Memasang Sertifikat Digital pada Situs Web Menggunakan Kubernetes

Cynthia Rusadi - 13519118
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13519118@std.stei.itb.ac.id

Abstrak*—This electronic document is a "live" template and already defines the components of your paper [title, text, heads, etc.] in its style sheet.** ****CRITICAL: Do Not Use Symbols, Special Characters, or Math in Paper Title or Abstract. **(*Abstract*)**

Keywords—component; formatting; style; styling; insert (key words)

I. PENDAHULUAN

Pada awalnya, setiap koneksi ke situs web tidak terenkripsi, yang berarti koneksinya tidak aman dan dapat di-'serang' oleh siapapun. Maka dari itu, solusi yang ada pada saat ini agar koneksi tersebut menjadi lebih aman adalah untuk memasang sertifikat digital pada situs web. Fungsi dari sertifikat digital adalah untuk memverifikasi identitas dari suatu perangkat atau pengguna, dan enkripsi koneksi tersebut. Pembeda utama yang membedakan apakah suatu situs web menggunakan sertifikat digital atau tidak adalah situs web yang tidak menggunakan sertifikat digital berawalan *http* (*Hypertext Transfer Protocol*), sedangkan situs web yang menggunakan sertifikat digital berawalan *https* (*Hypertext Transfer Protocol Secure*).

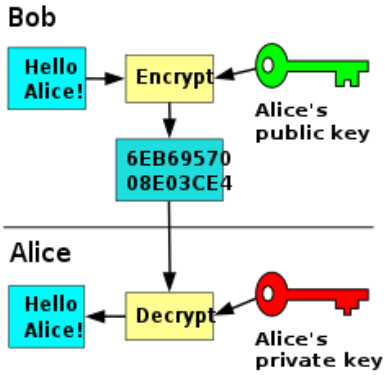
Sertifikat digital dapat diperoleh dari CA (*Certificate Authority*) yang berbeda-beda, CA yang terkenal beberapa di antaranya adalah COMODO, DigiCert, GeoTrust, Thawte, GlobalSign, Let's Encrypt, Entrust, Network Solutions, The SSL Store, dan seterusnya. Setiap CA memiliki keunikan, kelebihan, dan kekurangannya masing-masing.

Terdapat beberapa metode untuk memasang sertifikat digital pada suatu situs web, dan salah satunya adalah dengan menggunakan Kubernetes, yang berarti *deployment* dari situs web tersebut juga harus dilakukan menggunakan Kubernetes, dengan CA-nya adalah Let's Encrypt.

II. TEORI DASAR

A. *Kriptografi Kunci-Publik*

Kriptografi kunci-publik dapat disebut juga sebagai kriptografi kunci-nirsimetri atau *asymmetric-key cryptography* karena kunci yang digunakan untuk enkripsi dan dekripsi berbeda. 'Publik' berarti kunci untuk enkripsi diumumkan kepada publik dan tidak bersifat rahasia, sedangkan 'privat' berarti hanya pemilik kunci privat yang mengetahui kuncinya sendiri.



Gambar 1. Ilustrasi Kriptografi Kunci-Publik

Keuntungan dari kriptografi kunci-publik adalah tidak diperlukan pengiriman kunci privat, yang berarti setiap orang memiliki kunci privatnya masing-masing. Keuntungan lainnya adalah jumlah kunci dapat ditekan, yang berarti setiap orang hanya perlu memiliki sepasang kunci saja (privat dan publik yang saling berhubungan).

Kriptografi kunci-publik didasari pada fakta bahwa komputasi untuk enkripsi atau dekripsi pesan mudah dilakukan dan secara komputasi hampir tidak mungkin menurunkan kunci privat apabila kunci publik sudah diketahui.

Kelebihan dari kriptografi ini adalah hanya kunci privat saja yang perlu dijaga kerahasiaannya oleh setiap entitas yang melakukan komunikasi dan tidak ada kebutuhan untuk mengirim kunci privat. Selanjutnya adalah pasangan kunci publik dan kunci privat tidak perlu sering diubah, bahkan dalam periode waktu yang lama. Lalu, kriptografi ini juga dapat digunakan untuk mengamankan pengiriman kunci simetri. Yang terakhir adalah terdapat beberapa algoritma kunci-publik yang dapat digunakan untuk memberi tanda tangan digital pada pesan.

