Provisioning Kubernetes Cluster On Developer Laptops
Author: Arvind Kumar Pulijala

1. Project Overview

The project goal is to deploy a production-like kubernetes cluster on Developer's local machines for developers to practice development and deployment of microservices on their local machines. The solution has been developed in such a way that by just running one script an entire cluster will be provisioned and running another one, cleanly destroys the cluster and other artefacts related to managing the cluster and restores the machine in a state it was before deploying the cluster.

It also builds a realistic Environment which is like Production with only external load balancer missing. The missing features can also be incorporated, but are not in scope for this Sprint.

1.1 Advantages of Local Kubernetes Clusters compared to using a cloud-based environment

a) **Direct Kubernetes Access:** At first, the developers are admins of their own clusters and can thus freely configure everything as they need it and they can play around with all available Kubernetes features, without any fear of breaking shared resources.

b) **Perfect Isolation:** Due to the use of individual clusters on their own computers, the developers are perfectly isolated from each other. They could even use the local clusters without an internet connection and practice their skills and become better in developing and deploying microservices. This has the advantage that the different engineers cannot interfere with each other even if someone messes up something. In such cases, it is also easily possible to start from scratch with a clean system. The solution provides for clean destruction and recreation of the cluster.Additionally, strict isolation enforces a clear responsibility. If something breaks, there is only one person to blame.

c) **No Computing Cost:** Finally, local clusters run on the existing machines of the developers and therefore, there will be no extra cost associated with their use (compared to sometimes costly cloud resources). In some companies, this has the additional benefit that no complicated budget approvals are necessary to work with local Kubernetes solutions. Don't know what the procedure at Zeppelin is.

Kubernetes Knowledge Necessary: The drawback of using local clusters during development is that the developers need to have some Kubernetes knowledge. While it would be certainly great if every engineer would be a Kubernetes expert, the complexity of Kubernetes makes this quite unrealistic and undesirable given the effort to master it. **But for developers using this solution no Kubernetes admin knowledge is required as all admin tasks have been automated using a combination of Ansible and Bash Scripts.**

Use Cases for Local Kubernetes Clusters:

First Kubernetes/Microservices Experiments: The ability to run local clusters on already available computers without further budget approvals and no access to a cloud environment makes them the typical environment for first experiments with Kubernetes and microservices

Small Teams : They can also be a good solution for smaller teams with standardized hardware, so admins with more Kubernetes experience can either set up the local clusters on every machine themselves. Using this solution, all that needs to be done is checkout a git repo and issue one command to provision a cluster and another to decommission the cluster and start building and deploying Microservices.

1.2 Evaluating Other Solutions.

Product	**Disadvantages**
Minikube	While minikube has made improvements, scheduling features required to develop a High performance Microservice seems to be missing. Less realistic compared to Cluster with VMs on the local machine
Docker Desktop	No Longer free and licensing required

	and suffers from the same disadvantages as minikube.

Steps to use the solution to provision/decommission Kubernetes cluster.

Currently the solution has been tested and working on Ubuntu Desktops. Small effort is required to port the solution to Mac Machines. I am

10:39

1) Checkout the code from git repo
   https://github.com/apulijala/kubernetes-devmachines.git

2) Change to the current directory

   cd kubernetes-devmachines/

3) Provision the cluster.
   a) chmod +x deploy_kub_cluster.sh
   b) Invoke the script ./deploy_kub_cluster.sh

Select 1 when the prompt comes as **Available bridge network interfaces.**

==> master: Available bridged network interfaces:

1) wlp64s0

2) enp59s0f1

3) virbr2

master: Which interface should the network bridge to? 1

==> kubworkerone: Available bridged network interfaces:

1) wlp64s0

2) enp59s0f1

3) virbr2

==> kubworkertwo: which network to bridge to.

==> kubworkertwo: Available bridged network interfaces:

1) wlp64s0

2) enp59s0f1

kubworkertwo: Which interface should the network bridge to? 1

I will be looking to automate this step too, so manual selection won't be necessary.

It will take about 20 mins to provision the cluster completely for the first time. Subsequently you can start and stop the VMs when required.

4) Testing the cluster.

   kubectl get nodes Should return 3 nodes 1 Master and 2 workers in ready state.

5) There are some sample deployments and services which deploys a nginx to kubernetes cluster. I will also be looking to put some example Helm charts and maybe even a Hello World Spring boot example with Helm Chart.

   cd testcluster/

   kubectl apply -f mynginx.yml

kubectl get deployments.apps

```
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
mynginx  3/3     3            3           7m18s
```

Deploy the **NodePort** Service

kubectl apply -f nginxsvc.yaml

kubectl get svc

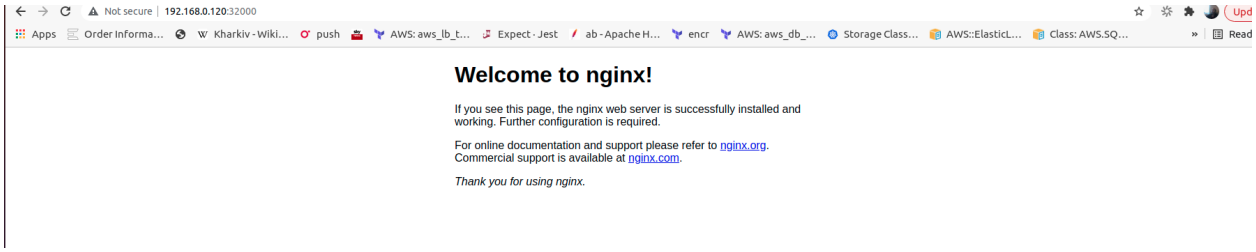
```
NAME           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
kubernetes     ClusterIP   10.96.0.1        <none>        443/TCP        15m
nginx-service  NodePort    10.101.115.149   <none>        80:32000/TCP   2m45s
```

Final test.

curl http://192.168.0.120:32000/

In the browser.

