Jurnal Politeknik Caltex Riau
Terbit Online pada laman https://jurnal.pcr.ac.id/index.php/jkt/
| e- ISSN : 2460-5255 (Online) | p- ISSN : 2443-4159 (Print) |

# Perbandingan Kinerja Kubernetes Cluster dengan Virtualisasi KVM, Vagrant dan LXD

**Muhammad Arif Fadhly Ridha[1], Rahmat Suhatman[2*]**

[1]Politeknik Caltex Riau, Teknik Informatika, email: fadhly@pcr.ac.id
[2]Politeknik Caltex Riau, Teknik Informatika, email: rahmat@pcr.ac.id
*Corresponding Author: rahmat@pcr.ac.id

**[1] Abstrak**

*Teknologi yang dapat mengemas sebuah aplikasi dan kebutuhannya dalam sebuah virtualisasi disebut container. Pada proses pengembangan aplikasi saat ini dibutuhkan sebuah software yang dapat melakukan management terhadap container, salah satunya adalah docker. Namun kebutuhan yang selalu meningkat, mengakibatkan sistem tidak mampu untuk menampung banyak aplikasi, oleh karena itu pengembang membutuhkan sebuah software yang mampu melakukan management dari clustering container dalam skala besar. Salah satu software tersebut adalah kubernetes. Kubernetes dapat berjalan di dalam virtualisasi. Jenis virtualisasi yang dapat menjalankan kubernetes didalamnya antara lain Vagrant, LXD dan KVM. Pada penelitian ini dilakukan perbandingan antara 3 jenis virtualisasi tersebut, yaitu melihat tingkat keberhasilan dari high-availability, scalability dan performance testing. Hasil dari penelitian yang telah dilakukan adalah bahwa virtualisasi KVM lebih unggul dari virtualisasi vagrant dan LXD, yang dapat dilihat dari perbedaan latency dan performance test yang cukup besar antara kedua virtualisasi tersebut.*

***Kata kunci:*** *Cluster, Virtualisasi, Kubernetes, Vagrant, KVM, LXD.*

**[2] Abstract**

*The development of technology that can package an application and its needs in a virtualization called a container, therefore we need software that can manage containers, one of which is docker. However, the need is always increasing, the system is not able to accommodate many applications, therefore we need software capable of managing clustering containers on a large scale. One such software is Kubernetes. Kubernetes can run in virtualization, virtualization that can run Kubernetes in it include vagrant, LXD and KVM. In this study, a comparison was made between the 3 types of virtualization, the comparison was made to see the success rate of high-availability, scalability and performance testing. The results of the research that was carried out when the Kubernetes cluster was implemented in Vagrant, LXD and KVM virtualization, it was found that KVM virtualization is superior to Vagrant and LXD virtualization, which can be seen from the large difference in latency and performance tests between the two virtualizations.*

