» random_id

The resource `random_id` generates random numbers that are intended to be used as unique identifiers for other resources.

This resource *does* use a cryptographic random number generator in order to minimize the chance of collisions, making the results of this resource when a 16-byte identifier is requested of equivalent uniqueness to a type-4 UUID.

This resource can be used in conjunction with resources that have the `create_before_destroy` lifecycle flag set to avoid conflicts with unique names during the brief period where both the old and new resources exist concurrently.

» Example Usage

The following example shows how to generate a unique name for an AWS EC2 instance that changes each time a new AMI id is selected.

```
resource "random_id" "server" {
  keepers = {
    # Generate a new id each time we switch to a new AMI id
    ami_id = "${var.ami_id}"
  }

  byte_length = 8
}

resource "aws_instance" "server" {
  tags = {
    Name = "web-server ${random_id.server.hex}"
  }

  # Read the AMI id "through" the random_id resource to ensure that
  # both will change together.
  ami = "${random_id.server.keepers.ami_id}"

  # ... (other aws_instance arguments) ...
}
```

» Argument Reference

The following arguments are supported:

- `byte_length` - (Required) The number of random bytes to produce. The minimum value is 1, which produces eight bits of randomness.

- `keepers` - (Optional) Arbitrary map of values that, when changed, will trigger a new id to be generated. See the main provider documentation for more information.
- `prefix` - (Optional) Arbitrary string to prefix the output value with. This string is supplied as-is, meaning it is not guaranteed to be URL-safe or base64 encoded.

» Attributes Reference

The following attributes are exported:

- `b64_url` - The generated id presented in base64, using the URL-friendly character set: case-sensitive letters, digits and the characters `_` and `-`.
- `b64_std` - The generated id presented in base64 without additional transformations.
- `hex` - The generated id presented in padded hexadecimal digits. This result will always be twice as long as the requested byte length.
- `dec` - The generated id presented in non-padded decimal digits.

» Import

Random Ids can be imported using the `b64_url` with an optional `prefix`. This can be used to replace a config value with a value interpolated from the random provider without experiencing diffs.

Example with no prefix: `$ terraform import random_id.server p-9hUg`

Example with prefix (prefix is separated by a `,`): `$ terraform import random_id.server my-prefix-,p-9hUg`

» random_pet

The resource `random_pet` generates random pet names that are intended to be used as unique identifiers for other resources.

This resource can be used in conjunction with resources that have the `create_before_destroy` lifecycle flag set, to avoid conflicts with unique names during the brief period where both the old and new resources exist concurrently.

» Example Usage

The following example shows how to generate a unique pet name for an AWS EC2 instance that changes each time a new AMI id is selected.

```
resource "random_pet" "server" {
  keepers = {
    # Generate a new pet name each time we switch to a new AMI id
    ami_id = "${var.ami_id}"
  }
}

resource "aws_instance" "server" {
  tags = {
    Name = "web-server-${random_pet.server.id}"
  }

  # Read the AMI id "through" the random_pet resource to ensure that
  # both will change together.
  ami = "${random_pet.server.keepers.ami_id}"

  # ... (other aws_instance arguments) ...
}
```

The result of the above will set the Name of the AWS Instance to `web-server-simple-snake`.

» Argument Reference

The following arguments are supported:

- `keepers` - (Optional) Arbitrary map of values that, when changed, will trigger a new id to be generated. See the main provider documentation for more information.
- `length` - (Optional) The length (in words) of the pet name.
- `prefix` - (Optional) A string to prefix the name with.
- `separator` - (Optional) The character to separate words in the pet name.

» Attribute Reference

The following attributes are supported:

- `id` - (string) The random pet name

» random_shuffle

The resource `random_shuffle` generates a random permutation of a list of strings given as an argument.

