» Data Source: heroku_app

Use this data source to get information about a Heroku App.

» Example Usage

```
# Create a new Heroku app
data "heroku_app" "default"
  name   = "my-cool-app"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the application. In Heroku, this is also the unique ID, so it must be unique and have a minimum of 3 characters.

» Attributes Reference

The following attributes are exported:

- `name` - (Required) The name of the application. In Heroku, this is also the unique .
- `stack` - (Optional) The application stack is what platform to run the application in.
- `buildpacks` - (Optional) A list of buildpacks that this app uses.
- `space` - (Optional) The private space in which the app runs. Not present if this is a common runtime app.
- `region` - (Required) The region in which the app is deployed.
- `git_url` - (Required) The Git URL for the application. This is used for deploying new versions of the app.
- `web_url` - (Required) The web (HTTP) URL that the application can be accessed at by default.
- `heroku_hostname` - (Required) A hostname for the Heroku application, suitable for pointing DNS records.
- `config_vars` - (Optional) A map of all of the configuration variables for the app.
- `acm` - (Required) True if Heroku ACM is enabled for this app, false otherwise.
- `organization` - (Optional) The organization that owns this app, if the app is owned by an organization. The fields for this block are documented below.

The `organization` block supports:

- `name` (string) - The name of the organization.
- `locked` (boolean)
- `personal` (boolean)

» Data Source: heroku_space

Use this data source to get information about a Heroku Private Space.

» Example Usage

```
# Look up a Heroku Private Space
data "heroku_space" "default" {
  name   = "my-secret-space"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the Heroku Private Space.

» Attributes Reference

The following attributes are exported:

- `name` - The name of the Heroku Private Space. In Heroku, this is also the unique .
- `id` - The unique ID of the Heroku Private Space.
- `region` - The region in which the Heroku Private Space is deployed.
- `state` - The state of the Heroku Private Space. Either `allocating` or `allocated`.
- `shield` - Whether or not the space has Shield turned on. One of `on` or `off`.
- `organization` - The organization that owns this space, if the space is owned by an organization. The fields for this block are documented below.
- `outbound_ips` - The space's stable outbound NAT IPs.

The `organization` block supports:

- `name` (string) - The name of the organization.

» Data Source: heroku_space_peering_info

Use this data source to get peering information about a Heroku Private Space.

» Example Usage

```
# Look up a Heroku Private Space's peering info.
data "heroku_space_peering_info" "default" {
  name   = "my-secret-space"
}

# Initiate a VPC peering connection request.
resource "aws_vpc_peering_connection" "foo" {
  peer_owner_id = "${data.heroku_space_peering_info.default.aws_account_id}"
  peer_vpc_id   = "${data.heroku_space_peering_info.default.vpc_id}"
  vpc_id        = "${aws_vpc.foo.id}"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the Heroku Private Space.

» Attributes Reference

The following attributes are exported:

- `aws_account_id` - The AWS account ID that the Heroku Private Space runs in.
- `aws_region` - The AWS region that the Heroku Private Space runs in.
- `vpc_id` - The VPC ID of the Heroku Private Space.
- `vpc_cidr` - The CIDR block of the VPC ID.
- `dyno_cidr_blocks` - The CIDR blocks that the Dynos run on.
- `unavailable_cidr_blocks` - A list of unavailable CIDR blocks.

» heroku_addon

Provides a Heroku Add-On resource. These can be attach services to a Heroku app.

» Example Usage

```
# Create a new Heroku app
resource "heroku_app" "default" {
  name = "test-app"
}

# Create a database, and configure the app to use it
resource "heroku_addon" "database" {
  app  = "${heroku_app.default.name}"
  plan = "heroku-postgresql:hobby-basic"
}

# Add a web-hook addon for the app
resource "heroku_addon" "webhook" {
  app  = "${heroku_app.default.name}"
  plan = "deployhooks:http"

  config {
    url = "http://google.com"
  }
}
```

» Argument Reference

The following arguments are supported:

- `app` - (Required) The Heroku app to add to.
- `plan` - (Required) The addon to add.
- `config` - (Optional) Optional plan configuration.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the add-on
- `name` - The add-on name
- `plan` - The plan name
- `provider_id` - The ID of the plan provider
- `config_vars` - The Configuration variables of the add-on

» Import

Addons can be imported using the Addon `id`, e.g.

