Kubernetes Structure Diagram

Feasibility Demo - Project Artefact

Finlay Reid

CMP400: Honours Project

BSc Ethical Hacking Year 4

2021/22

Note that the Information contained in this document is for educational purposes.

.

Contents

.

1.1 INTRODUCTION

THIS DOCUMENT DEMONSTRATES THE CORE COMPONENTS WITHIN A KUBERNETES ENVIRONMENT INCLUDING MY OWN. RELEVANT TECHNOLOGIES OF KUBERNETES ARE EXPLAINED, FIVE DIAGRAMS ARE PRESENTED INCLUDING A BREAKDOWN OF THE WAY POD TO POD NETWORKING FUNCTIONS AND ONE THAT DISPLAYS A WORKER NODE INTERNAL COMPONENTS. THE FINAL SECTION DETAILS THE REFERENCES WHICH ASSISTED ME IN CONSTRUCTING THE DIAGRAMS AND EXPLAINING THE KUBERNETES MECHANISMS.

.

2 KUBERNETES DIAGRAMS

2.1 KUBERNETES STRUCTURE DIAGRAM

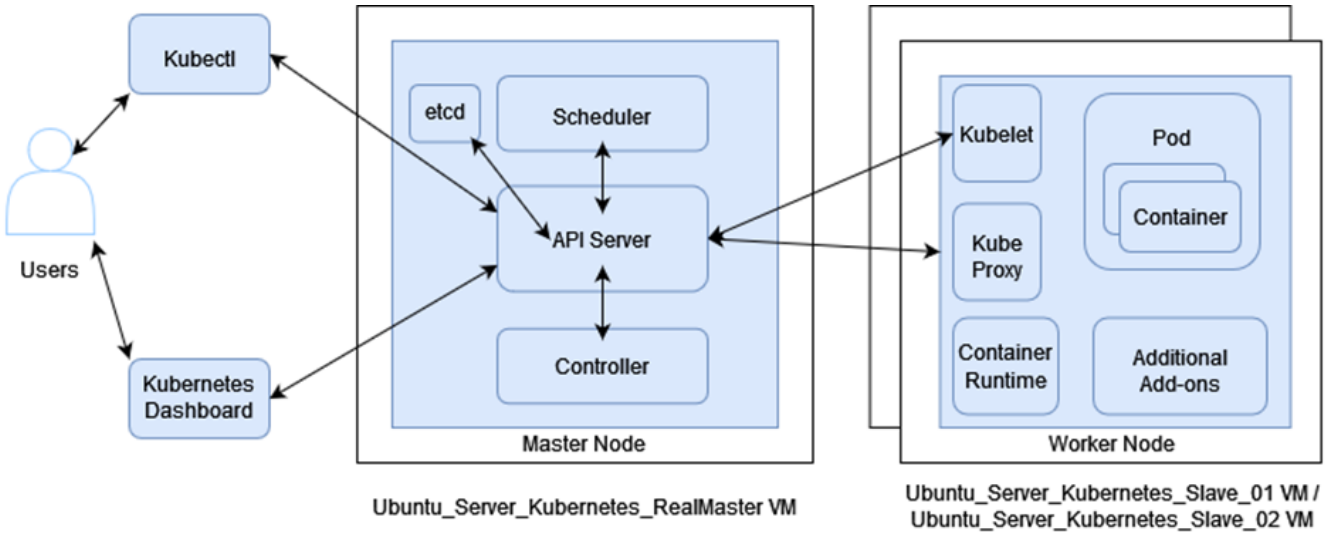

Figure 1 - Main structure diagram

Breakdown

Virtual machines

Ubuntu_Server_Kubernetes_Slave_01/02 VM: A pair of virtual machines running ubuntu server 20.04 using the hypervisor VMware workstation. Both were configured properly accounting for best practices when Kubernetes was installed and the cluster was hosted. Used to host the worker nodes.

Ubuntu_Server_Kubernetes_RealMaster VM: A singular virtual machine operating ubuntu server 20.04 using the hypervisor VMware workstation. Both were configured properly accounting for best practices when Kubernetes was installed and the cluster was hosted. Used to host master node.

Nodes

Worker Node: Occasionally referred to as slaves, these servers run/operate the containers and assist with other Kubernetes mechanisms. Every Kubernetes cluster utilizes at least one of these technologies and helps host the pods.

Master Node: This server orchestrates the group of worker nodes through the use of many processes including the API server, Controller Manager, Scheduler and etcd. Again like the worker node, all clusters require a Master node to function. These nodes also operate the control plane and are the main point of contact with users attempting to configure their cluster.

.

Master Node

Scheduler(Master node): Looks for pods that have recently been created that have no pods and determines their destination based on several factors.

