

**iSIM KUBERNETES KURULUMU**

**(ONLINE)**

**Kurulum yapılacak ortamın bilgileri**:

ESXI içinde beş (5) adet makine olmalı ve bu makinelerin fiziksel yapısı şu şartları sağlamak zorundadır;

CPU : 4 vCPUs

GPU : 4 MB

RAM : 16 GB

HDD : 300 GB

Debian 10 yüklü ve SSH ile kurulum makinelerine erişimi olan harici bir makine ile kurulum başlatılacaktır.

**Gerekli dosyalar**:

Cluster oluşturulması için;

<https://jenkins.isimplatform.io/job/Deployment/job/iSIM_ISO/>

<https://bitbucket.org/isimplatform/k8s-auto-install/src/master/>

<https://bitbucket.org/isimplatform/fullisimstack-helm/src/master/>

<https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1>

iSIM Server’ın kurulması için bu dosyalara ek olarak kurulumun yapılacağı versiyona ait release ile gönderilen aşağıdaki paketlere ihtiyaç duyulmaktadır:

iSIM.Server

iSIM.Tools.DatabaseGenerator

iSIM.CDN.Server

VideoGW.NET

Ek olarak ihtiyaç duyulan paketler;

iSIM.Tools.EncryptionDecryption

**iSIM İÇİN KUBERNETES (K8s) CLUSTER OLUŞTURULMASI:**

**ISO’lar aşağıdaki şekilde makinelere yedirilir**;

|  |  |
| --- | --- |
| Windows | -en\_windows\_server\_2019\_x64 |
| Master | -isim-10-k8s-amd64-CD-1 |
| Slave1 | -isim-10-k8s-amd64-CD-1 |
| Slave2 | -isim-10-k8s-amd64-CD-1 |
| NFS | -isim-10-nfs-amd64-CD-1 |

**Harici Debian 10 Makine:**

Kurulumlar tamamlandıktan sonra Debian 10 yüklü SSH ile kurulum makinelerine erişimi olan başka bir makinedan root kullanıcısıyla şu işlemler yapılır:

1. Aşağıdaki komut ile gerekli paketler yüklenir.

*apt-get install ansible sshpass python3-winrm*

**Not:** Ansible versiyonu 2.7.x ve üzeri olmalıdır.

2. [https://bitbucket.org/isimplatform/k8s-auto-install/src/master/](https://bitbucket.org/isimplatform/k8s-auto-install/src/master/%20) üzerinden “*k8s-auto-install”* çekilir.

3. Script çalıştırılmadan önce kurulum yapılacak ortamın konfigürasyonuna göre düzenlenmelidir. Bunun için “*k8s-auto-install”* içerisinde yer alan “*isim\_hosts”* dosyası kurulum yapılacak ortamdaki makinelerin konfigürasyonuna göre düzenlenir.

**Not:** *isim\_hosts* içinde yer alan *repo* değeri ifadesi online kurulum yapılacağı zaman kaldırılır.

4. Bir çakışma yaşanmaması için “.ssh/known\_hosts” içindeki değerler önceden silinmelidir.

*echo > .ssh/known\_hosts*

5. Kurulumların tamamlanması ve cluster'ın oluşturulması için aşağıdaki komut çalıştırılır. Bu aşamadan sonra istenilirse aynı dizinde bulunan ve aynı şekilde kullanımı olan “*reset\_cluster.sh*” script’i ile cluster resetlenebilir.

*./scripts/create\_cluster.sh isim\_hosts*

**Not**: Docker Reposu müşteride olacaksa aşağıdaki komut çalıştırılır.

*ansible-playbook playbooks/repo.yaml -i isim\_hosts*

**Kubernetes Versiyon Güncellemesi**

Kurulumlar bitirildikten sonra istenirse cluster içindeki makinaların kubernetes versiyonları upgrade edilebilir. Bunun için öncelikle cluster’ın master makinasında bulunan mevcut paketlerin versiyonları kontrol edilmelidir. Master makinadan aşağıdaki komut çalıştırılır.

*dpkg -l | grep kube*

**Not:** Versiyonlanma isimlendirme şu şekilde okunmalıdır;

A.B.C paketi için;  
  
 A => Majör versiyonu

B => Minör versiyonu

C => Patch versiyonu

Yükseltilecek versiyonun bilgisi her seferinde minör versiyonu bir arttıracak şekilde harici debian makinasında bulunan isim\_hosts dosyasındaki ilgili alana yazılır ve aşağıdaki komut çalıştırılır.

*./scripts/upgrade\_cluster.sh isim\_hosts*

**Not:** 1.19.x > 1.20.x sonrası “*ansible-playbook -i isim\_hosts playbooks/mixedprotocol.yaml*” çalıştırılır. Bu işlemin ardından Master makinadan “*sudo cat /etc/kubernetes/manifests/kube-apiserver.yaml | grep -i mixed*” komutu çalıştırılıp true döndürdüğü control edilir.

**iSIM KURULUMU:**

Cluster oluşturulduktan sonra iSIM'ın çalıştırılması için gerekli olan işlemler:

1. Master olarak belirlenmiş makineye erişim sağlanır ve buraya; <https://bitbucket.org/isimplatform/fullisimstack-helm/src/master/> üzerinden “*fullstack-helm”* çekilir.

