Bolted System: Auto-deployment Cloud Project
(Sprint 5)

Vidya Anandamurali

Pei Jia

Yuxi Jiang

Jiangnan Zou

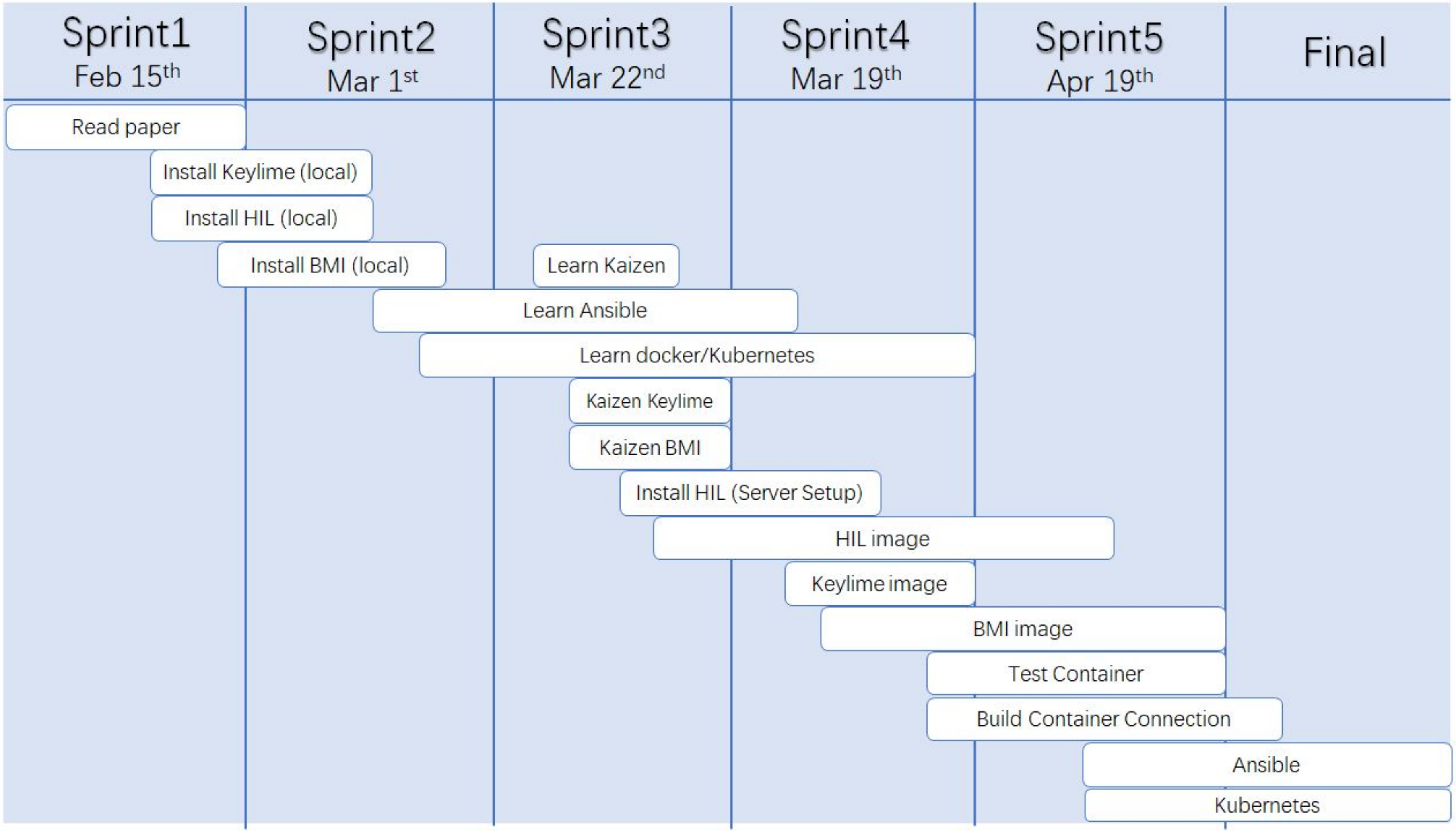
BOSTON UNIVERSITY

Project Description (Recap)

Automate the deployment of Bolted which consists of:

- Container image of each component of the Bolted system (HIL, BMI, Keylime and orchestration)
- Automated deployment of component containers on a cloud platform

