ORACLE

Automating Oracle Container Engine for Kubernetes (OKE) Deployment on Oracle Compute Cloud@Customer

Step-by-step to automate deployments of Oracle Container Engine for Kubernetes (OKE) on Compute Cloud@Customer using Terraform

Version [1.0]

Public

ORACLE

Purpose statement

This document provides the step-by-step to fully automate the deployment of Oracle Container Engine for Kubernetes (OKE) on Oracle Compute Cloud@Customer using customized Terraform scripts.

Disclaimer

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle software license and service agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This document is for informational purposes only and is intended solely to assist you in planning for the implementation and upgrade of the product features described. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document remains at the sole discretion of Oracle. Due to the nature of the product architecture, it may not be possible to safely include all features described in this document without risking significant destabilization of the code.

ORACLE

Table of contents

Introduction

In today's changing world of business IT, automation and native solutions play a crucial role in ensuring flexibility, scalability and effectiveness. As more companies embrace Kubernetes for managing containers the importance of deployment processes becomes clear. This guide aims to address this need by offering a step-by-step on automating the setup of Oracle Container Engine for Kubernetes (OKE) using tailored Terraform scripts. By utilizing Terraform - an infrastructure as code tool - organizations can achieve automated deployments reducing manual work and minimizing the chances of errors.

Prerequisites

Before delving into the steps for setting up Oracle Container Engine for Kubernetes (OKE), it's important to make sure you have all the necessary prerequisites ready. This section outlines the elements needed to carry out the automation process using customized Terraform scripts.

The prerequisites covered in this document encompass operational aspects, including:

- **Having access to Compute Cloud@Customer:** Make sure you have an active account with the appropriate permissions to create and manage resources on Oracle Compute Cloud@Customer.
- **Terraform Setup:** Confirm that Terraform is installed and set up on your machine or CI/CD pipeline environment. It's also helpful to be familiar with Terraform concepts and syntax.
- **Networking Considerations:** Ensure that your network setup allows for communication, which includes setting up VCN (Virtual Cloud Network) configuring subnets and managing security lists.
- **Access Credentials:** You need a bastion host to connect with your Compute Cloud@Customer to deploy the OKE. Collect the API keys SSH keys and Oracle Compute Cloud@Customer credentials required for authentication and authorization throughout the deployment process.
- **Infrastructure Planning:** Develop a strategy outlining the resources needed capacity projections and design aspects customized to suit your organizations requirements.
- **Fundamental Understanding of Scripting:** Proficiency, in scripting languages in adapting and tailoring Terraform scripts for deployment situations.

By fulfilling these criteria, you will be ready to follow the instructions outlined in this paper guaranteeing a seamless and effective automation of your Oracle Container Engine for Kubernetes on Oracle Compute Cloud@Customer.

Bastion Host

A bastion host, also referred to as a jump server, serves as a server created to enable access to a private network from an external network, like the internet. For the OKE deployment on Oracle Compute Cloud@Customer, a bastion host is utilized to host all Terraform scripts utilized to deploy the OKE infrastructure. Also, it is utilized for reaching resources and instances within a subnet in your Compute Cloud@Customer environment.

Advantages of Using a Bastion Host are:

- Access Point: Acts as an entry point for administrators and users to connect to instances in the private network without exposing them directly to the internet.
- Offers a controlled gateway thereby reducing the vulnerability surface.
- Isolation of Critical Resources: Ensures that computing resources within the network are safeguarded and isolated from internet exposure. Helps in minimizing risks associated with attacks on systems.
- Monitoring: Centralizes access logging simplifying tracking and auditing access to resources. Can be integrated with security information and event management (SIEM) systems, for monitoring capabilities.

By implementing a bastion host, you can securely and efficiently manage access to your Oracle Compute Cloud@Customer resources, ensuring that your private instances remain protected and accessible only through a secure, controlled entry point.

Bastion Host Configuration

1. Install OCI CLI in your bastion host. For the step-by-step and how to install OCI CLI and properly setup the OCI CLI configuration file in your instance, refer to the following links below: https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/cliinstall.htm and https://docs.oracle.com/en/engineered-systems/private-cloud-appliance/3.0-latest/user/user-usr-ce-cli.html#usr-cli-obtain-cabundle

2. Ensure you have Terraform properly installed: To use the Oracle Cloud Infrastructure (OCI) Terraform provider, you must install both Terraform and the OCI Terraform provider. You can install both Terraform and the OCI Terraform provider with yum, or directly download them from HashiCorp. Refer to this link: https://docs.oracle.com/en-us/iaas/Content/API/SDKDocs/terraforminstallation.htm
3. Ensure that your bastion host can access your Compute Cloud@Customer and properly authenticate using OCI CLI and Terraform

Terraform Scripts Overview

To make it easier to set up and manage Oracle Kubernetes Engine (OKE) on Oracle Compute Cloud@Customer, we use a range of Terraform scripts. The set of scripts listed below will automate the OKE deployment (end-to-end) including all resources needed, such as VCN, subnetwork, security lists, OKE clusters, and node pool. Listed below is the description of each Terraform script utilized to fully deploy OKE on Compute Cloud@Customer.