2. Script ile kuruluma başlamadan önce dizinde bulunan konfigürasyon dosyaları cluster’ın konfigürasyonuna ve açıklamalara göre düzenlenir. Bu açıklamaları ve bir konfigürasyon örneğini bir sonraki sayfada bulabilirsiniz. Aşağıdaki komut ile düzenlenmesi için bir *module.conf* dosyası oluşturulur.

*cp module.conf.example module.conf*

**Not:** Spesifik olarak IP ve versiyon set etmek için *module.conf* dosyasını düzenlemek gerekmektedir. Eğer bu bilgiler statik olarak eklenmek istenmiyorsa IP ve versiyon için “0” değeri verilir.

Örnek: elk=<ip>,<versiyon>

Statik olarak verilmesi gereken modüller aşağıdaki gibidir. *Module.conf* dosyası örneği dökümanın ilerisinde vardır.

isimclient : Client’ın açılması gereken IP yazılır.

coturn : Kurulumun başarılı bir şekilde tamamlanması için statik bir IP alması yazılır.

kurento : Kurulumun başarılı bir şekilde tamamlanması için statik bir IP alması yazılır.

kurentotestui : Kurulumun başarılı bir şekilde tamamlanması için statik bir IP alması yazılır.

isimdbstr : Bu değer için iSIM Encryption Decryption programına aşağıdaki girdi yazılır;  
  
*data source=*<Server Makinası>*;initial catalog=isimdb;integrated security=false;persist security info=True;User ID=sa;Password=!2345qawsedrf;Min Pool Size=16*

3. Gerekli düzenlemeler yapıldıktan sonra aşağıdaki komut ile configmap yaratılır.

*kubectl apply -f metallb-config.yaml*

4. Gerekli düzenlemeler yapıldıktan sonra aşağıdaki komut ile multusnet için belirlenmiş konfigürasyon kubernetes konfigürasyonuna tanımlanır ve script ile kurulum aşamasına geçilir.

*kubectl apply -f multus-daemonset.yml*

***module.conf***

*module.conf* düzenlenirken kullanılmayan paketlerin başına “#” yazarak kurulum scripti tarafından kurulması engellenebilir. Paket yanına istenirse alması istenilen statik IP ve versiyon bilgileri boşluk bırakmadan ve virgül ile ayrılacak şekilde yazılır. "multusnet” alanı düzenlenirken önemli olan şey cluster tarafından kullanılmayan iki IP’nin belirlenmesidir. ipstart-ipend,subnet,gw şekilde aralarında boşluk olmadan düzenlenmelidir. “dmiintegstring” ve “haydaintegstring” alanlarına ise ilgili paketlerin çalıştığı adresler yazılır. “domainname” alanı iSIM uygulamasının yayınlanacağı site adresini belirlemek için kullanılır.

#module=ip,version <- if version is 0 latest version will be installed, if ip is 0, automatic ip will be used

