» vault_approle_auth_backend_role

Reads the Role ID of an AppRole from a Vault server.

» Example Usage

```
data "vault_approle_auth_backend_role_id" "role" {
  backend   = "my-approle-backend"
  role_name = "my-role"
}

output "role-id" {
  value = "${data.vault_approle_auth_backend_role_id.role.role_id}"
}
```

» Argument Reference

The following arguments are supported:

- `role_name` - (Required) The name of the role to retrieve the Role ID for.
- `backend` - (Optional) The unique name for the AppRole backend the role to retrieve a RoleID for resides in. Defaults to "approle".

» Attributes Reference

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

- `role_id` - The RoleID of the role.

» vault_aws_access_credentials

Reads AWS credentials from an AWS secret backend in Vault.

Important All data retrieved from Vault will be written in cleartext to state file generated by Terraform, will appear in the console output when Terraform runs, and may be included in plan files if secrets are interpolated into any resource attributes. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_aws_secret_backend" "aws" {
  access_key = "AKIA....."
  secret_key = "SECRETKEYFROMAWS"
}

resource "vault_aws_secret_backend_role" "role" {
  backend = "${vault_aws_secret_backend.aws.path}"
  name    = "test"

  policy = <<EOT
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
EOT
}

# generally, these blocks would be in a different module
data "vault_aws_access_credentials" "creds" {
  backend = "${vault_aws_secret_backend.aws.path}"
  role    = "${vault_aws_secret_backend_role.role.name}"
}

provider "aws" {
  access_key = "${data.vault_aws_access_credentials.creds.access_key}"
  secret_key = "${data.vault_aws_access_credentials.creds.secret_key}"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Required) The path to the AWS secret backend to read credentials from, with no leading or trailing `/`s.
- `role` - (Required) The name of the AWS secret backend role to read credentials from, with no leading or trailing `/`s.

- `type` - (Optional) The type of credentials to read. Defaults to `"creds"`, which just returns an AWS Access Key ID and Secret Key. Can also be set to `"sts"`, which will return a security token in addition to the keys.

» Attributes Reference

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

- `access_key` - The AWS Access Key ID returned by Vault.
- `secret_key` - The AWS Secret Key returned by Vault.
- `security_token` - The STS token returned by Vault, if any.
- `lease_id` - The lease identifier assigned by Vault.
- `lease_duration` - The duration of the secret lease, in seconds relative to the time the data was requested. Once this time has passed any plan generated with this data may fail to apply.
- `lease_start_time` - As a convenience, this records the current time on the computer where Terraform is running when the data is requested. This can be used to approximate the absolute time represented by `lease_duration`, though users must allow for any clock drift and response latency relative to the Vault server.
- `lease_renewable` - `true` if the lease can be renewed using Vault's `sys/renew/{lease-id}` endpoint. Terraform does not currently support lease renewal, and so it will request a new lease each time this data source is refreshed.

» vault_generic_secret

Reads arbitrary data from a given path in Vault.

This resource is primarily intended to be used with Vault's "generic" secret backend, but it is also compatible with any other Vault endpoint that supports the `vault read` command.

Important All data retrieved from Vault will be written in cleartext to state file generated by Terraform, will appear in the console output when Terraform runs, and may be included in plan files if secrets are interpolated into any resource attributes. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
data "vault_generic_secret" "rundeck_auth" {
  path = "secret/rundeck_auth"
}

# Rundeck Provider, for example
provider "rundeck" {
  url        = "http://rundeck.example.com/"
  auth_token = "${data.vault_generic_secret.rundeck_auth.data["auth_token"]}"
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The full logical path from which to request data. To read data from the "generic" secret backend mounted in Vault by default, this should be prefixed with `secret/`. Reading from other backends with this data source is possible; consult each backend's documentation to see which endpoints support the `GET` method.

» Required Vault Capabilities

Use of this resource requires the `read` capability on the given path.

» Attributes Reference

The following attributes are exported:

- `data_json` - A string containing the full data payload retrieved from Vault, serialized in JSON format.
- `data` - A mapping whose keys are the top-level data keys returned from Vault and whose values are the corresponding values. This map can only represent string data, so any non-string values returned from Vault are serialized as JSON.
- `lease_id` - The lease identifier assigned by Vault, if any.
- `lease_duration` - The duration of the secret lease, in seconds relative to the time the data was requested. Once this time has passed any plan generated with this data may fail to apply.
- `lease_start_time` - As a convenience, this records the current time on the computer where Terraform is running when the data is requested. This can

be used to approximate the absolute time represented by `lease_duration`, though users must allow for any clock drift and response latency relative to to the Vault server.

