**ECS- Services**

* **Pre-requirements:**
* Make sure that Ansible server already has been done the setup from which machine we are ansible script
* Existing AWS account.

Github Link: <https://github.com/absuri/ecs-cluster.git>

#default

provider "aws" {

access\_key = "${var.aws\_access\_key}"

secret\_key = "${var.aws\_secret\_key}"

region = "${var.aws\_region}"

}

variable "aws\_amis" {

type = "map"

default = {

dev = "ami-e4e3fd8e"

staging = "ami-e4e3fd8e"

prod = "ami-e4e3fd8e"

}

}

#### Networking

data "aws\_availability\_zones" "available" {}

resource "aws\_vpc" "ecs-vpc" {

cidr\_block = "10.0.0.0/16"

enable\_dns\_hostnames = true

}

resource "aws\_internet\_gateway" "ecs-igw" {

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

}

resource "aws\_subnet" "ecs-public-1" {

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

cidr\_block = "10.0.1.0/24"

availability\_zone = "${data.aws\_availability\_zones.available.names[0]}"

}

resource "aws\_subnet" "ecs-public-2" {

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

cidr\_block = "10.0.2.0/24"

availability\_zone = "${data.aws\_availability\_zones.available.names[1]}"

}

resource "aws\_route\_table" "ecs-public" {

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = "${aws\_internet\_gateway.ecs-igw.id}"

}

}

resource "aws\_route\_table\_association" "ecs-public" {

subnet\_id = "${aws\_subnet.ecs-public-1.id}"

route\_table\_id = "${aws\_route\_table.ecs-public.id}"

}

resource "aws\_route\_table\_association" "ecs-public-2" {

subnet\_id = "${aws\_subnet.ecs-public-2.id}"

route\_table\_id = "${aws\_route\_table.ecs-public.id}"

}

#### Security

resource "aws\_security\_group" "ecs-lb-sg" {

description = "application ELB security group"

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

name = "tf-ecs-lbsg"

ingress {

protocol = "tcp"

from\_port = 80

to\_port = 80

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = [

"0.0.0.0/0",

]

}

}

resource "aws\_security\_group" "ecs-ec2-sg" {

description = "ECS EC2 security group"

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

name = "ec2-ecs-sec"

ingress {

protocol = "tcp"

from\_port = 8080

to\_port = 8080

security\_groups = [

"${aws\_security\_group.ecs-lb-sg.id}",

]

}

ingress {

protocol = "tcp"

from\_port = 80

to\_port = 80

security\_groups = [

"${aws\_security\_group.ecs-lb-sg.id}",

]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

}

#### ALB

resource "aws\_alb\_target\_group" "ecs" {

name = "hello-world-ecs"

port = 80

protocol = "HTTP"

vpc\_id = "${aws\_vpc.ecs-vpc.id}"

}

resource "aws\_alb" "ecs-alb" {

name = "ecs-alb"

subnets = ["${aws\_subnet.ecs-public-1.\*.id}"]

security\_groups = ["${aws\_security\_group.ecs-lb-sg.id}"]

}

resource "aws\_alb\_listener" "front\_end" {

load\_balancer\_arn = "${aws\_alb.ecs-alb.id}"

port = "80"

protocol = "HTTP"

default\_action {

target\_group\_arn = "${aws\_alb\_target\_group.ecs.id}"

type = "forward"

}

}

#### ECS

resource "aws\_ecs\_cluster" "hw-ecs" {

name = "hw-ecs"

}

#### Auto Scaling Launch Config

resource "aws\_launch\_configuration" "ecs" {

name = "ecs"

image\_id = "$${var.aws\_amis[dev]}"

instance\_type = "${var.instance\_type}"

user\_data = "#!/bin/bash\necho ECS\_CLUSTER=${aws\_ecs\_cluster.hw-ecs.name} > /etc/ecs/ecs.config"

key\_name = "${var.key\_name}"

security\_groups = ["${aws\_security\_group.ecs-ec2-sg.id}"]

}

#### Autoscaling group.

resource "aws\_autoscaling\_group" "ecs" {

name = "ecs-asg"

launch\_configuration = "${aws\_launch\_configuration.ecs.name}"

min\_size = 1

max\_size = 10

desired\_capacity = 1

}

resource "aws\_ecs\_task\_definition" "hello-world" {

family = "hello-world"

container\_definitions = "${file("task-definitions/hello-world.json")}"

placement\_constraints {

type = "memberOf"

expression = "attribute:ecs.availability-zone in [us-east-1b, us-east-1c]"

}

}

resource "aws\_ecs\_service" "hello-world" {

name = "hello-world"

name = "family"

task\_definition = "${aws\_ecs\_task\_definition.hello-world.id}"

cluster = "${aws\_ecs\_cluster.hw-ecs.id}"

desired\_count = 2

**Explanation:**

* Create an IAM user give the EC2-container full-access
* Create a EC2-container registry
* Build a custom image(hello-world application)
* Tag and push the images to ECS registry.
* Than after... IAM user credentials enter into this path "~/.aws"
* Command: "aws ECS describe-services" is used to check the connection B/W ansible server and IAM user.
* After connection established than above playbooks are run in ansible-server.
* Than deploy hello-world application in ECS container service by using ansible