NOTE: Refer to the Appendix section of this white paper for the source code of all Terraform scripts.

main.tf:

- **Description**: Acts as the primary configuration file, orchestrating the deployment of all resources within the Oracle Kubernetes Engine (OKE) environment. It integrates various modules and defines the overarching structure of the Terraform deployment.

provider.tf:

- **Description**: Specifies provider configurations and credentials necessary for authenticating and interacting with Oracle Compute Cloud@Customer.

oke_worker_subnet.tf:

- **Description**: Configures the subnet designated for OKE worker nodes, detailing the network settings and parameters to ensure proper isolation and connectivity within the OKE cluster.

oke_worker_seclist.tf:

- **Description**: Defines the security list configurations for the OKE worker nodes subnet, specifying the inbound and outbound rules to manage traffic and ensure secure operations of worker nodes.

oke_vcn.tf:

- **Description**: Establishes the Virtual Cloud Network (VCN) setup, including the definition of subnets, route tables, and other network components necessary for the OKE cluster's infrastructure.

oke_kmi_subnet.tf:

- **Description**: Configures the subnet for Kubernetes Master Instances within the OKE cluster, ensuring that the master nodes have a dedicated and properly configured network space.

oke_kmi_seclist.tf:

- **Description**: Specifies the security list settings for the Kubernetes Master Instances subnet, detailing the rules required to control network traffic and secure the master nodes.

variables.tf:

- **Description**: Declares input variables used throughout the Terraform configuration, allowing for customization and reuse of the deployment parameters.

image.tf:

- **Description**: Specifies the Oracle Compute Cloud@Customer images used to create the compute instances within the OKE cluster.

oke_cluster.tf:

- **Description**: Defines the settings and parameters for deploying the OKE cluster, including cluster size, version, and other critical configurations necessary for the cluster setup.

oke_pool.tf:

- **Description**: Configures the node pool for the OKE cluster, specifying the characteristics and sizing of the worker nodes.

terraform.tfvars:

- **Description**: Contains the variable values used to customize the Terraform deployment, including sensitive information such as API keys, OCID of Tenancy, and compartment details. This file centralizes configuration data, making it easier to manage and secure.

NOTE: Ensure you have all scripts listed below together in a folder in your bastion host. For demonstration purposes, all scripts are utilizing standard network CIDR configuration, OKE cluster/node pool names, and sample names for authentication. Ensure to adjust the highlighted in red below to best fit your environment. Below is a sample of a bastion host instance with all Terraforms scripts needed for OKE deployment on Compute Cloud@Customer.

```
-rw-r--r--. 1 root root   345 Jul  9 21:35 provider.tf
-rw-r--r--. 1 root root  1083 Jul 12 18:57 oke_worker_subnet.tf
-rw-r--r--. 1 root root  1538 Jul 12 18:57 oke_worker_seclist.tf
-rw-r--r--. 1 root root  1400 Jul 12 18:57 oke_vcn.tf
-rw-r--r--. 1 root root  1145 Jul 12 18:57 oke_kmi_subnet.tf
-rw-r--r--. 1 root root  2429 Jul 12 18:57 oke_kmi_seclist.tf
-rw-r--r--. 1 root root  2310 Jul 12 18:57 main.tf
-rw-r--r--. 1 root root  2405 Jul 15 21:50 variables.tf
-rw-r--r--. 1 root root   323 Jul 15 21:51 image.tf
-rw-r--r--. 1 root root  1400 Jul 15 22:07 oke_cluster.tf
-rw-r--r--. 1 root root   897 Jul 15 22:09 oke_pool.tf
-rw-r--r--. 1 root root  1412 Jul 16 15:35 terraform.tfvars
```

Figure 1. Terraform scripts utilized for OKE deployment on Compute Cloud@Customer

NOTE: Refer to the Appendix section for detailed information of all Terraform scripts utilized to deploy OKE on Compute Cloud@Customer.

OKE Deployment Workflow

Setting up Oracle Kubernetes Engine (OKE) on Oracle Compute Cloud@Customer involves a deployment process divided into three phases, they are:

1. **Establishing Network Infrastructure:** The initial step focuses on creating the network infrastructure, such, as configuring Virtual Cloud Networks (VCNs) subnets and security lists tailored for OKE worker and master nodes. This setup enables the network environment to support isolated communication channels required for the Kubernetes clusters operation.

2. **Deploying OKE Cluster:** After setting up the network the next phase involves deploying the OKE cluster itself. This process includes creating the Kubernetes cluster serving as the orchestration platform for applications. The deployment ensures that the cluster is correctly initialized and prepared to handle workloads.
3. **Configuring OKE Node Pools:** The final stage revolves around setting up and deploying node pools, for the OKE cluster. This step encompasses provisioning worker nodes, where containerized applications will be executed. By deploying node pools, the cluster is furnished with resources to manage application workloads effectively.

OKE Deployment Using Terraform

Assuming you have the bastion host properly configured and all Terraform scripts adjusted to your environment, perform the following steps to deploy OKE on Compute Cloud@Customer. Listed below are the steps-by-steps.

