Cloudflare Provider

The Cloudflare provider is used to interact with resources supported by Cloudflare. The provider needs to be configured with the proper credentials before it can be used.

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

Example Usage

```
# Configure the Cloudflare provider
provider "cloudflare" {
  email = "${var.cloudflare_email}"
  token = "${var.cloudflare_token}"
}

# Create a record
resource "cloudflare_record" "www" {
  # ...
}

# Create a page rule
resource "cloudflare_page_rule" "www" {
  # ...
}
```

Argument Reference

The following arguments are supported:

- `email` - (Required) The email associated with the account. This can also be specified with the `CLOUDFLARE_EMAIL` shell environment variable.
- `token` - (Required) The Cloudflare API token. This can also be specified with the `CLOUDFLARE_TOKEN` shell environment variable.
- `rps` - (Optional) RPS limit to apply when making calls to the API. Default: 4. This can also be specified with the `CLOUDFLARE_RPS` shell environment variable.
- `retries` - (Optional) Maximum number of retries to perform when an API request fails. Default: 3. This can also be specified with the `CLOUDFLARE_RETRIES` shell environment variable.
- `min_backoff` - (Optional) Minimum backoff period in seconds after failed API calls. Default: 1. This can also be specified with the `CLOUDFLARE_MIN_BACKOFF` shell environment variable.
- `max_backoff` - (Optional) Maximum backoff period in seconds after failed API calls Default: 30. This can also be specified with the `CLOUDFLARE_MAX_BACKOFF` shell environment variable.
- `api_client_logging` - (Optional) Whether to print logs from the API client (using the default log library logger). Default: false. This can also be specified with the `CLOUDFLARE_API_CLIENT_LOGGING` shell environment variable.
- `org_id` - (Optional) Configure API client with this organisation ID, so calls use the organization API rather than the (default) user API. This is required for other users in your organization to have access to the resources you manage.

This can also be specified with the `CLOUDFLARE_ORG_ID` shell environment variable.

- `use_org_from_zone` - (Optional) Takes a zone name value. This is used to lookup the organization ID that owns this zone, which will be used to configure the API client. If `org_id` is also specified, this field will be ignored. This can also be specified with the `CLOUDFLARE_ORG_ZONE` shell environment variable.

cloudflare_ip_ranges

Use this data source to get the IP ranges (https://www.cloudflare.com/ips/) of Cloudflare edge nodes.

Example Usage

```
data "cloudflare_ip_ranges" "cloudflare" {}

resource "google_compute_firewall" "allow_cloudflare_ingress" {
  name    = "from-cloudflare"
  network = "default"

  source_ranges = ["${data.cloudflare_ip_ranges.cloudflare.ipv4_cidr_blocks}"]

  allow {
    ports    = "443"
    protocol = "tcp"
  }
}
```

Attributes Reference

- `cidr_blocks` - The lexically ordered list of all CIDR blocks.
- `ipv4_cidr_blocks` - The lexically ordered list of only the IPv4 CIDR blocks.
- `ipv6_cidr_blocks` - The lexically ordered list of only the IPv6 CIDR blocks.

cloudflare_access_application

Provides a Cloudflare Access Application resource. Access Applications are used to restrict access to a whole application using an authorisation gateway managed by Cloudflare.

Example Usage

```
resource "cloudflare_access_application" "staging_app" {
  zone_id          = "1d5fdc9e88c8a8c4518b068cd94331fe"
  name             = "staging application"
  domain           = "staging.example.com"
  session_duration = "24h"
}
```

Argument Reference

The following arguments are supported:

- `zone_id` - (Required) The DNS zone to which the access rule should be added.
- `name` - (Required) Friendly name of the Access Application.
- `domain` - (Required) The complete URL of the asset you wish to put Cloudflare Access in front of. Can include subdomains or paths. Or both.
- `session_duration` - (Optional) How often a user will be forced to re-authorise. Must be one of `30m`, `6h`, `12h`, `24h`, `168h`, `730h`.

Import

Access Applications can be imported using a composite ID formed of zone ID and application ID.

```
$ terraform import cloudflare_access_application.staging cb029e245cfdd66dc8d2e570d5dd3322/d41d8cd98f00b20
4e9800998ecf8427e
```

cloudflare_access_policy

Provides a Cloudflare Access Policy resource. Access Policies are used in conjunction with Access Applications to restrict access to a particular resource.

Example Usage

```
# Allowing access to `test@example.com` email address only
resource "cloudflare_access_policy" "test_policy" {
  application_id = "cb029e245cfdd66dc8d2e570d5dd3322"
  zone_id        = "d41d8cd98f00b204e9800998ecf8427e"
  name           = "staging policy"
  precedence     = "1"
  decision       = "allow"

  include = {
    email = ["test@example.com"]
  }
}

# Allowing `test@example.com` to access but only when coming from a
# specific IP.
resource "cloudflare_access_policy" "test_policy" {
  application_id = "cb029e245cfdd66dc8d2e570d5dd3322"
  zone_id        = "d41d8cd98f00b204e9800998ecf8427e"
  name           = "staging policy"
  precedence     = "1"
  decision       = "allow"

  include = {
    email = ["test@example.com"]
  }

  require = {
    ip = ["${var.office_ip}"]
  }
}
```

Argument Reference

The following arguments are supported:

- `application_id` - (Required) The ID of the application the policy is associated with.
- `zone_id` - (Required) The DNS zone to which the access rule should be added.
- `decision` - (Required) The complete URL of the asset you wish to put Cloudflare Access in front of. Can include subdomains or paths. Or both.
- `name` - (Required) Friendly name of the Access Application.
- `precedence` - (Optional) The unique precedence for policies on a single application. Integer.
- `require` - (Optional) A series of access conditions, see below for full list.

- `exclude` - (Optional) A series of access conditions, see below for full list.
- `include` - (Required) A series of access conditions, see below for full list.