Project Plan



Sprint1 Feb 15th

Sprint2 Mar 1st

Sprint3 Mar 22nd

Sprint4 Mar 19th

Sprint5 Apr 19th

Final

Read paper

Install Keylime (local)

Install HIL (local)

Install BMI (local)

Learn Kaizen

Learn Ansible

Learn docker/Kubernetes

Kaizen Keylime

Kaizen BMI

Install HIL (Server Setup)

HIL image

Keylime image

BMI image

Test Container

Build Container Connection

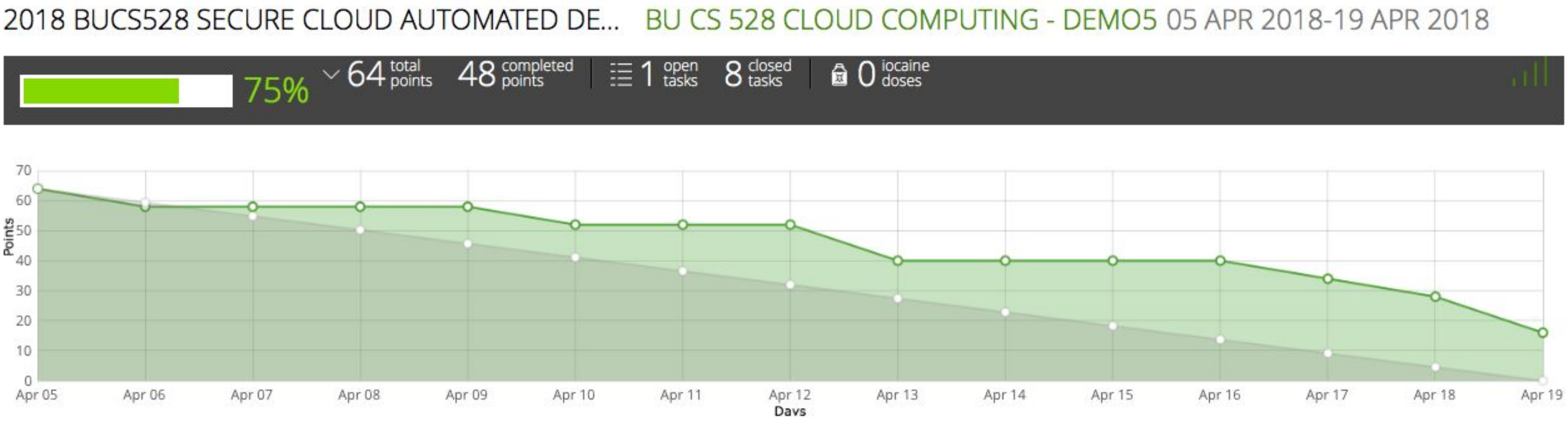
Ansible

Kubernetes

Last Sprint Report

- Finished building container image of HIL and BMI, now these two are under testing
- Learning and installing Kubernetes (In Progress)
- Found a solution to deploy containers automatically using Ansible as deployment method
- Built necessary connection between components (BMI & HIL, BMI & Ceph, HIL Servers and etc)
- Tested Keylime running on multiple VMs
- Tested HIL and BMI containers running on multiple VMs

Burndown Chart



2018 BUCS528 SECURE CLOUD AUTOMATED DE... BU CS 528 CLOUD COMPUTING - DEMO5 05 APR 2018-19 APR 2018

75% 64 total points 48 completed points 1 open tasks 8 closed tasks 0 iocaine doses

Project Progress (BMI)

- Finished deploying ceph server on Kaizen
  - One-node ceph for testing purpose
- Finished deploying BMI container on Kaizen
  - Require further configuration
  - Finished configuring ISCSI, DHCP, Sqlite3
  - Problem: Local or
- Established connection between BMI and ceph
- Have settled most of the configuration problem
- 90%

Project Progress (HIL)

Solution for HIL servers is using two containers to fulfill the requirements that HIL needs to run its server.

