Docker

With Great power comes great responsibility

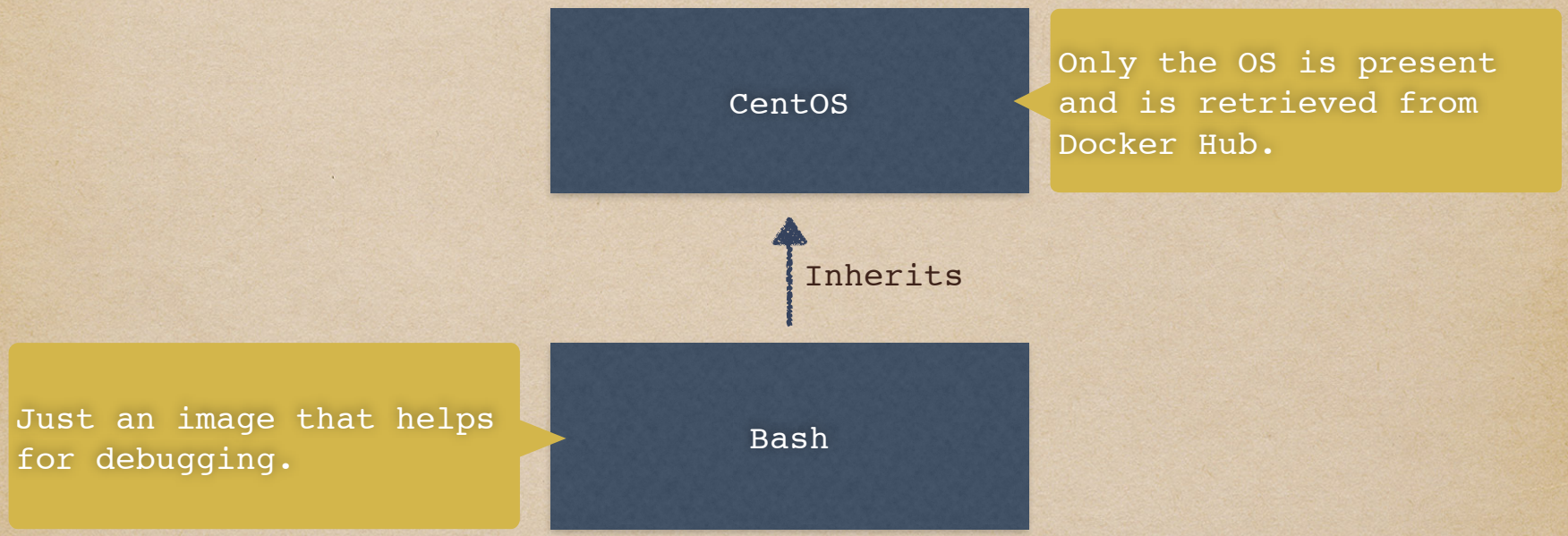
Laurent Prévost

Schedule

- Basics
- Data sharing
  - Volumes
  - Host volumes
  - Summary
- Network
  - links
- A practical complex use case
  - Stage 1 - Static
  - Stage 2 - Dynamic
  - Dig into details

Basics

Inheritance



Inheritance

CentOS

Only the OS is present and is retrieved from Docker Hub.

Inherits

Bash

Just an image that helps for debugging.

Dockerfile

```
FROM ansible/centos7-ansible:stable

ENTRYPOINT ["/bin/bash"]
```

No rocket science there. Inherit from the OS image and start a container with bash.

Inheritance

CentOS

Only the OS is present and is retrieved from Docker Hub.

Inherits

Bash

Just an image that helps for debugging.

Dockerfile

```
FROM ansible/centos7-ansible:stable

ENTRYPOINT ["/bin/bash"]
```

No rocket science there. Inherit from the OS image and start a container with bash.

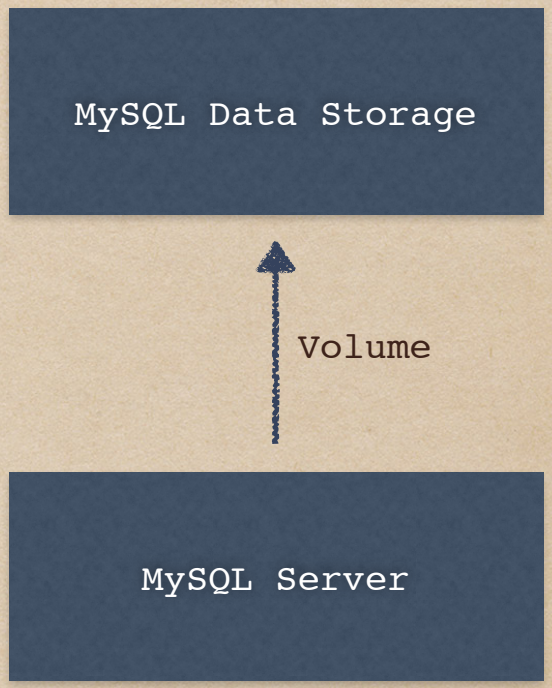
Build the bash image and run a new container from the image.

```
$> docker build bash
$> docker run -ti bash
```

