GitLab-CI and Docker Registry

Oleg Fiksel

Security Consultant @ CSPI GmbH

oleg.fiksel@cspi.com | *oleg@fiksel.info* | *Matrix: @oleg:fiksel.info*

FrOSCon 2017

Agenda

ABOUT ME

- Security Consultant @ CSPI [1] (former MODCOMP [2])
- **Main topics**
  - Architecture
  - Development cycle
  - Perl Coding

[1] About CSPi
[2] Wikipedia: MODCOMP

GOALS OF THIS TALK

Goals of this talk

- This is not a comparision of CI tools

Goals of this talk

- This is not a comparision of CI tools
- Provide an overview of dependencies needed to deploy GitLab-CI Community Edition and Docker Registry **on-premise**

Goals of this talk

- This is not a comparision of CI tools
- Provide an overview of dependencies needed to deploy GitLab-CI Community Edition and Docker Registry **on-premise**
- Disclamer: The means and methods presented are my own expirience

GitLab 101

What is GitLab?

What is GitLab?

- Web-based Git repository manager and more...

What is GitLab?

- Web-based Git repository manager and more...
- Started as a pet-project in 2011 and now has more then 150 employees

What is GitLab?

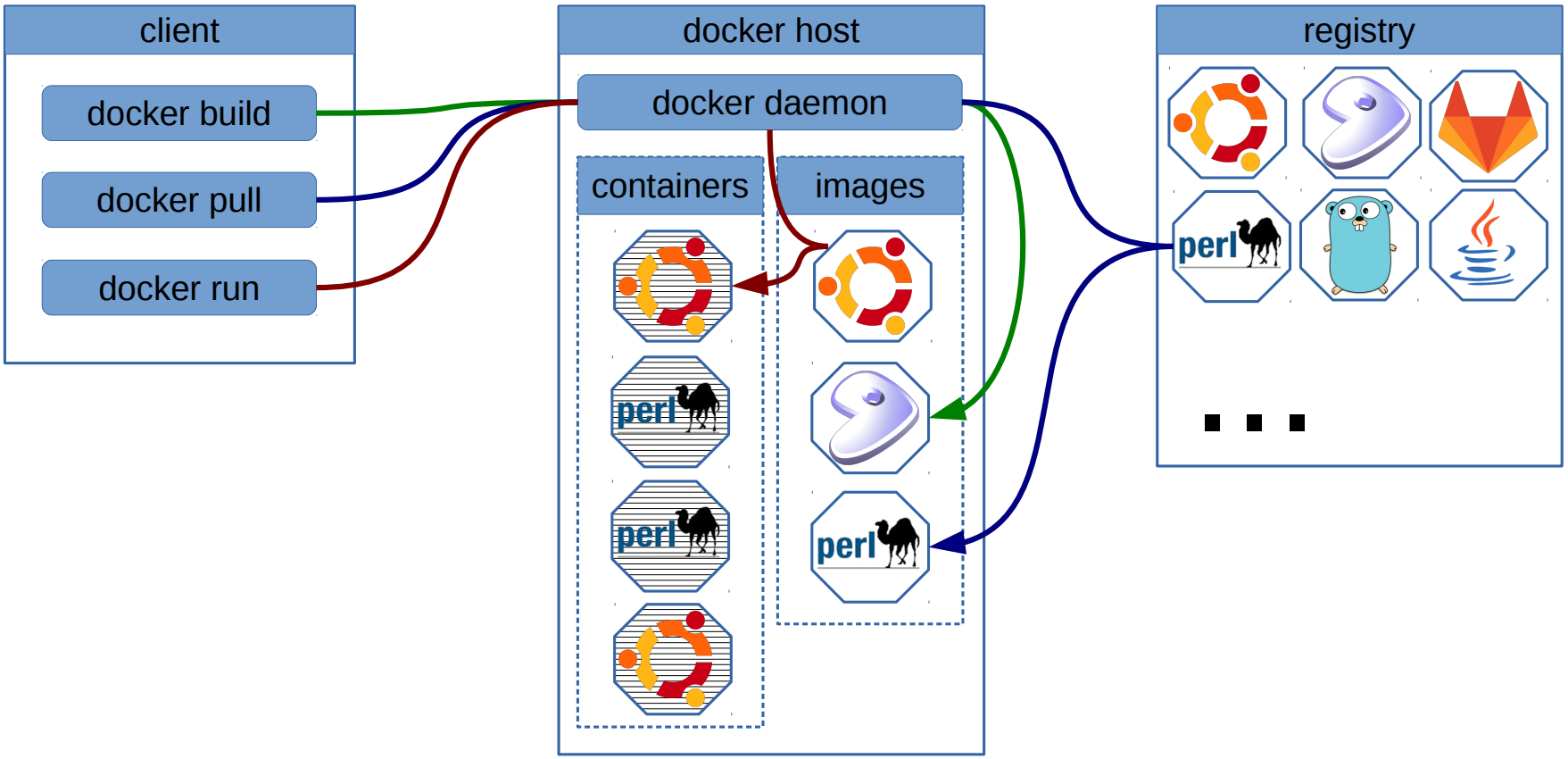
- Web-based Git repository manager and more...
- Started as a pet-project in 2011 and now has more then 150 employees
- Introduced Pipelines (CI) in version 8.8 (2016-05-28)