Conditions

`require`, `exclude` and `include` arguments share the available conditions which can be applied. The conditions are:

- `ip` - (Optional) A list of IP addresses or ranges. Example: `ip = ["1.2.3.4", "10.0.0.0/2"]`
- `email` - (Optional) A list of email addresses. Example: `email = ["test@example.com"]`
- `email_domain` - (Optional) A list of email domains. Example: `email_domain = ["example.com"]`
- `everyone` - (Optional) Boolean indicating permitting access for all requests. Example: `everyone = true`

Import

Access Policies can be imported using a composite ID formed of zone ID, application ID and policy ID.

```
$ terraform import cloudflare_access_policy.staging cb029e245cfdd66dc8d2e570d5dd3322/d41d8cd98f00b204e980
0998ecf8427e/67ea780ce4982c1cfbe6b7293afc765d
```

where

- `cb029e245cfdd66dc8d2e570d5dd3322` - Zone ID
- `d41d8cd98f00b204e9800998ecf8427e` - Access Application ID
- `67ea780ce4982c1cfbe6b7293afc765d` - Access Policy ID

cloudflare_access_rule

Provides a Cloudflare IP Firewall Access Rule resource. Access control can be applied on basis of IP addresses, IP ranges, AS numbers or countries.

Example Usage

```
# Challenge requests coming from known Tor exit nodes.
resource "cloudflare_access_rule" "tor_exit_nodes" {
  notes = "Requests coming from known Tor exit nodes"
  mode = "challenge"
  configuration {
    target = "country"
    value = "T1"
  }
}

# Whitelist (sic!) requests coming from Antarctica, but only for single zone.
resource "cloudflare_access_rule" "antarctica" {
  notes = "Requests coming from known Tor exit nodes"
  mode = "whitelist"
  configuration {
    target = "country"
    value = "AQ"
  }
  zone = "example.com"
}

# Whitelist office's network IP ranges on all Organization's zones (or other lists of resources).
# Resulting Terraform state will be a list of resources.
provider "cloudflare" {
  # ... other provider configuration
  org_id = "d41d8cd98f00b204e9800998ecf8427e"
}
variable "my_office" {
  type = "list"
  default = ["192.0.2.0/24", "198.51.100.0/24", "2001:db8::/56"]
}
resource "cloudflare_access_rule" "office_network" {
  count = "${length(var.my_office)}"
  notes = "Requests coming from office network"
  mode = "whitelist"
  configuration {
    target = "ip_range"
    value = "${element(var.my_office, count.index)}"
  }
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Optional) The DNS zone to which the access rule should be added. Will be resolved to `zone_id` upon creation.
- `zone_id` - (Optional) The DNS zone to which the access rule should be added.

- `mode` - (Required) The action to apply to a matched request. Allowed values: "block", "challenge", "whitelist", "js_challenge"
- `notes` - (Optional) A personal note about the rule. Typically used as a reminder or explanation for the rule.
- `configuration` - (Required) Rule configuration to apply to a matched request. It's a complex value. See description below.

Note: If both `zone` and `zone_id` are empty, then access rule will be set to User's or Organization's account and apply to all their zones.

The **configuration** block supports:

- `target` - (Required) The request property to target. Allowed values: "ip", "ip_range", "asn", "country"
- `value` - (Required) The value to target. Depends on target's type.

Attributes Reference

The following attributes are exported:

- `id` - The access rule ID.
- `zone_id` - The DNS zone ID.

Import

Records can be imported using a composite ID formed of access rule type, access rule type identifier and identifer value, e.g.

```
$ terraform import cloudflare_access_rule.default zone/cb029e245cfdd66dc8d2e570d5dd3322/d41d8cd98f00b204e
9800998ecf8427e
```

where:

- `zone` - access rule type (`account`, `zone` or `user`)
- `cb029e245cfdd66dc8d2e570d5dd3322` - access rule type ID (i.e the zone ID or account ID you wish to target)
- `d41d8cd98f00b204e9800998ecf8427e` - access rule ID as returned by respective API endpoint for the type you are attempting to import.

cloudflare_account_member

Provides a resource which manages Cloudflare account members.

Example Usage

```
resource "cloudflare_account_member" "example_user" {
  email_address = "user@example.com"
  role_ids = [
    "68b329da9893e34099c7d8ad5cb9c940",
    "d784fa8b6d98d27699781bd9a7cf19f0"
  ]
}
```

Argument Reference

The following arguments are supported:

- `email_address` - (Required) The email address of the user who you wish to manage. Note: Following creation, this field becomes read only via the API and cannot be updated.
- `role_ids` - (Required) Array of account role IDs that you want to assign to a member.

Import

Account members can be imported using a composite ID formed of account ID and account member ID, e.g.

```
$ terraform import cloudflare_account_member.example_user d41d8cd98f00b204e9800998ecf8427e/b58c6f14d29255
6214bd64909bcdb118
```

where:

- `d41d8cd98f00b204e9800998ecf8427e` - account ID as returned by the API (https://api.cloudflare.com/#accounts-account-details)
- `b58c6f14d292556214bd64909bcdb118` - account member ID as returned by the API (https://api.cloudflare.com/#account-members-member-details)

cloudflare_custom_pages

Provides a resource which manages Cloudflare custom error pages.

Example Usage

```
resource "cloudflare_custom_pages" "basic_challenge" {
  zone_id = "d41d8cd98f00b204e9800998ecf8427e"
  type    = "basic_challenge"
  url     = "https://example.com/challenge.html"
  state   = "customized"
}
```

Argument Reference

The following arguments are supported:

- `zone_id` - (Optional) The zone ID where the custom pages should be updated. Either `zone_id` or `account_id` must be provided.
- `account_id` - (Optional) The account ID where the custom pages should be updated. Either `account_id` or `zone_id` must be provided. If `account_id` is present, it will override the zone setting.
- `type` - (Required) The type of custom page you wish to update. Must be one of `basic_challenge`, `waf_challenge`, `waf_block`, `ratelimit_block`, `country_challenge`, `ip_block`, `under_attack`, `500_errors`, `1000_errors`, `always_online`.
- `url` - (Required) URL of where the custom page source is located.
- `state` - (Required) Managed state of the custom page. Must be one of `default`, `customised`. If the value is `default` it will be removed from the Terraform state management.

Import

Custom pages can be imported using a composite ID formed of:

- `customPageLevel` - Either `account` or `zone`.
- `identifier` - The ID of the account or zone you intend to manage.
- `pageType` - The value from the `type` argument.

Example for a zone:

```
$ terraform import cloudflare_custom_pages.basic_challenge zone/d41d8cd98f00b204e9800998ecf8427e/basic_ch
allenge
```

Example for an account:

```
$ terraform import cloudflare_custom_pages.basic_challenge account/e268443e43d93dab7ebef303bbe9642f/basic
_challenge
```

cloudflare_filter

Filter expressions that can be referenced across multiple features, e.g. Firewall Rule (/docs/providers/cloudflare/r/firewall_rule.html). The expression format is similar to Wireshark Display Filter (https://www.wireshark.org/docs/man-pages/wireshark-filter.html).

Example Usage

```
resource "cloudflare_filter" "wordpress" {
  zone_id = "d41d8cd98f00b204e9800998ecf8427e"
  description = "Wordpress break-in attempts that are outside of the office"
  expression = "(http.request.uri.path ~ \".*wp-login.php\" or http.request.uri.path ~ \".*xmlrpc.php\")
and ip.src ne 192.0.2.1"
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Optional) The DNS zone to which the Filter should be added. Will be resolved to `zone_id` upon creation.
- `zone_id` - (Optional) The DNS zone to which the Filter should be added.
- `paused` - (Optional) Whether this filter is currently paused. Boolean value.
- `expression` - (Required) The filter expression to be used.
- `description` - (Optional) A note that you can use to describe the purpose of the filter.
- `ref` - (Optional) Short reference tag to quickly select related rules.

Attributes Reference

The following attributes are exported:

- `id` - Filter identifier.
- `zone_id` - The DNS zone ID.

Import

Filter can be imported using a composite ID formed of zone ID and filter ID, e.g.

```
$ terraform import cloudflare_filter.default d41d8cd98f00b204e9800998ecf8427e/9e107d9d372bb6826bd81d3542a
419d6
```

where:

- `d41d8cd98f00b204e9800998ecf8427e` - zone ID
- `9e107d9d372bb6826bd81d3542a419d6` - filter ID as returned by API (https://api.cloudflare.com/#zone-firewall-filters)

cloudflare_firewall_rule

Define Firewall rules using filter expressions for more control over how traffic is matched to the rule. A filter expression permits selecting traffic by multiple criteria allowing greater freedom in rule creation.

Filter expressions needs to be created first before using Firewall Rule. See Filter (/docs/providers/cloudflare/r/filter.html).

Example Usage

```
resource "cloudflare_filter" "wordpress" {
  zone_id = "d41d8cd98f00b204e9800998ecf8427e"
  description = "Wordpress break-in attempts that are outside of the office"
  expression = "(http.request.uri.path ~ \".*wp-login.php\" or http.request.uri.path ~ \".*xmlrpc.php\")
and ip.src ne 192.0.2.1"
}

resource "cloudflare_firewall_rule" "wodpress" {
  zone_id = "d41d8cd98f00b204e9800998ecf8427e"
  description = "Block wordpress break-in attempts"
  filter_id = "${cloudflare_filter.wordpress.id}"
  action = "block"
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Optional) The DNS zone to which the Firewall Rule should be added. Will be resolved to `zone_id` upon creation.
- `zone_id` - (Optional) The DNS zone to which the Filter should be added.
- `action` - (Required) The action to apply to a matched request. Allowed values: "block", "challenge", "allow", "js_challenge".
- `priority` - (Optional) The priority of the rule to allow control of processing order. A lower number indicates high priority. If not provided, any rules with a priority will be sequenced before those without.
- `paused` - (Optional) Whether this filter based firewall rule is currently paused. Boolean value.
- `expression` - (Required) The filter expression to be used.
- `description` - (Optional) A description of the rule to help identify it.

Attributes Reference

The following attributes are exported:

- `id` - Firewall Rule identifier.
- `zone_id` - The DNS zone ID.

Import

Firewall Rule can be imported using a composite ID formed of zone ID and rule ID, e.g.

```
$ terraform import cloudflare_filter.default d41d8cd98f00b204e9800998ecf8427e/9e107d9d372bb6826bd81d3542a
419d6
```

where:

- `d41d8cd98f00b204e9800998ecf8427e` - zone ID
- `9e107d9d372bb6826bd81d3542a419d6` - rule ID as returned by API (https://api.cloudflare.com/#zone-firewall-filter-rules)

cloudflare_load_balancer

Provides a Cloudflare Load Balancer resource. This sits in front of a number of defined pools of origins and provides various options for geographically-aware load balancing. Note that the load balancing feature must be enabled in your Clouflare account before you can use this resource.

Example Usage

```
# Define a load balancer which always points to a pool we define below
# In normal usage, would have different pools set for different pops (cloudflare points-of-presence) and/
or for different regions
# Within each pop or region we can define multiple pools in failover order
resource "cloudflare_load_balancer" "bar" {
  zone = "example.com"
  name = "example-load-balancer"
  fallback_pool_id = "${cloudflare_load_balancer_pool.foo.id}"
  default_pool_ids = ["${cloudflare_load_balancer_pool.foo.id}"]
  description = "example load balancer using geo-balancing"
  proxied = true
  steering_policy = "geo"
  pop_pools {
    pop = "LAX"
    pool_ids = ["${cloudflare_load_balancer_pool.foo.id}"]
  }
  region_pools {
    region = "WNAM"
    pool_ids = ["${cloudflare_load_balancer_pool.foo.id}"]
  }
}

resource "cloudflare_load_balancer_pool" "foo" {
  name = "example-lb-pool"
  origins {
    name = "example-1"
    address = "192.0.2.1"
    enabled = false
  }
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The zone to add the load balancer to.
- `name` - (Required) The DNS name to associate with the load balancer.
- `fallback_pool_id` - (Required) The pool ID to use when all other pools are detected as unhealthy.
- `default_pool_ids` - (Required) A list of pool IDs ordered by their failover priority. Used whenever region/pop pools are not defined.
- `description` - (Optional) Free text description.