bluetooth-engine=0,1.0.584

bluetooth-integrator=0,1.0.353

#dms-integration=0,1.0.197

dms-engine=0,1.0.378

dms-integortana=0,1.1.48

#dms-integintetra=0,1.0.19

nova=0,1.0.444

cabinet-engine=0,1.0.390

cabinet-integration=0,1.0.327

hayda-engine=0,1.0.433

hayda-integration=0,1.0.334

#hayda-integortana=0,1.0.67

dmi-emulator=0,0.5.21

bluetooth-scenario-manager=0,0.5.26

#bluetooth-emulator=0,0.5.117

#hayda-emulator=0,0.5.24

postgis=10.194.18.155,1.0.44

mongo=10.194.18.154,1.0.131

#videogwdb=0,1.0.42

#isimdb-postgresql=0,1.2.42

imanager=0,0.11.21

plato=0,0.4.27

optimusprime=0,0.4.1

#orochimaru=0,#70

isimcore=0,0.1.67

isimclient=10.194.18.150,1.5.123

isim-print-server=0,1.0.30

arangodb=0,1.0.87

kafka-connect=0,1.0.87

rest-proxy=0,1.0.87

weyland=0,1.0.87

zookeeper=0,1.0.87

mosquitto=0,1.0.87

kibana=0,1.0.87

elasticsearch=0,1.0.87

redis=0,1.0.87

librenms=0,1.0.87

logstash=0,1.0.87

isimdebugui=0,1.0.8

isimcdn=0,1.0.12

riemannnvr=0,0.1.67

isimbackup=0,1.0.4

kurento=10.194.18.151,1.0.111

coturn=10.194.18.152,1.0.111

kurentotestui=0,1.0.10

videogw=10.194.18.153,1.2.463

geoserver=0,1.0.422

erebor=0,1.0.197

nazgul=0,1.0.151

cthulhu=0,1.0.23

nominatim=0,1.0.36

#kibanadash=0,1.2.1361

riemann=0,1.15.28

riemannui=0,1.4.99

riemanngo=0,0.17.7

zonemanager=0,1.0.287

#cdn=0,2.20.199

#isimcontentwall=0,1.3.46

#videogwdotnet=0,1.2.556

eventintegmstone=0,1.0.1

reimdall=0,0.2.13

ntpproxy=0,1.0.22

videogwinteg=0,0.1.33

# No commas here!!!

isimdbstr=RVo9hEV00fUl1rL9oWrB0E4n9c1xjqm4WdouPcKKRSA30gTRwkJfhPrdEJjRqwL6T4dQ+gXQnjiiimSqa1OPvcgVkLYjbIrKmah94Qr63LNHD5kEj1b5/45X0nvNA5BO3UX1et2YXEMhSBYiDdpJKrWfCI+n/oKojpZzmoxUl6pJrC5fXbZdSX4J+Nmn4Mb0N4XO3XXSkdk/Os5ewy0NvA2GFRbuLVziS5a+5MMqZuo2IlmMSEq6z/Zxd3RqLajOfHVbpxt64WAd0eBe0whwfR/2P6DEZx50XHmalH/DVJU=

dmiintegstring=brand%Proline,model%default,url%http://10.194.18.140:9002

haydaintegstring= brand%Proline,model%default,url%http://hayda-integration.default.svc.isim.internal:4002

connectors=cabinet\_datas,weathersensor\_datas,bluetooth\_travel\_averages,maintenance\_topic,bluetooth\_devices,cabinet\_alarms,bluetooth\_travels

domainname=isim.carpark.security

ntpserver=ntp.pool.org

sslenabled=no

prometheus=yes

# Needed for RIEMANNNVR, example: ipstart-ipend,subnet,gw

multusnet=10.194.18.179-10.194.18.180,10.194.18.0/24,10.194.18.1

# isim DB backups

# Options: no -> no backups, yes only backup to master, NFS ip -> backup to NFS

isimbackups=yes

***metallb-config.yaml***

*metallb-config.yaml* düzenlenirken, “isim-static-ip-space” bölümünde bulunan “addresses” alanına cluster’da iletişim kurulan makinalar (Windows makine dahil) ve *module.conf*’ta IP verilen paketlerin IP aralığını kapsayacak şekilde bir aralık yazılır. “isim-ip-space” bölümünde bulunan “adresses” alanına ise pod’ların otomatik IP alacağı aralık yazılır.

apiVersion: v1

kind: ConfigMap

metadata:

namespace: metallb-system

name: config

data:

config: |

address-pools:

- name: isim-static-ip-space

protocol: layer2

addresses:

- 10.194.18.140-10.194.18.155

auto-assign: false

- name: isim-ip-space

protocol: layer2

addresses:

- 10.194.18.156-10.194.18.178

***multus-conf.yaml***

*multus-conf.yaml* 4. adımdaki işlemin ardından ilk start verildiğinde otomatik olarak oluşur. Bu IP’ler *module.conf* içinde bulunan multusnet alanından çekilir. Sırasıyla “subnet”, “gateway”, “rangeStart” ve “rangeEnd” alanlarına parse edilir.

apiVersion: "k8s.cni.cncf.io/v1"

kind: NetworkAttachmentDefinition

metadata:

name: isimk8svlan2-conf

spec:

config: '{

"cniVersion": "0.3.1",

"type": "macvlan",

"capabilities": { "ips": true },

"master": "ens192",

"mode": "bridge",

"isDefaultGateway": true,

"ipam": {

"type": "host-local",

"subnet": "10.194.18.0/24",

"rangeStart": "10.194.18.179",

"rangeEnd": "10.194.18.180",

"routes": [

{ "dst": "0.0.0.0/0" }

],

"gateway": "10.194.18.1"

}

}'

**Script İle Kurulum Aşaması**

Script sadece kurulum yapmaya değil aynı zamanda servisleri durdurmaya, bir önceki çalışan servislerden kalan datayı temizleyerek sıfır kurulum yapılması için ortam hazırlanmasına ve çalışan servislerin durumları hakkında bilgilerde verir.

**Not:** *install\_isim.sh* scripti için parametreler ve görevleri;

*addrepo* : iSIM reposunu Helm reposuna taşır.

*start* : Kurulumu başlatır ve servisleri ayağa kaldırır.

*stop* : Çalışan servisleri durdurur.

*status* : Servislerin durumu hakkında bilgi verir.

*deletedata* : Helm reposunu siler.

*upgrade* : Ayağa kalkmış olan servislerin çalışma parametrelerini günceller.

*cleanup* : Script çalıştırıldığında ortaya çıkan değişiklikleri geri alır.

5. Aşağıdaki komut ile iSIM reposu Helm reposuna tanımlanır.

*./install\_isim.sh addrepo*

6. Aşağıdaki komut çalıştırılarak kurulum tamamlanır.

*./install\_isim.sh start*

**Not:** Script ile bütün paketler kurulacaktır. İstenilirse paketlere manuel olarak da müdahale edilebilir. Bunun için helm komutlarını kullanabilirsiniz. Bu komutlar “Genel Komutlar” başlığı ile dökümanın ilerisinde yer almaktadır.

7. Aşağıdaki komut ile ayağa kalkan servislerin çalıştığı IP’ler listelenir. iSIM client tarafında konfigürasyon değerlerini bu IP’lere göre düzenleyin.