- `lease_renewable` - `true` if the lease can be renewed using Vault's `sys/renew/{lease-id}` endpoint. Terraform does not currently support lease renewal, and so it will request a new lease each time this data source is refreshed.

» vault_approle_auth_backend_role

Manages an AppRole auth backend role in a Vault server. See the Vault documentation for more information.

» Example Usage

```
resource "vault_auth_backend" "approle" {
  type = "approle"
}

resource "vault_approle_auth_backend_role" "example" {
  backend   = "${vault_auth_backend.approle.path}"
  role_name = "test-role"
  policies  = ["default", "dev", "prod"]
}
```

» Argument Reference

The following arguments are supported:

- `role_name` - (Required) The name of the role.
- `role_id` - (Optional) The RoleID of this role. If not specified, one will be auto-generated.
- `bind_secret_id` - (Optional) Whether or not to require `secret_id` to be presented when logging in using this AppRole. Defaults to `true`.
- `bound_cidr_list` - (Optional) If set, specifies blocks of IP addresses which can perform the login operation.
- `policies` - (Optional) An array of strings specifying the policies to be set on tokens issued using this role.

- `secret_id_num_uses` - (Optional) The number of times any particular SecretID can be used to fetch a token from this AppRole, after which the SecretID will expire. A value of zero will allow unlimited uses.
- `secret_id_ttl` - (Optional) The number of seconds after which any SecretID expires.
- `token_num_uses` - (Optional) The number of times issued tokens can be used. A value of 0 means unlimited uses.
- `token_ttl` - (Optional) The TTL period of tokens issued using this role, provided as a number of seconds.
- `token_max_ttl` - (Optional) The maximum allowed lifetime of tokens issued using this role, provided as a number of seconds.
- `period` - (Optional) If set, indicates that the token generated using this role should never expire. The token should be renewed within the duration specified by this value. At each renewal, the token's TTL will be set to the value of this field. The maximum allowed lifetime of token issued using this role. Specified as a number of seconds.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_approle_auth_backend_role_login

Logs into Vault using the AppRole auth backend. See the Vault documentation for more information.

» Example Usage

```
resource "vault_auth_backend" "approle" {
  type = "approle"
}

resource "vault_approle_auth_backend_role" "example" {
  backend   = "${vault_auth_backend.approle.path}"
  role_name = "test-role"
  policies  = ["default", "dev", "prod"]
}

resource "vault_approle_auth_backend_role_secret_id" "id" {
  backend   = "${vault_auth_backend.approle.path}"
```

```
  role_name = "${vault_approle_auth_backend_role.example.role_name}"
}

resource "vault_approle_auth_backend_role_login" "login" {
  backend   = "${vault_auth_backend.approle.path}"
  role_id   = "${vault_approle_auth_backend_role.example.role_id}"
  secret_id = "${vault_approle_auth_backend_role_secret_id.id.secret_id}"
}
```

» Argument Reference

The following arguments are supported:

- `role_id` - (Required) The ID of the role to log in with.
- `secret_id` - (Optional) The secret ID of the role to log in with. Required unless `bind_secret_id` is set to false on the role.
- `backend` - The unique path of the Vault backend to log in with.

» Attributes Reference

In addition to the fields above, the following attributes are exported:

- `policies` - A list of policies applied to the token.
- `renewable` - Whether the token is renewable or not.
- `lease_duration` - How long the token is valid for, in seconds.
- `lease_started` - The date and time the lease started, in RFC 3339 format.
- `accessor` - The accessor for the token.
- `client_token` - The Vault token created.
- `metadata` - The metadata associated with the token.

» vault_approle_auth_backend_role_secret_id

Manages an AppRole auth backend SecretID in a Vault server. See the Vault documentation for more information.

» Example Usage

```
resource "vault_auth_backend" "approle" {
  type = "approle"
}

resource "vault_approle_auth_backend_role" "example" {
  backend   = "${vault_auth_backend.approle.path}"
  role_name = "test-role"
  policies  = ["default", "dev", "prod"]
}

resource "vault_approle_auth_backend_role_secret_id" "id" {
  backend   = "${vault_auth_backend.approle.path}"
  role_name = "${vault_approle_auth_backend_role.example.role_name}"

  metadata = <<EOT
{
  "hello": "world"
}
EOT
}
```

» Argument Reference

The following arguments are supported:

- `role_name` - (Required) The name of the role to create the SecretID for.
- `metadata` - (Optional) A JSON-encoded string containing metadata in key-value pairs to be set on tokens issued with this SecretID.
- `cidr_list` - (Optional) If set, specifies blocks of IP addresses which can perform the login operation using this SecretID.
- `secret_id` - (Optional) The SecretID to be created. If set, uses "Push" mode. Defaults to Vault auto-generating SecretIDs.

