**GitLab + Runner + Registry + CI/CD Stack Setup and Management Best Practices by GitOps #1**

[[Ahmet Vehbi Olgaç](https://medium.com/@ahmetvehbiolgac?source=post_page-----2ebb26ed9ede--------------------------------)](https://medium.com/@ahmetvehbiolgac?source=post_page-----2ebb26ed9ede--------------------------------)

[[Trendyol Tech](https://medium.com/trendyol-tech?source=post_page-----2ebb26ed9ede--------------------------------)](https://medium.com/trendyol-tech?source=post_page-----2ebb26ed9ede--------------------------------)

[Ahmet Vehbi Olgaç](https://medium.com/@ahmetvehbiolgac?source=post_page-----2ebb26ed9ede--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F8e95cebf8a3f&operation=register&redirect=https%3A%2F%2Fmedium.com%2Ftrendyol-tech%2Fgitlab-runner-registry-ci-cd-stack-setup-and-management-best-practices-by-gitops-1-2ebb26ed9ede&user=Ahmet+Vehbi+Olga%C3%A7&userId=8e95cebf8a3f&source=post_page-8e95cebf8a3f----2ebb26ed9ede---------------------post_header-----------)

Published in

[Trendyol Tech](https://medium.com/trendyol-tech?source=post_page-----2ebb26ed9ede--------------------------------)

·

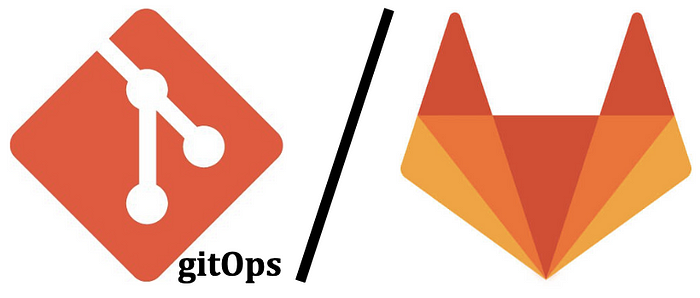
6 min read

·

Jan 10, 2020

690

1



With this article series, I wanna tell end-to-end experience and best practices, while our infrastructure transformation at [Trendyol Tech](https://www.youtube.com/channel/UCUBiayLMggBAsiYvGLzQJ5w" \t "_blank). The [company](https://www.linkedin.com/company/trendyolgroup) is the highest traffic and volumed e-commerce website in the country and region. The technology department consists of 20+ teams with 400+ members. Regarding to complexity concerns, we started to manage our resources & applications by using [GitLab](https://about.gitlab.com/) with [GitOps](https://www.weave.works/technologies/gitops" \t "_blank) methodologies, and easily track who/when/why did changes.

“GitOps: versioned CI/CD on top of declarative infrastructure. Stop scripting and start shipping.”

— [Kelsey Hightower](https://twitter.com/kelseyhightower/status/953638870888849408)

Simplicity is our **main** principle. Due to this vision/goal, we selected GitLab and achieved a complete CI/CD toolchain and [VCS](https://en.wikipedia.org/wiki/Version_control). Let’s start, how we installed/managed those pieces step-by-step. 💪

**Step 1: Provisioning Machine and Requirements**

* We are provisioning the node *(8 Core, 16GB Memory Ubuntu 16.04.6 OS)*, by installing [Docker Engine CE](https://docs.docker.com/install) and initialize [Swarm Mode](https://docs.docker.com/engine/swarm) for providing a lightweight container orchestration to serve GitLab.

#Install Docker Engine CE  
~$ curl -fsSL <https://get.docker.com/> | sh#Check Version  
~$ docker -v  
Docker version 19.03.5, build 633a0ea838#Init Docker Swarm Mode  
~$ docker swarm init --advertise-addr 192.168.10.10#Check Nodes  
~# docker node ls  
ID HOSTNAME STATUS AVAILABILITY MANAGER-STATUS ENGINE-VERSION  
uz73dfr \* node1 Ready Active Leader 19.03.5#Create Network  
~$ docker network create --driver=overlay --attachable **ty**#Check Network  
~# docker network ls  
NETWORK-ID NAME DRIVER SCOPE  
0b6cdf1 bridge bridge local  
1306494 docker\_gwbridge bridge local  
2a13a9b host host local  
gv39nyf ingress overlay swarm  
ut1oc58 **ty** overlay swarm  
57a9ce7 none null local

