Deploying External S-TAP on an Google GKE cluster

Before you begin

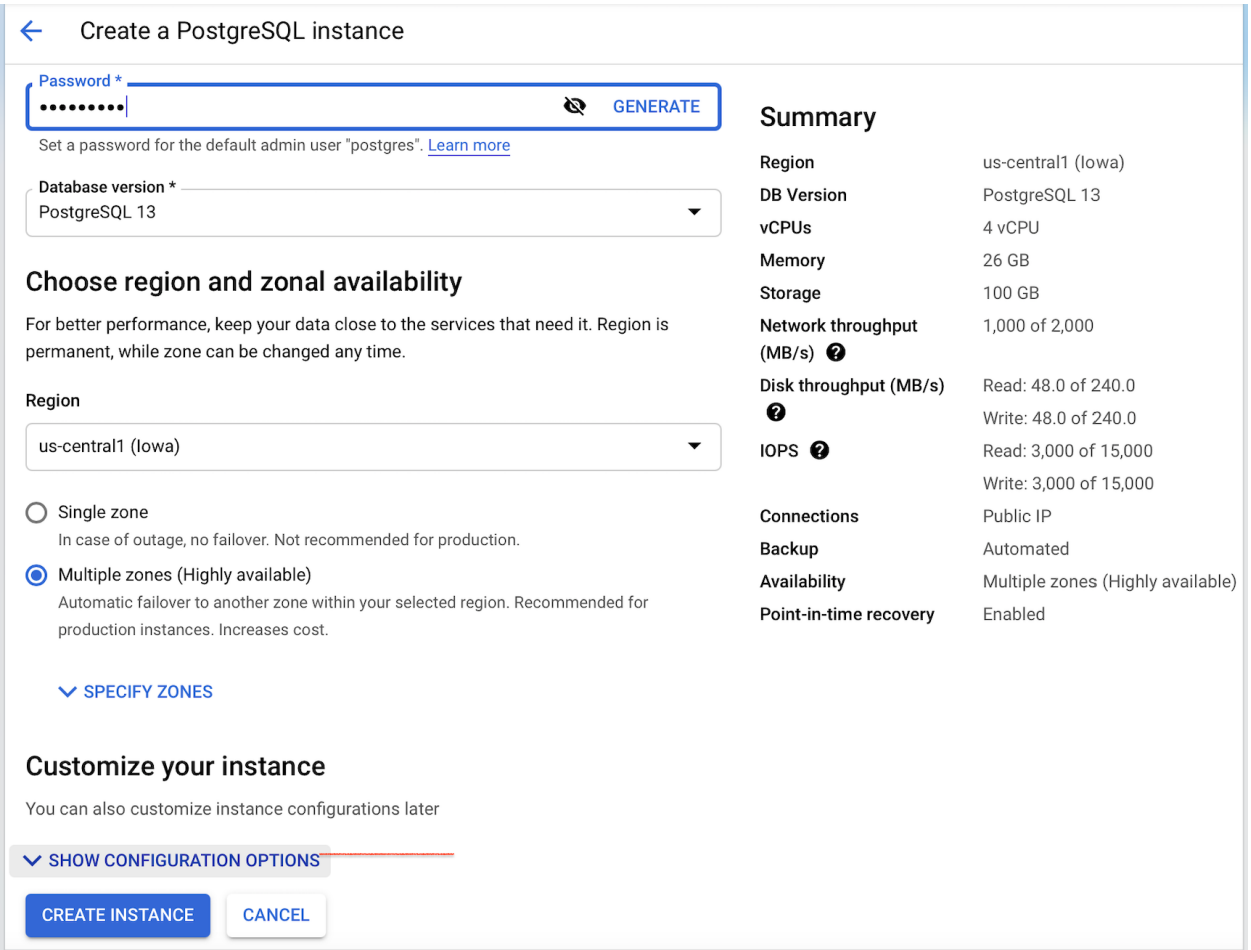
- Make sure that the following components are installed and set up before you start with Google GKE:

