» fastly_ip_ranges

Use this data source to get the IP ranges of Fastly edge nodes.

» Example Usage

```
data "fastly_ip_ranges" "fastly" {}

resource "aws_security_group" "from_fastly" {
  name = "from_fastly"

  ingress {
    from_port   = "443"
    to_port     = "443"
    protocol    = "tcp"
    cidr_blocks = ["${data.fastly_ip_ranges.fastly.cidr_blocks}"]
  }
}
```

» Attributes Reference

- `cidr_blocks` - The lexically ordered list of CIDR blocks.

» fastly_service_v1

Provides a Fastly Service, representing the configuration for a website, app, API, or anything else to be served through Fastly. A Service encompasses Domains and Backends.

The Service resource requires a domain name that is correctly set up to direct traffic to the Fastly service. See Fastly's guide on Adding CNAME Records on their documentation site for guidance.

» Example Usage

Basic usage:

```
resource "fastly_service_v1" "demo" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
```

```
    comment = "demo"
  }

  backend {
    address = "127.0.0.1"
    name    = "localhost"
    port    = 80
  }

  force_destroy = true
}
```

Basic usage with an Amazon S3 Website and that removes the `x-amz-request-id` header:

```
resource "fastly_service_v1" "demo" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
    comment = "demo"
  }

  backend {
    address = "demo.notexample.com.s3-website-us-west-2.amazonaws.com"
    name    = "AWS S3 hosting"
    port    = 80
  }

  header {
    destination = "http.x-amz-request-id"
    type        = "cache"
    action      = "delete"
    name        = "remove x-amz-request-id"
  }

  gzip {
    name          = "file extensions and content types"
    extensions    = ["css", "js"]
    content_types = ["text/html", "text/css"]
  }

  default_host = "${aws_s3_bucket.website.name}.s3-website-us-west-2.amazonaws.com"

  force_destroy = true
}
```

```
resource "aws_s3_bucket" "website" {
  bucket = "demo.notexample.com"
  acl    = "public-read"

  website {
    index_document = "index.html"
    error_document = "error.html"
  }
}
```

Basic usage with custom VCL (must be enabled on your Fastly account):

```
resource "fastly_service_v1" "demo" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
    comment = "demo"
  }

  backend {
    address = "127.0.0.1"
    name    = "localhost"
    port    = 80
  }

  force_destroy = true

  vcl {
    name    = "my_custom_main_vcl"
    content = "${file("${path.module}/my_custom_main.vcl")}"
    main    = true
  }

  vcl {
    name    = "my_custom_library_vcl"
    content = "${file("${path.module}/my_custom_library.vcl")}"
  }
}
```

Basic usage with custom Director:

```
resource "fastly_service_v1" "demo" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
    comment = "demo"
```

```
  }

  backend {
    address = "127.0.0.1"
    name    = "origin1"
    port    = 80
  }

  backend {
    address = "127.0.0.2"
    name    = "origin2"
    port    = 80
  }

  director {
    name = "mydirector"
    quorum = 0
    type = 3
    backends = [ "origin1", "origin2" ]
  }

  force_destroy = true
}
```

Note: For an AWS S3 Bucket, the Backend address is `<domain>.s3-website-<region>.amazonaws.com`. The `default_host` attribute should be set to `<bucket_name>.s3-website-<region>.amazonaws.com`. See the Fastly documentation on Amazon S3.

» Argument Reference

The following arguments are supported:

- `activate` - (Optional) Conditionally prevents the Service from being activated. The apply step will continue to create a new draft version but will not activate it if this is set to false. Default true.
- `name` - (Required) The unique name for the Service to create.
- `comment` - (Optional) Description field for the service. Default `Managed by Terraform`.
- `version_comment` - (Optional) Description field for the version.
- `domain` - (Required) A set of Domain names to serve as entry points for your Service. Defined below.
- `backend` - (Optional) A set of Backends to service requests from your Domains. Defined below. Backends must be defined in this argument, or defined in the `vcl` argument below
- `condition` - (Optional) A set of conditions to add logic to any basic configuration object in this service. Defined below.

- `cache_setting` - (Optional) A set of Cache Settings, allowing you to override
- `director` - (Optional) A director to allow more control over balancing traffic over backends. when an item is not to be cached based on an above `condition`. Defined below
- `gzip` - (Required) A set of gzip rules to control automatic gzipping of content. Defined below.
- `header` - (Optional) A set of Headers to manipulate for each request. Defined below.
- `healthcheck` - (Optional) Automated healthchecks on the cache that can change how Fastly interacts with the cache based on its health.
- `default_host` - (Optional) The default hostname.
- `default_ttl` - (Optional) The default Time-to-live (TTL) for requests.
- `force_destroy` - (Optional) Services that are active cannot be destroyed. In order to destroy the Service, set `force_destroy` to `true`. Default `false`.
- `request_setting` - (Optional) A set of Request modifiers. Defined below
- `s3logging` - (Optional) A set of S3 Buckets to send streaming logs too. Defined below.
- `papertrail` - (Optional) A Papertrail endpoint to send streaming logs too. Defined below.
- `sumologic` - (Optional) A Sumologic endpoint to send streaming logs too. Defined below.
- `gcslogging` - (Optional) A gcs endpoint to send streaming logs too. Defined below.
- `bigquerylogging` - (Optional) A BigQuery endpoint to send streaming logs too. Defined below.
- `syslog` - (Optional) A syslog endpoint to send streaming logs too. Defined below.
- `logentries` - (Optional) A logentries endpoint to send streaming logs too. Defined below.
- `splunk` - (Optional) A Splunk endpoint to send streaming logs too. Defined below.
- `blobstoragelogging` - (Optional) An Azure Blob Storage endpoint to send streaming logs too. Defined below.
- `response_object` - (Optional) Allows you to create synthetic responses that exist entirely on the varnish machine. Useful for creating error or maintenance pages that exists outside the scope of your datacenter. Best when used with Condition objects.
- `snippet` - (Optional) A set of custom, "regular" (non-dynamic) VCL Snippet configuration blocks. Defined below.
- `dynamicsnippet` - (Optional) A set of custom, "dynamic" VCL Snippet configuration blocks. Defined below.
- `vcl` - (Optional) A set of custom VCL configuration blocks. The ability to upload custom VCL code is not enabled by default for new Fastly accounts (see the Fastly documentation for details).