» Attributes Reference

In addition to the fields above, the following attributes are exported:

- `accessor` - The unique ID for this SecretID that can be safely logged.

» vault_auth_backend

» Example Usage

```
resource "vault_auth_backend" "example" {
  type = "github"
}
```

» Argument Reference

The following arguments are supported:

- `type` - (Required) The name of the policy
- `path` - (Optional) The path to mount the auth backend. This defaults to the name.
- `description` - (Optional) A description of the auth backend

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_auth_backend_cert

Manages a certificate to be used with an AWS Auth Backend in Vault.

This resource sets the AWS public key and the type of document that can be verified against the key that Vault can then use to verify the instance identity documents making auth requests.

For more information, see the Vault docs.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_auth_backend" "aws" {
  type = "aws"
}
```

```
resource "vault_aws_auth_backend_cert" "cert" {
  backend         = "${vault_auth_backend.aws.path}"
  cert_name       = "my-cert"
  aws_public_cert = "${file("${path.module}/aws_public_key.crt})"
  type            = "pkcs7"
}
```

» Argument Reference

The following arguments are supported:

- `cert_name` - (Required) The name of the certificate.
- `aws_public_cert` - (Required) The Base64 encoded AWS Public key required to verify PKCS7 signature of the EC2 instance metadata. You can find this key in the AWS documentation.
- `type` - (Optional) Either "pkcs7" or "identity", indicating the type of document which can be verified using the given certificate. Defaults to "pkcs7".
- `backend` - (Optional) The path the AWS auth backend being configured was mounted at. Defaults to `aws`.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_auth_backend_client

Configures the client used by an AWS Auth Backend in Vault.

This resource sets the access key and secret key that Vault will use when making API requests on behalf of an AWS Auth Backend. It can also be used to override the URLs Vault uses when making those API requests.

For more information, see the Vault docs.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_auth_backend" "example" {
  type = "aws"
}

resource "vault_aws_auth_backend_client" "example" {
  backend = "${vault_auth_backend.example.path}"
  access_key = "INSERT_AWS_ACCESS_KEY"
  secret_key = "INSERT_AWS_SECRET_KEY"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Optional) The path the AWS auth backend being configured was mounted at. Defaults to `aws`.
- `access_key` - (Optional) The AWS access key that Vault should use for the auth backend.
- `secret_key` - (Optional) The AWS secret key that Vault should use for the auth backend.
- `ec2_endpoint` - (Optional) Override the URL Vault uses when making EC2 API calls.
- `iam_endpoint` - (Optional) Override the URL Vault uses when making IAM API calls.
- `sts_endpoint` - (Optional) Override the URL Vault uses when making STS API calls.
- `iam_server_id_header_value` - (Optional) The value to require in the `X-Vault-AWS-IAM-Server-ID` header as part of `GetCallerIdentity` requests that are used in the IAM auth method.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_auth_backend_identity_whitelist

Configures the periodic tidying operation of the whitelisted identity entries.

For more information, see the Vault docs.

» Example Usage

```
resource "vault_auth_backend" "example" {
  type = "aws"
}

resource "vault_aws_auth_backend_identity_whitelist" "example" {
  backend       = "${vault_auth_backend.example.path}"
  safety_buffer = 3600
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Optional) The path of the AWS backend being configured.
- `safety_buffer` - (Optional) The amount of extra time, in minutes, that must have passed beyond the roletag expiration, before it is removed from the backend storage.
- `disable_periodic_tidy` - (Optional) If set to true, disables the periodic tidying of the identity-whitelist entries.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_auth_backend_login

Logs into a Vault server using an AWS auth backend. Login can be accomplished using a signed identity request from IAM or using ec2 instance metadata. For more information, see the Vault documentation.

