accenture

myWizard® DevOps

SSL Enablement Guide_AWS

Version 1.3

May 2021

Table of Contents

Table of Figures

Overview

SSL is Secure Sockets Layer, that are integral part of web security. SSL enablement is required to protect the transfer of sensitive information over the internet. This document elaborates the steps to be followed to enable SSL for the myWizard® DevOps Platform.

SSL Enablement for Platform on AWS

To enable the SSL for the myWizard® DevOps Platform running on AWS, do the following steps.

Creating a Record Set

Creating a record set is required to attach the ELB where you want to route the traffic to. To create a record set, a domain must be registered. A domain is the name which along with record sets can be used to display or access any hosted application or website. To register the domain, do the following.

Note: *The following steps 1 to 7 are to be followed if you have not registered a domain only.*

1. Login to the AWS console.
2. Click **Services**.
   The different services are listed.
3. Click **Route 53** in Network & Content Delivery.
   The **Amazon Route 53** page appears.
4. Click **Registered Domain** under Domains in the left pane.
   The **Registered Domain** page appears.
5. Click **Register Domain** option.
   The **Domain Search** page appears.

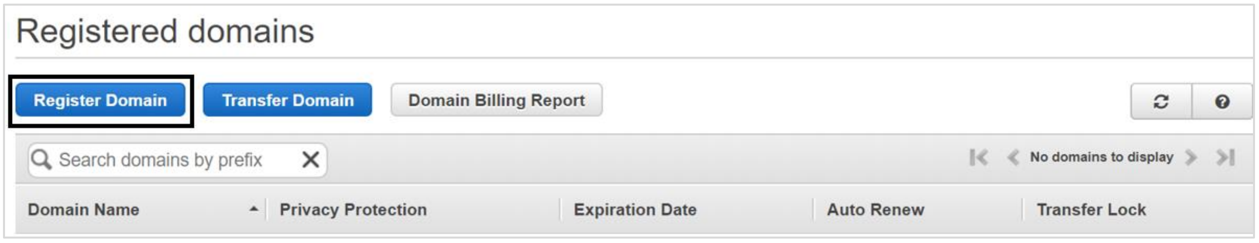
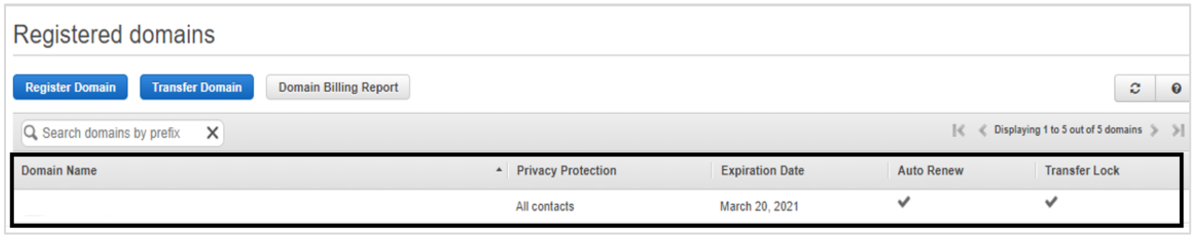


Figure 1: Domain Registration

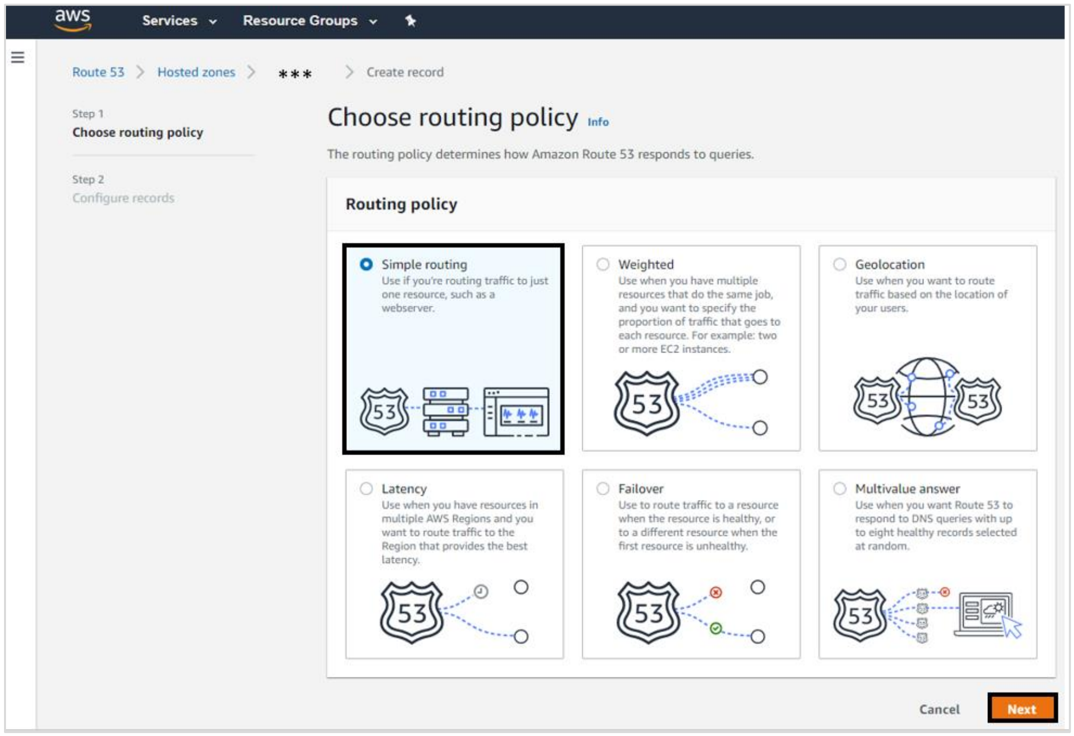
6. Search the Domain Name, provide the **Contact Details**, and **Verify and Purchase**.
7. You see the domain name in the console.


Registered domains

Domain Name	Privacy Protection	Expiration Date	Auto Renew	Transfer Lock
	All contacts	March 20, 2021	✓	✓

Figure 2: Registered Domain Page

8. Click **Hosted Zones** in the left pane and choose the hosted zone where you want to create the records in. For example, select the newly created domain if you want to create a record in it.
9. Click **Create Record**.
   The **Choose routing policy** page appears.



