» kubernetes_secret

The resource provides mechanisms to inject containers with sensitive information, such as passwords, while keeping containers agnostic of Kubernetes. Secrets can be used to store sensitive information either as individual properties or coarse-grained entries like entire files or JSON blobs. The resource will by default create a secret which is available to any pod in the specified (or default) namespace.

Read more about security properties and risks involved with using Kubernetes secrets: Kubernetes reference

Note: All arguments including the secret data will be stored in the raw state as plain-text. Read more about sensitive data in state.

» Example Usage

```
data "kubernetes_secret" "example" {
  metadata {
    name = "basic-auth"
  }
}
```

» Argument Reference

The following arguments are supported:

- `metadata` - (Required) Standard secret's metadata. For more info see Kubernetes reference

» Nested Blocks

» `metadata`

» Arguments

- `name` - (Required) Name of the secret, must be unique. For more info see Kubernetes reference
- `namespace` - (Optional) Namespace defines the space within which name of the secret must be unique.

» Attributes

- `generation` - A sequence number representing a specific generation of the desired state.

- `resource_version` - An opaque value that represents the internal version of this secret that can be used by clients to determine when secret has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this secret.
- `uid` - The unique in time and space value for this secret. For more info see Kubernetes reference

» Attribute Reference

- `data` - A map of the secret data.
- `type` - The secret type. Defaults to `Opaque`. For more info see Kubernetes reference

» kubernetes_service

A Service is an abstraction which defines a logical set of pods and a policy by which to access them - sometimes called a micro-service. This data source allows you to pull data about such service.

» Example Usage

```
data "kubernetes_service" "example" {
  metadata {
    name = "terraform-example"
  }
}

resource "aws_route53_record" "example" {
  zone_id = "${data.aws_route53_zone.k8.zone_id}"
  name    = "example"
  type    = "CNAME"
  ttl     = "300"
  records = ["${data.kubernetes_service.example.load_balancer_ingress.0.hostname}"]
}
```

» Argument Reference

The following arguments are supported:

- `metadata` - (Required) Standard service's metadata. For more info see Kubernetes reference

» Attributes

- `spec` - Spec defines the behavior of a service. Kubernetes reference
- `load_balancer_ingress` - A list containing ingress points for the load-balancer (only valid if `type = "LoadBalancer"`)

» Nested Blocks

» `metadata`

» Arguments

- `name` - (Optional) Name of the service, must be unique. Cannot be updated. For more info see Kubernetes reference
- `namespace` - (Optional) Namespace defines the space within which name of the service must be unique.

» Attributes

- `annotations` - (Optional) An unstructured key value map stored with the service that may be used to store arbitrary metadata. For more info see Kubernetes reference
- `labels` - (Optional) Map of string keys and values that can be used to organize and categorize (scope and select) the service. May match selectors of replication controllers and services. For more info see Kubernetes reference
- `generation` - A sequence number representing a specific generation of the desired state.
- `resource_version` - An opaque value that represents the internal version of this service that can be used by clients to determine when service has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this service.
- `uid` - The unique in time and space value for this service. For more info see Kubernetes reference

» `port`

» Attributes

- `name` - The name of this port within the service. All ports within the service must have unique names. Optional if only one ServicePort is defined on this service.
- `node_port` - The port on each node on which this service is exposed when `type` is `NodePort` or `LoadBalancer`. Usually assigned by the system. If specified, it will be allocated to the service if unused or else creation of

the service will fail. Default is to auto-allocate a port if the `type` of this service requires one. For more info see Kubernetes reference
- `port` - The port that will be exposed by this service.
- `protocol` - The IP protocol for this port. Supports `TCP` and `UDP`. Default is `TCP`.
- `target_port` - Number or name of the port to access on the pods targeted by the service. Number must be in the range 1 to 65535. This field is ignored for services with `cluster_ip = "None"`. For more info see Kubernetes reference

» `spec`

» Attributes

- `cluster_ip` - The IP address of the service. It is usually assigned randomly by the master. If an address is specified manually and is not in use by others, it will be allocated to the service; otherwise, creation of the service will fail. `None` can be specified for headless services when proxying is not required. Ignored if type is `ExternalName`. For more info see Kubernetes reference
- `external_ips` - A list of IP addresses for which nodes in the cluster will also accept traffic for this service. These IPs are not managed by Kubernetes. The user is responsible for ensuring that traffic arrives at a node with this IP. A common example is external load-balancers that are not part of the Kubernetes system.
- `external_name` - The external reference that kubedns or equivalent will return as a CNAME record for this service. No proxying will be involved. Must be a valid DNS name and requires `type` to be `ExternalName`.
- `load_balancer_ip` - Only applies to `type = LoadBalancer`. LoadBalancer will get created with the IP specified in this field. This feature depends on whether the underlying cloud-provider supports specifying this field when a load balancer is created. This field will be ignored if the cloud-provider does not support the feature.
- `load_balancer_source_ranges` - If specified and supported by the platform, this will restrict traffic through the cloud-provider load-balancer will be restricted to the specified client IPs. This field will be ignored if the cloud-provider does not support the feature. For more info see Kubernetes reference
- `port` - The list of ports that are exposed by this service. For more info see Kubernetes reference
- `selector` - Route service traffic to pods with label keys and values matching this selector. Only applies to types `ClusterIP`, `NodePort`, and `LoadBalancer`. For more info see Kubernetes reference
- `session_affinity` - Used to maintain session affinity. Supports `ClientIP` and `None`. Defaults to `None`. For more info see Kubernetes reference