- `acl` - (Optional) A set of ACL configuration blocks. Defined below.
- `dictionary` - (Optional) A set of dictionaries that allow the storing of key values pair for use within VCL functions. Defined below.

The `domain` block supports:

- `name` - (Required) The domain to which this Service will respond.
- `comment` - (Optional) An optional comment about the Domain.

The `backend` block supports:

- `name` - (Required, string) Name for this Backend. Must be unique to this Service.
- `address` - (Required, string) An IPv4, hostname, or IPv6 address for the Backend.
- `auto_loadbalance` - (Optional, boolean) Denotes if this Backend should be included in the pool of backends that requests are load balanced against. Default `true`.
- `between_bytes_timeout` - (Optional) How long to wait between bytes in milliseconds. Default `10000`.
- `connect_timeout` - (Optional) How long to wait for a timeout in milliseconds. Default `1000`
- `error_threshold` - (Optional) Number of errors to allow before the Backend is marked as down. Default `0`.
- `first_byte_timeout` - (Optional) How long to wait for the first bytes in milliseconds. Default `15000`.
- `max_conn` - (Optional) Maximum number of connections for this Backend. Default `200`.
- `port` - (Optional) The port number on which the Backend responds. Default `80`.
- `override_host` - (Optional) The hostname to override the Host header.
- `request_condition` - (Optional, string) Name of already defined `condition`, which if met, will select this backend during a request.
- `use_ssl` - (Optional) Whether or not to use SSL to reach the backend. Default `false`.
- `max_tls_version` - (Optional) Maximum allowed TLS version on SSL connections to this backend.
- `min_tls_version` - (Optional) Minimum allowed TLS version on SSL connections to this backend.
- `ssl_ciphers` - (Optional) Comma separated list of OpenSSL Ciphers to try when negotiating to the backend.
- `ssl_ca_cert` - (Optional) CA certificate attached to origin.
- `ssl_client_cert` - (Optional) Client certificate attached to origin. Used when connecting to the backend.
- `ssl_client_key` - (Optional) Client key attached to origin. Used when connecting to the backend.
- `ssl_check_cert` - (Optional) Be strict about checking SSL certs. Default `true`.

- `ssl_hostname` - (Optional, deprecated by Fastly) Used for both SNI during the TLS handshake and to validate the cert.
- `ssl_cert_hostname` - (Optional) Overrides ssl_hostname, but only for cert verification. Does not affect SNI at all.
- `ssl_sni_hostname` - (Optional) Overrides ssl_hostname, but only for SNI in the handshake. Does not affect cert validation at all.
- `shield` - (Optional) The POP of the shield designated to reduce inbound load.
- `weight` - (Optional) The portion of traffic to send to this Backend. Each Backend receives `weight / total` of the traffic. Default `100`.
- `healthcheck` - (Optional) Name of a defined `healthcheck` to assign to this backend.

The `condition` block supports allows you to add logic to any basic configuration object in a service. See Fastly's documentation "About Conditions" for more detailed information on using Conditions. The Condition `name` can be used in the `request_condition`, `response_condition`, or `cache_condition` attributes of other block settings.

- `name` - (Required) The unique name for the condition.
- `statement` - (Required) The statement used to determine if the condition is met.
- `type` - (Required) Type of condition, either `REQUEST` (req), `RESPONSE` (req, resp), or `CACHE` (req, beresp).
- `priority` - (Optional) A number used to determine the order in which multiple conditions execute. Lower numbers execute first. Default `10`.

The `director` block supports:

- `name` - (Required) Unique name for this Director.
- `backends` - (Required) Names of defined backends to map the director to. Example: `[ "origin1", "origin2" ]`
- `comment` - (Optional) An optional comment about the Director.
- `shield` - (Optional) Selected POP to serve as a "shield" for origin servers.
- `capacity` - (Optional) Load balancing weight for the backends. Default `100`.
- `quorum` - (Optional) Percentage of capacity that needs to be up for the director itself to be considered up. Default `75`.
- `type` - (Optional) Type of load balance group to use. Integer, 1 to 4. Values: `1` (random), `3` (hash), `4` (client). Default `1`.
- `retries` - (Optional) How many backends to search if it fails. Default `5`.