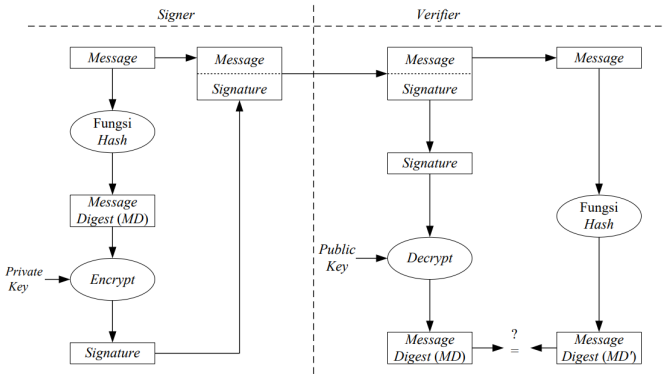
Selain kelebihan-kelebihan tersebut, terdapat beberapa kekurangan juga yang harus diperhatikan, yaitu prosesnya yang lebih lambat daripada sistem kriptografi simetri karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang cukup besar. Lalu, ukuran cipherteks akan lebih besar daripada plainteks. Selain

itu, ukuran kunci relatif lebih besar daripada ukuran kunci simetri. Yang keempat adalah cipherteks tidak dapat memberikan informasi mengenai otentikasi pengirim karena kunci publik diketahui secara luas, dan yang terakhir adalah tidak ada algoritma kunci-publik yang terbukti aman.

B. Tanda Tangan Digital

Tanda tangan digital adalah nilai kriptografis yang bergantung pada isi pesan dan kunci. Perbedaan dari tanda tangan dokumen dan tanda tangan digital adalah tanda tangan dokumen akan selalu sama dan tidak bergantung pada isi dokumennya, sedangkan tanda tangan digital bergantung pada isi dari pesannya, yang berarti akan berbeda-beda antara satu pesan dan yang lainnya.

Proses tanda tangan digital terbagi menjadi dua, yaitu menandatangani pesan (*signing*) dan memverifikasi pesan (*verification*). Proses ini dilakukan dengan menggunakan kombinasi fungsi *hash* dan kriptografi kunci-publik. Ide ini dikemukakan oleh Diffie dan Hellman. Prosesnya adalah pesan dienkripsi dengan kunci privat pengirim, dengan begitu pesan dide`kripsi dengan menggunakan kunci publik pengirim. Dengan menggunakan cara ini, kerahasiaan pesan dan otentikasi pengirim dicapai sekaligus. Penerima dapat mengotentikasi pengirim karena kunci publik dan kunci privat selalu berpasangan.



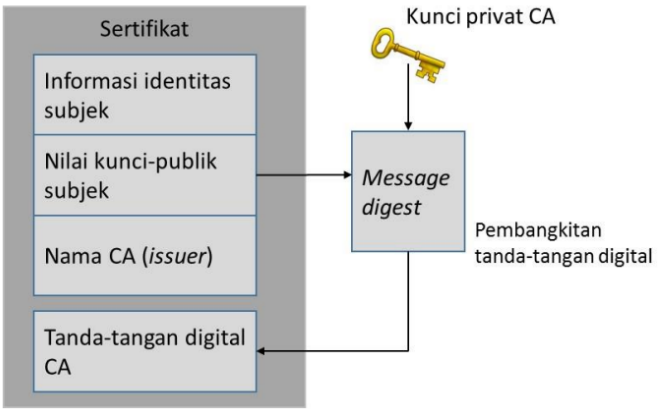
Gambar 2. Proses *Signing* dan *Verification*

C. Sertifikat Digital

Kunci publik perlu disertifikasi dengan diberikan sertifikat digital karena kunci publik tersedia secara publik. Sertifikat digital merupakan dokumen digital yang mengikat kunci publik dengan informasi pemiliknya. Sertifikat digital dikeluarkan oleh pemegang otoritas sertifikasi, yaitu CA, *Certificate Authority*, atau *Issuer*. CA akan menandatangani sertifikat digital. Sertifikat digital memiliki fungsi yang sama seperti SIM atau paspor.

