Process of enabling kubtctl logs feature

1. Make sure the **ca.crt** and **ca.key** files are in the kubernetes directory. If the kubernetes cluster is built up by kubeadm. Then, check the directory with the following command:
ls /etc/kubernetes/pki/

2. Set CLOUDCOREIPS env (must be created as root). This environment variable should be the cloudcore IP address.
Setting CLOUDCOREIPS needs the following command:
export CLOUDCOREIPS=”192.168.0.139”
(Warning: we must use the same “terminal” to continue our work, or we need to type this command again.)
Checking the environment variable with the following command:
echo $CLOUDCOREIPS

3. Generate the certificates for CloudStream on cloud node, however, the generation file is not in the /etc/kubeedge/, we need to copy it from the repository which was git cloned from GitHub.
Change user to root:
sudo su
Copy certificates generation file from original cloned repository:
cp $GOPATH/src/github.com/kubeedge/kubeedge/build/tools/certgen.sh /etc/kubeedge/
Change directory to the kubeedge directory:
cd /etc/kubeedge/
Generate certificates from **certgen.sh**
certgen.sh stream

4. It is needed to set iptables or firewall on the host. (It is referred to the KubeEdge document.)
Setting up the iptables for CloudStream (Port 10003 and 10350 are the default ports for the CloudStream and edgecore.):
(This command should be executed on every apiserver deployed node.)
(In my case, this is my master node, and execute this command by root.)
iptables -t nat -A OUTPUT -p tcp --dport 10350 -j DNAT --to $CLOUDCOREIPS:10003
Also, we need to clean up the iptables on Raspberry Pi 4 (Edge node) to make **kubectl logs** work.
iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X

Process of enabling kubtctl logs feature

1. Make sure the **ca.crt** and **ca.key** files are in the kubernetes directory. If the kubernetes cluster is built up by kubeadm. Then, check the directory with the following command:

   ls /etc/kubernetes/pki/

2. Set CLOUDCOREIPS env (must be created as root). This environment variable should be the cloudcore IP address.

   Setting CLOUDCOREIPS needs the following command:

   export CLOUDCOREIPS=”192.168.0.139”

   (Warning: we must use the same “terminal” to continue our work, or we need to type this command again.)

   Checking the environment variable with the following command:

   echo $CLOUDCOREIPS

3. Generate the certificates for CloudStream on cloud node, however, the generation file is not in the /etc/kubeedge/, we need to copy it from the repository which was git cloned from GitHub.

   Change user to root:

   sudo su

   Copy certificates generation file from original cloned repository:

   cp $GOPATH/src/github.com/kubeedge/kubeedge/build/tools/certgen.sh /etc/kubeedge/

   Change directory to the kubeedge directory:

   cd /etc/kubeedge/

   Generate certificates from **certgen.sh**

   certgen.sh stream

4. It is needed to set iptables or firewall on the host. (It is referred to the KubeEdge document.)

   Setting up the iptables for CloudStream (Port 10003 and 10350 are the default ports for the CloudStream and edgecore.):

   (This command should be executed on every apiserver deployed node.)

   (In my case, this is my master node, and execute this command by root.)

   iptables -t nat -A OUTPUT -p tcp --dport 10350 -j DNAT --to $CLOUDCOREIPS:10003

   Also, we need to clean up the iptables on Raspberry Pi 4 (Edge node) to make kubectl logs work.

   iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X

5. Modify “both” **/etc/kubeedge/config/cloudcore.yaml** and **/etc/kubeedge/config/edgecore.yaml** on cloudcore and edgecore. Set up cloudStream and edgeStream to enable: true.

Also, you have to change the server IP to your cloudcore IP (the same as $CLOUDCOREIPS).

