Optimization as Code *for Containers*

About Densify's Container Optimization

Densify's technology learns your cloud and container workload patterns and recommends the perfectly fit resources to deploy on, enabling improved application performance, and less risk.

Densify allows you to stop hard coding resource specifications, and enables full automation in your cloud infrastructure, giving your cloud and container workloads the ability to become self-aware and continuously optimized. This seamlessly extends your CI/CD pipeline to enable **CI/CD/CO (Continuous Optimization)**.

How it Works

Densify provides an API call that returns a map file of all the current sizes, recommendations, approval codes and other information that that is used to upside or downsize instances, based on workload demands. Densify's integration module is available on the Terraform public repository: www.densify.com/tfregistry

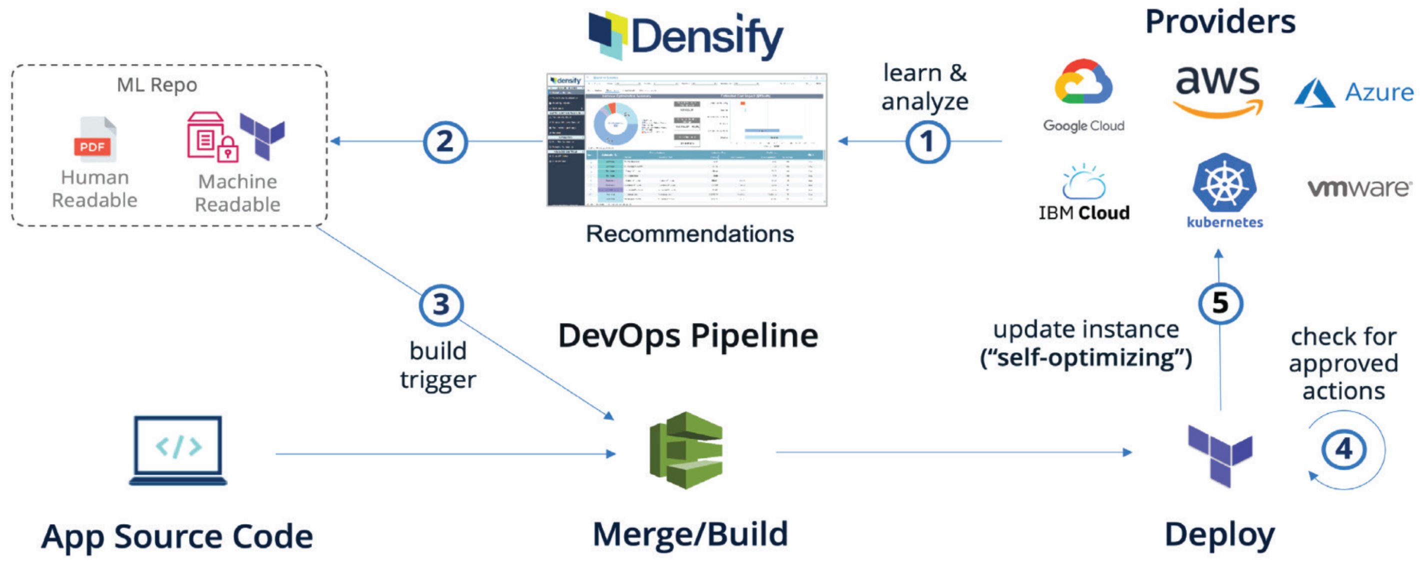
In order to enable this module, you will need 3 inputs:

1. **Densify Recommendations:** These can be attained in Terraform format from the Densify API
2. **A Fallback Profile:** This is the size that will be used when a recommendation isn't provided by Densify
3. **Densify Unique ID:** This is the system name. However, if you have system names that need to be duplicated, you can set provisioning_id on the system

The module has several output variables that can be used:

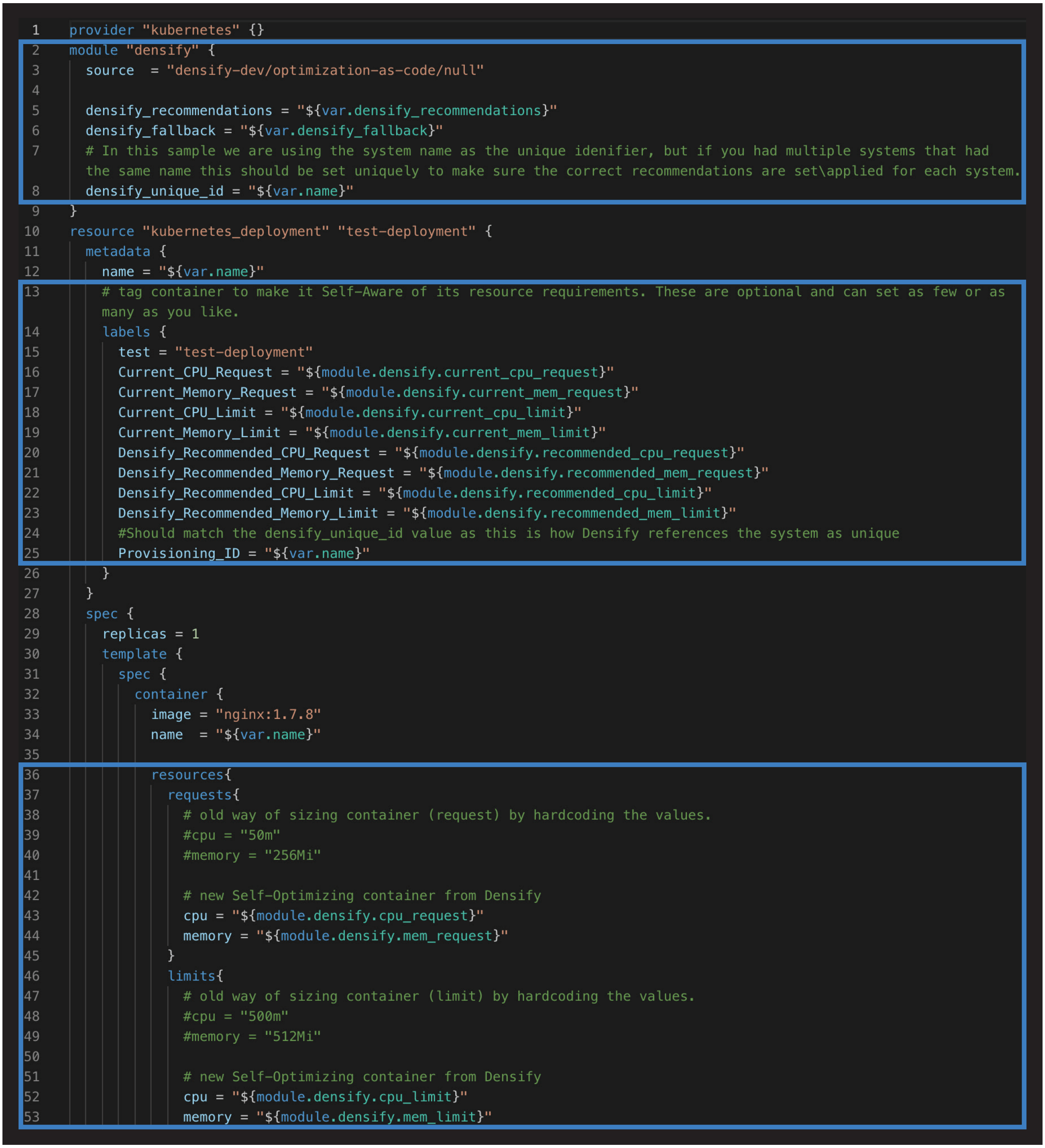
- With the module installed, items such as *instance_type, cpu_request, cpu_limit, mem_request*, and *mem_limit* are meant to replace hard-coded values in your script and will provide values based on approved changes and recommendations
- Items such as recommended_type, current_type, approval, and savings_estimate are there to allow you to **set labels or tags** on the systems **to make them self-aware**

The Provisioning_ID is the tag that would need to be set if you have hostnames that are duplicates to ensure the results can be linked from Densify to Terraform.



Densify Enables Full Automation By:

1. Learning the workload patterns of the cloud and container resources
2. Making specific recommendations to improve app performance, and drive down risk and cost, while generating human readable and machine readable output, which are used to build and initiate triggers in the DevOps Pipeline
3. Going through an automated approval process, allowing cloud and container workloads to achieve true self-optimization



```
provider "kubernetes" {}
module "densify" {
  source  = "densify-dev/optimization-as-code/null"

  densify_recommendations = "${var.densify_recommendations}"
  densify_fallback = "${var.densify_fallback}"
  # In this sample we are using the system name as the unique idenifier, but if you had multiple systems that had the same name this should be set uniquely to make sure the correct recommendations are set\applied for each system.
  densify_unique_id = "${var.name}"
}
resource "kubernetes_deployment" "test-deployment" {
  metadata {
    name = "${var.name}"
    # tag container to make it Self-Aware of its resource requirements. These are optional and can set as few or as many as you like.
    labels {
      test = "test-deployment"
      Current_CPU_Request = "${module.densify.current_cpu_request}"
      Current_Memory_Request = "${module.densify.current_mem_request}"
      Current_CPU_Limit = "${module.densify.current_cpu_limit}"
      Current_Memory_Limit = "${module.densify.current_mem_limit}"
      Densify_Recommended_CPU_Request = "${module.densify.recommended_cpu_request}"
      Densify_Recommended_Memory_Request = "${module.densify.recommended_mem_request}"
      Densify_Recommended_CPU_Limit = "${module.densify.recommended_cpu_limit}"
      Densify_Recommended_Memory_Limit = "${module.densify.recommended_mem_limit}"
      #Should match the densify_unique_id value as this is how Densify references the system as unique
      Provisioning_ID = "${var.name}"
    }
  }
  spec {
    replicas = 1
    template {
      spec {
        container {
          image = "nginx:1.7.8"
          name  = "${var.name}"

          resources{
            requests{
              # old way of sizing container (request) by hardcoding the values.
              #cpu = "50m"
              #memory = "256Mi"

              # new Self-Optimizing container from Densify
              cpu = "${module.densify.cpu_request}"
              memory = "${module.densify.mem_request}"
            }
            limits{
              # old way of sizing container (limit) by hardcoding the values.
              #cpu = "500m"
              #memory = "512Mi"

              # new Self-Optimizing container from Densify
              cpu = "${module.densify.cpu_limit}"
              memory = "${module.densify.mem_limit}"
```

Reach out to us at
answers@densify.com
to learn more and get started!