**Step 2: Deploy Initial GitLab**

* For LDAP Integration, obtain a read-only Service User *(with the name: GitLab Integration)*from LDAP Administrator.
* For Slack Integration, obtain Slack Hook URL and Channel from Slack Administrator.
* For JIRA Integration, obtain JIRA Integration user & pass from JIRA Administrator.
* Create a *gitlab* directory and put a *docker-compose.yml* file with content, as seen below 👇🏻.
* Create */data/gitlab* to store data files and */var/log/gitlab* to store logs.
* We selected to install *GitLab EE*, because we may buy EE in the future. For details check the [comparison table](https://about.gitlab.com/pricing/self-managed/feature-comparison) and [EE explanation](https://about.gitlab.com/install/ce-or-ee).

GitLab docker-compose.yml

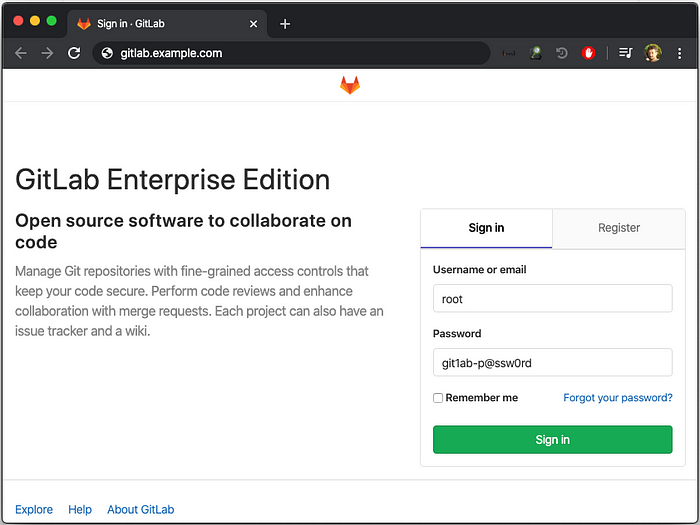
|  |
| --- |
| version: '3.7' |
|  | services: |
|  | gitlab: |
|  | image: gitlab/gitlab-ee:12.6.2-ee.0 |
|  | hostname: gitlab.example.com |
|  | environment: |
|  | GITLAB\_ROOT\_PASSWORD: 'git1ab-p@ssw0rd :)' |
|  | GITLAB\_OMNIBUS\_CONFIG: | |
|  | external\_url 'http://gitlab.example.com' |
|  |  |
|  | registry\_external\_url 'http://registry.example.com:5005' |
|  | registry['enable'] = true |
|  |  |
|  | gitlab\_rails['lfs\_enabled'] = true |
|  | gitlab\_rails['gitlab\_shell\_ssh\_port'] = 2222 |
|  | gitlab\_rails['pipeline\_schedule\_worker\_cron'] = "\*/10 \* \* \* \*" |
|  | gitlab\_rails['ldap\_enabled'] = true |
|  | gitlab\_rails['ldap\_servers'] = YAML.load <<-EOS |
|  | main: |
|  | label: 'LDAP' |
|  | host: '10.10.10.10' #LDAP Server IP Address |
|  | port: 389 |
|  | uid: 'sAMAccountName' |
|  | bind\_dn: 'CN=GitLab Integration,OU=Service Accounts,DC=example,DC=com' |
|  | password: 'ldap-p@ssw0rd :)' |
|  | encryption: 'plain' |
|  | active\_directory: true |
|  | allow\_username\_or\_email\_login: false |
|  | block\_auto\_created\_users: false |
|  | base: 'DC=example,DC=com' |
|  | EOS |
|  | networks: |
|  | - ty |
|  | ports: |
|  | - '443:443' |
|  | - '80:80' |
|  | - '2222:22' |
|  | - '5005:5005' |
|  | volumes: |
|  | - /data/gitlab/config:/etc/gitlab |
|  | - /data/gitlab/data:/var/opt/gitlab |
|  | - /var/log/gitlab:/var/log/gitlab |
|  |  |
|  | networks: |
|  | ty: |
|  | external: true |