- `ttl` - (Optional) Time to live (TTL) of this load balancer's DNS `name`. Conflicts with `proxied` - this cannot be set for proxied load balancers. Default is `30`.
- `steering_policy` - (Optional) Determine which method the load balancer uses to determine the fastest route to your origin. Valid values are: "off", "geo", "dynamic_latency" or "". Default is "".
- `proxied` - (Optional) Whether the hostname gets Cloudflare's origin protection. Defaults to `false`.
- `region_pools` - (Optional) A set containing mappings of region/country codes to a list of pool IDs (ordered by their failover priority) for the given region. Fields documented below.
- `pop_pools` - (Optional) A set containing mappings of Cloudflare Point-of-Presence (PoP) identifiers to a list of pool IDs (ordered by their failover priority) for the PoP (datacenter). This feature is only available to enterprise customers. Fields documented below.
- `session_affinity` - (Optional) Associates all requests coming from an end-user with a single origin. Cloudflare will set a cookie on the initial response to the client, such that consequent requests with the cookie in the request will go to the same origin, so long as it is available.

region_pools requires the following:

- `region` - (Required) A region code which must be in the list defined here (https://support.cloudflare.com/hc/en-us/articles/115000540888-Load-Balancing-Geographic-Regions). Multiple entries should not be specified with the same region.
- `pool_ids` - (Required) A list of pool IDs in failover priority to use in the given region.

pop_pools requires the following:

- `pop` - (Required) A 3-letter code for the Point-of-Presence. Allowed values can be found in the list of datacenters on the status page (https://www.cloudflarestatus.com/). Multiple entries should not be specified with the same PoP.
- `pool_ids` - (Required) A list of pool IDs in failover priority to use for traffic reaching the given PoP.

Attributes Reference

The following attributes are exported:

- `id` - Unique identifier in the API for the load balancer.
- `zone_id` - ID associated with the specified `zone`.
- `created_on` - The RFC3339 timestamp of when the load balancer was created.
- `modified_on` - The RFC3339 timestamp of when the load balancer was last modified.

cloudflare_load_balancer_monitor

If you're using Cloudflare's Load Balancing to load-balance across multiple origin servers or data centers, you configure one of these Monitors to actively check the availability of those servers over HTTP(S).

Example Usage

```
resource "cloudflare_load_balancer_monitor" "test" {
  expected_body = "alive"
  expected_codes = "2xx"
  method = "GET"
  timeout = 7
  path = "/health"
  interval = 60
  retries = 5
  description = "example load balancer"
  header {
    header = "Host"
    values = ["example.com"]
  }
}
```

Argument Reference

The following arguments are supported:

- `expected_body` - (Required) A case-insensitive sub-string to look for in the response body. If this string is not found, the origin will be marked as unhealthy.
- `expected_codes` - (Required) The expected HTTP response code or code range of the health check. Eg `2xx`
- `method` - (Optional) The HTTP method to use for the health check. Default: "GET".
- `timeout` - (Optional) The timeout (in seconds) before marking the health check as failed. Default: 5.
- `path` - (Optional) The endpoint path to health check against. Default: "/".
- `interval` - (Optional) The interval between each health check. Shorter intervals may improve failover time, but will increase load on the origins as we check from multiple locations. Default: 60.
- `retries` - (Optional) The number of retries to attempt in case of a timeout before marking the origin as unhealthy. Retries are attempted immediately. Default: 2.
- `header` - (Optional) The HTTP request headers to send in the health check. It is recommended you set a Host header by default. The User-Agent header cannot be overridden. Fields documented below.
- `type` - (Optional) The protocol to use for the healthcheck. Currently supported protocols are 'HTTP' and 'HTTPS'. Default: "http".
- `description` - (Optional) Free text description.

header requires the following:

- `header` - (Required) The header name.
- `values` - (Required) A list of string values for the header.

Attributes Reference

The following attributes are exported:

- `id` - Load balancer monitor ID.
- `created_on` - The RFC3339 timestamp of when the load balancer monitor was created.
- `modified_on` - The RFC3339 timestamp of when the load balancer monitor was last modified.

cloudflare_load_balancer_pool

Provides a Cloudflare Load Balancer pool resource. This provides a pool of origins that can be used by a Cloudflare Load Balancer. Note that the load balancing feature must be enabled in your Clouflare account before you can use this resource.

