by ClearPoint

Connect - Continuous Delivery Platform

Built by ClearPoint <connect@clearpoint.co.nz> v1.0 2017-07-07

The repo for this Documentation can be found on Github: https://github.com/ClearPointNZ/connect

[Build Status] | *https://api.travis-ci.org/ClearPointNZ/connect.svg*

Introduction

Welcome to the Connect Continuous Delivery Platform.

Connect is a collection of open source projects put together to help share best practice approaches to Continuous Delivery (CD) using containerisation (via Docker), orchestrated and deployed by Kubernetes.

The libraries are designed to be used either standalone or together and are grouped into the following areas:

Repositories

Connect is made up of a number of repositories grouped into the following areas.

Acceptance Testing

Shared Libraries & Samples

Connect tile release

It defines how a Maven project is to be released to Central and what plugins must run (and pass) for this to happen

Connect sample apps

Sample app running in a CD pipeline using all the shared libraries and tools we've built.

Infrastructure as Code

Pipeline and Tools

Connect Jenkins Bootstrap

Bootstraps Jenkins and some plugins into a Kubernetes cluster.

TIP | See blog Part 2: Bootstrapping Jenkins in a Kubernetes cluster

Kubernetes

Tack

An open source Terraform module that helps us create a highly-available, redundant Kubernetes cluster on AWS.

TIP | See blog Part 1: Creating a Kubernetes Cluster on AWS

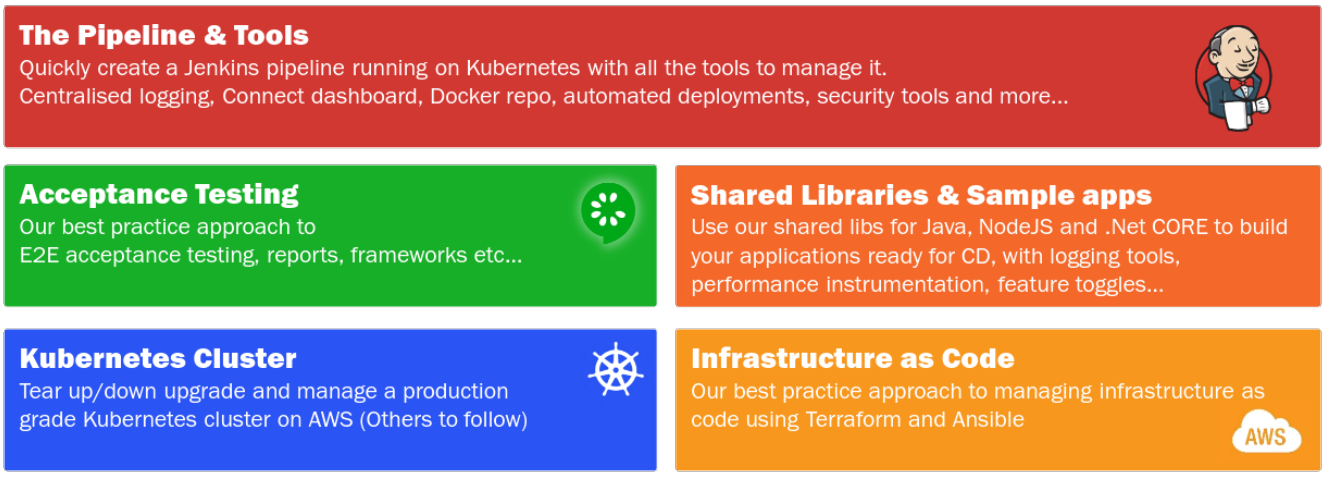
Blogs

We are putting together a series of blogs on Connect as a guide to building a Continuous Delivery (CD) pipeline on top of Kubernetes.

Introduction to Connect

Part 1: Creating a Kubernetes Cluster on AWS

Part 2: Bootstrapping Jenkins in a Kubernetes cluster



Kubernetes

Building clusters on AWS

Follow along with our blog Part 1: Creating a Kubernetes Cluster on AWS to learn how to easily create yourself a remote cluster on AWS using Connect with Tack.

You can also use kops to create an AWS cluster.

Building a local cluster

We prefer using Minikube to spin up a local cluster for testing purposes.

Bootstrapping Jenkins

You can follow our walkthrough in our blog or run through the instructions below. The official documentation for Connect will also step you through installation.

Prerequisites

Before you bootstrap Jenkins into your cluster you'll need to set one up. You can either create a local cluster with Minikube or create an AWS cluster as described on this page or another cluster from the provider of your choice. For Minikube you can execute `minikube start --kubernetes-version=v1.7.0` and wait until it's all setup.

Running Jenkins

SSH Keys

Before you provision anything Jenkins needs SSH keys so that it can checkout code from GitHub. Generate SSH keys in the repository folder using `ssh-keygen -t rsa -b 4096 -C "your_email@example.com" -N ""` and add them to the GitHub account that you would like Jenkins to use.

Cluster setup

For a local setup, just run `run-minikube.sh`. Here's what will happen:

- The namespace `jenkins` will be created and set to the current `kubectl` context
- Keys will be set to the Kubernetes secret storage
- The configuration map for the Kubernetes master URL will be created
- The Kubernetes deployment will be applied

Init Containers

In order to get the configuration to the Jenkins node there are some initContainers that are run in the pod before starting the main container.

To join all the results and store them across the containers in the pod, the volume `ref-volume` is used - you can see that it's mounted as `${JENKINS_HOME}/ref` folder to every init container.

Checkout container sets up SSH keys and checks out the repository from GitHub.

Install plugins container installs plugins listed in the `plugins` file to the `${JENKINS_HOME}/ref` folder for Jenkins to pick them up on startup

Override config container updates the Jenkins host and Kubernetes master host values in the `config.xml.override` using `scripts/hack-jenkins-env.sh`. Then it copies the updated config to the `${JENKINS_HOME}/ref` folder so it's picked up by Jenkins when it starts up. It also copies `scripts/security.groovy` to `${JENKINS_HOME}/ref/init.groovy.d/` and when Jenkins starts up it

executes all Groovy scripts from that folder. `security.groovy` sets up the `admin` user and initial password and Jenkins API token.

Copy jobs container copies predefined job `config.xml` to `${JENKINS_HOME}/ref` folder.

Jenkins container

As the Jenkins container starts it runs scripts and copies plugins as described in Jenkins CI Docker image docs. In our setup we avoid running the Jenkins install wizard by setting the environment variable `JAVA_OPTS=-Djenkins.install.runSetupWizard=false`.

To demonstrate that the environment is working and creating slaves to execute a job, the main container has a postStart hook that runs `scripts/wait-for-jenkins-and-run-job.sh` that waits until the Jenkins API is available and then triggers a job `test` that is restricted to run on a slave node. To authenticate, it uses the admin API token that is generated by the `scripts/security.groovy`.

Getting Help

ClearPoint Organisation on Github

http://github.com/ClearPointNZ

Source repositories

- Main bootstrap and documentation https://github.com/ClearPointNZ/connect

Support

For enterprise support, please contact ClearPoint or email connect@clearpoint.co.nz

License