What is GitLab?

- Web-based Git repository manager and more...
- Started as a pet-project in 2011 and now has more then 150 employees
- Introduced Pipelines (CI) in version 8.8 (2016-05-28)
- GitLab is used by many organisations such as: IBM, Sony, NASA, Alibaba, SpaceX and CSPi

What is Docker?

What is Docker?



Wording

Wording

- **GitLab Server**: git repository hosting service

Wording

- **GitLab Server**: git repository hosting service
- **GitLab-CI Runner**: user-space daemon that executes build/tests

Wording

- **GitLab Server**: git repository hosting service
- **GitLab-CI Runner**: user-space daemon that executes build/tests
- **Artifacts**: build results pushed into an internal GitLab storage

Wording

- **GitLab Server**: git repository hosting service
- **GitLab-CI Runner**: user-space daemon that executes build/tests
- **Artifacts**: build results pushed into an internal GitLab storage
- **GitLab Container Registry**: integrated docker registry frontend

Wording

- **GitLab Server**: git repository hosting service
- **GitLab-CI Runner**: user-space daemon that executes build/tests
- **Artifacts**: build results pushed into an internal GitLab storage
- **GitLab Container Registry**: integrated docker registry frontend
- **Docker Registry**: mandatory container registry service

Deploying on-premise

Checklist

Checklist

- 2 VMs or Rancher/Kubernetes/Mesos cluster

Checklist

- 2 VMs or Rancher/Kubernetes/Mesos cluster
- Reverse proxy/loadabalancer for SSL offload (optional) supporting HTTP 1.1 to the backend (! Lighttpd)

Checklist

- 2 VMs or Rancher/Kubernetes/Mesos cluster
- Reverse proxy/loadabalancer for SSL offload (optional) supporting HTTP 1.1 to the backend (! Lighttpd)
- Direct internet connection (for pulling docker images)

Checklist

- 2 VMs or Rancher/Kubernetes/Mesos cluster
- Reverse proxy/loadabalancer for SSL offload (optional) supporting HTTP 1.1 to the backend (! Lighttpd)
- Direct internet connection (for pulling docker images)
- SSL Certificates (own CA or official)