» Example Usage

```
resource "vault_auth_backend" "aws" {
  type = "aws"
}
```

```
resource "vault_aws_auth_backend_client" "example" {
  backend    = "${vault_auth_backend.aws.path}"
  access_key = "123456789012"
  secret_key = "AWSSECRETKEYGOESHERE"
}

resource "vault_aws_auth_backend_role" "example" {
  backend                         = "${vault_auth_backend.aws.path}"
  role                            = "test-role"
  auth_type                       = "ec2"
  bound_ami_id                    = "ami-8c1be5f6"
  bound_account_id                = "123456789012"
  bound_vpc_id                    = "vpc-b61106d4"
  bound_subnet_id                 = "vpc-133128f1"
  bound_iam_instance_profile_arn = "arn:aws:iam::123456789012:instance-profile/MyProfile"
  ttl                             = 60
  max_ttl                         = 120
  policies                        = ["default", "dev", "prod"]

  depends_on                      = ["vault_aws_auth_backend_client.example"]
}

resource "vault_aws_auth_backend_login" "example" {
  backend = "${vault_auth_backend.example.path}"
  role = "${vault_aws_auth_backend_role.example.role}"
  identity = "BASE64ENCODEDIDENTITYDOCUMENT"
  signature = "BASE64ENCODEDSHA256IDENTITYDOCUMENTSIGNATURE"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Optional) The unique name of the AWS auth backend. Defaults to 'aws'.
- `role` - (Optional) The name of the AWS auth backend role to create tokens against.
- `identity` - (Optional) The base64-encoded EC2 instance identity document to authenticate with. Can be retrieved from the EC2 metadata server.
- `signature` - (Optional) The base64-encoded SHA256 RSA signature of the instance identity document to authenticate with, with all newline characters removed. Can be retrieved from the EC2 metadata server.

- `pkcs7` - (Optional) The PKCS#7 signature of the identity document to authenticate with, with all newline characters removed. Can be retrieved from the EC2 metadata server.
- `nonce` - (Optional) The unique nonce to be used for login requests. Can be set to a user-specified value, or will contain the server-generated value once a token is issued. EC2 instances can only acquire a single token until the whitelist is tidied again unless they keep track of this nonce.
- `iam_http_request_method` - (Optional) The HTTP method used in the signed IAM request.
- `iam_request_url` - (Optional) The base64-encoded HTTP URL used in the signed request.
- `iam_request_body` - (Optional) The base64-encoded body of the signed request.
- `iam_request_headers` - (Optional) The base64-encoded, JSON serialized representation of the GetCallerIdentity HTTP request headers.

» Attributes Reference

In addition to the fields above, the following attributes are also exposed:

- `lease_duration` - The duration in seconds the token will be valid, relative to the time in `lease_start_time`.
- `lease_start_time` - The approximate time at which the token was created, using the clock of the system where Terraform was running.
- `renewable` - Set to true if the token can be extended through renewal.
- `metadata` - A map of information returned by the Vault server about the authentication used to generate this token.
- `auth_type` - The authentication type used to generate this token.
- `policies` - The Vault policies assigned to this token.
- `accessor` - The token's accessor.
- `client_token` - The token returned by Vault.

» vault_aws_auth_backend_role

Manages an AWS auth backend role in a Vault server. Roles constrain the instances or principals that can perform the login operation against the backend. See the Vault documentation for more information.

» Example Usage