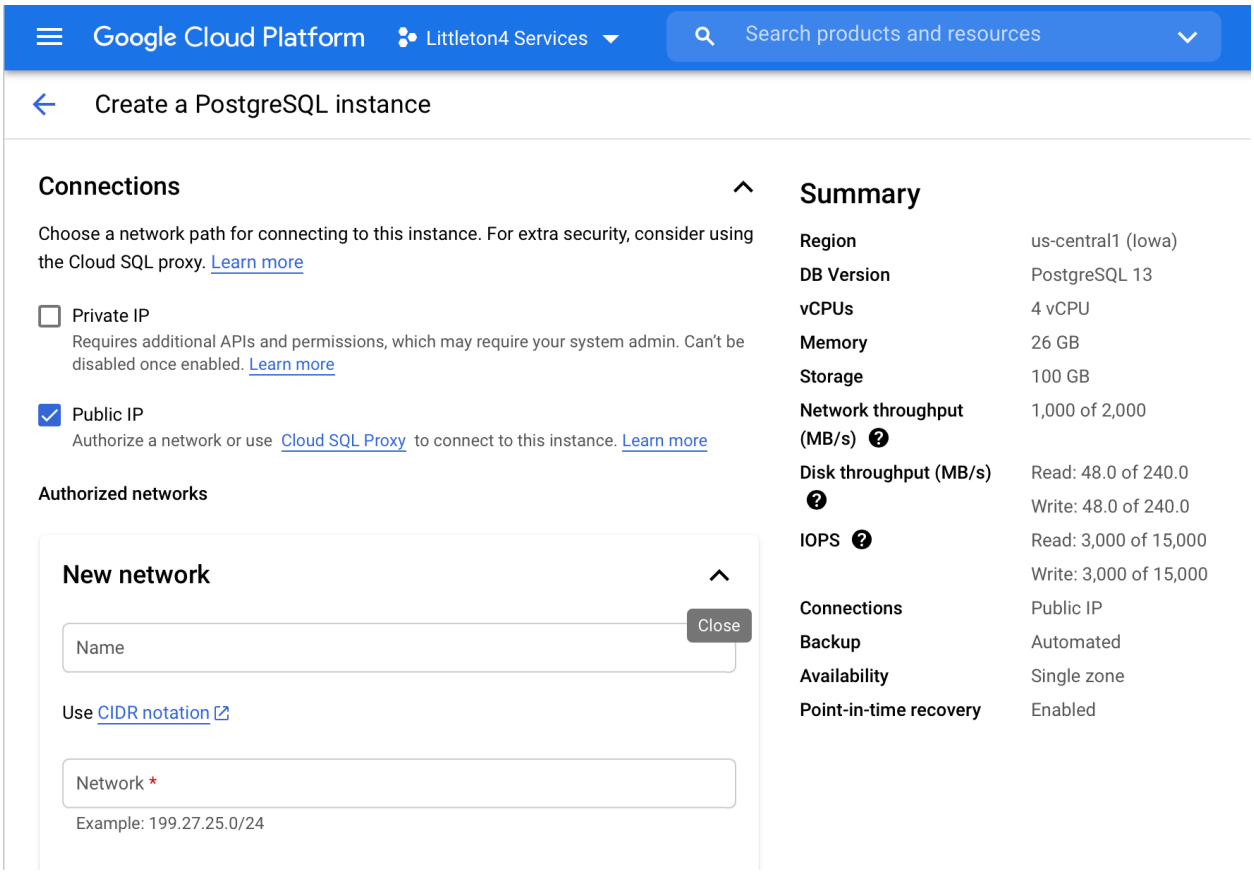
1. Create a docker hub user. For more information, see https://hub.docker.com/signup?next=%2F%3Fref%3Dlogin
2. Create a Guardium Collector in the cloud. For more information, see https://www-01.ibm.com/support/docview.wss?uid=swg27049576