- Run **terraform init** command line: The **terraform init** command is used to initialize a working directory containing Terraform configuration files. This command performs several essential tasks to set up the working directory for future Terraform commands:

  - **Initializes Backend Configuration**: Sets up the backend where Terraform's state file will be stored. This could be a local file or a remote storage solution.
  - **Installs Provider Plugins**: Downloads the necessary provider plugins specified in the configuration files to manage the resources.
  - **Validates Configuration Files**: Checks the syntax and structure of the Terraform configuration files to ensure they are correct.

```
[root@oke-bastion-c3-se tf-oke]# terraform init
Initializing the backend...
Initializing provider plugins...
- Reusing previous version of oracle/oci from the dependency lock file
- Using previously-installed oracle/oci v6.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Figure 2. Terraform init command line output

- Adjust the script variables to best fit your environment. For example: OCIDs, public keys, VCN/subnet name, network CIDR, OKE Cluster and pool name. Refer to the appendix section for details.
- In the same folder in your bastion host, run **terraform apply** command, then when asked by Terraform, type **yes** to perform the actions needed to fully deploy OKE on Compute Cloud@Customer.

```
Plan: 16 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

Figure 3. Terraform scripts execution to deploy OKE on Compute Cloud@Customer

NOTE: It takes about 15 minutes to fully deploy OKE on Compute Cloud@Customer.

- First, the Terraform scripts will create all network infrastructure, such as VCNs, subnets, security lists, internet and NAT gateways needed for OKE. Listed below is output log from the Terraform scripts during the infrastructure deployment for OKE on Compute Cloud@Customer:

```
oci_core_vcn.oke_vcn: Creating...

oci_core_vcn.oke_vcn: Still creating... [10s elapsed]

oci_core_vcn.oke_vcn: Still creating... [20s elapsed]

oci_core_vcn.oke_vcn: Still creating... [30s elapsed]

oci_core_vcn.oke_vcn: Still creating... [40s elapsed]

oci_core_vcn.oke_vcn: Still creating... [50s elapsed]

oci_core_vcn.oke_vcn: Still creating... [1m0s elapsed]

oci_core_vcn.oke_vcn: Still creating... [1m10s elapsed]

oci_core_vcn.oke_vcn: Still creating... [1m20s elapsed]

oci_core_vcn.oke_vcn: Still creating... [1m30s elapsed]

oci_core_vcn.oke_vcn: Creation complete after 1m36s

oci_core_internet_gateway.vcn_igs: Creating...

oci_core_nat_gateway.vcn_ngs: Creating...

oci_core_security_list.workerlb: Creating...

oci_core_security_list.kmilb: Creating...

oci_core_default_security_list.oke_vcn: Creating...

oci_core_security_list.kmi: Creating...

oci_core_security_list.worker: Creating
```

- Check if all network configuration has been properly deployed in your Compute Cloud@Customer. To do this step, login in the UI of your Compute Cloud@Customer, click on Dashboard and Networking to view your Virtual Cloud Network configuration, then select your compartment which you are deploying your new OKE cluster. The new VCN will be created. The new VCN for your OKE cluster will be available.

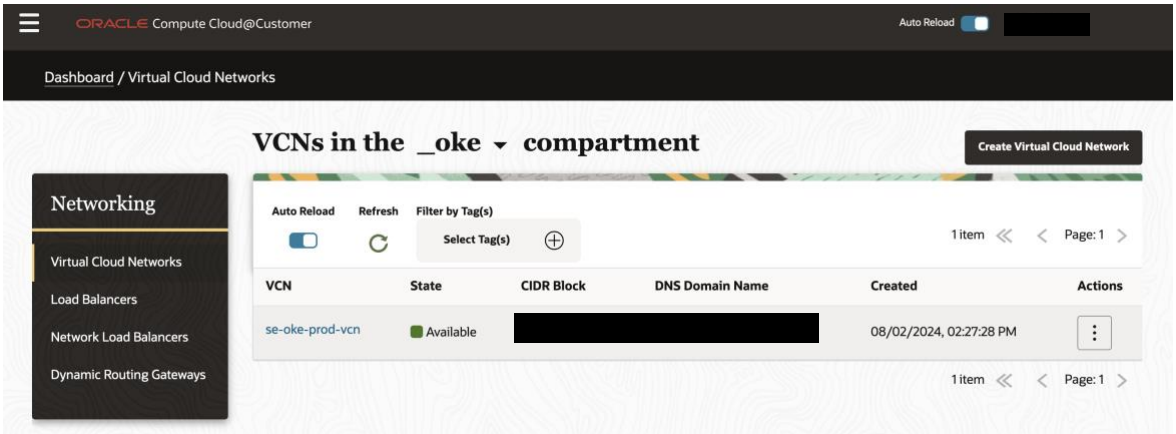

Figure 4. Compute Cloud@Customer Virtual Cloud Network Configuration

- On the same place, click on your VCN name, then go to subnets to list all subnets and network resources deployed by the Terraform scripts, such as: Security lists, internet and NAT gateways, and route tables. NOTE: On this example, we are using the se-oke-prod-vcn.