Dokumen diterima pada 27 Desember, 2021
Dipublikasikan pada 26 Juni, 2022

***Keywords:*** *Cluster, Virtualization, Kubernetes, Vagrant, KVM, LXD.*

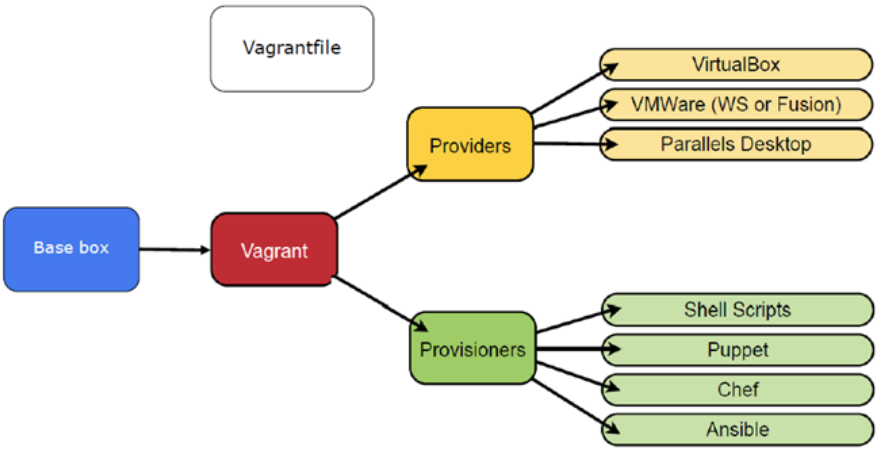
## 1. Pendahuluan

Pesatnya perkembangan teknologi yang mampu memudahkan seluruh bidang pekerjaan, teknologi terus berkembang karena kebutuhan dan sumber daya yang semakin meningkat. Permasalahan yang dihadapi adalah beban *traffic* yang besar sering kali membuat server *down* dan *overload*. Dengan adanya *clustering computing* dapat membuka peluang untuk dapat hadir dimanapun, memberikan kenyamanan, akses jaringan sesuai permintaan (*on-demand*) ke lokasi sumber daya komputasi terkonfigurasi (misalnya, jaringan, server, penyimpanan, aplikasi dan layanan), yang dapat dengan cepat dijalankan dan diluncurkan, dengan upaya pengelolaan minimal atau dengan menggunakan penyedia jasa layanan[1].

*Clustering computing* dapat dikombinasikan dengan *container, container* adalah virtualisasi pada level sistem operasi dimana tiap proses atau aplikasi yang dijalankan tiap container memiliki kernel yang sama. dimana virtual machine membutuhkan kernel sistem operasi yang berbeda-beda tiap aplikasi yang dijalankan[2]. Oleh karna itu dibutuhkan sebuah aplikasi yang mampu melakukan *management* terhadap *container*, salah satu aplikasi docker. Docker adalah sebuah project open-source yang menyediakan platform terbuka untuk developer maupun sysadmin untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun di dalam sebuah *container*[2]. Namun karena produktivitas semakin meningkat Maka dibutuhkan sebuah *software* yang dapat melakukan *clustering management container* agar aplikasi selalu tersedia dan dapat menerima banyak *traffic*. Salah satu *platform* yang dapat melakukan *clustering management container* adalah Kubernetes. Kubernetes adalah sistem open source untuk mengotomatisasi penyebaran, penskalaan, dan pengelolaan *container*.

Kubernetes sendiri berfungsi sebagai pengelola *container-container* serta menyediakan Platform untuk mendukung fungsinya. Design Kubernetes terdiri dari *Pods, Labels and Selectors, Controllers, Services*[3]. Kubernetes dapat berjalan didalam *virtual machine*, salah satu *virtual machine* yang dapat menjalankan kubernetes didalamnya adalah vagrant. Vagrant adalah sebuah *open source* untuk membangun dan mengelola lingkungan virtualisasi dalam satu alur kerja yang berfokus pada otomatisasi[4].

Vagrant adalah alat untuk otomatisasi dengan bahasa khusus yaitu DSL. DSL adalah sebuah *script* Yang digunakan untuk melakukan otomatisasi pembuatan *virtual machine* dan lingkungan *virtual machine*. Pengguna dapat membuat sekumpulan instruksi dengan menggunakan vagrant DSL untuk melakukan konfigurasi pada *virtual machine* tersebut[4]. Kemudian kubernetes *cluster* juga dapat berjalan pada kernel based virtual machine (KVM), KVM adalah teknologi virtualisasi yang dikembangkan oleh linux dengan perangkat keras type x86 (64-bit). KVM diimplementasikan sebagai modul kernel *loadable* yang mengubah kernel Linux menjadi *bare metal hypervisor*[5], Salah satu keunggulan utama KVM adalah terintegrasi nya modul KVM dengan kernel Linux sehingga KVM dapat langsung dipergunakan pada native kernel tanpa harus melakukan patch atau melakukan instalasi kernel terpisah[6].




Gambar 1. Arsitektur Vagrant

Kubernetes Cluster juga dapat berjalan pada LXD, LXD *Container* merupakan sekumpulan satu atau lebih suatu proses yang diisolasi dari seluruh sistem [7]. Dari penelitian ini didapatkan hasil bahwa kubernetes *cluster* dapat diterapkan dengan baik pada virtualisasi KVM, LXD dan vagrant. Kemudian dilakukan pengujian *high-availability*, *scalability* dan *performance testing* sebanyak 15 kali, data pengujian dianalisis untuk melihat perbedaan antara 2 jenis teknologi virtualisasi tersebut saat diterapkan kubernetes *cluster* didalamnya.