Notes:
Use the following CLI commands to set up the ntp

```
cli> store system ntp server <cloud ntp server>
cli> store system ntp state on
```

Step 1: Create postgraSQL database instance in Google SQL databases





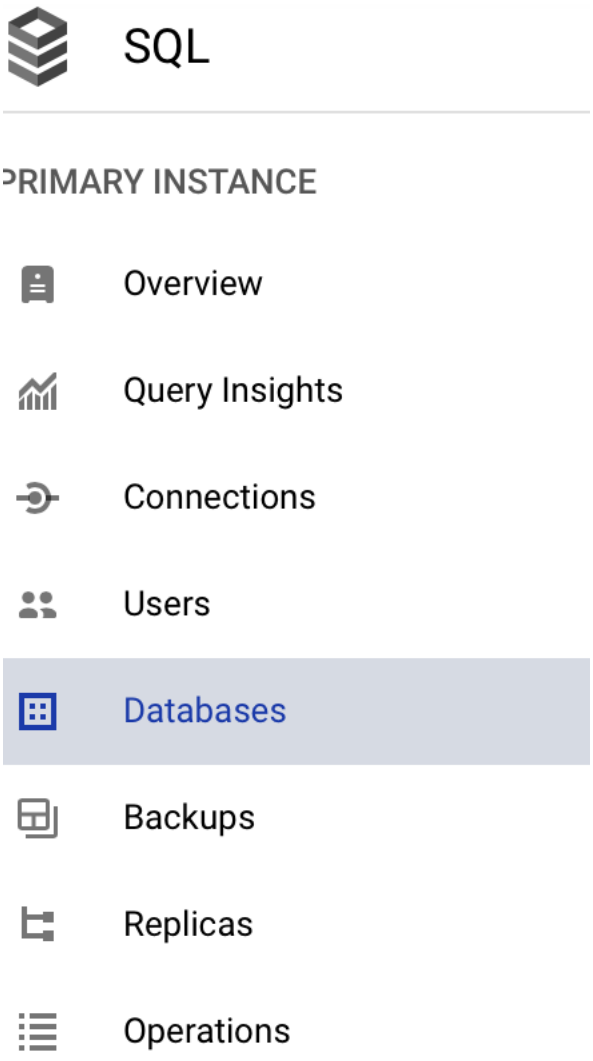
In the "Connections". Configure, check "public IP"

Add "Authorized network" for example address of your computer or network in CIDR notation. if your IPv4 address as single IPv4 address or CIDR notation to allocate addresses from a range, enter the entire range, such as 203.0.113.0/24.

To find out your IPv4 address , you can use the command:
curl ifconfig.me

Or go to brower and search for "what is my IP address"

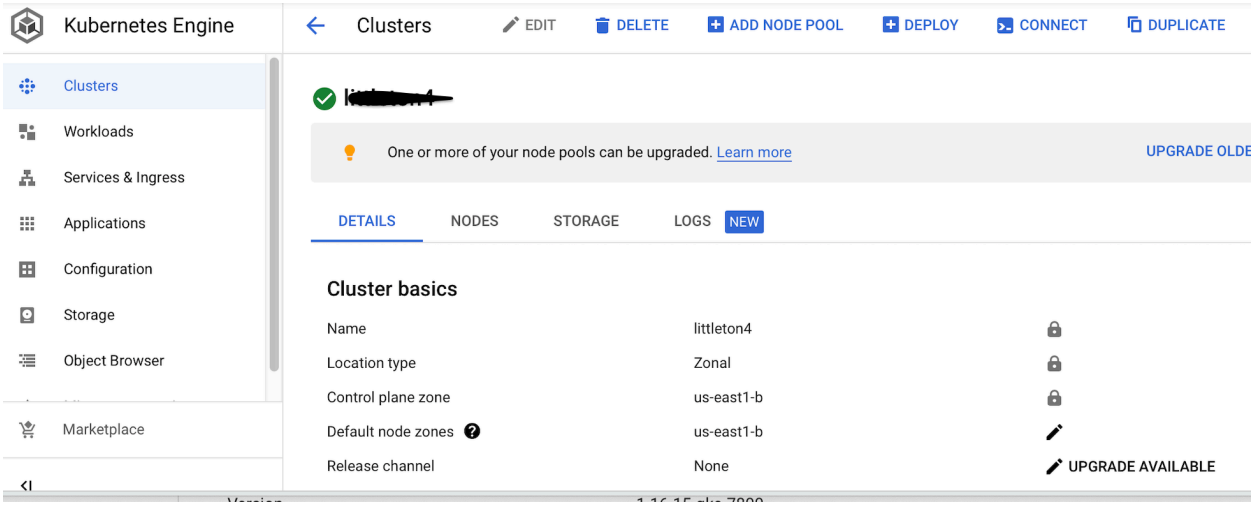
Create a database after the postgresql instance created :