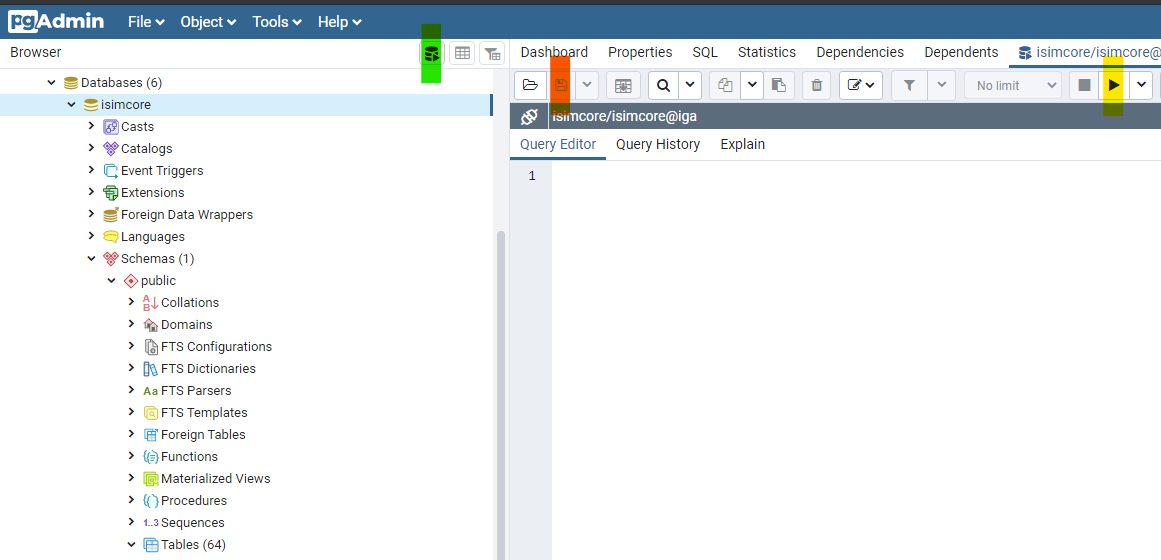
*watch kubectl get services*

**iSIM DB Config Yapılandırması**

1. *module.conf* dosyasının içinde “postgis” alanına verilen IP kullanılarak postgresql veri tabanına bağlanılır.

**Not:** Default şifre “isim123456”

2. Release’de bulunan isimconfig-x.x.x.sql doysası notepad++ ile açılır ve dosya içinde bulunan “testdemo.isimplatform.io” geçen text, *module.conf* dosyası içinde bulunan “domainname” alanında bulunan text ile değiştirilir.



3. Bağlanılan veri tabanında “isimcore” database’i seçiliyken görselde yeşil olarak işaretlenmiş butona basılarak Query editör açılır.

4. Değiştirilmiş text buraya yapıştırılır ve sarı olarak işaretlenmiş butona basılarak query çalıştırılır. Bu sayede “tbl\_configuration” tablosu oluşturulur.

**Konfigürasyon Tablosunda Yapılacak Değişiklikler:**

Konfigürasyon değerlerini düzenlemek için listelenen servisler arasından isimclient IP’si ile Client’a Chrome tarayıcısından erişim sağlayıp sol menüde Maintenance altında bulunan Configurations seçeneğini seçin. Açılan sayfada ilgili Key üzerinde arama yaparak ilgili Value değişimini yapabilirsiniz.

**Key: Servis Adı:**

RiemannNgUrl riemann

RiemannTunnelsRestAPI riemann

KurentoServerUrl kurento

IceServers coturn

EreborUrl erebor

EreborCdnUrl erebor

TrafficViolationAlarmServiceAddress nova

AlarmWebSocket nova

ElasticSearchUrl elasticsearch

KibanaHaydaReportsUrl kibanadash

KibanaBluetoothReportsUrl kibanadash

KibanaAlarmReportsUrl kibanadash

KibanaSystemReportsUrl kibanadash

KibanaUserReportsUrl kibanadash

DmsManagement dms-engine

FieldControlUnitApiUrl dms-engine

Routes bluetooth-engine

HAYDAApiUrl hayda-engine

CdnUrl cdn

PlatoApiUrl plato

ntpClientApiUrl ntpserver

PrintServerApiUrl isim-print-server

iManagerApiUrl imanager

ZoneService zonemanager

KafkaServer kafka-connect

WorkflowUiUrl riemannui

**Genel Komutlar**

**Kubernetes (K8s) Komutları:**

*kubectl get pod* : Tüm podlar listelenir.

*kubectl get all -o wide* : Genel durum çıktısı gösterilir.

*kubectl describe* <pod> : Belirtilen podun durumu hakkında tüm bilgiler gösterilir.

*kubectl logs* <pod> : Belirtilen podun loglarını incelemek için kullanılır. Komuta *-f* parametresi eklenerek çıktısı devamlı akacak şekilde izlenme gerçekleştirilebilir.

**Helm Komutları:**

*helm list* : Helm reposundaki paketleri listeler.

*helm repo update* : Yeni versiyonları çekerek reposunu günceller.