Data Sharing

Volumes

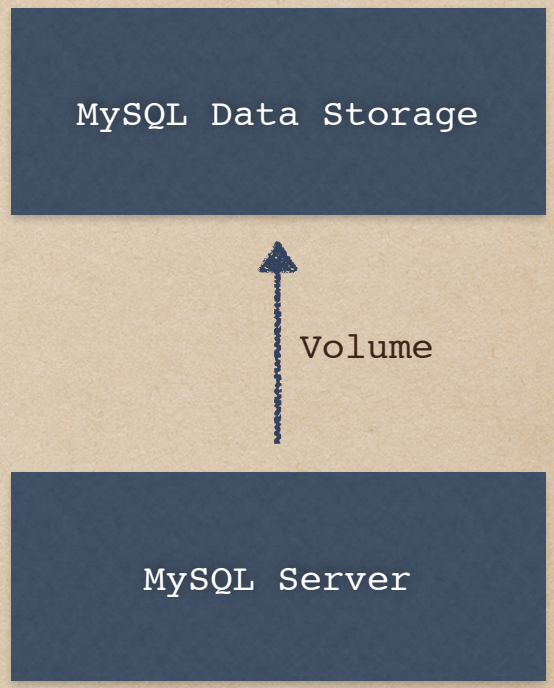
Sharing data across one or multiple containers



Volumes

```
$> docker run -v /data --name mysql-data mysql-data
```

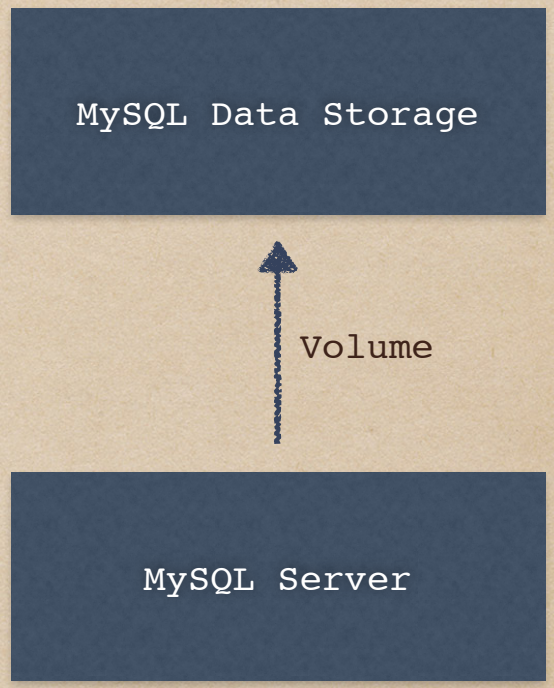
Sharing data across one or multiple containers



Volumes

```
$> docker run -v /data --name mysql-data mysql-data
```

Sharing data across one or multiple containers

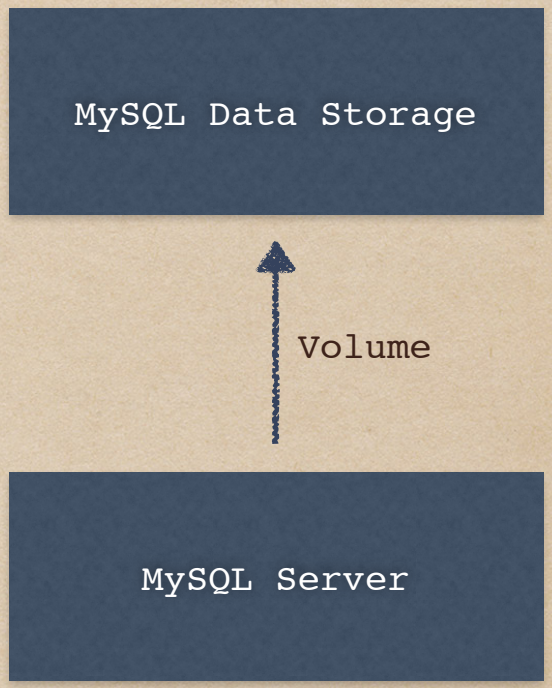



```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```

Volumes