From the instance overview , find the connection public IP for the database instance.

Step 2: Set up the Google GKE Kubernetes Cluster

Create a google GKE cluster by console→Google Cloud Platform→Kubernetes Engine, select cluster mode (Autopilot or Stanadard mode) , and use GUI to create GKE cluster .



1. Find the "CONNECT" button on GUI of GKE cluster and open a google cloud shell
   gcloud container clusters get-credentials your-credetial --zone your-zone --project your-project

2. Test google gke configuration with the `get svc` command:

   ```
   >kubectl get svc

   >kubectl. Get node
   ```

Step 3: Get GOOGLE GKS Kubernetes Cluster information

1. Get master URL of the GOOGLE GKE Cluster

   ```
   >kubectl cluster-info
   ```

2. Create a text file named as ibm-service-account.yaml with content as following:

   ```
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: eks-admin
     namespace: kube-system
   ---
   ```

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
 apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system
```

3. Get the admin access token of the GOOGLE GKE Cluster

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

4. 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 4: Prepare Certificate to be used for External S-TAP (Optional if connection is nonssl)

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.
SQL databases on GOOGLE CLOUD don’t require a common name (CN) match for certificate verification with SSL transport encryption.
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. Example as following:

```
demo>_create csr external_stap

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-----
MIIC5TCCAc0CAQAwWzELMAkGA1UEBhMCdXMxCzAJBgNVBAgTAm1hMQswCQYDVQQHEwJsdDERMA8G
A1UEChMIZ3VhcmRpdW0xDDAKBgNVBAsTA2libTERMA8GA1UEAxMIb3JhY2xlMTIwggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDrbIXWk4VWuviaSXfAuqE8/BNDMvEXaUF3bpebAH3xq8Bt
gC+n2ImvtReVc2YiZ0LXKSffAqT525IUt4mpUteJeou/v55Q2tbz5oAQGKTmBlc50eGLiq63HdxZ
6Tk8OV7KzpNNBsB1WsmWyJl0xS9q1b1UxvC9u23YZTZi2OZJnf5AlRxyJYxJTFZLQnznDGJTGNkS
0i/OizPUwBBoMLwDDMfKISwv1al/O+6zVBv3/OTCPBwVzGsW4B8ZkEKNGFSVAOekq++zZzKW3F98
WbPywTxUCgnaQYRFuiqTMUt7gwzFNEPEQpLseuQh4D+qSFIoxWjBFklMkXDwXjthfZWfAgMBAAGg
RTBDBgkqhkiG9w0BCQ4xNjA0MBMGA1UdEQQMMAqCCG9yYWNsZTEyMB0GA1UdDgQWBBRvr2t2R8kz
40N5u1WcGmadm5LKTzANBgkqhkiG9w0BAQsFAAOCAQEAAoSWJU4re0X4pgwihm3my58lmd5ERPBX
DtUzHMI813qPsHW4vm7KgB1VFk5uE10pkYVsR4hrH4IAvCOPpLq/uAOPyGIUNLdkVaKJp+geb/gL
cSXqQC/tEwj2cWalhBvbTQFVr22vq88D4ux2WeFqcta1f/fGOPsGsOIOgArUif5x3xRcaCrYRbnY
wIeGz0T+C3HhvLTCLAa0fJn+Fdn6TpoHnw3iB0XGI7fxE9CM/CZnsI8lHuMT8ezIEueIv3UzGxbT
V9SD3hY8vBFrM4B9Vgu7eKOdnf4exOFxh0H9OQRmkmF1kghz9ZwMoclCvQKvVmGL0vrTHgDBhBmg
aB/NaA==
-----END NEW CERTIFICATE REQUEST-----

