

» 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
# For this example, in Vault there is a key named "auth_token" and the value is the token we
# In general usage, replace "auth_token" with the key you wish to extract from Vault.

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 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_kubernetes_auth_backend_config`

Reads the Role of an Kubernetes from a Vault server. See the Vault documentation for more information.

» Example Usage

```
data "vault_kubernetes_auth_backend_config" "config" {
  backend = "my-kubernetes-backend"
}

output "token_reviewer_jwt" {
  value = "${data.vault_kubernetes_auth_backend_config.config.token_reviewer_jwt}"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Optional) The unique name for the Kubernetes backend the config to retrieve Role attributes for resides in. Defaults to "kubernetes".

» Attributes Reference

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

- `kubernetes_host` - Host must be a host string, a host:port pair, or a URL to the base of the Kubernetes API server.
- `kubernetes_ca_cert` - PEM encoded CA cert for use by the TLS client used to talk with the Kubernetes API.

- **pem_keys** - Optional list of PEM-formatted public keys or certificates used to verify the signatures of Kubernetes service account JWTs. If a certificate is given, its public key will be extracted. Not every installation of Kubernetes exposes these keys.

» vault_kubernetes_auth_backend_role

Reads the Role of an Kubernetes from a Vault server. See the Vault documentation for more information.

» Example Usage

```
data "vault_kubernetes_auth_backend_role" "role" {
  backend  = "my-kubernetes-backend"
  role_name = "my-role"
}

output "policies" {
  value = "${data.vault_kubernetes_auth_backend_role.role.policies}"
}
```

» Argument Reference

The following arguments are supported:

- **role_name** - (Required) The name of the role to retrieve the Role attributes for.
- **backend** - (Optional) The unique name for the Kubernetes backend the role to retrieve Role attributes for resides in. Defaults to "kubernetes".

» Attributes Reference

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

- **bound_cirs** - List of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.
- **bound_service_account_names** - List of service account names able to access this role. If set to "" *all names are allowed, both this and bound_service_account_namespaces can not be ""*.

- `bound_service_account_namespaces` - List of namespaces allowed to access this role. If set to "" *all namespaces are allowed, both this and bound_service_account_names can not be set to ""*.
- `ttl` - The TTL period of tokens issued using this role in seconds.
- `max_ttl` - The maximum allowed lifetime of tokens issued in seconds using this role.
- `num_uses` - Number of times issued tokens can be used. Setting this to 0 or leaving it unset means unlimited uses.
- `period` - 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 parameter.
- `policies` - Policies to be set on tokens issued using this role.

» `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.
- **backend** - (Optional) The unique name of the auth backend to configure. Defaults to **aprole**.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

AppRole authentication backend roles can be imported using the **path**, e.g.

```
$ terraform import vault_aprole_auth_backend_role.example auth/aprole/role/test-role
```


» vault_approle_auth_backend_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_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__audit

» Example Usage (file audit device)

```
resource "vault_audit" "test" {
  type = "file"

  options = {
    file_path = "C:/temp/audit.txt"
  }
}
```

» Example Usage (socket audit device)

```
resource "vault_audit" "test" {
  type = "socket"
  path = "app_socket"

  options = {
    address      = "127.0.0.1:8000"
    socket_type = "tcp"
    description = "application x socket"
  }
}
```

» Argument Reference

The following arguments are supported:

- **type** - (Required) Type of the audit device, such as 'file'.
- **path** - (optional) The path to mount the audit device. This defaults to the type.
- **description** - (Optional) Human-friendly description of the audit device.
- **options** - (Required) Configuration options to pass to the audit device itself.

For a reference of the device types and their options, consult the Vault documentation.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

Audit devices can be imported using the **path**, e.g.

```
$ terraform import vault_audit.test syslog
```

» 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 auth method type
- **path** - (Optional) The path to mount the auth method — this defaults to the name of the type
- **description** - (Optional) A description of the auth method
- **default_lease_ttl_seconds** - (Optional) The default lease duration in seconds.

- `max_lease_ttl_seconds` - (Optional) The maximum lease duration in seconds.
- `listing_visibility` - (Optional) Specifies whether to show this mount in the UI-specific listing endpoint.
- `local` - (Optional) Specifies if the auth method is local only.

» Attributes Reference

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

- `accessor` - The accessor for this auth method

» Import

Auth methods can be imported using the `path`, e.g.

```
$ terraform import vault_auth_backend.example github
```

» `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.

» Import

AWS auth backend certificates can be imported using **auth/**, the **backend** path, **/config/certificate/**, and the **cert_name** e.g.

```
$ terraform import vault_aws_auth_backend_cert.example auth/aws/config/certificate/my-cert
```

» 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.

» Import

AWS auth backend clients can be imported using **auth/**, the **backend** path, and **/config/client** e.g.

```
$ terraform import vault_aws_auth_backend_client.example auth/aws/config/client
```

» **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.

» **Import**

AWS auth backend identity whitelists can be imported using `auth/`, the `backend` path, and `/config/tidy/identity-whitelist` e.g.

```
$ terraform import vault_aws_auth_backend_identity_whitelist.example auth/aws/config/tidy/id
```


» 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_ids   = ["ami-8c1be5f6"]
  bound_account_ids = ["123456789012"]
  bound_vpc_ids   = ["vpc-b61106d4"]
  bound_subnet_ids = ["vpc-133128f1"]
  bound_iam_role_arns = ["arn:aws:iam::123456789012:role/MyRole"]
  bound_iam_instance_profile_arns = ["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_ids** - (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_ids** - (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_regions** - (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_ids** - (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_ids** - (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_arns** - (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_arns** - (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_arns** - (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_arns` are resolved to AWS Unique IDs for the bound principal ARN. This field is ignored when a `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 prior 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 seconds.
- `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.
- `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.

» Import

AWS auth backend roles can be imported using `auth/`, the `backend` path, `/role/`, and the `role` name e.g.

```
$ terraform import vault_aws_auth_backend_role.example auth/aws/role/test-role
```

» vault_aws_auth_backend_role_tag

Reads role tag information from an AWS auth backend in Vault.

» Example Usage

```
resource "vault_auth_backend" "aws" {
  path = "%s"
  type = "aws"
}

resource "vault_aws_auth_backend_role" "role" {
  backend      = "${vault_auth_backend.aws.path}"
  role         = "%s"
  auth_type    = "ec2"
  bound_account_id = "123456789012"
  policies     = ["dev", "prod", "qa", "test"]
  role_tag     = "VaultRoleTag"
}

resource "vault_aws_auth_backend_role_tag" "test" {
  backend      = "${vault_auth_backend.aws.path}"
  role         = "${vault_aws_auth_backend_role.role.role}"
  policies     = ["prod", "dev", "test"]
  max_ttl      = "1h"
  instance_id  = "i-1234567"
}
```

» Argument Reference

The following arguments are supported:

- **role** - (Required) The name of the AWS auth backend role to read role tags from, with no leading or trailing `/s`.
- **backend** - (Optional) The path to the AWS auth backend to read role tags from, with no leading or trailing `/s`. Defaults to `"aws"`.

- **policies** - (Optional) The policies to be associated with the tag. Must be a subset of the policies associated with the role.
- **max_ttl** - (Optional) The maximum TTL of the tokens issued using this role.
- **instance_id** - (Optional) Instance ID for which this tag is intended for. If set, the created tag can only be used by the instance with the given ID.
- **allow_instance_migration** - (Optional) If set, allows migration of the underlying instances where the client resides. Use with caution.
- **disallow_reauthentication** - (Optional) If set, only allows a single token to be granted per instance ID.

» Attributes Reference

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

- **tag_key** - The key of the role tag.
- **tag_value** - The value to set the role key.

» 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.

» Import

AWS auth backend STS roles can be imported using `auth/`, the `backend` path, `/config/sts/`, and the `account_id` e.g.