Choose routing policy

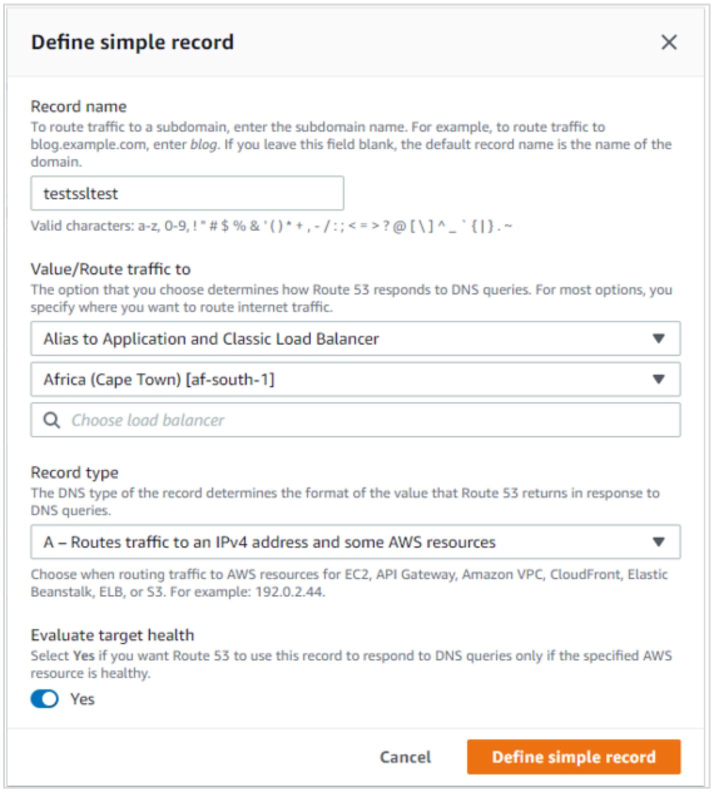
The routing policy determines how Amazon Route 53 responds to queries.

Routing policy

- Simple routing: Use if you're routing traffic to just one resource, such as a webserver.
- Weighted: Use when you have multiple resources that do the same job, and you want to specify the proportion of traffic that goes to each resource. For example: two or more EC2 instances.
- Geolocation: Use when you want to route traffic based on the location of your users.
- Latency: Use when you have resources in multiple AWS Regions and you want to route traffic to the Region that provides the best latency.
- Failover: Use to route traffic to a resource when the resource is healthy, or to a different resource when the first resource is unhealthy.
- Multivalue answer: Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.

Figure 3: Choose Routing Policy

10. Click **Simple routing** and **Next**.
    The **Define simple record** page appears.


Define simple record

Record name
To route traffic to a subdomain, enter the subdomain name. For example, to route traffic to blog.example.com, enter *blog*. If you leave this field blank, the default record name is the name of the domain.

testssltest

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Value/Route traffic to
The option that you choose determines how Route 53 responds to DNS queries. For most options, you specify where you want to route internet traffic.

Alias to Application and Classic Load Balancer

Africa (Cape Town) [af-south-1]

Choose load balancer

Record type
The DNS type of the record determines the format of the value that Route 53 returns in response to DNS queries.

A – Routes traffic to an IPv4 address and some AWS resources

Choose when routing traffic to AWS resources for EC2, API Gateway, Amazon VPC, CloudFront, Elastic Beanstalk, ELB, or S3. For example: 192.0.2.44.

Evaluate target health
Select **Yes** if you want Route 53 to use this record to respond to DNS queries only if the specified AWS resource is healthy.

Yes

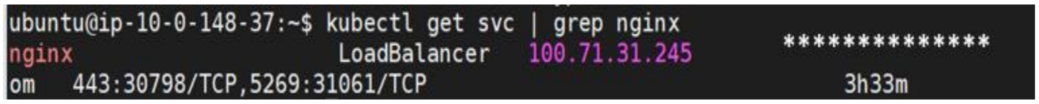
Figure 4: Define Simple Record Page

Provide the following details.
Record name – Provide the name of the subdomain where you want to route the traffic.
Value/Route traffic to – Provide the value where you want to route traffic to. For example, Alias to Application and Classic Load Balancer, the Region, and the ELB in that region.
Record type – Provide the DNS Record Type value based on AWS resource where you are routing the traffic. For example, select A – Ipv4 address.

Generating SSL Certs

To generate SSL Certs for nginx, do the following.

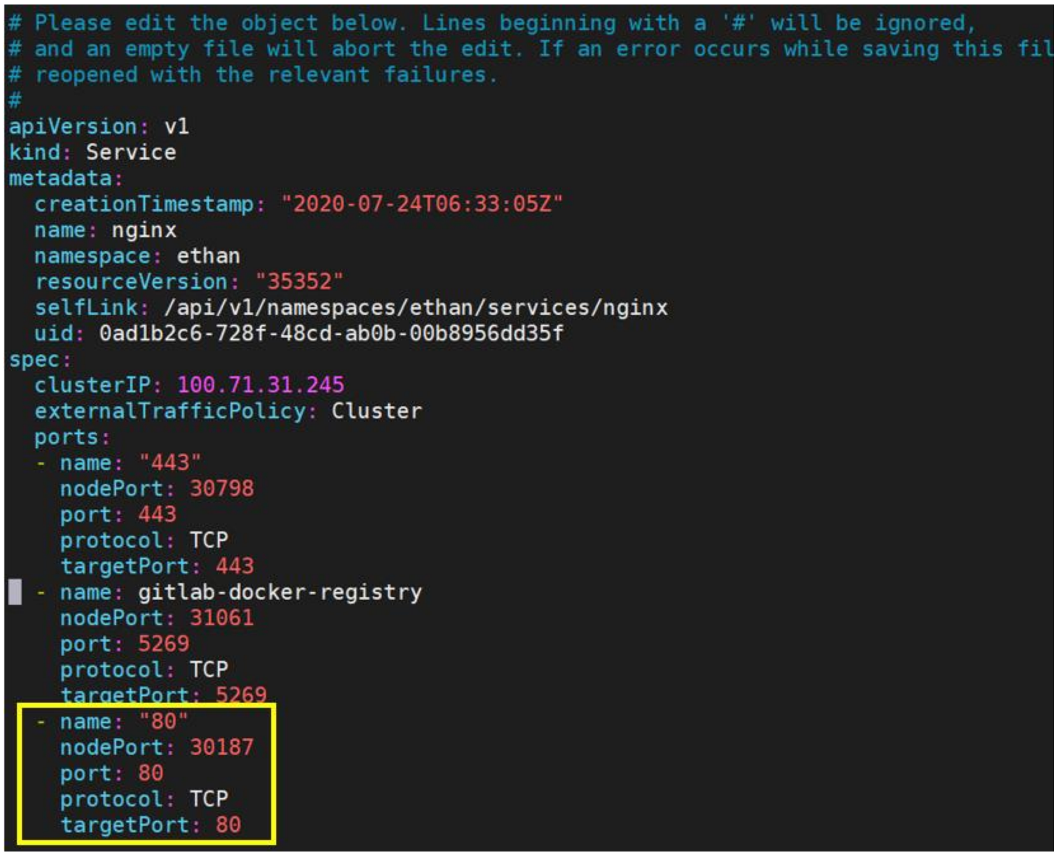
1. Provide the following command to check if the port 80 is opened.
   kubectl get svc | grep nginx



```
ubuntu@ip-10-0-148-37:~$ kubectl get svc | grep nginx
nginx                    LoadBalancer   100.71.31.245        **************
om   443:30798/TCP,5269:31061/TCP                                 3h33m
```