```
$ terraform import heroku_addon.foobar 12345678
```

» heroku_addon_attachment

Attaches a Heroku Addon Resource to an additional Heroku App.

» Example Usage

```
resource "heroku_addon_attachment" "database" {
  app_id  = "${heroku_app.default.id}"
  addon_id = "${heroku_addon.database.id}"
}
```

» Argument Reference

The following arguments are supported:

- `app_id` - (Required) The ID of the Heroku App to attach to.
- `addon_id` - (Required) The ID of the existing Heroku Addon to attach.
- `name` - (Optional) A friendly name for the Heroku Addon Attachment.

» Attributes Reference

The following attributes are exported:

- `id` - The unique ID of the add-on attachment

» Import

Addons can be imported using the unique Addon Attachment `id`, e.g.

```
$ terraform import heroku_addon_attachment.foobar 01234567-89ab-cdef-0123-456789abcdef
```

» heroku_app

Provides a Heroku App resource. This can be used to create and manage applications on Heroku.

» Example Usage

```
# Create a new Heroku app
resource "heroku_app" "default" {
  name   = "my-cool-app"
  region = "us"

  config_vars {
    FOOBAR = "baz"
  }

  buildpacks = [
    "heroku/go"
  ]
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the application. In Heroku, this is also the unique ID, so it must be unique and have a minimum of 3 characters.
- `region` - (Required) The region that the app should be deployed in.
- `stack` - (Optional) The application stack is what platform to run the application in.
- `buildpacks` - (Optional) Buildpack names or URLs for the application. Buildpacks configured externally won't be altered if this is not present.
- `config_vars` - (Optional) Configuration variables for the application. The config variables in this map are not the final set of configuration variables, but rather variables you want present. That is, other configuration variables set externally won't be removed by Terraform if they aren't present in this list.
- `space` - (Optional) The name of a private space to create the app in.
- `organization` - (Optional) A block that can be specified once to define organization settings for this app. The fields for this block are documented below.
- `acm` - (Optional) The flag representing Automated Certificate Management for the app.

The `organization` block supports:

- `name` (string) - The name of the organization.
- `locked` (boolean)
- `personal` (boolean)

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the app. This is also the name of the application.
- `name` - The name of the application. In Heroku, this is also the unique ID.
- `stack` - The application stack is what platform to run the application in.
- `space` - The private space the app should run in.
- `region` - The region that the app should be deployed in.
- `git_url` - The Git URL for the application. This is used for deploying new versions of the app.
- `web_url` - The web (HTTP) URL that the application can be accessed at by default.
- `heroku_hostname` - A hostname for the Heroku application, suitable for pointing DNS records.
- `all_config_vars` - A map of all of the configuration variables that exist for the app, containing both those set by Terraform and those set externally.

» Import

Apps can be imported using the App `id`, e.g.

```
$ terraform import heroku_app.foobar MyApp
```

» heroku_app_feature

Provides a Heroku App Feature resource. This can be used to create and manage App Features on Heroku.

» Example Usage

```
resource "heroku_app_feature" "log_runtime_metrics" {
  app = "test-app"
  name = "log-runtime-metrics"
}
```

» Argument Reference

The following arguments are supported:

- `app` - (Required) The Heroku app to link to.
- `name` - (Required) The name of the App Feature to manage.

- `enabled` - (Optional) Whether to enable or disable the App Feature. The default value is true.

» heroku_app_release

Provides a Heroku App Release resource.

An app release represents a combination of code, config vars and add-ons for an app on Heroku.

