

Kubernetes Cluster Setup

|  |  |
| --- | --- |
| **Author** |  |
| **Approved By** |  |
| **Application Version** |  |
| **Date of Issue** |  |
| **Release Note Number** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document Control Section** | | | | |
|  | | | | |
| **Revision History:** | | | | |
| **Document version** | **Pages** | **Action** | **Date of revision** | **By** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | | | | |

Table of Content

[1.0 Confidentiality / Legal notification 3](#_Toc331583772)

[2.0 Introduction 5](#_Toc331583773)

2.1 why you need kubernetes ……………………………………………………………………………….6

[**3.0 KUBERNETES INFRA PROVISION 5**](#_Toc331583774)

**3.1 RESERVING STATIC INTERNAL IP ADDRESS ………………………………………………………….……… 5**

**3.2 PROVISION GCP INSTANCE …………………………………………………………………………………………. 7**

**4.0 KUBERNETES CLUSTER SETUP ……………….………………………………………………………………………………… 8**

**4.1 PREREQUITES FOR CLUSTER SETUP ……………………………………………………………………………… 9**

**4.2 RUN ANSIBLE PLAYBOOK ……….…………………………………………………………………………………… 9**

**5.0 CLUSTER VALIDATION ……………………………………………………………………………………………………………… 10**

**5.1 KUBECTL INSTALL IN CLIENT MACHINE ………………………………………………………………………… 11**

**5.2 Configure kubectl in client machine …………………………………………………………….……………... 12**

**5.3 Deploy workload and validate ……………………………………………………………………………………. 13**

1. Confidentiality / Legal notification

Confidential. Copyright © AI Enterprise., 2018. All rights reserved. This document, including the information contained

herein, is restricted, confidential and proprietary to AI Enterprise, and is to be used only by and disclosed only to those

within AI Enterprise with a need-to-know. DO NOT COPY OR FORWARD INTERNALLY OR RELEASE outside AI Enterprise

without authorization in writing by a Sr. Vice President, Principal, or Director-level manager or a direct designee

thereof who has responsibility for the information contained herein.

1. Introduction

This document describes about Kubernetes infra provision using Terraform and setup Kubernetes Cluster using Ansible playbook on Google Cloud.

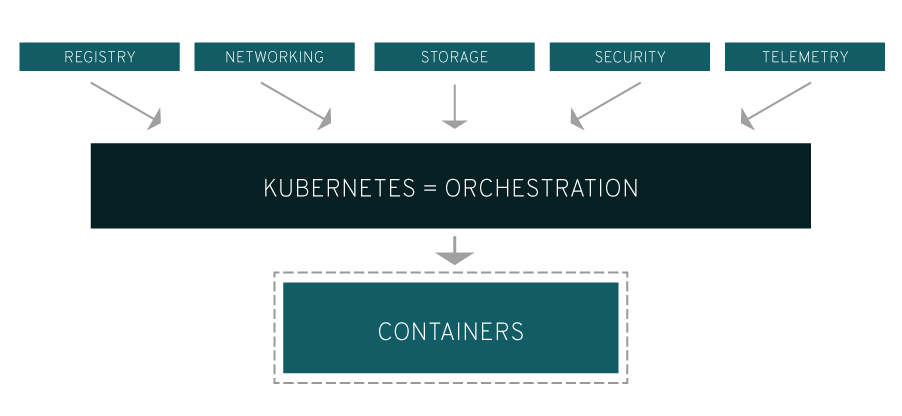
Kubernetes, or k8s (k, 8 characters, s...get it?), or “kube” if you’re into brevity, is an open source platform that automates Linux container operations. It eliminates many of the manual processes involved in deploying and scaling containerized applications. In other words, you can cluster together groups of hosts running Linux containers, and Kubernetes helps you easily and efficiently manage those clusters. These clusters can span hosts across public, private, or hybrid clouds. For this reason, Kubernetes is an ideal platform for hosting cloud-native applications that require rapid scaling, like real-time data streaming through Apache Kafka.