Figure 5: List Service Filtering Nginx

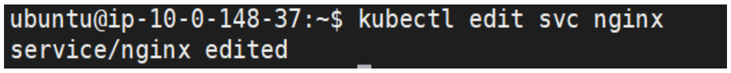
2. If the port 80 is not opened, then do the following to open it. This is required for generation of SSL certificate.
   Run **kubectl edit svc nginx** and then add the below block.
   Note: *To edit the file click **i**. Click esc to exit the edit mode and **:wq!** to save and exit the svc file. Click **:q!** to exit the without saving the file.*
   - name: "80"
   nodePort: 30187
   port: 80
   protocol: TCP
   targetPort: 80



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this fil
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: "2020-07-24T06:33:05Z"
  name: nginx
  namespace: ethan
  resourceVersion: "35352"
  selfLink: /api/v1/namespaces/ethan/services/nginx
  uid: 0ad1b2c6-728f-48cd-ab0b-00b8956dd35f
spec:
  clusterIP: 100.71.31.245
  externalTrafficPolicy: Cluster
  ports:
  - name: "443"
    nodePort: 30798
    port: 443
    protocol: TCP
    targetPort: 443
  - name: gitlab-docker-registry
    nodePort: 31061
    port: 5269
    protocol: TCP
    targetPort: 5269
  - name: "80"
    nodePort: 30187
    port: 80
    protocol: TCP
    targetPort: 80
```

Figure 6: Add Port 80 Block

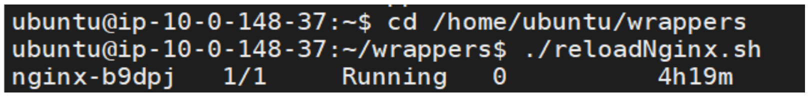
3. Once you edit the file, save and exit.



```
ubuntu@ip-10-0-148-37:~$ kubectl edit svc nginx
service/nginx edited
```

Figure 7: Edit Service File of Nginx

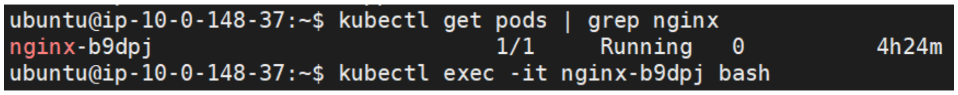
4. Execute **cd /home/ubuntu/wrappers** and run **./reloadNginx.sh** to reload the nginx after making the changes.



```
ubuntu@ip-10-0-148-37:~$ cd /home/ubuntu/wrappers
ubuntu@ip-10-0-148-37:~/wrappers$ ./reloadNginx.sh
nginx-b9dpj   1/1     Running   0          4h19m
```

Figure 8: Command to Navigate into Wrappers and Reload Nginx

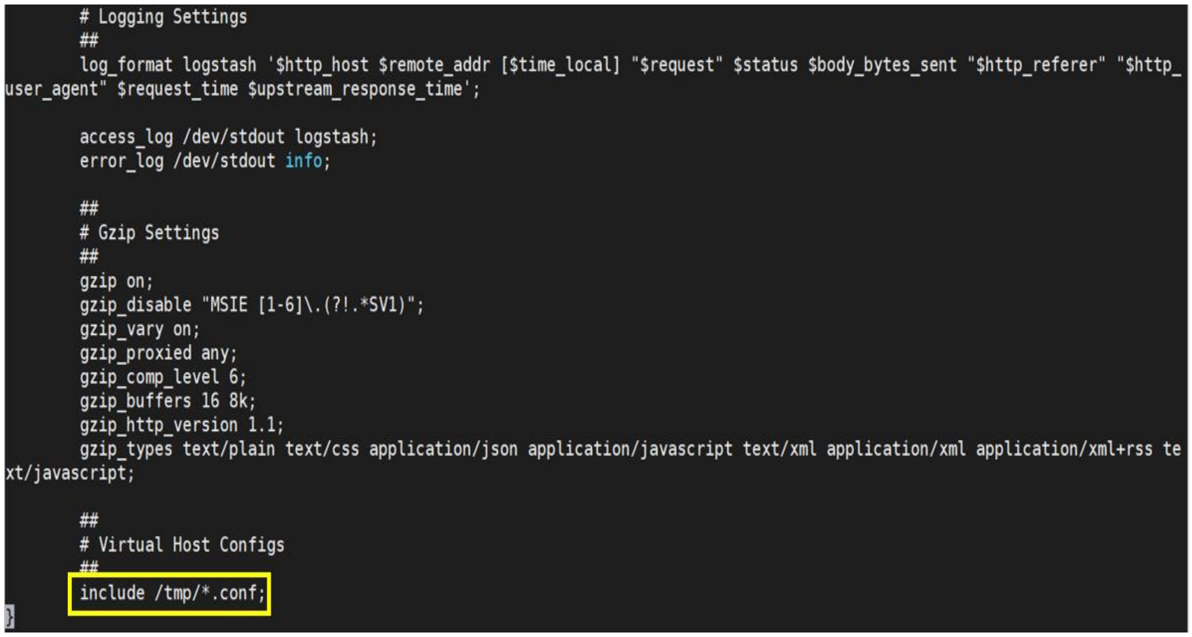
5. Exec into the nginx pod and the route to generate the certificate.
   kubectl get pods | grep nginx
   kubectl exec -it <nginxpodname> bash



```
ubuntu@ip-10-0-148-37:~$ kubectl get pods | grep nginx
nginx-b9dpj                          1/1     Running   0          4h24m
ubuntu@ip-10-0-148-37:~$ kubectl exec -it nginx-b9dpj bash
```

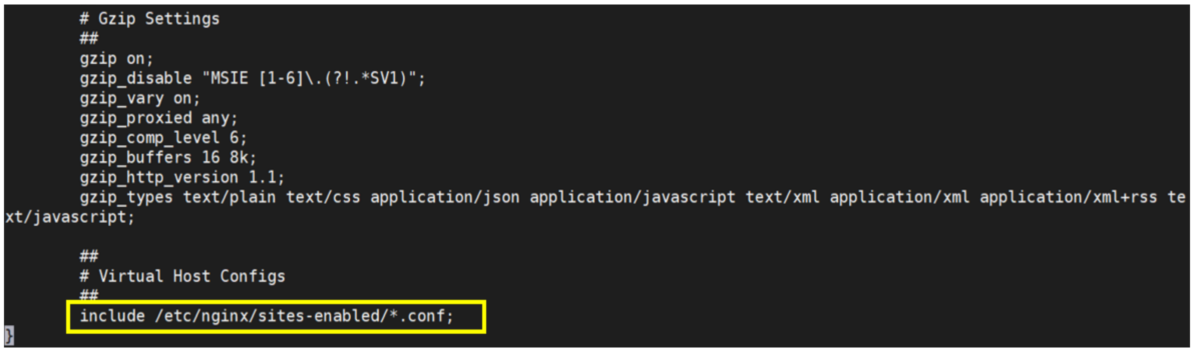
Figure 9: Command to get Nginx Pod Details

6. To install vim editor, enter the following.
   apt-get install vim -y
7. Enter **vim /etc/nginx/nginx.conf** to edit the global nginx configuration file.
8. Check if the last line of the configuration file is pointing to the **include /tmp/*.conf;**. If yes, change it to the following.
   /etc/nginx/sites-enabled/*.conf;