```
$> docker run -v /data --name mysql-data mysql-data
```

Sharing data across one or multiple containers

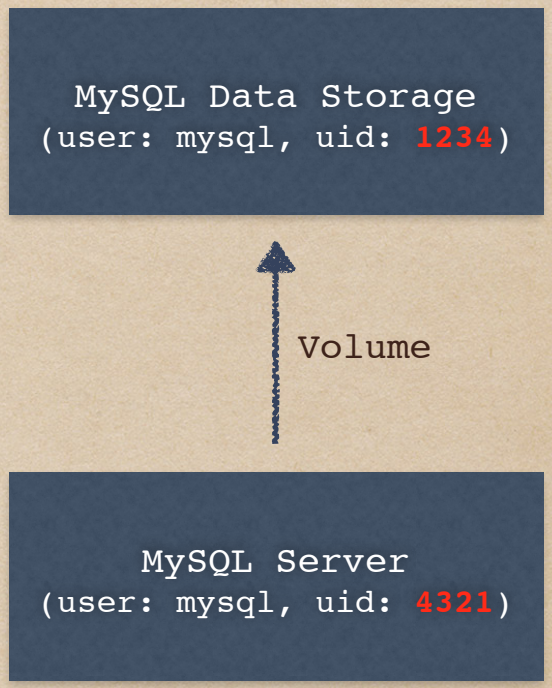



```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```

```
$> docker ps -a
CONTAINER ID   IMAGE               STATUS                  PORTS                     NAMES
b419203704cd   mysql-srv:latest    Up 3 hours              0.0.0.0:49181->3306/tcp   mysql-srv
1fcef70322a3   mysql-data:latest   Exited (0) 3 hours ago                            mysql-data
```

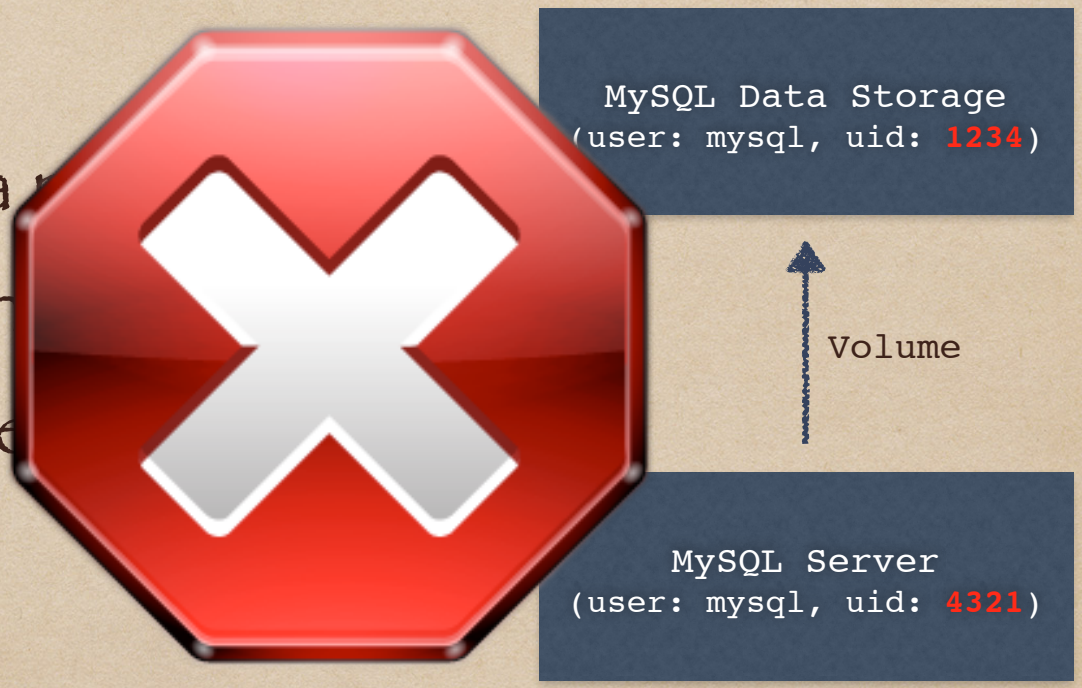
Volumes - Warning

Sharing data requires some caution. We need to handle users!



Volumes - Warning

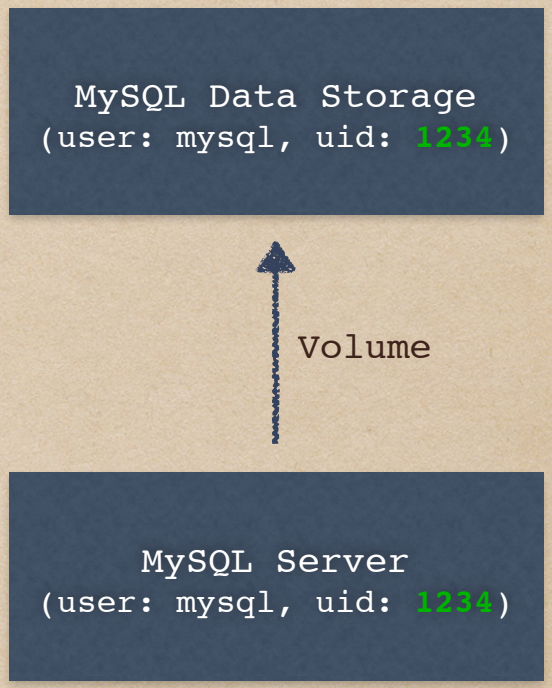
Sharing data
some cautio
to handle use



UID on both containers MUST BE the same

Volumes - Warning

Sharing data requires some caution. We need to take care about users!