Example Usage

```
resource "cloudflare_load_balancer_pool" "foo" {
  name = "example-pool"
  origins {
    name = "example-1"
    address = "192.0.2.1"
    enabled = false
  }
  origins {
    name = "example-2"
    address = "192.0.2.2"
  }
  description = "example load balancer pool"
  enabled = false
  minimum_origins = 1
  notification_email = "someone@example.com"
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) A short name (tag) for the pool. Only alphanumeric characters, hyphens, and underscores are allowed.
- `origins` - (Required) The list of origins within this pool. Traffic directed at this pool is balanced across all currently healthy origins, provided the pool itself is healthy. It's a complex value. See description below.
- `check_regions` - (Optional) A list of regions (specified by region code) from which to run health checks. Empty means every Cloudflare data center (the default), but requires an Enterprise plan. Region codes can be found here (https://support.cloudflare.com/hc/en-us/articles/115000540888-Load-Balancing-Geographic-Regions).
- `description` - (Optional) Free text description.
- `enabled` - (Optional) Whether to enable (the default) this pool. Disabled pools will not receive traffic and are excluded from health checks. Disabling a pool will cause any load balancers using it to failover to the next pool (if any).
- `minimum_origins` - (Optional) The minimum number of origins that must be healthy for this pool to serve traffic. If the number of healthy origins falls below this number, the pool will be marked unhealthy and we will failover to the next available pool. Default: 1.
- `monitor` - (Optional) The ID of the Monitor to use for health checking origins within this pool.
- `notification_email` - (Optional) The email address to send health status notifications to. This can be an individual mailbox or a mailing list.

The **origins** block supports:

- `name` - (Required) A human-identifiable name for the origin.
- `address` - (Required) The IP address (IPv4 or IPv6) of the origin, or the publicly addressable hostname. Hostnames entered here should resolve directly to the origin, and not be a hostname proxied by Cloudflare.
- `weight` - (Optional) The weight (0.01 - 1.00) of this origin, relative to other origins in the pool. Equal values mean equal weighting. A weight of 0 means traffic will not be sent to this origin, but health is still checked. Default: 1.
- `enabled` - (Optional) Whether to enable (the default) this origin within the Pool. Disabled origins will not receive traffic and are excluded from health checks. The origin will only be disabled for the current pool.

Attributes Reference

The following attributes are exported:

- `id` - ID for this load balancer pool.
- `created_on` - The RFC3339 timestamp of when the load balancer was created.
- `modified_on` - The RFC3339 timestamp of when the load balancer was last modified.

cloudflare_page_rule

Provides a Cloudflare page rule resource.

Example Usage

```
# Add a page rule to the domain
resource "cloudflare_page_rule" "foobar" {
  zone = "${var.cloudflare_zone}"
  target = "sub.${var.cloudflare_zone}/page"
  priority = 1

  actions = {
    ssl = "flexible"
    email_obfuscation = "on"
  }
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The DNS zone to which the page rule should be added.
- `target` - (Required) The URL pattern to target with the page rule.
- `actions` - (Required) The actions taken by the page rule, options given below.
- `priority` - (Optional) The priority of the page rule among others for this target.
- `status` - (Optional) Whether the page rule is active or disabled.

Action blocks support the following:

- `always_online` - (Optional) Whether this action is `"on"` or `"off"`.
- `always_use_https` - (Optional) Boolean of whether this action is enabled. Default: false.
- `automatic_https_rewrites` - (Optional) Whether this action is `"on"` or `"off"`.
- `browser_cache_ttl` - (Optional) The Time To Live for the browser cache.
- `browser_check` - (Optional) Whether this action is `"on"` or `"off"`.
- `cache_level` - (Optional) Whether to set the cache level to `"bypass"`, `"basic"`, `"simplified"`, `"aggressive"`, or `"cache_everything"`.
- `disable_apps` - (Optional) Boolean of whether this action is enabled. Default: false.
- `disable_performance` - (Optional) Boolean of whether this action is enabled. Default: false.
- `disable_railgun` - (Optional) Boolean of whether this action is enabled. Default: false.
- `disable_security` - (Optional) Boolean of whether this action is enabled. Default: false.

- `edge_cache_ttl` - (Optional) The Time To Live for the edge cache.
- `email_obfuscation` - (Optional) Whether this action is `"on"` or `"off"`.
- `forwarding_url` - (Optional) The URL to forward to, and with what status. See below.
- `host_header_override` - (Optional) Value of the Host header to send.
- `ip_geolocation` - (Optional) Whether this action is `"on"` or `"off"`.
- `mirage` - (Optional) Whether this action is `"on"` or `"off"`.
- `opportunistic_encryption` - (Optional) Whether this action is `"on"` or `"off"`.
- `polish` - (Optional) Whether this action is `"off"`, `"lossless"` or `"lossy"`.
- `resolve_override` - (Optional) Overridden origin server name.
- `rocket_loader` - (Optional) Whether to set the rocket loader to `"on"`, `"off"`.
- `security_level` - (Optional) Whether to set the security level to `"off"`, `"essentially_off"`, `"low"`, `"medium"`, `"high"`, or `"under_attack"`.
- `server_side_exclude` - (Optional) Whether this action is `"on"` or `"off"`.
- `smart_errors` - (Optional) Whether this action is `"on"` or `"off"`.
- `ssl` - (Optional) Whether to set the SSL mode to `"off"`, `"flexible"`, `"full"`, or `"strict"`.
- `waf` - (Optional) Whether this action is `"on"` or `"off"`.

Forwarding URL actions support the following:

- `url` - (Required) The URL to which the page rule should forward.
- `status_code` - (Required) The status code to use for the redirection.

Attributes Reference

The following attributes are exported:

- `id` - The page rule ID.
- `zone_id` - The ID of the zone in which the page rule will be applied.
- `target` - The URL pattern targeted by the page rule.
- `actions` - The actions applied by the page rule.
- `priority` - The priority of the page rule.
- `status` - Whether the page rule is active or disabled.

Import

Page rules can be imported using a composite ID formed of zone name and page rule ID, e.g.

```
$ terraform import cloudflare_page_rule.default example.com/ch8374ftwdghsif43
```

cloudflare_rate_limit

Provides a Cloudflare rate limit resource for a given zone. This can be used to limit the traffic you receive zone-wide, or matching more specific types of requests/responses.