```
        # Logging Settings
        ##
        log_format logstash '$http_host $remote_addr [$time_local] "$request" $status $body_bytes_sent "$http_referer" "$http_
user_agent" $request_time $upstream_response_time';

        access_log /dev/stdout logstash;
        error_log /dev/stdout info;

        ##
        # Gzip Settings
        ##
        gzip on;
        gzip_disable "MSIE [1-6]\.(?!.*SV1)";
        gzip_vary on;
        gzip_proxied any;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss te
xt/javascript;

        ##
        # Virtual Host Configs
        ##
        include /tmp/*.conf;
}
```

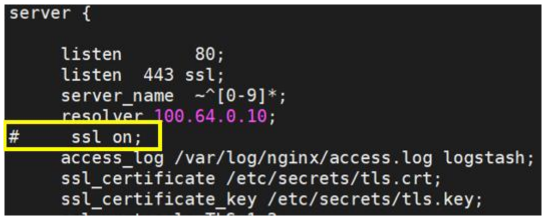
Figure 10: Unmodified Nginx Configuration File



```
        # Gzip Settings
        ##
        gzip on;
        gzip_disable "MSIE [1-6]\.(?!.*SV1)";
        gzip_vary on;
        gzip_proxied any;
        gzip_comp_level 6;
        gzip_buffers 16 8k;
        gzip_http_version 1.1;
        gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss te
xt/javascript;

        ##
        # Virtual Host Configs
        ##
        include /etc/nginx/sites-enabled/*.conf;
}
```

Figure 11: Modified Nginx Configuration File

9. Save the file and exit.
10. Do **vim /etc/nginx/sites-enabled/tools-context.conf** and comment out ssl on directive with #. This is required to be turned off since SSL certificate generation is done in Port 80.

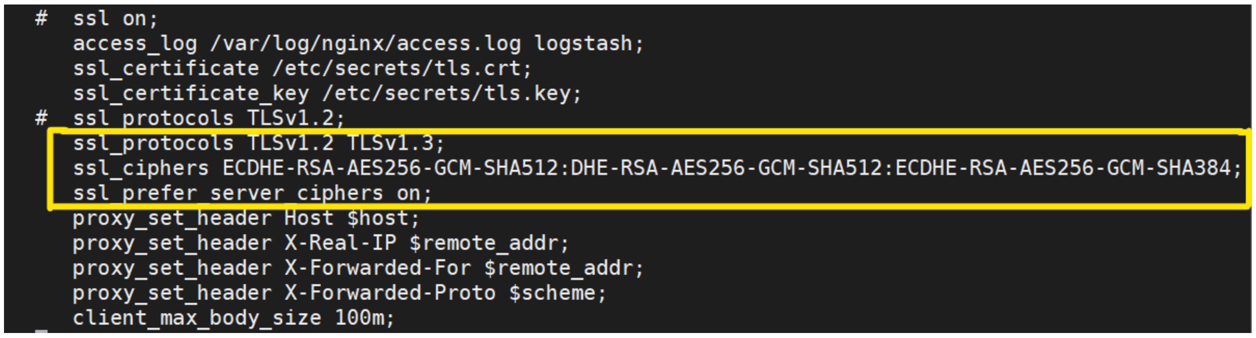


```
server {

     listen       80;
     listen  443 ssl;
     server_name  ~^[0-9]*;
     resolver 100.64.0.10;
#     ssl on;
     access_log /var/log/nginx/access.log logstash;
     ssl_certificate /etc/secrets/tls.crt;
     ssl_certificate_key /etc/secrets/tls.key;
```

Figure 12: Comment SSL ON

11. In the same file, add the below lines of code to add TLS protocols and ciphers as shown in Figure 13. User must comment out line **ssl_protocols TLSv1.2**

ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384;
ssl_prefer_server_ciphers on;

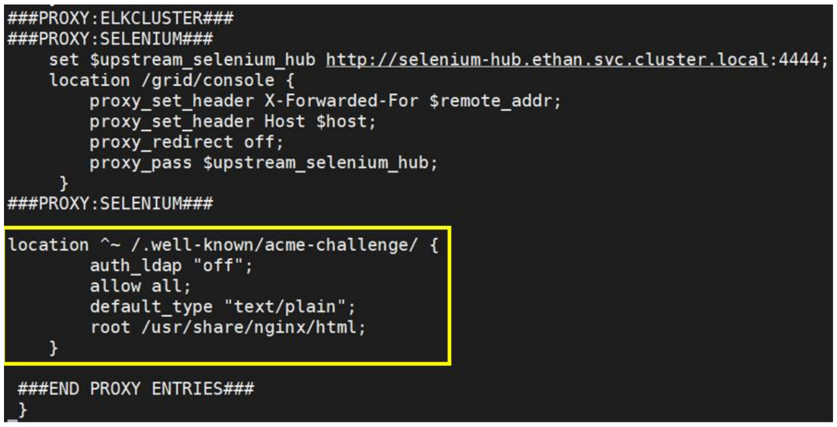


```
  #  ssl on;
     access_log /var/log/nginx/access.log logstash;
     ssl_certificate /etc/secrets/tls.crt;
     ssl_certificate_key /etc/secrets/tls.key;
  #  ssl_protocols TLSv1.2;
     ssl_protocols TLSv1.2 TLSv1.3;
     ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384;
     ssl_prefer_server_ciphers on;
     proxy_set_header Host $host;
     proxy_set_header X-Real-IP $remote_addr;
     proxy_set_header X-Forwarded-For $remote_addr;
     proxy_set_header X-Forwarded-Proto $scheme;
     client_max_body_size 100m;
```

Figure 13: Adding TLS Protocols

12. In the same file go to the end of the file and add the following block. This is required to accept the response from the SSL server.