Informasi minimal yang terdapat dalam sertifikat digital adalah identitas subjek (perusahaan/individu dari pemilik kunci publik), kunci publik subjek, nama CA, dan tanda tangan CA. Terdapat digital sertifikat menggunakan format X.509, dengan tujuan agar setiap sertifikat digital seragam, dan informasi yang harus terdapat pada sertifikat digital menggunakan format ini adalah sebagai berikut.

1. Versi
2. Nomor serial
3. Algoritma tanda tangan sertifikat
4. Nama CA
5. Periode valid
6. Nama subjek
7. Kunci publik subjek dan algoritma kriptografi kunci-publik yang digunakan
8. ID opsional yang secara unik mengidentifikasi CA
9. ID opsional yang secara unik mengidentifikasi subjek
10. Ekstensi (opsional)
11. Tanda tangan digital
12. Algoritma tanda tangan



Gambar 3. Ilustrasi Sertifikat Digital

Proses untuk mendapatkan sertifikat digital adalah subjek meminta sertifikat digital pada CA, dengan memberikan kunci publiknya, lalu CA akan membuatkan sertifikat digital tersebut dan menandatanganinya menggunakan kunci privat CA. Dalam proses ini, CA akan membangkitkan nilai *hash* dari kunci publik dan semua informasi dari subjek, kemudian CA akan melakukan enkripsi nilai *hash* tersebut dengan kunci privat CA (hasilnya adalah tanda tangan CA). Sertifikat digital ini tidak bersifat rahasia dan disimpan ke dalam *certificate repositories* oleh CA.

D. Mekanisme Challenge and Response

Mekanisme *challenge and response* merupakan sebuah mekanisme untuk memverifikasi pemilik dari sertifikat digital. Langkah pertamanya adalah *client* mengirim *challenge* ke *server* berupa sebuah *string* acak yang panjangnya 128 bit, maksud dari *challenge* tersebut adalah *client* meminta *server* untuk mengenkripsi *string* tersebut menggunakan kunci privat *server*. Hasil enkripsi, atau cipherteks, tersebut kemudian dikirimkan kembali kepada *client*. *Client* akan mendekripsi cipherteks tersebut dengan kunci publik *server*. Apabila hasil dekripsi dan *string* acak pada awal sama, maka *server* asli dan terverifikasi.

E. Kubernetes

Kubernetes, atau dapat dikenal juga sebagai K8s, merupakan sebuah sistem yang *portable*, *extensible*, dan bersifat *open-source*, yang digunakan untuk melakukan automasi *deployment*, *scaling*, dan manajemen *containerized applications*. Kubernetes memiliki ekosistem yang besar dan

terus berkembang karena layanan, dukungan, dan alat-alatnya yang tersedia secara luas.

Gambar 4. Logo Kubernetes

Hal-hal yang disediakan oleh Kubernetes adalah sebagai berikut.

1. *Service discovery* dan *load balancing*

   Kubernetes dapat 'membuka' sebuah *container* dengan DNS atau alamat IP-nya. Apabila terlalu banyak *traffic*, yang berarti *client* yang mengakses DNS atau alamat IP tersebut, Kubernetes dapat melakukan *load balance* dan membagikan jaringan *traffic* tersebut agar *server* tetap stabil.

2. *Storage orchestration*

   Kubernetes mengizinkan pengguna apabila adanya keinginan untuk meningkatkan sebuah sistem penyimpanan secara otomatis.

3. *Rollout* dan *rollback* otomatis

   Kubernetes memperbolehkan pengguna untuk mengubah kondisi *container*.

4. *Bin packing* otomatis

   Dengan definisi seberapa banyak CPU dan memory (RAM) yang dibutuhkan untuk setiap *container*, yang disediakan oleh pengguna, Kubernetes dapat langsung mengalokasikan *container-container* tersebut ke *nodes*.

5. *Self-healing*

   Kubernetes secara otomatis akan *restart container* yang memiliki status gagal dengan mengganti *container* tersebut.

6. Manajemen Secret dan konfigurasi

   Kubernetes memberi pengguna kebebasan untuk menyimpan dan mengelola informasi sensitif, seperti kata sandi, token OAuth, kunci-kunci SSH, dan lain-lainnya. Selain itu, pengguna juga dapat melakukan *deploy* dan memperbarui Secret dan konfigurasi tanpa harus membuat *container* yang baru, dan membongkar Secret.