» Example Usage

```
resource "random_shuffle" "az" {
  input = ["us-west-1a", "us-west-1c", "us-west-1d", "us-west-1e"]
  result_count = 2
}

resource "aws_elb" "example" {
  # Place the ELB in any two of the given availability zones, selected
  # at random.
  availability_zones = ["${random_shuffle.az.result}"]

  # ... and other aws_elb arguments ...
}
```

» Argument Reference

The following arguments are supported:

- `input` - (Required) The list of strings to shuffle.
- `result_count` - (Optional) The number of results to return. Defaults to the number of items in the `input` list. If fewer items are requested, some elements will be excluded from the result. If more items are requested, items will be repeated in the result but not more frequently than the number of items in the input list.
- `keepers` - (Optional) Arbitrary map of values that, when changed, will trigger a new id to be generated. See the main provider documentation for more information.
- `seed` - (Optional) Arbitrary string with which to seed the random number generator, in order to produce less-volatile permutations of the list. **Important:** Even with an identical seed, it is not guaranteed that the same permutation will be produced across different versions of Terraform. This argument causes the result to be *less volatile*, but not fixed for all time.

» Attributes Reference

The following attributes are exported:

- `result` - Random permutation of the list of strings given in `input`.

» random_string

The resource `random_string` generates a random permutation of alphanumeric characters and optionally special characters.

This resource *does* use a cryptographic random number generator.

» Example Usage

```
resource "random_string" "password" {
  length = 16
  special = true
  override_special = "/@\" "
}

resource "aws_db_instance" "example" {
  password = "${random_string.password.result}"
}
```

» Argument Reference

The following arguments are supported:

- `length` - (Required) The length of the string desired
- `upper` - (Optional) (default true) Include uppercase alphabet characters in random string.
- `min_upper` - (Optional) (default 0) Minimum number of uppercase alphabet characters in random string.
- `lower` - (Optional) (default true) Include lowercase alphabet characters in random string.
- `min_lower` - (Optional) (default 0) Minimum number of lowercase alphabet characters in random string.
- `number` - (Optional) (default true) Include numeric characters in random string.
- `min_numeric` - (Optional) (default 0) Minimum number of numeric characters in random string.
- `special` - (Optional) (default true) Include special characters in random string. These are '!@#$%&*()-_=+[]{}<>:?'

- `min_special` - (Optional) (default 0) Minimum number of special characters in random string.
- `override_special` - (Optional) Supply your own list of special characters to use for string generation. This overrides characters list in the special argument. The special argument must still be set to true for any overwritten characters to be used in generation.
- `keepers` - (Optional) Arbitrary map of values that, when changed, will trigger a new id to be generated. See the main provider documentation for more information.

» Attributes Reference

The following attributes are exported:

- `result` - Random string generated.

» random_integer

The resource `random_integer` generates random values from a given range, described by the `min` and `max` attributes of a given resource.

This resource can be used in conjunction with resources that have the `create_before_destroy` lifecycle flag set, to avoid conflicts with unique names during the brief period where both the old and new resources exist concurrently.

» Example Usage

The following example shows how to generate a random priority between 1 and 99999 for a `aws_alb_listener_rule` resource:

```
resource "random_integer" "priority" {
  min     = 1
  max     = 99999
  keepers = {
    # Generate a new integer each time we switch to a new listener ARN
    listener_arn = "${var.listener_arn}"
  }
}

resource "aws_alb_listener_rule" "main" {
  listener_arn = "${var.listener_arn}"
  priority     = "${random_integer.priority.result}"
```

```
  action {
    type             = "forward"
    target_group_arn = "${var.target_group_arn}"
  }
  # ... (other aws_alb_listener_rule arguments) ...
}
```

The result of the above will set a random priority.

» Argument Reference

The following arguments are supported:

- `min` - (int) The minimum inclusive value of the range.
- `max` - (int) The maximum inclusive value of the range.
- `keepers` - (Optional) Arbitrary map of values that, when changed, will trigger a new id to be generated. See the main provider documentation for more information.
- `seed` - (Optional) A custom seed to always produce the same value.

» Attribute Reference

The following attributes are supported:

- `id` - (string) An internal id.
- `result` - (int) The random Integer result.

» Import

Random integers can be imported using the `result`, `min`, and `max`, with an optional `seed`. This can be used to replace a config value with a value interpolated from the random provider without experiencing diffs.

Example (values are separated by a `,`): `$ terraform import random_integer.priority 15390,1,99999`