```
location ^~ /.well-known/acme-challenge/ {
        auth_ldap "off";
        allow all;
        default_type "text/plain";
        root /usr/share/nginx/html;
    }
```

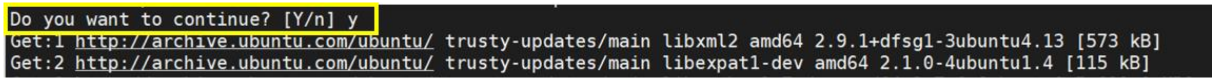


```
###PROXY:ELKCLUSTER###
###PROXY:SELENIUM###
    set $upstream_selenium_hub http://selenium-hub.ethan.svc.cluster.local:4444;
    location /grid/console {
        proxy_set_header X-Forwarded-For $remote_addr;
        proxy_set_header Host $host;
        proxy_redirect off;
        proxy_pass $upstream_selenium_hub;
     }
###PROXY:SELENIUM###

location ^~ /.well-known/acme-challenge/ {
        auth_ldap "off";
        allow all;
        default_type "text/plain";
        root /usr/share/nginx/html;
    }

 ###END PROXY ENTRIES###
 }
```

Figure 14: Add Location Block

13. Save the file and do **nginx -s reload** to reload the updated configuration files.
14. If there are any errors shown, rectify the configuration file changes and reload the updated configuration files.
15. Install certbot with the following command.
    wget https://raw.githubusercontent.com/certbot/certbot/v1.9.0/letsencrypt-auto-source/letsencrypt-auto -O certbot-auto
16. Provide execute permissions to file using **chmod +x certbot-auto** command.
17. To get the certificates, execute the following command.
    ./certbot-auto certonly -a webroot --webroot-path=/usr/share/nginx/html --no-self-upgrade -d <dnsrecordset.domainname.com>

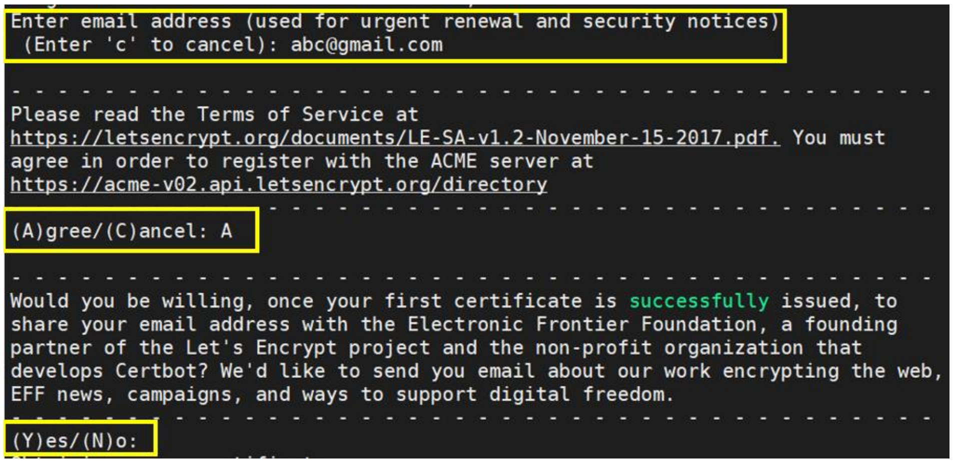
    For example, if myweb.mydomain is the domain name, then provide the command **./certbot-auto certonly -a webroot --webroot-path=/usr/share/nginx/html --no-self-upgrade -d myweb.mydomain.com**
18. Running the above command asks for confirmation to install the python packages that are required to generate the certificates, enter **Y** to confirm the installation.



```
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libxml2 amd64 2.9.1+dfsg1-3ubuntu4.13 [573 kB]
Get:2 http://archive.ubuntu.com/ubuntu/ trusty-updates/main libexpat1-dev amd64 2.1.0-4ubuntu1.4 [115 kB]
```

Figure 15: Confirmation for Installation of Python Packages

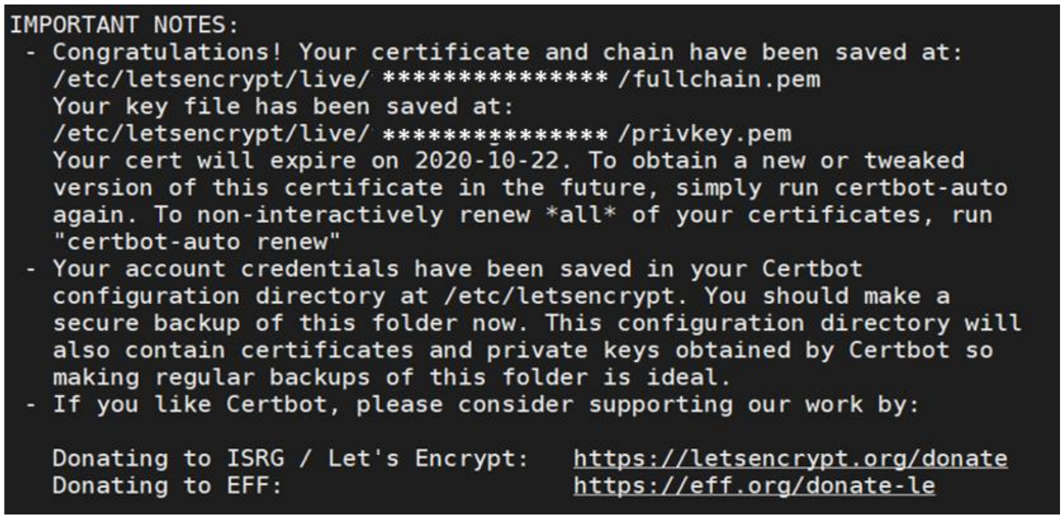
19. Enter the **email id** and other details requested accordingly.



```
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): abc@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server at
https://acme-v02.api.letsencrypt.org/directory
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(A)gree/(C)ancel: A

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o:
```

Figure 16: Provide Details for Certificate Creation

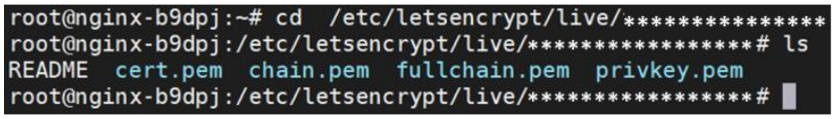
20. Once the certificates are generated successfully, the following message is displayed.