*helm install* <paket adı> : Manuel olarak bir paketi kurmaya yarar.

*helm search* <paket adı> : O isimde bir paket olup olmadığını arar.

*helm upgrade* <paket adı> : Çalışan bir paketin güncellenmesini sağlar.

*helm delete* <paket adı> : Bir paketi helm reposundan kaldırır.

**TOOLTIPS**

Git logu bilgisi için => *git log --name-status HEAD^..HEAD*

IP bilgisi için => *ip addr*

Gateway bilgisi için => *ip r*

DNS / Nameserver için => *grep nameserver /etc/resolv.conf*

Versiyon bilgisi için => *dpkg -l | grep kube*

Cluster network bilgisi için => *kubectl describe Network-attachment-definition*

Servisleri listelemek için => *kubectl get services*

Configmap bilgileri için => *kubectl get configmap*

Podları listelemek için => *kubectl get pods -o wide*

Podun içine girmek için => *kubectl exec -it* <POD ADI> *-c /bin/bash*

Paketin configmap’ı için => *kubectl describe configmap* <paket adı>

**Hızlı Erişim Komutları:**

Podun parametreleri için => *kubectl describe pods* <POD ADI>

\* *kubectl describe pods $(kubectl get pods | awk '{print $1}' | grep* <POD ADI>*)*

Pod’un loguna erişmek için => *kubectl logs* <POD ADI>

\* *kubectl logs $(kubectl get pods | awk '{print $1}' | grep <POD ADI>) -f*

Pod CPU/Memory kullanımı => *kubectl top pod* <POD ADI>

\* *kubectl top pod $(kubectl get pods | awk '{print $1}' | grep <POD ADI>)*

*kubectl exec -i -t -n default isimcdn-0 -c isimcdn "--" sh -c "clear; (bash || ash || sh)"*

**Özellikli Kullanımlar:**

Manuel olarak bir pod’un configmap ve env değerleri değiştirilmişse, podun bu yapılan değişiklikler ile birlikte tekrar ayağa kalkabilmesi için öncelikle replica sayısının 0’a indirilip tekrar istenilen sayıya çıkartılması gerekmektedir. Aşağıda bulunan örnekte replica sayısı 1 olan bir pod restart atılarak konfigürasyon değişiklikleri ile ayağa kaldırılmıştır.

Pod’u restart etmek için sırasıyla aşağıdaki komutlar çalıştırılmalıdır.

*kubectl scale deployment* <POD ADI> *--replicas 0*

*kubectl scale deployment* <POD ADI> *--replicas 1*

Sertifikasyon ile ilgili bir problem durumunda yapılacak işlemler

*kubectl get secrets*

*kubectl describe secrets* <SECRET>

*kubectl edit secrets* <SECRET>

ESC ile Vi normal moda geçirilir ve “:q” yazıp (sol altta gözüküyor) Enter’lanarak çıkılır.

Sertifikanın yeninden yaratılması lazım. Sertifika ilk kurulumda yaratılıyor. Durumu çözünce buraya çözümü detaylıca açıklayacak şekilde ekleyeceğim.

**Kullanışlı Senaryolar:**

Spesifik olarak helm reposundan kaldırılmış bir paketi tekrar kurmak için:

1. Önce mutlaka helm reposu update edilmelidir.

*isim@master:~$ helm repo update*

2. Önce son kurulum loglarının kayıt edildiği dizine gidilir

*isim@master:~$ cd fullisimstack-helm/*

3. Bu dizinde aşağıdaki komut çalıştırılarak en son oluşturulmuş log dosyası tespit edilir (Komut çıktısındaki en aşağıdaki log uzantılı dosya)

*isim@master:~/fullisimstackhelm$ ls -latr*

4. İlgili log dosyasının içindeki ilgili pakete yönelik "helm install" komutu tespit edilir.

*isim@master:~/fullisimstackhelm$ grep <PAKET ADI> isim\_installation\_X\_X.log*

5. Komut çalıştırılarak kurulum tamamlanır.

*isim@master:~/fullisimstackhelm$ helm install chartmuseum/*<PAKET ADI> .........

Manuel olarak bir paketin kurulması için aşağıdaki komut parametreleri ile çalıştırılmalıdır.

*helm install* <paket adı> *--repo chartmuseum --name-template* <Kurulum ismi> *--version* <versiyon> *--set=service.loadBalancerIP=*<IP> *--set=image.env.isimServer=http://*<isimserverip>*/server*

Birden fazla paketi aynı anda kurmak için (örneğin emulatörler) *--name-template* parametresine her kurulum için farklı isimler verilmelidir. Bu şekilde bir paket bir ortama birden fazla sayıda çalışacağı şekilde kurulabilir.

Emulator kurmak için (yeni versiyonların çekilmesi için helm install komutundan önce helm repo update yapılması gereklidir)

*helm repo update*

*helm install chartmuseum/bluetooth-emulator --name-template bemu1 --set=service.loadBalancerIP=*<IP> *--version* <VERSION>

*helm install chartmuseum/dmi-emulator --name-template demu1 --set=service.loadBalancerIP=*<IP> *--version* <VERSION>

*helm install chartmuseum/hayda-emulator --name-template hemu1 --set=service.loadBalancerIP=*<IP> *--version* <VERSION>

*helm install chartmuseum/bluetooth-scenario-manager --name-template bsm --set=service.loadBalancerIP=*<IP> *--version* <VERSION>

Evicted podların temizlenmesi için aşağıdaki komut parametreleri ile çalıştırılmalıdır.

