Developer/Single-node installation

Thoughts on developer and single-node installation as per Locally-managed workbench.

Use case

The primary use case is for a developer to run a local copy of Labs Workbench on their laptop or a remote VM (AWS, GCE, Azure, OpenStack, etc). The basic requirements are simple:

- Bring up a (virtual) Kubernetes cluster
- Deploy Labs Workbench
- Ideally, process should mimic deployment in other environments
- Ideally, user can create new Docker containers and publish all in local environment

A secondary use case is the single-node installation for a project. This has come up a few times, but for example:

- SEAD/IML-CZO want to run Labs Workbench – want a simple install process that runs on a single VM during initial evaluation. May never scale up.

Current environment (ndslabs-startup)

For developers, we achieve this with the ndslabs-startup repo. A simple shell script kube-up.sh brings up Kubernetes via Hyperkube. This requires access to /var/lib/docker and /var/lib/kubelet, which has proven problematic on MacOS and Windows with newer Docker installs. A second shell script ndslabs-up.sh collects information from the user (domain name, IP address, email address) and uses kubectl to deploy a number of templated YAML files.

Pros:

- Simple, scripted process gets things running simply and quickly

Cons:

- Doesn't currently work on MacOS/Windows – requires running a separate VM
- Repo is messy and has unused stuff (could be cleaned up)
- "Mustache" variable replacement is inflexible.
- Doesn't include support for local DNS (can be added easily with bind container)
- Templates are duplicated in deploy-tools repository

Other options:

- Minikube (works on MacOS, Windows, Linux – doesn't require Docker; but also doesn't work on OpenStack VMs)
- Hyperkube (works well on VMs running Docker; doesn't work well on MacOS/Windows due to filesystem permission problems)
- Vagrant (works on MacOS via VirtualBox)
- Deploy-tools (could be used to deploy single-node without hyperkube)

Minikube

Minikube runs Kubernetes in a VM and sets up kubeconfig environment

- Requires virtualization environment, such as VirtualBox
- Download binary and run
- minikube start creates a VM in VirtualBox (or other virtualization environment)
- Sets up kubeconfig and local environment to access VM via kubectl
- In general it's functional and easy. I've run into snags for our development (i.e., running apiserver binary locally, trying to connect to kubernetes via Minikube), but this is an advanced case.

Minikube on MacOS

- Install VirtualBox
- Download binary
- minikube start
- git clone ndslabs-startup
- Don't need kube-up.sh, since we're running Minikube
- ndslabs-up.sh
    - Local DNS via bind works!
    - IP detection, etcd, etc, won't work as-is
    - Labeling won't work as is

Minikube Windows 10

- Install VirtualBox (can also use Hyper-V, but not tested here)
    - Disable Hyper-V: Settings > Turn Windows Features on or off; uncheck Hyper-V
- Download minikube for Windows
- minikube.exe start
    - Will make changes to VirtualBox
- Works well on Windows
- Hit snags with bind for local DNS, but likely user error configuring Windows DNS.
- Minikube was much easier than trying to run Docker on Windows.

Linux/VM

- Minikube will work on Linux, but so far we haven't been able to use it on a VM (e.g., OpenStack). Hyperkube works for this.

Hyperkube (ndslabs-startup)

Our current ndslabs-startup/kube-up.sh relies on Hyperkube (Kubernetes in Docker)

- gcr.io/google_containers/hyperkube-amd64
- The CoreOS project provides a process for launching Kubernetes on CoreOS, using another hyperkube image
    - https://coreos.com/kubernetes/docs/latest/kubernetes-on-generic-platforms.html
    - Uses quay.io/coreos/hyperkube image
    - This actually supports different configuration options (i.e., insecure-port) that might be helpful for some installations.

MacOS

- On MacOS, removing the /var/lib/kubelet volume actually works via kube-up.sh
- Adding bind container makes DNS reasonably easy

Windows 10

- Docker for Windows requires using Hyper-V (not Virtual Box)
- So far unable to get Hyperkube to work under Windows 10/Hyper-V without Vm.

