## 5. Creating New Applications

**oc new-app (IMAGE | IMAGESTREAM | TEMPLATE | PATH | URL ...) [options]**

List all local templates and image streams that can be used to create an app
**# oc new-app --list**

Create an application based on the source code in the current git repository (with a public remote)and a Docker image
```
# oc new-app . --docker-image=repo/langimage
```

Create a Ruby application based on the provided [image]~[source code] combination
```
# oc new-app centos/ruby-22-centos7~https://github.com/openshift/ruby-ex.git
```

Use the public Docker Hub MySQL image to create an app. Generated artifacts will be labeled with db=mysql
```
# oc new-app mysql MYSQL_USER=user MYSQL_PASSWORD=pass MYSQL_DATABASE=testdb -l db=mysql
```

Use a MySQL image in a private registry to create an app and override application artifacts' names
```
# oc new-app --docker-image=myregistry.com/mycompany/mysql --name=private
```

Create an application from a remote repository using its beta4 branch
```
# oc new-app https://github.com/openshift/ruby-hello-world#beta4
```

Create an application based on a stored template, explicitly setting a parameter value
```
# oc new-app --template=ruby-helloworld-sample --param=MYSQL_USER=admin
```

Create an application from a remote repository and specify a context directory
```
# oc new-app https://github.com/youruser/yourgitrepo --context-dir=src/build
```

Create an application based on a template file, explicitly setting a parameter value
```
# oc new-app --file=./example/myapp/template.json --param=MYSQL_USER=admin
```

Search all templates, image streams, and Docker images for the ones that match "ruby"
```
# oc new-app --search ruby
```

Search for "ruby", but only in stored templates (--template, --image-stream and --docker-image can be used to filter search results)
```
# oc new-app --search --template=ruby
```

Search for "ruby" in stored templates and print the output as an YAML
```
# oc new-app --search --template=ruby --output=yaml
```

### 5.1. Creating application from Dockerfile in Git Repository Server

**$ oc new-app --name=echo --strategy=docker --code http://registry.lab.example.com/rhel7-echo --insecure-registry --dry-run**
```
--> Found Docker image 93bb76d (17 months old) from registry.lab.example.com:5000 for
"registry.lab.example.com:5000/rhel7:7.3"
…
```

**$ oc new-app --name=echo --strategy=docker --code http://registry.lab.example.com/rhel7-echo --insecure-registry**
```
--> Found Docker image 93bb76d (17 months old) from registry.lab.example.com:5000 for
"registry.lab.example.com:5000/rhel7:7.3"
…
```

#### 5.1.1. Clone Dockerfile from Git Server & perform updating
```
$ mkdir git-dir
$ cd git-dir/

$ git clone http://registry.lab.example.com/rhel7-echo
Cloning into 'rhel7-echo'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.

$ cd rhel7-echo/
$ ls -alR
-rw-rw-r--. 1 student student  99 Nov 29 05:54 Dockerfile
drwxrwxr-x. 8 student student 163 Nov 29 05:55 .git

$ more Dockerfile
FROM registry.lab.example.com:5000/rhel7:7.3
CMD bash -c "while true; do echo test; sleep 5; done"

$ vi Dockerfile
FROM registry.lab.example.com:5000/rhel7:7.3
CMD bash -c "while true; do (( i++ )); echo test \$i; sleep 5; done"
```

```
$ git commit -a -m 'fixed ke 5'
[master e7e349f] fixed ke 5
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git push
Counting objects: 14, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.03 KiB | 0 bytes/s, done.
Total 12 (delta 3), reused 0 (delta 0)
To http://registry.lab.example.com/rhel7-echo
   b524fbd..e7e349f  master -> master

$ oc start-build echo
build "echo-5" started

$ oc status|oc status -v|oc logs -f bc/app|oc logs -f db/app|oc get events
$ oc get all|oc describe all
```

**Mencoba beberapa option git**

```
$ git checkout
$ git log
commit 3bc302a6805004b86d9ea0dfc86c18de2b0d9671
Author: Student User <student@example.com>
Date:   Thu Jan 4 21:35:55 2018 +0000
    Initial commit
$ git show
commit 3bc302a6805004b86d9ea0dfc86c18de2b0d9671
Author: Student User <student@example.com>
Date:   Thu Jan 4 21:35:55 2018 +0000

    Initial commit

diff --git a/Dockerfile b/Dockerfile
new file mode 100644
index 0000000..ace1778
--- /dev/null
+++ b/Dockerfile
@@ -0,0 +1,2 @@
+FROM registry.lab.example.com:5000/rhel7:7.3
+CMD bash -c "while true; do echo test; sleep 5; done"

$ git status
# On branch master
nothing to commit, working directory clean
$ git tag
```