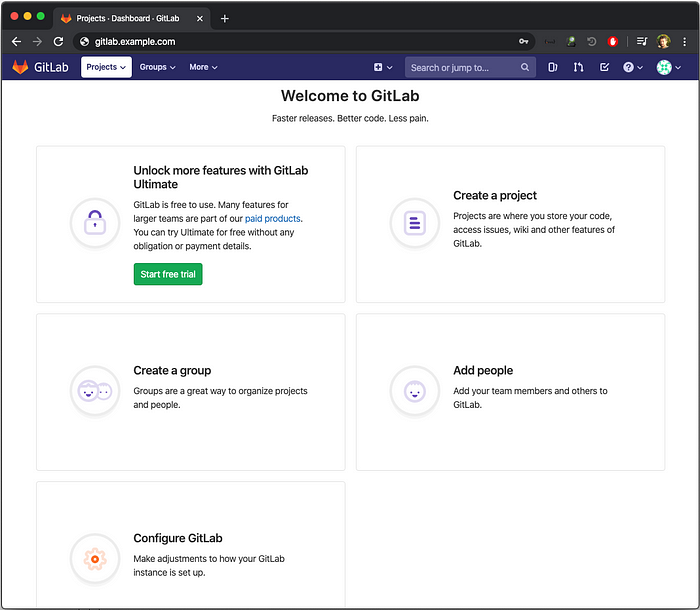
#Deploy GitLab  
docker stack deploy -c [docker-compose.yml](https://gitlab.com/olgac/gitlab/blob/master/gitlab/docker-compose.yml) gitlab

**Step 3: Important lines of the**[**docker-compose.yml**](https://gitlab.com/olgac/gitlab/blob/master/gitlab/docker-compose.yml)**file.**

* **Line 9:**GitLab URL, that you login with LDAP credential.
* **Line 11:** GitLab Container Registry URL, that you login with LDAP.
* **Line 14:** Enabling [LFS](https://docs.gitlab.com/ee/administration/lfs/manage_large_binaries_with_git_lfs.html), to store large files in the repository.
* **Line 16:** Set, Checking Scheduled CI/CD Jobs every 10 minutes.

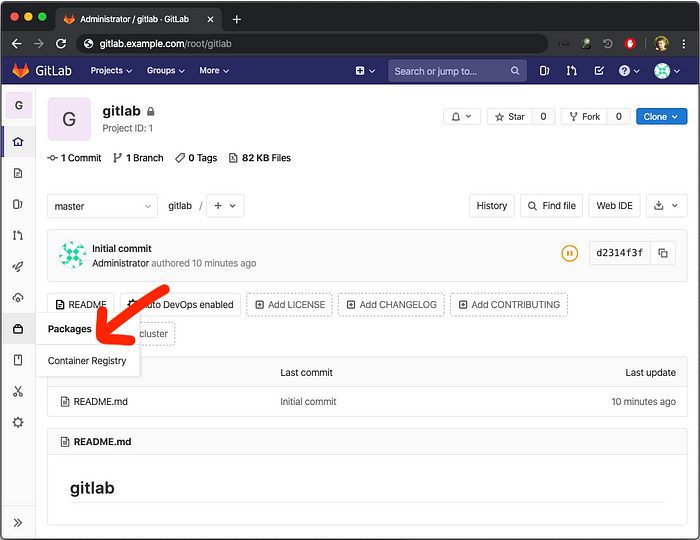
After ~2 minutes, GitLab and Container Registry are ready. Please, check the [URL](http://gitlab.example.com/) and Sign-in with root user credentials. 👊🏻



