DevOpsSchool
Lets Learn, Share & Practice DevOps

Terraform Functions

DevOpsSchool
Lets Learn, Share & Practice DevOps

Terraform Functions

The Terraform language includes a number of built-in functions that you can call from within expressions to transform and combine values. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses:

```
max(5, 12, 9)
```

DevOpsSchool
Lets Learn, Share & Practice DevOps

Terraform Types

- Numeric Functions
- String Functions
- Collection Functions
- Encoding Functions
- Filesystem Functions
- Date and Time Functions
- Hash and Crypto Functions
- IP Network Functions
- Type Conversion Functions

Terraform Functions

```
#Configure networking

variable network_info {

 default = “10.0.0.0/8” #type, default, description

}

#Returns 10.1.0.0/16

cidr_block = ${cidrsubnet(var.network_info, 8, 1)}

#Returns 10.2.0.0/16

cidr_block = ${cidrsubnet(var.network_info, 8, 2)}
```

Terraform functions

Function	Description	Example
basename(path)	Returns the filename (last elements) of a path	Basename("/home/Edward / file.text") returns file.txt
coalesce(string1, string2, ...) coalescelist(list1, list2, ...)	Returns the first non-empty value Returns the first non-empty list	coalease("","","hello") returns hello
elements(list, index)	Returns a single element from a list at the given index	element(module.vpc.public_subn ets, count.index)
format(format, args, ...) formatlist(format, args, ...)	Formats a string/list according to the given format	format("server-%03d",count.index+1) returns server-001, server-002

Terraform functions

Function	Description	Example
index(list, elem)	Finds the index of a given element in a list	index(aws_instance.foo.*.tags.Env, "prod")
join(delim, list)	Joins a list together with a delimiter	join(“,”, var.AMIS) returns “**ami-123,ami-456,ami-789**”
list(item1, item2, …)	create a new list	join(":", list(“a","b","c")) returns **a:b:c**
lookup(map, key, [default])	Perform a lookup on a map, using “key”. Returns value representing “key” in the map	lookup(map("k", "v"), "k", "not found") returns “**v**”

Terraform functions

Function	Description	Example
Split(delim, string)	splits a string into a list	split(",","a,b,c,d") returns ["a","b","c","d"]
substr(string, offset, length)	extract substring from string	substr("abcde", -3, 3) returns cde
timestamp()	returns RFC 3339 timestamp	"Server started at ${timestamp()}" server started at 2018-06-16T18:46:46Z
upper(string)	Returns uppercased string	upper ("string") returns STRING

Terraform functions

Function	Description	Example
lower(string)	returns lowercase value of "string"	lower("Hello") returns **hello**
map(key, value, …)	returns a new map using key:value	map("k", "v", "k2", "v2") returns: **{ "k" = "v", "k2" = "v2" }**
merge(map1, map2, …)	Merges maps (union)	merge(map("k", "v"), map("k2", "v2")) returns: **{ "k" = "v", "k2" = "v2" }**
replace(string, search, replace)	Performs a search and replace on string	replace("aaab", "a", "b") returns **bbbb**

