP discovery switching profile can be imported into this resource, via the following command:

```
terraform import nsxt_ip_discovery_switching_profile.ip_discovery_switching_profile UUID
```

The above would import the IP discovery switching profile named `ip_discovery_switching_profile` with the nsx id `UUID`

» nsxt_mac_management_switching_profile

Provides a resource to configure MAC management switching profile on NSX-T manager

» Example Usage

```
resource "nsxt_mac_management_switching_profile" "mac_management_switching_profile" {
  description        = "mac_management_switching_profile provisioned by Terraform"
  display_name       = "mac_management_switching_profile"
  mac_change_allowed = "true"

  mac_learning {
    enabled                  = "true"
    limit                    = "4096"
    limit_policy             = "ALLOW"
    unicast_flooding_allowed = "false"
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this MAC management switching profile.
- `mac_change_allowed` - (Optional) A boolean flag indicating allowing source MAC address change.
- `mac_learning` - (Optional) Mac learning configuration:
  - `enabled` - (Optional) A boolean flag indicating allowing source MAC address learning.
  - `unicast_flooding_allowed` - (Optional) A boolean flag indicating allowing flooding for unlearned MAC for ingress traffic. Can be True only if mac_learning is enabled.
  - `limit` - (Optional) The maximum number of MAC addresses that can be learned on this port.

  - `limit_policy` - (Optional) The policy after MAC Limit is exceeded: ALLOW/DROP.

» Attributes Reference

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

- `id` - ID of the MAC management switching profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing MAC management switching profile can be imported into this resource, via the following command:

```
terraform import nsxt_mac_management_switching_profile.mac_management_switching_profile UUID
```

The above would import the MAC management switching profile named `mac_management_switching_profile` with the nsx id `UUID`

» nsxt_spoofguard_switching_profile

Provides a resource to configure spoofguard switching profile on NSX-T manager

» Example Usage

```
resource "nsxt_spoofguard_switching_profile" "spoofguard_switching_profile" {
  description                       = "spoofguard_switching_profile provisioned by Terraform
  display_name                      = "spoofguard_switching_profile"
  address_binding_whitelist_enabled = "true"

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.

- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this spoofguard switching profile.
- `address_binding_whitelist_enabled` - (Optional) A boolean flag indicating whether this profile overrides the default system wide settings for Spoof Guard when assigned to ports.

» Attributes Reference

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

- `id` - ID of the spoofguard switching profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing spoofguard switching profile can be imported into this resource, via the following command:

```
terraform import nsxt_spoofguard_switching_profile.spoofguard_switching_profile UUID
```

The above would import the spoofguard switching profile named `spoofguard_switching_profile` with the nsx id `UUID`