#### 5.2. Creating Application from Source Code in Git Server/Image Stream, Scaling & Rebuilding

```
$ oc whoami
developer

$ oc new-project manage-deploy
$ oc new-app --name scale http://services.lab.example.com/php-scale
--> Found image c101534 (15 months old) in image stream "openshift/php" under tag "7.0" for "php"
…
$ oc scale --replicas=2 dc/scale
deploymentconfig "scale" scaled
```

##### 5.2.1. Update the PHP source code & redeploy the application

```
$ pwd
/home/student/git-dir

$ git clone http://services.lab.example.com/php-scale
Cloning into 'php-scale'...
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.

$ cd php-scale/
$ ls
index.php

$ cat index.php
<?php
print "This is version 1 of the app. I am running on host...
?>

$ vi index.php
<?php
print "This is version 2 of the app. I am running on host...
?>
```

```
$ git commit -a -m "update to versio 2.0"
[master 1458474] update to versio 2.0
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git push
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 278 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To http://services.lab.example.com/php-scale
   2431c76..1458474  master -> master

$ oc start-build scale
build "scale-2" started
```

#### 5.3. Redeploying Application from New Docker Image in form of .tar file

```
$ oc whoami
developer

$ oc new-project trigger-builds

$ oc new-app --name=trigger http://services.lab.example.com/trigger-builds
--> Found image c101534 (15 months old) in image stream "openshift/php"
…
$ oc describe bc/trigger |grep -i trigger
Name:          trigger
Namespace:     trigger-builds
Labels:        app=trigger
URL:           http://services.lab.example.com/trigger-builds
Output to:     ImageStreamTag trigger:latest
Triggered by:          Config, ImageChange
        URL:    https://master.lab.example.com:443/oapi/v1/namespaces/trigger-
builds/buildconfigs/trigger/webhooks/oUKrfp9DYR3ARK6-Db3w/github
        URL:           https://master.lab.example.com:443/oapi/v1/namespaces/trigger-
builds/buildconfigs/trigger/webhooks/R2E6CD5A2GbfYAo54aa9/generic
…

$ pwd
/home/student/DO288/labs/trigger-builds

$ ls
oc-new-app.sh  php-70-rhel7-newer.tar.gz  push-image.sh

$ more push-image.sh
#!/bin/bash

cd ~/DO288/labs/trigger-builds
docker load -i php-70-rhel7-newer.tar.gz
docker tag \
    registry.lab.example.com:5000/rhscl/php-70-rhel7:7.0-5.14 \
    registry.lab.example.com:5000/rhscl/php-70-rhel7:latest
docker push \
    registry.lab.example.com:5000/rhscl/php-70-rhel7:latest

$ docker load -i php-70-rhel7-newer.tar.gz
d4d408077555: Loading layer [==================================================>] 205.9 MB/205.9 MB
5444fe2e6b50: Loading layer [==================================================>] 10.24 kB/10.24 kB
45f0d85c3257: Loading layer [==================================================>] 17.22 MB/17.22 MB
aa29c7023a3c: Loading layer [==================================================>] 261.1 MB/261.1 MB
411dbf4b1b4e: Loading layer [==================================================>] 107.2 MB/107.2 MB
453f9841c215: Loading layer [==================================================>]  2.56 kB/2.56 kB
Loaded image: registry.lab.example.com:5000/rhscl/php-70-rhel7:7.0-5.14

$ docker images
REPOSITORY                                      TAG                 IMAGE ID            CREATED
SIZE
registry.lab.example.com:5000/rhscl/php-70-rhel7   7.0-5.14            42167603f124        12 months ago
564.8 MB

$ docker tag \
registry.lab.example.com:5000/rhscl/php-70-rhel7:7.0-5.14 \
registry.lab.example.com:5000/rhscl/php-70-rhel7:latest

$ docker images
REPOSITORY                                      TAG                 IMAGE ID            CREATED
SIZE
registry.lab.example.com:5000/rhscl/php-70-rhel7   7.0-5.14            42167603f124        12 months ago
564.8 MB
registry.lab.example.com:5000/rhscl/php-70-rhel7   latest              42167603f124        12 months ago
564.8 MB
```

```
$ docker push registry.lab.example.com:5000/rhscl/php-70-rhel7:latest
The push refers to a repository [registry.lab.example.com:5000/rhscl/php-70-rhel7]
453f9841c215: Pushed
411dbf4b1b4e: Pushed
aa29c7023a3c: Pushed
45f0d85c3257: Pushed
5444fe2e6b50: Pushed
d4d408077555: Pushed
latest: digest: sha256:74f7ceff5941433d7b05224f5c1ef0cfd0eab3655b07a569647cd33e15526d75 size: 1579

$ docker images
REPOSITORY                                          TAG                 IMAGE ID            CREATED
SIZE
registry.lab.example.com:5000/rhscl/php-70-rhel7    7.0-5.14            42167603f124        12 months ago
564.8 MB
registry.lab.example.com:5000/rhscl/php-70-rhel7    latest              42167603f124        12 months ago
564.8 MB
```