Open the YAML file in cloudcore:

sudo nano /etc/kubeedge/config/cloudcore.yaml

Modify the file in the following part:

```
cloudStream:
    enable: true
    streamPort: 10003
    tlsStreamCAFile: /etc/kubeedge/ca/streamCA.crt
    tlsStreamCertFile: /etc/kubeedge/certs/stream.crt
    tlsStreamPrivateKeyFile: /etc/kubeedge/certs/stream.key
    tlsTunnelCAFile: /etc/kubeedge/ca/rootCA.crt
    tlsTunnelCertFile: /etc/kubeedge/certs/server.crt
    tlsTunnelPrivateKeyFile: /etc/kubeedge/certs/server.key
    tunnelPort: 10004
```

Open the YAML file in edgecore:

sudo nano /etc/kubeedge/config/edgecore.yaml

Modify the file in the following part:

```
edgeStream:
    enable: true
    handshakeTimeout: 30
    readDeadline: 15
    server: 192.168.0.139:10004
    tlsTunnelCAFile: /etc/kubeedge/ca/rootCA.crt
    tlsTunnelCertFile: /etc/kubeedge/certs/server.crt
    tlsTunnelPrivateKeyFile: /etc/kubeedge/certs/server.key
    writeDeadline: 15
```

6. Restart all the cloudcore and edgecore

*Warning: there are some containers which are related to the edgecore. Thus, we have to restart edgecore.service first and make those kube-proxy related containers “pending” or “pause”, and then kill those containers. Finally, we can restart edgecore again to make it work properly.

Before you start to restart any service, please turn your user into root. Since you have to possess the rights of accessing the docker and system services directly.

sudo su

cloudCore:

pkill cloudcore

nohup cloudcore > cloudcore.log 2>&1 &

edgeCore:

systemctl restart edgecore.service

docker ps

```
}root@raspberrypi:/home/pi# docker ps
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS              PORTS               N
AMES
d40ad407d6f9        kubeedge/pause-arm:3.1   "/pause"                 12 minutes ago      Up 12 minutes                           k
8s_POD_weave-net-fr9vx_kube-system_b30f1ab0-8cf2-448a-8588-ad191f5f9b37_13
f59bdacbfc48        b732e226e0f1             "/usr/local/bin/kube…"   12 minutes ago      Up 12 minutes                           k
8s_kube-proxy_kube-proxy-s8476_kube-system_5c0e5bc4-73ae-4daa-a0fd-d8d8713ddad5_14
f15a4b9aae7a        kubeedge/pause-arm:3.1   "/pause"                 13 minutes ago      Up 13 minutes                           k
8s_POD_kube-proxy-s8476_kube-system_5c0e5bc4-73ae-4daa-a0fd-d8d8713ddad5_13
2559412c2401        c1fd304a4885             "temperature-mapper"     13 minutes ago      Up 13 minutes                           k
8s_temperature_temperature-mapper-79f4989f65-5tswz_default_867db8c5-f606-45ca-9f61-7912fe4df84d_0
38ac0fc561b7        kubeedge/pause-arm:3.1   "/pause"                 13 minutes ago      Up 13 minutes                           k
8s_POD_temperature-mapper-79f4989f65-5tswz_default_867db8c5-f606-45ca-9f61-7912fe4df84d_0
```

docker container kill d40ad407d6f9

docker container kill f59bdacbfc48

docker container kill f15a4b9aae7a

systemctl restart edgecore.service

Use the following command to check if **kubectl logs** feature is activated or not on localhost (The Raspberry Pi 4 is the KubeEdge Edge node in my case).