Example Usage

```
resource "cloudflare_rate_limit" "example" {
  zone = "${var.cloudflare_zone}"
  threshold = 2000
  period = 2
  match {
    request {
      url_pattern = "${var.cloudflare_zone}/*"
      schemes = ["HTTP", "HTTPS"]
      methods = ["GET", "POST", "PUT", "DELETE", "PATCH", "HEAD"]
    }
    response {
      statuses = [200, 201, 202, 301, 429]
      origin_traffic = false
    }
  }
  action {
    mode = "simulate"
    timeout = 43200
    response {
      content_type = "text/plain"
      body = "custom response body"
    }
  }
  correlate {
    by = "nat"
  }
  disabled = false
  description = "example rate limit for a zone"
  bypass_url_patterns = ["${var.cloudflare_zone}/bypass1","${var.cloudflare_zone}/bypass2"]
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The DNS zone to apply rate limiting to.
- `threshold` - (Required) The threshold that triggers the rate limit mitigations, combine with period. i.e. threshold per period (min: 2, max: 1,000,000).
- `period` - (Required) The time in seconds to count matching traffic. If the count exceeds threshold within this period the action will be performed (min: 1, max: 86,400).
- `action` - (Required) The action to be performed when the threshold of matched traffic within the period defined is exceeded.
- `match` - (Optional) Determines which traffic the rate limit counts towards the threshold. By default matches all traffic in

the zone. See definition below.

- `disabled` - (Optional) Whether this ratelimit is currently disabled. Default: `false`.
- `description` - (Optional) A note that you can use to describe the reason for a rate limit. This value is sanitized and all tags are removed.
- `bypass_url_patterns` - (Optional) URLs matching the patterns specified here will be excluded from rate limiting.
- `correlate` - (Optional) Determines how rate limiting is applied. By default if not specified, rate limiting applies to the clients IP address.

The **match** block supports:

- `request` - (Optional) Matches HTTP requests (from the client to Cloudflare). See definition below.
- `response` (Optional) Matches HTTP responses before they are returned to the client from Cloudflare. If this is defined, then the entire counting of traffic occurs at this stage. This field is not required.

The **match.request** block supports:

- `methods` - (Optional) HTTP Methods, can be a subset ['POST','PUT'] or all ['_ALL_']. Default: ['_ALL_'].
- `schemes` - (Optional) HTTP Schemes, can be one ['HTTPS'], both ['HTTP','HTTPS'] or all ['_ALL_']. Default: ['_ALL_'].
- `url_pattern` - (Optional) The URL pattern to match comprised of the host and path, i.e. example.org/path. Wildcard are expanded to match applicable traffic, query strings are not matched. Use * for all traffic to your zone. Default: '*'.

The **match.response** block supports:

- `statuses` - (Optional) HTTP Status codes, can be one [403], many [401,403] or indicate all by not providing this value.
- `origin_traffic` - (Optional) Only count traffic that has come from your origin servers. If true, cached items that Cloudflare serve will not count towards rate limiting. Default: `true`.

The **action** block supports:

- `mode` - (Required) The type of action to perform. Allowable values are 'simulate' and 'ban'.
- `timeout` - (Required) The time in seconds as an integer to perform the mitigation action. Must be the same or greater than the period (min: 1, max: 86400).
- `response` - (Optional) Custom content-type and body to return, this overrides the custom error for the zone. This field is not required. Omission will result in default HTML error page. Definition below.

The **action.response** block supports:

- `content_type` - (Required) The content-type of the body, must be one of: 'text/plain', 'text/xml', 'application/json'.
- `body` - (Required) The body to return, the content here should conform to the content_type.

The **correlate** block supports:

- `by` - (Optional) If set to 'nat', NAT support will be enabled for rate limiting.

Attributes Reference

The following attributes are exported:

- `id` - The Rate limit ID.
- `zone_id` - The DNS zone ID.

Import

Rate limits can be imported using a composite ID formed of zone name and rate limit ID, e.g.

```
$ terraform import cloudflare_rate_limit.default example.com/ch8374ftwdghsif43
```

cloudflare_record

Provides a Cloudflare record resource.

Example Usage

```
# Add a record to the domain
resource "cloudflare_record" "foobar" {
  domain = "${var.cloudflare_zone}"
  name   = "terraform"
  value  = "192.168.0.11"
  type   = "A"
  ttl    = 3600
}
```

Argument Reference

The following arguments are supported:

- `domain` - (Required) The DNS zone to add the record to
- `name` - (Required) The name of the record
- `type` - (Required) The type of the record
- `value` - (Optional) The (string) value of the record. Either this or `data` must be specified
- `data` - (Optional) Map of attributes that constitute the record value. Primarily used for LOC and SRV record types. Either this or `value` must be specified
- `ttl` - (Optional) The TTL of the record (automatic: '1' (https://api.cloudflare.com/#dns-records-for-a-zone-create-dns-record))
- `priority` - (Optional) The priority of the record
- `proxied` - (Optional) Whether the record gets Cloudflare's origin protection; defaults to `false`.

Attributes Reference

The following attributes are exported:

- `id` - The record ID
- `hostname` - The FQDN of the record
- `proxiable` - Shows whether this record can be proxied, must be true if setting `proxied=true`
- `created_on` - The RFC3339 timestamp of when the record was created
- `modified_on` - The RFC3339 timestamp of when the record was last modified

- `metadata` - A key-value map of string metadata cloudflare associates with the record
- `zone_id` - The zone id of the record

Import

Records can be imported using a composite ID formed of zone name and record ID, e.g.

```
$ terraform import cloudflare_record.default example.com/d41d8cd98f00b204e9800998ecf8427e
```

where:

- `example.com` - the zone name
- `d41d8cd98f00b204e9800998ecf8427e` - record ID as returned by API (https://api.cloudflare.com/#dns-records-for-a-zone-list-dns-records)

cloudflare_waf_rule

Provides a Cloudflare WAF rule resource for a particular zone. This can be used to configure firewall behaviour for pre-defined firewall rules.

Example Usage

```
resource "cloudflare_waf_rule" "100000" {
  rule_id = "100000"
  zone = "domain.com"
  mode = "simulate"
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The DNS zone to apply to.
- `rule_id` - (Required) The WAF Rule ID.
- `mode` - (Required) The mode of the rule, can be one of ["block", "challenge", "default", "disable, "simulate"].

Attributes Reference

The following attributes are exported:

- `id` - The WAF Rule ID, the same as rule_id.
- `zone_id` - The DNS zone ID.
- `package_id` - The ID of the WAF Rule Package that contains the rule.