#### 5.4. Creating Application from Docker Images & Creating Template Files

```
$ oc whoami
developer

$ oc new-project strategy

$ oc new-app --name mysql --docker-image registry.lab.example.com:5000/rhscl/mysql-57-rhel7 --insecure-
registry MYSQL_USER=test MYSQL_PASSWORD=redhat MYSQL_DATABASE=testdb MYSQL_ROOT_PASSWORD=redhat
--> Found Docker image 4ae3a3f (15 months old) from registry.lab.example.com:5000 for
"registry.lab.example.com:5000/rhscl/mysql-57-rhel7
…
--> Creating resources ...
    imagestream "mysql" created
    deploymentconfig "mysql" created
    service "mysql" created
--> Success
    Run 'oc status' to view your app.

$ oc get all

NAME       DOCKER REPO                                        TAGS      UPDATED
is/mysql   docker-registry.default.svc:5000/strategy/mysql    latest    13 seconds ago

NAME       REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/mysql   1          1         1         config,image(mysql:latest)

NAME         DESIRED   CURRENT   READY     AGE
rc/mysql-1   1         1         1         13s

NAME        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
svc/mysql   172.30.103.180   <none>        3306/TCP   13s

NAME               READY     STATUS    RESTARTS   AGE
po/mysql-1-d7kpf   1/1       Running   0          8s
```

Creating template file :

```
$ oc export is,dc,rc,svc,route --as-template mysql > mysql-from-docker-image-template.yaml

$ oc delete all -l app=mysql
imagestream "mysql" deleted
deploymentconfig "mysql" deleted
service "mysql" deleted
pod "mysql-1-d7kpf" deleted
```

#### 5.5. Creating a Multi-Container Template

```
$ lab create-template setup

$ oc whoami
Developer

$ oc project
Using project "quotes-dev" on server "https://master.lab.example.com:443".

oc get all
NAME           TYPE      FROM      LATEST
bc/quotesapi   Source    Git       1

NAME                 TYPE      FROM          STATUS     STARTED             DURATION
builds/quotesapi-1   Source    Git@8294041   Complete   About an hour ago   1m20s

NAME           DOCKER REPO                                          TAGS      UPDATED
is/quotesapi   docker-registry.default.svc:5000/quotes-dev/quotesapi   latest    About an hour ago
```

```
NAME           REVISION   DESIRED   CURRENT   TRIGGERED BY
dc/quotesapi   1          1         1         config,image(quotesapi:latest)
dc/quotesdb    2          1         1         config,image(mysql:5.7)

NAME             DESIRED   CURRENT   READY     AGE
rc/quotesapi-1   1         1         1         1h
rc/quotesdb-1    0         0         0         1h
rc/quotesdb-2    1         1         1         1h

NAME               HOST/PORT                      PATH      SERVICES    PORT       TERMINATION   WILDCARD
routes/quotesapi   quotes.apps.lab.example.com             quotesapi   8080-tcp                 None

NAME            CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
svc/quotesapi   172.30.233.245   <none>        8080/TCP   1h
svc/quotesdb    172.30.68.43     <none>        3306/TCP   1h

NAME                    READY     STATUS      RESTARTS   AGE
po/quotesapi-1-build    0/1       Completed   0          1h
po/quotesapi-1-rnm4h    1/1       Running     0          1h
po/quotesdb-2-gng7g     1/1       Running     0          1h
```

**$ oc export is,bc,dc,svc,route,pvc --as-template test > test-template.yaml**

**Create template in openshift project by cluster admin user**

```
$ oc login -u admin -p redhat
$ oc create -f quotes-template-clean.yaml -n openshift
$ oc get template -n openshift|grep -i quotes
quotes The Quotes application provides an HTTP API that returns a random, funny quote.    3 (1 blank) 8
```

**Create the application using template from web console.**

Clean up

```
$ oc delete template/quotes -n openshift
```

**5.6. Creating Application from YAML Resources Configuration Files**.

```
$ oc get is/apache-httpd -o yaml > hello-is.yaml
$ oc get dc/hello -o yaml > hello-dc.yaml
$ oc get svc/hello -o yaml > hello-svc.yaml
$ oc get routes/hello-route  -o yaml > hello-route.yaml

$ for file in hello-is.yaml hello-dc.yaml hello-svc.yaml hello-route.yaml
> do
> oc create -f $file
> done
imagestream "apache-httpd" created
deploymentconfig "hello" created
service "hello" created
route "hello-route" created

$ oc get all
is/apache-httpd   docker-registry.default.svc:5000/common/apache-httpd   latest    9 seconds ago
dc/hello   1          1         1         config,image(apache-httpd:latest)
rc/hello-1   1         1         1         9s
routes/hello-route   hello.apps.lab.example.com            hello      8080-tcp                 None
svc/hello   172.30.251.245   <none>        8080/TCP   9s
po/hello-1-deploy   1/1       Running   0          9s
po/hello-1-ng20z    1/1       Running   0          5s
```