```
$ terraform import vault_aws_auth_backend_sts_role.example auth/aws/config/sts/1234567890
```

» `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.

» Import

AWS secret backends can be imported using the **path**, e.g.

```
$ terraform import vault_aws_secret_backend.aws aws
```

» 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"
  credential_type = "assumed_role"

  policy_document = <<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_document** - (Optional) The JSON-formatted policy to associate with this role. Either **policy_document** or **policy_arns** must be specified.
- **policy_arns** - (Optional) The ARN for a pre-existing policy to associate with this role. Either **policy_document** or **policy_arns** must be specified.

- `credential_type` - (Required) Specifies the type of credential to be used when retrieving credentials from the role. Must be one of `iam_user`, `assumed_role`, or `federation_token`.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

AWS secret backend roles can be imported using the `path`, e.g.

```
$ terraform import vault_aws_secret_backend_role.role aws/roles/deploy
```

» `vault_azure_auth_backend_config`

Configures the Azure 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 Azure 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 = "azure"
}

resource "vault_azure_auth_backend_config" "example" {
  backend      = "${vault_auth_backend.example.path}"
  tenant_id    = "11111111-2222-3333-4444-555555555555"
  client_id    = "11111111-2222-3333-4444-555555555555"
  client_secret = "01234567890123456789"
  resource     = "https://vault.hashicorp.com"
}
```

» Argument Reference

The following arguments are supported:

- **tenant_id** - (Required) The tenant id for the Azure Active Directory organization.
- **resource** - (Required) The configured URL for the application registered in Azure Active Directory.
- **backend** - (Optional) The path the Azure auth backend being configured was mounted at. Defaults to **azure**.
- **client_id** - (Optional) The client id for credentials to query the Azure APIs. Currently read permissions to query compute resources are required.
- **client_secret** - (Optional) The client secret for credentials to query the Azure APIs.
- **environment** - (Optional) The Azure cloud environment. Valid values: **AzurePublicCloud**, **AzureUSGovernmentCloud**, **AzureChinaCloud**, **AzureGermanCloud**. Defaults to **AzurePublicCloud**.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

Azure auth backends can be imported using **auth/**, the **backend** path, and **/config** e.g.

```
$ terraform import vault_azure_auth_backend_config.example auth/azure/config
```

» vault_azure_auth_backend_role

Manages an Azure 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" "azure" {  
  type = "azure"
```

```

}

resource "vault_azure_auth_backend_role" "example" {
  backend          = "${vault_auth_backend.azure.path}"
  role             = "test-role"
  bound_subscription_ids = ["11111111-2222-3333-4444-555555555555"]
  bound_resource_groups = ["123456789012"]
  ttl             = 60
  max_ttl         = 120
  policies         = ["default", "dev", "prod"]
}

```

» Argument Reference

The following arguments are supported:

- **role** - (Required) The name of the role.
- **bound_service_principal_ids** - (Optional) If set, defines a constraint on the service principals that can perform the login operation that they should be possess the ids specified by this field.
- **bound_group_ids** - (Optional) If set, defines a constraint on the groups that can perform the login operation that they should be using the group ID specified by this field.
- **bound_locations** - (Optional) If set, defines a constraint on the virtual machines that can perform the login operation that the location in their identity document must match the one specified by this field.
- **bound_subscription_ids** - (Optional) If set, defines a constraint on the subscriptions that can perform the login operation to ones which matches the value specified by this field.
- **bound_resource_groups** - (Optional) If set, defines a constraint on the virtual machines that can perform the login operation that they be associated with the resource group that matches the value specified by this field.
- **bound_scale_sets** - (Optional) If set, defines a constraint on the virtual machines that can perform the login operation that they must match the scale set specified by this field.
- **ttl** - (Optional) The TTL period of tokens issued using this role, provided as a number of seconds.
- **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.
- **policies** - (Optional) An array of strings specifying the policies to be set on tokens issued using this role.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

Azure auth backend roles can be imported using `auth/`, the `backend` path, `/role/`, and the `role` name e.g.

```
$ terraform import vault_azure_auth_backend_role.example auth/azure/role/test-role
```

» vault__cert__auth__backend__role

Provides a resource to create a role in an Cert auth backend within Vault.

» Example Usage

```
resource "vault_auth_backend" "cert" {
  path = "cert"
  type = "cert"
}

resource "vault_cert_auth_backend_role" "cert" {
  name           = "foo"
  certificate     = "${file("/path/to/certs/ca-cert.pem")}"
  backend        = "${vault_auth_backend.cert.path}"
  allowed_names  = ["foo.example.org", "baz.example.org"]
  ttl            = 300
  max_ttl        = 600
  policies       = ["foo"]
}
```

» Argument Reference

The following arguments are supported:

- **name** - (Required) Name of the role
- **certificate** - (Required) CA certificate used to validate client certificates
- **allowed_names** - (Optional) Allowed subject names for authenticated client certificates
- **allowed_common_names** - (Optional) Allowed the common names for authenticated client certificates
- **allowed_dns_sans** - (Optional) Allowed alternative dns names for authenticated client certificates
- **allowed_email_sans** - (Optional) Allowed emails for authenticated client certificates
- **allowed_uri_sans** - (Optional) Allowed URIs for authenticated client certificates
- **allowed_organization_units** - (Optional) Allowed organization units for authenticated client certificates
- **required_extensions** - (Optional) TLS extensions required on client certificates
- **ttl** - (Optional) Default TTL of tokens issued by the backend
- **max_ttl** - (Optional) Maximum TTL of tokens issued by the backend
- **period** - (Optional) Duration in seconds for token. If set, the issued token is a periodic token.
- **policies** - (Optional) Policies to grant on the issued token
- **display_name** - (Optional) The name to display on tokens issued under this role.
- **bound_cidrs** - (Optional) Restriction usage of the certificates to client IPs falling within the range of the specified CIDRs
- **backend** - (Optional) Path to the mounted Cert auth backend

For more details on the usage of each argument consult the Vault Cert API documentation.

» Attribute Reference

No additional attributes are exposed by this resource.

» vault_consul_secret_backend

Creates a Consul Secret Backend for Vault. Consul secret backends can then issue Consul tokens, 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_consul_secret_backend" "test" {
  path          = "consul"
  description   = "Manages the Consul backend"

  address = "127.0.0.1:8500"
  token   = "4240861b-ce3d-8530-115a-521ff070dd29"
}
```

» Argument Reference

The following arguments are supported:

- **token** - (Required) The Consul management token this backend should use to issue new tokens.

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

- **path** - (Optional) The unique location this backend should be mounted at. Must not begin or end with a /. Defaults to **consul**.
- **description** - (Optional) A human-friendly description for this backend.
- **address** - (Required) Specifies the address of the Consul instance, provided as "host:port" like "127.0.0.1:8500".
- **scheme** - (Optional) Specifies the URL scheme to use. Defaults to **http**.
- **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.

» Import

Consul secret backends can be imported using the `path`, e.g.

```
$ terraform import vault_consul_secret_backend.example consul
```

» 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.
- **mysql_rds** - (Optional) A nested block containing configuration options for RDS MySQL connections.
- **mysql_aurora** - (Optional) A nested block containing configuration options for Aurora MySQL connections.
- **mysql_legacy** - (Optional) A nested block containing configuration options for legacy 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.

» Import

Database secret backend connections can be imported using the `backend`, `/config/`, and the `name` e.g.

```
$ terraform import vault_database_secret_backend_connection.example postgres/config/postgres
```

» 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 {
    connection_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 U
```

» 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.

» Import

Database secret backend roles can be imported using the **backend**, **/roles/**, and the **name** e.g.