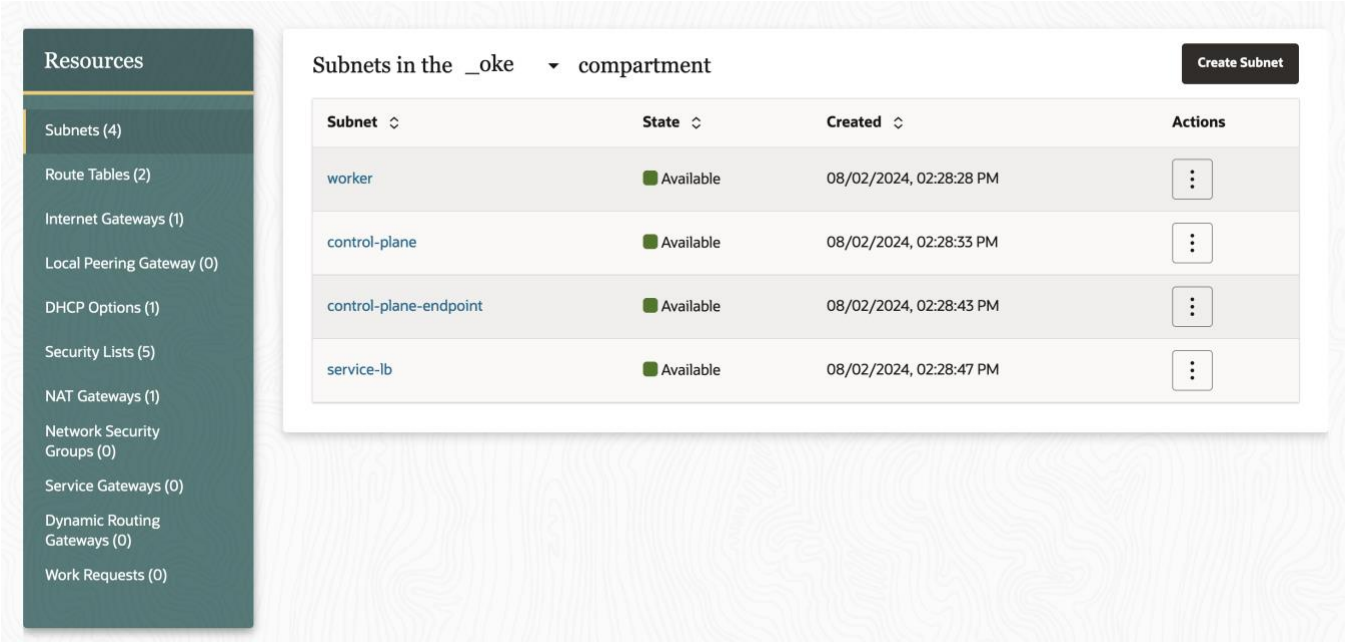

Figure 5. Compute Cloud@Customer subnets and network resources overview

- Second, the Terraform scripts will create the Kubernetes cluster. This step can take up to 10-12 minutes. During this step, the new OKE cluster will be on the "Creating" status on the Compute Cloud@Customer dashboard. See output log from Terraform script below:

```
oci_containerengine_cluster.SEOKECluster: Creating...

oci_containerengine_cluster.SEOKECluster: Still creating... [10s elapsed]...[10m50s elapsed]
```

- Check if the new OKE cluster has been properly deployed in your Compute Cloud@Customer. To do this step, login in your Compute Cloud@Customer UI, click on Dashboard, then Containers. Your new OKE cluster will be available. See Figures 5 and 6 below.

NOTE: On this example, we are using the **oke_cluster-1** as the name of our new OKE cluster, and **oke_node_pool** as the name of the OKE node pool on Compute Cloud@Customer.

