Deploying External S-TAP on an AWS EKS cluster

Before you begin

Make sure that the following components are installed and set up before you start with Amazon EKS:

1. AWS CLI – For information about downloading the AWS CLI, see https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
2. Kubectl – Used for communicating with the cluster API server. For information about installing Kubectl, see https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html

   Follow the instructions to install the Kubernetes Dashboard GUI as described in https://docs.aws.amazon.com/eks/latest/userguide/dashboard-tutorial.html
3. AWS-IAM-Authenticator – To allow IAM authentication with the Kubernetes cluster. For more information, see https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html
4. Create a docker hub user. For more information, see https://hub.docker.com/signup?next=%2F%3Fref%3Dlogin
5. Create a Guardium Collector in the cloud. For more information, see https://www-01.ibm.com/support/docview.wss?uid=swg27049576

Step 1: Set up the AWS CLI for AWS EKS Kubernetes Cluster

For information about creating an AWS EKS Kubernetes Cluster, see Additional information: Setting up the AWS EKS Cluster. This step assumes the AWS EKS exists.

1. Login to the Amazon server from CLI using IAM user Access ID (The same IAM that created the AWS EKS Cluster)

   ```
   > aws configure

     AWS Access Key ID [****************cd]: <yourID>
     AWS Secret Access Key [****************gRTY]:  <your Key>
     Default region name [us-east-1]:
     Default output format [json]:
   ```

2. Check the name of the login user:

```
>aws sts get-caller-identity
```

3. Create or update AWS EKS Kubernetes Cluster kubeconfig:

   ```
   >aws eks --region <region> update-kubeconfig --name <cluster_name>
   ```

   For example:

   ```
   >aws eks --region us-east-1 update-kubeconfig --name myinstance01
   ```

4. Test aws eks configuration with the `get svc` command:

   ```
   >kubectl get svc
   ```

Step 2: Get AWS EKS Kubernetes Cluster information

1. Get master URL of the AWS EKS Cluster

   ```
   >kubectl cluster-info
   ```

2. Get the admin access token of the AWS EKS Cluster

   ```
   >kubectl -n kube-system describe secret $(kubectl -n kube-system get
   secret | grep eks-admin | awk '{print $1}')
   ```

3. Create an Image Secret using Docker hub user account:

   ```
   >kubectl create secret docker-registry eksproxyacr \
     --docker-server=<DOCKER_REGISTRY_SERVER>\
     --docker-username=<DOCKER_USER> \
     --docker-password=<DOCKER_PASSWORD> \
     --docker-email=<DOCKER_EMAIL>
   ```

   Replace the parameters as follows:

   - DOCKER_REGISTRY_SERVER=docker.io
   - DOCKER_USER=yourlogin
   - DOCKER_EMAIL=youremail
   - DOCKER_PASSWORD=yourpassword

Step 3: Prepare Certificate to be used for External S-TAP

Create a certificate signing request (CSR) for each or multiple Guardium External S-TAPs. Generating the CSR also creates the token (the shared secret) that you

need to install the External S-TAP. The certificate is required for SSL enabled databases.

Most databases on AWS RDS don't require a common name (CN) match for certificate verification with SSL transport encryption, If CN match is required, use an asterisk wildcard (*) for the CN= parameter.

You need a certificate signed by a CA service for the production database, but for testing you can create a self-signed certificate.

To create a certificate:

1. From the Guardium CLI, run the create csr external_stap command:

```
cli>create csr external_stap
```

2. Respond to each question as follows:

```
Please enter the hostname as the alias used to identify this
certificate:
oracle12

What is the Common Name for this certificate (CN=) (Please enter the
name of the database server) ? oracle12

What is the name of your organizational unit (OU=) ? IBM

What is the name of your organization (O=) ? guardium

What is the name of your city or locality (L=) ? LT

What is the name of your state or province (ST=) ? MA

What is the two-letter country code for this unit (C=) ? US

What encryption algorithm should be used (1=DSA or 2=RSA. Default
'RSA') ?

Invalid input or no input. Using default 'RSA'

What is the keysize to use (1=1024 or 2=2048. Default '2048') ?

Invalid input or no input. Using default '2048'

Generating CSR...
```

```
**********************************************************************

Certificate Request:
Data:
Version: 0 (0x0)
Subject: C=US, ST=MA, L=LT, O=guardium, OU=IBM, CN=oracle12

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC5TCCAc0CAQAwWzELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAk1BMQswCQYDVQQHEwJMVDERMA8G
. . .
A7SDzqQfvAHGZtZvvPXmX2ILkVa1U5xAGsguObbny3mWhZK6StVWvkYi+YrF2oBFhOtm0qbobSM7
vFw72fRVc4UCYVMZsfXEAJaUpIZTkCGMXWwdfzfVKYJSZlGKGo5bToP5ekA9EV4KxzPfWKfjKA0I
ufVSVw==

-----END NEW CERTIFICATE REQUEST-----

**********************************************************************
```

3. Copy and paste the Certificate Signing Request (CSR) to a file, beginning with the `-----BEGIN NEW CERTIFICATE REQUEST-----` tag and ending with the `-----END NEW CERTIFICATE REQUEST-----` tag.
4. Send the CSR file to a Certificate Authority (CA) of your choice in order to obtain a valid certificate.