- PostgreSql container as hil database server
- Apache/httpd container as hil wsgi apache server and network server

Two containers are built separately under the same LAN environment as HIL server

Project Progress (Ansible)

ANSIBLE

Basic concept

- **Ansible** is software that automates software provisioning, configuration management, and application deployment.
- configuration management: Mange software on top of hardware.
- Features:
  - Agentless
  - Build on top of Python
  - Use ssh for secure connection
  - Push based architecture
  - Simply
- Write playbook ---> Run playbook

Host inventory

- Contains list of hosts, grouped together.
- Default location is

  /etc/ansible/hosts

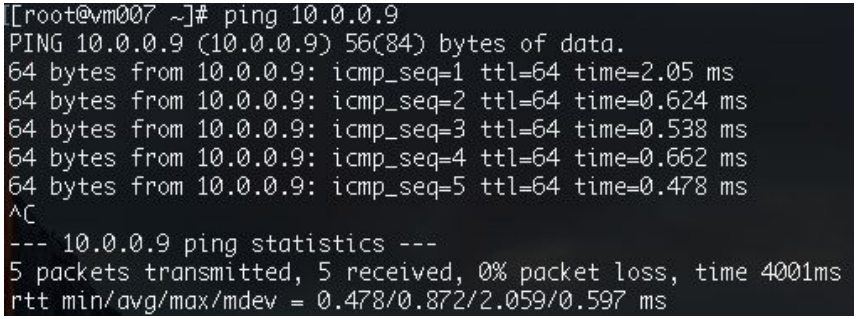
Installation

- sudo pip install ansible 🔴
- On RedHat/CentOS systems, python-pip and ansible are available via the EPEL repository
- rpm -ivh 🟢
  http://dl.fedoraproject.org/pub/epel/7/x86_64/\

Project Progress (Ansible)

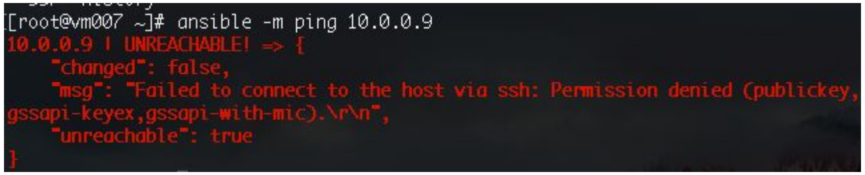
Ping Pong between VMs

- Success ping between to VM.



```
[root@vm007 ~]# ping 10.0.0.9
PING 10.0.0.9 (10.0.0.9) 56(84) bytes of data.
64 bytes from 10.0.0.9: icmp_seq=1 ttl=64 time=2.05 ms
64 bytes from 10.0.0.9: icmp_seq=2 ttl=64 time=0.624 ms
64 bytes from 10.0.0.9: icmp_seq=3 ttl=64 time=0.538 ms
64 bytes from 10.0.0.9: icmp_seq=4 ttl=64 time=0.662 ms
64 bytes from 10.0.0.9: icmp_seq=5 ttl=64 time=0.478 ms
^C
--- 10.0.0.9 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.478/0.872/2.059/0.597 ms
```

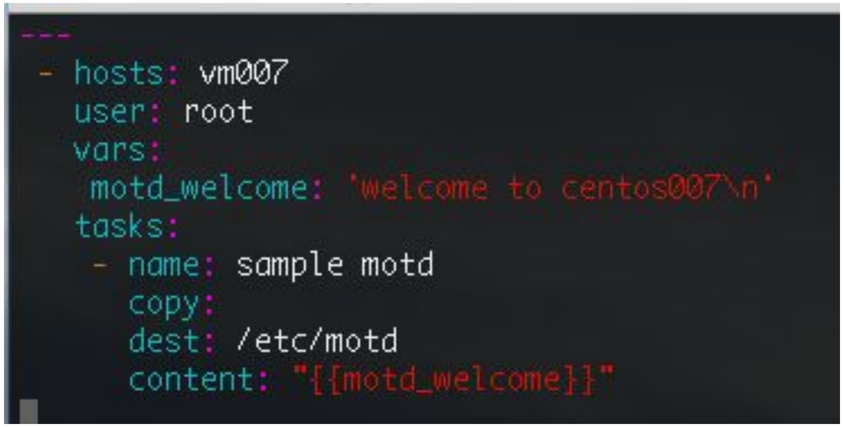
- Still unable to ping using ansible command



```
[root@vm007 ~]# ansible -m ping 10.0.0.9
10.0.0.9 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: Permission denied (publickey,
gssapi-keyex,gssapi-with-mic).\r\n",
    "unreachable": true
}
```