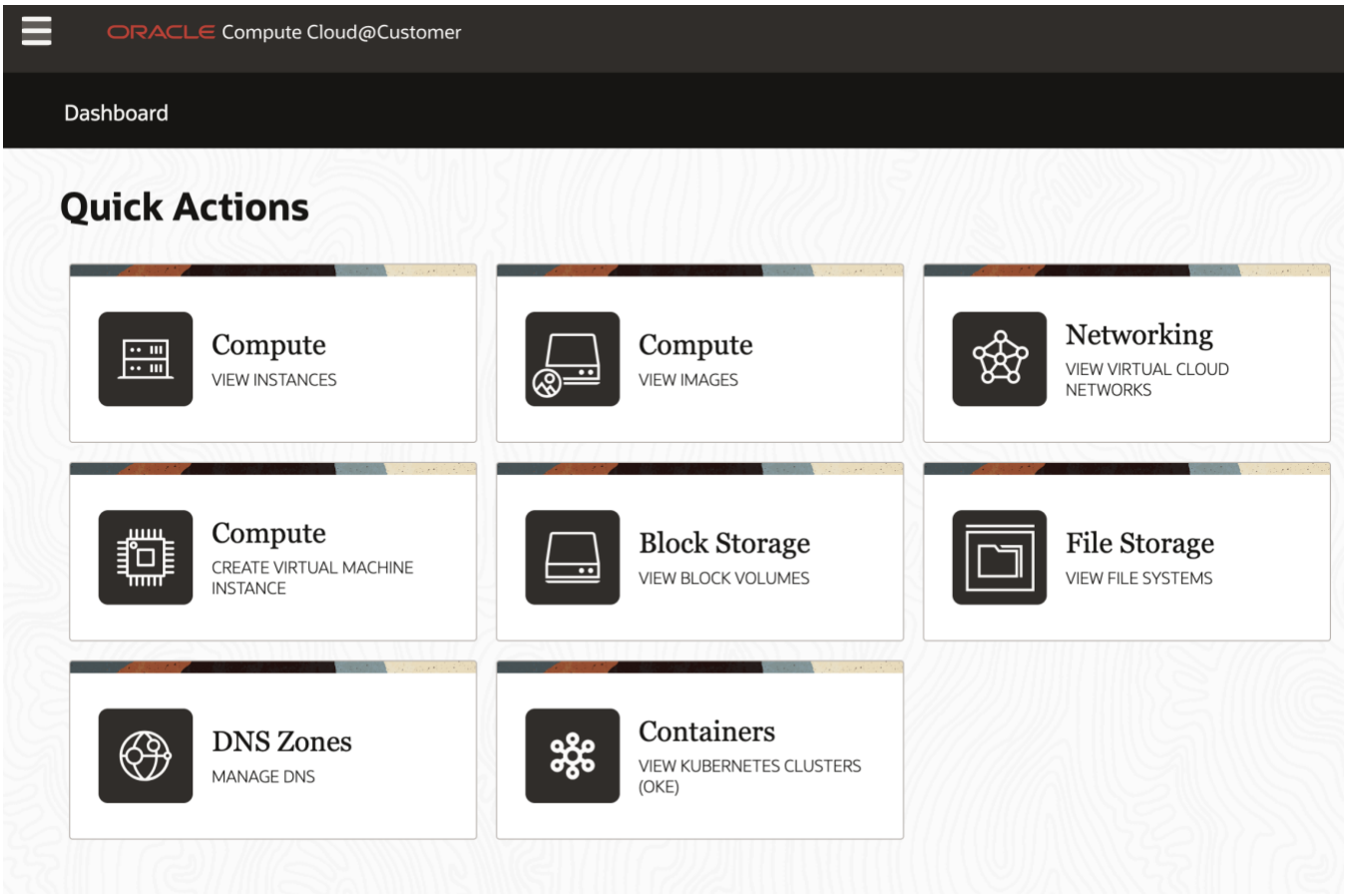

Figure 6. Compute Cloud@Customer Dashboard

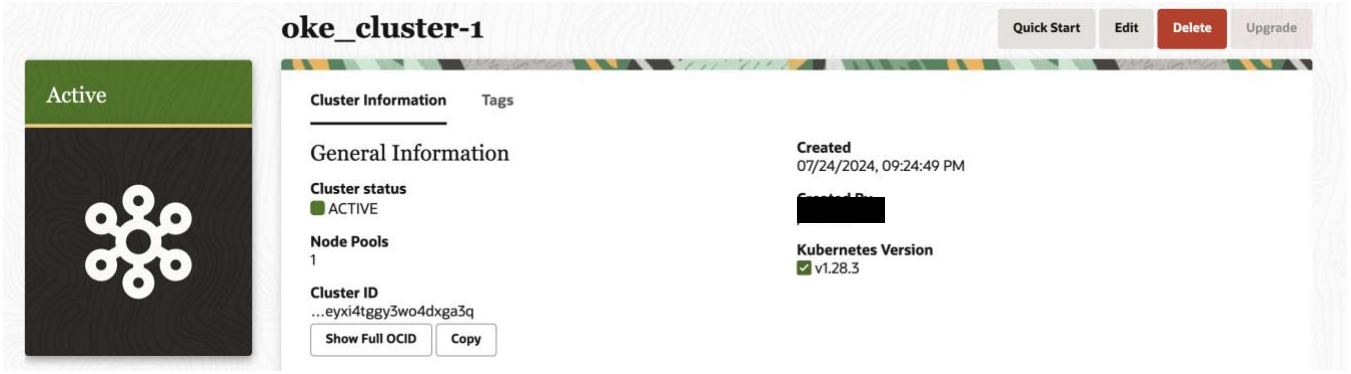

Figure 7. Compute Cloud@Customer Dashboard. OKE Clusters.

- Third, the Terraform scripts will create the node pool for your Kubernetes cluster and the new OKE cluster will be available in your Compute Cloud@Customer for new workload.

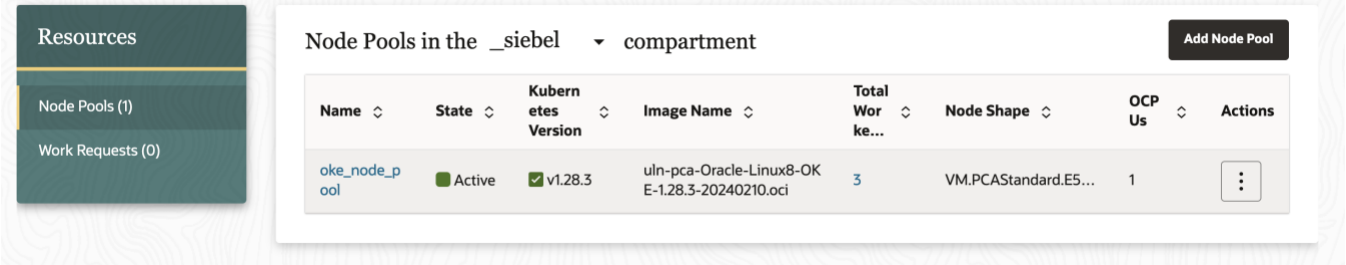

Figure 8. Compute Cloud@Customer Dashboard. OKE Node Pools.