Dockerfiles

MySQL Data Dockerfile

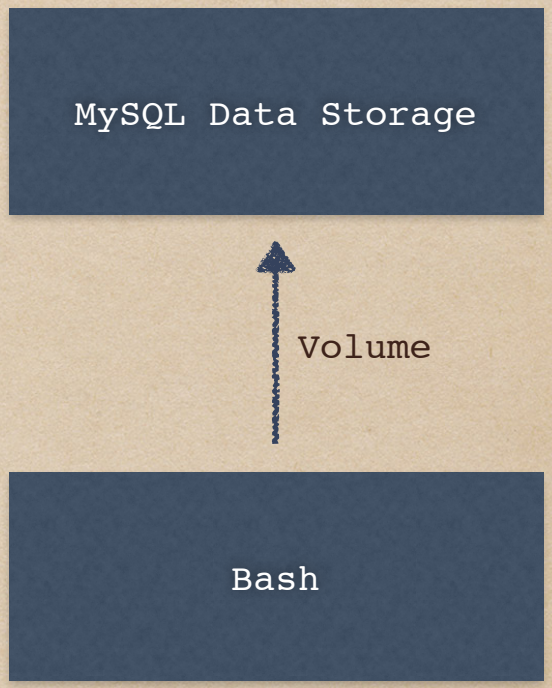
```
FROM centos7

# Create the group and user
RUN groupadd mysql
RUN useradd mysql -g mysql -u 4242

# Create the data directory and set the rights
WORKDIR /
RUN mkdir data
RUN chown mysql:mysql data

# Default command when the container is run
CMD /bin/bash
```

MySQL Server Dockerfile

```
FROM centos7

# Create the group and user
RUN groupadd mysql
RUN useradd mysql -g mysql -u 4242

# MySQL installation
...

# Entry point when the container is run
ENTRYPOINT [“mysqld”]
```

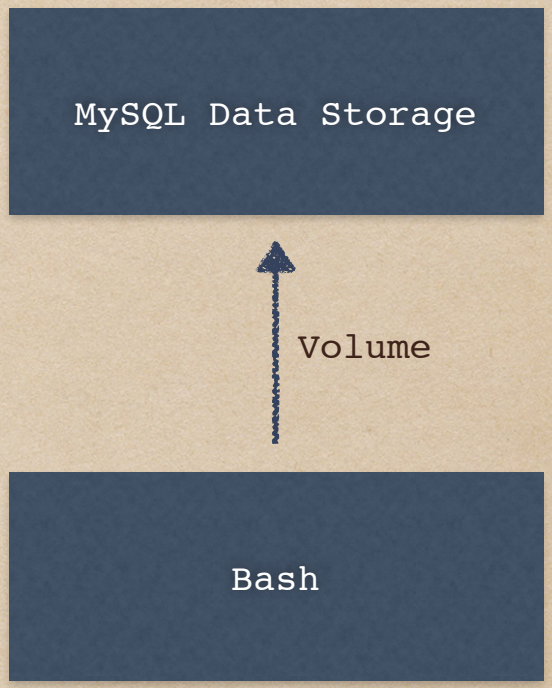
Volumes - Testing

Can we access the data container to take a look at the data? Of course, yes!



Volumes - Testing

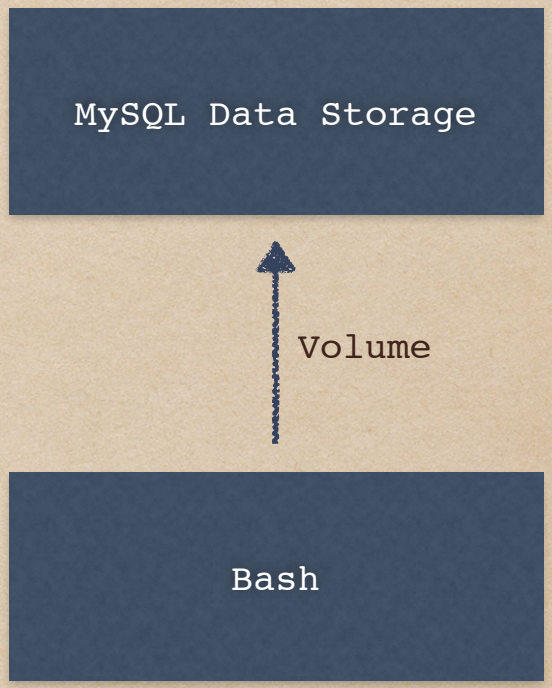
Can we access the data container to take a look at the data? Of course, yes!




```
$> docker run -ti --volumes-from mysql-data bash
```