Pitfalls

Pitfalls

- ☹ Internal CA

Pitfalls

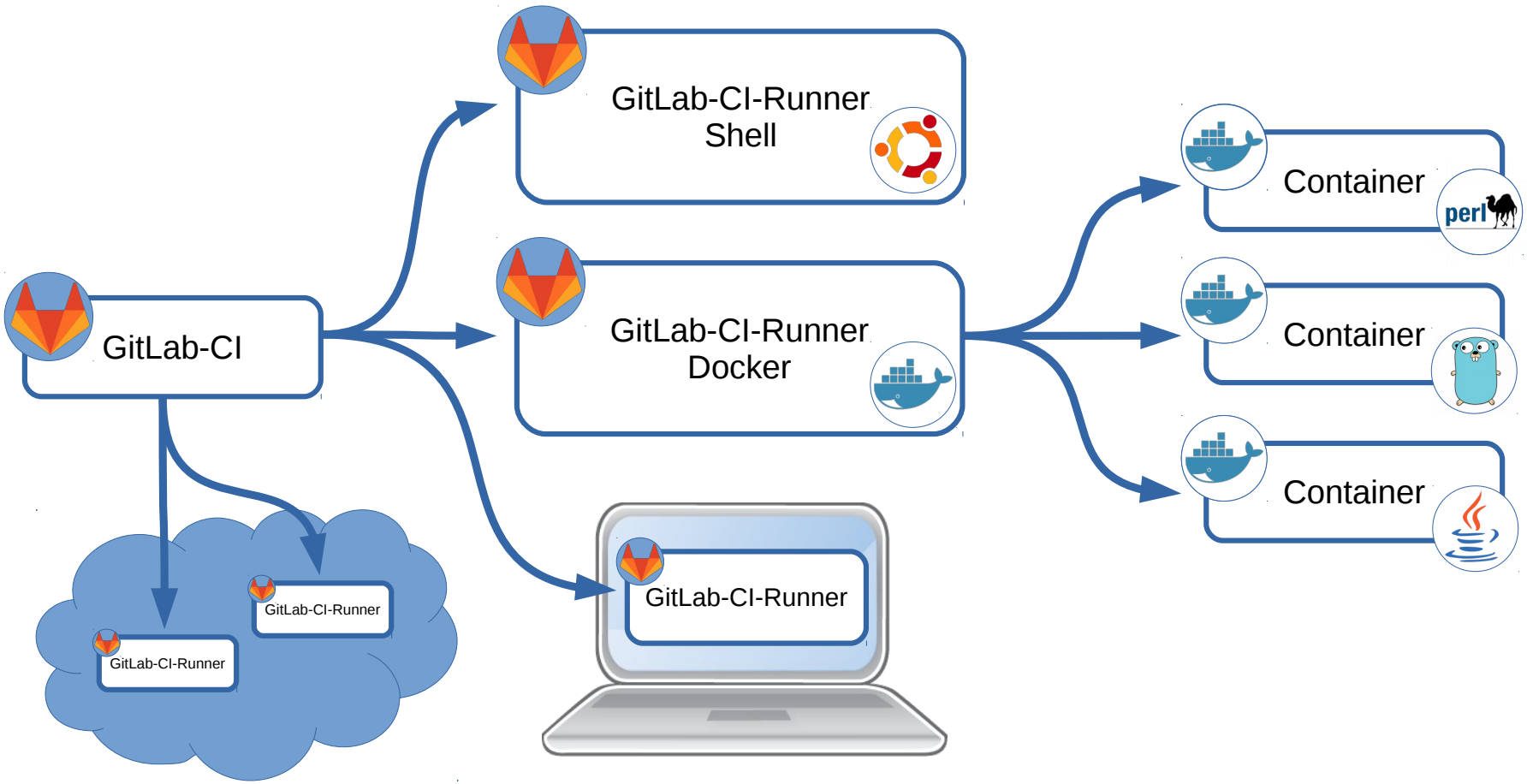
- ☹ Internal CA
- ☹ Forward proxy

Pitfalls

- ☹ Internal CA
- ☹ Forward proxy
- 😕 DNS split horizon (not handled in this talk)