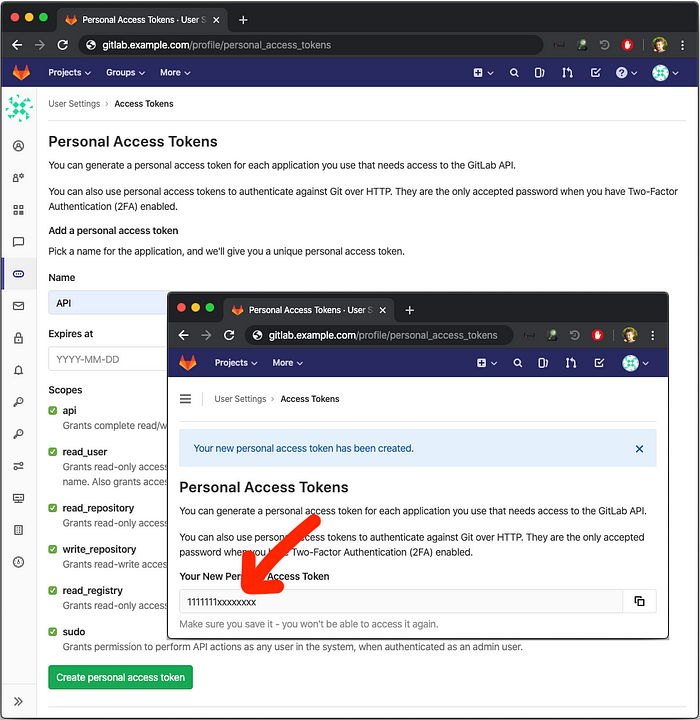


**Step 4: Prepare repository & API access credential.**

* Create a project with the name *gitlab.*

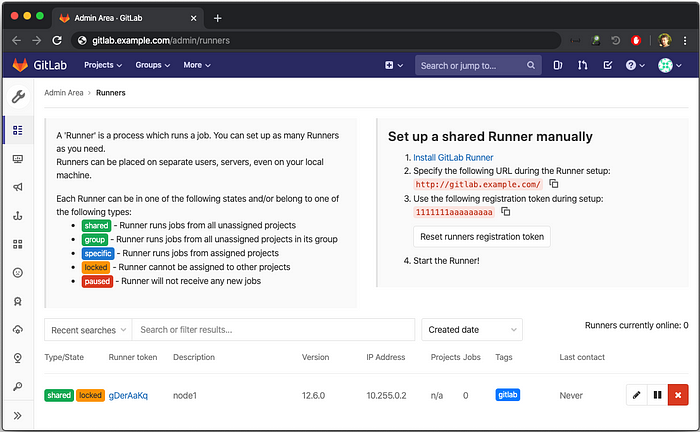


* Add all files from <https://gitlab.com/olgac/gitlab> repository to your gitlab repository.
* Create a full-powered GitLab Access Token from this [URL](http://gitlab.example.com/profile/personal_access_tokens) for accessing API to manage GitLab by GitOps methodology.



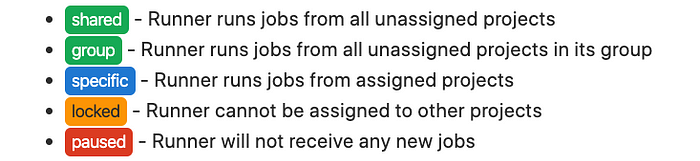
**Step 5: Install Runner**

* Run [*gitlab-runner.sh*](https://gitlab.com/olgac/gitlab/blob/master/runner/gitlab-runner.sh) script to create a runner with tag *“[gitlab](https://gitlab.com/olgac/gitlab/blob/master/runner/gitlab-runner.sh" \l "L24" \t "_blank)”*. The script installs required executables and registers a runner to runner node.
* There are some [Types of GitLab Runners](https://docs.gitlab.com/runner/executors/README.html), for different purposes. We are using 🐳 [*Docker Executor Runners*](https://docs.gitlab.com/runner/executors/docker.html). Because it runs the job in a docker container and dies. This feature gave us the power to execute any type of executables and commands with the runner, by setting image property. 💪🏻 *(.gitlab-ci.yml*[*Line 7*](https://gitlab.com/olgac/gitlab/blob/master/.gitlab-ci.yml#L7)*&*[*Line 16*](https://gitlab.com/olgac/gitlab/blob/master/.gitlab-ci.yml#L16)*)*.