## Deploy Wordpress & MySQL directly from Pod Definition File

**1. Create NFS Share for MySQL database & Wordpress**

```
$ ssh root@services
# mkdir /exports/wordpress /exports/mysql-wp
# chown nfsnobody:nfsnobody /exports/wordpress
# chown nfsnobody:nfsnobody /exports/mysql-wp
# chmod 777 /exports/wordpress/mysql-wp
# chmod 777 /exports/wordpress

# vi /etc/exports.d/openshift-ansible.exports
  ...
  "/exports/mysql-wp" *(rw,async,all_squash)
  "/exports/wordpress" *(rw,async,all_squash)

# exportfs -av
exporting *:/exports/wordpress
exporting *:/exports/mysql-wp
...

# exportfs -s
...
/exports/mysql-wp  *(rw,async,wdelay,hide,no_subtree_check,sec=sys,secure,root_squash,all_squash)
/exports/wordpress  *(rw,async,wdelay,hide,no_subtree_check,sec=sys,secure,root_squash,all_squash)
```

**2. Create Persistent Volume for MySQL database & Wordpress**

```
# more pv-mysql-wordpress.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-wp-volume
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 3Gi
  persistentVolumeReclaimPolicy: Recycle
  nfs:
    path: /exports/mysql-wp
    server: services.lab.example.com
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpress-volume
spec:
  accessModes:
  - ReadWriteMany
  capacity:
    storage: 1Gi
  persistentVolumeReclaimPolicy: Recycle
  nfs:
    path: /exports/wordpress
    server: services.lab.example.com

# oc create -f pv-mysql-wordpress.yaml
persistentvolume "mysql-wp-volume" created
persistentvolume "wordpress-volume" created

# oc get pv
NAME              CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mysql-wp-volume   3Gi        RWX            Recycle          Available                                   7s
wordpress-volume  1Gi        RWX            Recycle          Available                                   7s
```

**3. Create Pod, Service & Persistent Volume Claim Definition Files for MySQL.**

```
$ vi pod-mysql.yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-wp
  labels:
    name: wordpress
spec:
  containers:
  - env:
    - name: MYSQL_ROOT_PASSWORD
      value: redhat
    - name: MYSQL_USER
      value: user1
    - name: MYSQL_PASSWORD
      value: redhat
    - name: MYSQL_DATABASE
      value: wordpress
    image: registry.lab.example.com/rhscl/mysql-56-rhel7
```

```
    name: mysql-wp
    ports:
    - containerPort: "3306"
      protocol: TCP
    resources:
      limits:
        cpu: "0.5"
    volumeMounts:
    - mountPath: /var/lib/mysql/data
      name: db-volume
  volumes:
  - name: db-volume
    persistentVolumeClaim:
      claimName: mysql-wp-claim
```

```
$ vi svc-mysql.yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-wp
  labels:
    name: wordpress
spec:
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
  selector:
    name: mysql-wp
  type: ClusterIP
```

```
$ vi pvc-mysql.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-wp-claim
  labels:
    name: wordpress
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
```

**4. Create Pod, Service & Persistent Volume Definition Files for Wordpress.**

```
$ vi pod-wordpress.yaml
apiVersion: v1
kind: Pod
metadata:
  name: wordpress
  labels:
    name: wordpress
spec:
  containers:
  - env:
    - name: WORDPRESS_DB_USER
      value: user1
    - name: WORDPRESS_DB_PASSWORD
      value: redhat
    - name: WORDPRESS_DB_NAME
      value: wordpress
    - name: WORDPRESS_DB_HOST
      value: mysql-wp
    image: registry.lab.example.com/do280/mywordpress
    name: wordpress
    ports:
    - containerPort: 8080
      protocol: TCP
    - containerPort: 8443
      protocol: TCP
    volumeMounts:
    - mountPath: /var/www/html
      name: wordpress-volume
  volumes:
  - name: wordpress-volume
    persistentVolumeClaim:
      claimName: wordpress-claim
```

```
$ vi svc-wordpress.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress
```

```
  labels:
    name: wordpress
spec:
  ports:
  - name: 8080-tcp
    port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    name: wordpress
  type: ClusterIP
```

```
$ vi pvc-wordpress.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: wordpress-claim
  labels:
    name: wordpress
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

**5. Build Pod, Service & Persistent Volume Claim for MySQL Database**

```
$ oc create -f pod-mysql.yaml
pod "mysql-wp" created

$ oc create -f svc-mysql.yaml
service "mysql-wp" created

$ oc create -f pvc-mysql.yaml
persistentvolumeclaim "mysql-wp-claim" created

$ oc logs -f po/mysql-wp
=> sourcing 20-validate-variables.sh ...
=> sourcing 25-validate-replication-variables.sh ...
=> sourcing 30-base-config.sh ...
---> 04:28:03     Processing basic MySQL configuration files ...
=> sourcing 60-replication-config.sh ...
=> sourcing 70-s2i-config.sh ...
---> 04:28:03     Processing additional arbitrary  MySQL configuration provided by s2i ...
…