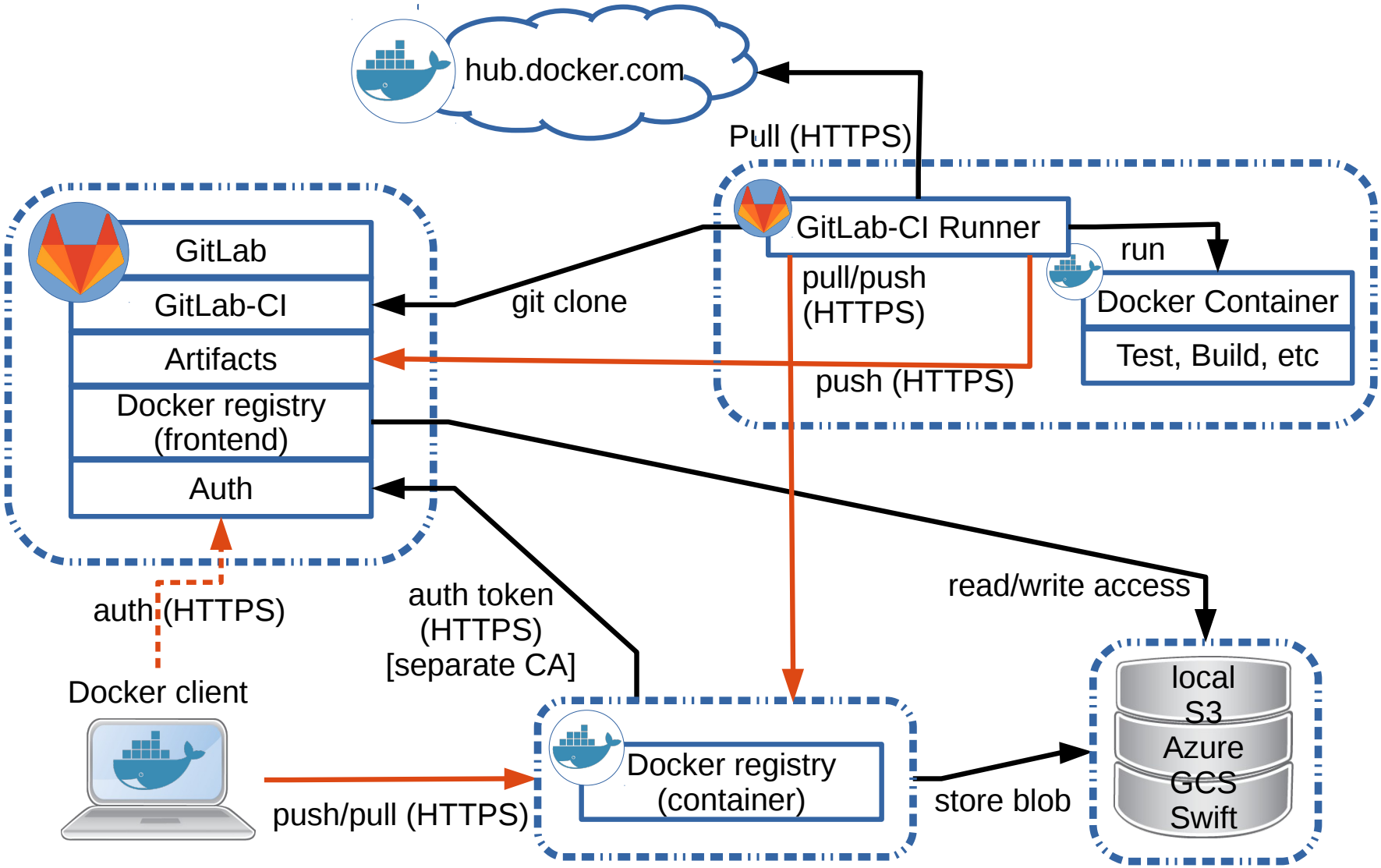
GitLab-CI runner architecture

GitLab-CI runner architecture



On-premise deployment architecture

On-premise deployment architecture



Internal CA

Internal CA

Every GitLab HTTPS client must trust internal CA including:

Internal CA

Every GitLab HTTPS client must trust internal CA including:

- gitlab-ci-runner

Internal CA

Every GitLab HTTPS client must trust internal CA including:

- gitlab-ci-runner
- docker container building docker images

Internal CA

- Problem: docker images are pulled from docker hub and doesn't trust intern CA.

Internal CA

- Problem: docker images are pulled from docker hub and doesn't trust intern CA.
- Solution: extend all base images with internal CA and use them for building.

Switch Docker storage backend

[1] Source

Switch Docker storage backend

By default, when using docker:dind, Docker uses the vfs storage driver which copies the filesystem on every run. This is a very disk-intensive operation which can be avoided if a different driver is used, for example overlay.[1]

[1] Source

Switch Docker storage backend

OS Setup:

Switch Docker storage backend

OS Setup:

- add *overlay* to */etc/modules* (Ubuntu 16.04)

Switch Docker storage backend

OS Setup:

- add *overlay* to */etc/modules* (Ubuntu 16.04)
- modprobe *overlay* or reboot the system

Switch Docker storage backend

Adjust */etc/docker/daemon.json*

```
1 {
2   "storage-driver": "overlay"
3 }
```

and restart Docker.

Warning: make sure you have no important local images or containers. You will start with an empty Docker storage.