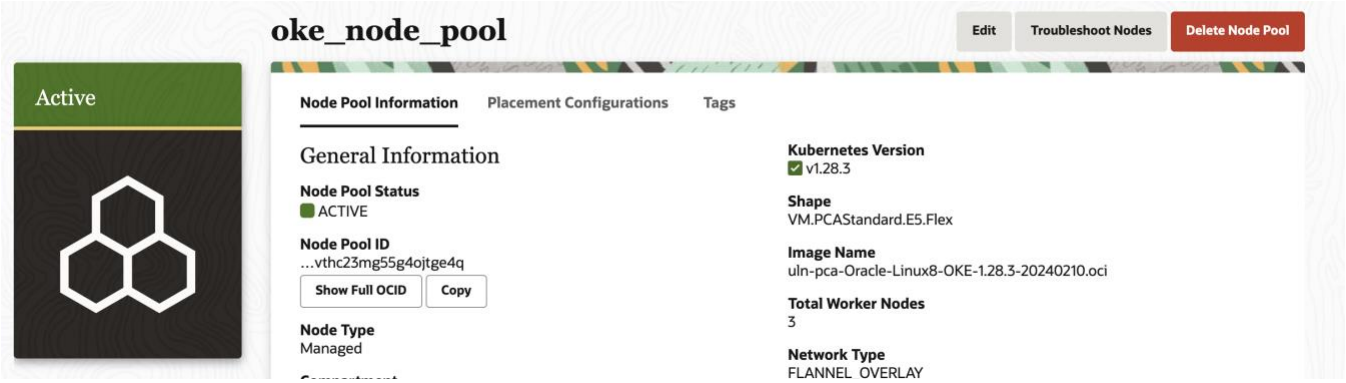

Figure 9. Compute Cloud@Customer Dashboard. OKE Node Pool details

At this stage, the new OKE cluster will be fully deployed on Compute Cloud@Customer and ready to receive new workloads.

ORACLE

Appendix

main.tf

```
terraform {
  required_providers {
    oci = {
      source  = "oracle/oci"
      version = ">= 4.50.0"
    }
  }
  required_version = ">= 1.1"
}
locals {
  kube_internal_cidr = "253.255.0.0/16"
  worker_lb_ingress_rules = var.worker_lb_ingress_rules
  worker_ingress_rules = flatten([var.worker_ingress_rules, [
    {
      source   = var.vcn_cidr
      port_min = 22
      port_max = 22
    },
    {
      source   = var.workerlb_cidr
      port_min = 30000
      port_max = 32767
    },
    {
      source   = var.workerlb_cidr
      port_min = 10256
      port_max = 10256
    },
    {
      source   = var.kmi_cidr
      port_min = 22
```

```
    port_max = 65535
  },
]])
worker_ingress_udp_rules = [
  {
    source   = var.worker_cidr
    port_min = 8285
    port_max = 8472
  },
  {
    source   = var.kmi_cidr
    port_min = 8285
    port_max = 8472
  },
]
kmi_lb_ingress_rules = [
  {
    source   = local.kube_internal_cidr
    port_min = var.kubernetes_api_port
    port_max = var.kubernetes_api_port
  },
  {
    source   = var.kube_client_cidr
    port_min = var.kubernetes_api_port
    port_max = var.kubernetes_api_port
  },
  {
    source   = var.vcn_cidr
    port_min = var.kubernetes_api_port
    port_max = var.kubernetes_api_port
  },
]
kmi_ingress_rules = [
  {
```

```
      source   = var.kube_client_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.kmilb_cidr
      port_min = var.kubernetes_api_port
      port_max = var.kubernetes_api_port
    },
    {
      source   = var.worker_cidr
      port_min = 1024
      port_max = 65535
    },
    {
      source   = var.kmi_cidr
      port_min = 1024
      port_max = 65535
    },
  ]
  kmi_ingress_udp_rules = [
    {
      source   = var.worker_cidr
      port_min = 8285
      port_max = 8472
    },
    {
      source   = var.kmi_cidr
      port_min = 8285
      port_max = 8472
    },
  ]
}
```

provider.tf

```
provider "oci" {

  config_file_profile = var.oci_config_file_profile

  tenancy_ocid        = var.tenancy_ocid

}
```

oke_worker_subnet.tf

```
resource "oci_core_subnet" "worker" {

  cidr_block     = var.worker_cidr

  compartment_id = var.compartment_id

  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name               = "worker"

  dns_label                  = "worker"

  prohibit_public_ip_on_vnic = true

  security_list_ids = [

    oci_core_default_security_list.oke_vcn.id,

    oci_core_security_list.worker.id

  ]

}

resource "oci_core_subnet" "worker_lb" {

  cidr_block     = var.workerlb_cidr

  compartment_id = var.compartment_id

  vcn_id         = oci_core_vcn.oke_vcn.id

  display_name               = "service-lb"

  dns_label                  = "servicelb"

  prohibit_public_ip_on_vnic = false

  route_table_id             = oci_core_route_table.public.id

  security_list_ids = [

    oci_core_default_security_list.oke_vcn.id,

    oci_core_security_list.workerlb.id

  ]

}
```