2019-01-18 04:28:10 1 [Warning] 'user' entry '@mysql-wp' ignored in --skip-name-resolve mode.
2019-01-18 04:28:10 1 [Warning] 'proxies_priv' entry '@ root@mysql-wp' ignored in --skip-name-resolve mode.
2019-01-18 04:28:10 1 [Note] Event Scheduler: Loaded 0 events
2019-01-18 04:28:10 1 [Note] /opt/rh/rh-mysql56/root/usr/libexec/mysqld: ready for connections.
Version: '5.6.39'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  MySQL Community Server (GPL)

$ oc get all
NAME          READY     STATUS    RESTARTS   AGE
po/mysql-wp   1/1       Running   0          2m

NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
svc/mysql-wp   ClusterIP   172.30.200.26   <none>        3306/TCP   1m

$ oc get pvc
NAME            STATUS    VOLUME            CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-wp-claim   Bound     mysql-wp-volume   3Gi        RWX                           1m

# oc get pv|grep mysql
mysql-wp-volume    3Gi        RWX            Recycle          Bound       wordpress/mysql-wp-claim
```

**6. Build Pod, Service & Persistent Volume Claim for Wordpress.**

```
$ oc create -f pod-wordpress.yaml
pod "wordpress" created

$ oc create -f svc-wordpress.yaml
service "wordpress" created

$ oc create -f pvc-wordpress.yaml
persistentvolumeclaim "wordpress-claim" created

$ oc logs -f po/wordpress
=> sourcing 20-copy-config.sh ...
---> 04:32:34     Processing additional arbitrary httpd configuration provided by s2i ...
=> sourcing 00-documentroot.conf ...
=> sourcing 50-mpm-tuning.conf ...
=> sourcing 40-ssl-certs.sh ...
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.128.0.152.
Set the 'ServerName' directive globally to suppress this message
```

```
…
[Fri Jan 18 04:32:34.912772 2019] [lbmethod_heartbeat:notice] [pid 1] AH02282: No slotmem from
mod_heartmonitor
[Fri Jan 18 04:32:35.093278 2019] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.27 (Red Hat)
OpenSSL/1.0.1e-fips configured -- resuming normal operations
[Fri Jan 18 04:32:35.093351 2019] [core:notice] [pid 1] AH00094: Command line: 'httpd -D FOREGROUND'
```

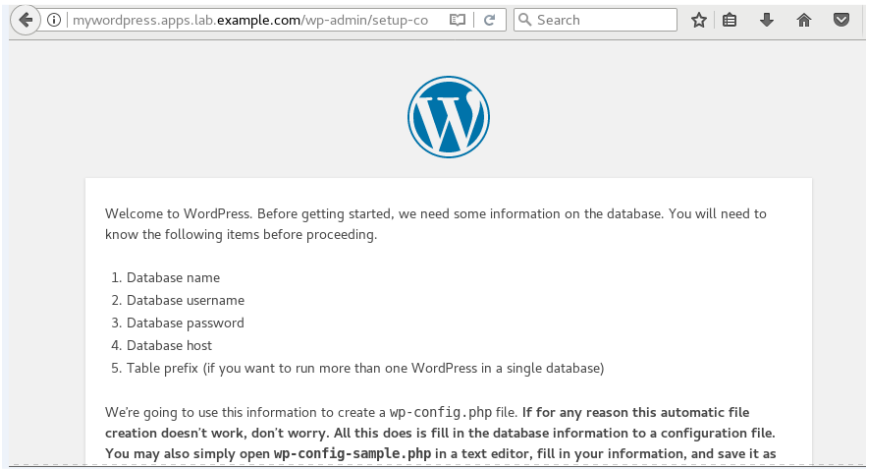
**6. Verify MySQL & Wordpress Configuration**

```
$ oc get all
NAME           READY     STATUS    RESTARTS   AGE
po/mysql-wp    1/1       Running   0          6m
po/wordpress   1/1       Running   0          2m

NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
svc/mysql-wp     ClusterIP   172.30.200.26   <none>        3306/TCP   5m
svc/wordpress    ClusterIP   172.30.9.51     <none>        8080/TCP   1m

$ oc get pvc
NAME              STATUS    VOLUME             CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-wp-claim    Bound     mysql-wp-volume    3Gi        RWX                           6m
wordpress-claim   Bound     wordpress-volume   1Gi        RWX                           2m
```

```
$ oc expose svc/wordpress --hostname=wordpress.apps.lab.example.com
route "wordpress" exposed

$ oc get route
NAME        HOST/PORT                          PATH      SERVICES    PORT       TERMINATION   WILDCARD
wordpress   wordpress.apps.lab.example.com               wordpress   8080-tcp                 None

