Configure new service or scheduled task

Overview

Amazon Elastic Container Service (ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances. Amazon ECS provides a service scheduler (for long-running tasks and applications), the ability to run tasks manually (for batch jobs or single run tasks), with Amazon ECS placing tasks on your cluster for you. In this article, the two modules fargate-service and fargate-task which are used to create service and scheduled tasks respectively are discussed.

Content

New Service

Amazon Elastic Container **Service** (**ECS**) is a highly scalable, high-performance container management **service** that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon **EC2** instances. The module fargate-service provisions a docker container into an ECS(AWS Elastic Container Service) cluster with networking, service discovery registration, and health checks provided. It also provides settings on how to run and configure the container (e.g., private/public, inject secrets, CPU, and memory).

It creates:

- Secret Manager: Secret Manager is used to storing service secrets with encryption, using KMS key
- Service: Creates a public OR a private service based on whether it would be attached to a load balancer or not
- Load Balancer listener Rule: Listener Rule is attached to the load balancer which is passed in from the variables
- Log Group: Log group in Cloudwatch where all the application logs are stored
- Alarms: For monitoring service metrics. Alarms can send notifications to multiple emails. Type of notifications which can be received are:
  - Average container CPU utilization over the last 5 minutes >80%
  - Average container memory utilization over the last 5 minutes >80%
  - Average container CPU utilization over the last 5 minutes >80%
  - Average container memory utilization over the last 5 minutes >80%

More details about the module can be found here

The example `service-api` in canary application uses the `fargate-service` module and creates the new service named `service-api`. The module code initializes the variables `cpu` and `memory` from the terraform variables which need to be configured for the workspace before executing the terraform script.

```
module "service_api" {
  source  = "app.terraform.io/min-au-infra/fargate-service/aws"
  version = "v5.1.0"

  service_name = "${var.application_name}-${local.service_api_name}"
  charge_code  = var.charge_code
  custom_tags  = var.custom_tags

  cpu    = var.api_service_info.cpu_allocation
  memory = var.api_service_info.mem_allocation

  service_count = var.api_service_info.num_containers
  service_port  = local.service_api_port
  service_image = var.api_ecr_repository_url
  envvars       = concat(local.service_api_envvars, module.postgres_db.
postgres_env_vars)
  email_ids     = var.monitoring_email_ids

  attach_lb        = true
```

```
  alb_path         = "/api/*"
  alb_priority     = 100
  alb_listener_arn = module.ecs_cluster.alb_listener_http_arn

  ecs_cluster_name = module.ecs_cluster.ecs_cluster_name
  vpc_id           = var.vpc_id
  subnets          = var.private_subnet_ids
  security_groups = concat(
    module.ecs_cluster.ecs_task_security_group_ids,
    [module.postgres_db_cluster.postgres_user_security_group],
  )

  service_discovery_namespace = module.ecs_cluster.
service_discovery_namespace

  # This must be set to the length of the "ecs_task_role_policy_arns"
array.
  # This is to go around a Terraform v11 bug regarding count() with
computed values
  ecs_task_role_policy_arns_count = 2

  ecs_task_role_policy_arns = [
    module.postgres_db.rds_policy_arn,
    module.test_bucket.consumer_policy_arn,
  ]

  health_check = {
    interval            = 20
    path                = "/health-check"
    timeout             = 10
    healthy_threshold   = 3
    unhealthy_threshold = 3
    matcher             = "200,201,204"
  }

  secrets_count = 1

  secrets = [{
    name  = "default"
    value = "${jsonencode(var.api_secret_value)}"
  }]

  account_alias      = var.spoke_name
  log_aggregation_s3 = var.log_aggregation_s3

  providers = {
    aws.service = aws.spoke
    aws.logging = aws.logging
  }
}
```

Scheduled Task

Amazon ECS provides a service scheduler (for long-running tasks and applications), the ability to run tasks manually (for batch jobs or single run tasks), with Amazon ECS placing tasks on your cluster for you. Fargate Tasks can be used for one-off (short running) jobs. Fargate task essentially creates a set of containers as defined in the Task definition and the containers terminate after executing the assigned job. Tasks will have to be triggered again when jobs arrive. This component provisions a set of containers into an ECS cluster, with networking, for event-based architectures.

> This component will require ECS cluster to host the task definition and containers.

The module creates:

- Task Definition: Describe one or more containers that form your application
- Secret Manager: Secret Manager is used to storing task secrets with encryption using the KMS key. The secret injection is optional and secured through the secret manager. You can define multiple secrets and inject values.
- Log Group: Log group in Cloudwatch where all the application logs are stored. These logs also aggregate to the logging account for audit purposes.
- Monitoring: Alerts would be sent to monitoring emails when CPU or RAM utilization of the containers goes above 80%

Example template for creating the scheduled task.

> Template resembles as that of service but underlying modules used are different.

```
module "task-runner" {
  source = "https://gitlab.com/mc-components/terraform-aws-fargate-task"

  container_name = "${var.application_name}-${local.task_name}"
  charge_code  = var.charge_code

  cpu    = var.task-runner_container_info.cpu_allocation
  memory = var.task-runner_container_info.mem_allocation

  container_count = var.task-runner.num_containers
  container_port  = local.task-runner_port
  container_image = var.task-runner_ecr_repository_url
  envvars       = local.task-runner_envvars
  email_ids     = var.monitoring_email_ids

  ecs_cluster_name = module.ecs_cluster.ecs_cluster_name
  vpc_id           = var.vpc_id
  subnets          = var.private_subnet_ids

  ecs_task_role_policy_arns = [
    module.data_bucket.consumer_bucket_policy_arn,
  ]

  secrets = [{
    name  = "default"
    value = "${jsonencode(var.task-runner_secret_value)}"
  }]

  account_alias      = var.spoke_name
  log_aggregation_s3 = var.log_aggregation_s3

  providers = {
    aws.compute = aws.spoke
    aws.logging = aws.logging
  }
}
```

Relevant materials

- Task Module definition https://app.terraform.io/app/min-au-infra/modules/view/fargate-task/aws/2.1.1
- Service Module definition https://app.terraform.io/app/min-au-infra/modules/view/fargate-service/aws/6.0.4
- ECS services https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs_services.html
- Scheduling an ECS task https://docs.aws.amazon.com/AmazonECS/latest/developerguide/scheduling_tasks.html
- Task module code https://app.terraform.io/app/min-au-infra/modules/view/fargate-task/aws/2.1.1
- Service module code https://gitlab.com/mc-components/terraform-aws-fargate-service