### Terraform EKS

### Prerequisites

* Terraform
* An AWS account with permissions to provision an ESK cluster
* TeamCity to build a pipeline

### Getting Started

* Setup Terraform script to spin up EKS cluster, including worker nodes
* Bitbucket Repository contain terraform script to deploy eks cluster.

Bitbucket repository - <https://bitbucket.ciena.com/projects/BP_DEVOPS/repos/eks-terraform-script/browse>

* Build a pipeline in Teamcity using terraform scripts

Teamcity : EKS Terraform Scripts <https://teamcity.ciena.com/blueplanet/admin/editProject.html?projectId=Platform_PlatformKubernetes_EksTerraformScript>

#Create a Build for EKS Terraform Script to Create a cluster

### Deploy the resources

* Build Step 1: Create-eks-cluster
* Name: create-eks-cluster
* Working directory: eks/environment/dev (add your Repository)
* Custom script: (Enter build script content):

-terraform init

-terraform apply -auto-approve

#*before running the build enable the build step*

* Run the build
* check the Build logs
* check if any errors in the build
* build is successful

Terraform has been successfully initialized!

* Check the EKS services on AWS console

### AWS Console

* go to EKS services
* check if cluster is in creating or in active state
* you can see the the cluster created successfully

Cluster name: bpc-dev-eks-cluster

Kubernetes version: 1.14

Status: Active

* the cluster is in active state you can see the Kubernetes Version, EKS Version, API server endpoint, Cluster IAM Role ARN, VPC, Subnets, Cluster security group
* go to EC2 you can see two nodes(instances) are in running status.
* Check the nodes are available on the cluster

$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-10-20-4-116.ec2.internal Ready <none> 11m v1.11.10-eks-17cd81

ip-10-20-5-166.ec2.internal Ready <none> 11m v1.11.10-eks-17cd81

### What resources are created

* VPC
* Internet Gateway (IGW)
* Public and Private Subnets
* Security Groups, Route Tables
* IAM roles, instance profiles and policies
* An EKS Cluster
* Autoscaling group and Launch Configuration
* Worker Nodes in a private Subnet
* The Config-Map required to register Nodes with EKS

### Highlights:

* EKS Cluster AWS managed Kubernetes cluster of master servers
* Auto-Scaling Group contains m5. large instances based on the latest EKS Amazon Linux AMI
* Associated VPC, Internet Gateway, Security Groups, and Subnets Operator managed networking resources for the EKS Cluster and worker node instances
* Associated IAM Roles and Policies Operator managed access resources for EKS and worker node instances
* Define a pool of worker nodes: create Auto-scaling Group (ASG) with launch configuration and provision nodes to attempt joining the cluster
* Configure the cluster to allow worker nodes to join

### Cleaning up

### # You can destroy this cluster entirely by running the build

### Build Step 2: delete-eks-cluster

* Name: delete-eks-cluster
* Working directory: eks/environment/dev
* Custom script: (Enter build script content):

- terraform destroy -auto-approve

\**before running the build enable the build step to destroy the cluster*

* Run the build and check the Build logs
* the build is successful

Cluster is destroyed