Kubernetes memiliki banyak komponen, tetapi untuk implementasi pemasangan sertifikat digital pada situs web, komponen-komponen yang perlu diperhatikan adalah Deployment, Pod, Service, Ingress, Secret (opsional), dan ConfigMap (opsional). Penjelasan secara singkat untuk masing-masing komponen adalah sebagai berikut.

1. Namespace

   Tujuannya adalah untuk *environment* yang digunakan oleh banyak pengguna, seperti di berbagai tim atau projek. Namespace menyediakan cakupan untuk nama-nama. Nama dari *resource* harus unik di dalam suatu Namespace, tetapi tidak harus unik jika berada di beda Namespace. Di dalam Namespace tidak terdapat Namespace dan setiap *resource* Kubernetes harus berada di dalam satu Namespace saja.

2. Deployment

   Menyediakan pembaruan deklaratif untuk Pod. Keadaan yang diinginkan dideskripsikan pada Deployment, yang kemudian Controller Deployment akan mengubah keadaan aktual menjadi keadaan yang diinginkan.

3. Pod

   Unit komputasi terkecil yang dapat digunakan untuk membuat dan mengelola Kubernetes.

4. PersistentVolumeClaim (PVC)

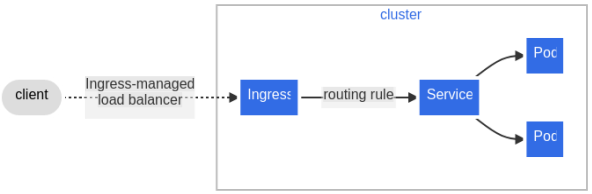
   Merupakan sebuah permintaan untuk tempat penyimpanan oleh pengguna. PVC dapat melakukan permintaan untuk ukuran secara spesifik dan *access modes* (ReadWriteOnce, ReadOnlyMany, atau ReadWriteMany).

5. Service

   Merupakan sebuah metode untuk *expose* sebuah jaringan aplikasi yang berjalan pada 1 atau lebih Pod. Tujuan utama dari Service adalah pengguna tidak perlu memodifikasi aplikasi yang sudah ada untuk menggunakan Service yang tidak dikenal sebelumnya. Service digunakan agar Pod akan terus tersedia pada jaringan dan dapat diakses.

6. Ingress

   Merupakan sebuah objek API yang mengelola akses eksternal dari Service dan pada umumnya adalah HTTP. Ingress membuka rute HTTP dan HTTPS dari jaringan luar ke Service dan *traffic routing* dikontrol oleh aturan-aturan yang didefinisikan pada Ingress.



Gambar 5. Cara Kerja Ingress

7. ConfigMap

   ConfigMap adalah sebuah objek API yang digunakan untuk menyimpan data nonkonfidensial dalam

pasangan *key-value*. Pod akan menggunakan ConfigMap dan dapat menganggap pasangan *key-value* tersebut sebagai *environment variables*, *command-line arguments*, atau *file* konfigurasi.

8. Secret

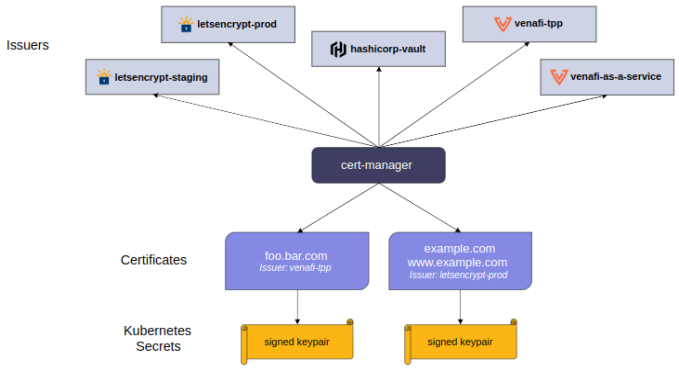
   Sebuah objek yang mengandung sejumlah data sensitif, seperti kata sandi, token, atau kunci. Menggunakan Secret berarti pengguna tidak perlu memasukkan data konfidensial pada kode program pengguna. Karena Secret dapat dibuat secara independen terhadap Pod, resiko yang perlu dihadapi terminimalisir. Perbedaan dari Secret dan ConfigMap adalah tujuan dari Secret adalah untuk menyimpan data konfidensial. Data pada Secret dienkripsi menggunakan Base64 Encode.