$ oc describe route/wordpress
Name:                   wordpress
Namespace:              wordpress
Created:                21 seconds ago
Labels:                 name=wordpress
Annotations:            <none>
Requested Host:                 wordpress.apps.lab.example.com
                          exposed on router router 21 seconds ago
Path:                   <none>
TLS Termination:        <none>
Insecure Policy:        <none>
Endpoint Port:          8080-tcp

Service:        wordpress
Weight:         100 (100%)
Endpoints:      10.128.0.152:8080, 10.129.0.163:8080
```

```
$ ping wordpress.apps.lab.example.com
PING wordpress.apps.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from node1.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64 time=0.309 ms
64 bytes from node1.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64 time=0.322 ms
^C
--- wordpress.apps.lab.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.309/0.315/0.322/0.018 ms
```

```
$ curl -I http://wordpress.apps.lab.example.com
HTTP/1.1 302 Found
Date: Fri, 18 Jan 2019 04:37:28 GMT
Server: Apache/2.4.27 (Red Hat) OpenSSL/1.0.1e-fips
Location: http://wordpress.apps.lab.example.com/wp-admin/setup-config.php
Content-Type: text/html; charset=UTF-8
Set-Cookie: b604f1ae24875dc228e0eb865799d470=35df2572bc0acbe2842652bb99b311e1; path=/; HttpOnly
```



## GOGS Installation

**GOGS Template**

```
$ cat gogs-template.yaml
kind: Template
apiVersion: v1
metadata:
  annotations:
    description: The Gogs git server (https://gogs.io/)
    tags: instant-app,gogs,go,golang
  name: gogs

objects:
- kind: ServiceAccount
  apiVersion: v1
  metadata:
    creationTimestamp: null
    labels:
      app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}

- kind: Service
  apiVersion: v1
  metadata:
    annotations:
      description: Exposes the database server
    name: ${APPLICATION_NAME}-postgresql
  spec:
    ports:
    - name: postgresql
      port: 5432
      targetPort: 5432
    selector:
      name: ${APPLICATION_NAME}-postgresql

- kind: DeploymentConfig                # for Postgres
  apiVersion: v1
  metadata:
    annotations:
      description: Defines how to deploy the database
    name: ${APPLICATION_NAME}-postgresql
    labels:
      app: ${APPLICATION_NAME}
  spec:
    replicas: 1
    selector:
      name: ${APPLICATION_NAME}-postgresql
    strategy:
      type: Recreate
    template:
      metadata:
        labels:
          name: ${APPLICATION_NAME}-postgresql
        name: ${APPLICATION_NAME}-postgresql
      spec:
        serviceAccountName: ${APPLICATION_NAME}
        containers:
        - env:
          - name: POSTGRESQL_USER
            value: ${DATABASE_USER}
          - name: POSTGRESQL_PASSWORD
            value: ${DATABASE_PASSWORD}
          - name: POSTGRESQL_DATABASE
            value: ${DATABASE_NAME}
          - name: POSTGRESQL_MAX_CONNECTIONS
            value: ${DATABASE_MAX_CONNECTIONS}
          - name: POSTGRESQL_SHARED_BUFFERS
            value: ${DATABASE_SHARED_BUFFERS}
          - name: POSTGRESQL_ADMIN_PASSWORD
            value: ${DATABASE_ADMIN_PASSWORD}
          image: ' '
          livenessProbe:
            initialDelaySeconds: 30
            tcpSocket:
              port: 5432
            timeoutSeconds: 1
          name: postgresql
          ports:
          - containerPort: 5432
          readinessProbe:
            exec:
```

```
            command:
            - /bin/sh
            - -i
            - -c
            - psql -h 127.0.0.1 -U ${POSTGRESQL_USER} -q -d ${POSTGRESQL_DATABASE} -c 'SELECT 1'
          initialDelaySeconds: 5
          timeoutSeconds: 1
        resources:
          limits:
            memory: 512Mi
        volumeMounts:
        - mountPath: /var/lib/pgsql/data
          name: gogs-postgres-data
      volumes:
      - name: gogs-postgres-data
        emptyDir: {}
    triggers:
    - imageChangeParams:
        automatic: true
        containerNames:
        - postgresql
        from:
          kind: ImageStreamTag              # container postgres come from Image Stream postgresql:9.5
          name: postgresql:9.5
          namespace: openshift
      type: ImageChange
    - type: ConfigChange

- kind: Service
  apiVersion: v1
  metadata:
    annotations:
      description: The Gogs server's http port
      service.alpha.openshift.io/dependencies: '[{"name":"${APPLICATION_NAME}-
postgresql","namespace":"","kind":"Service"}]'
    labels:
      app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}
  spec:
    ports:
    - name: 3000-tcp
      port: 3000
      protocol: TCP
      targetPort: 3000
    selector:
      app: ${APPLICATION_NAME}
      deploymentconfig: ${APPLICATION_NAME}
    sessionAffinity: None
    type: ClusterIP
  status:
    loadBalancer: {}