API server(Master node): Component used to facilitate external interaction between Kubernetes and the user. Has the ability to run numerous instances and is essentially the face of the software.

Controller (Master node): This mechanism operates the controller processes, gauges the state of the cluster, and attempts to fix any faulty nodes. Uses many controllers including node controller, job controller, endpoints controller, and service account controller.

Etcd(Master node): Used to store key-value storage, such as the condition of the cluster. Holds the configuration, and allows for the recovery of the cluster through snapshots.

Other

Kubectl: Command line tool that allows control of Kubernetes clusters. Has its specific syntax and is a key component of Kubernetes. Used to orchestrate the cluster.

Kubernetes Dashboard: Web portal that permits users to deploy containers to Kubernetes clusters. Provides users with an overview of their cluster and can be used to troubleshoot. Utilizes tokens to ensure the cluster remains secure.

2.2 WORKER NODE DETAILED DIAGRAM

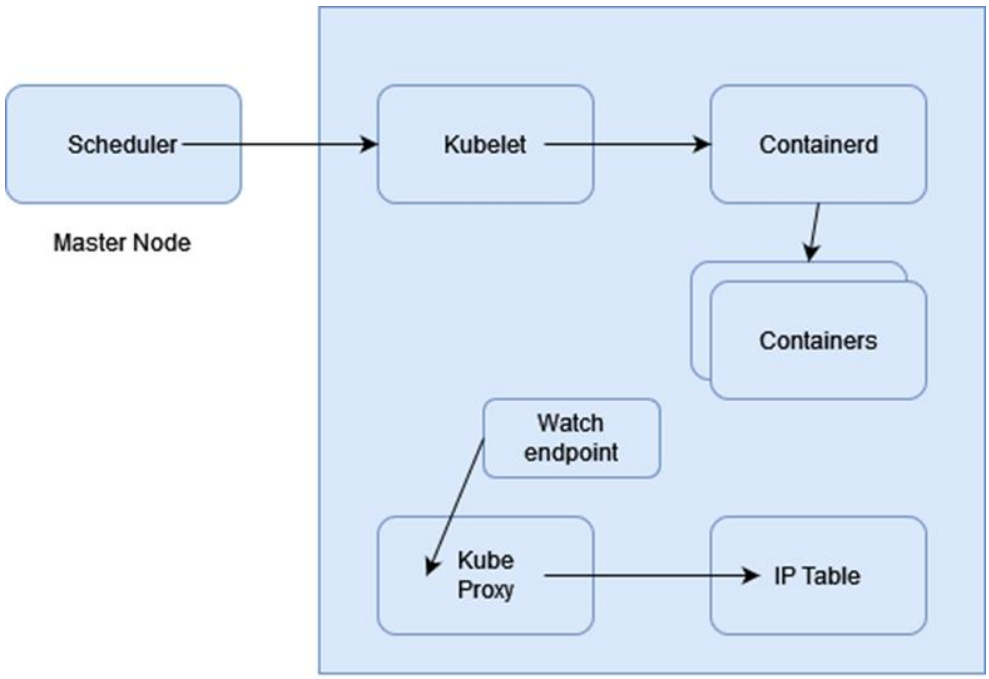

Figure 2 - Worker node diagram

.

Breakdown

Kubelet: A process that operates within the individual nodes. Responsible for ensuring containers are running in a pod. Registers the node with the API server that is located within the Master node and the technology functions through a YAML or JSON file.

Containerd: The container runtime that was selected due to dockers deprecation, the software essentially assists in selecting container images, then loads the selected images to allow the container to function. Responsible for the processes used by containers and the management of container resources.

Containers: Two containers are currently being run through the VM's using the ubuntu server operating system. As previously mentioned containers are essentially lightweight VM's that can run many services, processes, and applications, through operating system virtualization.

Watch Endpoint: Kubernetes uses its own terms that are distinguishable from HTTPS verbs such as GET, POST, etc. When attempting to return data, a watch request is used. So this is the end of the watch request.

Kube Proxy: The Kube Proxy is a network proxy used by Kubernetes and it has a few main functions including looking after the network rules on specific nodes, this allows data to travel inside and externally to the cluster. This functionality works through OS packet filtering if one is present.

Iptables: Kubernetes utilizes network tables which permit the software to control links between specific pods/nodes. Allowing users to customise the rules of the network and redirect traffic. Kubernetes maintains much of the network making it hard to monitor network traffic, giving security professionals a tough time when attempting to secure a Kubernetes cluster.

.

