# **Continues Integration and Continues Delivery with Jenkins GIT Docker Swarms and Docker Registries.**

Today I want share my solution to setup CI/CD with GIT(GOGS) Jenkins Docker swarms, Docker Registry.

Requirements:

1. Docker Swarms:

I`ll not explain how to create docker swarm. Also all swarm nodes connected to share storage, but each swarm has his own NFS share. In my case it`s NFS share mounted to folder /external (it`s important that all nodes have same connection in /etc/fstab):

Example:

10.148.138.10:/docker /external nfs rw,suid,dev,exec,auto,user,async 0 0

2. Ubuntu 18.04 server used as OS for servers.

3. Harbor docker container used as registry (<https://goharbor.io/> <https://github.com/goharbor/harbor> )

4. Jenkins installed as a ubuntu package (<https://pkg.jenkins.io/debian-stable/>) Official Site (<https://jenkins.io/download/> ).

5. Ansible

6. Portainer as WebUI for docker swarms (<https://www.portainer.io/installation/> ; <https://portainer.readthedocs.io/en/stable/deployment.html>).

7. Traefik (reverse proxy https://docs.traefik.io/configuration/backends/docker/).

Structure:

We need few docker swarms: dev, qa, prod. I remanding to have 3 manages node (or more) and 1 (or more) worker node in swarms. Also I use one more swarm for git, portainer, redmine, redis and etc. (more information in <https://docs.docker.com/engine/swarm/admin_guide/> (Add manager nodes for fault tolerance )). 3 registry Dev, QA, Prod. Also I use dedicated server (virtual) for ansible. Jenkins installed on docker manager host (it`s important) , jenkins slave workers have to be installed only on docker managers hosts(if jenkins slave will run on docker worker it not able to create docker stack). All swarm have to have portainer agent up and running. Prod swarm have to have traefik on manager nodes

All servers have to be able to connect each other by ssh with user without password with su privilages.

Here is steps for initial ubuntu setup:

1. Create RSA key : ssh-keygen -o -a 100 -b 4096 -t RSA -f ~/.ssh/id\_robot -C "robot\_com"

2. ssh-copy-id -i ~/.ssh/id\_robot.pub robot@%ansible\_server% where %ansible\_server% - your FQDN ansible server name with existed user robot.

2a. I have administrator user with root privileges and here is how to create user robot: (# - root privileges or use sudo if you see $)

# ssh administrator@%ansible\_server%

$ sudo adduser robot

I create new user group without password for su commands with name: sudonp also need install some components for ansible. Command is:

sudo groupadd sudonp && sudo usermod -a -G sudonp administrator \

&& sudo usermod -a -G sudonp robot && sudo apt install python \

&& sudo rm -rf /etc/localtime && sudo ln -s /usr/share/zoneinfo/Europe/Kiev /etc/localtime \

&& sudo visudo && exit

When visudo is open I add next:

# Allow sudonp grup execute su command without password confirmation.

%sudonp ALL=(ALL:ALL) NOPASSWD: ALL

and type ESC > w > q :)

2b. Docker setup:

curl -fsSL [https://get.docker.com](https://get.docker.com/) -o get-docker.sh

sudo sh get-docker.sh

don`t forget add user(s) to docker group.

3. ssh-copy-id -i ~/.ssh/id\_robot.pub robot@%server\_name%

4. check passwordless ssh connection: ssh robot@%server\_name%

Make changes in your project:

1. clone to your pc <https://github.com/SkazochnikZlodey/gogs_jenkins>

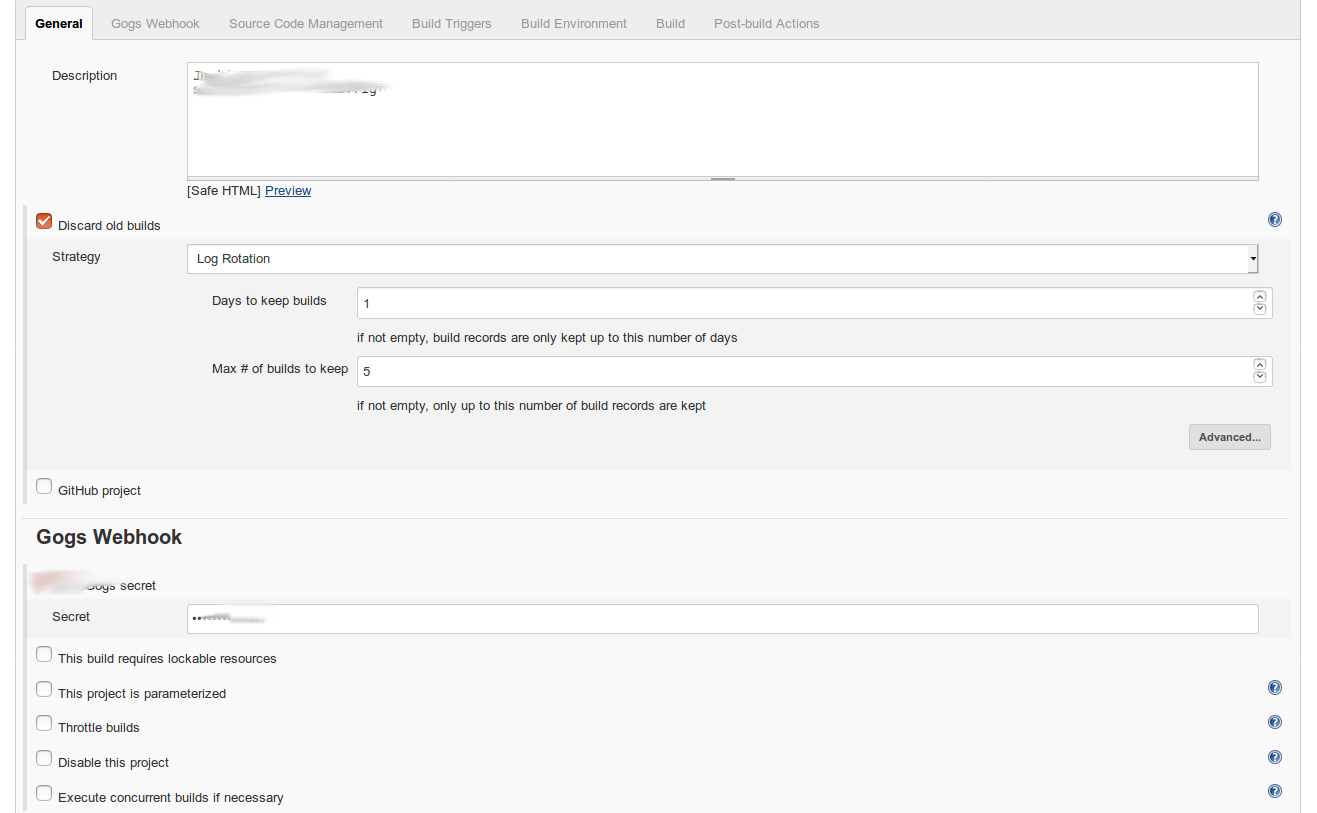
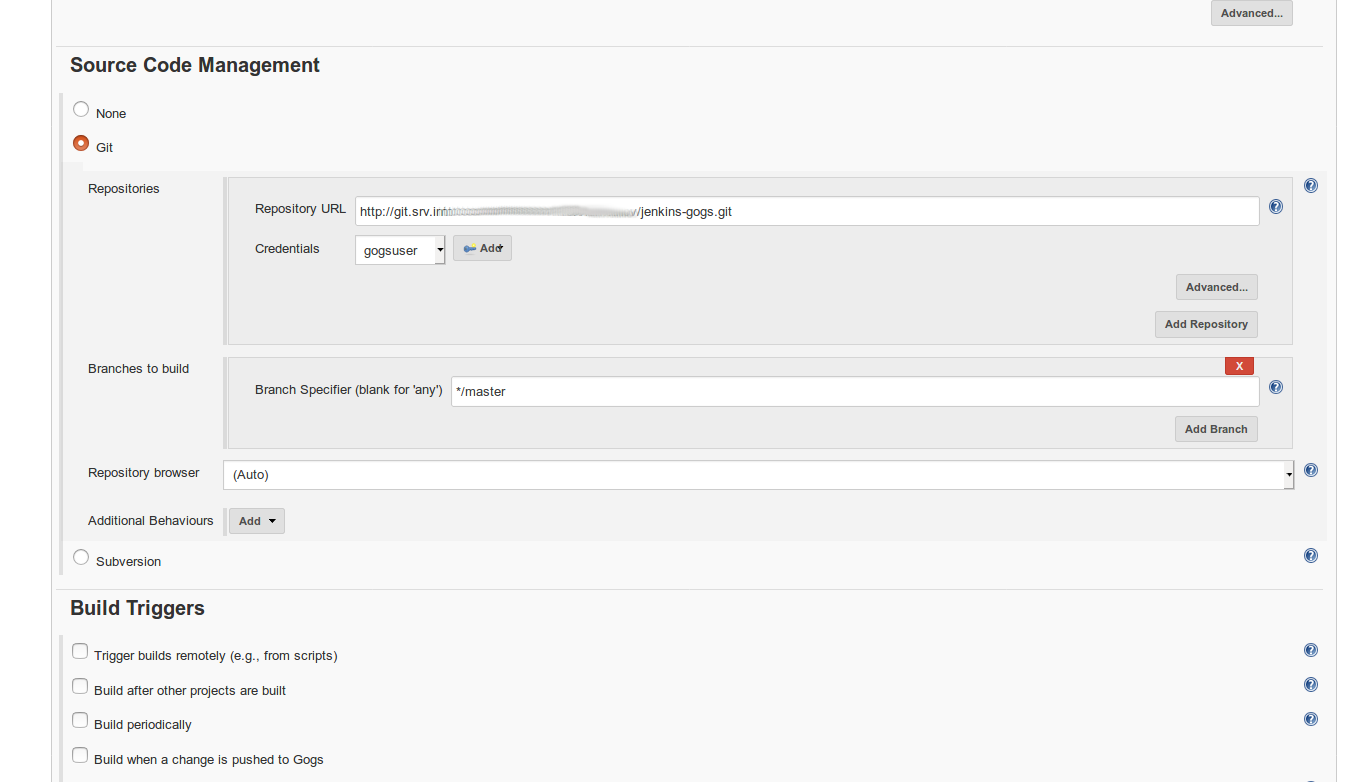
2. Create folder code in your project and move your code to folder code (or copy folder code as example from cloned project).