The `cache_setting` block supports:

- `name` - (Required) Unique name for this Cache Setting.
- `action` - (Optional) One of `cache`, `pass`, or `restart`, as defined on Fastly's documentation under "Caching action descriptions".
- `cache_condition` - (Optional) Name of already defined `condition` used to test whether this settings object should be used. This `condition` must

be of type `CACHE`.
- `stale_ttl` - (Optional) Max "Time To Live" for stale (unreachable) objects.
- `ttl` - (Optional) The Time-To-Live (TTL) for the object.

The `gzip` block supports:

- `name` - (Required) A unique name.
- `content_types` - (Optional) The content-type for each type of content you wish to have dynamically gzip'ed. Example: `["text/html", "text/css"]`.
- `extensions` - (Optional) File extensions for each file type to dynamically gzip. Example: `["css", "js"]`.
- `cache_condition` - (Optional) Name of already defined `condition` controlling when this gzip configuration applies. This `condition` must be of type `CACHE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.

The `header` block supports adding, removing, or modifying Request and Response headers. See Fastly's documentation on Adding or modifying headers on HTTP requests and responses for more detailed information on any of the properties below.

- `name` - (Required) Unique name for this header attribute.
- `action` - (Required) The Header manipulation action to take; must be one of `set`, `append`, `delete`, `regex`, or `regex_repeat`.
- `type` - (Required) The Request type on which to apply the selected Action; must be one of `request`, `fetch`, `cache` or `response`.
- `destination` - (Required) The name of the header that is going to be affected by the Action.
- `ignore_if_set` - (Optional) Do not add the header if it is already present. (Only applies to the `set` action.). Default `false`.
- `source` - (Optional) Variable to be used as a source for the header content. (Does not apply to the `delete` action.)
- `regex` - (Optional) Regular expression to use (Only applies to the `regex` and `regex_repeat` actions.)
- `substitution` - (Optional) Value to substitute in place of regular expression. (Only applies to the `regex` and `regex_repeat` actions.)
- `priority` - (Optional) Lower priorities execute first. Default: `100`.
- `request_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `REQUEST`.
- `cache_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `CACHE`.
- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.

The `healthcheck` block supports:

- `name` - (Required) A unique name to identify this Healthcheck.
- `host` - (Required) The Host header to send for this Healthcheck.
- `path` - (Required) The path to check.
- `check_interval` - (Optional) How often to run the Healthcheck in milliseconds. Default `5000`.
- `expected_response` - (Optional) The status code expected from the host. Default `200`.
- `http_version` - (Optional) Whether to use version 1.0 or 1.1 HTTP. Default `1.1`.
- `initial` - (Optional) When loading a config, the initial number of probes to be seen as OK. Default `2`.
- `method` - (Optional) Which HTTP method to use. Default `HEAD`.
- `threshold` - (Optional) How many Healthchecks must succeed to be considered healthy. Default `3`.
- `timeout` - (Optional) Timeout in milliseconds. Default `500`.
- `window` - (Optional) The number of most recent Healthcheck queries to keep for this Healthcheck. Default `5`.

The `request_setting` block allow you to customize Fastly's request handling, by defining behavior that should change based on a predefined `condition`:

- `name` - (Required) Unique name to refer to this Request Setting.
- `request_condition` - (Optional) Name of already defined `condition` to determine if this request setting should be applied.
- `max_stale_age` - (Optional) How old an object is allowed to be to serve `stale-if-error` or `stale-while-revalidate`, in seconds.
- `force_miss` - (Optional) Force a cache miss for the request. If specified, can be `true` or `false`.
- `force_ssl` - (Optional) Forces the request to use SSL (Redirects a non-SSL request to SSL).
- `action` - (Optional) Allows you to terminate request handling and immediately perform an action. When set it can be `lookup` or `pass` (Ignore the cache completely).
- `bypass_busy_wait` - (Optional) Disable collapsed forwarding, so you don't wait for other objects to origin.
- `hash_keys` - (Optional) Comma separated list of varnish request object fields that should be in the hash key.
- `xff` - (Optional) X-Forwarded-For, should be `clear`, `leave`, `append`, `append_all`, or `overwrite`. Default `append`.
- `timer_support` - (Optional) Injects the X-Timer info into the request for viewing origin fetch durations.
- `geo_headers` - (Optional) Injects Fastly-Geo-Country, Fastly-Geo-City, and Fastly-Geo-Region into the request headers.
- `default_host` - (Optional) Sets the host header.

The `s3logging` block supports:

- `name` - (Required) A unique name to identify this S3 Logging Bucket.