oke_worker_seclist.tf

```
resource "oci_core_security_list" "workerlb" {
  display_name   = "${var.vcn_name}-workerlb"
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_lb_ingress_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}
resource "oci_core_security_list" "worker" {
  display_name   = "${var.vcn_name}-worker"
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_ingress_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
```

```
    }
  }
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.worker_ingress_udp_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "17"
      udp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}
```

oke_vcn.tf

```
resource "oci_core_vcn" "oke_vcn" {
  cidr_block     = var.vcn_cidr
  dns_label      = var.vcn_name
  compartment_id = var.compartment_id
  display_name   = "${var.vcn_name}-vcn"
}
resource "oci_core_nat_gateway" "vcn_ngs" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "VCN nat g6s"
}
resource "oci_core_internet_gateway" "vcn_igs" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "VCN i6t g6s"
  enabled      = true
}
```

```
resource "oci_core_default_route_table" "private" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_route_table_id
  display_name               = "Default - private"
  route_rules {
    destination       = "0.0.0.0/0"
    destination_type  = "CIDR_BLOCK"
    network_entity_id = oci_core_nat_gateway.vcn_ngs.id
  }
}
resource "oci_core_route_table" "public" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "public"
  route_rules {
    destination       = "0.0.0.0/0"
    destination_type  = "CIDR_BLOCK"
    network_entity_id = oci_core_internet_gateway.vcn_igs.id
  }
}
```

oke_kmi_subnet.tf

```
resource "oci_core_subnet" "kmi" {
  cidr_block                 = var.kmi_cidr
  compartment_id             = var.compartment_id
  display_name               = "control-plane"
  dns_label                  = "kmi"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  prohibit_public_ip_on_vnic = true
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmi.id
  ]
}
resource "oci_core_subnet" "kmi_lb" {
  cidr_block                 = var.kmilb_cidr
```

```
  compartment_id             = var.compartment_id
  dns_label                  = "kmilb"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  display_name               = "control-plane-endpoint"
  prohibit_public_ip_on_vnic = false
  route_table_id             = oci_core_route_table.public.id
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmilb.id
  ]
}
```

oke_kmi_seclist.tf

```
resource "oci_core_subnet" "kmi" {
  cidr_block                 = var.kmi_cidr
  compartment_id             = var.compartment_id
  display_name               = "control-plane"
  dns_label                  = "kmi"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  prohibit_public_ip_on_vnic = true
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
    oci_core_security_list.kmi.id
  ]
}
resource "oci_core_subnet" "kmi_lb" {
  cidr_block                 = var.kmilb_cidr
  compartment_id             = var.compartment_id
  dns_label                  = "kmilb"
  vcn_id                     = oci_core_vcn.oke_vcn.id
  display_name               = "control-plane-endpoint"
  prohibit_public_ip_on_vnic = false
  route_table_id             = oci_core_route_table.public.id
  security_list_ids = [
    oci_core_default_security_list.oke_vcn.id,
```

```
    oci_core_security_list.kmilb.id
  ]
}
resource "oci_core_default_security_list" "oke_vcn" {
  manage_default_resource_id = oci_core_vcn.oke_vcn.default_security_list_id
  egress_security_rules {
    destination      = "0.0.0.0/0"
    destination_type = "CIDR_BLOCK"
    protocol         = "all"
  }
  dynamic "ingress_security_rules" {
    iterator = icmp_type
    for_each = [3, 8, 11]
    content {
      # ping from VCN; unreachable/TTL from anywhere
      source      = (icmp_type.value == "8" ? var.vcn_cidr : "0.0.0.0/0")
      source_type = "CIDR_BLOCK"
      protocol    = "1"
      icmp_options {
        type = icmp_type.value
      }
    }
  }
}
resource "oci_core_security_list" "kmilb" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "${var.vcn_name}-kmilb"
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_lb_ingress_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
```

```
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
}
resource "oci_core_security_list" "kmi" {
  compartment_id = var.compartment_id
  vcn_id         = oci_core_vcn.oke_vcn.id
  display_name = "${var.vcn_name}-kmi"
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_ingress_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "6"
      tcp_options {
        min = port.value.port_min
        max = port.value.port_max
      }
    }
  }
  dynamic "ingress_security_rules" {
    iterator = port
    for_each = local.kmi_ingress_udp_rules
    content {
      source      = port.value.source
      source_type = "CIDR_BLOCK"
      protocol    = "17"
      udp_options {
        min = port.value.port_min
```

```
        max = port.value.port_max
      }
    }
  }
}
```