```
$ terraform import vault_database_secret_backend_role.example postgres/roles/my-role
```

» vault_gcp_auth_backend_role

Provides a resource to create a role in an GCP auth backend within Vault.

» Example Usage

```
resource "vault_auth_backend" "gcp" {
  path = "gcp"
  type = "gcp"
}

resource "vault_gcp_auth_backend_role" "gcp" {
  backend          = "${vault_auth_backend.cert.path}"
  project_id       = "foo-bar-baz"
  bound_service_accounts = ["database-server@foo-bar-baz.iam.gserviceaccount.com"]
  policies         = ["database-server"]
}
```

» Argument Reference

The following arguments are supported:

- **role** - (Required) Name of the GCP role
- **type** - (Required) Type of GCP authentication role (either `gce` or `iam`)
- **project_id** - (Optional; Deprecated, use `bound_projects` instead) GCP Project that the role exists within
- **bound_projects** - (Optional) An array of GCP project IDs. Only entities belonging to this project can authenticate under the role.
- **ttl** - (Optional) Default TTL of tokens issued by the backend
- **max_ttl** - (Optional) Maximum TTL of tokens issued by the backend
- **period** - (Optional) Duration in seconds for token. If set, the issued token is a periodic token.
- **policies** - (Optional) Policies to grant on the issued token
- **backend** - (Optional) Path to the mounted GCP auth backend
- **bound_service_accounts** - (Optional) GCP Service Accounts allowed to issue tokens under this role. (Note: **Required** if role is `iam`)

» iam-only Parameters

- **max_jwt_exp** - (Optional) The number of seconds past the time of authentication that the login param JWT must expire within. For example, if a user attempts to login with a token that expires within an hour and this is set to 15 minutes, Vault will return an error prompting the user to

create a new signed JWT with a shorter `exp`. The GCE metadata tokens currently do not allow the `exp` claim to be customized.

- `allow_gce_inference` - (Optional) A flag to determine if this role should allow GCE instances to authenticate by inferring service accounts from the GCE identity metadata token.

» `gce-only` Parameters

The following parameters are only valid when the role is of type `"gce"`:

- `bound_zones` - (Optional) The list of zones that a GCE instance must belong to in order to be authenticated. If `bound_instance_groups` is provided, it is assumed to be a zonal group and the group must belong to this zone.
- `bound_regions` - (Optional) The list of regions that a GCE instance must belong to in order to be authenticated. If `bound_instance_groups` is provided, it is assumed to be a regional group and the group must belong to this region. If `bound_zones` are provided, this attribute is ignored.
- `bound_instance_groups` - (Optional) The instance groups that an authorized instance must belong to in order to be authenticated. If specified, either `bound_zones` or `bound_regions` must be set too.
- `bound_labels` - (Optional) A comma-separated list of GCP labels formatted as `"key:value"` strings that must be set on authorized GCE instances. Because GCP labels are not currently ACL'd, we recommend that this be used in conjunction with other restrictions.
- `bound_projects` - (Optional) GCP Projects that the role exists within

For more details on the usage of each argument consult the Vault GCP API documentation.

» Attribute Reference

No additional attributes are exposed 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** capability 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.

» Import

Generic secrets can be imported using the **path**, e.g.

```
$ terraform import vault_mount.example secret/foo
```

» vault_identity_entity

Creates an Identity Entity for Vault. The Identity secrets engine is the identity management solution for Vault. It internally maintains the clients who are recognized by Vault.

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_identity_entity" "test" {
  name      = "tester1"
  policies  = ["test"]
  metadata = {
    foo = "bar"
  }
}
```

» Argument Reference

The following arguments are supported:

- **name** - (Required) Name of the identity entity to create.

- **policies** - (Optional) A list of policies to apply to the entity.
- **metadata** - (Optional) A Map of additional metadata to associate with the user.
- **disabled** - (Optional) True/false Is this entity currently disabled. Defaults to **false**

» Attributes Reference

- **id** - The id of the created entity.

» vault_identity_entity_alias

Creates an Identity Entity Alias for Vault.

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_identity_entity_alias" "test" {
  name           = "user_1"
  mount_accessor = "token_1f2bd5"
  canonical_id   = "49877D63-07AD-4B85-BDA8-B61626C477E8"
}
```

» Argument Reference

The following arguments are supported:

- **name** - (Required) Name of the alias. Name should be the identifier of the client in the authentication source. For example, if the alias belongs to userpass backend, the name should be a valid username within userpass backend. If alias belongs to GitHub, it should be the GitHub username.
- **mount_accessor** - (Required) Accessor of the mount to which the alias should belong to.
- **canonical_id** - (Required) Entity ID to which this alias belongs to.

» Attributes Reference

- `id` - ID of the entity alias.

» `vault_jwt_auth_backend`

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

» Example Usage

```
resource "vault_jwt_auth_backend" "example" {
  description = "Demonstration of the Terraform JWT auth backend"
  path        = "jwt"
  oidc_discovery_url = "https://myco.auth0.com/"
  bound_issuer = "https://myco.auth0.com/"
}
```

» Argument Reference

The following arguments are supported:

- `path` - (Required) Path to mount the JWT auth backend
- `description` - (Optional) The description of the auth backend
- `oidc_discovery_url` - (Optional) The OIDC Discovery URL, without any .well-known component (base path). Cannot be used in combination with `jwt_validation_pubkeys`
- `bound_issuer` - (Optional) The value against which to match the iss claim in a JWT
- `oidc_discovery_ca_pem` - (Optional) The CA certificate or chain of certificates, in PEM format, to use to validate connections to the OIDC Discovery URL. If not set, system certificates are used
- `jwt_validation_pubkeys` - (Optional) A list of PEM-encoded public keys to use to authenticate signatures locally. Cannot be used in combination with `oidc_discovery_url`

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_jwt_auth_backend_role

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

» Example Usage

```
resource "vault_jwt_auth_backend" "jwt" {
  path = "jwt"
}

resource "vault_jwt_auth_backend_role" "example" {
  backend      = "${vault_jwt_auth_backend.jwt.path}"
  role_name    = "test-role"
  policies     = ["default", "dev", "prod"]

  bound_audiences = ["https://myco.test"]
  user_claim      = "https://vault/user"
}
```

» Argument Reference

The following arguments are supported:

- **role_name** - (Required) The name of the role.
- **bound_audiences** - (Required) List of **aud** claims to match against. Any match is sufficient.
- **user_claim** - (Required) The claim to use to uniquely identify the user; this will be used as the name for the Identity entity alias created due to a successful login.
- **policies** - (Optional) Policies to be set on tokens issued using this role.
- **ttl** - (Optional) The initial/renewal TTL of tokens issued using this role, in seconds.
- **max_ttl** - (Optional) The maximum allowed lifetime of tokens issued using this role, in seconds.
- **period** - (Optional) If set, indicates that the token generated using this role should never expire, but instead always use the value set here as the TTL for every renewal.
- **num_uses** - (Optional) If set, puts a use-count limitation on the issued token.

- **bound_subject** - (Optional) If set, requires that the **sub** claim matches this value.
- **bound_cidrs** - (Optional) If set, a list of CIDRs valid as the source address for login requests. This value is also encoded into any resulting token.
- **groups_claim** - (Optional) The claim to use to uniquely identify the set of groups to which the user belongs; this will be used as the names for the Identity group aliases created due to a successful login. The claim value must be a list of strings.
- **backend** - (Optional) The unique name of the auth backend to configure. Defaults to **jwt**.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

JWT authentication backend roles can be imported using the **path**, e.g.