*kubectl get pods --all-namespaces -o json | jq '.items[] | select(.status.reason!=null) | select(.status.reason | contains("Evicted")) | "kubectl delete pods \(.metadata.name) -n \(.metadata.namespace)"' | xargs -n 1 bash -c*

**DB Şifreleri:**

PostgreSQL

Password: “isim123456”

User: “osm”,

Password: “osm123\*.”

User: “videogw”,

Password: “vid123.”

User: “zonemanager”,

Password: “zone123.”

MSSQL

User: “sa”,

Password: “Ms123456.”

MongoDB

User: “bluetooth”,

Password: “Bl123456.”,

DB: “Bluetooth”

User: “cabinet”,

Password: “Ca123456.”,

DB: “cabinet”

User: “dms”,

Password: “Dm123456.”,

DB: “dms”

**Riemann Ayarları**

Kafka: weyland.default.svc.isim.internal:9092

Kafka-connect: http://<connect-ip>:8083

**Windows Makine Kurulumu**:

1. Windows makinede administrator olarak powershell’i açıp aşağıdaki komutlar sırasıyla girilir.

*Install-Module -Name DockerMsftProvider -Repository PSGallery -Force*

*Install-Package -Name Docker -ProviderName DockerMsftProvider*

*Restart-Computer -Force*

2. Aşağıdaki metnin tamamı komut satırına girilerek *ConfigureRemotingForAnsible.ps1* scripti Windows makineye indirilir.

*Invoke-WebRequest -URI https://raw.githubusercontent.com/ansible/ansible/devel/examples/scripts/ConfigureRemotingForAnsible.ps1 -Outfile script.ps1*

3. Script indirildikten sonra aşağıdaki şekilde çalıştırılır.

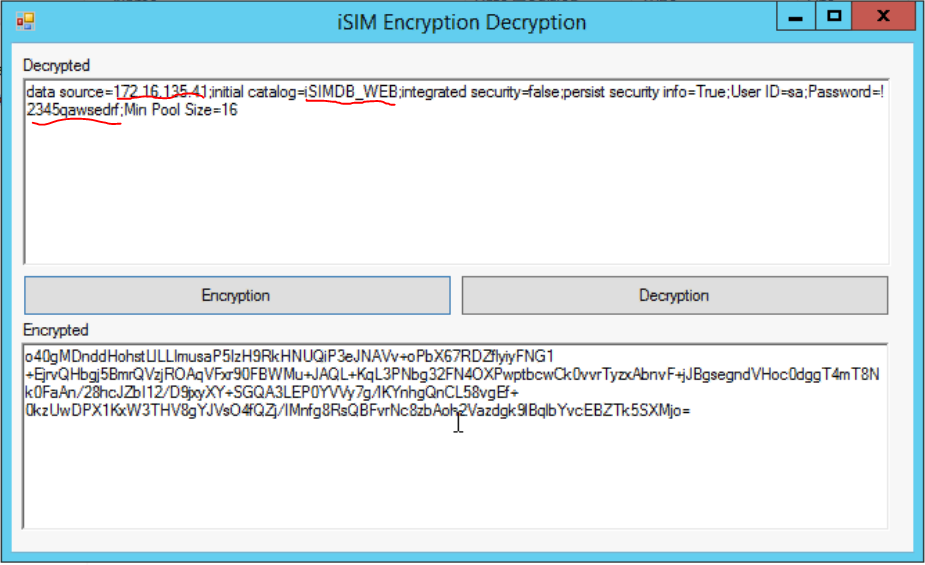
*./script.ps1*

**iSIM Server Kurulumu:**

1. Release paketi ile gönderilen iSIM.Server dosyası windows makineye indirilir.

2. Server’ın kurulacağı hosta remote bağlantı yapılır. iSIM.Server dosyasının içindekiler kopyalanır ve  C:\inetpub\wwwroot\isim\server pathine gidilerek kopyalanan dosyalar yapıştırılır.

3. *web.config* dosyası notepad ile açılır. *Connectionstring* değeri *iSIM Encryption Decryption* programına yapıştırılır. Aşağıda işaretlenmiş alanlar düzenlenir ve encrypt butonuna tıklanır. Encrypt edilmiş değer ConnectionString değerine yapıştırılır ve web.config dosyası save edilir.