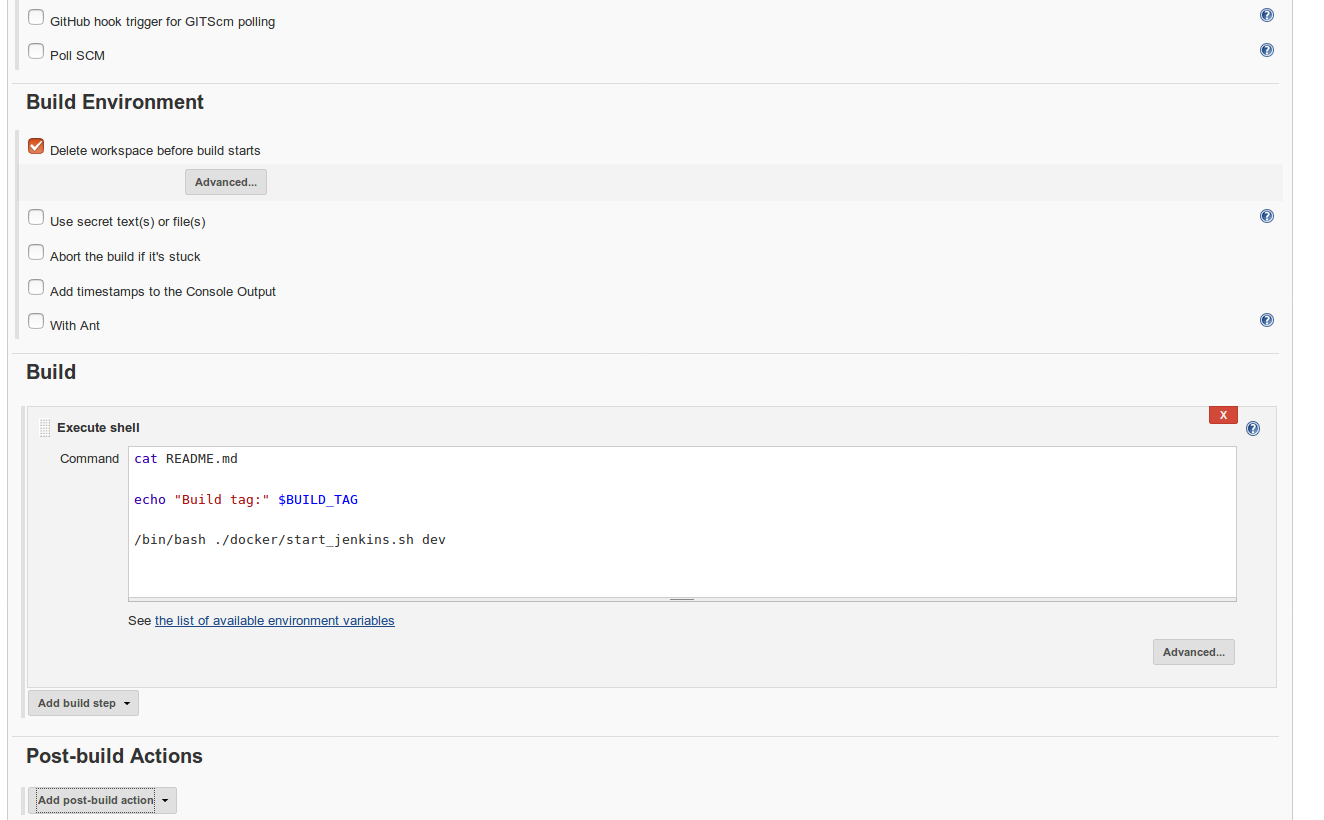
3. copy folder docker from cloned project to your project into folder docker.