```
$ terraform import vault_jwt_auth_backend_role.example auth/jwt/role/test-role
```

» vault_kubernetes_auth_backend_config

Manages an Kubernetes auth backend config in a Vault server. See the Vault documentation for more information.

» Example Usage

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

resource "vault_kubernetes_auth_backend_config" "example" {
  backend          = "${vault_auth_backend.kubernetes.path}"
  kubernetes_host  = "http://example.com:443"
  kubernetes_ca_cert = "-----BEGIN CERTIFICATE-----\nexample\n-----END CERTIFICATE-----"
  token_reviewer_jwt = "ZXhhbXBsZQo="
}
```

» Argument Reference

The following arguments are supported:

- `kubernetes_host` - (Required) Host must be a host string, a host:port pair, or a URL to the base of the Kubernetes API server.
- `kubernetes_ca_cert` - (Optional) PEM encoded CA cert for use by the TLS client used to talk with the Kubernetes API.
- `token_reviewer_jwt` - (Optional) A service account JWT used to access the TokenReview API to validate other JWTs during login. If not set the JWT used for login will be used to access the API.
- `pem_keys` - (Optional) List of PEM-formatted public keys or certificates used to verify the signatures of Kubernetes service account JWTs. If a certificate is given, its public key will be extracted. Not every installation of Kubernetes exposes these keys.

» Attributes Reference

No additional attributes are exported by this resource.

» `vault_kubernetes_auth_backend_role`

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

» Example Usage

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

resource "vault_kubernetes_auth_backend_role" "example" {
  backend          = "${vault_auth_backend.kubernetes.path}"
  role_name        = "example-role"
  bound_service_account_names = ["example"]
  bound_service_account_namespaces = ["example"]
  ttl              = 3600
  policies         = ["default", "dev", "prod"]
}
```

» Argument Reference

The following arguments are supported:

- **role_name** - (Required) Name of the role.
- **bound_service_account_names** - (Required) List of service account names able to access this role. If set to ["*"] all names are allowed, both this and **bound_service_account_namespaces** can not be "*".
- **bound_service_account_namespaces** - (Required) List of namespaces allowed to access this role. If set to ["*"] all namespaces are allowed, both this and **bound_service_account_names** can not be set to "*".
- **bound_cirs** - (Optional) List of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.
- **ttl** - (Optional) The TTL period of tokens issued using this role in seconds.
- **max_ttl** - (Optional) The maximum allowed lifetime of tokens issued in seconds using this role.
- **num_uses** - (Optional) Number of times issued tokens can be used. Setting this to 0 or leaving it unset means unlimited uses.
- **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 parameter.
- **policies** - (Optional) Policies to be set on tokens issued using this role.
- **backend** - (Optional) Unique name of the kubernetes backend to configure.

» Attributes Reference

No additional attributes are exported by this resource.

» vault_ldap_auth_backend

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

» Example Usage

```
resource "vault_ldap_auth_backend" "ldap" {  
  path      = "ldap"
```



```

url          = "ldaps://dc-01.example.org"
userdn       = "OU=Users,OU=Accounts,DC=example,DC=org"
userattr     = "sAMAccountName"
upndomain    = "EXAMPLE.ORG"
discoverdn   = false
groupdn      = "OU=Groups,DC=example,DC=org"
groupfilter  = "(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={.UserDN}))"
}

```

» Argument Reference

The following arguments are supported:

- **url** - (Required) The URL of the LDAP server
- **starttls** - (Optional) Control use of TLS when connecting to LDAP
- **tls_min_version** - (Optional) Minimum acceptable version of TLS
- **tls_max_version** - (Optional) Maximum acceptable version of TLS
- **insecure_tls** - (Optional) Control whether or TLS certificates must be validated
- **certificate** - (Optional) Trusted CA to validate TLS certificate
- **binddn** - (Optional) DN of object to bind when performing user search
- **bindpass** - (Optional) Password to use with **binddn** when performing user search
- **userdn** - (Optional) Base DN under which to perform user search
- **userattr** - (Optional) Attribute on user object matching username passed in
- **upndomain** - (Optional) The `userPrincipalDomain` used to construct UPN string
- **discoverdn** - (Optional) Use anonymous bind to discover the bind DN of a user.
- **deny_null_bind** - (Optional) Prevents users from bypassing authentication when providing an empty password.
- **upndomain** - (Optional) The `userPrincipalDomain` used to construct the UPN string for the authenticating user.
- **groupfilter** - (Optional) Go template used to construct group membership query
- **groupdn** - (Optional) Base DN under which to perform group search

- **groupattr** - (Optional) LDAP attribute to follow on objects returned by **groupfilter**
- **path** - (Optional) Path to mount the LDAP auth backend under
- **description** - (Optional) Description for the LDAP auth backend mount

For more details on the usage of each argument consult the Vault LDAP API documentation.

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

» Attributes Reference

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

- **accessor** - The accessor for this auth mount.

» vault_ldap_auth_backend_user

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

» Example Usage

```
resource "vault_ldap_auth_backend" "ldap" {
  path      = "ldap"
  url       = "ldaps://dc-01.example.org"
  userdn    = "OU=Users,OU=Accounts,DC=example,DC=org"
  userattr  = "sAMAccountName"
  upndomain = "EXAMPLE.ORG"
  discoverdn = false
  groupdn   = "OU=Groups,DC=example,DC=org"
  groupfilter = "(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={.UserDN}))"
}

resource "vault_ldap_auth_backend_user" "user" {
  username = "test-user"
  policies = ["dba", "sysops"]
  backend  = "${vault_ldap_auth_backend.ldap.path}"
}
```

» Argument Reference

The following arguments are supported:

- **username** - (Required) The LDAP username
- **policies** - (Optional) Policies which should be granted to user
- **groups** - (Optional) Override LDAP groups which should be granted to user
- **backend** - (Optional) Path to the authentication backend

For more details on the usage of each argument consult the Vault LDAP API documentation.

» Attribute Reference

No additional attributes are exposed by this resource.

» vault_ldap_auth_backend_group

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

» Example Usage

```
resource "vault_ldap_auth_backend" "ldap" {
  path      = "ldap"
  url       = "ldaps://dc-01.example.org"
  userdn    = "OU=Users,OU=Accounts,DC=example,DC=org"
  userattr  = "sAMAccountName"
  upndomain = "EXAMPLE.ORG"
  discoverdn = false
  groupdn   = "OU=Groups,DC=example,DC=org"
  groupfilter = "(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={.UserDN}))"
}

resource "vault_ldap_auth_backend_group" "group" {
  groupname = "dba"
  policies  = ["dba"]
  backend   = "${vault_ldap_auth_backend.ldap.path}"
}
```

» Argument Reference

The following arguments are supported:

- **groupname** - (Required) The LDAP groupname
- **policies** - (Optional) Policies which should be granted to members of the group
- **backend** - (Optional) Path to the authentication backend

For more details on the usage of each argument consult the Vault LDAP API documentation.

» Attribute 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
- **options** - (Optional) Specifies mount type specific options that are passed to the backend

» Attributes Reference

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

- **accessor** - The accessor for this mount.

» Import

Mounts can be imported using the **path**, e.g.

```
$ terraform import vault_mount.example dummy
```

» 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
- **bypass_okta_mfa** - (Optional) When true, requests by Okta for a MFA check will be bypassed. This also disallows certain status checks on the account, such as whether the password is expired.
- **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_pki_secret_backend

Creates an PKI Secret Backend for Vault. PKI secret backends can then issue certificates, once a role has been added to the backend.

» Example Usage

```
resource "vault_pki_secret_backend" "pki" {  
  path = "pki"  
  default_lease_ttl_seconds = 3600  
  max_lease_ttl_seconds = 86400  
}
```

» Argument Reference

The following arguments are supported:

- **path** - (Required) The unique path this backend should be mounted at. Must not begin or end with a /.
- **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.

» Import

PKI secret backends can be imported using the `path`, e.g.

