BAB III

ANALISI DAN PERANCANGAN

3.1 Analisi Sistem

Tahap pertama dalam merancang sebuah mesin *orchestration* untuk memanajemen *container* yaitu dengan melakukan tahap analisis terhadap sistem yang berjalan pada *cluster kubernetes*. Analisi sistem bertujuan untuk mengetahui sistem yang dibutuhkan dalam menjalankan *container* didalam wadah *kubernetes* agar dapat berfungsi dengan baik sesuai dengan kebutuhan *production.*

Dalam melakukan analisis *container* pada *cluster* di *kubernetes* berbasis virtualisasi terdapat beberapa aspek penting yang perlu diperhatikan yaitu jaringan internet, membuat 3 server virtual di *Amazon Web Services* yaitu 1 server *master node* dan 2 server *client* atau yang biasa disebut *worker node* serta memastikan koneksi antara *master node* dan *worker node* terhubung dengan baik.

3.2 Rancangan Sistem

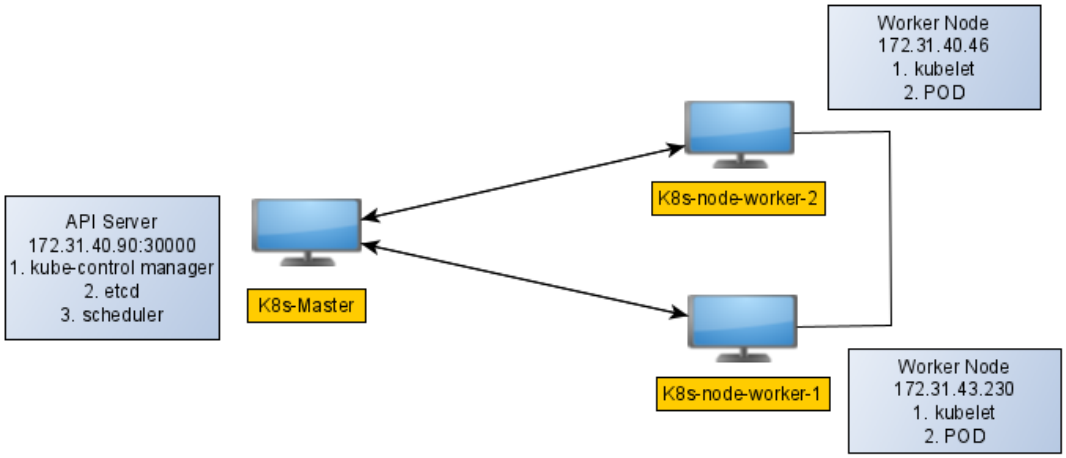
Sebelum memanajemen *container* menggunakan kubernetes dalam *Amazon Web Services* (AWS) sebagai server virtual, maka dilakukan perancangan sistem terlebih dahulu. Perancangan sistem dimulai dengan membangun *Virtual Private Server* (VPS) dengan sistem operasi AMI ID Ubuntu 16.04 *Xenial Server* yang akan di install di Amazon Web Services EC2.

Setelah *setup* VPS dengan sistem operasi Ubuntu 16.04 *Xenial Server*, selanjutnya adalah menjalankan sistem operasi tersebut di terminal atau dapat menggunakan PuTTY sebagai *remote access* untuk mengendalikan sistem dari jarak

jauh atau tempat lokasi yang berbeda. Di sistem operasi Ubuntu Xenial 16.04 dapat dilakukan penginstalan *docker* sebagai aplikasi *container* dan mengisntal Dashboard UI web based kubernetes untuk kebutuhan manajemen cluster.

3.2.1 Perancangan Logika Sistem

Perancangan logika kerja sistem bertujuan untuk memudahkan dalam melakukan pembuatan sistem. Dengan demikian pada rancangan sistem ini akan menggambarkan seperti apa kinerja sistem yang akan dibuat dan bagaimana proses yang terjadi sehingga efisien perangkat dapat terwujudkan dengan melakukan penganalisaan terhadap proses yang telah dilakukan.