Kubernetes was originally developed and designed by engineers at Google. Google was one of the early contributors to Linux container technology and has talked publicly about how everything at Google runs in containers. (This is the technology behind Google’s cloud services.) Google generates more than 2 billion container deployments a week—all powered by an internal platform: Borg. Borg was the predecessor to Kubernetes and the lessons learned from developing Borg over the years became the primary influence behind much of the Kubernetes technology.

2.1 Why do you need Kubernetes?

Real production apps span multiple containers. Those containers must be deployed across multiple server hosts. [Security for containers](https://www.redhat.com/en/topics/security/container-security) is multilayered and can be complicated. That's where Kubernetes can help. Kubernetes gives you the orchestration and management capabilities required to deploy containers, at scale, for these workloads. Kubernetes orchestration allows you to build application services that span multiple containers, schedule those containers across a cluster, scale those containers, and manage the health of those containers over time. With Kubernetes you can take [real steps towards better IT security](https://www.redhat.com/en/topics/security).

Kubernetes also needs to integrate with networking, [storage](https://www.redhat.com/en/topics/data-storage), security, telemetry and other services to provide a comprehensive container infrastructure.

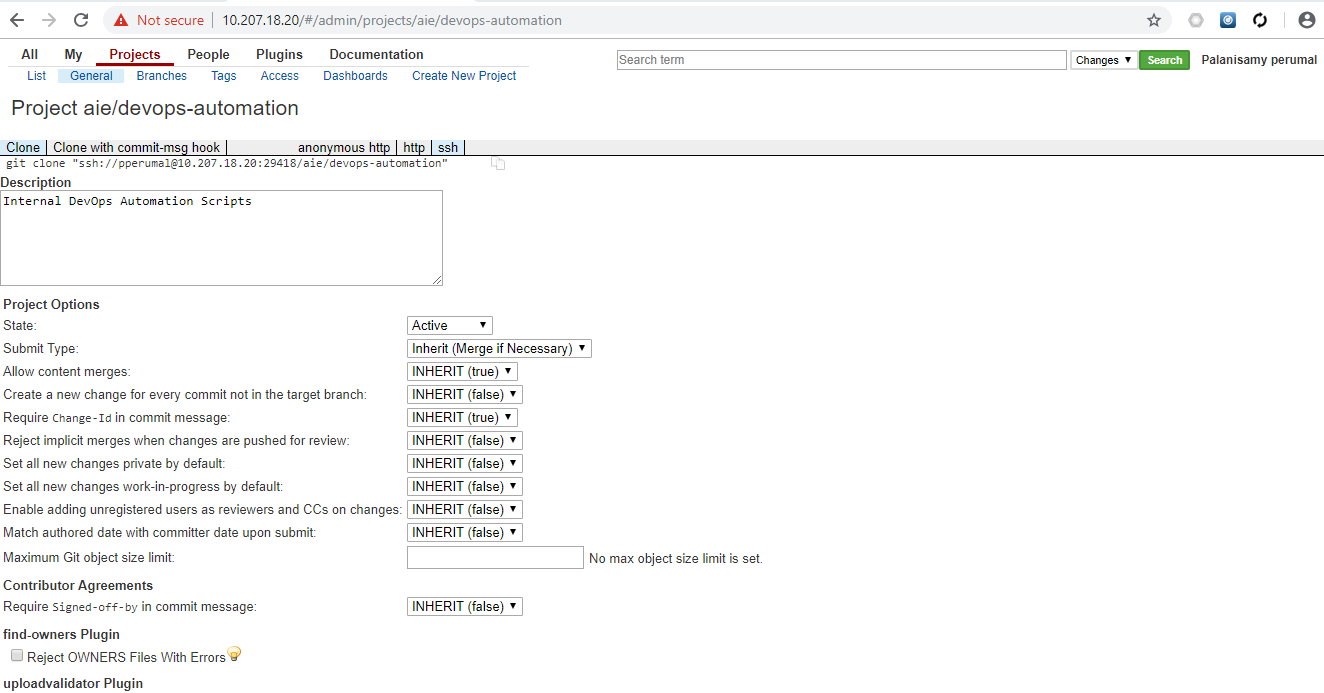


Of course, this depends on how you’re using containers in your environment. A rudimentary application of Linux containers treats them as efficient, fast virtual machines. Once you scale this to a production environment and multiple applications, it's clear that you need multiple, colocated containers working together to deliver the individual services. This significantly multiplies the number of containers in your environment and as those containers accumulate, the complexity also grows.