4. web.config dosyasında aşağıdaki parametreler ayarlanmalıdır:

<add key="UseLog4NetForUserLogging" value="True" />

5. ELK ip ve portu verilmelidir:

<appender name="SystemLogAppender" type="log4net.ElasticSearch.ElasticSearchAppender, log4net.ElasticSearch"> <connectionString value="Server=<ELK’nın IP’si>;Index=system\_log;Port=<ELK’nın Portu>;rolling=true" />

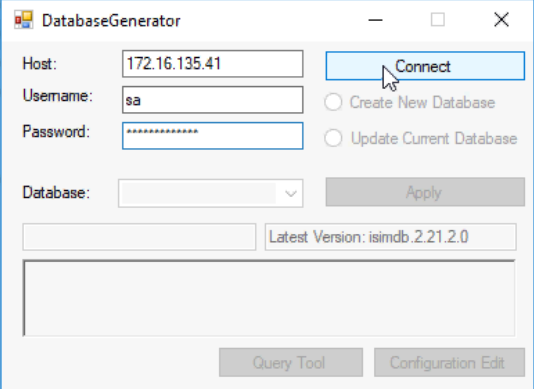
<appender name="UserLogAppander" type="log4net.ElasticSearch.ElasticSearchAppender, log4net.ElasticSearch">

<connectionString value="Server=<ELK’nın IP’si>;Index=user\_log;Port=<ELK’nın Portu>;rolling=true" />

**Database Generator**

1. iSIM.Tools.DatabaseGenerator dosyası açılır ve iSIM.Tools.DatabaseGenerator application dosyası çalıştırılır.