## 2. Metode Penelitian

Pada penelitian ini digunakan 3 buah server berbasis sistem operasi linux yang dipasang teknologi virtualisasi yang berbeda yaitu KVM, LXD dan Vagrant. Setiap server akan memiliki 3 buah *nodes* KVM dan vagrant.

Adapun metode pengujian yang dilakukan pada penelitian ini yaitu :

1. Pengujian *High-Availability*

Pengujian ini dilakukan sebanyak 15 kali dengan meng-*offline*-kan salah satu *node* pada server pengujian ini bertujuan untuk mengetahui keberhasilan *failover, failback* dan *loadbalancer*. Adapun pengujian yang dilakukan sebagai berikut :

- Melihat tingkat keberhasilan *Failover* dan *Load Balancing.*
- Melihat tingkat keberhasilan *Failback.*

2. Pengujian *Scalability*

Pengujian *scalability* adalah kemampuan melakukan penambahan saat beban layanan meningkat dan melakukan pengurangan layanan pada saat penurunan beban. Untuk mengetahui tingkat keberhasilan dari *scalability* sebagai berikut :

- Dapat menambah pod pada saat beban layanan meningkat, dan mematikan pod pada saat tidak digunakan.
- Dapat menambah node saat layanan meningkat dan mengurangi node pada saat node tidak digunakan.

3. Pengujian *Performance testing*

Pengujian *performance* dilakukan dengan menggunakan aplikasi *stress tools web server* yaitu JMeter. pembebanan dilakukan untuk mengetahui besar penggunaan CPU dan *memory* pada kedua server tersebut. Untuk monitoring tersebut menggunakan sebuah *software* SNMP untuk menganalisis perbedaan penggunaaan CPU dan *Memory* pada kedua server tersebut. Adapun pengujian yang dilakukan seperti berikut :

- Pengujian menggunakan aplikasi JMeter untuk dilakukan pembebanan dan *request* pada layanan web server.
- Pengujian perbandingan persentase penggunaan CPU dan Memory pada server KVM dan server vagrant menggunakan aplikasi SNMP.

## 3. Hasil dan Pembahasan

Hasil dan Analisis pengujian *High-Availability*.
Setelah melakukan pengujian *failover, failback* dan *load balancer* yang dilakukan sebanyak 15 kali dengan skenario melakukan *offline* pada salah satu *nodes* masing-masing server. Berikut tabel hasil dari pengujian tingkat keberhasilan dari *failback* dan *failover* :

Tabel 1. Tingkat Keberhasilan *failback, failover*

| Pengujian Ke - | Vagrant | | KVM | |
|---|---|---|---|---|
| | Failback | Failover | Failback | Failover |
| 1 | ✓ | ✓ | ✓ | ✓ |
| 2 | | | | |
| 3 | ✓ | ✓ | ✓ | ✓ |
| 4 | ✓ | ✓ | ✓ | ✓ |
| 5 | | | | |
| 6 | ✓ | ✓ | ✓ | ✓ |
| 8 | ✓ | ✓ | ✓ | ✓ |
| 9 | | | | |
| 10 | ✓ | ✓ | ✓ | ✓ |
| 11 | ✓ | ✓ | ✓ | ✓ |
| 12 | | | | |
| 13 | ✓ | ✓ | ✓ | ✓ |
| 14 | ✓ | ✓ | ✓ | ✓ |
| 15 | | | | |
| | ✓ | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ | ✓ |
| | ✓ | ✓ | ✓ | ✓ |

Pada tabel 1. tingkat keberhasilan *failback* dan *failover* pada 2 server berhasil dilakukan sebanyak 15 kali. *Loadbalancer* yang diterapkan pada kubernetes *cluster* yaitu distribusi *resource/pods* kepada setiap *nodes.*

2. Hasil dan Analisis pengujian *Scalability*

1) Melakukan penambahan pods & Pengurangan pods

Teknik yang dilakukan untuk melakukan penambahan dan pengurangan *pods* yaitu dengan menggunakan *horizontal pods autoscaller* (HPA) yang merupakan salah satu teknik otomatisasi dengan menyesuaikan beban dari *traffic*. Sehingga ketiga beban *traffic* meningkat, *pods* akan otomatis melakukan penambahan. Dan pada saat *traffic* menurut *pods* akan menyesuaikan layanan sesuai dengan user yang mengakses.

```
NAME            REFERENCE                     TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
wordpress-hpa   Deployment/wordpress-deploy   192%/50%   1         10        4          52m
```