```
resource "vault_auth_backend" "aws" {
  type = "aws"
}

resource "vault_aws_auth_backend_role" "example" {
  backend                         = "${vault_auth_backend.aws.path}"
  role                            = "test-role"
  auth_type                       = "iam"
  bound_ami_id                    = "ami-8c1be5f6"
  bound_account_id                = "123456789012"
  bound_vpc_id                    = "vpc-b61106d4"
  bound_subnet_id                 = "vpc-133128f1"
  bound_iam_role_arn              = "arn:aws:iam::123456789012:role/MyRole"
  bound_iam_instance_profile_arn = "arn:aws:iam::123456789012:instance-profile/MyProfile"
  inferred_entity_type            = "ec2_instance"
  inferred_aws_region             = "us-east-1"
  ttl                             = 60
  max_ttl                         = 120
  policies                        = ["default", "dev", "prod"]
}
```

» Argument Reference

The following arguments are supported:

- `role` - (Required) The name of the role.
- `auth_type` - (Optional) The auth type permitted for this role. Valid choices are `ec2` and `iam`. Defaults to `iam`.
- `bound_ami_id` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they should be using the AMI ID specified by this field. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_account_id` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they should be using the account ID specified by this field. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_region` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that the region in their identity document must match the one specified by this field. `auth_type` must

be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.

- `bound_vpc_id` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they be associated with the VPC ID that matches the value specified by this field. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_subnet_id` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they be associated with the subnet ID that matches the value specified by this field. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_iam_role_arn` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they must match the IAM role ARN specified by this field. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_iam_instance_profile_arn` - (Optional) If set, defines a constraint on the EC2 instances that can perform the login operation that they must be associated with an IAM instance profile ARN which has a prefix that matches the value specified by this field. The value is prefix-matched as though it were a glob ending in `*`. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `role_tag` - (Optional) If set, enable role tags for this role. The value set for this field should be the key of the tag on the EC2 instance. `auth_type` must be set to `ec2` or `inferred_entity_type` must be set to `ec2_instance` to use this constraint.
- `bound_iam_principal_arn` - (Optional) If set, defines the IAM principal that must be authenticated when `auth_type` is set to `iam`. Wildcards are supported at the end of the ARN.
- `inferred_entity_type` - (Optional) If set, instructs Vault to turn on inferencing. The only valid value is `ec2_instance`, which instructs Vault to infer that the role comes from an EC2 instance in an IAM instance profile. This only applies when `auth_type` is set to `iam`.
- `inferred_aws_region` - (Optional) When `inferred_entity_type` is set, this is the region to search for the inferred entities. Required if `inferred_entity_type` is set. This only applies when `auth_type` is set to `iam`.
- `resolve_aws_unique_ids` - (Optional) If set to `true`, the `bound_iam_principal_arn` is resolved to an AWS Unique ID for the bound principal ARN. This field

is ignored when `bound_iam_principal_arn` ends in a wildcard. Resolving to unique IDs more closely mimics the behavior of AWS services in that if an IAM user or role is deleted and a new one is recreated with the same name, those new users or roles won't get access to roles in Vault that were permissioned to the prioer principals of the same name. Defaults to `true`. Once set to `true`, this cannot be changed to `false`--the role must be deleted and recreated, with the value set to `true`.

- `ttl` - (Optional) The TTL period of tokens issued using this role, provided as a number of minutes.
- `max_ttl` - (Optional) The maximum allowed lifetime of tokens issued using this role, provided as a number of minutes.
- `period` - (Optional) If set, indicates that the token generated using this role should never expire. The token should be renewed within the duration specified by this value. At each renewal, the token's TTL will be set to the value of this field. The maximum allowed lifetime of token issued using this role. Specified as a number of minutes.
- `policies` - (Optional) An array of strings specifying the policies to be set on tokens issued using this role.
- `allow_instance_migration` - (Optional) If set to `true`, allows migration of the underlying instance where the client resides.
- `disallow_reauthentication` - (Optional) IF set to `true`, only allows a single token to be granted per instance ID. This can only be set when `auth_type` is set to `ec2`.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_auth_backend_sts_role

Manages an STS role in a Vault server. STS roles are mappings between account IDs and STS ARNs. When a login attempt is made from an EC2 instance in the account ID specified, the associated STS role will be used to verify the request. For more information, see the Vault documentation.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_auth_backend" "aws" {
  type = "aws"
}

resource "vault_aws_auth_backend_sts_role" "role" {
  backend    = "${vault_auth_backend.aws.path}"
  account_id = "1234567890"
  sts_role   = "arn:aws:iam::1234567890:role/my-role"
}
```

» Argument Reference

The following arguments are supported:

- `account_id` - (Optional) The AWS account ID to configure the STS role for.
- `sts_role` - (Optional) The STS role to assume when verifying requests made by EC2 instances in the account specified by `account_id`.
- `backend` - (Optional) The path the AWS auth backend being configured was mounted at. Defaults to `aws`.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_secret_backend

Creates an AWS Secret Backend for Vault. AWS secret backends can then issue AWS access keys and secret keys, once a role has been added to the backend.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_aws_secret_backend" "aws" {
  access_key = "AKIA....."
```