- `type` - Determines how the service is exposed. Defaults to `ClusterIP`. Valid options are `ExternalName`, `ClusterIP`, `NodePort`, and `LoadBalancer`. `ExternalName` maps to the specified `external_name`. For more info see Kubernetes reference

» `load_balancer_ingress`

» Attributes

- `hostname` - Hostname which is set for load-balancer ingress points that are DNS based (typically AWS load-balancers)
- `ip` - IP which is set for load-balancer ingress points that are IP based (typically GCE or OpenStack load-balancers)

» kubernetes_storage_class

Storage class is the foundation of dynamic provisioning, allowing cluster administrators to define abstractions for the underlying storage platform.

Read more at http://blog.kubernetes.io/2017/03/dynamic-provisioning-and-storage-classes-kubernetes.html

» Example Usage

```
data "kubernetes_storage_class" "example" {
  metadata {
    name = "terraform-example"
  }
}
```

» Argument Reference

The following arguments are supported:

- `metadata` - (Required) Standard storage class's metadata. For more info see Kubernetes reference

» Nested Blocks

» `metadata`

» Arguments

- `name` - (Required) Name of the storage class, must be unique. For more info see Kubernetes reference

» Attributes

- `generation` - A sequence number representing a specific generation of the desired state.
- `resource_version` - An opaque value that represents the internal version of this storage class that can be used by clients to determine when storage class has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this storage class.
- `uid` - The unique in time and space value for this storage class. For more info see Kubernetes reference

» Argument Reference

The following attributes are exported:

- `parameters` - The parameters for the provisioner that creates volume of this storage class. Read more about available parameters.
- `storage_provisioner` - Indicates the type of the provisioner this storage class represents
- `reclaim_policy` - Indicates the reclaim policy used.
- `volume_binding_mode` - Indicates when volume binding and dynamic provisioning should occur.

» kubernetes_config_map

The resource provides mechanisms to inject containers with configuration data while keeping containers agnostic of Kubernetes. Config Map can be used to store fine-grained information like individual properties or coarse-grained information like entire config files or JSON blobs.

» Example Usage

```
resource "kubernetes_config_map" "example" {
  metadata {
    name = "my-config"
  }

  data {
    api_host = "myhost:443"
    db_host  = "dbhost:5432"
```

```
  }
}
```

» Argument Reference

The following arguments are supported:

- `data` - (Optional) A map of the configuration data.
- `metadata` - (Required) Standard config map's metadata. For more info see Kubernetes reference

» Nested Blocks

» `metadata`

» Arguments

- `annotations` - (Optional) An unstructured key value map stored with the config map that may be used to store arbitrary metadata. For more info see Kubernetes reference
- `generate_name` - (Optional) Prefix, used by the server, to generate a unique name ONLY IF the `name` field has not been provided. This value will also be combined with a unique suffix. For more info see Kubernetes reference
- `labels` - (Optional) Map of string keys and values that can be used to organize and categorize (scope and select) the config map. May match selectors of replication controllers and services. For more info see Kubernetes reference
- `name` - (Optional) Name of the config map, must be unique. Cannot be updated. For more info see Kubernetes reference
- `namespace` - (Optional) Namespace defines the space within which name of the config map must be unique.

» Attributes

- `generation` - A sequence number representing a specific generation of the desired state.
- `resource_version` - An opaque value that represents the internal version of this config map that can be used by clients to determine when config map has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this config map.
- `uid` - The unique in time and space value for this config map. For more info see Kubernetes reference

» Import

Config Map can be imported using its namespace and name, e.g.

```
$ terraform import kubernetes_config_map.example default/my-config
```

» kubernetes_cluster_role

A ClusterRole creates a role at the cluster level and in all namespaces.