```
$ terraform import vault_pki_secret_backend.pki pki
```

» vault_pki_secret_backend_cert

Generates a certificate from the PKI Secret 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_pki_secret_backend_cert" "app" {
  depends_on = [ "vault_pki_secret_backend_role.admin" ]

  backend = "${vault_pki_secret_backend.intermediate.path}"
  name = "${vault_pki_secret_backend_role.test.name}"

  common_name = "app.my.domain"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Required) The PKI secret backend the resource belongs to.
- `name` - (Required) Name of the role to create the certificate against
- `common_name` - (Required) CN of certificate to create
- `alt_names` - (Optional) List of alternative names
- `ip_sans` - (Optional) List of alternative IPs
- `other_sans` - (Optional) List of other SANs

- `ttl` - (Optional) Time to leave
- `format` - (Optional) The format of data
- `private_key_format` - (Optional) The private key format
- `exclude_cn_from_sans` - (Optional) Flag to exclude CN from SANs

» Attributes Reference

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

- `certificate` - The certificate
- `issuing_ca` - The issuing CA
- `ca_chain` - The CA chain
- `private_key` - The private key
- `private_key_type` - The private key type
- `serial_number` - The serial number

» `vault_pki_secret_backend_config_ca`

Submits the CA information to a PKI Secret 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_pki_secret_backend_config_ca" "intermediate" {
  depends_on = [ "vault_pki_secret_backend.intermediate" ]

  backend = "${vault_pki_secret_backend.intermediate.path}"
  pem_bundle = <<EOT
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAwvEHeJcXnFgi88rE1dTX6FHdBPK0wSjedh0ywVnCZxLWbBv/
5PytjTcCPdrfW7g2sfbPwOge/WF3X2KeYSP8SxZA0czmz6QDspeG921JkZWtyp5o
++N01eLTIUAhq339p301onA0U01k4sHfmCwfrDpTn2hcx4URa5Pzzb1fHigusjIH
1mcGdncaA6Z2Cz01w4E8kPOUukIDrcZT4fa0ZrWUIQZKQw2JzTyKJ+ZMDCZq2TFz
WwpL3eG48wB7J7mibFQ/9nFvxpIf1BjDAZ8QiqkwYr5N0DNsTxcfTCSeubfJDCUf
IWwFZhLitzw0xazazUQKXX/SPMQ11/L9o3nnHwIDAQABAoIBAAQidJQcDPs162fc
```

Txxx7TpiMhvewfKu2TkMGX18V+EzxxR364+BxHSQTB3fvIkHeTGBGJrw0WdyX8PI
Ja/NwZYeHLXWcLbKtcFd8WDiEoNh910q1HMzOc/MBcpYv94RSAX7MEkHs2YIAvHE
RufFV86hVhC1d/JLYjkz5CHi+Fd9XTYjBK78tHhJd4IJPu5LYvwlmc1zeS7s1Tg
QW1FQuVDV8tWa4PMTTrQHwfaGqn95AKc+tbg+ubpCiWl5bBNI3Ghuh4sAC9dMdAkd
w27i2909/Y3XJSSGUZ1ZqDBP4YU388RgHpzLDUxgRcaQt9vdeEz6frULPW67e9D2
mPPDzjECgYEA4aP0wvnSwGoOKsS6vANGy4Ajsq09PR+11tMJUR5kD1XGuZW172eX
3/GAnovDuCp0tbYt0r7Fmkfel00re7SYM18TH5QGpPddcZLvKUf7AchCIOYY0Te3
pS9+7S1lEGrLXyuox4N260v6wHVrmZTcQoZsDwbjYxNNsNACsiQnjGMCgYEA3SvQ
Jets9e9SgNVvao2TijX+/vcNKRfcWB71T9Xc4BuSNEu5+ZLtp1waSnVCVu1Xilk
sWDh+3EhByl4EteENPvE/7A2s1sfCD0prvg0r52aBZKeTp0AukrT8+Ad4hap7g1x
2Lz11MFDkhRqt2KqQaIL+5Mq5WfptbBJOYI7ARUCgYAD6iSfK1hlsDFYupsGwgPL
agi0g97pHZC38ida0e3AdeqBs79xb9mpr/XsSj52Bn6J3IRFALxK5e5Nr4XdGo/9
bCvXw2iuGgCMBOGTVMVdDY1gJr3Ne2r70ay5Dq2PMFsg5pACDhzVA6sRBbh9LKD5
on1jaiKNyHrzkihIo0l/QwKBgA+0v2uLbfS2yvTpDpd0Miyss603r6N0XF+0fe8J
uinBhr1K/mAB59muveuh18Z6vv1KqByaFgtb39jJH+Eja9dWRns95/sh08pOuAbo
yrv3uBfgQmaBQMXZ8aLcBv4aXgWyyG1YkWP1fL2oLMZq6RGQ9WEeqX8c0ImjmrA
YGopAoGBAJZPF1zi2Rfq4MfFZp/X1/zM09hphZwkxkSI+RnsjDUjTgB8CuQu15ep
KWE98yLw4C25Cqw5fKKQ2addizLnZCAIfJKVNRjYlWlWyGQydDEUzqwX1SLS9LVX
LxLkWDajIyjeFn21Ttb42L9pBo3TAQIXuEnom/1P2SQTVCKBiPai
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIDazCCA10gAwIBAgIUahce2sC07Bom/Rznd5HsNA1r1NgwDQYJKoZIhvcNAQEL
BQAwRTElMAkGA1UEBhMCVUxEzARBgNVBAGMC1NvbWUuU3RhdGUxITAfBgNVBAoM
GEludGVybWV0IFdpZGdpdHMgUHR5IEUxOZDAeFw0xODEyMDIwMTAxNDRAfW00NjEy
MTUwMTAxNDRAeUxhZG9uZG9uZG9uZG9uZG9uZG9uZG9uZG9uZG9uZG9uZG9uZG9u
HwYDVQQKBHJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggEiMA0GCSqGSIb3DQEBA
QUAA4IBDwAwggEKAoIBAQCDC8Qd4kJecWCLzysTV1NfoUd0E8rTBKN52HTLBWcJn
EtZsG//k/K2NNwI92t9buDax9s/A6B79YXdfYp5hI/xLFkDRz0bPpA0yl4b3bUmR
la3Knmj743SV4tMhQCGrff2nc7WicA5Q7WTiwd+YLB+s010faFzHhRfrk/PNvV8e
KC6yMgfWZwZ2dxoDpnYLM7XDgTyQ85S6Qg0tx1Ph9o5mtZQhBkpDDYnNPION5kwM
JmrZMXNbCkvd4bjzAHsnuaJsVD/2cW/Gkh+UGMMBnxCKqTBivk3QM2xPFx9MJJ65
t8kMJR8hbAVmEuK3PA7FrNrNRApdf9I8xDWX8v2jeecfAgMBAAGjUzBRMBOGA1Ud
DgQWBBQXGfrns80qxTGKsXG5pDZS/WyyYDAfBgNVHSMEGDAWgBQXGfrns80qxTGK
sXG5pDZS/WyyYDAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQCt
8aUX26c12PgdiEByZSHAX5G+2b0IEtTclPk14uDyyKRY4dVq6gK3ueVSU5eUmBip
JbV5aRetovG0cV//8vbxkZm/ntQ80o+2sfGR51Izd0Udl0r5pkD6g3bFy/zJ+4DR
DAe8fk1Uacfz6CFmD+H8GyHm+fKmf+mjr4o0GQW60egRDJHuiipUk21JyuXdlPSa
FpNR02sGbjn000ANinFgnFiVzGDnxO/G1Kii/6GWrI6rrdVmXioQzF+8AloWckeB
+hmbwbkwQa/JrLb5SWcBDOXSgtn1Li3XF5AQQBBjA3p0lyBXqnI94Irw89Lv9uPT
MUR4qFxeUOW/GJGccMud
-----END CERTIFICATE-----
EOT
}

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The PKI secret backend the resource belongs to.
- **pem_bundle** - (Required) The key and certificate PEM bundle

» Attributes Reference

No additional attributes are exported by this resource.