```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/ *************** /fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/ *************** /privkey.pem
   Your cert will expire on 2020-10-22. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot-auto
   again. To non-interactively renew *all* of your certificates, run
   "certbot-auto renew"
 - Your account credentials have been saved in your Certbot
   configuration directory at /etc/letsencrypt. You should make a
   secure backup of this folder now. This configuration directory will
   also contain certificates and private keys obtained by Certbot so
   making regular backups of this folder is ideal.
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

Figure 17: Confirmation of Certificate Generation

21. Go to the path **cd /etc/letsencrypt/live/<dnsrecordset.domainname.com>/** and enter **ls** to get the pem keys.



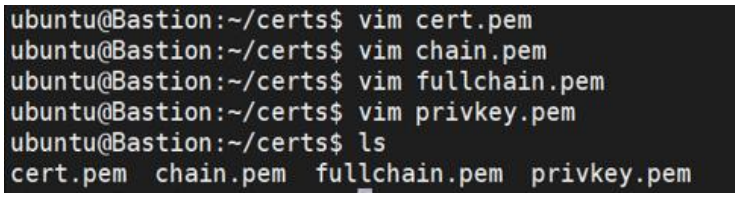
```
root@nginx-b9dpj:~# cd  /etc/letsencrypt/live/****************
root@nginx-b9dpj:/etc/letsencrypt/live/*****************# ls
README  cert.pem  chain.pem  fullchain.pem  privkey.pem
root@nginx-b9dpj:/etc/letsencrypt/live/*****************#
```

Figure 18: List SSL Certificate Files

22. The certificates are generated in the nginx pod.
    Create a directory from the bastion to store a copy of the certificates generated in the nginx pod. (You can duplicate the tab and execute in the bastion.)

Execute the following commands in the bastion to create a directory and create certificate files.

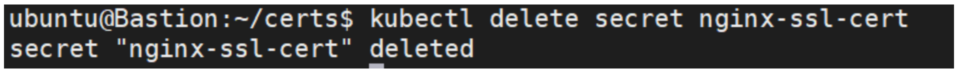
- **mkdir certs** to create the directory and execute **cd certs** to enter into the cert directory.
- **vim cert.pem** and copy the contents of cert.pem file generated in the nginx pod.
- **vim fullchain.pem** and copy the contents of fullchain.pem file generated in the nginx pod.
- **vim chain.pem** and copy the contents of chain.pem file generated in the nginx pod.
- **vim privkey.pem** and copy the contents of privkey.pem file generated in the nginx pod.



```
ubuntu@Bastion:~/certs$ vim cert.pem
ubuntu@Bastion:~/certs$ vim chain.pem
ubuntu@Bastion:~/certs$ vim fullchain.pem
ubuntu@Bastion:~/certs$ vim privkey.pem
ubuntu@Bastion:~/certs$ ls
cert.pem  chain.pem  fullchain.pem  privkey.pem
```

Figure 19: Certificate Files in Cert Directory

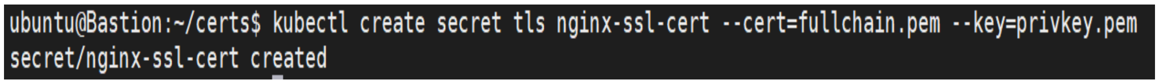
23. Delete the nginx-ssl-cert secret that has the dummy ssl certificate that comes along with the provisioning by using command **kubectl delete secret nginx-ssl-cert**



```
ubuntu@Bastion:~/certs$ kubectl delete secret nginx-ssl-cert
secret "nginx-ssl-cert" deleted
```

Figure 20: Delete Nginx SSL Certificate

24. Create the nginx-ssl-cert secret using the files generated for SSL using the command **kubectl create secret tls <secret name> --cert=fullchain.pem --key=privkey.pem**

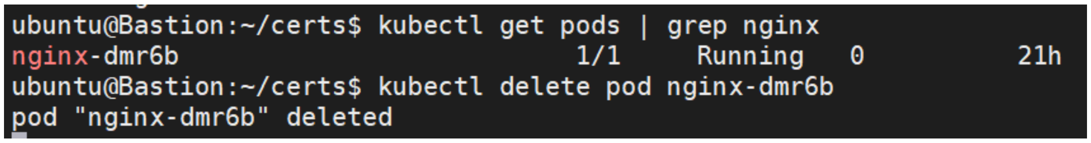


```
ubuntu@Bastion:~/certs$ kubectl create secret tls nginx-ssl-cert --cert=fullchain.pem --key=privkey.pem
secret/nginx-ssl-cert created
```

Figure 21: Generate New Secret

Note: *This command must be executed from the same path where the .pem files are copied. For example, secret name is nginx-ssl-cert.*

25. Delete the nginx pod to pick the SSL from the new secret generated using the command **kubectl delete pod <nginx pod name>**



```
ubuntu@Bastion:~/certs$ kubectl get pods | grep nginx
nginx-dmr6b                              1/1     Running   0          21h
ubuntu@Bastion:~/certs$ kubectl delete pod nginx-dmr6b
pod "nginx-dmr6b" deleted
```

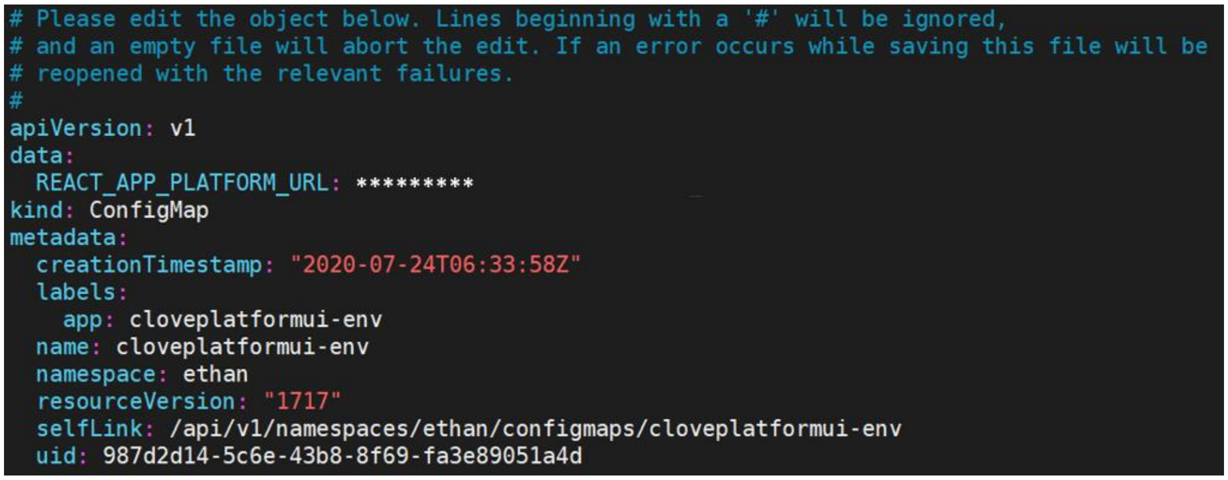
Figure 22: Delete Nginx Pod

26. Exit the cert folder using command **cd**

Updating the DNS

The following steps must be followed to update the DNS in the tools list and MongoDB from the bastion.