variables.tf

```
variable "ClusterName" {
default = "Your-OKE-Cluster-Name" #NOTE: Enter your cluster name.
}
variable "public_key_oci" {
default = "/root/.oci/your-ssh-public-key.pub" #NOTE: Replace with your public key.
}
variable "kubernetes_version" {
   default = "v1.28.3"
}
variable "node_pool_size" {
  default = 3 #NOTE: Adjust to the size of your OKE node pool.
}
variable "Shape" {
 default = "VM.PCAStandard.E5.Flex"
}
variable "oci_config_file_profile" {
  type    = string
  default = "DEFAULT"
}
variable "tenancy_ocid" {
  description = "tenancy OCID"
  type        = string
  nullable    = false
}
variable "compartment_id" {
  description = "compartment OCID"
  type        = string
  nullable    = false
```

```
}
variable "vcn_name" {
  description = "VCN name"
  nullable    = false
}
variable "kube_client_cidr" {
  description = "CIDR of Kubernetes API clients"
  type        = string
  nullable    = false
}
variable "kubernetes_api_port" {
  description = "port used for kubernetes API"
  type        = string
  default     = "6443"
}
variable "worker_lb_ingress_rules" {
  description = "traffic allowed to worker load balancer"
  type = list(object({
    source   = string
    port_min = string
    port_max = string
  }))
  nullable = false
}
variable "worker_ingress_rules" {
  description = "traffic allowed directly to workers"
  type = list(object({
    source   = string
    port_min = string
    port_max = string
  }))
  nullable = true
}
```

```
# IP network addressing

variable "vcn_cidr" {

  default = "172.31.252.0/23"

}

variable "kmi_cidr" {

  description = "K8s control plane subnet CIDR"

  default     = "172.31.252.224/28"

}

# Subnet for KMI load balancer

variable "kmilb_cidr" {

  description = "K8s control plane LB subnet CIDR"

  default     = "172.31.252.240/28"

}

# Subnet for worker nodes, max 128 nodes

variable "worker_cidr" {

  description = "K8s worker subnet CIDR"

  default     = "172.31.253.0/24"

}

# Subnet for worker load balancer (for use by CCM)

variable "workerlb_cidr" {

  description = "K8s worker LB subnet CIDR"

  default     = "172.31.252.0/25"

}
```

image.tf

```
data "oci_core_images" "image1" {

    compartment_id           = var.compartment_id

    operating_system         = "OracleLinux"

    operating_system_version = "8"

    filter {

        name   = "display_name"

        values = ["uln-pca-Oracle-Linux8-OKE-1.28.3"]

        regex  = true

    }

}
```

oke_cluster.tf

```
resource "oci_containerengine_cluster" "Change to the name of your new OKE cluster." {

        compartment_id = var.compartment_id

        kubernetes_version = var.kubernetes_version

        name =  "${var.ClusterName}-1"

        vcn_id = "${oci_core_vcn.oke_vcn.id}"

        cluster_pod_network_options {

               cni_type = "FLANNEL_OVERLAY"

        }

        endpoint_config {

                 subnet_id = "${oci_core_subnet.kmi_lb.id}"

               is_public_ip_enabled = "true"

        }

        freeform_tags = {"Department"= "Finance"} #Replace with the correct tag to best fit your environment

        options {

                 kubernetes_network_config {

                       pods_cidr = "10.96.0.0/16"

                       services_cidr = "10.244.0.0/16"

                 }

                 service_lb_subnet_ids =  ["${oci_core_subnet.worker_lb.id}"]

       }

}
```

oke_pool.tf

```
resource "oci_containerengine_node_pool" "Replace with your OKE Node Pool name" {

  cluster_id         = oci_containerengine_cluster.SEOKECluster.id #Note: Replace with your OKE Cluster name.

  compartment_id     = var.compartment_id

  kubernetes_version = var.kubernetes_version

  name               = "SEOKENodePool" Replace with your OKE Node Pool name

  node_shape         = var.Shape

  freeform_tags = {"Department"= "Finance"} #Replace with the correct tag to best fit your environment

  node_source_details {

     image_id =  data.oci_core_images.image1.images[0]["id"]

     source_type = "IMAGE"

  }
```

```
  node_shape_config {
    ocpus = 1 #Replace with the amount of OCPUs to be allocated for the instances in your node pool.
    memory_in_gbs = 10 #Replace with the amount of memory to be allocated for the instances in your node pool.
  }
  node_config_details {
    size      = var.node_pool_size
    freeform_tags = {"Department"= "Finance"} #Replace with the correct tag to best fit your environment
    placement_configs {
      availability_domain = "AD-1"
      subnet_id           =  "${oci_core_subnet.worker.id}"
     }
  }
  ssh_public_key      = file(var.public_key_oci)
}
```

terraform.tfvars

```
oci_config_file_profile = "DEFAULT"
# tenancy ocid from the above profile
tenancy_ocid = "Replace with your tenancy OCID"
# compartment in which to build the OKE cluster
compartment_id = "Replace with your compartment ID"
# display-name for the OKE VCN
vcn_name = "Replace with the name of your new VCN"
# CIDR of clients that are allowed to contact k8s apiserver
kube_client_cidr = "10.0.0.0/8"
# security list rules for who is allowed to contact the worker load balancer
# (adjust for your applications)
worker_lb_ingress_rules = [
  {
    source   = "10.0.0.0/8"
    port_min = 80
    port_max = 80
  },
  {
    source   = "10.0.0.0/8"
```

```
    port_min = 443
    port_max = 443
  },
]
# security list rules for who is allowed to contact worker nodes directly
# This example allows 10/8 to contact the default nodeport range
worker_ingress_rules = [
  {
    source   = "10.0.0.0/8"
    port_min = 30000
    port_max = 32767
  },
```

ORACLE

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle

Author: Anderson Souza, Amardeep Dhillon