» vault_pki_secret_backend_intermediate_cert_request

Generates a new private key and a CSR for signing the PKI Secret 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_pki_secret_backend_intermediate_cert_request" "test" {  
  depends_on = [ "vault_pki_secret_backend.pki" ]  
  
  backend = "${vault_pki_secret_backend.pki.path}"  
  
  type = "internal"  
  common_name = "app.my.domain"  
}
```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The PKI secret backend the resource belongs to.
- **type** - (Required) Type of intermediate to create. Must be either `"exported"` or `"internal"`
- **common_name** - (Required) CN of intermediate to create
- **alt_names** - (Optional) List of alternative names

- `ip_sans` - (Optional) List of alternative IPs
- `other_sans` - (Optional) List of other SANs
- `format` - (Optional) The format of data
- `private_key_format` - (Optional) The private key format
- `key_type` - (Optional) The desired key type
- `key_bits` - (Optional) The number of bits to use
- `exclude_cn_from_sans` - (Optional) Flag to exclude CN from SANs
- `ou` - (Optional) The organization unit
- `organization` - (Optional) The organization
- `country` - (Optional) The country
- `locality` - (Optional) The locality
- `province` - (Optional) The province
- `street_address` - (Optional) The street address
- `postal_code` - (Optional) The postal code

» Attributes Reference

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

- `csr` - The CSR
- `private_key` - The private key
- `private_key_type` - The private key type
- `serial_number` - The serial number

» `vault__pki__secret__backend__intermediate__set__signed`

Submits the CA certificate to the PKI Secret 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_pki_secret_backend_intermediate_set_signed" "intermediate" {
  backend = "${vault_pki_secret_backend.intermediate.path}"

  certificate = "<...>"
}
```

» Argument Reference

The following arguments are supported:

- `backend` - (Required) The PKI secret backend the resource belongs to.
- `certificate` - (Required) The certificate

» Attributes Reference

No additional attributes are exported by this resource.

» vault_pki_secret_backend_role

Creates a role on an PKI Secret Backend for Vault.

» Example Usage

```
resource "vault_pki_secret_backend" "pki" {
  path = "%s"
  default_lease_ttl_seconds = 3600
  max_lease_ttl_seconds = 86400
}

resource "vault_pki_secret_backend_role" "role" {
  backend = "${vault_pki_secret_backend.pki.path}"
  name    = "my_role"
}
```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The path the PKI 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.
- **ttl** - (Optional) The TTL
- **max_ttl** - (Optional) The maximum TTL
- **allow_localhost** - (Optional) Flag to allow certificates for localhost
- **allowed_domains** - (Optional) List of allowed domains for certificates
- **allow_bare_domains** - (Optional) Flag to allow certificates matching the actual domain
- **allow_subdomains** - (Optional) Flag to allow certificates matching subdomains
- **allow_glob_domains** - (Optional) Flag to allow names containing glob patterns.
- **allow_any_name** - (Optional) Flag to allow any name
- **enforce_hostnames** - (Optional) Flag to allow only valid host names
- **allow_ip_sans** - (Optional) Flag to allow IP SANs
- **allowed_uri_sans** - (Optional) Defines allowed URI SANs
- **allowed_other_sans** - (Optional) Defines allowed custom SANs
- **server_flag** - (Optional) Flag to specify certificates for server use
- **client_flag** - (Optional) Flag to specify certificates for client use
- **code_signing_flag** - (Optional) Flag to specify certificates for code signing use
- **email_protection_flag** - (Optional) Flag to specify certificates for email protection use
- **key_type** - (Optional) The type of generated keys
- **key_bits** - (Optional) The number of bits of generated keys
- **key_usage** - (Optional) Specify the allowed key usage constraint on issued certificates
- **ext_key_usage** - (Optional) Specify the allowed extended key usage constraint on issued certificates
- **use_csr_common_name** - (Optional) Flag to use the CN in the CSR
- **use_csr_sans** - (Optional) Flag to use the SANs in the CSR
- **ou** - (Optional) The organization unit of generated certificates

- **organization** - (Optional) The organization of generated certificates
- **country** - (Optional) The country of generated certificates
- **locality** - (Optional) The locality of generated certificates
- **province** - (Optional) The province of generated certificates
- **street_address** - (Optional) The street address of generated certificates
- **postal_code** - (Optional) The postal code of generated certificates
- **generate_lease** - (Optional) Flag to generate leases with certificates
- **no_store** - (Optional) Flag to not store certificates in the storage backend
- **require_cn** - (Optional) Flag to force CN usage
- **policy_identifiers** - (Optional) Specify the list of allowed policies IODs
- **basic_constraints_valid_for_non_ca** - (Optional) Flag to mark basic constraints valid when issuing non-CA certificates

» Attributes Reference

No additional attributes are exported by this resource.

» Import

PKI secret backend roles can be imported using the **path**, e.g.

```
$ terraform import vault_pki_secret_backend_role.role pki/roles/my_role
```

» vault_pki_secret_backend_root_cert

Generates a new self-signed CA certificate and private keys for the PKI Secret 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_pki_secret_backend_root_cert" "test" {
  depends_on = [ "vault_pki_secret_backend.pki" ]
}
```



```

backend = "${vault_pki_secret_backend.pki.path}"

type = "internal"
common_name = "Root CA"
ttl = "315360000"
format = "pem"
private_key_format = "der"
key_type = "rsa"
key_bits = 4096
exclude_cn_from_sans = true
ou = "My OU"
organization = "My organization"
}

```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The PKI secret backend the resource belongs to.
- **type** - (Required) Type of intermediate to create. Must be either `\`”exported\” or `\`”internal\”
- **common_name** - (Required) CN of intermediate to create
- **alt_names** - (Optional) List of alternative names
- **ip_sans** - (Optional) List of alternative IPs
- **uri_sans** - (Optional) List of alternative URIs
- **other_sans** - (Optional) List of other SANs
- **ttl** - (Optional) Time to leave
- **format** - (Optional) The format of data
- **private_key_format** - (Optional) The private key format
- **key_type** - (Optional) The desired key type
- **key_bits** - (Optional) The number of bits to use
- **max_path_length** - (Optional) The maximum path length to encode in the generated certificate
- **exclude_cn_from_sans** - (Optional) Flag to exclude CN from SANs
- **permitted_dns_domains** - (Optional) List of domains for which certificates are allowed to be issued
- **ou** - (Optional) The organization unit

- `organization` - (Optional) The organization
- `country` - (Optional) The country
- `locality` - (Optional) The locality
- `province` - (Optional) The province
- `street_address` - (Optional) The street address
- `postal_code` - (Optional) The postal code

» Attributes Reference

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

- `certificate` - The certificate
- `issuing_ca` - The issuing CA
- `private_key` - The private key
- `serial` - The serial

» `vault_pki_secret_backend_root_sign_intermediate`

Creates an PKI certificate.

» Example Usage