- kind: Route
  apiVersion: v1
  id: ${APPLICATION_NAME}-http
  metadata:
    annotations:
      description: Route for application's http service.
    labels:
      app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}
  spec:
    host: ${HOSTNAME}
    to:
      name: ${APPLICATION_NAME}

- kind: DeploymentConfig             # for GOGS
  apiVersion: v1
  metadata:
    labels:
      app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}
  spec:
    replicas: 1
    selector:
      app: ${APPLICATION_NAME}
      deploymentconfig: ${APPLICATION_NAME}
    strategy:
      resources: {}
      rollingParams:
        intervalSeconds: 1
        maxSurge: 25%
        maxUnavailable: 25%
        timeoutSeconds: 600
```

```
      updatePeriodSeconds: 1
    type: Rolling
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: ${APPLICATION_NAME}
        deploymentconfig: ${APPLICATION_NAME}
    spec:
      serviceAccountName: ${APPLICATION_NAME}
      containers:
      - image: " "
        imagePullPolicy: Always
        name: ${APPLICATION_NAME}
        ports:
        - containerPort: 3000
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        volumeMounts:
        - name: gogs-data
          mountPath: /opt/gogs/data
        - name: gogs-config
          mountPath: /etc/gogs/conf
        readinessProbe:
            httpGet:
              path: /
              port: 3000
              scheme: HTTP
            initialDelaySeconds: 3
            timeoutSeconds: 1
            periodSeconds: 20
            successThreshold: 1
            failureThreshold: 3
        livenessProbe:
            httpGet:
              path: /
              port: 3000
              scheme: HTTP
            initialDelaySeconds: 3
            timeoutSeconds: 1
            periodSeconds: 10
            successThreshold: 1
            failureThreshold: 3
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
      - name: gogs-data
        emptyDir: {}
      - name: gogs-config
        configMap:
          name: gogs-config
          items:
            - key: app.ini
              path: app.ini
  test: false
  triggers:
  - type: ConfigChange
  - imageChangeParams:
      automatic: true
      containerNames:
      - ${APPLICATION_NAME}
      from:
        kind: ImageStreamTag
        name: ${APPLICATION_NAME}:${GOGS_VERSION}
    type: ImageChange

- kind: ImageStream            # for GOGS, IS from Postgres already exist
  apiVersion: v1
  metadata:
    labels:
      app: ${APPLICATION_NAME}
    name: ${APPLICATION_NAME}
  spec:
    tags:
    - name: "${GOGS_VERSION}"
      from:
         kind: DockerImage
         name: docker.io/openshiftdemos/gogs:${GOGS_VERSION}  <-- Replace with:
                                                              registry.lab.example.com/do280/gogs
      importPolicy: {}
      annotations:
```

```
        description: The Gogs git server docker image
        tags: gogs,go,golang
        version: "${GOGS_VERSION}"
```

**- kind: ConfigMap**

```
  apiVersion: v1
  metadata:
    name: gogs-config
    labels:
      app: ${APPLICATION_NAME}
  data:
    app.ini: |
      RUN_MODE = prod
      RUN_USER = gogs

      [database]
      DB_TYPE  = postgres
      HOST     = ${APPLICATION_NAME}-postgresql:5432
      NAME     = ${DATABASE_NAME}
      USER     = ${DATABASE_USER}
      PASSWD   = ${DATABASE_PASSWORD}

      [repository]
      ROOT = /opt/gogs/data/repositories

      [server]
      ROOT_URL=http://${HOSTNAME}
      SSH_DOMAIN=${HOSTNAME}

      [security]
      INSTALL_LOCK = ${INSTALL_LOCK}

      [service]
      ENABLE_CAPTCHA = false

      [webhook]
      SKIP_TLS_VERIFY = ${SKIP_TLS_VERIFY}
```

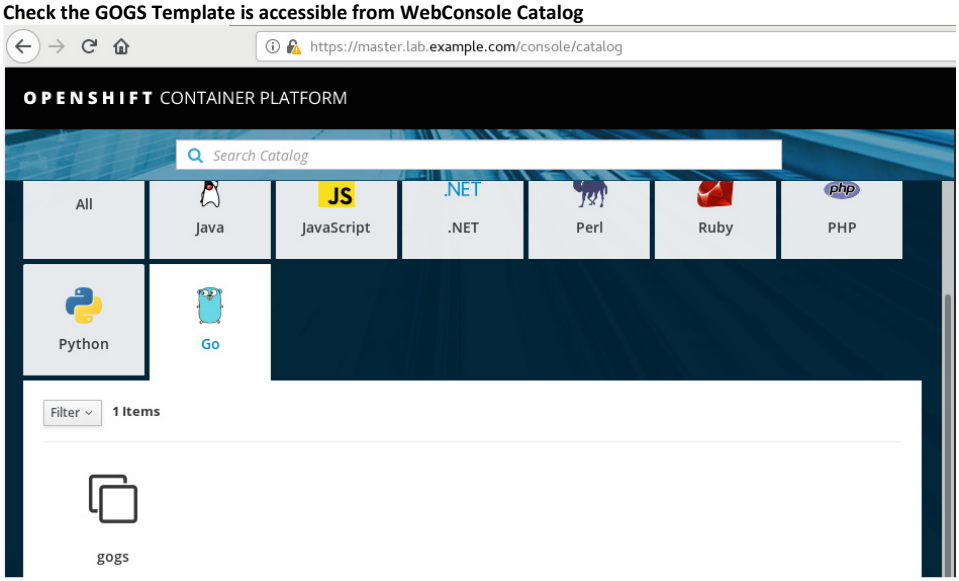
**parameters:**