Import

Rules can be imported using a composite ID formed of zone name and the WAF Rule ID, e.g.

```
$ terraform import cloudflare_waf_rule.100000 example.com/100000
```

cloudflare_worker_route

Provides a Cloudflare worker route resource. A route will also require a `cloudflare_worker_script`.

Example Usage

NOTE: This is for non-enterprise accounts where there is one script per zone. The `enabled` flag determines whether to run the worker script for a request that matches the specified `pattern`. For enterprise accounts, see the "multi-script" example below.

```
# Enables the zone's worker script for all URLs that match `example.com/*`
resource "cloudflare_worker_route" "my_route" {
  zone = "example.com"
  pattern = "example.com/*"
  enabled = true

  # it's recommended to set `depends_on` to point to the cloudflare_worker_script
  # resource in order to make sure that the script is uploaded before the route
  # is created
  depends_on = ["cloudflare_worker_script.my_script"]
}

resource "cloudflare_worker_script" "my_script" {
  # see "cloudflare_worker_script" documentation ...
}
```

Multi-script example usage

NOTE: This is only for enterprise accounts. With multi-script, each route points to a particular script instead of setting an `enabled` flag

```
# Runs the specified worker script for all URLs that match `example.com/*`
resource "cloudflare_worker_route" "my_route" {
  zone = "example.com"
  pattern = "example.com/*"
  script_name = "${cloudflare_worker_script.my_script.name}"
}

resource "cloudflare_worker_script" "my_script" {
  # see "cloudflare_worker_script" documentation ...
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The zone to add the route to.
- `pattern` - (Required) The route pattern (https://developers.cloudflare.com/workers/api/route-matching/)

- `enabled` (For single-script accounts only) Whether to run the worker script for requests that match the route pattern. Default is `false`
- `script_name` (For multi-script accounts only) Which worker script to run for requests that match the route pattern. If `script_name` is empty, workers will be skipped for matching requests.

Attributes Reference

The following attributes are exported:

- `zone_id` - The zone id of the route

Import

Records can be imported using a composite ID formed of zone name and route ID, e.g.

```
$ terraform import cloudflare_worker_route.default example.com/9a7806061c88ada191ed06f989cc3dac
```

where:

- `9a7806061c88ada191ed06f989cc3dac` - route ID as returned by API (https://api.cloudflare.com/#worker-filters-list-filters)

cloudflare_worker_script

Provides a Cloudflare worker script resource. In order for a script to be active, you'll also need to setup a `cloudflare_worker_route`.

Example Usage

NOTE: This is for non-enterprise accounts where there is one script per zone. For enterprise accounts, see the "multi-script" example below.

```
# Sets the script for the example.com zone
resource "cloudflare_worker_script" "my_script" {
  zone = "example.com"
  content = "${file("script.js")}"
}
```

Multi-script example usage

NOTE: This is only for enterprise accounts. With multi-script, each script is given a `name` instead of a `zone`

```
# Sets the script with the name "script_1"
resource "cloudflare_worker_script" "my_script" {
  name = "script_1"
  content = "${file("script.js")}"
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required for single-script accounts) The zone for the script.
- `name` - (Required for multi-script accounts) The name for the script.
- `content` - (Required) The script content.

Attributes Reference

The following attributes are exported:

- `zone_id` - The zone id of the script (only for non-multi-script resources)

Import

single-script

To import a script from a single-script account, use an id like `zone:example.com`

```
$ terraform import cloudflare_worker_script.default zone:example.com
```

where:

- `example.com` - the zone name

multi-script

To import a script from a multi-script account, use an id like `name:script_name`

```
$ terraform import cloudflare_worker_script.default name:script_name
```

where:

- `script_name` - the script name

cloudflare_zone

Provides a Cloudflare Zone resource. Zone is the basic resource for working with Cloudflare and is roughly equivalent to a domain name that the user purchases.

Example Usage

```
resource "cloudflare_zone" "example" {
    zone = "example.com"
}
```

Argument Reference

The following arguments are supported:

- `zone` - (Required) The DNS zone name which will be added.
- `paused` - (Optional) Boolean of whether this zone is paused (traffic bypasses Cloudflare). Default: false.
- `jump_start` - (Optional) Boolean of whether to scan for DNS records on creation. Ignored after zone is created. Default: false.
- `plan` - (Optional) The name of the commercial plan to apply to the zone, can be updated once the one is created; one of `free`, `pro`, `business`, `enterprise`.

Attributes Reference

The following attributes are exported:

- `id` - The zone ID.
- `plan` - The name of the commercial plan to apply to the zone.
- `vanity_name_servers` - List of Vanity Nameservers (if set).
- `meta.wildcard_proxiable` - Indicates whether wildcard DNS records can receive Cloudflare security and performance features.
- `meta.phishing_detected` - Indicates if URLs on the zone have been identified as hosting phishing content.
- `status` - Status of the zone. Valid values: `active`, `pending`, `initializing`, `moved`, `deleted`, `deactivated`
- `type` - A full zone implies that DNS is hosted with Cloudflare. A partial zone is typically a partner-hosted zone or a CNAME setup. Valid values: `full`, `partial`
- `name_servers` - Cloudflare-assigned name servers. This is only populated for zones that use Cloudflare DNS.

Import

Zone resource can be imported using a zone ID, e.g.

```
$ terraform import cloudflare_zone.example d41d8cd98f00b204e9800998ecf8427e
```

where:

- `d41d8cd98f00b204e9800998ecf8427e` - zone ID, as returned from API (https://api.cloudflare.com/#zone-list-zones)

cloudflare_zone_lockdown

Provides a Cloudflare Zone Lockdown resource. Zone Lockdown allows you to define one or more URLs (with wildcard matching on the domain or path) that will only permit access if the request originates from an IP address that matches a safelist of one or more IP addresses and/or IP ranges.