2.3 API SERVER/CONTROL MANAGER DIAGRAM

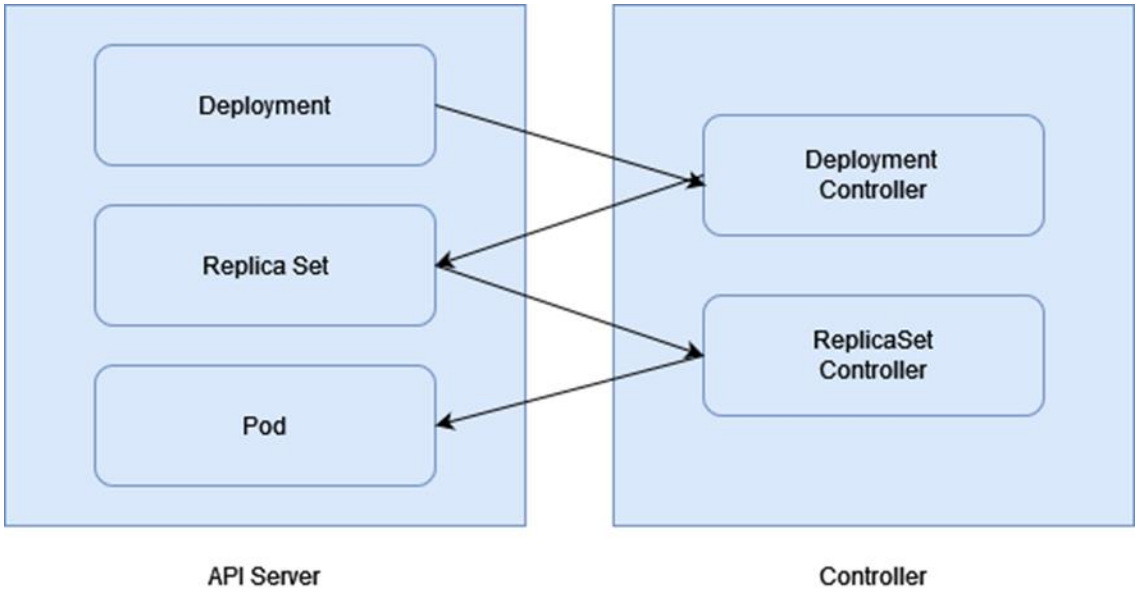

Figure 3 -API/control diagram

Breakdown

API Server

Pod: A pod is a group of containers with the same configuration, storage, and network. Pods have their own individual IP addresses for each address family and pods can be configured to have shared storage volumes, authorizing containers to share files.

Replica Set: The replica set component within the API server is responsible for making sure that a of number pod replicas are in working order. Essentially the replica set ensures that the pod or set of pods are running smoothly and available for use. Furthermore, this component allows users to set the specific number of available pods.

Deployment: The deployment object works in unison with the deployment controller to allow users to create new replica sets. It also assists the pods when the Kubernetes version is updated.

Controller manager

Deployment controller:

The Deployment controller supplies both the pods and replica sets with updates. Working in conjunction with the deployment object.

ReplicaSet Controller:

Helps manage the lifecycle of a pod, working in conjunction with the replica object.

.