F. cert-manager

cert-manager adalah kontroler sertifikat X.509 untuk Kubernetes dan OpenShift. Kontroler ini akan mendapatkan sertifikat dari berbagai CA, baik dari CA publik maupun CA privat, menjamin bahwa sertifikat-sertifikat tersebut valid dan terkini, dan akan memperbarui sertifikat pada suatu waktu tertentu sebelum masa berakhir dari sertifikat.

Gambar 6. Logo cert-manager

cert-manager menambahkan sertifikat dan *certificate issuers* (CA) sebagai *resource* pada Kubernetes. Terdapat berbagai sertifikat yang dapat diperoleh oleh cert-manager, seperti Let's Encrypt, HashiCorp Vault, dan Venafi.



Gambar 7. Cara Kerja cert-manager

III. PERANCANGAN

Dengan menggunakan Kubernetes, berarti diperlukan suatu aplikasi yang sudah ada *image* dari suatu aplikasi dan pada perancangan ini sebaiknya menggunakan *image* yang sudah ada secara publik. Aplikasi yang akan digunakan adalah WordPress karena WordPress merupakan sebuah aplikasi situs web dan *image*-nya yang tersedia pada Docker Hub. *Tag* yang akan digunakan adalah 4.8-apache. Basis data yang digunakan oleh WordPress adalah MySQL, maka dari itu pada Kubernetes juga diperlukan untuk melakukan instalasi MySQL Untuk melakukan hal ini, dibutuhkan dua PVC untuk menyimpan data dari WordPress dan MySQL, dua Deployment dan Service untuk masing-masing WordPress dan MySQL. Untuk kedua WordPress dan MySQL dibutuhkan sebuah Secret untuk menyimpan kata sandi dari *user* root MySQL, lalu sebuah ConfigMap untuk Wordpress untuk menyimpan nilai *host* basis data MySQL. Yang terpenting adalah Ingress, agar WordPress dapat dipasangkan sertifikat digital.

Untuk memasang sertifikat digital pada Kubernetes, digunakan cert-manager. Agar cert-manager dapat digunakan, diperlukan konfigurasi terlebih dahulu, dengan CA-nya Let's Encrypt. Konfigurasi pertama dilakukan untuk ClusterIssuer, dengan memberikan URL server ACME dan yang terpenting di sini adalah sebelumnya pengguna sudah harus memiliki akun pada Cloudflare (salah satu fungsinya untuk menyimpan DNS). Konfigurasi ini dilakukan pada Namespace yang terpisah dan nama dari Namespace tersebut harus cert-manager dan ketika adanya aplikasi yang meminta sertifikat digital pada Namespace yang berbeda, cert-manager akan merespon walaupun berada di berbeda Namespace.

IV. IMPLEMENTASI

Hal yang pertama dilakukan adalah untuk melakukan konfigurasi untuk MySQL, dengan membuat PersistentVolumeClaim, Secret, Deployment, dan Service. Untuk skala pengujian pemasangan digital sertifikat saja, *storage* yang ditentukan untuk PVC sebanyak 1 GB saja.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: mysql-volume
 labels:
   app: wordpress
spec:
 resources:
   requests:
     storage: 1Gi
 accessModes:
   - ReadWriteOnce
```

Data yang dimasukkan ke dalam Secret adalah MYSQL_ROOT_PASSWORD, dengan nilainya adalah mysql1234. Sebelum memasukkan nilai tersebut, harus dienkripsi dengan Base64 Encode terlebih dahulu, sehingga nilainya adalah bXlzcWwxMjM0.