Example Usage

```
# Restrict access to these endpoints to requests from a known IP address.
resource "cloudflare_zone_lockdown" "endpoint_lockdown" {
  zone        = "api.mysite.com"
  paused      = "false"
  description = "Restrict access to these endpoints to requests from a known IP address"
  urls = [
    "api.mysite.com/some/endpoint*",
  ]
  configurations = [
    {
      "target" = "ip"
      "value" = "198.51.100.4"
    },
  ]
}
```

Argument Reference

The following arguments are supported:

- `zone` - The DNS zone to which the lockdown will be added. Will be resolved to `zone_id` upon creation.
- `zone_id` - The DNS zone to which the access rule should be added.
- `description` - (Optional) A description about the lockdown entry. Typically used as a reminder or explanation for the lockdown.
- `urls` - (Required) A list of simple wildcard patterns to match requests against. The order of the urls is unimportant.
- `configurations` - (Required) A list of IP addresses or IP ranges to match the request against specified in target, value pairs. It's a complex value. See description below. The order of the configuration entries is unimportant.
- `paused` - (Optional) Boolean of whether this zone lockdown is currently paused. Default: false.

Note: Either `zone` or `zone_id` is required and `zone` will be resolved to `zone_id` upon creation.

The list item in **configurations** block supports:

- `target` - (Required) The request property to target. Allowed values: "ip", "ip_range"
- `value` - (Required) The value to target. Depends on target's type. IP addresses should just be standard IPv4/IPv6 notation i.e. `198.51.100.4` or `2001:db8::/32` and IP ranges in CIDR format i.e. `198.51.0.0/16`.

Attributes Reference

The following attributes are exported:

- `id` - The access rule ID.

Import

Records can be imported using a composite ID formed of zone name and record ID, e.g.

```
$ terraform import cloudflare_zone_lockdown  api.mysite.com/d41d8cd98f00b204e9800998ecf8427e
```

where:

- `d41d8cd98f00b204e9800998ecf8427e` - zone lockdown ID as returned by API (https://api.cloudflare.com/#zone-lockdown-list-lockdown-rules)

cloudflare_zone_settings_override

Provides a resource which customizes Cloudflare zone settings. Note that after destroying this resource Zone Settings will be reset to their initial values.

Example Usage

```
resource "cloudflare_zone_settings_override" "test" {
    name = "${var.cloudflare_zone}"
    settings {
        brotli = "on"
        challenge_ttl = 2700
        security_level = "high"
        opportunistic_encryption = "on"
        automatic_https_rewrites = "on"
        mirage = "on"
        waf = "on"
        minify {
            css = "on"
            js = "off"
            html = "off"
        }
        security_header {
            enabled = true
        }
    }
}
```

Argument Reference

The following arguments are supported:

- `name` - (Required) The DNS zone to which apply settings.
- `settings` - (Optional) Settings overrides that will be applied to the zone. If a setting is not specified the existing setting will be used. For a full list of available settings see below.

The **settings** block supports settings that may be applied to the zone. These may be on/off values, unitary fields, string values, integers or nested objects.

On/Off Values

These can be specified as "on" or "off" string. Similar to boolean values, but here the empty string also means to use the existing value. Attributes available:

- `advanced_ddos`
- `always_online`
- `always_use_https`
- `automatic_https_rewrites`

- `brotli`
- `browser_check`
- `cache_level`
- `development_mode`
- `email_obfuscation`
- `hotlink_protection`
- `http2`
- `ip_geolocation`
- `ipv6`
- `mirage`
- `opportunistic_encryption`
- `opportunistic_onion`
- `origin_error_page_pass_thru`
- `prefetch_preload`
- `privacy_pass`
- `response_buffering`
- `rocket_loader`
- `server_side_exclude`
- `sha1_support`
- `sort_query_string_for_cache`
- `tls_1_2_only`
- `tls_client_auth`
- `true_client_ip_header`
- `waf`
- `webp`. Note that the value specified will be ignored unless `polish` is turned on (i.e. is "lossless" or "lossy")
- `websockets`

String Values

- `cache_level`. Allowed values: "aggressive", "basic", "simplified".
- `cname_flattening`. Allowed values: "flatten_at_root", "flatten_all", "flatten_none".
- `min_tls_version`. Allowed values: "1.0", "1.1", "1.2", "1.3".

- `polish`. Allowed values: "off", "lossless", "lossy".
- `pseudo_ipv4`. Allowed values: "off", "add_header", "overwrite_header".
- `security_level`. Allowed values: "off" (Enterprise only), "essentially_off", "low", "medium", "high", "under_attack".
- `ssl`. Allowed values: "off", "flexible", "full", "strict", "origin_pull".
- `tls_1_3`. Allowed values: "off", "on", "zrt".

Integer Values

- `browser_cache_ttl`
- `challenge_ttl`
- `edge_cache_ttl`
- `max_upload`

Nested Objects

- `minify`
- `mobile_redirect`
- `security_header`

The **minify** attribute supports the following fields:

- `css` (Required) "on"/"off"
- `html` (Required) "on"/"off"
- `js` (Required)"on"/"off"

The **mobile_redirect** attribute supports the following fields:

- `mobile_subdomain` (Required) String value
- `status` (Required) "on"/"off"
- `strip_uri` (Required) true/false

The **security_header** attribute supports the following fields:

- `enabled` (Optional) true/false
- `preload` (Optional) true/false
- `max_age` (Optional) Integer
- `include_subdomains` (Optional) true/false
- `nosniff` (Optional) true/false

Attributes Reference

The following attributes are exported:

- `id` - The zone ID.
- `initial_settings` - Settings present in the zone at the time the resource is created. This will be used to restore the original settings when this resource is destroyed. Shares the same schema as the `settings` attribute (Above).
- `intial_settings_read_at` - Time when this resource was created and the `initial_settings` were set.
- `readonly_settings` - Which of the current `settings` are not able to be set by the user. Which settings these are is determined by plan level and user permissions.
- `zone_status`. A full zone implies that DNS is hosted with Cloudflare. A partial zone is typically a partner-hosted zone or a CNAME setup.
- `zone_type`. Status of the zone. Valid values: active, pending, initializing, moved, deleted, deactivated.