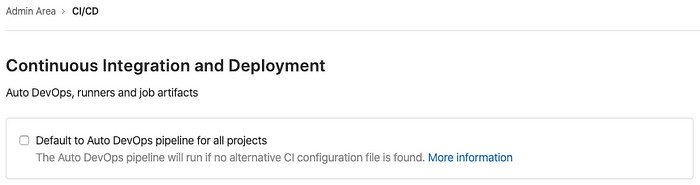


GitLab Runners List by Tags

* Each Runner can be in one of the following states and/or belong to one of the following types:



* Disable Default to Auto DevOps pipeline for all projects by unchecking on URL [http://gitlab.example.com/admin/application\_settings/ci\_cd](http://mplc-gitlab.trendyol.com/admin/application_settings/ci_cd). Because we will manage our pipelines with *.gitlab-ci.yml*



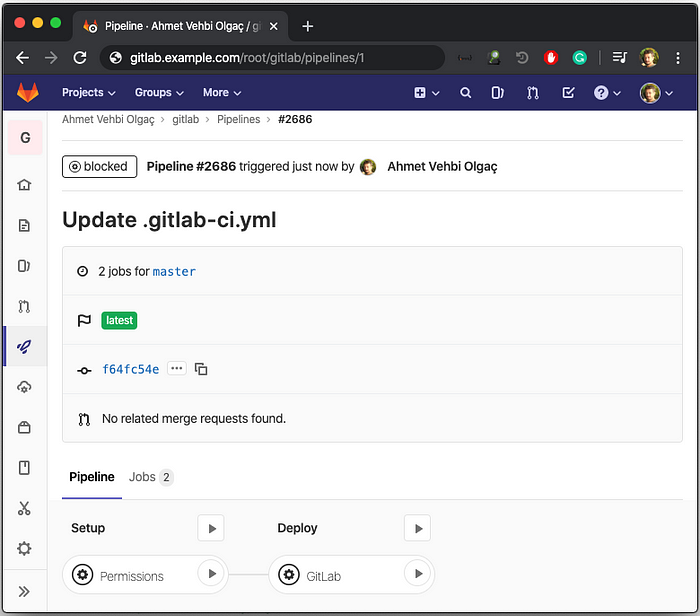
Disable Auto DevOps

**Step 6: Prepare .gitlab-ci.yml**

* With given .gitlab-ci.yml, developers can easily manage stages and jobs as-seen below.

.gitlab-ci.yml

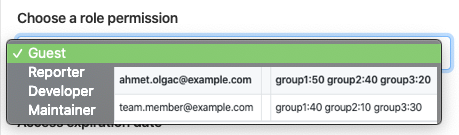
* After, registering [*Docker Executor Runner*](https://docs.gitlab.com/runner/executors/docker.html), we can run the first job of the pipeline to set-up permissions regarding *permissions.csv*.
* The second *(gitlab)*job is waiting for a manual trigger to re-deploy Gitlab Server with given parameters and version.
* The *tags*property is very important to select which job will run on which runner. If runner and job tag matches, the job executes on this runner. We are setting tag value while registering the runner in [*gitlab-runner.sh*](https://gitlab.com/olgac/gitlab/blob/master/runner/gitlab-runner.sh#L24)*script*via --tag-list property.