```
apiVersion: v1
kind: Secret
metadata:
 name: mysql-secret
type: Opaque
data:
 MYSQL_ROOT_PASSWORD: bXlzcWwxMjM0
```

Selanjutnya adalah Deployment untuk MySQL, dengan *image* dan *tag* yang digunakan adalah mysql:5.6. Hal-hal lainnya yang perlu diperhatikan adalah *port* yang didefinisikan pada Deployment ini adalah 3306 (*port default* dari MySQL), yang nantinya akan digunakan pada Service. Adanya VolumeMounts yang harus didefinisikan juga agar apabila terdapat *error* pada Pod MySQL, datanya tidak akan hilang dan tetap ada.

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: wordpress-mysql
 labels:
   app: wordpress
spec:
 selector:
   matchLabels:
     app: wordpress
     tier: mysql
 strategy:
   type: Recreate
 template:
   metadata:
     labels:
       app: wordpress
       tier: mysql
   spec:
     containers:
     - name: mysql
       image: mysql:5.6
       resources:
         limits:
           memory: 512Mi
           cpu: "1"
         requests:
           memory: 256Mi
           cpu: "0.2"
       envFrom:
       - secretRef:
           name: mysql-secret
       ports:
       - containerPort: 3306
         name: mysql
       volumeMounts:
       - name: mysql-persistent-storage
         mountPath: /var/lib/mysql
     volumes:
     - name: mysql-persistent-storage
       persistentVolumeClaim:
         claimName: mysql-volume
```

Yang terakhir adalah Service, yang akan menyatakan bahwa MySQL berjalan di Kubernetes dan tidak menggunakan ClusterIP karena hanya digunakan secara internal saja. Selain itu, situs web yang akan digunakan adalah WordPress, bukan MySQL. *Port* yang digunakan adalah 3306, sesuai yang sudah didefinisikan pada Deployment. Nama dari Service ini adalah wordpress-mysql, yang nantinya akan digunakan pada konfigurasi WordPress.

```
apiVersion: v1
kind: Service
metadata:
 name: wordpress-mysql
 labels:
   app: wordpress
spec:
 selector:
   app: wordpress
   tier: mysql
 ports:
 - port: 3306
 clusterIP: None
```

Selanjutnya adalah konfigurasi WordPress, yang membutuhkan PVC, Secret, ConfigMap, Deployment, dan Service. Secara singkat, fungsi PVC WordPress sama seperti PVC MySQL, sebagai berikut.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: wp-pv-claim
```

```
  labels:
    app: wordpress
spec:
  resources:
    requests:
      storage: 5Gi
  accessModes:
    - ReadWriteOnce
```

Selanjutnya adalah Secret, yang juga serupa dengan Secret MySQL, dengan perbedaannya adalah data yang tertera di Secret ini adalah WORDPRESS_DB_PASSWORD, yang menyatakan kata sandi dari MySQL yang harus diketahui oleh WordPress.

```
apiVersion: v1
kind: Secret
metadata:
  name: wp-secret
type: Opaque
data:
  WORDPRESS_DB_PASSWORD: bXlzcWwxMjM0
```

ConfigMap merupakan konfigurasi selanjutnya yang harus dilakukan karena mendefinisikan *host* dari basis data MySQL. Nilainya adalah wordpress-mysql, sesuai dengan nama Service dari MySQL.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: wp-configmap
data:
  WORDPRESS_DB_HOST: wordpress-mysql
```

Hal-hal yang perlu diperhatikan untuk Deployment WordPress juga sama seperti Deployment MySQL, dengan perbedaannya adalah *port*-nya 80 dan *path* dari VolumeMounts.

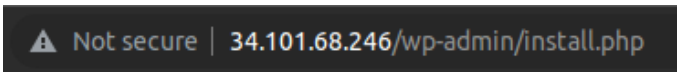
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - name: wordpress
        image: wordpress:4.8-apache
        resources:
          limits:
            memory: 512Mi
            cpu: "1"
          requests:
            memory: 256Mi
            cpu: "0.2"
        envFrom:
          - secretRef:
              name: wp-secret
          - configMapRef:
              name: wp-configmap
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wp-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wp-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
```

Konfigurasi terakhir yang perlu dilakukan adalah Service, dengan tipenya adalah LoadBalancer, agar dapat terekspos ke jaringan dan diakses oleh siapapun.