```
  secret_key = "AWS secret key"
}
```

» Argument Reference

The following arguments are supported:

- `access_key` - (Required) The AWS Access Key ID this backend should use to issue new credentials.
- `secret_key` - (Required) The AWS Secret Key this backend should use to issue new credentials.

Important Because Vault does not support reading the configured credentials back from the API, Terraform cannot detect and correct drift on `access_key` or `secret_key`. Changing the values, however, *will* overwrite the previously stored values.

- `region` - (Optional) The AWS region for API calls. Defaults to `us-east-1`.
- `path` - (Optional) The unique path this backend should be mounted at. Must not begin or end with a `/`. Defaults to `aws`.
- `description` - (Optional) A human-friendly description for this backend.
- `default_lease_ttl_seconds` - (Optional) The default TTL for credentials issued by this backend.
- `max_lease_ttl_seconds` - (Optional) The maximum TTL that can be requested for credentials issued by this backend.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_aws_secret_backend_role

Creates a role on an AWS Secret Backend for Vault. Roles are used to map credentials to the policies that generated them.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_aws_secret_backend" "aws" {
  access_key = "AKIA....."
  secret_key = "AWS secret key"
}

resource "vault_aws_secret_backend_role" "role" {
  backend = "${vault_aws_secret_backend.aws.path}"
  name    = "deploy"

  policy = <<EOT
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
EOT
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Required) The path the AWS secret backend is mounted at, with no leading or trailing `/`s.
- `name` - (Required) The name to identify this role within the backend. Must be unique within the backend.
- `policy` - (Optional) The JSON-formatted policy to associate with this role. Either `policy` or `policy_arn` must be specified.
- `policy_arn` - (Optional) The ARN for a pre-existing policy to associate with this role. Either `policy` or `policy_arn` must be specified.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_database_secret_backend_connection

Creates a Database Secret Backend connection in Vault. Database secret backend connections can be used to generate dynamic credentials for the database.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_mount" "db" {
  path = "postgres"
  type = "database"
}

resource "vault_database_secret_backend_connection" "postgres" {
  backend       = "${vault_mount.db.path}"
  name          = "postgres"
  allowed_roles = ["dev", "prod"]

  postgresql {
    connection_url = "postgres://username:password@host:port/database"
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A unique name to give the database connection.
- `backend` - (Required) The unique name of the Vault mount to configure.
- `verify_connection` - (Optional) Whether the connection should be verified on initial configuration or not.
- `allowed_roles` - (Optional) A list of roles that are allowed to use this connection.
- `cassandra` - (Optional) A nested block containing configuration options for Cassandra connections.
- `mongodb` - (Optional) A nested block containing configuration options for MongoDB connections.

- `hana` - (Optional) A nested block containing configuration options for SAP HanaDB connections.
- `mssql` - (Optional) A nested block containing configuration options for MSSQL connections.
- `mysql` - (Optional) A nested block containing configuration options for MySQL connections.
- `postgresql` - (Optional) A nested block containing configuration options for PostgreSQL connections.
- `oracle` - (Optional) A nested block containing configuration options for Oracle connections.

Exactly one of the nested blocks of configuration options must be supplied.

» Cassandra Configuration Options

- `hosts` - (Required) The hosts to connect to.
- `username` - (Required) The username to authenticate with.
- `password` - (Required) The password to authenticate with.
- `port` - (Optional) The default port to connect to if no port is specified as part of the host.
- `tls` - (Optional) Whether to use TLS when connecting to Cassandra.
- `insecure_tls` - (Optional) Whether to skip verification of the server certificate when using TLS.
- `pem_bundle` - (Optional) Concatenated PEM blocks configuring the certificate chain.
- `pem_json` - (Optional) A JSON structure configuring the certificate chain.
- `protocol_version` - (Optional) The CQL protocol version to use.
- `connect_timeout` - (Optional) The number of seconds to use as a connection timeout.

» MongoDB Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.

» SAP HanaDB Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.
- `max_open_connections` - (Optional) The maximum number of open connections to use.
- `max_idle_connections` - (Optional) The maximum number of idle connections to maintain.
- `max_connection_lifetime` - (Optional) The maximum number of seconds to keep a connection alive for.

» MSSQL Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.
- `max_open_connections` - (Optional) The maximum number of open connections to use.
- `max_idle_connections` - (Optional) The maximum number of idle connections to maintain.
- `max_connection_lifetime` - (Optional) The maximum number of seconds to keep a connection alive for.

» MySQL Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.
- `max_open_connections` - (Optional) The maximum number of open connections to use.
- `max_idle_connections` - (Optional) The maximum number of idle connections to maintain.
- `max_connection_lifetime` - (Optional) The maximum number of seconds to keep a connection alive for.

» PostgreSQL Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.

- `max_open_connections` - (Optional) The maximum number of open connections to use.
- `max_idle_connections` - (Optional) The maximum number of idle connections to maintain.
- `max_connection_lifetime` - (Optional) The maximum number of seconds to keep a connection alive for.

» Oracle Configuration Options

- `connection_url` - (Required) A URL containing connection information. See the Vault docs for an example.
- `max_open_connections` - (Optional) The maximum number of open connections to use.
- `max_idle_connections` - (Optional) The maximum number of idle connections to maintain.
- `max_connection_lifetime` - (Optional) The maximum number of seconds to keep a connection alive for.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_database_secret_backend_role

Creates a Database Secret Backend role in Vault. Database secret backend roles can be used to generate dynamic credentials for the database.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_mount" "db" {
  path = "postgres"
  type = "database"
}

resource "vault_database_secret_backend_connection" "postgres" {
```

```
  backend       = "${vault_mount.db.path}"
  name          = "postgres"
  allowed_roles = ["dev", "prod"]

  postgresql {
    role_url = "postgres://username:password@host:port/database"
  }
}

resource "vault_database_secret_backend_role" "role" {
  backend             = "${vault_mount.db.path}"
  name                = "my-role"
  db_name             = "${vault_database_secret_backend_connection.postgres.name}"
  creation_statements = "CREATE ROLE {{name}} WITH LOGIN PASSWORD '{{password}}' VALID UNTI
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) A unique name to give the role.
- `backend` - (Required) The unique name of the Vault mount to configure.
- `db_name` - (Required) The unique name of the database connection to use for the role.
- `creation_statements` - (Required) The database statements to execute when creating a user.
- `revocation_statements` - (Optional) The database statements to execute when revoking a user.
- `rollback_statements` - (Optional) The database statements to execute when rolling back creation due to an error.
- `renew_statements` - (Optional) The database statements to execute when renewing a user.
- `default_ttl` - (Optional) The default number of seconds for leases for this role.
- `max_ttl` - (Optional) The maximum number of seconds for leases for this role.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_generic_secret

Writes and manages arbitrary data at a given path in Vault.

This resource is primarily intended to be used with Vault's "generic" secret backend, but it is also compatible with any other Vault endpoint that supports the `vault write` command to create and the `vault delete` command to delete.

Important All data provided in the resource configuration will be written in cleartext to state and plan files generated by Terraform, and will appear in the console output when Terraform runs. Protect these artifacts accordingly. See the main provider documentation for more details.

» Example Usage