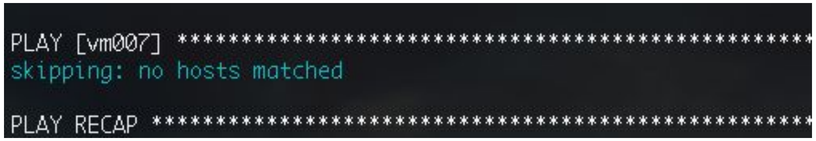
Ansible playbook

- Written in YAML



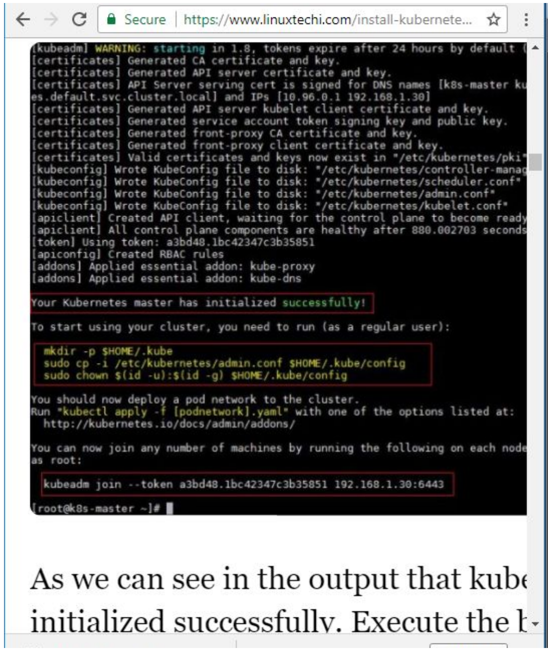
```
---
 - hosts: vm007
   user: root
   vars:
    motd_welcome: 'welcome to centos007\n'
   tasks:
     - name: sample motd
       copy:
       dest: /etc/motd
       content: "{{motd_welcome}}"
```

- No syntax error. Run failed due to previous reason



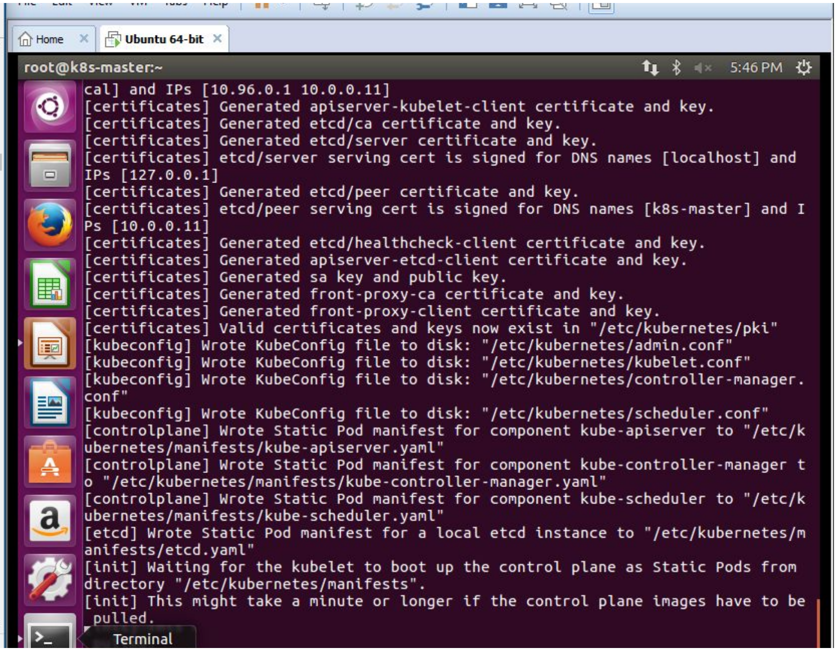
```
PLAY [vm007] ***************************************************
skipping: no hosts matched

PLAY RECAP *****************************************************
```

Project Progress (Kubernetes)
Challenges:



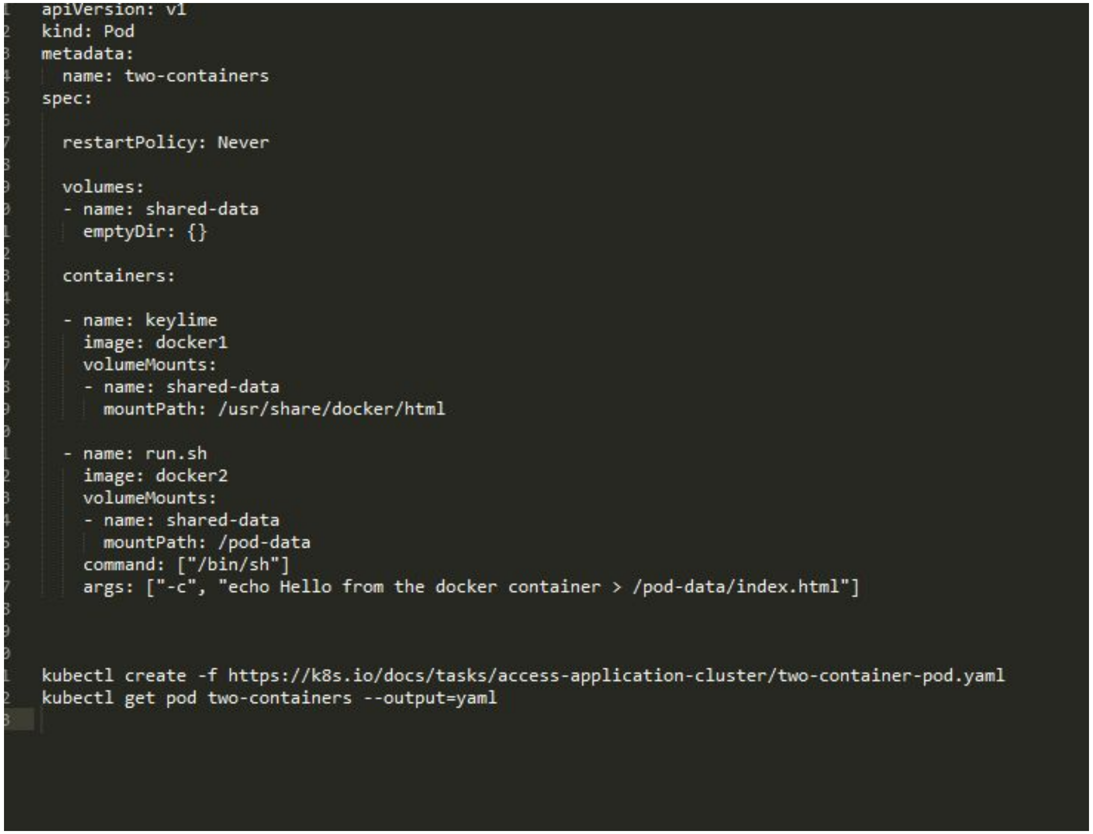
```
[kubeadm] WARNING: starting in 1.8, tokens expire after 24 hours by default (
[certificates] Generated CA certificate and key.
[certificates] Generated API server certificate and key.
[certificates] API Server serving cert is signed for DNS names [k8s-master ku
es.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.30]
[certificates] Generated API server kubelet client certificate and key.
[certificates] Generated service account token signing key and public key.
[certificates] Generated front-proxy CA certificate and key.
[certificates] Generated front-proxy client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manag
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[apiclient] Created API client, waiting for the control plane to become ready
[apiclient] All control plane components are healthy after 880.002703 seconds
[token] Using token: a3bd48.1bc42347c3b35851
[apiconfig] Created RBAC rules
[addons] Applied essential addon: kube-proxy
[addons] Applied essential addon: kube-dns

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run (as a regular user):

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  http://kubernetes.io/docs/admin/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join --token a3bd48.1bc42347c3b35851 192.168.1.30:6443

[root@k8s-master ~]#
```

As we can see in the output that kube
initialized successfully. Execute the b



```
cal] and IPs [10.96.0.1 10.0.0.11]
[certificates] Generated apiserver-kubelet-client certificate and key.
[certificates] Generated etcd/ca certificate and key.
[certificates] Generated etcd/server certificate and key.
[certificates] etcd/server serving cert is signed for DNS names [localhost] and
IPs [127.0.0.1]
[certificates] Generated etcd/peer certificate and key.
[certificates] etcd/peer serving cert is signed for DNS names [k8s-master] and I
Ps [10.0.0.11]
[certificates] Generated etcd/healthcheck-client certificate and key.
[certificates] Generated apiserver-etcd-client certificate and key.
[certificates] Generated sa key and public key.
[certificates] Generated front-proxy-ca certificate and key.
[certificates] Generated front-proxy-client certificate and key.
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.
conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
[controlplane] Wrote Static Pod manifest for component kube-apiserver to "/etc/k
ubernetes/manifests/kube-apiserver.yaml"
[controlplane] Wrote Static Pod manifest for component kube-controller-manager t
o "/etc/kubernetes/manifests/kube-controller-manager.yaml"
[controlplane] Wrote Static Pod manifest for component kube-scheduler to "/etc/k
ubernetes/manifests/kube-scheduler.yaml"
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/m
anifests/etcd.yaml"
[init] Waiting for the kubelet to boot up the control plane as Static Pods from
directory "/etc/kubernetes/manifests".
[init] This might take a minute or longer if the control plane images have to be
 pulled.
```