NOTE: This resource requires the slug be uploaded to Heroku prior to running terraform.

» Example Usage

```
resource "heroku_app" "foobar" {
    name = "foobar"
    region = "us"
}

# Upload your slug

resource "heroku_app_release" "foobar-release" {
    app = "${heroku_app.foobar.name}"
    slug_id = "01234567-89ab-cdef-0123-456789abcdef"
}
```

» Argument Reference

- `app` - (Required) The name of the application
- `slug_id` - unique identifier of slug
- `description` - description of changes in this release

» Attributes Reference

The following attributes are exported: * `id` - The ID of the app release

» Import

Existing app releases can be imported using the combination of the application name, a colon, and the formation's type.

For example: `$ terraform import heroku_app_release.foobar-release foobar`

» heroku_cert

Provides a Heroku SSL certificate resource. It allows to set a given certificate for a Heroku app.

» Example Usage

```
# Create a new Heroku app
resource "heroku_app" "default" {
  name = "test-app"
}

# Add-on SSL to application
resource "heroku_addon" "ssl" {
  app  = "${heroku_app.default.name}"
  plan = "ssl"
}

# Establish certificate for a given application
resource "heroku_cert" "ssl_certificate" {
  app               = "${heroku_app.default.name}"
  certificate_chain = "${file("server.crt")}"
  private_key       = "${file("server.key")}"
  depends_on        = "heroku_addon.ssl"
}
```

» Argument Reference

The following arguments are supported:

- `app` - (Required) The Heroku app to add to.
- `certificate_chain` - (Required) The certificate chain to add
- `private_key` - (Required) The private key for a given certificate chain

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the add-on

- `cname` - The CNAME for the SSL endpoint
- `name` - The name of the SSL certificate

» Importing

When importing a Heroku cert resource, the ID must be built using the app name colon the unique ID from the Heroku API. For an app named `production-api` with a certificate ID of `b85d9224-310b-409b-891e-c903f5a40568`, you would import it as: `$ terraform import heroku_cert.production_api production-api:b85d9224-310b-409b-891e-c903f5a40568`.

» heroku_domain

Provides a Heroku App resource. This can be used to create and manage applications on Heroku.

» Example Usage

```
# Create a new Heroku app
resource "heroku_app" "default" {
  name = "test-app"
}

# Associate a custom domain
resource "heroku_domain" "default" {
  app      = "${heroku_app.default.name}"
  hostname = "terraform.example.com"
}
```

» Argument Reference

The following arguments are supported:

- `hostname` - (Required) The hostname to serve requests from.
- `app` - (Required) The Heroku app to link to.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the of the domain record.
- `hostname` - The hostname traffic will be served as.

- `cname` - The CNAME traffic should route to.

» Importing

When importing a Heroku domain resource, the ID must be built using the app name colon the unique ID from the Heroku API. For an app named `production-api` with a domain ID of `b85d9224-310b-409b-891e-c903f5a40568`, you would import it as: `$ terraform import heroku_domain.production_api production-api:b85d9224-310b-409b-891e-c903f5a40568`.

» heroku_drain

Provides a Heroku Drain resource. This can be used to create and manage Log Drains on Heroku.

» Example Usage

```
resource "heroku_drain" "default" {
  app = "test-app"
  url = "syslog://terraform.example.com:1234"
}
```

» Argument Reference

The following arguments are supported:

- `url` - (Required) The URL for Heroku to drain your logs to.
- `app` - (Required) The Heroku app to link to.

» Attributes Reference

The following attributes are exported:

- `token` - The unique token for your created drain.

» Importing

When importing a Heroku drain resource, the ID must be built using the app name colon the unique ID from the Heroku API. For an app named `production-api` with a drain ID of `b85d9224-310b-409b-891e-c903f5a40568`,

you would import it as: `$ terraform import heroku_drain.production_api production-api:b85d9224-310b-409b-891e-c903f5a40568`

» heroku_formation

Provides a Heroku Formation resource.