Gambar 2. *Horizontal Pods Autoscaler*

2) Melakukan penambahan dan pengurangan nodes

Pengurangan *nodes* dilakukan sebanyak 1 kali dengan skenario melakukan *offline* pada salah satu *nodes*. Kemudian untuk penambahan *nodes, nodes* yang sebelumnya *offline* di-*online*-kan kembali.
Melakukan *offline nodes* :

```
[root@madanvagrant vagrant]# vagrant halt node2
==> node2: Attempting graceful shutdown of VM...
```

Gambar 3. *Offline node 2*

Pada Gambar 3 adalah printah untuk melakukan *offline* terhadap *nodes*.

Status *nodes* :

```
[root@madanvagrant vagrant]# vagrant status
Current machine states:

master                    running (virtualbox)
node1                     running (virtualbox)
node2                     poweroff (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

Gambar 4. Status *node2* setelah *offline*

Pada Gambar 4 adalah perintah untuk melihat status *nodes* pada lingkungan virtualisasi. Dimana *node 2* telah dilakukan *offline*.

Melakukan *online nodes* :

```
[root@madanvagrant vagrant]# vagrant up node2
Bringing machine 'node2' up with 'virtualbox' provider...
```

Gambar 5. *Online node 2*

Pada Gambar 5 adalah perintah untuk melakukan *online* terhadap *nodes*.
Status *nodes* :

```
[root@madanvagrant vagrant]# vagrant status
Current machine states:

master                    running (virtualbox)
node1                     running (virtualbox)
node2                     running (virtualbox)