VirtualBox VM

- You must follow the steps described here: https://coreos.com/os/docs/latest/booting-on-virtualbox.html
- You must add a Host-only adapter to your VM instance to SSH in
- You must manually load the .iso containing your ssh key, as described in the guide above
- You will a DNS entry pointing 192.168.99.XXX to a hostname, like www.local.ndslabs.org
    - This can be done by populating /etc/hosts, or using a tool such as Dnsmasq
- You can disable TLS (this is likely optional)
- Still minor bugs ongoing (for example: sending e-mail for signup throws a 503)

Openstack VM

- Process works well for OpenStack VMs
- Editing PATH is annoying, but otherwise fine.

Azure VM

- Deployed Ubuntu 16.10 VM via Azure portal
- Standard DS2_v2 (2 Cores, 7 GB memory). (Wow, that takes a really long time!)
    - Add endpoint for 443 access
- Start Labs Workbench
    - apt-install docker.io
    - git clone ndslabs-startup
    - kube-up.sh (+ bash_profile changes)
- Same SMTP problem as before – but manual registration worked

AWS VM

- Deployed CoreOS 1298.5 stable via AWS Management Console
    - t2.micro, because it was free! (way too small)
    - Expose HTTP/HTTPS ports via security group
    - Create a DNS rule pointing to the public IP of this node
- SSH in and start Labs Workbench
    - git clone ndslabs-startup
    - kube-up.sh (+ bash_profile changes)
    - ndslabs-up.sh
- Same SMTP problem as before

Vagrant

Vagrant is a tool to manage virtual machines. The approach described below comes from CoreOS as a method to deploy a Kubernetes cluster via VirtualBox (or similar). Vagrant has multiple providers, so it might be possible to use the same Vagrant configuration to deploy a Kubernetes cluster on VirtualBox on your laptop and deploy a full cluster on OpenStack. However, it's not clear how useful this might be.

MacOS

The VirtualBox/Vagrant approach suggested by David works on MacOS. As with Minikube, this will not work on an OpenStack or other VM without running a different virtualization package (VirtualBox won't work in a VM). It's not clear that Vagrant will support this configuration.

Following instructions from Multi Node Kubernetes developer environment - using VBox/Vagrant/CoreOS

1. Install Vagrant
    a. vagrant version
    b. Got error "failed generating SSL artifacts", solved with https://github.com/mitchellh/vagrant/issues/7747

    ```
    sudo ln -sf /usr/local/bin/openssl /opt/vagrant/embedded/bin/openssl
    ```

2. git clone https://github.com/coreos/coreos-kubernetes.git
3. cd coreos-kubernetes/multi-node/vagrant/
4. cp config.rb.sample config.rb, uncomment values
    a. Want 2GB for compute for us
5. vagrant up --provider virtualbox
6. vagrant status
    a. Should show three instances e1 (etcd), w1 (compute), c1 (master/controller)
    b. vagrant ssh c1 will take you into the controller node, will need to
7. Takes a few minutes for things to come up
    a. kubectl get nodes

ndslabs-up

- Domain name (dnsmasq)? For now add /etc/hosts entry
- IP address – IP of master node – same as used by kubeconfig
- ndslabs-up.sh
- kubectl label will change
- Email fails, but could output to log?
- No addons, currently
- Added bind service to support local DNS, based on instructions in
    - http://www.damagehead.com/blog/2015/04/28/deploying-a-dns-server-using-docker/
    - Worked like a charm, did have to add DNS server to network settings

Recommendations

- Support both Minikube and Hyperkube installs of Labs Workbench to support
- Refactor ndslabs-startup to work under both Minikube and Hyperkube installs.
    - Fix PATH
    - Improve kube-up (monitor status)
    - Improve ndslabs-up (more optional components, mustache, etc)