2. Host, username ve password alanları doldurulduktan sonra Connect butonuna tıklanılır.

****

3. Database adı belirlendikten sonra Apply butonuna tıklanılır.

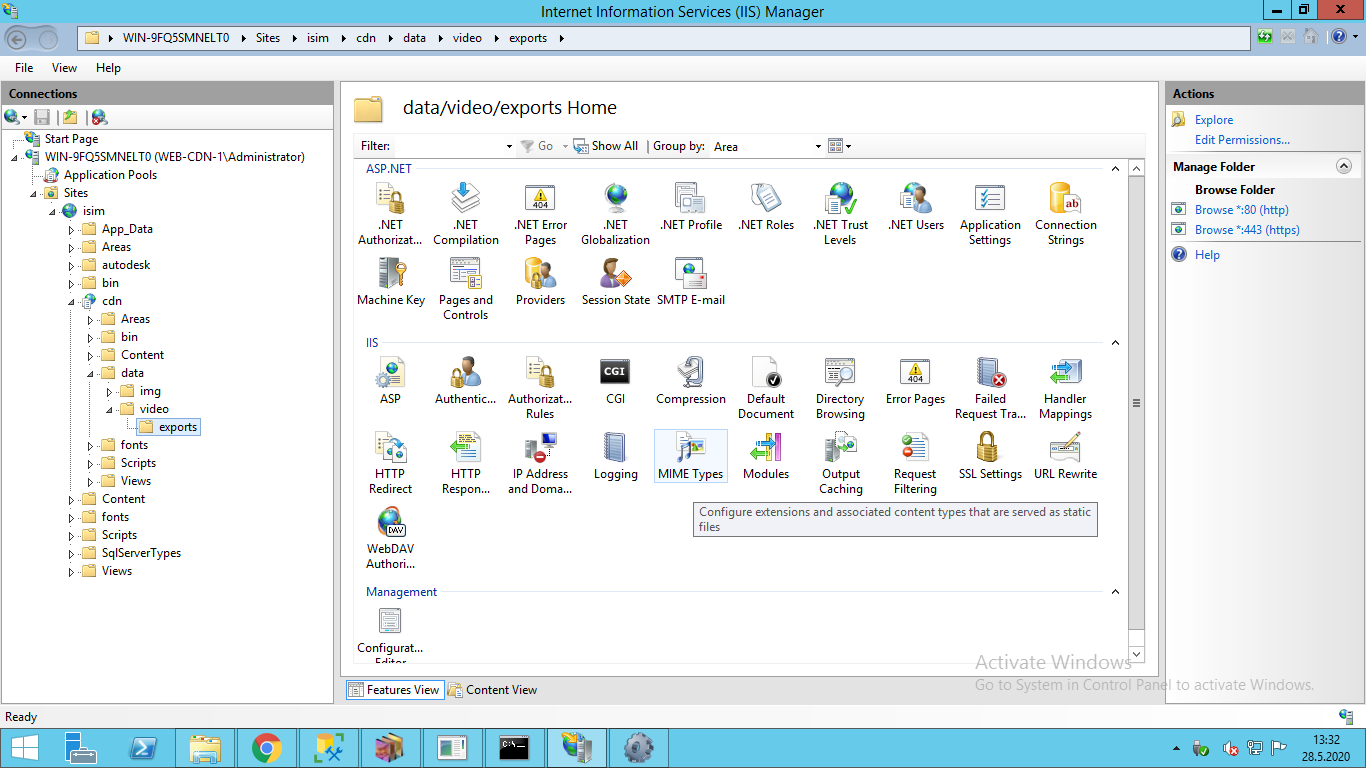
**iSIM CDN Kurulumu**

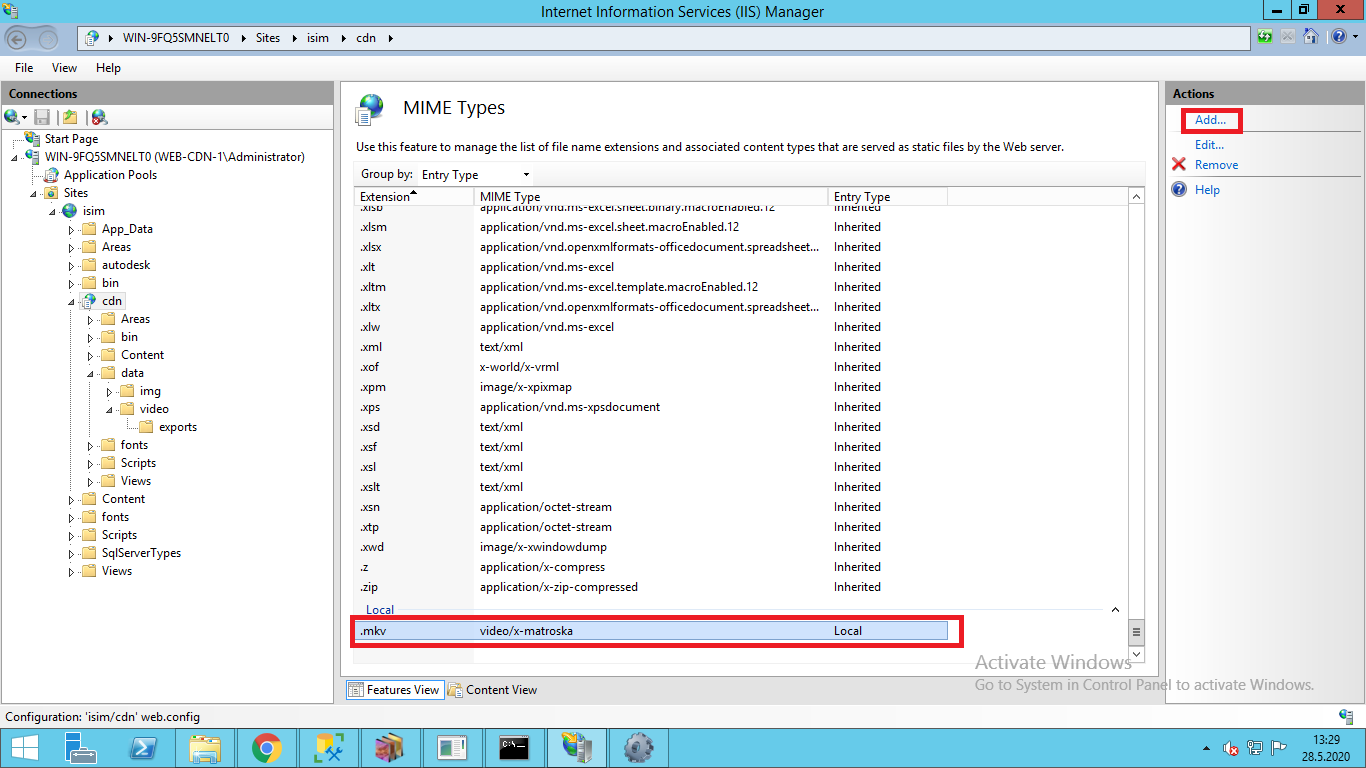
1. Windows server’a remote yapılır.

2. Release paket içinde gönderilen iSIM.CDN.Server folderının içindekiler kopyalanır.

3. C->inetpub->wwwroot->isim->cdn pathine gidilir. js ve data folderları haricindekiler silinir ve kopyalanan dosyalar yapıştırılır. Versiyon sogrulamak için [<Server>/cdn/api/version](http://172.16.135.66/api/version) adresini kullanınız.

4. MKV formatında Mime Type eklemek için aşağıdaki adımlar takip edilir:

****

****

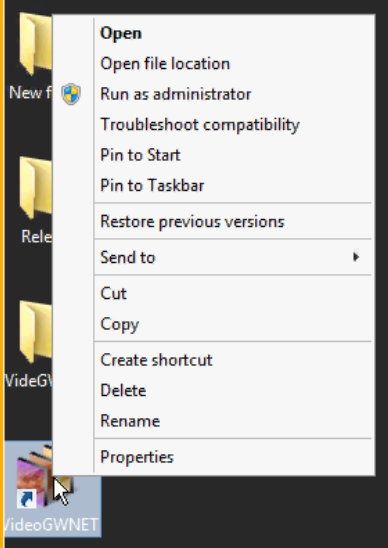
**VideoGW.Net Kurulumu:**

1. Server’a remote bağlantı yapılır.

3. Release paketinin içinde gönderilen VideoGW.NET exe’si buraya kopyalanır.

4. VideoGW.Net exe’si çalıştırılarak kurulum yapılır.

5. Open file location ile VideoGW.NET.exe config dosyası açılır.



6. Açılan dosya içerisinde <add key=”BaseAddress” value=”<http://localhost:9000/>” /> alanındaki localhost kaldırılır yerine server’ın IP adresi yazılarak Save denilir.

Örn: <add key="BaseAddress" value="http://172.16.135.55:9000/" />

7. <add key="CDNAddress" value="[http://CDNipadresi/cdn/](http://192.168.1.116/cdn/)"  /> alanındaki value kısmı kullanılan CDN e ait ip adresi ile doldurulur.

8. Uygulama çalıştırılır.