» Example Usage

```
resource "kubernetes_cluster_role" "example" {
    metadata {
        name = "terraform-example"
    }

    rule {
        api_groups = [""]
        resources  = ["namespaces", "pods"]
        verbs      = ["get", "list", "watch"]
    }
}
```

» Argument Reference

The following arguments are supported:

- `metadata` - (Required) Standard kubernetes metadata. For more info see Kubernetes reference
- `rule` - (Required) The PolicyRoles for this ClusterRole. For more info see Kubernetes reference

» Nested Blocks

» `metadata`

» Arguments

- `annotations` - (Optional) An unstructured key value map stored with the cluster role binding that may be used to store arbitrary metadata. For more info see Kubernetes reference

- `generate_name` - (Optional) Prefix, used by the server, to generate a unique name ONLY IF the `name` field has not been provided. This value will also be combined with a unique suffix. For more info see Kubernetes reference
- `labels` - (Optional) Map of string keys and values that can be used to organize and categorize (scope and select) the cluster role binding. For more info see Kubernetes reference
- `name` - (Optional) Name of the cluster role binding, must be unique. Cannot be updated. For more info see Kubernetes reference

» Attributes

- `generation` - A sequence number representing a specific generation of the desired state.
- `resource_version` - An opaque value that represents the internal version of this object that can be used by clients to determine when the object has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this cluster role binding.
- `uid` - The unique in time and space value for this cluster role binding. For more info see Kubernetes reference

» `rule`

» Arguments

- `api_groups` - (Optional) APIGroups is the name of the APIGroup that contains the resources. If multiple API groups are specified, any action requested against one of the enumerated resources in any API group will be allowed.
- `non_resource_urls` - (Optional) NonResourceURLs is a set of partial urls that a user should have access to. *s are allowed, but only as the full, final step in the path Since non-resource URLs are not namespaced, this field is only applicable for ClusterRoles referenced from a ClusterRoleBinding. Rules can either apply to API resources (such as "pods" or "secrets") or non-resource URL paths (such as "/api"), but not both.
- `resource_names` - (Optional) ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.
- `resources` - (Optional) Resources is a list of resources this rule applies to. ResourceAll represents all resources.
- `verbs` - (Required) Verbs is a list of Verbs that apply to ALL the ResourceKinds and AttributeRestrictions contained in this rule. VerbAll represents all kinds.

» Import

ClusterRole can be imported using the name, e.g.

```
$ terraform import kubernetes_cluster_role.example terraform-name
```

» kubernetes_cluster_role_binding

A ClusterRoleBinding may be used to grant permission at the cluster level and in all namespaces

» Example Usage

```
resource "kubernetes_cluster_role_binding" "example" {
    metadata {
        name = "terraform-example"
    }
    role_ref {
        api_group = "rbac.authorization.k8s.io"
        kind = "ClusterRole"
        name = "cluster-admin"
    }
    subject {
        kind = "User"
        name = "admin"
        api_group = "rbac.authorization.k8s.io"
    }
    subject {
        kind = "ServiceAccount"
        name = "default"
        namespace = "kube-system"
    }
    subject {
        kind = "Group"
        name = "system:masters"
        api_group = "rbac.authorization.k8s.io"
    }
}
```

» Argument Reference

The following arguments are supported:

- `metadata` - (Required) Standard kubernetes metadata. For more info see Kubernetes reference
- `role_ref` - (Required) The ClusterRole to bind Subjects to. For more info see Kubernetes reference
- `subject` - (Required) The Users, Groups, or ServiceAccounts to grant permissions to. For more info see Kubernetes reference

» Nested Blocks

» `metadata`

» Arguments

- `annotations` - (Optional) An unstructured key value map stored with the cluster role binding that may be used to store arbitrary metadata. For more info see Kubernetes reference
- `generate_name` - (Optional) Prefix, used by the server, to generate a unique name ONLY IF the `name` field has not been provided. This value will also be combined with a unique suffix. For more info see Kubernetes reference
- `labels` - (Optional) Map of string keys and values that can be used to organize and categorize (scope and select) the cluster role binding. For more info see Kubernetes reference
- `name` - (Optional) Name of the cluster role binding, must be unique. Cannot be updated. For more info see Kubernetes reference

» Attributes

- `generation` - A sequence number representing a specific generation of the desired state.
- `resource_version` - An opaque value that represents the internal version of this object that can be used by clients to determine when the object has changed. For more info see Kubernetes reference
- `self_link` - A URL representing this cluster role binding.
- `uid` - The unique in time and space value for this cluster role binding. For more info see Kubernetes reference

» `role_ref`

» Arguments

- `name` - (Required) The name of this ClusterRole to bind Subjects to.
- `kind` - (Required) The type of binding to use. This value must be and defaults to `ClusterRole`

- `api_group` - (Optional) The API group to drive authorization decisions. This value must be and defaults to `rbac.authorization.k8s.io`

» `subject`

» Arguments

- `name` - (Required) The name of this ClusterRole to bind Subjects to.
- `namespace` - (Optional) Namespace defines the namespace of the ServiceAccount to bind to. This value only applies to kind `ServiceAccount`
- `kind` - (Required) The type of binding to use. This value must be `ServiceAccount`, `User` or `Group`
- `api_group` - (Optional) The API group to drive authorization decisions. This value only applies to kind `User` and `Group`. It must be `rbac.authorization.k8s.io`

» Import

ClusterRoleBinding can be imported using the name, e.g.

```
$ terraform import kubernetes_cluster_role_binding.example terraform-name
```