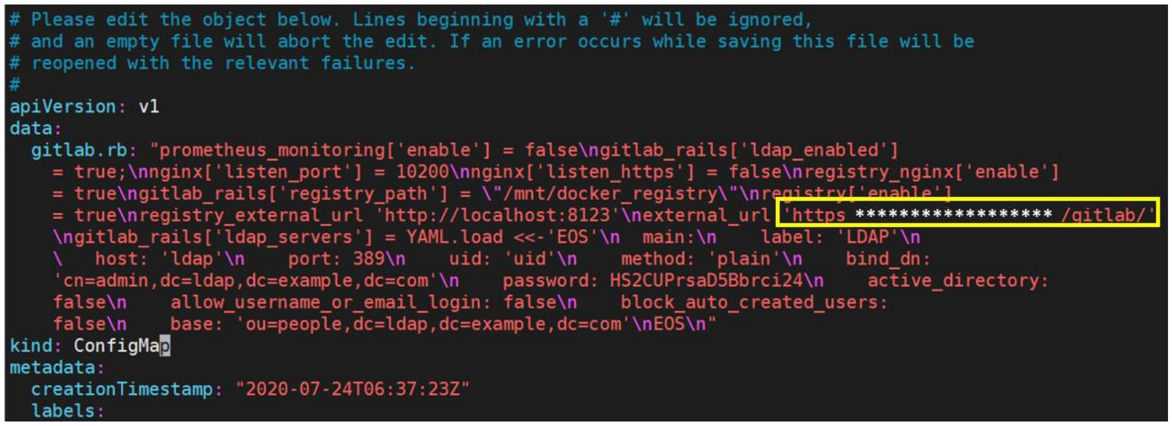
1. Execute **kubectl edit cm cloveplatformui-env** to edit the configuration map of cloveplatformui-env
   Replace the ELB with DNS name in REACT_APP_PLATFORM_URL and save the file. For example, **https://<dnsrecordset.domainname.com>**



```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  REACT_APP_PLATFORM_URL: *********
kind: ConfigMap
metadata:
  creationTimestamp: "2020-07-24T06:33:58Z"
  labels:
    app: cloveplatformui-env
  name: cloveplatformui-env
  namespace: ethan
  resourceVersion: "1717"
  selfLink: /api/v1/namespaces/ethan/configmaps/cloveplatformui-env
  uid: 987d2d14-5c6e-43b8-8f69-fa3e89051a4d
```

Figure 23: Edit Platform UI Configmap

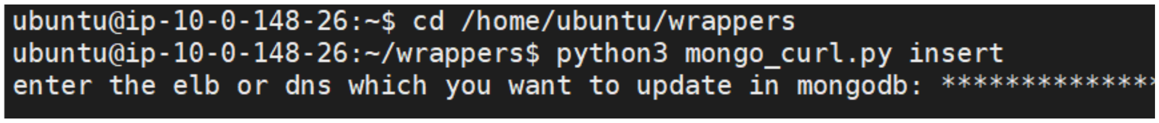
2. Delete the pod using **kubectl delete pod -l app=cloveplatformuiparent** command to bring up a new parent pod with the new configuration.
3. Do **kubectl edit cm gitlab-env** to edit the configuration map of gitlab-env (This step is not applicable for base platform, that is, platform provisioned without cartridge).
   Replace the ELB with DNS in external URL variable. For example, **'https://<dnsrecordset.domainname.com>/gitlab/'**



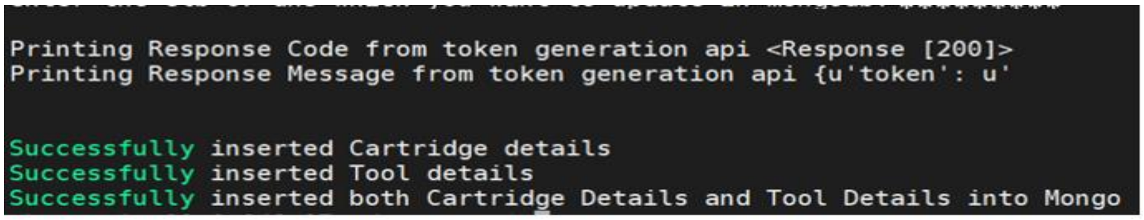
```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  gitlab.rb: "prometheus_monitoring['enable'] = false\ngitlab_rails['ldap_enabled']
    = true;\nnginx['listen_port'] = 10200\nnginx['listen_https'] = false\nregistry_nginx['enable']
    = true\ngitlab_rails['registry_path'] = \"/mnt/docker_registry\"\nregistry['enable']
    = true\nregistry_external_url 'http://localhost:8123'\nexternal_url 'https ****************** /gitlab/'
    \ngitlab_rails['ldap_servers'] = YAML.load <<-'EOS'\n  main:\n    label: 'LDAP'\n
    \   host: 'ldap'\n    port: 389\n    uid: 'uid'\n    method: 'plain'\n    bind_dn:
    'cn=admin,dc=ldap,dc=example,dc=com'\n    password: HS2CUPrsaD5Bbrci24\n    active_directory:
    false\n    allow_username_or_email_login: false\n    block_auto_created_users:
    false\n    base: 'ou=people,dc=ldap,dc=example,dc=com'\nEOS\n"
kind: ConfigMap
metadata:
  creationTimestamp: "2020-07-24T06:37:23Z"
  labels:
```

Figure 24: Edit GitLab Configmap

4. Delete the pod using **kubectl delete pod -l app=gitlab** to bring up a new pod with new configurations (This step is not applicable for base platform, that is, platform provisioned without cartridge).
5. Execute **pip3 install -r requirements.txt** to install the packages.
6. Go to wrappers folder with command **cd /home/ubuntu/wrappers** and run **python3 mongo_curl.py insert**
7. Input the DNS name. For example, **<dnsrecordset.domainname.com**>



```
ubuntu@ip-10-0-148-26:~$ cd /home/ubuntu/wrappers
ubuntu@ip-10-0-148-26:~/wrappers$ python3 mongo_curl.py insert
enter the elb or dns which you want to update in mongodb: *************
```



```
Printing Response Code from token generation api <Response [200]>
Printing Response Message from token generation api {u'token': u'


Successfully inserted Cartridge details
Successfully inserted Tool details
Successfully inserted both Cartridge Details and Tool Details into Mongo
```

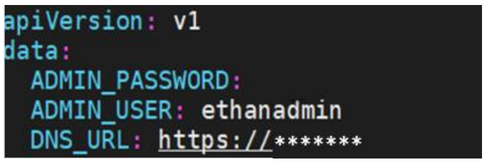
Figure 25: Update DNS in MongoDB

Steps for Multi-Cloud Platform

Include the following steps only if you have provisioned a **Multi-Cloud Platform**.