Internal CA - bootstrap procedure

Internal CA - bootstrap procedure

- Adjust runner configuration

Internal CA - bootstrap procedure

- Adjust runner configuration
- Build docker first docker images locally and push them to the registry

Internal CA - bootstrap procedure

- Adjust runner configuration
- Build docker first docker images locally and push them to the registry
- Create CI configuration and build images automatically

Internal CA - bootstrap procedure

- Adjust runner configuration
- Build docker first docker images locally and push them to the registry
- Create CI configuration and build images automatically
- Update images daily using scheduled builds (CI feature)

Internal CA - bootstrap procedure

Adjust runner configuration:

```
1 # /etc/gitlab-runner/config.toml
2 [[runners]]
3 ...
4   executor = "docker"
5   [runners.docker]
6 ...
7     privileged = true
8     volumes = ["/cache", "/var/run/docker.sock:/var/run/docker.
          sock:rw"]
```

Internal CA - Docker image

Dockerfile for Docker image with internal CA:

Internal CA - Docker image

Dockerfile for Docker image with internal CA:

```
1 # Dockerfile
2 FROM docker:latest
3
4 COPY my_ca.crt /tmp/
5 RUN cat /tmp/my_ca.crt >>/etc/ssl/certs/ca-certificates.crt &&
      rm /tmp/my_ca.crt
6
7 ENTRYPOINT ["docker-entrypoint.sh"]
8 CMD ["sh"]
```

Internal CA - Docker image

CI configuration for Docker image with internal CA:

Internal CA - Docker image

CI configuration for Docker image with internal CA:

```
1  # .gitlab-ci.yml
2  variables:
3    DOCKER_DRIVER: overlay
4    IMAGE_TAG: $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_NAME
5
6  before_script:
7    - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN
       $CI_REGISTRY
8
9  build_docker_image:
10   stage: build
11   image: $CI_REGISTRY/gitlab-ci/docker:master
12   services:
13     - $CI_REGISTRY/gitlab-ci/dind:master
14   tags:
15     - dind
16   script:
17     - docker build -t $IMAGE_TAG .
18     - docker push $IMAGE_TAG
```

Internal CA - Docker-in-Docker image

Dockerfile for Docker-in-Docker image with internal CA:

Internal CA - Docker-in-Docker image

Dockerfile for Docker-in-Docker image with internal CA:

```
1  # Dockerfile
2  FROM docker:dind
3
4  COPY my_ca.crt /tmp/
5  RUN cat /tmp/my_ca.crt >>/etc/ssl/certs/ca-certificates.crt &&
       rm /tmp/my_ca.crt
6
7  VOLUME /var/lib/docker
8  EXPOSE 2375
9
10 ENTRYPOINT ["dockerd-entrypoint.sh"]
11 CMD []
```

Internal CA - Docker-in-Docker image

CI configuration for Docker-in-Docker image with internal CA:

Internal CA - Docker-in-Docker image

CI configuration for Docker-in-Docker image with internal CA:

```
# .gitlab-ci.yml
variables:
  DOCKER_DRIVER: overlay
  IMAGE_TAG: $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_NAME

before_script:
  - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN
      $CI_REGISTRY

build_docker_image:
  stage: build
  image: $CI_REGISTRY/gitlab-ci/docker:master
  services:
    - $CI_REGISTRY/gitlab-ci/dind:master
  tags:
    - dind
  script:
    - docker build -t $IMAGE_TAG .
    - docker push $IMAGE_TAG
```

Internal CA - building images

Now we can build Docker images with GitLab-CI!

Internal CA - building images

Now we can build Docker images with GitLab-CI!

```
1  # .gitlab-ci.yml
2  variables:
3    DOCKER_DRIVER: overlay
4    IMAGE_TAG: $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_NAME
5
6  before_script:
7    - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN
       $CI_REGISTRY
8
9  build_docker_image:
10   stage: build
11   image: $CI_REGISTRY/gitlab-ci/docker:master
12   services:
13     - $CI_REGISTRY/gitlab-ci/dind:master
14   tags:
15     - dind
16   script:
17     - docker build -t $IMAGE_TAG .
18     - docker push $IMAGE_TAG
```