```
- description: The name for the application.
  name: APPLICATION_NAME
  required: true
  value: gogs
- description: 'Custom hostname for http service route.  Leave blank for default hostname, e.g.:
<application-name>-<project>.<default-domain-suffix>'
  name: HOSTNAME
  required: true
- displayName: Database Username
  from: gogs
  value: gogs
  name: DATABASE_USER
- displayName: Database Password
  from: '[a-zA-Z0-9]{8}'
  value: gogs
  name: DATABASE_PASSWORD
- displayName: Database Name
  name: DATABASE_NAME
  value: gogs
- displayName: Database Admin Password
  from: '[a-zA-Z0-9]{8}'
  generate: expression
  name: DATABASE_ADMIN_PASSWORD
- displayName: Maximum Database Connections
  name: DATABASE_MAX_CONNECTIONS
  value: "100"
- displayName: Shared Buffer Amount
  name: DATABASE_SHARED_BUFFERS
  value: 12MB
- name: GOGS_VERSION
  displayName: Gogs Version
  description: 'Version of the Gogs container image to be used (check the available version
https://hub.docker.com/r/openshiftdemos/gogs/tags)'
  value: "0.9.97"
  required: true
- name: INSTALL_LOCK
  displayName: Installation lock
  description: 'If set to true, installation (/install) page will be disabled. Set to false if you want to
run the installation wizard via web'
  value: "true"
- name: SKIP_TLS_VERIFY
  displayName: Skip TLS verification on webhooks
  description: Skip TLS verification on webhooks. Enable with caution!
  value: "false"
```

**Create Resources Template in OpenShift Namespace**

```
$ oc login -u system:admin

$ oc create -f gogs-template.yaml -n openshift
template "gogs" created

$ oc get template -n openshift|grep gogs
gogs         The Gogs git server (https://gogs.io/)      11 (1 blank)      8
```

**Check the GOGS Template is accessible from WebConsole Catalog**




**Explore the GOGS Template Wizard**

* Add to Project

Select or create project

* APPLICATION_NAME

gogs

The name for the application.

* HOSTNAME

Custom hostname for http service route. Leave blank for default hostname, e.g.: <application-name>-<project>.<default-domain-suffix>

Database Username

gogs

Database Password

gogs

Database Name

gogs

Database Admin Password

*(generated if empty)*

Maximum Database Connections

100

Shared Buffer Amount

12MB

* Gogs Version

0.9.97

Version of the Gogs container image to be used (check the available version https://hub.docker.com/r/openshiftdemos/gogs/tags)

**Installation lock**

true

If set to true, installation (/install) page will be disabled. Set to false if you want to run the installation wizard via web

**Skip TLS verification on webhooks**

false

Skip TLS verification on webhooks. Enable with caution!

## Labels

About Labels

Each label is applied to each created resource.

*Name* | *Value*

Add Label

**Create GOGS Application Directly from the Template File**

```
$ oc new-app gogs-template.yaml -p HOSTNAME=gogs.example.com
--> Deploying template "gogs/gogs" for "gogs-template.yaml" to project gogs
     gogs
     ---------
     The Gogs git server (https://gogs.io/)

     * With parameters:
        * APPLICATION_NAME=gogs
        * HOSTNAME=gogs.example.com
        * Database Username=gogs
        * Database Password=gogs
        * Database Name=gogs
        * Database Admin Password=IkwDijBk # generated
        * Maximum Database Connections=100
        * Shared Buffer Amount=12MB
        * Gogs Version=0.9.97
        * Installation lock=true
        * Skip TLS verification on webhooks=false

--> Creating resources ...
    serviceaccount "gogs" created
    service "gogs-postgresql" created
    deploymentconfig "gogs-postgresql" created
    service "gogs" created
    route "gogs" created
    deploymentconfig "gogs" created
    imagestream "gogs" created
    configmap "gogs-config" created
--> Success
    Access your application via route 'gogs.example.com'
    Run 'oc status' to view your app.
```