```
resource "vault_generic_secret" "example" {
  path = "secret/foo"

  data_json = <<EOT
{
  "foo":   "bar",
  "pizza": "cheese"
}
EOT
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The full logical path at which to write the given data. To write data into the "generic" secret backend mounted in Vault by default, this should be prefixed with `secret/`. Writing to other backends with this resource is possible; consult each backend's documentation to see which endpoints support the `PUT` and `DELETE` methods.
- `data_json` - (Required) String containing a JSON-encoded object that will be written as the secret data at the given path.
- `allow_read` - (Optional, Deprecated) True/false. Set this to true if your vault authentication is able to read the data, this allows the resource to be compared and updated. Defaults to false.
- `disable_read` - (Optional) True/false. Set this to true if your vault authentication is not able to read the data. Setting this to `true` will break drift detection. Defaults to false.

» Required Vault Capabilities

Use of this resource requires the `create` or `update` capability (depending on whether the resource already exists) on the given path, along with the `delete` capbility if the resource is removed from configuration.

This resource does not *read* the secret data back from Terraform on refresh by default. This avoids the need for `read` access on the given path, but it means that Terraform is not able to detect and repair "drift" on this resource should the data be updated or deleted outside of Terraform. This limitation can be negated by setting `allow_read` to true

» Attributes Reference

No additional attributes are exported by this resource.

» vault_okta_auth_backend

Provides a resource for managing an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    description = "Demonstration of the Terraform Okta auth backend"
    organization = "example"
    token = "something that should be kept secret"
    group {
        group_name = "foo"
        policies = ["one", "two"]
    }
    user {
        username = "bar"
        groups = ["foo"]
    }
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) Path to mount the Okta auth backend
- `description` - (Optional) The description of the auth backend

- `organization` - (Required) The Okta organization. This will be the first part of the url `https://XXX.okta.com`
- `token` - (Optional) The Okta API token. This is required to query Okta for user group membership. If this is not supplied only locally configured groups will be enabled.
- `base_url` - (Optional) The Okta url. Examples: oktapreview.com, okta.com
- `ttl` - (Optional) Duration after which authentication will be expired. See the documentation for info on valid duration formats.
- `max_ttl` - (Optional) Maximum duration after which authentication will be expired See the documentation for info on valid duration formats.
- `group` - (Optional) Associate Okta groups with policies within Vault. See below for more details.
- `user` - (Optional) Associate Okta users with groups or policies within Vault. See below for more details.