   Note: To import the certificate into that Guardium appliance, it must be in PEM format.

5. After you receive the certificate from your CA, use the following CLI command to start the import process:

   ```
   >store certificate external_stap
   ```

6. At the prompt, enter the entire alias, as follows (with the token as shown):

   ```
   oracle12 proxy_keycert b69f8f7d-d69d-11e9-9c8d-a44b66f1e859
   ok
   ```

7. Save the CSR into a file named `proxy.csr.`
8. After creating the CSR, send proxy.csr to a CA authority for signing.

For testing purposes only, use the following openssl command to create your own CA:

1. Create root CA:

   ```
   openssl genrsa -out rootCA.key 2048
   ```

```
openssl req -x509 -sha256 -new -key rootCA.key -days 3650 -out
rootCA.pem
```

2. Self-sign the CSR:

```
openssl x509 -sha256 -req -days 3650 -CA rootCA.pem -CAkey rootCA.key -
CAcreateserial -CAserial serial -in proxy.csr -out proxy.pem

cli>store certificate keystore_external_stap
```

3. Store the certificate of root CA in rootCA.pem from above example

```
cli>store certificate external_stap
```

4. Store the External S-TAP certificate in proxy.pem (from the example above) and use the same alias from the CSR that you created.

Step 4: Deploy an External S-TAP from Guardium

You can deploy an External S-TAP from the Guardium GUI. The Kubernetes Cluster manages the External S-TAP container and a load balancing service. The load balancer service is automatically assigned an external IP address, which becomes the new database endpoint.

To install the External S-TAP:

1. From AWS, get AWS RDS database info:

```
Database host:  <your database host>
Database port:  <your port>
Database option: TLS on or off TLS version (Amazon RDS->option groups)
```

2. Run External S-TAP AWS CLI Command Line

```
>kubectl apply -f aws_eks_exernal_stap_v10_6.yaml
```

Deployment yaml configure template:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myinstance01
  namespace: default
  labels:
    app: myinstance01
```

```
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myinstance01
  template:
    metadata:
      name: myinstance01
      labels:
        app: myinstance01
    spec:
      volumes:
      - name: dshm
        emptyDir:
          medium: Memory
          sizeLimit: 500Mi
      containers:
      - name: myinstance01
        image: store/ibmcorp/guardium_external_s-tap:v10.6.0.313
        securityContext:
          capabilities:
            add: ["NET_ADMIN", "SYS_TIME"]
        env:
            - name: STAP_CONFIG_TAP_TAP_IP
              value: ""
# Change the value to make it unique of the proxy uuid
            - name: STAP_CONFIG_PROXY_GROUP_UUID
              value: 24ef0fc8-3f9a-417f-afa6-32ecb73c32b9
              - name: STAP_CONFIG_PROXY_GROUP_UUID
              value: 24ef0fc8-3f9a-417f-afa6-32ecb73c32b9
            - name: STAP_CONFIG_PROXY_GROUP_MEMBER_COUNT
              value: "1"
# Change the value to match the database to be monitored
          - name: STAP_CONFIG_PROXY_DB_HOST
            value: "127.0.0.1"
          - name: STAP_CONFIG_PROXY_NUM_WORKERS
            value: "1"
          - name: STAP_CONFIG_PROXY_PROXY_PROTOCOL
            value: "0"
          - name: STAP_CONFIG_PROXY_DISCONNECT_ON_INVALID_CERTIFICATE
            value: "0"
          - name: STAP_CONFIG_PROXY_NOTIFY_ON_INVALID_CERTIFICATE
            value: "1"
          - name: STAP_CONFIG_DB_0_REAL_DB_PORT
            value: "2484"
          - name: STAP_CONFIG_PROXY_LISTEN_PORT
            value: "2484"
          - name: STAP_CONFIG_PROXY_DEBUG
            value: "4"
# Change this value to match the proxy secret returned during S-TAP certificate
request creation.
          - name: STAP_CONFIG_PROXY_SECRET
            value: 535e846c-9694-11e8-b124-f8278860ba21
          - name: STAP_CONFIG_DB_0_DB_TYPE
            value: mssql
# Change this value to match the ip address of the Guardium collector where the
certificate is registered.
          - name: STAP_CONFIG_SQLGUARD_0_SQLGUARD_IP
            value: 169.62.149.110
          - name: container
            value: docker
```