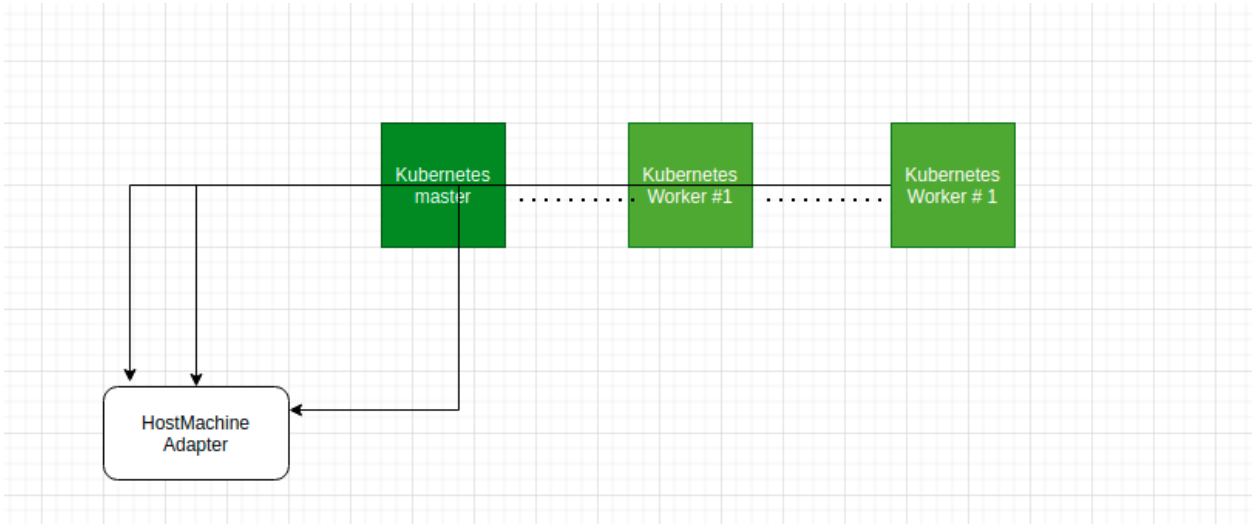
http://192.168.0.120:32000/



Detailed Technical Design

Whole solution is designed using hosted hypervisor Oracle Virtualbox.

A bridge network allows VMs that are provisioned to be in the same network as the host machine, which makes it easier to manage the Kubernetes cluster. IP addresses for the VMs are fixed with DHCP disabled on the Bridged Adapter. DNS server for the VMs are set to Public Dns Service Provided by Google **(8.8.8.8s)**



Node (hostname)	Role	IP address	Router	DNS

kubernetesmaster	kubernetes master	192.168.0.120	192.168.0.1	8.8.8.8
kubernetesworkerone	kubernetes worker # 1	192.168.0.121	192.168.0.1	8.8.8.8
kubernetesworkertwo	kubernetes worker # 2	192.168.0.122	192.168.0.1	8.8.8.8

Provisioning of VMs and Initial User.

Admin user	**Privileges**
student	sudo privileges/kubernetes admin

VMs are provisioned using Vagrant with a shell provisioner within vagrant creating the user student with sudo privileges and the student user will also be acting as a Kubernetes administrator.

Installing Docker, Kubernetes and Creating the Cluster

All the Vms us are on Centos 7

Container runtime to run kubernetes in docker which is configured to use systemd driver. Kubernetes is on the latest version.

Ansible roles and shell scripts

docker	Installs docker and configures it to use systemd driver
kubernetes	1) Completes prerequisite tasks for installing kubernetes on all nodes. a) Disable swap b) Disable selinux c) Disable Firewall. 2) Install Kubernetes. 3) Disables default NAT interface from Vagrant and configures bridge adapter correctly with ip , router and DNS server. Had to do it here, as I was not able to do it from Vagrant. Terraform would have been better, but does not have a provisioner for Oracle virtual box. **Following variables are configurable** **networkplugin** : uses calico by default for Pod CIDR . Can use any other plugin. **apiaddress and gateway:** api server address and gateway can also be customized as the ips can vary for developer machines

kubecluster	creates the master node, copies the config file to administer the kubernetes cluster to local machine.
deploy_kub_cluster.sh	1) VM and network provisioning process 2) Sets up ssh entries in the config file so a single command ssh “nodename” can ssh you into any of the kubernetes nodes. 3) Sets up an ssh connection for student users created. 4) Initiates ansible playbooks to install docker, kubernetes and provision cluster.

Vms and Network in Kubernetes cluster

After the cluster is provisioned, you can just shut down the VMs when you are not using the cluster and restart them when you require it. Screenshot below.

Click on the Machine → Right mouse click → Power off