» nsxt_qos_switching_profile

Provides a resource to configure Qos switching profile on NSX-T manager

» Example Usage

```
resource "nsxt_qos_switching_profile" "qos_switching_profile" {
  description      = "qos_switching_profile provisioned by Terraform"
  display_name     = "qos_switching_profile"
  class_of_service = "5"
  dscp_trusted     = "true"
  dscp_priority    = "53"

  ingress_rate_shaper {
    enabled         = "true"
    peak_bw_mbps    = "800"
    burst_size      = "200"
    average_bw_mbps = "100"
  }

  egress_rate_shaper {
    enabled         = "true"
    peak_bw_mbps    = "800"
    burst_size      = "200"
    average_bw_mbps = "100"
  }

  ingress_broadcast_rate_shaper {
    enabled         = "true"
    average_bw_kbps = "111"
    burst_size      = "222"
    peak_bw_kbps    = "500"
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this qos switching profile.
- `class_of_service` - (Optional) Class of service.
- `dscp_trusted` - (Optional) Trust mode for DSCP (False by default)
- `dscp_priority` - (Optional) DSCP Priority (0-63)
- `ingress_rate_shaper` - (Optional) Ingress rate shaper configuration:
  - `enabled` - (Optional) Whether this rate shaper is enabled.
  - `average_bw_mbps` - (Optional) Average Bandwidth in MBPS.
  - `peak_bw_mbps` - (Optional) Peak Bandwidth in MBPS.
  - `burst_size` - (Optional) Burst size in bytes.
- `egress_rate_shaper` - (Optional) Egress rate shaper configuration:
  - `enabled` - (Optional) Whether this rate shaper is enabled.
  - `average_bw_mbps` - (Optional) Average Bandwidth in MBPS.
  - `peak_bw_mbps` - (Optional) Peak Bandwidth in MBPS.
  - `burst_size` - (Optional) Burst size in bytes.

- `ingress_broadcast_rate_shaper` - (Optional) Ingress rate shaper configuration:
  - `enabled` - (Optional) Whether this rate shaper is enabled.
  - `average_bw_kbps` - (Optional) Average Bandwidth in KBPS.
  - `peak_bw_kbps` - (Optional) Peak Bandwidth in KBPS.
  - `burst_size` - (Optional) Burst size in bytes.

» Attributes Reference

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

- `id` - ID of the QoS switching profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing qos switching profile can be imported into this resource, via the following command:

```
terraform import nsxt_qos_switching_profile.qos_switching_profile UUID
```

The above would import the Qos switching profile named `qos_switching_profile` with the nsx id `UUID`

» nsxt_switch_security_switching_profile

Provides a resource to configure switch security switching profile on NSX-T manager

» Example Usage

```
resource "nsxt_switch_security_switching_profile" "switch_security_switching_profile" {
  description           = "switch_security_switching_profile provisioned by Terraform"
  display_name          = "switch_security_switching_profile"
  block_non_ip          = true
  block_client_dhcp     = false
  block_server_dhcp     = false
  bpdu_filter_enabled   = true
  bpdu_filter_whitelist = ["01:80:c2:00:00:01"]

  rate_limits {
    enabled      = true
    rx_broadcast = 32
    rx_multicast = 32
    tx_broadcast = 32
    tx_multicast = 32
  }

  tag {
    scope = "color"
    tag   = "red"
  }
}
```

» Argument Reference

The following arguments are supported:

- `description` - (Optional) Description of this resource.
- `display_name` - (Optional) The display name of this resource. Defaults to ID if not set.
- `tag` - (Optional) A list of scope + tag pairs to associate with this qos switching profile.
- `block_non_ip` - (Optional) Indicates whether blocking of all traffic except IP/(G)ARP/BPDU is enabled.
- `block_client_dhcp` - (Optional) Indicates whether DHCP client blocking is enabled
- `block_server_dhcp` - (Optional) Indicates whether DHCP server blocking is enabled
- `bpdu_filter_enabled` - (Optional) Indicates whether BPDU filter is enabled
- `bpdu_filter_whitelist` - (Optional) Set of allowed MAC addresses to be excluded from BPDU filtering, if enabled.
- `rate_limits` - (Optional) Rate limit definitions for broadcast and multicast traffic.
  - `enabled` - (Optional) Whether rate limitimg is enabled.
  - `rx_broadcast` - (Optional) Incoming broadcast traffic limit in packets per second.
  - `rx_multicast` - (Optional) Incoming multicast traffic limit in packets per second.
  - `tx_broadcast` - (Optional) Outgoing broadcast traffic limit in packets per second.
  - `tx_multicast` - (Optional) Outgoing multicast traffic limit in packets per second.

» Attributes Reference

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

- `id` - ID of the switch security switching profile.
- `revision` - Indicates current revision number of the object as seen by NSX-T API server. This attribute can be useful for debugging.

» Importing

An existing switch security switching profile can be imported into this resource, via the following command:

```
terraform import nsxt_switch_security_switching_profile.switch_security_switching_profile UU
```

The above would import switching profile named `switch_security_switching_profile` with the nsx id `UUID`