Configuration Changes for MC Platform

1. Execute **kubectl edit cm multicloudapi-env** to edit the configuration map of multicloudapi-env
   Replace the ELB with DNS name in DNS_URL and save the file. For example, https://<dnsrecordset.domainname.com>



```
apiVersion: v1
data:
  ADMIN_PASSWORD:
  ADMIN_USER: ethanadmin
  DNS_URL: https://*******
```

Figure 26: Edit Configuration Map

2. Delete the multi-cloud child application pod using **kubectl delete pod -l app=mc-app** command to bring up a new pod will with the new configuration.
3. Delete the multi-cloud server application pod using **kubectl delete pod -l app=multicloudapi** command to bring up a new pod will with the new configuration.

External Access to GitLab Docker Registry

For some use cases secure access to the GitLab docker registry from outside Kubernetes is required without opening custom ports. The following approach hosts the docker registry at the root of the mywizard domain.

1. Open a shell on the bastion host.
2. Grab a copy of current config map for nginx.

   ```
   kubectl get cm nginx-conf -o jsonpath='{.data.tools-context\.conf}' > tools-context.conf
   ```

3. Edit the local copy using command **vim tools-context.conf** to add the following section before ###END PROXY ENTRIES###.

   ```
   ###PROXY:GITLAB-REGISTRY###
      set $upstream_gitlab_registry http://gitlab.ethan.svc.cluster.local:8223;
      location /v2 {
          client_max_body_size 512m;
          auth_ldap off;
          proxy_pass $upstream_gitlab_registry;
       }
   ###PROXY:GITLAB-REGISTRY###
   ```

4. Under the GitLab proxy block, add **'auth_ldap off;'** as highlighted below.

   ```
   ###PROXY:GITLAB###
      set $upstream_gitlab http://gitlab.ethan.svc.cluster.local:8084;
      location /gitlab {
         client_max_body_size 512m;
              auth_ldap off;
         proxy_pass $upstream_gitlab;
      }
   ###PROXY:GITLAB###
   ```

5. Recreate the nginx config map.

   ```
   kubectl delete cm nginx-conf
   kubectl create cm nginx-conf --from-file tools-context.conf
   ```

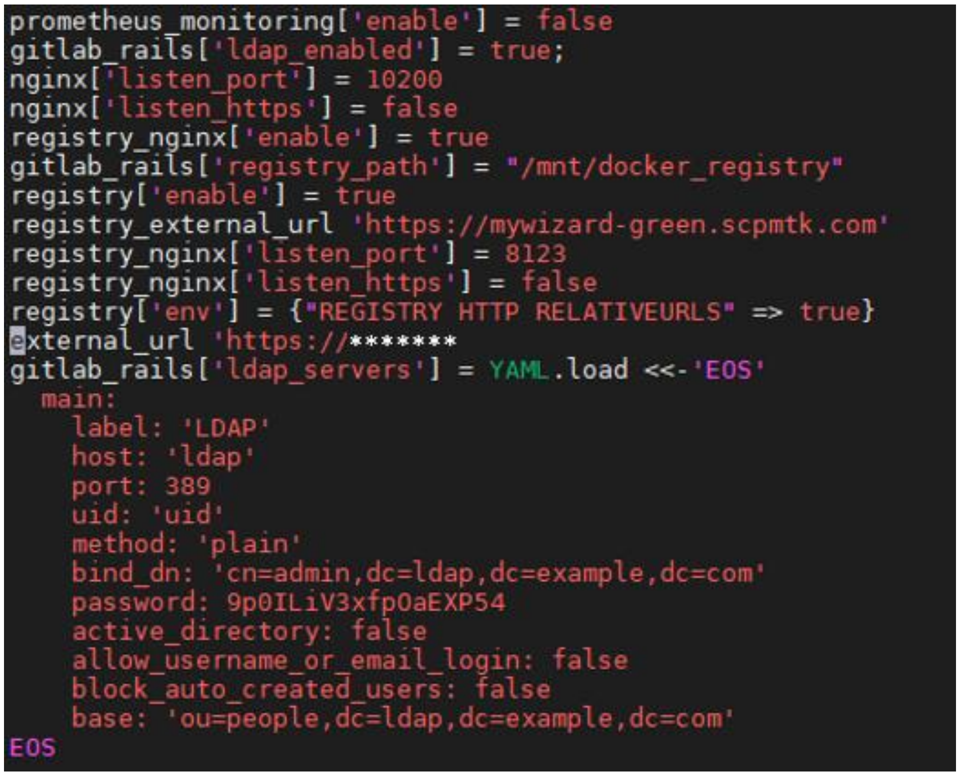
6. Delete nginx pods to trigger them to recreate with the new config map.

   ```
   kubectl delete pod -l app=nginx
   ```

7. Grab a copy of the config map for gitlab.

   ```
   kubectl get cm gitlab-env -o jsonpath='{.data.gitlab\.rb}' > gitlab.rb
   ```

8. Edit the local copy using command **vim gitlab.rb** to add/change the following lines as shown in the figure below.



```
prometheus_monitoring['enable'] = false
gitlab_rails['ldap_enabled'] = true;
nginx['listen_port'] = 10200
nginx['listen_https'] = false
registry_nginx['enable'] = true
gitlab_rails['registry_path'] = "/mnt/docker_registry"
registry['enable'] = true
registry_external_url 'https://mywizard-green.scpmtk.com'
registry_nginx['listen_port'] = 8123
registry_nginx['listen_https'] = false
registry['env'] = {"REGISTRY HTTP RELATIVEURLS" => true}
external_url 'https://*******
gitlab_rails['ldap_servers'] = YAML.load <<-'EOS'
  main:
    label: 'LDAP'
    host: 'ldap'
    port: 389
    uid: 'uid'
    method: 'plain'
    bind_dn: 'cn=admin,dc=ldap,dc=example,dc=com'
    password: 9p0ILiV3xfpOaEXP54
    active_directory: false
    allow_username_or_email_login: false
    block_auto_created_users: false
    base: 'ou=people,dc=ldap,dc=example,dc=com'
EOS
```

Figure 27: Modifying Gitlab.rb

registry_external_url 'https://mywizard-green.scpmtk.com'
registry_nginx['listen_port'] = 8123
registry_nginx['listen_https'] = false
registry['env'] = {"REGISTRY_HTTP_RELATIVEURLS" => true}

9. Recreate the gitlab config map.

   kubectl delete cm gitlab-env
   kubectl create cm gitlab-env --from-file gitlab.rb

10. Scale down/up gitlab to restart with new config map (due to single mount PVC the pod cannot be deleted).

   kubectl scale rs gitlab-rs --replicas=0
   kubectl scale rs gitlab-rs --replicas=1

11. If docker is not installed, run the following commands.

   sudo apt-get update
   sudo apt-get remove docker docker-engine docker.io
   sudo apt install docker.io
   sudo systemctl start docker
   sudo systemctl enable docker
   docker --version

12. Check if you can access the registry using docker login as mentioned below.

   sudo docker login <dnsrecordset.domainname.com>