Check if the edge stream feature is activated or not:

sudo netstat -tulpn | grep edgecore

```
pi@raspberrypi:~$ sudo netstat -tulpn | grep edgecore
tcp        0      0 172.17.0.1:4000         0.0.0.0:*               LISTEN      2613/edgecore
tcp        0      0 127.0.0.1:10350         0.0.0.0:*               LISTEN      2613/edgecore
udp        0      0 172.17.0.1:53           0.0.0.0:*                           2613/edgecore
```

Make sure if the your edge stream service can be accessed:

curl "http://localhost:10350/stats/summary?only_cpu_and_memory=true"

If you successfully activate the feature, you will see the following information:

```
root@raspberrypi:/home/pi# curl "http://localhost:10350/stats/summary?only_cpu_and_memory=true"
{
  "node": {
    "nodeName": "raspberrypi",
    "systemContainers": [
      {
        "name": "pods",
        "startTime": "2020-08-22T11:04:10Z",
        "cpu": {
          "time": "2020-08-23T02:29:35Z",
          "usageNanoCores": 12480180,
          "usageCoreNanoSeconds": 846749998105
        },
        "memory": {
          "time": "2020-08-23T02:29:35Z",
          "availableBytes": 8170373120,
          "usageBytes": 111263744,
          "workingSetBytes": 88150016,
          "rssBytes": 11223040,
          "pageFaults": 0,
          "majorPageFaults": 0
        }
      },
      {
        "name": "kubelet",
        "startTime": "2020-08-22T11:06:28Z",
        "cpu": {
```

```
    "time": "2020-08-23T02:29:34Z",
    "usageNanoCores": 72278481,
    "usageCoreNanoSeconds": 4337730875230
   },
   "memory": {
    "time": "2020-08-23T02:29:34Z",
    "availableBytes": 8796031279104,
    "usageBytes": 63361024,
    "workingSetBytes": 61739008,
    "rssBytes": 58212352,
    "pageFaults": 25766202,
    "majorPageFaults": 0
   }
  },
  {
   "name": "runtime",
   "startTime": "2020-08-22T11:04:09Z",
   "cpu": {
    "time": "2020-08-23T02:29:34Z",
    "usageNanoCores": 36197490,
    "usageCoreNanoSeconds": 1656552133365
   },
   "memory": {
    "time": "2020-08-23T02:29:34Z",
    "availableBytes": 8796035026944,
    "usageBytes": 120844288,
    "workingSetBytes": 57991168,
```

```
        "rssBytes": 39063552,
        "pageFaults": 12661506,
        "majorPageFaults": 198
      }
    }
  ],
  "startTime": "2020-08-22T11:04:09Z",
  "cpu": {
    "time": "2020-08-23T02:29:40Z",
    "usageNanoCores": 166651877,
    "usageCoreNanoSeconds": 10859131447776
  },
  "memory": {
    "time": "2020-08-23T02:29:40Z",
    "availableBytes": 7455789056,
    "usageBytes": 1336512512,
    "workingSetBytes": 802734080,
    "rssBytes": 150327296,
    "pageFaults": 39732,
    "majorPageFaults": 66
  }
},
"pods": [
  {
    "podRef": {
      "name": "kube-proxy-s8476",
      "namespace": "kube-system",
```

```
  "uid": "5c0e5bc4-73ae-4daa-a0fd-d8d8713ddad5"
},
"startTime": "2020-08-22T11:53:32Z",
"containers": [
 {
   "name": "kube-proxy",
   "startTime": "2020-08-22T11:53:43Z",
   "cpu": {
     "time": "2020-08-23T02:29:43Z",
     "usageNanoCores": 1063436,
     "usageCoreNanoSeconds": 103051749819
   },
   "memory": {
     "time": "2020-08-23T02:29:43Z",
     "availableBytes": 8796085219328,
     "usageBytes": 9015296,
     "workingSetBytes": 7798784,
     "rssBytes": 6275072,
     "pageFaults": 754347,
     "majorPageFaults": 0
   }
 }
],
"cpu": {
  "time": "2020-08-23T02:29:42Z",
  "usageNanoCores": 1099277,
  "usageCoreNanoSeconds": 111640636254
```

```
  },
  "memory": {
    "time": "2020-08-23T02:29:42Z",
    "availableBytes": 8796056662016,
    "usageBytes": 42168320,
    "workingSetBytes": 36356096,
    "rssBytes": 6246400,
    "pageFaults": 0,
    "majorPageFaults": 0
  }
},
{
  "podRef": {
    "name": "weave-net-fr9vx",
    "namespace": "kube-system",
    "uid": "b30f1ab0-8cf2-448a-8588-ad191f5f9b37"
  },
  "startTime": "2020-08-22T11:53:59Z",
  "containers": null,
  "cpu": {
    "time": "2020-08-23T02:29:32Z",
    "usageNanoCores": 0,
    "usageCoreNanoSeconds": 45122404150
  },
  "memory": {
    "time": "2020-08-23T02:29:32Z",
    "availableBytes": 8796046766080,
```

```
    "usageBytes": 57335808,
    "workingSetBytes": 46252032,
    "rssBytes": 0,
    "pageFaults": 0,
    "majorPageFaults": 0
   }
},
{
  "podRef": {
    "name": "temperature-mapper-79f4989f65-5tswz",
    "namespace": "default",
    "uid": "867db8c5-f606-45ca-9f61-7912fe4df84d"
  },
  "startTime": "2020-08-22T11:53:10Z",
  "containers": [
    {
      "name": "temperature",
      "startTime": "2020-08-22T11:53:12Z",
      "cpu": {
        "time": "2020-08-23T02:29:42Z",
        "usageNanoCores": 11894874,
        "usageCoreNanoSeconds": 654512498707
      },
      "memory": {
        "time": "2020-08-23T02:29:42Z",
        "availableBytes": 8796088307712,
        "usageBytes": 7143424,
```

```
            "workingSetBytes": 4710400,
            "rssBytes": 5926912,
            "pageFaults": 4092,
            "majorPageFaults": 0
          }
        }
      ],
      "cpu": {
        "time": "2020-08-23T02:29:43Z",
        "usageNanoCores": 11699776,
        "usageCoreNanoSeconds": 654543859283
      },
      "memory": {
        "time": "2020-08-23T02:29:43Z",
        "availableBytes": 8796087861248,
        "usageBytes": 7589888,
        "workingSetBytes": 5156864,
        "rssBytes": 5914624,
        "pageFaults": 0,
        "majorPageFaults": 0
      }
    }
  ]
}
```