```
        ports:
          - containerPort: 8080
        volumeMounts:
          - mountPath: /dev/shm
            name: dshm
# This should be the name of your dockerhub login credentials secret.
      imagePullSecrets:
      - name: eksproxyacr
---
apiVersion: v1
kind: Service
metadata:
  name: myinstance01
  namespace: default
  labels:
    app: myinstance01
spec:
  type: LoadBalancer
  selector:
    app: myinstance01
  externalTrafficPolicy: Local
  ports:
   - protocol: TCP
     name: myinstance01
     port: 2484
     targetPort: 2484
```

Update the following params before running the deployment:

```
   STAP_CONFIG_PROXY_GROUP_UUID --use uuidgen  to get a new UUID

   STAP_CONFIG_PROXY_DB_HOST   -- hostname of database

   STAP_CONFIG_DB_0_REAL_DB_PORT –port of database

   STAP_CONFIG_DB_0_DB_TYPE.   – type of database, valid DB types are "oracle",
"mssql", "sybase", "mongodb", "db2", "mysql", "memsql", "mariadb", "pgsql",
"greenplumdb", "verticadb", "redis"

   STAP_CONFIG_SQLGUARD_0_SQLGUARD_IP –guardium collector ip

    imagePullSecrets     --kubernates cluster secret
```

Step 5: Prepare the Client with new database endpoint

You can find database endpoint from Kubernetes cluster services. Your client can use the new database endpoint to connect to the database for monitoring.

However, if the original database endpoint is required for database connection, then you need to perform additional DNS config steps, as follows.

1. Get database endpoint from the AWS CLI:

   ```
   > kubectl get services
   ```

   ```
   NAME         TYPE           CLUSTER-IP      EXTERNAL-IP
   kubernetes   ClusterIP      10.100.0.1      <none>
   s-tap        LoadBalancer   10.100.105.148  a8716a346d65d11e9a4fa0299e6bb7e6-250668305.us-east-1.elb.amazonaws.com
   ```

   The database endpoint will be <External-IP> and port will stay the same

2. If the database uses TLS for JDBC connection, using the following commands to create or update JKS key store and import rootCA.pem into the trusted keystore on the host (if the client requires certificate verification):

   ```
   >openssl x509 -outform der -in proxy.pem -out proxy.der
    >keytool -import -alias oracle12 -keystore cacerts -file proxy.der
   ```

   The `cacerts parameter` is the keystore for your java application. In general, `cacerts` is in the `<java_home>jre/lib/security/` directory. The default password for `cacerts` is *changeit*.

3. A new JDBC URL is generated. For example (using Oracle):

   ```
   jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCPS)(HOST=
   <external-ip>)(PORT=<dbport>))(CONNECT_DATA=(SID=ORCL)))
   ```

Additional information: Setting up the AWS EKS Cluster

Create the EKS cluster

Use *eksctl* to create the EKS cluster.

1. Create an access key for the AWS IAM user, and log in as IAM from the AWS command line with the `aws configure` command. For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html.
2 Add the following AWS EKS policies to the IAM user:
   - AmazonEKSClusterPolicy
   - AmzonEKSWorkerNodePolicy
   - AdministratorAccess
   - AmzonEKSServicePolicy

- AmzonEKS_CNI_Policy

3 From AWS CLI command line, use the `eksctl` automation tool to create the AWS EKS cluster. You can download the `eksctl` tool from the following site: https://eksctl.io/introduction/installation/.

4 For information about using eksctl, see https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html.

For example, to create an AWS EKS cluster with 3 nodes, use the following syntax:

```
>eksctl create cluster --name=mycluster --region=us-east-1 --nodes=3 --
node-type=c3.2xlarge
```

Tips:

- To start with single AWS RDS database, Guardium recommends using 2 XLarge VMs (8 Vcore CPU) as nodes for the Kubernetes Cluster Nodes in AWS EKS.
- If Metric Server is installed with the AWS EKS cluster, auto scale is enabled when you deploy External S-TAP from the GUI. For more information, see https://eksworkshop.com/scaling/deploy_hpa/.

  To get auto scale information for your deployment:

  ```
  kubectl get hpa
  ```

  - Commonly used commands:

    ```
    aws

    kubectl

    aws-iam-authenticator

    aws configure

    aws sts get-caller-identity

    aws eks --region us-east-1 update-kubeconfig --name mycluster

    kubectl cluster-info

    kubectl -n kube-system describe secret $(kubectl -n kube-system get
    secret | grep eks-admin | awk '{print $1}')

    kubectl get secret
    ```

```
kubectl proxy

http://localhost:8001/api/v1/namespaces/kube-
system/services/https:kubernetes-dashboard:/proxy/#!/login
```