```
apiVersion: v1
kind: Service
metadata:
```

```
 name: wordpress
 labels:
   app: wordpress
spec:
 selector:
   app: wordpress
   tier: frontend
 ports:
   - port: 80
 type: LoadBalancer
```

Dengan menggunakan Service saja, pengguna sebenarnya sudah menyelesaikan tahapan untuk *deploy* WordPress. Alamat IP yang dihasilkan untuk mengakses WordPress bersifat acak. Tetapi hanya dengan menggunakan metode ini saja, situs web masih belum terpasang sertifikat digital, yang berarti koneksi ke situs web masih belum aman.



Gambar 8. Situs Web WordPress

Agar sertifikat digital dapat terpasang pada situs web tersebut, dibutuhkan cert-manager dan Ingress. Pertama-tama harus didefinisikan terlebih dahulu Secret untuk Issuer dan Issuer. Fungsi dari Secret adalah untuk menyimpan kunci API dari Cloudflare.

```
apiVersion: v1
kind: Secret
metadata:
 name: cloudflare-secret
type: Opaque
stringData:
 api-key: <API Key>
```

Setelah konfigurasi Secret, maka dilakukan konfigurasi untuk Issuer, dengan servernya adalah https://acme-v02.api.letsencrypt.org/directory. Let's Encrypt sebenarnya memiliki server ACME lainnya yang dapat digunakan untuk menguji apakah pemasangan sertifikat digital dapat dilakukan atau tidak, yaitu https://acme-staging-v02.api.letsencrypt.org/directory, tetapi di sini akan langsung menggunakan URL yang pertama untuk langsung menguji keberhasilannya. Issuer akan secara otomatis membuat sebuah Secret untuk menyimpan data konfidensial mengenai Let's Encrypt, yang di sini akan disimpan ke Secret yang bernama letsencrypt-prod.

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
 name: letsencrypt
spec:
 acme:
   server:
https://acme-staging-v02.api.letsencrypt.
org/directory
   email: 13519118@std.stei.itb.ac.id
   privateKeySecretRef:
     name: letsencrypt-prod
   solvers:
   - dns01:
       cloudflare:
         email:
my-cloudflare-acc@example.com
         apiKeySecretRef:
           name: cloudflare-secret
           key: api-key
```

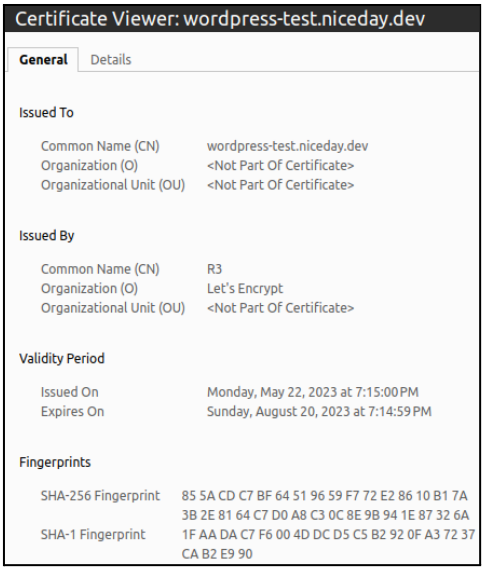
Setelah itu, Service untuk WordPress perlu diubah karena seterusnya aplikasi dari WordPress akan diekspos melalui Ingress, karena untuk pemasangan sertifikat digital hanya dapat dilakukan pada Ingress saja dan tidak pada Service. Perubahan Service adalah sebagai berikut.

```
apiVersion: v1
kind: Service
metadata:
 name: wordpress
 labels:
   app: wordpress
spec:
 selector:
   app: wordpress
   tier: frontend
 ports:
   - port: 80
     targetPort: 80
     protocol: TCP
 type: ClusterIP
```