A formation represents the formation of processes that should be set for an application.

NOTE: - The application must have a dyno in order to update its formation. - If the heroku formation resource is removed and deleted, this will be a no-op action in Heroku. The Heroku Platform does not have a `DELETE` endpoint for `formation`. - This resource works well with the `heroku_app_release` resource, which allows you to deploy a slug/release to an application before the formation can be updated.

» Example Usage

```
# Creates a new application called foobar
resource "heroku_app" "foobar" {
    name = "foobar"
    region = "us"
}

# Creates a new release for application foobar using a slug id
resource "heroku_app_release" "foobar-release" {
    app = "${heroku_app.foobar.name}"
    slug_id = "01234567-89ab-cdef-0123-456789abcdef"
}

# Update the web formation for the foobar application's web
resource "heroku_formation" "foobar-web" {
    app = "${heroku_app.foobar.name}"
    type = "web"
    quantity = 2
    size = "standard-2x"

    # Tells Terraform that this formation must be created/updated only after the app release
    depends_on = ["heroku_app_release.foobar-release"]
}
```

» Argument Reference

- `app` - (Required) The name of the application
- `type` - type of process such as "web"
- `quantity` - number of processes to maintain
- `size` - dyno size (Example: "standard-1X"). Capitalization does not matter.

» Attributes Reference

The following attributes are exported: * `id` - The ID of the formation

» Import

Existing formations can be imported using the combination of the application name, a colon, and the formation's type.

For example: `$ terraform import heroku_formation.foobar-web foobar:web`

» heroku_pipeline

Provides a Heroku Pipeline resource.

A pipeline is a group of Heroku apps that share the same codebase. Once a pipeline is created, and apps are added to different stages using `heroku_pipeline_coupling`, you can promote app slugs to the next stage.

» Example Usage

```
# Create Heroku apps for staging and production
resource "heroku_app" "staging" {
  name = "test-app-staging"
}

resource "heroku_app" "production" {
  name = "test-app-production"
}

# Create a Heroku pipeline
resource "heroku_pipeline" "test-app" {
  name = "test-app"
```

```
}

# Couple apps to different pipeline stages
resource "heroku_pipeline_coupling" "staging" {
  app      = "${heroku_app.staging.name}"
  pipeline = "${heroku_pipeline.test-app.id}"
  stage    = "staging"
}

resource "heroku_pipeline_coupling" "production" {
  app      = "${heroku_app.production.name}"
  pipeline = "${heroku_pipeline.test-app.id}"
  stage    = "production"
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the pipeline.

» Attributes Reference

The following attributes are exported:

- `id` - The UUID of the pipeline.
- `name` - The name of the pipeline.

» Import

Pipelines can be imported using the Pipeline `id`, e.g.

```
$ terraform import heroku_pipeline.foobar 12345678
```

» heroku_pipeline_coupling

Provides a Heroku Pipeline Coupling resource.

A pipeline is a group of Heroku apps that share the same codebase. Once a pipeline is created using `heroku_pipeline`, and apps are added to different stages using `heroku_pipeline_coupling`, you can promote app slugs to the downstream stages.

» Example Usage

```
# Create Heroku apps for staging and production
resource "heroku_app" "staging" {
  name = "test-app-staging"
}

resource "heroku_app" "production" {
  name = "test-app-production"
}

# Create a Heroku pipeline
resource "heroku_pipeline" "test-app" {
  name = "test-app"
}

# Couple apps to different pipeline stages
resource "heroku_pipeline_coupling" "staging" {
  app      = "${heroku_app.staging.name}"
  pipeline = "${heroku_pipeline.test-app.id}"
  stage    = "staging"
}