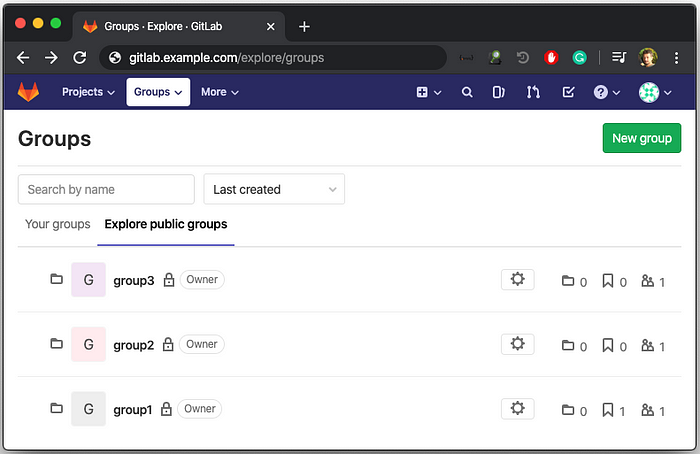
GitLab Pipeline

**Step 7: Run Pipelines**

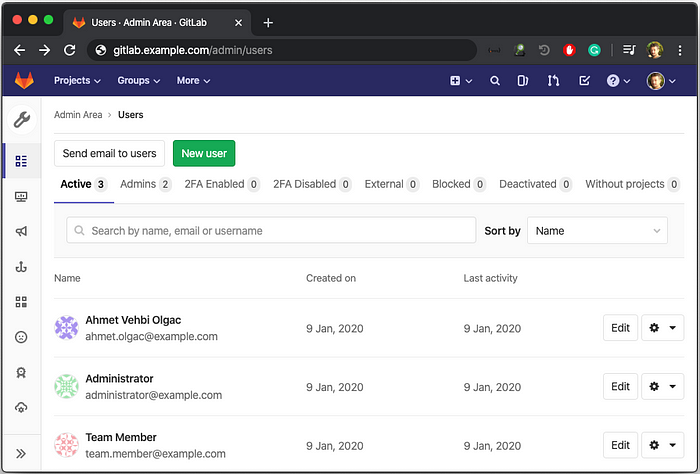
* After, Running *(*👆🏻*above ) Setup Permissions*Job; *Users*, *Groups,*[*Slack*](https://docs.gitlab.com/ee/user/project/integrations/slack.html)*and*[*JIRA*](https://docs.gitlab.com/ee/user/project/integrations/jira.html)*Integrations*should be created.
* Our Best Practice for arranging groups, per sub-team, will have a Slack Channel and a JIRA Project.
* As seen in [*permissions.csv*](https://raw.githubusercontent.com/olgac/gitlab/master/setup/permission.csv) file, each user has access levels to groups.



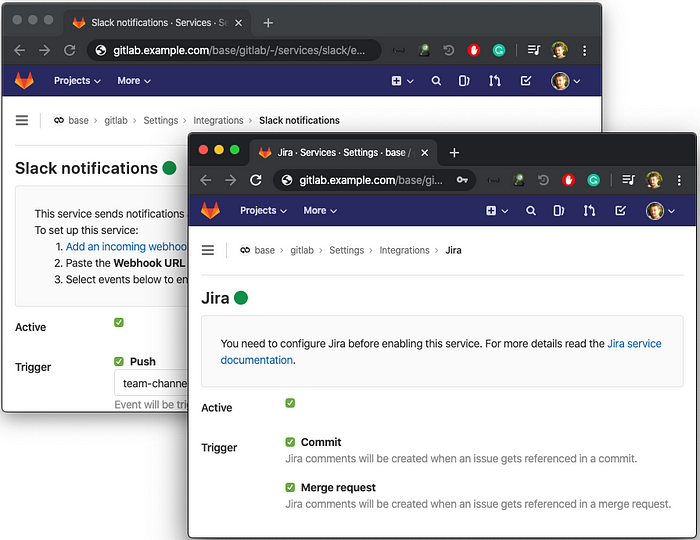
Access Levels 10=Guest 20=Reporter 30=Developer 40=Maintainer 50=Owner