Volumes - Testing

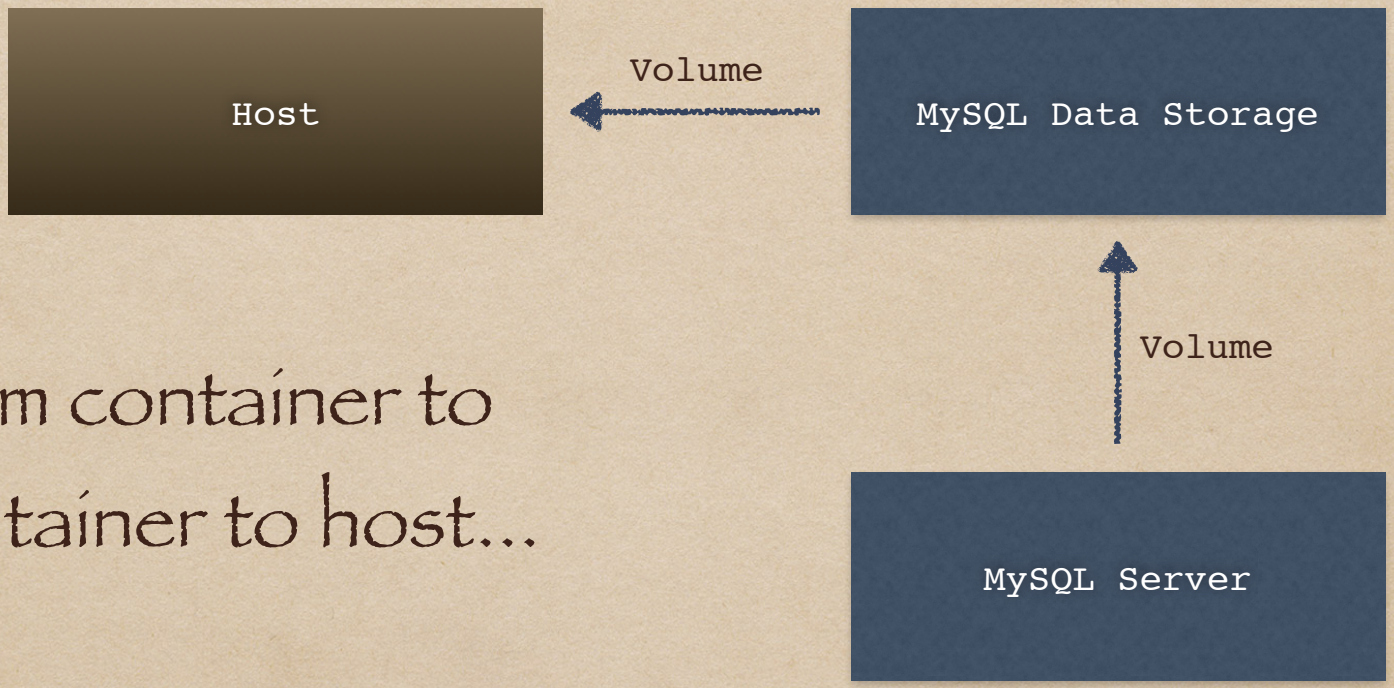
Can we access the data container to take a look at the data? Of course, yes!




```
$> docker run -ti --volumes-from mysql-data bash
```

```
$@bash> cd /data
$@bash> ls
mysql  mysql.sock  mysql.sock.lock  mysqld.log
```