resource "heroku_pipeline_coupling" "production" {
  app      = "${heroku_app.production.name}"
  pipeline = "${heroku_pipeline.test-app.id}"
  stage    = "production"
}
```

» Argument Reference

The following arguments are supported:

- `app` - (Required) The name of the app for this coupling.
- `pipeline` - (Required) The ID of the pipeline to add this app to.
- `stage` - (Required) The stage to couple this app to. Must be one of `review`, `development`, `staging`, or `production`.

» Attributes Reference

The following attributes are exported:

- `id` - The UUID of this pipeline coupling.
- `app` - The name of the application.
- `app_id` - The ID of the application.

- `pipeline` - The UUID of the pipeline.
- `stage` - The stage for this coupling.

» Import

Pipeline couplings can be imported using the Pipeline coupling `id`, e.g.

```
$ terraform import heroku_pipeline_coupling.foobar 12345678
```

» heroku_space

Provides a Heroku Space resource for running apps in isolated, highly available, secure app execution environments.

» Example Usage

```
// Create a new Heroku space
resource "heroku_space" "default" {
  name = "test-space"
  organization = "my-company"
  region = "virginia"
}

// Create a new Heroku app in test-space
resource "heroku_app" "default" {
  name = "test-app"
  space = "${heroku_space.default.name}"
  organization = {
    name = "my-company"
  }
}
```

» Argument Reference

The following arguments are supported:

- `name` - (Required) The name of the space.
- `organization` - (Required) The name of the organization which will own the space.
- `region` - (Optional) The region that the space should be created in.
- `shield` - (Optional) Whether or not the private space should be shielded.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the space.
- `name` - The space's name.
- `organization` - The space's organization.
- `region` - The space's region.
- `outbound_ips` - The space's stable outbound NAT IPs.

» Import

Spaces can be imported using the space `id`, e.g.

```
$ terraform import heroku_space.foobar MySpace
```

» heroku_space_peering_connection_accepter

Provides a resource for accepting VPC peering requests to Heroku Private Spaces.

» Example Usage

```
# Fetch the peering information for the Heroku Private Space.
data "heroku_space_peering_info" "peer_space" {
  name = "my-fancy-space"
}

# Initiate the request.
resource "aws_vpc_peering_connection" "request" {
  peer_owner_id = "${data.heroku_space_peering_info.peer_space.aws_account_id}"
  peer_vpc_id   = "${data.heroku_space_peering_info.peer_space.vpc_id}"
  vpc_id        = "${aws_vpc.main.id}"
}

# Accept the request.
resource "heroku_space_peering_connection_accepter" "accept" {
  space                     = "${heroku_space.peer_space.name}"
  vpc_peering_connection_id = "${aws_vpc_peering_connection.request.id}"
}
```

» Argument Reference

The following arguments are supported:

- `space` - (Required) The name of the space.
- `vpc_peering_connection_id` - (Required) The peering connection request ID.

» Attributes Reference

The following attributes are exported:

- `status` - The status of the peering connection request.
- `type` - The type of the peering connection.

» heroku_team_collaborator

Provides a Heroku Team Collaborator resource.

A team collaborator represents an account that has been given access to a team app on Heroku.

NOTE: Please only use this resource if you have team/organization apps

» Example Usage

```
# Create a new team collaborator for the foobar application that has view, operate, manage p
resource "heroku_team_collaborator" "foobar-collaborator" {
    app = "${heroku_app.foobar.name}"
    email = "collaborator@foobar.com"
    permissions = ["view", "operate", "manage"]
}
```

» Argument Reference

- `app` - (Required) The name of the team app that the team collaborator will be added to.
- `email` - (Required) Email address of the team collaborator
- `permissions` - (Required) List of permissions that will be granted to the team collaborator. The order in which individual permissions are set here does not matter. Please visit this link for more information on available permissions.

» Attributes Reference

The following attributes are exported:

- `id` - The ID of the team collaborator

» Import

Team Collaborators can be imported using the combination of the team application name, a colon, and the collaborator's email address

For example:

```
$ terraform import heroku_team_collaborator.foobar-collaborator foobar_app:collaborator@foob
```