Before you deploy metrics-server, you have to make sure that you deploy it on the node which has apiserver deployed on. In my case, that is my master node. As a consequence, I need to make master node

schedulable by the following command:

kubectl taint nodes --all node-role.kubernetes.io/master-

Then, in the deployment.yaml file, I must specify that metrics-server is deployed on master node.

(I used hostname as the marked label.)

In **metrics-server-deployment.yaml**

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values:
            - charlie-latest
```

Also, you need to check the metrics-server version if it is the latest one. Since the KubeEdge's kubelet is not the original, it uses the different port rather than the original kubelet. That is, metrics-server need to have the ability of searching other port automatically.

Prepare you image first.

Git clone latest metrics server repository:

git clone https://github.com/kubernetes-sigs/metrics-server.git

Get to the metrics server directory:

cd metrics-server

Make the docker image:

make container-arm64

Check if you have this docker image:

docker images

```
root@raspberrypi:/home/pi# docker images
REPOSITORY                                          TAG                                       IMAGE ID            CREATED
        SIZE
gcr.io/k8s-staging-metrics-server/metrics-server-arm64   6d92704c5a68cd29a7a81bce68e6c2230c7a6912   11439f12da24        4 days ago
        54.6MB
<none>                                              <none>                                    cafa4702d453        4 days ago
        1.41GB
kubeedge-temperature-mapper                         test                                      c1fd304a4885        12 days ago
        389MB
grafana/grafana                                     latest                                    9f31b3fc1ea3        2 weeks ago
        149MB
prom/prometheus                                     latest                                    72d6dda76d0b        2 weeks ago
        124MB
gcr.io/k8s-staging-metrics-server/metrics-server-arm     master                                    3289eeecc7ec        2 weeks ago
        50MB
weaveworks/weave-npc                                2.7.0                                     2d47a5fd0000        2 weeks ago
        36.7MB
weaveworks/weave-kube                               2.7.0                                     f58a4b249316        2 weeks ago
        90.1MB
k8s.gcr.io/kube-proxy                               v1.18.6                                   b732e226e0f1        5 weeks ago
        97.7MB
```

Make sure you change the tag of image by using its IMAGE ID to be compactable with image name in yaml file.

docker tag 11439f12da24 metrics-server-kubeedge:latest

Metrics server works finally!!!!!!!!]