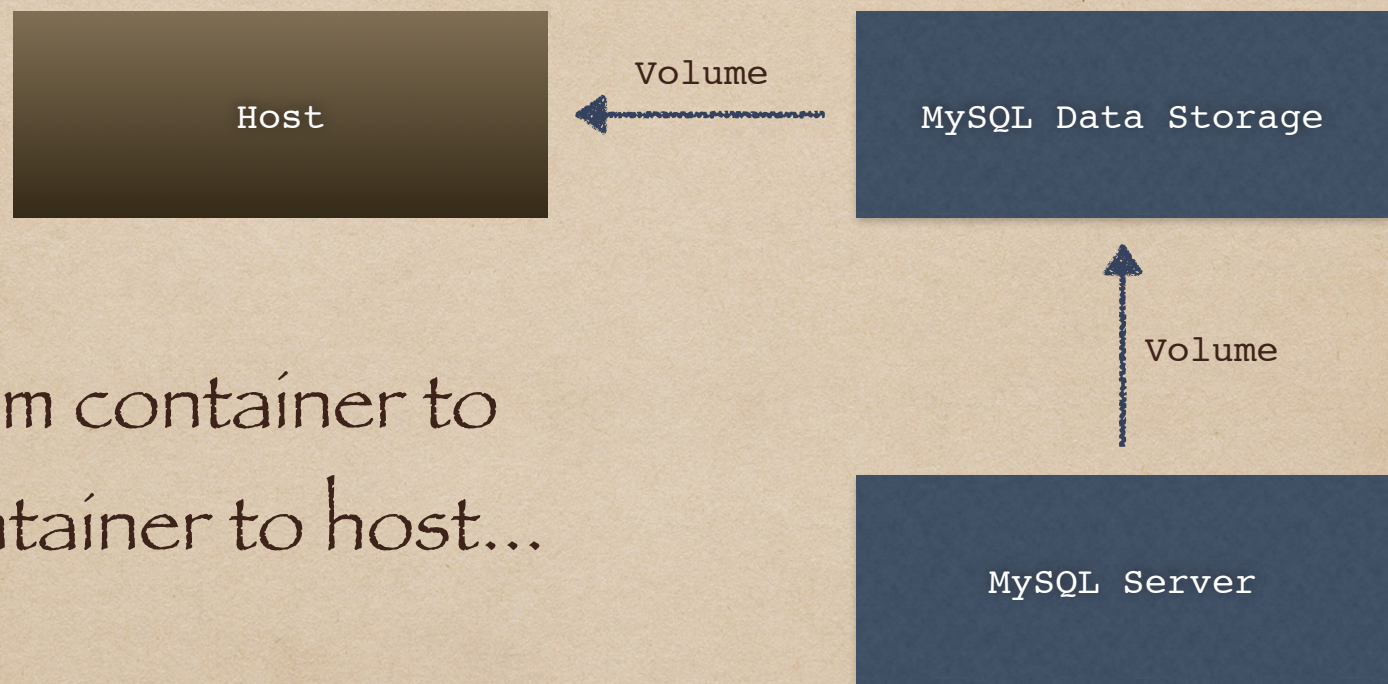
Host Volumes



From container to container to host...

Host Volumes

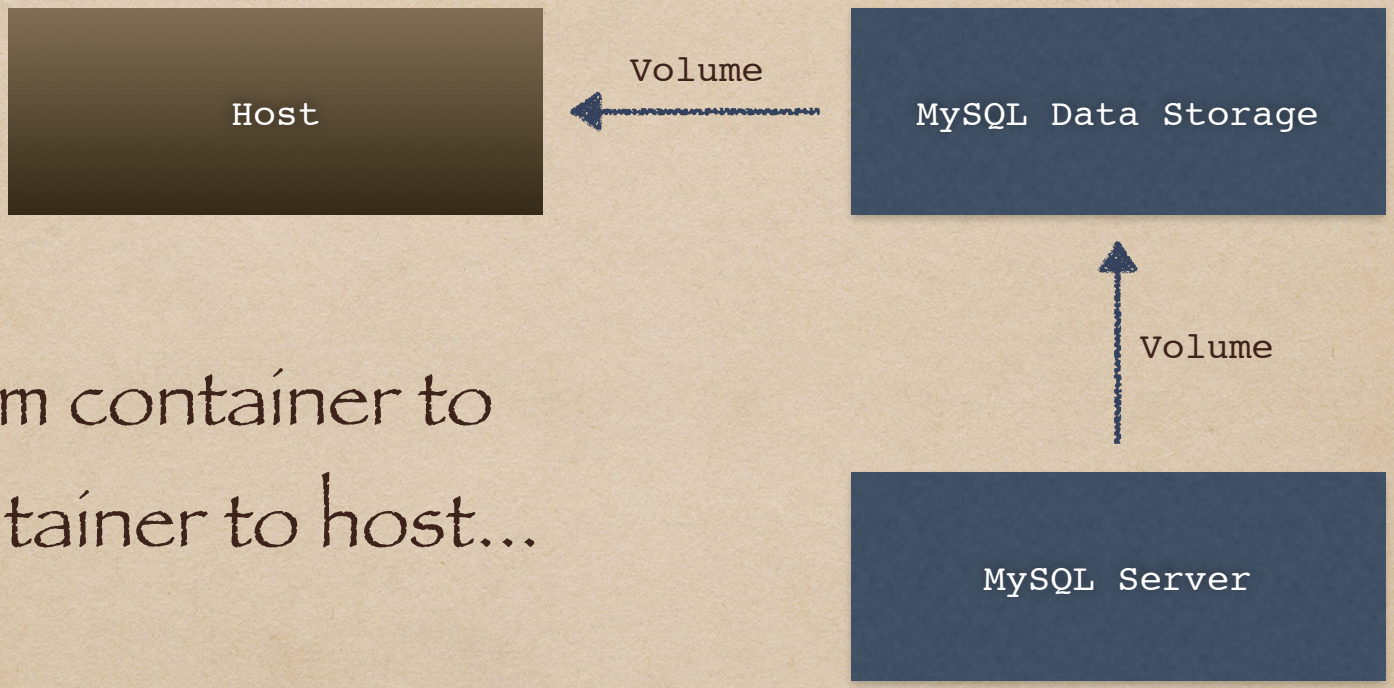
```
$> docker run -v /absolutePathFromHost:/data --name mysql-data mysql-data
```



From container to container to host...