Langkah yang terakhir adalah konfigurasi Ingress. Yang terpenting di sini adalah untuk menambahkan annotation agar Ingress dapat mengakses Issuer letsencrypt. Dalam Ingress dapat didefinisikan *host* (yang sebelumnya harus ditambahkan terlebih dahulu pada Cloudflare) dan di sini *host*, atau *domain*, yang akan digunakan adalah wordpress-test.niceday.dev.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
 name: wp-ingress
 annotations:
   cert-manager.io/cluster-issuer:
letsencrypt
spec:
 tls:
 - hosts:
     - wordpress-test.niceday.dev
   secretName: tls-wp-test
 rules:
 - host: wordpress-test.niceday.dev
   http:
     paths:
     - pathType: ImplementationSpecific
       backend:
         service:
           name: wordpress
           port:
             number: 80
```

Setelah ini, cert-manager akan membaca annotations dan membuat Certificate. Certificate nantinya akan membuat sebuah CertificateRequest, yang nantinya akan membuat sebuah Order. Pada Order, terjadi proses *challenge and response* untuk *domain* wordpress-test.niceday.dev. Apabila *challenge and response* berhasil, berarti Certificate berhasil diperoleh dari CA, yaitu Let's Encrypt.



Gambar 9. Sertifikat Digital https://wordpress-test.niceday.dev

Dengan begini, dapat dinyatakan bahwa pemasangan sertifikat digital berhasil dilakukan dan sudah ditandatangani oleh Let's Encrypt. Masa berlaku dari sertifikat digital Let's Encrypt adalah 90 hari dan cert-manager akan selalu memperbarui sertifikat digital sebelum masa berlakunya habis.

V. Kesimpulan

Pemasangan sertifikat digital pada situs web menggunakan Kubernetes dapat dilakukan dengan mudah. Alasan di balik penggunaan Kubernetes adalah karena salah satu keunggulan yang dimiliki oleh Kubernetes, yaitu pengelolaan yang mudah, sehingga aplikasi dapat di-*deploy* secara otomatis. Selain itu, dengan menggunakan cert-manager, pengguna juga dapat memasang sertifikat digital secara otomatis untuk aplikasi-aplikasi lainnya apabila konfigurasi cert-manager sudah dilakukan pada awal. Keamanan dari situs web juga terjaga karena selain adanya sertifikat digital, adanya Secret dari Kubernetes juga menjamin kerahasiaan data konfidensial dari suatu aplikasi. Contoh yang digunakan di sini hanyalah WordPress dan MySQL, tetapi pada nyatanya pasti ada lebih banyak data konfidensial.

Ucapan Terima Kasih

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya, sehingga penulis dapat menyelesaikan karya tulis dengan tepat waktu, dengan judul "Memasang Sertifikat Digital pada Situs Web Menggunakan Kubernetes". Harapannya adalah agar ilmu ini menjadi bermanfaat untuk pembaca. Ucapan terima kasih tak lupa pula kepada dosen pengampu mata kuliah IF4020 Kriptografi, Rinaldi Munir, atas segala ilmu dan pedoman yang telah diberikan. Tak lupa, ucapan terima kasih juga untuk teman-teman Penulis karena atas dukungannya dan semangatnya, karya tulis ini dapat diselesaikan dengan baik dan yang terakhir, ucapan terima kasih juga kepada diri saya sendiri karena dapat menyelesaikan karya tulis ini dengan sebaik mungkin dan tepat waktu.

Referensi

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/18-Kriptografi-Kunci-Publik-2023.pdf (diakses pada 21 Mei 2023, 23:48)

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/28-Tanda-tangan-digital-2023.pdf (diakses pada 21 Mei 2023, 23:25)

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/31-Sertifikat-digital-2023.pdf (diakses pada 21 Mei 2023, 22:53)

[4] https://kubernetes.io/ (diakses pada 22 Mei 2023, 00:10)

[5] https://kubernetes.io/docs/concepts/configuration/configmap/ (diakses pada 22 Mei 2023, 14:42)

[6] https://kubernetes.io/docs/concepts/configuration/secret/ (diakses pada 22 Mei 2023, 14:38)

[7] https://kubernetes.io/docs/concepts/overview/ (diakses pada 22 Mei 2023, 00:15)

[8] https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ (diakses pada 22 Mei 2023, 15:03)

[9] https://kubernetes.io/docs/concepts/services-networking/ingress/ (diakses pada 22 Mei 2023, 14:38)

[10] https://kubernetes.io/docs/concepts/services-networking/service/ (diakses pada 22 Mei 2023, 14:28)

[11] https://kubernetes.io/docs/concepts/workloads/controllers/deployment/ (diakses pada 22 Mei 2023, 14:24)

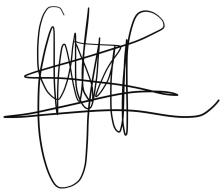
[12] https://kubernetes.io/docs/concepts/workloads/pods/ (diakses pada 22 Mei 2023, 14:23)

[13] https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/ (diakses pada 22 Mei 2023, 16:11)

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023



Cynthia Rusadi

13519118