```
root@charlie-latest:/var/log/kubeedge# kubectl get nodes
NAME              STATUS   ROLES          AGE   VERSION
charlie-latest    Ready    master         12d   v1.18.6
charlie-test      Ready    <none>         12d   v1.18.6
raspberrypi       Ready    agent,edge     12d   v1.18.6-kubeedge-v1.4.0
root@charlie-latest:/var/log/kubeedge# kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
temperature-mapper-79f4989f65-5tswz     1/1     Running   0          16h
root@charlie-latest:/var/log/kubeedge# kubectl top nodes
NAME              CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
charlie-latest    459m         7%     4608Mi          59%
charlie-test      126m         3%     1710Mi          45%
raspberrypi       204m         5%     751Mi           9%
root@charlie-latest:/var/log/kubeedge# kubectl top pods
NAME                                    CPU(cores)   MEMORY(bytes)
temperature-mapper-79f4989f65-5tswz     12m          4Mi
```

Hardware: Raspberry Pi 4

KubeEdge version: KubeEdge v1.4.0

Kubernetes version: v1.18.6

Running example: kubeedge-temperature-demo

Metrics-server version: latest-master-branch (Automatically searching port is 0.4.0 feature)

Raspberry Pi 4:

Performance units:

CPU: cores → usage nano cores

Memory: MiB (mebibytes) → 2^{20} Bytes → 1,048,576 Bytes

Q&A in slack channel

1. What I am trying to figure out here is that is there any latest version of metrics-server which is built for "ARM"?
   Latest image is available here gcr.io/k8s-staging-metrics-server/metrics-server:master It should be multi arch, so ARM should work with it.

2. Is there any other way to come-across this network issue?
   Each network issue is very diffent, you should check 192.168.0.128:10250 is the correct address where kubelet should be available and debug why it's not accessable

3. Are there any other tools to monitor the metrics without the network configuration?
   Depends what metrics do you want, metrics server serves very specific purpose of autoscaling based on CPU and Memory. You ca learn more here https://github.com/kubernetes-sigs/metrics-server/blob/master/README.md

serathius 4 hours ago

Ad 1 Lastest image is available here gcr.io/k8s-staging-metrics-server/metrics-server:master It should be multi arch, so ARM should work with it.

Ad 2 Each network issue is very diffent, you should check `192.168.0.128:10250` is the correct address where kubelet should be available and debug why it's not accessable

Ad 3 Depends what metrics do you want, metrics server serves very specific purpose of autoscaling based on CPU and Memory. You ca learn more here https://github.com/kubernetes-sigs/metrics-server/blob/master/README.md

Ref:

Git clone repository, certificates generation file, compilation process:

https://docs.kubeedge.io/en/v1.2.0/setup/setup.html

KubeEdge latest document:

https://docs.kubeedge.io/_/downloads/en/latest/pdf/

Slack channel for metrics-server in k8s:

https://kubernetes.slack.com/archives/C20HH14P7/p1597942669003600?thread_ts=1597797376.027100&cid=C20HH14P7

How to make k8s master node schedulable:

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

https://www.thegeekdiary.com/how-to-schedule-master-node-running-pod-service-as-a-worker-node/

KubeEdge example:

https://github.com/kubeedge/examples/tree/master/kubeedge-temperature-demo

Wikipedia-Memory-Units:

https://zh.wikipedia.org/wiki/Mebibyte