Host Volumes

```
$> docker run -v /absolutePathFromHost:/data --name mysql-data mysql-data
```



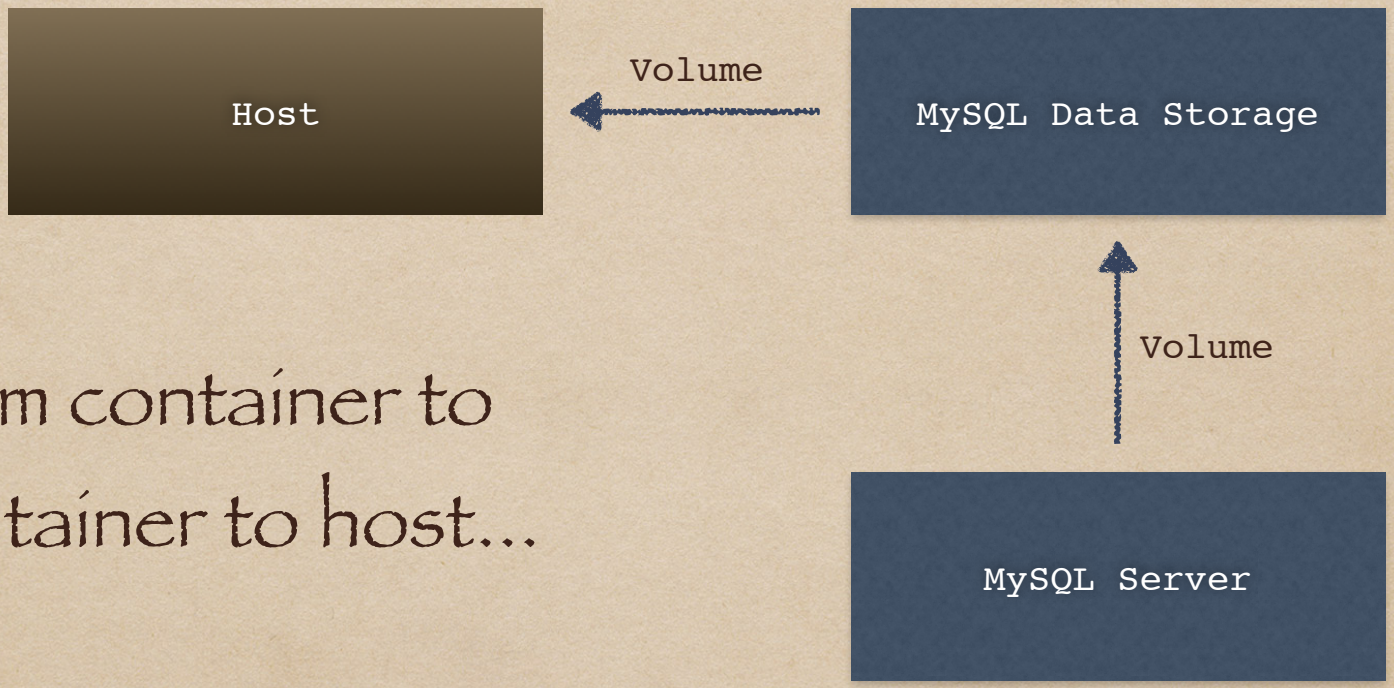
From container to container to host...

```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```

No change for this command

Host Volumes

```
$> docker run -v /absolutePathFromHost:/data --name mysql-data mysql-data
```

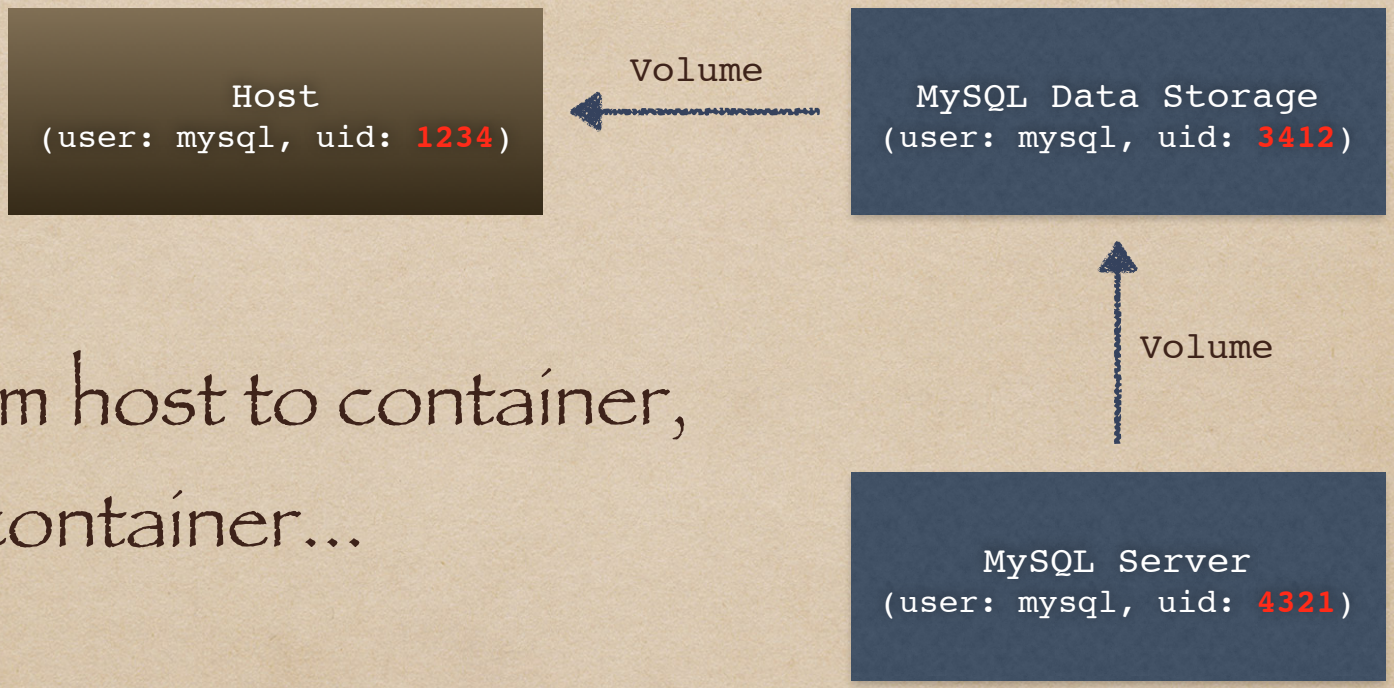


From container to container to host...