- `bucket_name` - (Required) The name of the bucket in which to store the logs.
- `s3_access_key` - (Required) AWS Access Key of an account with the required permissions to post logs. It is **strongly** recommended you create a separate IAM user with permissions to only operate on this Bucket. This key will be not be encrypted. You can provide this key via an environment variable, `FASTLY_S3_ACCESS_KEY`.
- `s3_secret_key` - (Required) AWS Secret Key of an account with the required permissions to post logs. It is **strongly** recommended you create a separate IAM user with permissions to only operate on this Bucket. This secret will be not be encrypted. You can provide this secret via an environment variable, `FASTLY_S3_SECRET_KEY`.
- `path` - (Optional) Path to store the files. Must end with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path.
- `domain` - (Optional) If you created the S3 bucket outside of `us-east-1`, then specify the corresponding bucket endpoint. Example: `s3-us-west-2.amazonaws.com`.
- `period` - (Optional) How frequently the logs should be transferred, in seconds. Default `3600`.
- `gzip_level` - (Optional) Level of GZIP compression, from `0-9`. `0` is no compression. `1` is fastest and least compressed, `9` is slowest and most compressed. Default `0`.
- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (`%h %l %u %t %r %>s`)
- `format_version` - (Optional) The version of the custom logging format used for the configured endpoint. Can be either 1 (the default, version 1 log format) or 2 (the version 2 log format).
- `message_type` - (Optional) How the message should be formatted; one of: `classic`, `loggly`, `logplex` or `blank`. Default `classic`.
- `timestamp_format` - (Optional) `strftime` specified timestamp formatting (default `%Y-%m-%dT%H:%M:%S.000`).
- `redundancy` - (Optional) The S3 redundancy level. Should be formatted; one of: `standard`, `reduced_redundancy` or null. Default `null`.
- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed; one of: `none` or `waf_debug`.

The `papertrail` block supports:

- `name` - (Required) A unique name to identify this Papertrail endpoint.
- `address` - (Required) The address of the Papertrail endpoint.
- `port` - (Required) The port associated with the address where the Papertrail endpoint can be accessed.
- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (`%h %l %u %t %r`

%>s)
- response_condition - (Optional) Name of already defined condition to apply. This condition must be of type RESPONSE. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- placement - (Optional) Where in the generated VCL the logging call should be placed; one of: none or waf_debug.

The sumologic block supports:

- name - (Required) A unique name to identify this Sumologic endpoint.
- url - (Required) The URL to Sumologic collector endpoint
- format - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (%h %l %u %t %r %>s)
- format_version - (Optional) The version of the custom logging format used for the configured endpoint. Can be either 1 (the default, version 1 log format) or 2 (the version 2 log format).
- response_condition - (Optional) Name of already defined condition to apply. This condition must be of type RESPONSE. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- message_type - (Optional) How the message should be formatted; one of: classic, loggly, logplex or blank. Default classic. See Fastly's Documentation on Sumologic
- placement - (Optional) Where in the generated VCL the logging call should be placed; one of: none or waf_debug.

The gcslogging block supports:

- name - (Required) A unique name to identify this GCS endpoint.
- email - (Required) The email address associated with the target GCS bucket on your account. You may optionally provide this secret via an environment variable, FASTLY_GCS_EMAIL.
- bucket_name - (Required) The name of the bucket in which to store the logs.
- secret_key - (Required) The secret key associated with the target gcs bucket on your account. You may optionally provide this secret via an environment variable, FASTLY_GCS_SECRET_KEY. A typical format for the key is PEM format, containing actual newline characters where required.
- path - (Optional) Path to store the files. Must end with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path.
- period - (Optional) How frequently the logs should be transferred, in seconds. Default 3600.
- gzip_level - (Optional) Level of GZIP compression, from 0-9. 0 is no compression. 1 is fastest and least compressed, 9 is slowest and most compressed. Default 0.
- format - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (%h %l %u %t %r %>s)

- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- `message_type` - (Optional) How the message should be formatted; one of: `classic`, `loggly`, `logplex` or `blank`. Default `classic`. Fastly Documentation
- `placement` - (Optional) Where in the generated VCL the logging call should be placed; one of: `none` or `waf_debug`.

The `bigquerylogging` block supports:

- `name` - (Required) A unique name to identify this BigQuery logging endpoint.
- `project_id` - (Required) The ID of your GCP project.
- `dataset` - (Required) The ID of your BigQuery dataset.
- `table` - (Required) The ID of your BigQuery table.
- `email` - (Optional) The email for the service account with write access to your BigQuery dataset. If not provided, this will be pulled from a `FASTLY_BQ_EMAIL` environment variable.
- `secret_key` - (Optional) The secret key associated with the sservice account that has write access to your BigQuery table. If not provided, this will be pulled from the `FASTLY_BQ_SECRET_KEY` environment variable. Typical format for this is a private key in a string with newlines.
- `format` - (Optional) Apache style log formatting. Must produce JSON that matches the schema of your BigQuery table.
- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- `template` - (Optional) Big query table name suffix template. If set will be interpreted as a strftime compatible string and used as the Template Suffix for your table.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed; one of: `none` or `waf_debug`.

The `syslog` block supports:

- `name` - (Required) A unique name to identify this Syslog endpoint.
- `address` - (Required) A hostname or IPv4 address of the Syslog endpoint.
- `port` - (Optional) The port associated with the address where the Syslog endpoint can be accessed. Default `514`.
- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (%h %l %u %t %r %>s)
- `format_version` - (Optional) The version of the custom logging format used for the configured endpoint. Can be either 1 (the default, version 1 log format) or 2 (the version 2 log format).
- `token` - (Optional) Whether to prepend each message with a specific token.