**************************************************************************

Copy and paste the Certificate Signing Request (CSR), starting at the
'-----BEGIN NEW CERTIFICATE REQUEST-----' tag and ending at the '-----END
NEW CERTIFICATE REQUEST-----' tag, to a file. The CSR file will need to be
provided to a Certificate Authority (CA) of your choice in order to obtain
a valid certificate. Please note that the certificate will need to be in PEM
format so that it can be imported into the Guardium appliance. Once you receive
the certificate from your CA, use the following CLI command to start the import
process:

                << store certificate external_stap >>

At the prompt, enter the entire alias, as follows: oracle12 proxy_keycert e2ba2a79-07b1-11ea-a92d-f499d77181e8
To deploy the external_stap, use the following token: e2ba2a79-07b1-11ea-a92d-f499d77181e8
ok
demo>_
```

   Please pay attention to the new **alias** and **token** created from this step

   Which will be needed in step 5 and External S-TAP deployment

3. Copy and paste the Certificate Signing Request (CSR) to a file (proxy.csr), 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. For testing purposes only, use the following openssl command to create your own CA:

   a. Create root CA:

   ```
   openssl genrsa -out rootCA.key 2048

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

   b. 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
   ```

5. After you receive the certificate from your CA or create self-signed proxy.pem, use the following CLI command to start the import process:

   ```
   cli>store certificate keystore_external_stap
   ```

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

   ```
   cli>store certificate external_stap
   ```

   Store the External S-TAP certificate in proxy.pem (from the example above) or from CA service

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

   This **alias** should be the one from step 1:

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