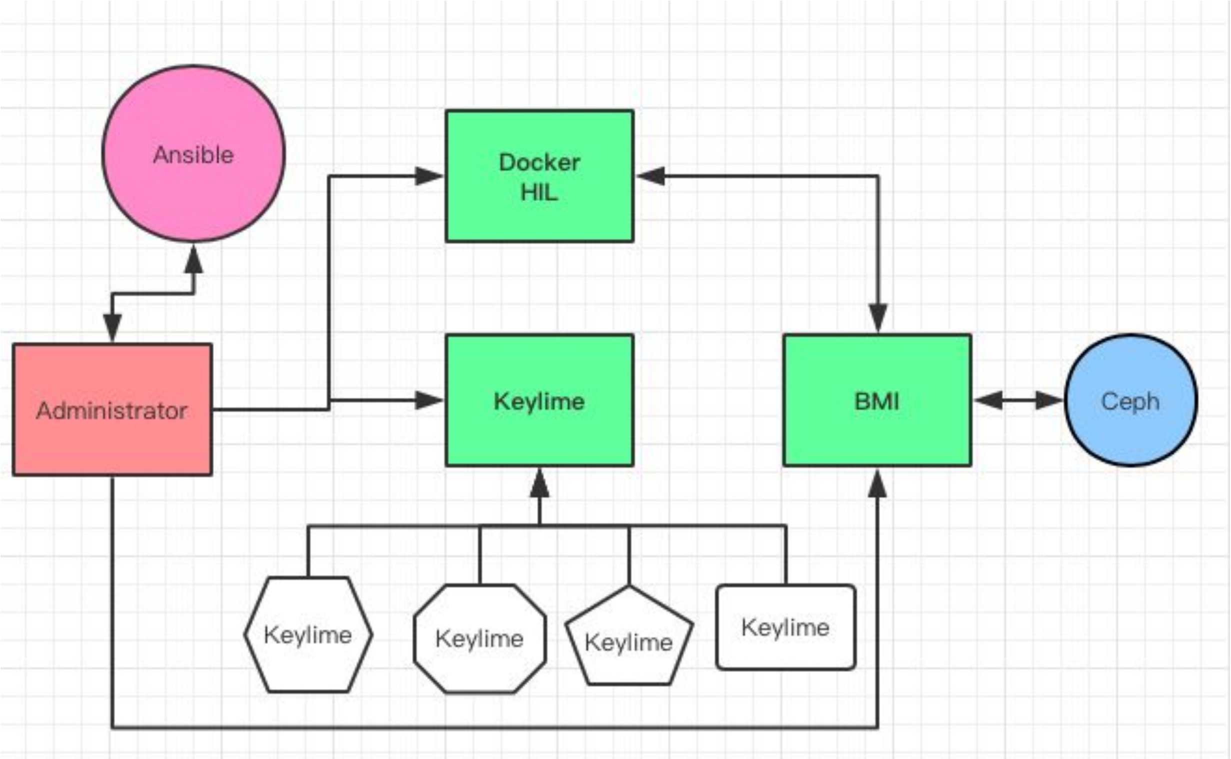
YAML file for Keylime Pod- Volume sharing between two containers:



```
apiVersion: v1
kind: Pod
metadata:
  name: two-containers
spec:

  restartPolicy: Never

  volumes:
  - name: shared-data
    emptyDir: {}

  containers:

  - name: keylime
    image: docker1
    volumeMounts:
    - name: shared-data
      mountPath: /usr/share/docker/html

  - name: run.sh
    image: docker2
    volumeMounts:
    - name: shared-data
      mountPath: /pod-data
    command: ["/bin/sh"]
    args: ["-c", "echo Hello from the docker container > /pod-data/index.html"]



kubectl create -f https://k8s.io/docs/tasks/access-application-cluster/two-container-pod.yaml
kubectl get pod two-containers --output=yaml
```

Demo

User Scenario



Responsibilities for next sprint

- Test and maintain container image for deploy
- An automated ansible script for installation of docker, HIL server, BMI server, Keylime server onto each VM from an admin VM for testing automated deploy
- Automate configuration between each component on Ansible Playbook based on user scenario

Thank you

Question ?