» Okta Group

- `group_name` - (Required) Name of the group within the Okta
- `policies` - (Optional) Vault policies to associate with this group

» Okta User

- `username` - (Required Optional) Name of the user within Okta
- `groups` - (Optional) List of Okta groups to associate with this user
- `policies` - (Optional) List of Vault policies to associate with this user

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_okta_auth_backend_user

Provides a resource to create a user in an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    path = "user_okta"
    organization = "dummy"
}

resource "vault_okta_auth_backend_user" "foo" {
    path = "${vault_okta_auth_backend.example.path}"
    username = "foo"
    groups = ["one", "two"]
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The path where the Okta auth backend is mounted
- `username` - (Required Optional) Name of the user within Okta
- `groups` - (Optional) List of Okta groups to associate with this user
- `policies` - (Optional) List of Vault policies to associate with this user

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_okta_auth_backend_group

Provides a resource to create a group in an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    path = "group_okta"
    organization = "dummy"
}

resource "vault_okta_auth_backend_group" "foo" {
    path = "${vault_okta_auth_backend.example.path}"
    group_name = "foo"
```

```
    policies = ["one", "two"]
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The path where the Okta auth backend is mounted
- `group_name` - (Required) Name of the group within the Okta
- `policies` - (Optional) Vault policies to associate with this group

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_mount

» Example Usage

```
resource "vault_mount" "example" {
  path = "dummy"
  type = "generic"
  description = "This is an example mount"
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) Where the secret backend will be mounted
- `type` - (Required) Type of the backend, such as "aws"
- `description` - (Optional) Human-friendly description of the mount
- `default_lease_ttl_seconds` - (Optional) Default lease duration for tokens and secrets in seconds
- `max_lease_ttl_seconds` - (Optional) Maximum possible lease duration for tokens and secrets in seconds

» Attributes Reference

No additional attributes are exported by this resource.

» vault_okta_auth_backend

Provides a resource for managing an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    description = "Demonstration of the Terraform Okta auth backend"
    organization = "example"
    token = "something that should be kept secret"
    group {
        group_name = "foo"
        policies = ["one", "two"]
    }
    user {
        username = "bar"
        groups = ["foo"]
    }
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) Path to mount the Okta auth backend
- `description` - (Optional) The description of the auth backend
- `organization` - (Required) The Okta organization. This will be the first part of the url `https://XXX.okta.com`
- `token` - (Optional) The Okta API token. This is required to query Okta for user group membership. If this is not supplied only locally configured groups will be enabled.
- `base_url` - (Optional) The Okta url. Examples: oktapreview.com, okta.com
- `ttl` - (Optional) Duration after which authentication will be expired. See the documentation for info on valid duration formats.

- `max_ttl` - (Optional) Maximum duration after which authentication will be expired See the documentation for info on valid duration formats.
- `group` - (Optional) Associate Okta groups with policies within Vault. See below for more details.
- `user` - (Optional) Associate Okta users with groups or policies within Vault. See below for more details.

» Okta Group

- `group_name` - (Required) Name of the group within the Okta
- `policies` - (Optional) Vault policies to associate with this group

» Okta User

- `username` - (Required Optional) Name of the user within Okta
- `groups` - (Optional) List of Okta groups to associate with this user
- `policies` - (Optional) List of Vault policies to associate with this user

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_okta_auth_backend_group

Provides a resource to create a group in an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    path = "group_okta"
    organization = "dummy"
}

resource "vault_okta_auth_backend_group" "foo" {
    path = "${vault_okta_auth_backend.example.path}"
    group_name = "foo"
    policies = ["one", "two"]
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The path where the Okta auth backend is mounted
- `group_name` - (Required) Name of the group within the Okta
- `policies` - (Optional) Vault policies to associate with this group

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_okta_auth_backend_user

Provides a resource to create a user in an Okta auth backend within Vault.

» Example Usage

```
resource "vault_okta_auth_backend" "example" {
    path = "user_okta"
    organization = "dummy"
}

resource "vault_okta_auth_backend_user" "foo" {
    path = "${vault_okta_auth_backend.example.path}"
    username = "foo"
    groups = ["one", "two"]
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) The path where the Okta auth backend is mounted
- `username` - (Required Optional) Name of the user within Okta
- `groups` - (Optional) List of Okta groups to associate with this user
- `policies` - (Optional) List of Vault policies to associate with this user

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_policy

» Example Usage

```
resource "vault_policy" "example" {
  name = "dev-team"

  policy = <<EOT
path "secret/my_app" {
  policy = "write"
}
EOT
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the policy
- `policy` - (Required) String containing a Vault policy

» Attributes Reference

No additional attributes are exported by this resource.