```
resource "vault_pki_secret_backend_root_sign_intermediate" "root" {
  depends_on = [ "vault_pki_secret_backend_intermediate_cert_request.intermediate" ]

  backend = "${vault_pki_secret_backend.root.path}"

  csr = "${vault_pki_secret_backend_intermediate_cert_request.intermediate.csr}"
  common_name = "Intermediate CA"
  exclude_cn_from_sans = true
  ou = "My OU"
  organization = "My organization"
}
```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The PKI secret backend the resource belongs to.
- **csr** - (Required) The CSR
- **common_name** - (Required) CN of intermediate to create
- **alt_names** - (Optional) List of alternative names
- **ip_sans** - (Optional) List of alternative IPs
- **uri_sans** - (Optional) List of alternative URIs
- **other_sans** - (Optional) List of other SANs
- **t1** - (Optional) Time to leave
- **format** - (Optional) The format of data
- **private_key_format** - (Optional) The private key format
- **key_type** - (Optional) The desired key type
- **key_bits** - (Optional) The number of bits to use
- **max_path_length** - (Optional) The maximum path length to encode in the generated certificate
- **exclude_cn_from_sans** - (Optional) Flag to exclude CN from SANs
- **use_csr_values** - (Optional) Preserve CSR values
- **permitted_dns_domains** - (Optional) List of domains for which certificates are allowed to be issued
- **ou** - (Optional) The organization unit
- **organization** - (Optional) The organization
- **country** - (Optional) The country
- **locality** - (Optional) The locality
- **province** - (Optional) The province
- **street_address** - (Optional) The street address
- **postal_code** - (Optional) The postal code

» Attributes Reference

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

- **certificate** - The certificate
- **issuing_ca** - The issuing CA
- **ca_chain** - The CA chain

- serial - The serial

» vault_pki_secret_backend_sign

Signs a new certificate based upon the provided CSR and the supplied parameters by the PKI Secret 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_pki_secret_backend_sign" "test" {
  depends_on = [ "vault_pki_secret_backend_role.admin" ]

  backend = "${vault_pki_secret_backend.pki.path}"

  name = "${vault_pki_secret_backend_role.admin.name}"
  csr = <<EOT
-----BEGIN CERTIFICATE REQUEST-----
MIIeQDCCApACAQAwYzELMAkGA1UEBhMCQVUxEzARBgNVBAgMC1NvbWUtU3RhdGUx
ITAfBgNVBAoMGEludGVybmV0IFdpZGdpdHMgUHR5IEExOZDECmBoGA1UEAwTY2Vy
dC50ZXN0Lm15LmRvbWVpYjCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIB
AJupYcQ8UVCWII1Zof1c6YcSSaM9hEaDU78cfKP5RoSeH10BvrWRfT+mzCONVpNP
CW9Iabtvk6hm0ot6ilnndEyVJbc0g7hdDLBX5BM25D+DGZGJRKUz1V+uBrWmXtIt
Vonj7JTDTE7ViHOGDsB7CvqXFGX02a2cDYBchLkL6vQiFPshxvUsLtwxuy/qdYgy
X6ya+AUoZcoQgy1XxNjfH6cPtWSWQGEp1oPR6vL9hU3laTZb3C+VV4jZem+he8/0
V+qV6fLG92WTXm2hmf8nrtUqqJ+C7mW/RJod+TviviBadIX00HXW7k5HVsZood01
te8vMRUNJNiZfa9EMIK5oncbQn0LcM3Wo9VrjpL7jREb/4HCS2gswYGV7hzk9cCS
kVY4rDucckKbApuI3kfzm07GFOF5eiSkYZpY/czNn7VVM3WCu6dpOX4+3rhgrZQw
kY14L930DaLVRUgve/zKVP2D2GHdEOs+MbV7s96UgigT9pXly/yHPj+1sSYqmnad
5b7jSeJusmz0/nrwXVGLsnezR87VzH19Ux9g5s6zh+R+PrZuVxYsLvoUpaasH470
gIcBzSb/6pSGZKAUizmYsHsR1k88dAvsQ+FsUDaNokdi9VndEB4QPmiFmjyLV+OI
1TFoXop4sW11NPz1YCq+IxnYrEaIN3PyhYOGvBJDFY1/AgMBAAGgADANBgkqhkiG
9w0BAQsFAAOCAGEAActuqnqS8Y9UF7e08w7tR3FPzGecWreuvxILr1FEZJxiLPfQL
It7uJvtypCVQvz6UQzKdBY07tMpRaWViB8DrWzXNZjLMrg+QHcpveg8C0Ett4scG
fnvLk6fTDFYrnGvwHTqiHos5i0y3bFLyS1BGwSpdLaykGtvC+VM8mRyw/Y7CPcKN
77kebY/9xduW1g2uxWLR0x90RuQDv9psPoJT+59tRLGSp5Kt0IeD3QtnAZEFE4aN
vt+Pd69eg3BgZ8ZeDgoqAw3yppv0kpAFiE5pw2qPZaM4SRph14d2Lek2zNIMyZqv
do5zh356H0gXtDaSg0P0nRGrN/Ua+LMCRTg6GEPUnx9uQb/z8ZuOhIexDGyykp1
OGqtWlv/Nc8UYuS38v0BeB6bMPeoqQUjkqs8nH1AEFnOKlgydtDC+7SdQx6wS4te
dBKRNDFC41S3jYJgs55jHqonZgkpsI3bamlxpfpW0ukGBcmq91wRe4b0w/4uD/vf
```

```

UwqMW0dCYcU3mdYNjTWy220RW3SGFQxMBwpUEURCSoeqWr6aJeQ7KAYkx1PrB5T8
OTec13lWf+B0PU9UJuGTsmpIuImPDVd0EVDaYr3mT5dDbqTVDbE8ppf2IswABmf0
o3DybUeUmknYj1109rdSf+76nuREICHatxXgN3xCMFuBaN4WLO+ksd6Y1Ys=
-----END CERTIFICATE REQUEST-----
EOT
  common_name = "test.my.domain"
}

```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The PKI secret backend the resource belongs to.
- **name** - (Required) Name of the role to create the certificate against
- **csr** - (Required) The CSR
- **common_name** - (Required) CN of certificate to create
- **alt_names** - (Optional) List of alternative names
- **other_sans** - (Optional) List of other SANs
- **ip_sans** - (Optional) List of alternative IPs
- **uri_sans** - (Optional) List of alternative URIs
- **t1** - (Optional) Time to leave
- **format** - (Optional) The format of data
- **exclude_cn_from_sans** - (Optional) Flag to exclude CN from SANs

» Attributes Reference

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

- **certificate** - The certificate
- **issuing_ca** - The issuing CA
- **ca_chain** - The CA chain
- **serial** - The serial

» 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.

» Import

Policies can be imported using the **name**, e.g.

```
$ terraform import vault_policy.example dev-team
```

» vault__egp__policy

Provides a resource to manage Endpoint Governing Policy (EGP) via Sentinel.

Note this feature is available only with Vault Enterprise.

» Example Usage

```
resource "vault_egp_policy" "allow-all" {
  name = "allow-all"
```

```

paths = ["*"]
enforcement_level = "soft-mandatory"

policy = <<EOT
main = rule {
  true
}
EOT
}

```

» Argument Reference

The following arguments are supported:

- **name** - (Required) The name of the policy
- **paths** - (Required) List of paths to which the policy will be applied to
- **enforcement_level** - (Required) Enforcement level of Sentinel policy.
Can be either **advisory** or **soft-mandatory** or **hard-mandatory**
- **policy** - (Required) String containing a Sentinel policy

» Attributes Reference

No additional attributes are exported by this resource.

» vault_rgp_policy

Provides a resource to manage Role Governing Policy (RGP) via Sentinel.

Note this feature is available only with Vault Enterprise.