Kubernetes fixes a lot of common problems with container proliferation—sorting containers together into a ”pod.” Pods add a layer of abstraction to grouped containers, which helps you schedule workloads and provide necessary services—like networking and storage—to those containers. Other parts of Kubernetes help you load balance across these pods and ensure you have the right number of containers running to support your workloads.

1. Kubernetes Infra provision

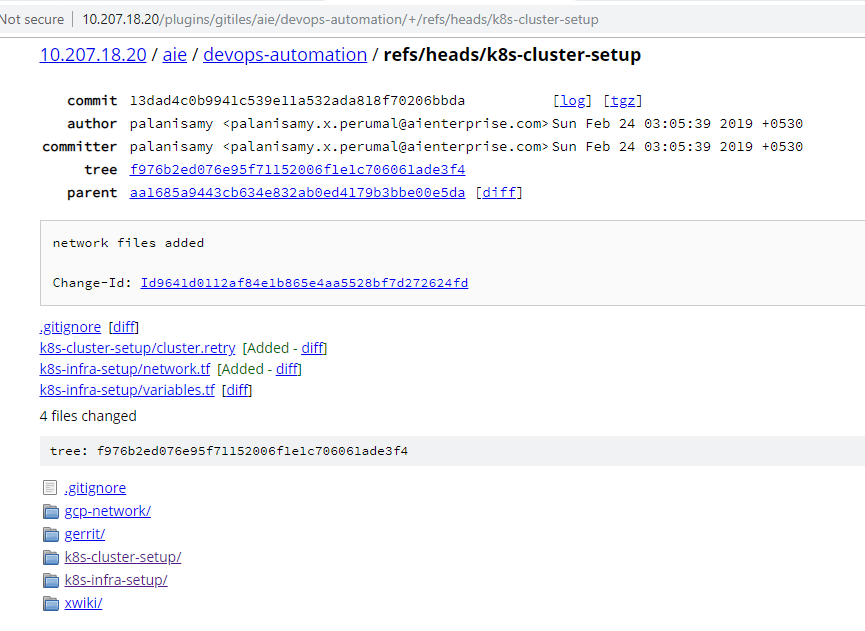
We will provision all require network, instances on Google Cloud using terraform scripts. These terraform scripts you can clone from AIE repository.



All the latest code is under in k8s-cluster-setup branch.

* 1. Reservising Static inetanl IP address

Kubernetes master and worker needs static internal IP address. These Terrafom scripts are in gcp-network directory



Sample terraform scripts file.