- `use_tls` - (Optional) Whether to use TLS for secure logging. Default `false`.
- `tls_hostname` - (Optional) Used during the TLS handshake to validate the certificate.
- `tls_ca_cert` - (Optional) A secure certificate to authenticate the server with. Must be in PEM format. You can provide this certificate via an environment variable, `FASTLY_SYSLOG_CA_CERT`
- `tls_client_cert` - (Optional) The client certificate used to make authenticated requests. Must be in PEM format. You can provide this certificate via an environment variable, `FASTLY_SYSLOG_CLIENT_CERT`
- `tls_client_key` - (Optional) The client private key used to make authenticated requests. Must be in PEM format. You can provide this key via an environment variable, `FASTLY_SYSLOG_CLIENT_KEY`
- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- `message_type` - (Optional) How the message should be formatted; one of: `classic`, `loggly`, `logplex` or `blank`. Default `classic`.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed; one of: `none` or `waf_debug`.

The `logentries` block supports:

- `name` - (Required) A unique name to identify this GCS endpoint.
- `token` - (Required) Logentries Token to be used for authentication (https://logentries.com/doc/input-token/).
- `port` - (Optional) The port number configured in Logentries to send logs to. Defaults to `20000`.
- `use_tls` - (Optional) Whether to use TLS for secure logging. Defaults to `true`
- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Defaults to Apache Common Log format (`%h %l %u %t %r %>s`).
- `format_version` - (Optional) The version of the custom logging format used for the configured endpoint. Can be either 1 (the default, version 1 log format) or 2 (the version 2 log format).
- `response_condition` - (Optional) Name of already defined `condition` to apply. This `condition` must be of type `RESPONSE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed; one of: `none` or `waf_debug`.

The `splunk` block supports:

- `name` - (Required) A unique name to identify the Splunk endpoint.
- `url` - (Required) The Splunk URL to stream logs to.
- `token` - (Required) The Splunk token to be used for authentication.

- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Default `%h %l %u %t \"%r\" %>s %b`.
- `format_version` - (Optional) The version of the custom logging format used for the configured endpoint. Can be either `1` or `2`. The logging call gets placed by default in `vcl_log` if `format_version` is set to `2` and in `vcl_deliver` if `format_version` is set to `1`. Default `2`.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed, overriding any `format_version` default. Can be either `none` or `waf_debug`.
- `response_condition` - (Optional) The name of the `condition` to apply. If empty, always execute.

The `blobstoragelogging` block supports:

- `name` - (Required) A unique name to identify the Azure Blob Storage endpoint.
- `account_name` - (Required) The unique Azure Blob Storage namespace in which your data objects are stored.
- `container` - (Required) The name of the Azure Blob Storage container in which to store logs.
- `sas_token` - (Required) The Azure shared access signature providing write access to the blob service objects. Be sure to update your token before it expires or the logging functionality will not work.
- `path` - (Optional) The path to upload logs to. Must end with a trailing slash. If this field is left empty, the files will be saved in the container's root path.
- `period` - (Optional) How frequently the logs should be transferred in seconds. Default `3600`.
- `timestamp_format` - (Optional) `strftime` specified timestamp formatting. Default `%Y-%m-%dT%H:%M:%S.000`.
- `gzip_level` - (Optional) Level of GZIP compression from `0`to `9`. `0` means no compression. `1` is the fastest and the least compressed version, `9` is the slowest and the most compressed version. Default `0`.
- `public_key` - (Optional) A PGP public key that Fastly will use to encrypt your log files before writing them to disk.
- `format` - (Optional) Apache-style string or VCL variables to use for log formatting. Default `%h %l %u %t \"%r\" %>s %b`.
- `format_version` - (Optional) The version of the custom logging format used for the configured endpoint. Can be either `1` or `2`. The logging call gets placed by default in `vcl_log` if `format_version` is set to `2` and in `vcl_deliver` if `format_version` is set to `1`. Default `2`.
- `message_type` - (Optional) How the message should be formatted. Can be either `classic`, `loggly`, `logplex` or `blank`. Default `classic`.
- `placement` - (Optional) Where in the generated VCL the logging call should be placed, overriding any `format_version` default. Can be either `none` or `waf_debug`.
- `response_condition` - (Optional) The name of the `condition` to apply.

If empty, always execute.

The `response_object` block supports:

- `name` - (Required) A unique name to identify this Response Object.
- `status` - (Optional) The HTTP Status Code. Default `200`.
- `response` - (Optional) The HTTP Response. Default `Ok`.
- `content` - (Optional) The content to deliver for the response object.
- `content_type` - (Optional) The MIME type of the content.
- `request_condition` - (Optional) Name of already defined `condition` to be checked during the request phase. If the condition passes then this object will be delivered. This `condition` must be of type `REQUEST`.
- `cache_condition` - (Optional) Name of already defined `condition` to check after we have retrieved an object. If the condition passes then deliver this Request Object instead. This `condition` must be of type `CACHE`. For detailed information about Conditionals, see Fastly's Documentation on Conditionals.

The `snippet` block supports:

- `name` - (Required) A name that is unique across "regular" and "dynamic" VCL Snippet configuration blocks.
- `type` - (Required) The location in generated VCL where the snippet should be placed (can be one of `init`, `recv`, `hit`, `miss`, `pass`, `fetch`, `error`, `deliver`, `log` or `none`).
- `content` (Required) The VCL code that specifies exactly what the snippet does.
- `priority` - (Optional) Priority determines the ordering for multiple snippets. Lower numbers execute first. Defaults to `100`.