4.Commit and push your changes into your git server.

Setup jenkins (dev edition):

New item > free style project > (name 10192\_Jenkins-GoGs-Trig or 10192\_Jenkins-GoGs-Trig\_Dev)





/bin/bash ./docker/start\_jenkins.sh dev – mean: starting development setup:

Build containers and upload them to dev registry after this start stack with name enkins-GoGs-Trig and accces port 10192 (name and port take from trig name !!!)

Setup jenkins (QA edition):

New item > free style project > (name 10192\_Jenkins-GoGs-Trig\_QA)

execute shell:

echo "Build tag:" $BUILD\_TAG

./docker/start\_jenkins.sh qa

Setup jenkins (Prod edition):

New item > free style project > (name 10192\_Jenkins-GoGs-Trig\_Prod)

execute shell:

echo "Build tag:" $BUILD\_TAG

export publicsitename="JGTrig.example.com"

./docker/start\_jenkins.sh prod

Some info:

export publicsitename="JGTrig.example.com"

provide site name for internet access.

After build in docker container will be environment variable with name:BUILD\_ENV

with dev or qa or prod.

Let`s take closer look into script(s):

start\_jenkins.sh

In project \*.tpl files used as initial template. This files existed in some additional folders inside of docker folders.

Script has array(s) with steps what to do:

|  |
| --- |
| step\_1=("websrv" "create" "default.conf" $endofsitename) |

|  |
| --- |
| step\_2=("websrv" "use" "default.conf" "replace" " server tplphp:9000" " server phpsrv1:9000") |

|  |
| --- |
| #step\_3=("nginx" "use" "default.conf" "add" " server phpsrv1:9000" " server phpsrv2:9000;") |

|  |
| --- |
| #step\_4=("nginx" "use" "default.conf" "add" " server phpsrv2:9000" " server phpsrv3:9000") |

|  |
| --- |
| #step\_5=("nginx" "use" "default.conf" "delete" " server phpsrv3:9000") |

|  |
| --- |
| step\_3=("websrv" "create" "Dockerfile" $endofsitename) |

|  |
| --- |
| step\_4=("phpsrv" "create" "Dockerfile" $endofsitename) |

|  |
| --- |
| step\_5=("websrv" "use" "Dockerfile" "replace" "tplwebsrv" "websrv") |

|  |
| --- |
| step\_6=("." "create" "docker-compose.yml" $endofsitename) |

step\_7=("." "use" "docker-compose.yml" "replace" "- tplphp" " - phpsrv1")

step1:copy file default.conf.tpl in websrv folder to default.conf

step2:replace in default.conf part of string " server tplphp:9000" to " server phpsrv1:9000" (space are important !!!)

commented (not used) step3: #step\_3=("nginx" "use" "default.conf" "add" " server phpsrv1:9000" " server phpsrv2:9000;")

add " server phpsrv2:9000;" by next line after " server phpsrv1:9000"

commented (not used) step5: #step\_5=("nginx" "use" "default.conf" "delete" " server phpsrv3:9000")

delete part " server phpsrv3:9000" of string from file default.conf

step6: copy file docker-compose.yml in current folder to docker-compose.yml

step\_16=("websrv" "build" "Dockerfile" ${sourceservername} ${projectname} "${sitename}")

build container with with sources websrv

bash ./${dockerdir}/scripts/build.sh "./${dockerdir}/" "deploy" "docker-compose.yml" "${projectname}" "${sitename}"

deploy docker stack :)

Let`s take closer look into tamplate(s):

./docker-compose.yuml

|  |
| --- |
| #labels: |

|  |
| --- |
| #- "traefik.backend.loadbalancer.sticky=false" |

|  |
| --- |
| #- "traefik.backend.loadbalancer.swarm=true" |

|  |
| --- |
| #- "traefik.backend=tplbackendsitename" |

|  |
| --- |
| #- "traefik.docker.network=traefiknet" |

|  |
| --- |
| #- "traefik.entrypoints=https" |

|  |
| --- |
| #- "traefik.frontend.passHostHeader=true" |

|  |
| --- |
| #- "traefik.frontend.rule=Host:tplpublicsitename" |

#- "traefik.port=99999"

changed by:

|  |
| --- |
| step\_11=("." "use" "docker-compose.yml" "replace" "#labels:" "labels:") |

|  |
| --- |
| step\_12=("." "use" "docker-compose.yml" "replace" '#- "traefik.backend.loadbalancer.sticky=false"' '- "traefik.backend.loadbalancer.sticky=false"') |

|  |
| --- |
| step\_13=("." "use" "docker-compose.yml" "replace" '#- "traefik.backend.loadbalancer.swarm=true"' '- "traefik.backend.loadbalancer.swarm=true"') |

|  |
| --- |
| step\_14=("." "use" "docker-compose.yml" "replace" '#- "traefik.backend=tplbackendsitename"' '- "traefik.backend=tplbackendsitename"') |

|  |
| --- |
| step\_15=("." "use" "docker-compose.yml" "replace" '#- "traefik.docker.network=traefiknet"' '- "traefik.docker.network=traefiknet"') |

|  |
| --- |
| step\_16=("." "use" "docker-compose.yml" "replace" '#- "traefik.entrypoints=https"' '- "traefik.entrypoints=https"') |

|  |
| --- |
| step\_17=("." "use" "docker-compose.yml" "replace" '#- "traefik.frontend.passHostHeader=true"' '- "traefik.frontend.passHostHeader=true"') |

|  |
| --- |
| step\_18=("." "use" "docker-compose.yml" "replace" '#- "traefik.frontend.rule=Host:tplpublicsitename"' '- "traefik.frontend.rule=Host:tplpublicsitename"') |