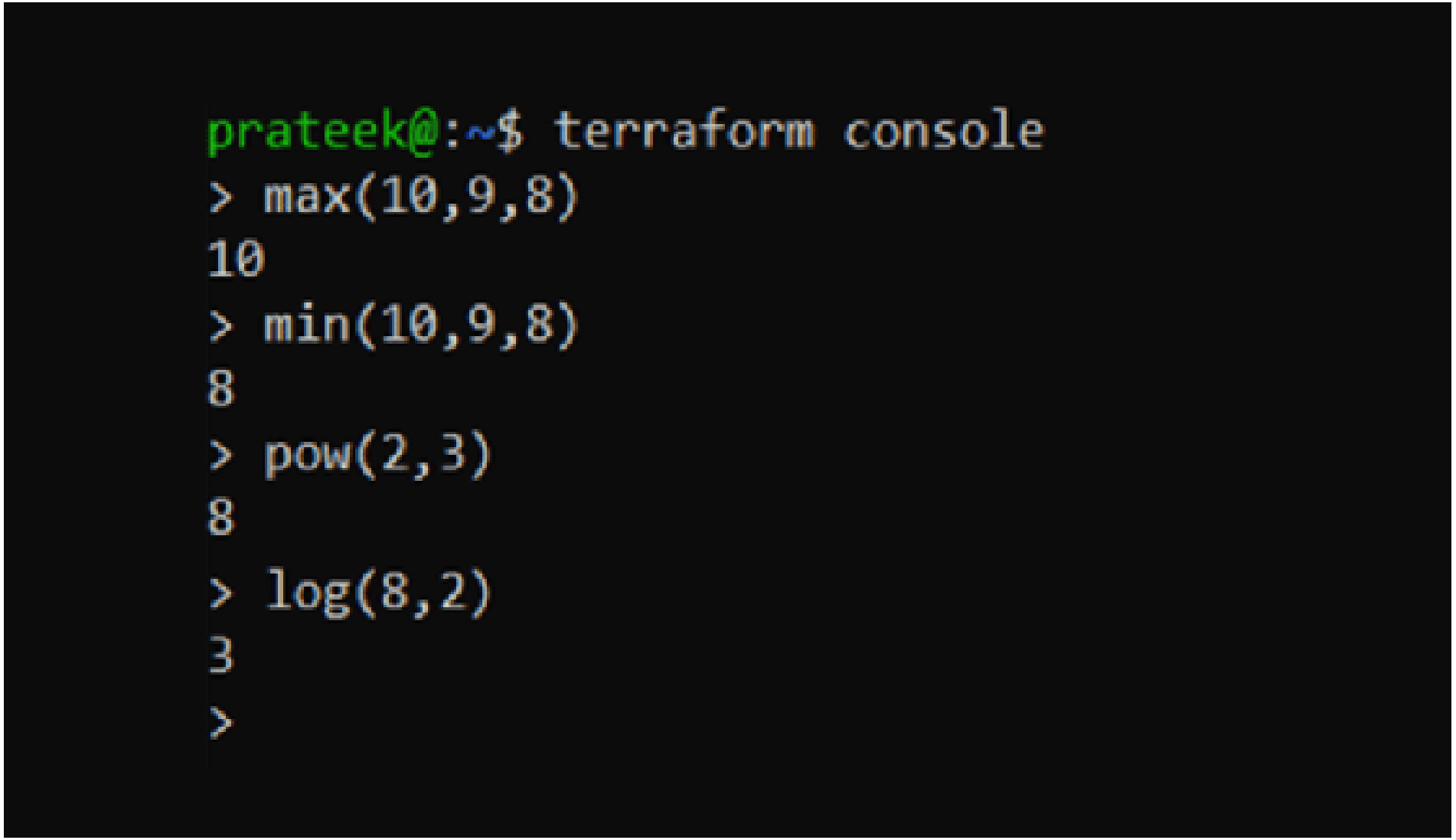
Terraform functions

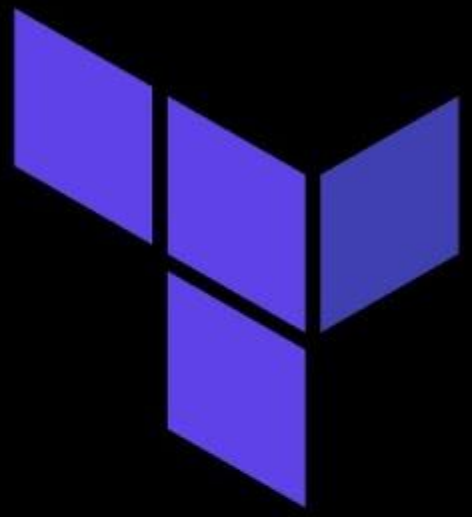
Function	Description	Example
uuid()	Returns a UUID string in RFC 4122 v4 format	uuid() returns: 65b8cf0a-685d-3295-73c1-1393e f71bcd6
values(map)	returns values of a map	values(map("k","v","k2","v2")) returns

```
prateek@:~$ terraform console
> lower("PRATEEK")
prateek
> upper("prateek")
PRATEEK
> title("prateek")
Prateek
```

```
prateek@:~$ terraform console
> split(" ","Prateek Singh")
[
  "Prateek",
  "Singh",
]
> join(" ",["Prateek","Singh"])
Prateek Singh
>

> replace("Password@123","a","@")
P@ssword@123
>
```


prateek@:~$ terraform console
> max(10,9,8)
10
> min(10,9,8)
8
> pow(2,3)
8
> log(8,2)
3
>



Lookup function in Terraform

Terraform Functions

```
#Create ami map
variable “amis” {
  type = “map”
  default = {
    us-east-1 = “ami-1234”
    us-west-1 = “ami-5678”
  }
}
ami = ${lookup(var.amis, “us-east-1”)}
```

```
[root@node1 functions_demo]# terraform console
> max(10,20,30)
30
> min(10,20,30)
10
> lookup({a="ay", b="bee"}, "a", "what?")
ay
> lookup({a="ay", b="bee"}, "b", "what?")
bee
> lookup({a="ay", b="bee"}, "c", "what?")
what?
> element(["a", "b", "c"], 1)
b
> element(["a", "b", "c"], 0)
a
> element(["a", "b", "c"], 2)
c
> element(["a", "b", "c"], 3)
a
>
```

DevOpsSchool
Lets Learn, Share & Practice DevOps

Terraform Workflow Functions

Step #	Command	Invoked Functions
1	`terraform apply` (initial deploy)	`Create()` `Read()`
2	`terraform plan`	`Read()`
3	`terraform apply` (update)	`Read()` `Update()` `Read()`
4	`terraform apply` (force new update)	`Read()` `Delete()` `Create()` `Read()`
5	`terraform destroy`	`Read()` `Delete()`

https://www.terraform.io/docs/language/functions/index.html