The `dynamicsnippet` block supports:

- `name` - (Required) A name that is unique across "regular" and "dynamic" VCL Snippet configuration blocks.
- `type` - (Required) The location in generated VCL where the snippet should be placed (can be one of `init`, `recv`, `hit`, `miss`, `pass`, `fetch`, `error`, `deliver`, `log` or `none`).
- `priority` - (Optional) Priority determines the ordering for multiple snippets. Lower numbers execute first. Defaults to `100`.

The `vcl` block supports:

- `name` - (Required) A unique name for this configuration block.
- `content` - (Required) The custom VCL code to upload.
- `main` - (Optional) If `true`, use this block as the main configuration. If `false`, use this block as an includable library. Only a single VCL block can be marked as the main block. Default is `false`.

The `acl` block supports:

- `name` - (Required) A unique name to identify this ACL.

The `dictionary` block supports:

- `name` - (Required) A unique name to identify this dictionary.
- `write_only` - (Optional) If `true`, the dictionary is a private dictionary, and items are not readable in the UI or via API. Default is `false`. It is important to note that changing this attribute will delete and recreate the dictionary, discard the current items in the dictionary. Using a write-only/private dictionary should only be done if the items are managed outside of Terraform.

» Attributes Reference

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

- `id` – The ID of the Service.
- `active_version` – The currently active version of your Fastly Service.
- `cloned_version` - The latest cloned version by the provider. The value gets only set after running `terraform apply`.

The `dynamicsnippet` block exports:

- `snippet_id` - The ID of the dynamic snippet.

The `acl` block exports:

- `acl_id` - The ID of the ACL.

The `dictionary` block exports:

- `dictionary_id` - The ID of the dictionary.

» Import

Fastly Service can be imported using their service ID, e.g.

```
$ terraform import fastly_service_v1.demo xxxxxxxxxxxxxxxxxxx
```

» fastly_service_dictionary_items_v1

Defines a map of Fastly dictionary items that can be used to populate a service dictionary. This resource will populate a dictionary with the items and will track their state.

Warning: Terraform will take precedence over any changes you make in the UI or API. Such changes are likely to be reversed if you run Terraform again.

If Terraform is being used to populate the initial content of a dictionary which you intend to manage via API or UI, then the lifecycle `ignore_changes` field

can be used with the resource. An example of this configuration is provided below.

» Example Usage

Basic usage:

```
variable "mydict_name" {
    type = string
    default = "My Dictionary"
}

resource "fastly_service_v1" "myservice" {
  name = "demofastly"

  domain {
      name    = "demo.notexample.com"
      comment = "demo"
  }

  backend {
      address = "demo.notexample.com.s3-website-us-west-2.amazonaws.com"
      name    = "AWS S3 hosting"
      port    = 80
    }

  dictionary {
    name       = var.mydict_name
  }

  force_destroy = true
}

resource "fastly_service_dictionary_items_v1" "items" {
    service_id = "${fastly_service_v1.myservice.id}"
    dictionary_id = "${{for s in fastly_service_v1.myservice.dictionary : s.name => s.dictio
    items = {
        key1: "value1"
        key2: "value2"
    }
}
```

Complex object usage:

```
variable "mydict" {
  type = object({ name=string, items=map(string) })
```

```
  default = {
    name = "My Dictionary"
    items = {
      key1: "value1x"
      key2: "value2x"
    }
  }
}

resource "fastly_service_v1" "myservice" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
    comment = "demo"
  }

  backend {
    address = "demo.notexample.com.s3-website-us-west-2.amazonaws.com"
    name    = "AWS S3 hosting"
    port    = 80
  }

  dictionary {
    name = var.mydict.name
  }

  force_destroy = true
  }

resource "fastly_service_dictionary_items_v1" "items" {
  service_id = fastly_service_v1.myservice.id
  dictionary_id = {for d in fastly_service_v1.myservice.dictionary : d.name => d.dictionary_
  items = var.mydict.items
}
```

Expression and functions usage:

```
// Local variables used when formating values for the "My Project Dictionary" example
locals {
  dictionary_name = "My Project Dictionary"
  host_base = "demo.ocnotexample.com"
  host_divisions = ["alpha", "beta", "gamma", "delta"]
}

// Define the standard service that will be used to manage the dictionaries.
resource "fastly_service_v1" "myservice" {
```

```
  name = "demofastly"

  domain {
    name = "demo.ocnotexample.com"
    comment = "demo"
  }

  backend {
    address = "demo.ocnotexample.com.s3-website-us-west-2.amazonaws.com"
    name = "AWS S3 hosting"
    port = 80
  }

  dictionary {
    name = local.dictionary_name
  }

  force_destroy = true
}

// This resource is dynamically creating the items from the local variables through for expr
resource "fastly_service_dictionary_items_v1" "project" {
  service_id = fastly_service_v1.myservice.id
  dictionary_id = {for d in fastly_service_v1.myservice.dictionary : d.name => d.dictionary_
  items = {
    for division in local.host_divisions:
      division => format("%s.%s", division, local.host_base)
  }
}
```

» Supporting API and UI dictionary updates with ignore_changes

The following example demonstrates how the lifecycle `ignore_changes` field can be used to suppress updates against the items in a dictionary. If, after your first deploy, the Fastly API or UI is to be used to manage items in a dictionary, then this will stop Terraform realigning the remote state with the initial set of dictionary items defined in your HCL.

```
...

resource "fastly_service_dictionary_items_v1" "items" {
    service_id = "${fastly_service_v1.myservice.id}"
    dictionary_id = "${{for s in fastly_service_v1.myservice.dictionary : s.name => s.dictio
    items = {
        key1: "value1"
```

```
        key2: "value2"
    }

    lifecycle {
      ignore_changes = [items,]
    }
}
```