» Example Usage

```

resource "vault_rgp_policy" "allow-all" {
  name = "allow-all"
  enforcement_level = "soft-mandatory"

  policy = <<EOT
main = rule {
  true
}
EOT

```

}

» Argument Reference

The following arguments are supported:

- **name** - (Required) The name of the policy
- **enforcement_level** - (Required) Enforcement level of Sentinel policy.
Can be either **advisory** or **soft-mandatory** or **hard-mandatory**
- **policy** - (Required) String containing a Sentinel policy

» Attributes Reference

No additional attributes are exported by this resource.

» vault_token_auth_backend_role

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

» Example Usage

```
resource "vault_token_auth_backend_role" "example" {  
  role_name      = "my-role"  
  allowed_policies = ["dev", "test"]  
  disallowed_policies = ["default"]  
  orphan         = true  
  period         = "86400"  
  renewable      = true  
  explicit_max_ttl = "115200"  
  path_suffix    = "path-suffix"  
}
```

» Argument Reference

The following arguments are supported:

- **role_name** - (Required) The name of the role.
- **allowed_policies** (Optional) List of allowed policies for given role.
- **disallowed_policies** (Optional) List of disallowed policies for given role.

- **orphan** (Optional) If true, tokens created against this policy will be orphan tokens.
- **period** (Optional) The duration in which a token should be renewed. At each renewal, the token's TTL will be set to the value of this parameter.
- **renewable** (Optional) Whether to disable the ability of the token to be renewed past its initial TTL.
- **explicit_max_ttl** (Optional) If set, the token will have an explicit max TTL set upon it.
- **path_suffix** (Optional) Tokens created against this role will have the given suffix as part of their path in addition to the role name.
- **ttl** (Optional) The TTL period of tokens issued using this role, provided as the number of minutes.
- **max_ttl** (Optional) The maximum allowed lifetime of tokens issued using this role.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

Token auth backend roles can be imported with `auth/token/roles/` followed by the `role_name`, e.g.

```
$ terraform import vault_token_auth_backend_role.example auth/token/roles/my-role
```

» vault_ssh_secret_backend_ca

Provides a resource to manage CA information in an SSH secret backend SSH secret backend within Vault.

» Example Usage

```
resource "vault_mount" "example" {
  type = "ssh"
}

resource "vault_ssh_secret_backend_ca" "foo" {
  backend = "${vault_mount.example.path}"
}
```

```
}
```

» Argument Reference

The following arguments are supported:

- **backend** - (Optional) The path where the SSH secret backend is mounted. Defaults to 'ssh'
- **generate_signing_key** - (Optional) Whether Vault should generate the signing key pair internally. Defaults to true
- **public_key** - (Optional) The public key part the SSH CA key pair; required if **generate_signing_key** is false.
- **private_key** - (Optional) The private key part the SSH CA key pair; required if **generate_signing_key** is false.

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

» Attributes Reference

No additional attributes are exposed by this resource.

» vault_ssh_secret_backend_role

Provides a resource to manage roles in an SSH secret backend SSH secret backend within Vault.

» Example Usage

```
resource "vault_mount" "example" {
  type = "ssh"
}

resource "vault_ssh_secret_backend_role" "foo" {
  name                = "my-role"
  backend              = "${vault_mount.example.path}"
  key_type             = "ca"
  allow_user_certificates = true
}
```

```

resource "vault_ssh_secret_backend_role" "bar" {
  name          = "otp-role"
  backend       = "${vault_mount.example.path}"
  key_type      = "otp"
  default_user  = "default"
  allowed_users = "default,baz"
  cidr_list     = "0.0.0.0/0"
}

```

» Argument Reference

The following arguments are supported:

- **name** - (Required) Specifies the name of the role to create.
- **backend** - (Required) The path where the SSH secret backend is mounted.
- **key_type** - (Required) Specifies the type of credentials generated by this role. This can be either `otp`, `dynamic` or `ca`.
- **allow_bare_domains** - (Optional) Specifies if host certificates that are requested are allowed to use the base domains listed in **allowed_domains**.
- **allow_host_certificates** - (Optional) Specifies if certificates are allowed to be signed for use as a 'host'.
- **allow_subdomains** - (Optional) Specifies if host certificates that are requested are allowed to be subdomains of those listed in **allowed_domains**.
- **allow_user_certificates** - (Optional) Specifies if certificates are allowed to be signed for use as a 'user'.
- **allow_user_key_ids** - (Optional) Specifies if users can override the key ID for a signed certificate with the **key_id** field.
- **allowed_critical_options** - (Optional) Specifies a comma-separated list of critical options that certificates can have when signed.
- **allowed_domains** - (Optional) The list of domains for which a client can request a host certificate.
- **cidr_list** - (Optional) The comma-separated string of CIDR blocks for which this role is applicable.
- **allowed_extensions** - (Optional) Specifies a comma-separated list of extensions that certificates can have when signed.
- **default_extensions** - (Optional) Specifies a map of extensions that certificates have when signed.
- **default_critical_options** - (Optional) Specifies a map of critical options that certificates have when signed.

- `allowed_users` - (Optional) Specifies a comma-separated list of usernames that are to be allowed, only if certain usernames are to be allowed.
- `default_user` - (Optional) Specifies the default username for which a credential will be generated.
- `key_id_format` - (Optional) Specifies a custom format for the key id of a signed certificate.
- `max_ttl` - (Optional) Specifies the Time To Live value.
- `ttl` - (Optional) Specifies the maximum Time To Live value.

» Attributes Reference

No additional attributes are exposed by this resource.

» Import

SSH secret backend roles can be imported using the `path`, e.g.

```
$ terraform import vault_ssh_secret_backend_role.foo ssh/roles/my-role
```

» `vault_rabbitmq_secret_backend`

Creates an RabbitMQ Secret Backend for Vault. RabbitMQ secret backends can then issue RabbitMQ credentials, 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_rabbitmq_secret_backend" "rabbitmq" {
  connection_uri = "https://....."
  username      = "user"
  password      = "password"
}
```

» Argument Reference

The following arguments are supported:

- **connection_uri** - (Required) Specifies the RabbitMQ connection URI.
- **username** - (Required) Specifies the RabbitMQ management administrator username.
- **password** - (Required) Specifies the RabbitMQ management administrator password.
- **verify_connection** - (Optional) Specifies whether to verify connection URI, username, and password. Defaults to **true**.

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

- **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.

» Import

RabbitMQ secret backends can be imported using the **path**, e.g.

```
$ terraform import vault_rabbitmq_secret_backend.rabbitmq rabbitmq
```

» vault__rabbitmq__secret__backend__role

Creates a role on an RabbitMQ 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_rabbitmq_secret_backend" "rabbitmq" {
  connection_uri = "https://....."
  username      = "user"
  password      = "password"
}

resource "vault_rabbitmq_secret_backend_role" "role" {
  backend = "${vault_rabbitmq_secret_backend.rabbitmq.path}"
  name    = "deploy"

  tags = "tag1,tag2"
  vhosts = "{\"\\\"/\\\": {\\\"configure\\\":\\\".*\\\", \\\"write\\\":\\\".*\\\", \\\"read\\\": \\\".*\\\"}}\""
}
```

» Argument Reference

The following arguments are supported:

- **backend** - (Required) The path the RabbitMQ 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.
- **tags** - (Optional) Specifies a comma-separated RabbitMQ management tags.
- **vhosts** - (Optional) Specifies a map of virtual hosts to permissions.

» Attributes Reference

No additional attributes are exported by this resource.

» Import

RabbitMQ secret backend roles can be imported using the `path`, e.g.

```
$ terraform import vault_rabbitmq_secret_backend_role.role rabbitmq/roles/deploy
```