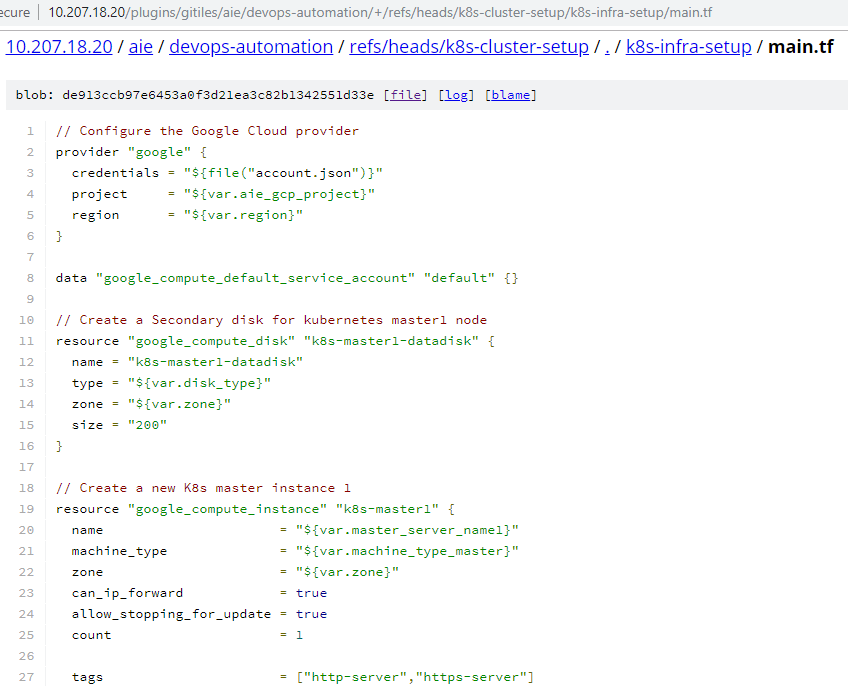
For example, we are reserving following ip address from ddh-aie-lle-central-prv01 subnet.

* 10.207.18.26
* 10.207.18.27
* 10.207.18.28
* 10.207.18.29
* 10.207.18.30
  1. Provision GCP instance

We can provision GCP instance using following sample terraform codes from k8s-infra-setup directory. This will create 2 master and 1 worker node along with 200GB secondary drive. We can adjust this number of nodes from variable.tf files as per our requirements.

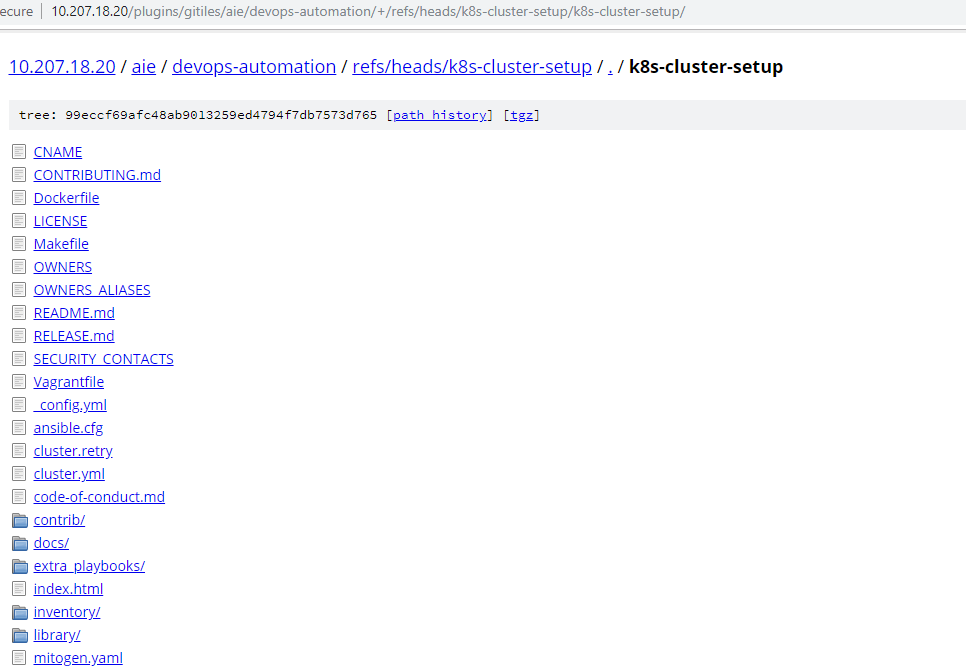
Sample code snippet.





1. KUBERNETES CLUSTER SETUP

Once require instance provisioned then we can setup Kubernetes using Ansible Playbook. You can refer all UpToDate code from k8s-cluster-setup directory.



* 1. PREREQUITES FOR CLUSTER SETUP

The following prerequires should be setup in the local where Ansible is installed and invoke scripts.