» Argument Reference

The following arguments are supported:

- `service_id` - (Required) The ID of the service that the dictionary belongs to
- `dictionary_id` - (Required) The ID of the dictionary that the items belong to
- `items` - (Optional) A map representing an entry in the dictionary, (key/value)

» Attributes Reference

- fastly-dictionary
- fastly-dictionary_item

» Import

This is an example of the import command being applied to the resource named `fastly_service_dictionary_items_v1.items` The resource ID is a combined value of the `service_id` and `dictionary_id` separated by a forward slash.

```
$ terraform import fastly_service_dictionary_items_v1.items xxxxxxxxxxxxxxxxxxxx/xxxxxxxxxx
```

If Terraform is already managing remote dictionary items against a resource being imported then the user will be asked to remove it from the existing Terraform state.
The following is an example of the Terraform state command to remove the resource named `fastly_service_dictionary_items_v1.items` from the Terraform state file.

```
$ terraform state rm fastly_service_dictionary_items_v1.items
```

» fastly_service_acl_entries_v1

Defines a set of Fastly ACL entries that can be used to populate a service ACL. This resource will populate an ACL with the entries and will track their state.

Warning: Terraform will take precedence over any changes you make in the UI or API. Such changes are likely to be reversed if you run Terraform again.

If Terraform is being used to populate the initial content of an ACL which you intend to manage via API or UI, then the lifecycle `ignore_changes` field can be used with the resource. An example of this configuration is provided below.

» Example Usage

Basic usage:

```
variable "myacl_name" {
    type = string
    default = "My ACL"
}

resource "fastly_service_v1" "myservice" {
  name = "demofastly"

  domain {
      name = "demo.notexample.com"
      comment = "demo"
  }

  backend {
      address = "demo.notexample.com.s3-website-us-west-2.amazonaws.com"
      name = "AWS S3 hosting"
      port = 80
    }

  acl {
    name = var.myacl_name
  }

  force_destroy = true
}

resource "fastly_service_acl_entries_v1" "entries" {
  service_id = fastly_service_v1.myservice.id
  acl_id = {for d in fastly_service_v1.myservice.acl : d.name => d.acl_id}[var.myacl_name]
  entry {
```

```
    ip = "127.0.0.1"
    subnet = "24"
    negated = false
    comment = "ALC Entry 1"
  }
}
```

Complex object usage:

The following example demonstrates the use of dynamic nested blocks to create ACL entries.

```
locals {
  acl_name = "my_acl"
  acl_entries = [
    {
      ip      = "1.2.3.4"
      comment = "acl_entry_1"
    },
    {
      ip      = "1.2.3.5"
      comment = "acl_entry_2"
    },
    {
      ip      = "1.2.3.6"
      comment = "acl_entry_3"
    }
  ]
}

resource "fastly_service_v1" "myservice" {
  name = "demofastly"

  domain {
    name    = "demo.notexample.com"
    comment = "demo"
  }

  backend {
    address = "1.2.3.4"
    name    = "localhost"
    port    = 80
  }

  acl {
    name = local.acl_name
  }
```

```
  force_destroy = true
}

resource "fastly_service_acl_entries_v1" "entries" {
  service_id = fastly_service_v1.myservice.id
  acl_id     = { for d in fastly_service_v1.myservice.acl : d.name => d.acl_id }[local.acl_n
  dynamic "entry" {
    for_each = [for e in local.acl_entries : {
      ip      = e.ip
      comment = e.comment
    }]

    content {
      ip      = entry.value.ip
      subnet  = 22
      comment = entry.value.comment
      negated = false
    }
  }
}
```

» Supporting API and UI ACL updates with ignore_changes

The following example demonstrates how the lifecycle `ignore_changes` field can be used to suppress updates against the entries in an ACL. If, after your first deploy, the Fastly API or UI is to be used to manage entries in an ACL, then this will stop Terraform realigning the remote state with the initial set of ACL entries defined in your HCL.

```
...

resource "fastly_service_acl_entries_v1" "entries" {
  service_id = fastly_service_v1.myservice.id
  acl_id = {for d in fastly_service_v1.myservice.acl : d.name => d.acl_id}[var.myacl_name]
  entry {
    ip = "127.0.0.1"
    subnet = "24"
    negated = false
    comment = "ALC Entry 1"
  }

  lifecycle {
    ignore_changes = [entry,]
  }
```

```
}
```

» Argument Reference

The following arguments are supported:

- `service_id` - (Required) The ID of the Service that the ACL belongs to
- `acl_id` - (Required) The ID of the ACL that the items belong to
- `entry` - (Optional) A Set ACL entries that are applied to the service. Defined below

The `entry` block supports:

- `ip` - (Required, string) An IP address that is the focus for the ACL
- `subnet` - (Optional, string) An optional subnet mask applied to the IP address
- `negated` - (Optional, boolean) A boolean that will negate the match if true
- `comment` - (Optional, string) A personal freeform descriptive note

» Attributes Reference

- fastly-acl
- fastly-acl_entry

» Import

This is an example of the import command being applied to the resource named `fastly_service_acl_entries_v1.entries` The resource ID is a combined value of the `service_id` and `acl_id` separated by a forward slash.

```
$ terraform import fastly_service_acl_entries_v1.entries xxxxxxxxxxxxxxxxxxx/xxxxxxxxxxxxx
```

If Terraform is already managing remote acl entries against a resource being imported then the user will be asked to remove it from the existing Terraform state.
The following is an example of the Terraform state command to remove the resource named `fastly_service_acl_entries_v1.entries` from the Terraform state file.

```
$ terraform state rm fastly_service_acl_entries_v1.entries
```

» fastly_service_dynamic_snippet_content_v1

Defines content that represents blocks of VCL logic that is inserted into your service. This resource will populate the content of a dynamic snippet and allow it to be manged without the creation of a new service verison.