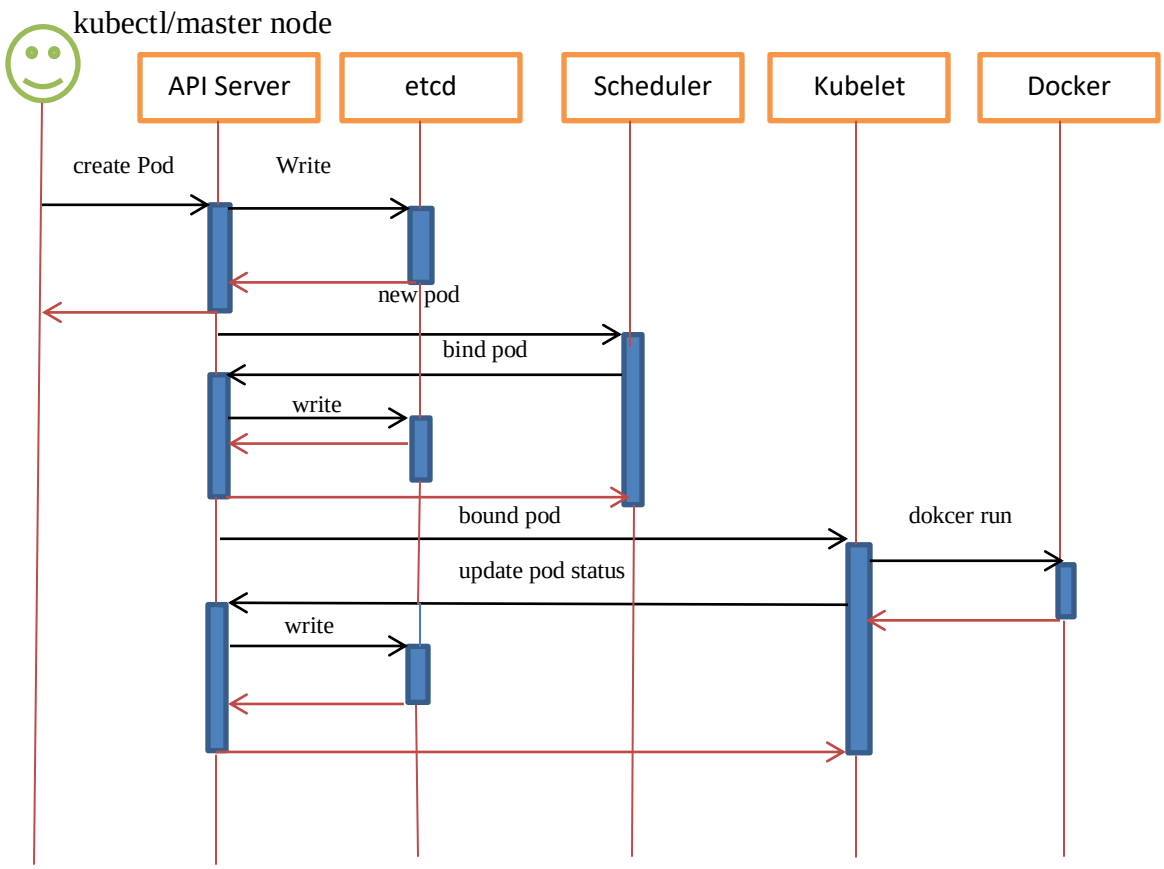


Gambar3. 1 rancangan logika kerja sistem komunikasi antar komponen kubernetes

Dari gambar diatas terdapat 3 server virtual sebagai *master node* dan *worker node*. Pada *master node* bertugas mengelola keadaan dan kondisi *cluster* atau komponen-komponen yang ada pada *master node* adalah otak dibalik semua operasi didalam sebuah *cluster*. *Master node* juga berperan dalam memastikan

antara jaringan *worker node* saling terhubung antara lain dan dapat di akses dengan baik.