* Ansible v2.7.8 (or newer) and python-netaddr is installed on the machine that will run Ansible commands
* Jinja 2.9 (or newer) is required to run the Ansible Playbooks
* The target servers must have access to the Internet in order to pull docker images. Otherwise, additional configuration is required (See Offline Environment)
* The target servers are configured to allow IPv4 forwarding.
* Your ssh key must be copied to all the server’s part of your inventory.
* The firewalls are not managed, you'll need to implement your own rules the way you used to. in order to avoid any issue during deployment you should disable your firewall.
* If AIEnterprise is ran from non-root user account, correct privilege escalation method should be configured in the target servers. Then the ansible become flag or command parameters --become or -b should be specified.
  1. RUN ANSIBLE PLAYBOOK

Execute following commands in the local box before deploying the cluster.

# Install dependencies from ``requirements.txt``

sudo pip install -r requirements.txt

# Copy ``inventory/sample`` as ``inventory/mycluster``

cp -rfp inventory/sample inventory/mycluster

# Update Ansible inventory file with inventory builder

declare -a IPS=(10.207.18.26 10.207.18.27 10.207.18.28)

CONFIG\_FILE=inventory/mycluster/hosts.yml python3 contrib/inventory\_builder/inventory.py ${IPS[@]}

# Review and change parameters under ``inventory/mycluster/group\_vars``

cat inventory/mycluster/group\_vars/all/all.yml

cat inventory/mycluster/group\_vars/k8s-cluster/k8s-cluster.yml

# Deploy with Ansible Playbook - run the playbook as root

# The option `-b` is required, as for example writing SSL keys in /etc/,

# installing packages and interacting with various systemd daemons.

# Without -b the playbook will fail to run!

ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user=root cluster.yml

1. CLUSTER VALIDATION

The Kubernetes command-line tool, kubectl, allows you to run commands against Kubernetes clusters. You can use kubectl to deploy applications, inspect and manage cluster resources, and view logs.

* 1. Install Kubectl in client machine

For Windows device

* Download the latest release v1.14.0

curl -LO <https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/windows/amd64/kubectl.exe>

* Add the binary in to your PATH.
* Test to ensure the version you installed is up-to-date:

kubectl version

**For Linux devices**

* Download the latest release with the command:

curl -LO <https://storage.googleapis.com/kubernetes-release/release/$(curl> -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl>

To download a specific version, replace the $(curl -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt>) portion of the command with the specific version.

For example, to download version v1.14.0 on Linux, type:

curl -LO <https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/linux/amd64/kubectl>

* Make the kubectl binary executable.

chmod +x ./kubectl

* Move the binary in to your PATH.

sudo mv ./kubectl /usr/local/bin/kubectl

* Test to ensure the version you installed is up-to-date:

kubectl version

For Mac devices

* Download the latest release:

curl -LO <https://storage.googleapis.com/kubernetes-release/release/$(curl> -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/darwin/amd64/kubectl>

To download a specific version, replace the $(curl -s <https://storage.googleapis.com/kubernetes-release/release/stable.txt>) portion of the command with the specific version.

* For example, to download version v1.14.0 on macOS, type:

curl -LO <https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/darwin/amd64/kubectl>

* Make the kubectl binary executable.

chmod +x ./kubectl

* Move the binary in to your PATH.

sudo mv ./kubectl /usr/local/bin/kubectl

* Test to ensure the version you installed is up-to-date:

kubectl version

* 1. Configure kubectl in client machine

For kubectl to find and access a Kubernetes cluster, it needs a kubeconfig file, which is created automatically and available when you create a cluster. You can copy this config file from /etc/kubernetes/config. Copy this config file and store at ~/.kube/config where kubectl installed.

Check that kubectl is properly configured by getting the cluster state:

**kubectl cluster-info**

* 1. Deploy workload and validate

Using kubectl tool, we can do deploy from localbox to remote Kubernetes cluster.

kubectl create deployment nginx --image=nginx