|  |
| --- |
| step\_19=("." "use" "docker-compose.yml" "replace" '#- "traefik.port=99999"' '- "traefik.port=tmp99999"') |

|  |
| --- |
| step\_20=("." "use" "docker-compose.yml" "replace" '# - traefiknet' ' - traefiknet') |

|  |
| --- |
| step\_21=("." "use" "docker-compose.yml" "replace" '# traefiknet:' ' traefiknet:') |

|  |
| --- |
| step\_22=("." "use" "docker-compose.yml" "replace" '# external: true' ' external: true') |

|  |
| --- |
| step\_23=("." "use" "docker-compose.yml" "replace" 'tplbackendsitename' ${sitename,,}) |

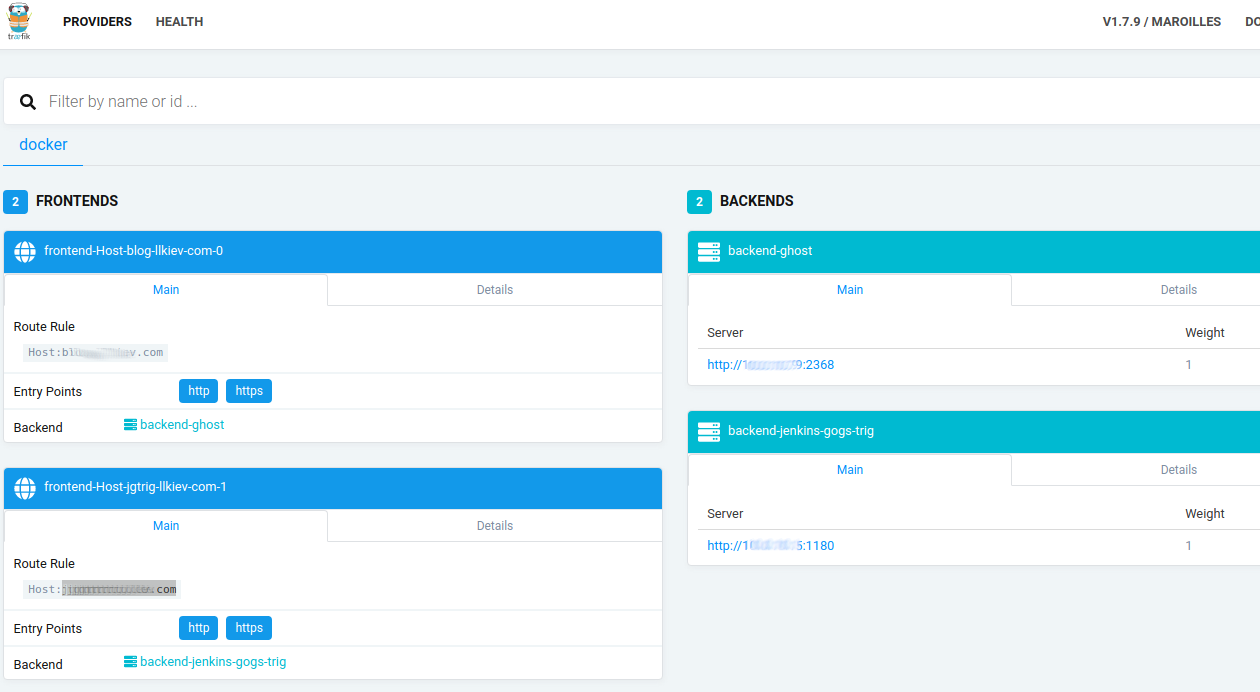
|  |
| --- |
| step\_24=("." "use" "docker-compose.yml" "replace" 'tplpublicsitename' ${tplpublicsitename,,}) |

|  |
| --- |
| # step\_25=("." "use" "docker-compose.yml" "replace" 'tmp99999' ${portnumber}) |

step\_25=("." "use" "docker-compose.yml" "replace" 'tmp99999' '1180')

and allow traefik to use container :)

result:



## Sorry for brief documentation !!!