This environment represents multiple VMs. The VMs are all listed
above with their current state. For more information about a specific
VM, run `vagrant status NAME`.
```

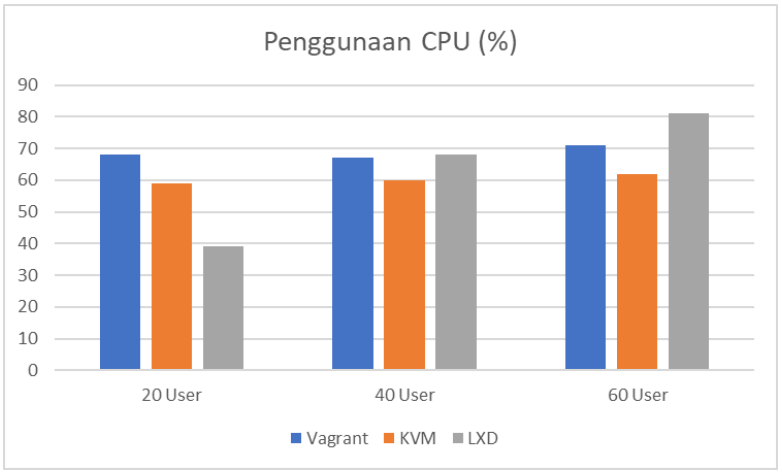
Gambar 6. Status *node 2* dilakukan *online*

Pada Gambar 6 adalah perintah untuk melihat status pada lingkungan virtualisasi, dimana *node 2* yang sebelumnya *offline* di-*online*-kan kembali.

3. *Performance Testing*

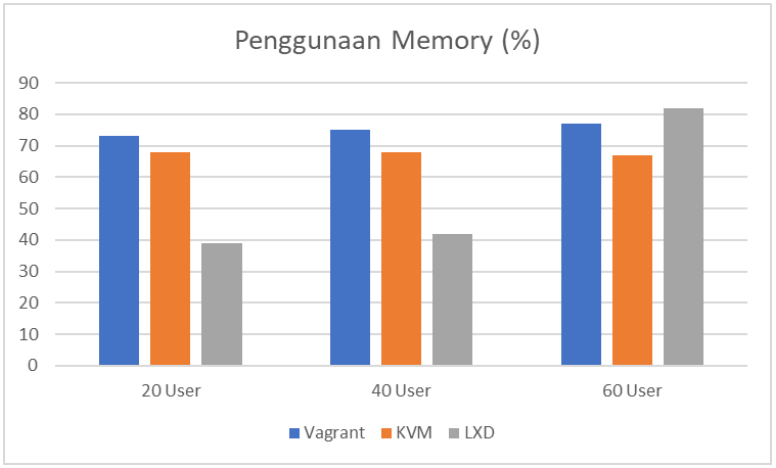
Pengujian *performance* CPU dan Memory dilakukan pada saat stress request server dalam menangani client.

Data persentase penggunaan CPU saat dilakukan *stress request* pada server vagrant, lxd dan KVM dapat dilihat pada Gambar 7. Adapun persentase penggunaan CPU saat dilakukan *stress request* pada server vagrant yaitu 20 user sebesar 68%, 40 user sebesar 67% dan 60 user yaitu 71%. Pada server KVM yaitu 20 user sebesar 59%, 40 user sebesar 60% dan 60 user sebesar 62%. Pada server LXD yaitu 20 user sebesar 39%, 40 user sebesar 68% dan 60 user sebesar 81%.




Gambar 7. Perbandingan Penggunaan CPU

Data persentase penggunaan memory pada saat dilakukan *stress request* pada server vagrant dan server KVM dapat dilihat pada Gambar 8. Adapun persentase penggunaan memory pada saat dilakukan *stress request* pada server vagrant yaitu 20 user sebesar 73%, 40 user sebesar 75% dan 60 user yaitu sebesar 77%. Pada server KVM yaitu 20 user sebesar 68%, 40 user sebesar 68% dan 60 user sebesar 67%. Pada server LXD yaitu 20 user sebesar 39%, 40 user sebesar 42% dan 60 user sebesar 82%.




Gambar 8. Perbandingan Penggunaan Memory

Setelah mendapatkan data pengujian *performance* CPU dan Memory, dapat dianalisis bahwa rata-rata penggunaan sumberdaya CPU dan Memory pada saat diakses 20, 40 dan 60 user pada server Vagrant dan LXD lebih besar dari penggunaan sumberdaya CPU dan Memory yang digunakan pada server KVM.

## 4. Kesimpulan

Dari hasil Pengujian dapat diambil kesimpulan sebagai berikut:

i. Kubernetes cluster dapat dibangun di dalam virtual mesin vagrant, LXD dan KVM.
ii. Horizontal pod autoscaler (HPA) dapat melakukan replikasi pods secara otomatis pada teknologi kubernetes.
iii. Failback dan failover dapat diimplementasikan setelah dilakukan pengujian sebanyak 15 kali pada virtualisasi vagrant, LXD dan KVM. Load Balancing melakukan distribusi resource berupa pods kepada nodes.
iv. Saat dilakukan stress request penggunaan CPU dan Memory pada virtualisasi vagrant dan LXD lebih besar dari virtualisasi KVM pada saat diakses 20, 40 dan 60 user.

## Daftar Pustaka

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *Gaithersbg. Elsevier.*, 11AD.
[2] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *Prosisko*, vol. 7, no. 2, pp. 165–175, 2020, [Online]. Available: https://e-jurnal.lppmunsera.org/index.php/PROSISKO/article/view/2520/.
[3] "Kubernetes," 2018. https://kubernetes.io/id/docs/concepts/.
[4] N. el A. Khalif, "Ceph Cluster Creation And Management On Vagrant Virtual Machine," 2017.
[5] IDreg, "Pengertian KVM, XEN, OPENVZ," 2013. https://www.idreg.net/pengertian-kvm-xen-openvz/.
[6] D. Kartikasari, "Analisa Perbandingan Metode KVM Dengan OpenVZ Pada Mesin VPS (Virtual Private Server) Di PT. Lintas Data Prima Yogyakarta," 2012.
[7] A. E. Saputra *et al.*, "Analisis Penggunaan System Process Pada Migrasi Aplikasi Dalam Linux Container ( Lxd ) Menggunakan Lxd Api Analysis Of The Use Of The Process System In Application Migration In Linux Container ( Lxd ) Using Lxd Api" in *ISSN : 2355-9365 e-Proceeding of Engineering*, 2019, vol. 6, no. 2, pp. 7862–7869.