Worker node yaitu sebagai penyedia environment bagi aplikasi/layanan klien. Melalui layanan-layanan kecil (*microservices*) yang saling berinteraksi dan dibungkus dalam masing-masing *container*, kumpulan layanan tersebut selanjutnya dibungkus dalam suatu komponen pada kubernetes yang disebut Pod. Pod yang ada pada worker node selalu dikontrol oleh komponen *control plane (*controller, *scheduler)* pada master node.

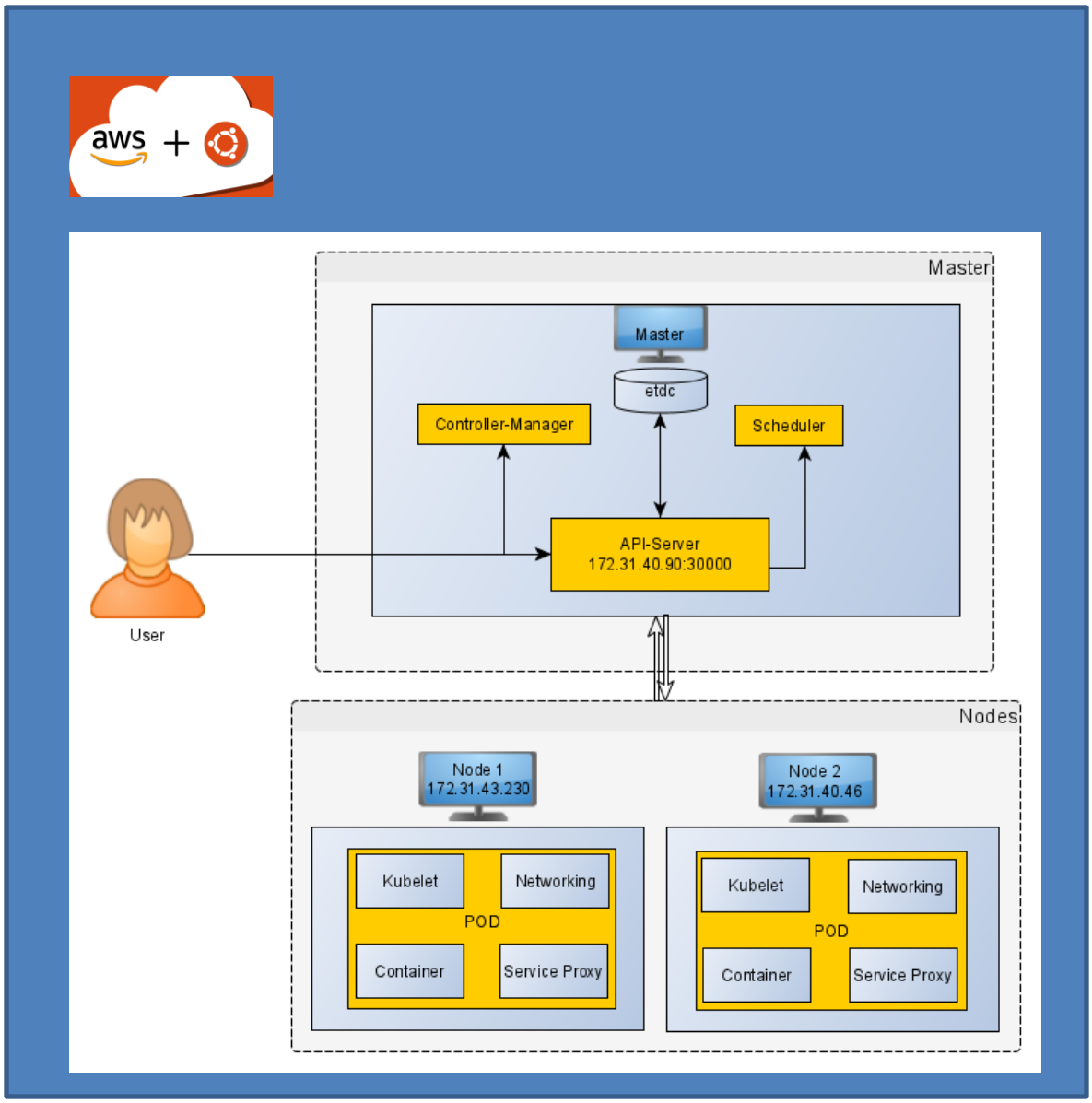


Gambar3. 2 flowchart kerja kubectl untuk create pod di dalam kubernetes

Dari gambar di atas dapat dilihat saat *master node* membuat pod sebagai *kubectl* untuk memerintahkan ke Server API dan memvalidasi permintaan serta menyimpannya ke etcd. Etcd akan memberitahu kembali API Server, setelah itu API server memanggil *scheduler* dilakukan penjadwal memutuskan tempat untuk menjalankan pod dan mengembalikannya ke server API. Setelah itu server API mempertahankannya ke etcd, etcd akan memberitahu kembali ke API Server. API Server juga memanggil *kubelet* di *node* yang sesuai. *Kubelet* akan saling konfirmasi dengan daemon docker menggunakan API melalui soket docker untuk membuat *container* serta *kubelet* akan memperbarui status pod ke Server API dan server API akan mempertahankan status baru di etcd dan pod akan terbuat, *container* atau aplikasi layanan klien dapat dijalankan.

3.2.2 Rancangan Topologi Kubernetes Sebagai Mesin Orchestration untuk Memanajemen Container

Untuk membangun kubernetes sebagai mesin orchestration dengan sistem operasi Ubuntu 16.04 *Xenial Server* dari layanan *Amazon Web Services* (AWS) diperlukan sebuah rancangan topologi, berikut adalah topologi *kubernetes*:



Gambar3. 3 Topologi Kubernetes