Warning: Terraform will take precedence over any changes you make through the API. Such changes are likely to be reversed if you run Terraform again.

If Terraform is being used to populate the initial content of a dynamic snippet which you intend to manage via the API, then the lifecycle `ignore_changes` field can be used with the resource. An example of this configuration is provided below.

» Example Usage

Basic usage:

```
resource "fastly_service_v1" "myservice" {
 name = "snippet_test"

 domain {
   name    = "snippet.fastlytestdomain.com"
   comment = "snippet test"
 }

 backend {
   address = "tftesting.tftesting.net.s3-website-us-west-2.amazonaws.com"
   name    = "AWS S3 hosting"
   port    = 80
 }

 dynamicsnippet {
   name     = "My Dynamic Snippet"
   type     = "recv"
   priority = 110
 }

 default_host = "tftesting.tftesting.net.s3-website-us-west-2.amazonaws.com"

 force_destroy = true
}

resource "fastly_service_dynamic_snippet_content_v1" "my_dyn_content" {

 service_id = fastly_service_v1.myservice.id
```

```
 snippet_id = { for s in fastly_service_v1.myservice.dynamicsnippet : s.name => s.snippet_id

 content = "if ( req.url ) {\n set req.http.my-snippet-test-header = \"true\";\n}"

}
```

Multiple dynamic snippets:

```
resource "fastly_service_v1" "myservice" {
 name = "snippet_test"

 domain {
   name    = "snippet.fastlytestdomain.com"
   comment = "snippet test"
 }

 backend {
   address = "tftesting.tftesting.net.s3-website-us-west-2.amazonaws.com"
   name    = "AWS S3 hosting"
   port    = 80
 }

 dynamicsnippet {
   name     = "My Dynamic Snippet One"
   type     = "recv"
   priority = 110
 }

 dynamicsnippet {
      name     = "My Dynamic Snippet Two"
      type     = "recv"
      priority = 110
    }

 default_host = "tftesting.tftesting.net.s3-website-us-west-2.amazonaws.com"

 force_destroy = true
}

resource "fastly_service_dynamic_snippet_content_v1" "my_dyn_content_one" {

 service_id = fastly_service_v1.myservice.id
 snippet_id = { for s in fastly_service_v1.myservice.dynamicsnippet : s.name => s.snippet_id

 content = "if ( req.url ) {\n set req.http.my-snippet-test-header-one = \"true\";\n}"

}
```

```
resource "fastly_service_dynamic_snippet_content_v1" "my_dyn_content_two" {

 service_id = fastly_service_v1.myservice.id
 snippet_id = { for s in fastly_service_v1.myservice.dynamicsnippet : s.name => s.snippet_id

 content = "if ( req.url ) {\n set req.http.my-snippet-test-header-two = \"true\";\n}"

}
```

» Supporting API dynamic snippet updates with ignore_changes

The following example demonstrates how the lifecycle `ignore_changes` field can be used to suppress updates against the content in a dynamic snippet. If, after your first deploy, the Fastly API is to be used to manage items in a dynamic snippet, then this will stop Terraform realigning the remote state with the initial content defined in your HCL.

```
...

resource "fastly_service_dynamic_snippet_content_v1" "my_dyn_content" {

  service_id = fastly_service_v1.myservice.id
  snippet_id = { for s in fastly_service_v1.myservice.dynamicsnippet : s.name => s.snippet_i

  content = "if ( req.url ) {\n set req.http.my-snippet-test-header = \"true\";\n}"

  lifecycle {
    ignore_changes = [content, ]
  }
}
```

» Argument Reference

The following arguments are supported:

- `service_id` - (Required) The ID of the service that the dynamic snippet belongs to
- `snippet_id` - (Required) The ID of the dynamic snippet that the content belong to
- `content` - (Required) The VCL code that specifies exactly what the snippet does.

» Attributes Reference

- fastly-vcl
- fastly-vcl-snippets

» Import

This is an example of the import command being applied to the resource named `fastly_service_dynamic_snippet_content_v1.content` The resource ID is a combined value of the `service_id` and `snippet_id` separated by a forward slash.

```
$ terraform import fastly_service_dynamic_snippet_content_v1.content xxxxxxxxxxxxxxxxxxx/xx
```

If Terraform is already managing remote content against a resource being imported then the user will be asked to remove it from the existing Terraform state. The following is an example of the Terraform state command to remove the resource named `fastly_service_dynamic_snippet_content_v1.content` from the Terraform state file.

```
$ terraform state rm fastly_service_dynamic_snippet_content_v1.content
```