Forward proxy

Forward proxy

- Not every application have proxy support

Forward proxy

- Not every application have proxy support
- Some application configuration is tricky

Forward proxy

- Not every application have proxy support
- Some application configuration is tricky
- Configuring proxy every time bloats CI configuration

Forward proxy

- Not every application have proxy support
- Some application configuration is tricky
- Configuring proxy every time bloats CI configuration
- Set proxy configuration via environmental variables while integrating your CA in the docker image

Forward proxy - local transparent proxy

For applications not supporting proxy
$\longrightarrow$ local squid in tranparent mode (doesn't work for HTTPS)

```
1  # squid configuration
2  acl docker src 172.17.0.0/16
3  acl SSL_ports port 443
4  cache_mem 16 MB
5  # upstream proxy ip
6  cache_peer 10.0.0.10 parent 8080 0 no-query proxy-only default
7  dns_v4_first on
8  http_access allow docker
9  http_access deny CONNECT !SSL_ports
10 http_access deny !Safe_ports
11 http_port 3129 intercept
12 memory_pools off
```

Forward proxy - local transparent proxy

iptables configuration:

```
1 iptables -t nat -A PREROUTING -s 172.17.0.0/16 -p tcp -m tcp --
      dport 80 -j REDIRECT --to-ports 3129
```

Known issues

GITLAB-CI WITH SUBMODULES

GITLAB-CI WITH SUBMODULES

Submodule init failing due to "SSL certificate problem".

```
fatal: unable to access 'https://github.com/minio/minio-go/':
    SSL certificate problem: unable to get local issuer
    certificate
```

GITLAB-CI WITH SUBMODULES

Submodule init failing due to "SSL certificate problem".

```
fatal: unable to access 'https://github.com/minio/minio-go/':
    SSL certificate problem: unable to get local issuer
    certificate
```

- Issue: 2148

GITLAB-CI WITH SUBMODULES

Submodule init failing due to "SSL certificate problem".

```
fatal: unable to access 'https://github.com/minio/minio-go/':
    SSL certificate problem: unable to get local issuer
    certificate
```

- Issue: 2148
- Will be fixed in gitlab-ci-multi-runner v9.4

GIT-LFS

[1] https://git-lfs.github.com

GIT-LFS

Git Large File Storage (LFS) replaces large files such as audio samples, videos, datasets, and graphics with text pointers inside Git, while storing the file contents on a remote server.[1]

[1] https://git-lfs.github.com

git-LFS

- **Problem**: GitLab-CI doesn't download git-LFS objects on CI run *(probably fixed by now)*

GIT-LFS

- **Problem**: GitLab-CI doesn't download git-LFS objects on CI run *(probably fixed by now)*
- **Workaround**: download git-LFS objects "manually" via CI script

GIT-LFS

```
1  # .gitlab-ci.yml
2  stages:
3  - build
4
5  create_package:
6    stage: build
7    image: $CI_REGISTRY/gitlab-ci/ubuntu:xenial
8    script:
9    - apt-get update && apt-get install -y wget git
10   - wget https://packagecloud.io/github/git-lfs/packages/ubuntu/
       xenial/git-lfs_1.5.2_amd64.deb/download -O /tmp/git-lfs_1
       .5.2_amd64.deb && dpkg -i /tmp/git-lfs_1.5.2_amd64.deb
11   - git lfs install && git lfs fetch && git-lfs checkout
12   - tar czf application-`cat application/version.txt`.tar.gz
       application
13   artifacts:
14     expire_in: 2 weeks
15     paths:
16     - application-*.tar.gz
17   only:
18     - /^release.*$/
```

Summary

Summary

- GitLab is a great product evolving rapidly

Summary

- GitLab is a great product evolving rapidly
- Deploying GitLab-CI in an enterprise environment can be quite challenging

Summary

- GitLab is a great product evolving rapidly
- Deploying GitLab-CI in an enterprise environment can be quite challenging
- Some of use cases and videos are focused on frontend development using Ruby-On-Rails and deployment to a Kubernetes cluster

Q & A

Thanks!

Oleg Fiksel

oleg.fiksel@cspi.com | oleg@fiksel.info | Matrix: @oleg:fiksel.info

Links

- Files from this talk on Github
- Introduction to GitLab pipelines
- Install a root CA in Ubuntu