Gambar diatas adalah rancangan topologi proses pembangunan *resource* pada *kubernetes* sebagai mesin *orchestration* untuk memanajemen *container*. Master *kubernetes* menjalankan komponen *Scheduler*, *Controller Manager*, API Server dan etcd dan bertanggung jawab untuk mengelola *cluster Kubernetes*. Pada dasarnya, *master* adalah otak *cluster kubernetes*. Pada bagian *nodes* terdapat 2

node sebagai *worker node* yang akan menjalankan sebuah wajah yang dinamakan POD. Komponen pod sendiri terdapat *networking, kubelet, service proxy*, dan *container*. Didalam *cluster* itu semua akan dilakukan manajemen *container* sehingga *container* dapat dipastikan bisa di akses oleh user dengan baik.

3.2.3 Perangkat yang Akan Digunakan

Perangkat yang akan digunakan dalam sebuah sistem sangat berpengaruh terhadap kinerja sistem. Adapun fitur-fitur dan fungsi-fungsi pada cluster yang belum disediakan oleh kubernetes yang disebut dengan *addons*, untuk menambahkannya digunakan melalui bantuan aplikasi/layanan pihak ketiga *(3rd-party)* pod dan *services.* Aplikasi atau layanan yang diperlukan yaitu:

1. DNS-*cluster* DNS adalah DNS server yang diperlukan untuk meng-assign DNS *records* ke sumber daya dan objek *kubernetes*.
2. Dashboard-UI web *based* untuk kebutuhan manajemen *cluster* yang akan di install terlebih dahulu oleh *master node* dengan 2 *worker node*.
3. Docker

   Docker merupakan project *open source* yang menyediakan platform terbuka dalam bentuk teknologi virtualisasi berbasis *container*, ditujukan bagi para *developer* maupun *sysadmin* untuk dapat membangun, membundel, dan menjalankan aplikasi dimanapun dalam satu *container* yang ringan.

Untuk menjalankan dokcer *container* dan *kubernetes* akan dilakukan dalam sebuah layanan server virtual yaitu *Amazon Web Services* (AWS). Adapun

environment yang disiapkan untuk menjalankan dokcer *container* dan *kubernetes* yaitu:

a. Install AWS EC2 free tier atau sistem operasi Ubuntu 16.04 *Xenial Server*.
b. Siapkan 3 server, 1 *master node* dna 2 *worker node*.
c. Install *Dashboard*-UI web *based.*