2.4 BUST A KUBE DIAGRAM

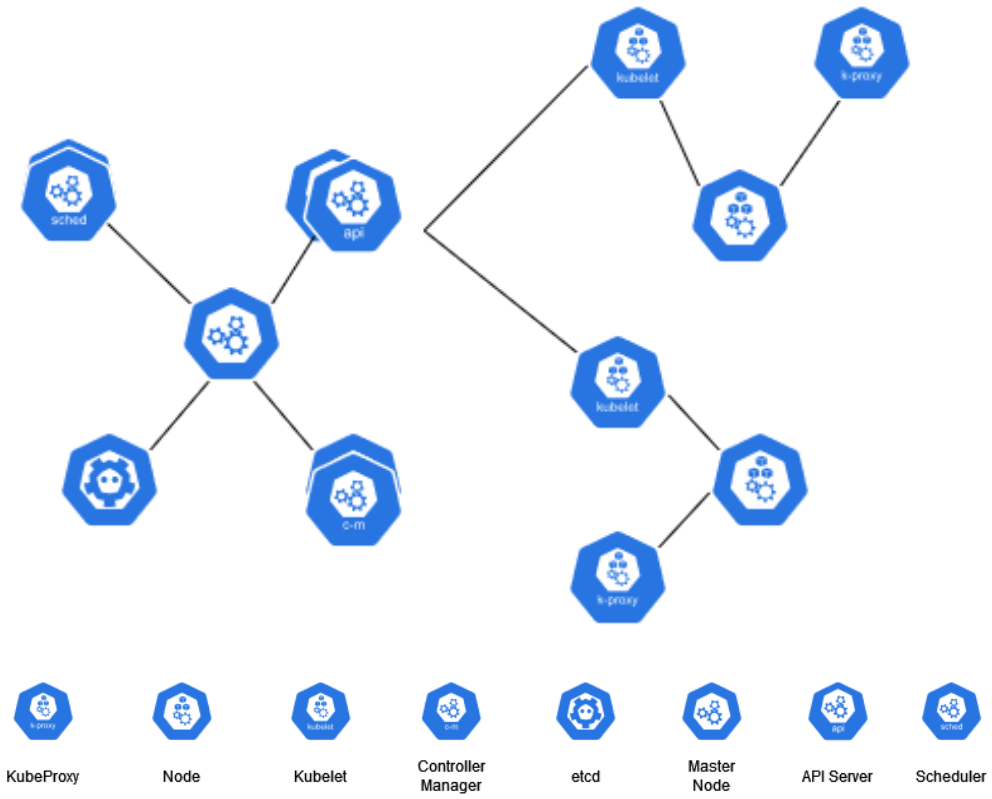

Figure 4 - Bust a kube diagram

.

2.5 KUBERNETES NETWORK DIAGRAM

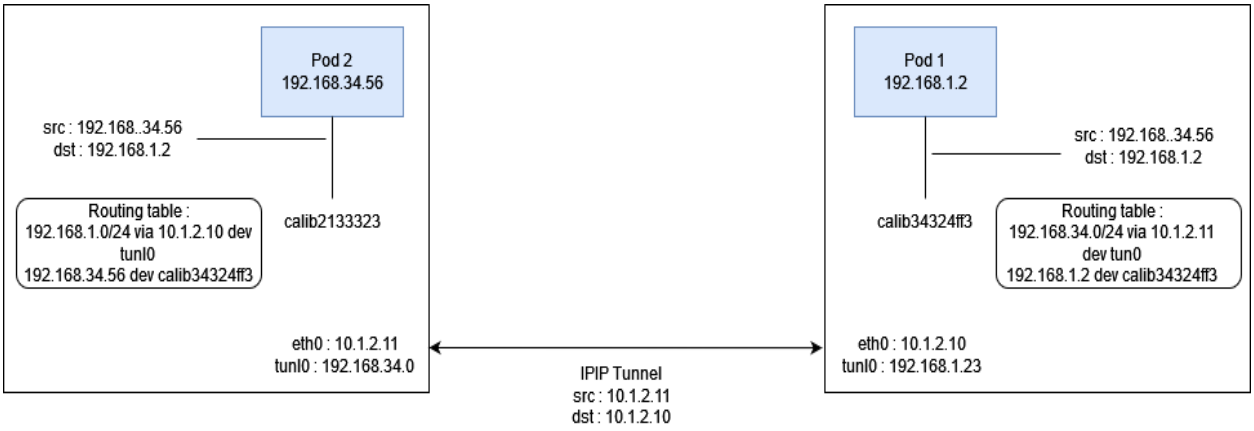

Figure 5 - Network diagram

Breakdown

Figure 5 demonstrates what steps are taken when two pods on different nodes want to communicate. Like the clean Kubernetes cluster, this implementation uses the Calcio network plugin. At the start of a cluster lifecycle when a container is first created it falls on the Calcio network plugin to ensure the node has been assigned Ip addresses and everything is in working order. The next step in the process involves the pods attempting to communicate, as pod1 will initially send a network packet to either pod1s IP or service IP. The Kube proxy that was previously discussed is responsible for handling the request if the service IP is used by translating the address. A routing table on that specific node controls where the packet's destination is, in **Figure 5** the packet is forwarded to the IPIP tunnel and encased with the IPIP tunnel header. After reaching the destination node the encasing will be detached. Finally, the routing table is then responsible for forwarding the packets to their correct destination i.e. pod 1, pod2.

.

REFERENCES

Artem, L., Tetiana, B., Larysa, M. and Vira, V., 2020. Eliminating privilage escalation to root in containers running on Kubernetes. *Scientific and practical cyber security journal*.

Gupta, G., 2021. *Components of Kubernetes Architecture*. [online] Medium. Available at: <https://kumargaurav1247.medium.com/components-of-kubernetes-architecture-6feea4d5c712> [Accessed 2 December 2021].

home, P., support, c. and </form>, 2021. *Understanding Kubernetes Architecture with Diagrams*. [online] Knowledge Base by phoenixNAP. Available at: <https://phoenixnap.com/kb/understanding-kubernetes-architecture-diagrams> [Accessed 2 December 2021].

Kubernetes. 2021. *Kubernetes Components*. [online] Available at: <https://kubernetes.io/docs/concepts/overview/components/> [Accessed 2 December 2021].

Lam, V., 2021. *Basic Components of Kubernetes Architecture*. [online] appvia. Available at: <https://www.appvia.io/blog/components-of-kubernetes-architecture> [Accessed 2 December 2021].