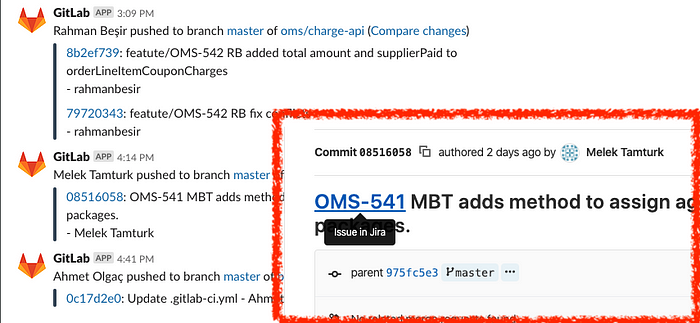
GitLab Groups



GitLab Users



Slack & JIRA Integrations

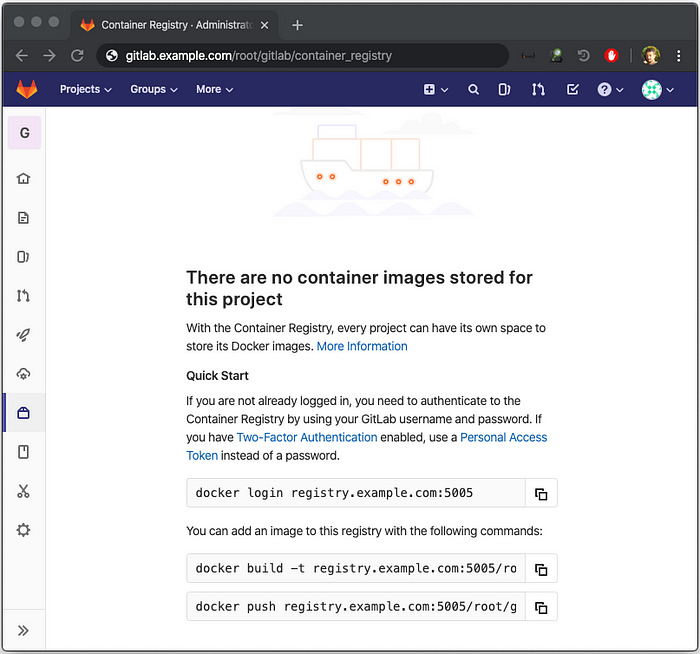


Slack Messages & JIRA Links

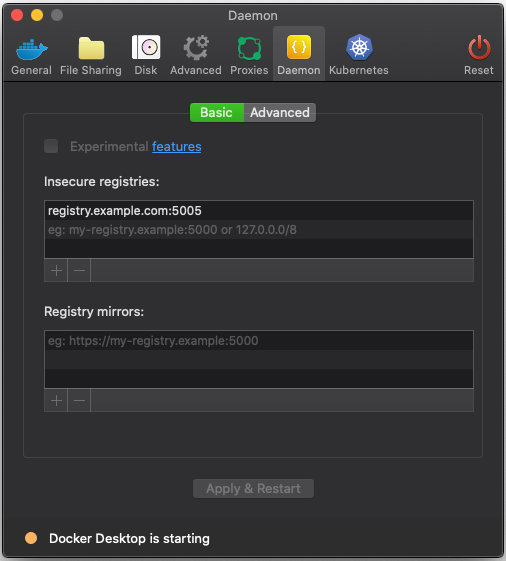
**Step 8: Login, Build & Push 🐳**

* Also, GitLab Docker Registry is available to push images with given commands.

#Login Docker Registry via GitLab Internal Registry User  
~$ docker login -u $CI\_REGISTRY\_USER -p $CI\_REGISTRY\_PASSWORD $CI\_REGISTRY#OR, Login Docker Registry via Your LDAP User  
~$ docker login registry.example.com:5005  
Username: ahmet.olgac  
Password: \*\*\*\*\*\*\*Login Succeeded#Push Builded Docker Image  
~$ docker push $DOCKER\_IMAGE



* First, you should add [registry.example.com:5005](http://registry.example.com:5005/) as an insecure registry record for Docker Daemon for clients. You can add an SSL Termination Layer in-front-of the registry *(for example: NGINX 443 to 5005)*, to work with the SSL secure registry.
* *For Mac OS, as seen screenshot below.*👇🏻 *For Ubuntu as seen script below.*



Docker Daemon Configuration

#For Ubuntu  
~$ echo "{\"insecure-registries\":[ \"registry.example.com:5005\"]}" > /etc/docker/daemon.json~$ service docker start

* Then, login with LDAP user & pass to GitLab Docker Registry with the given command below👇🏻 for push images to here.

~$ docker login registry.example.com:5005  
Username: ahmet.olgac  
Password:Login Succeeded

* Then, you can build and push 🐳 Docker images to the registry.
* In addition, GitLab provides very useful [predefined environment variables](https://docs.gitlab.com/ee/ci/variables/predefined_variables.html) that accessible at job runtime. For example, check out the below sample.

**Conclusion: Benefits with GitLab by GitOps Methodologies**

* Our cycle time is reduced, everything in one place ([Cycle Analytics](https://about.gitlab.com/product/cycle-analytics))
* Simplicity & Focus on the core domain.
* CI/CD pipelines simplified and easily maintained.
* Saving team from the complex toolchains & hard integration problems.
* Gained maintainable & manageable infrastructure.

**PS:***This article is the 1st part of the (5 articles) series, which digs Centralized Logging for K8S, Observability, APM for Cloud-Native Apps and Container Registry Topics with****GitOps****methodologies.*