Step 5: Deploy an External S-TAP from Guardium

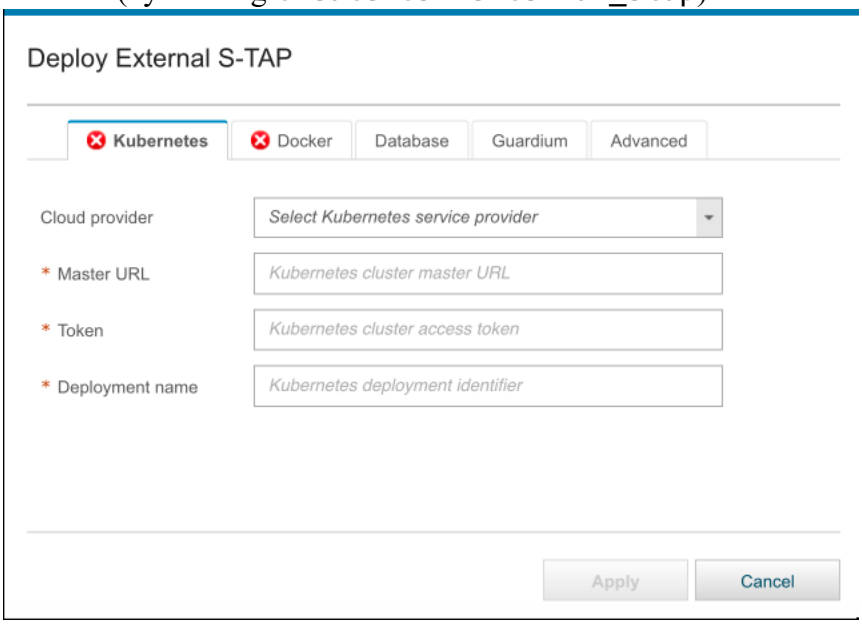
You can deploy an External S-TAP from the Guardium GUI. The Kubernetes Cluster manages the External S-TAP container and a load balancing service. The load balancer service is automatically assigned an external IP address, which becomes the new database endpoint.
To install the External S-TAP:

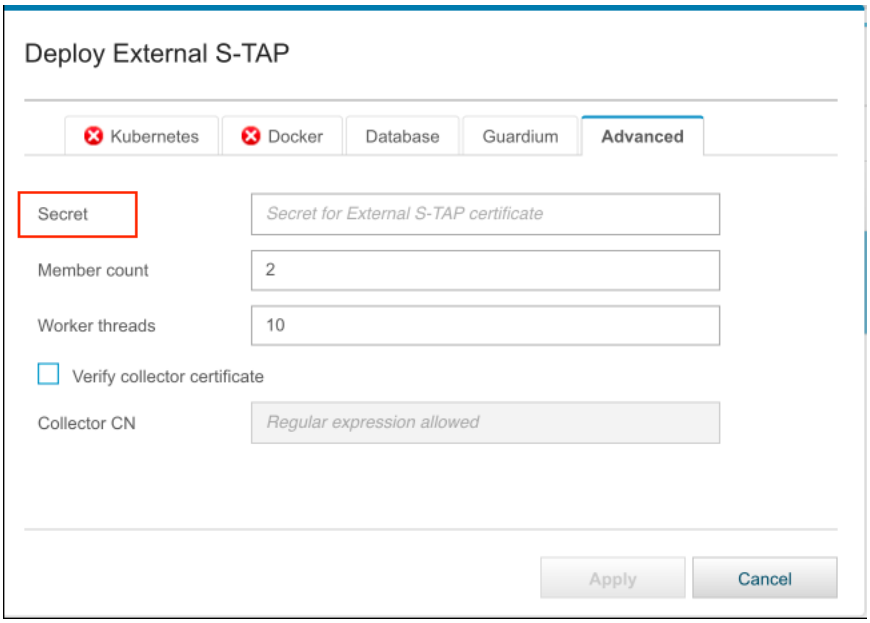
1. From GOOGLE Cloud , get SQL database info:

   ```
   Database host:  <The public IP of your postgreSQL instance>
   Database port:  5432
   ```

   From Guardium, open the External S-TAP UI as follows: Manage→Activity Monitoring->External S-TAP Control.

For information about using the External S-TAP GUI, see Deploy External S-TAP tab.
Notes: On the Advanced Tab, `Secret` is the token generated when you created the External S-TAP csr (by running `create csr external_stap`).





Step 6: 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 GOOGLE CLOUD SHELL:

   ```
   > kubectl get services
   ```

   ```
   ~ > kubectl get svc
   NAME                    TYPE           CLUSTER-IP    EXTERNAL-IP     PORT(S)
                                             AGE
   estap-api-service       LoadBalancer   10.0.7.178    [redacted]8     443:31368/TCP
   ```

   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) **(Optional)**

   ```
   openssl x509 -outform der -in rootCA.pem -out rootCA.der
   keytool -import -alias postgresql -keystore cacerts -file rootCA.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*.
   `Restart your java application after importing external stap certificate`

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

For non -encrypted connection:

jdbc:postgresql://<google gks lb>:5432/<your db name>

For -encrypted connection:

jdbc:postgresql://<google gks lb>:5432/<your db name> ?*sslmode*=require

Notes: Create GOOGLE GKS Kubernetes Cluster admin user required google IAM Account author permission, if you have only access permission, deploy estap using template instead.