```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```

```
$> docker ps -a
CONTAINER ID   IMAGE              STATUS                    PORTS                      NAMES
b419203704cd   mysql-srv:latest   Up 3 hours                0.0.0.0:49181->3306/tcp    mysql-srv
1fcef70322a3   mysql-data:latest  Exited (0) 3 hours ago                               mysql-data
```

Host Volumes - Warning



From host to container,

to container...

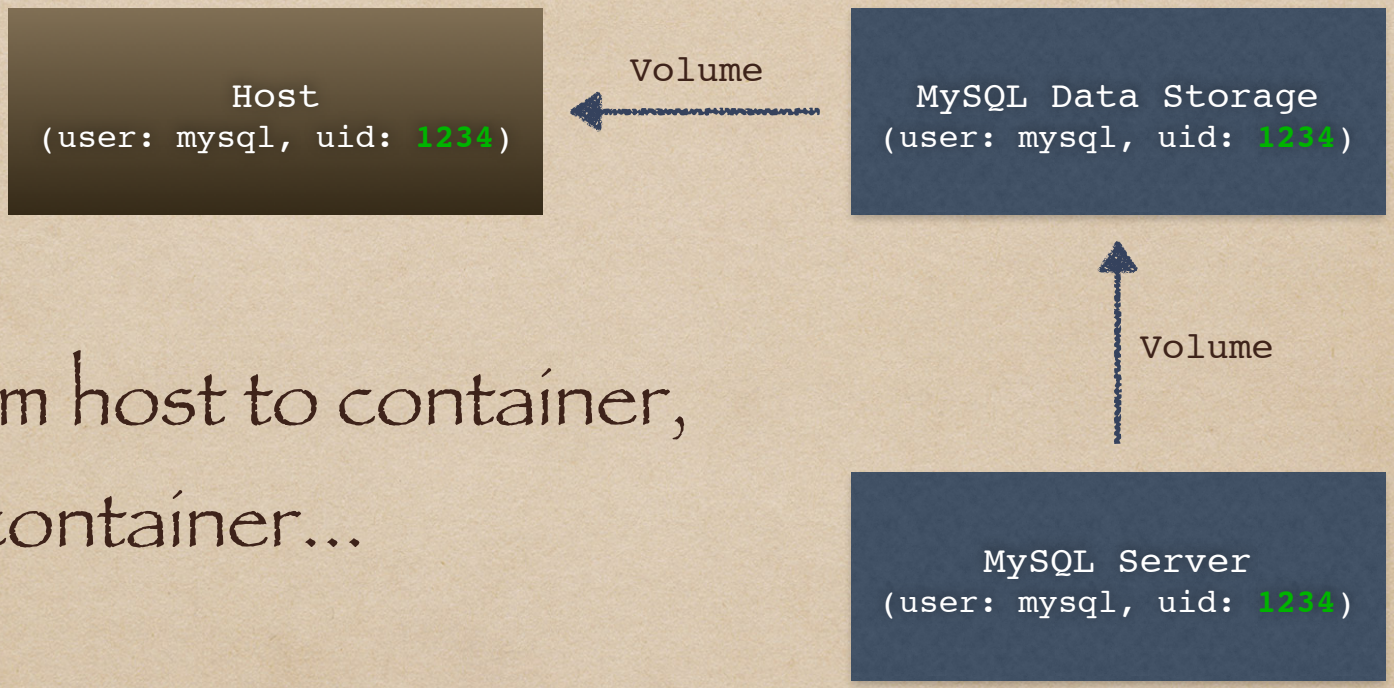
Host Volumes - Warning

From host to

to container...

UID on both containers AND host MUST BE the same

Host Volumes - Warning



From host to container,

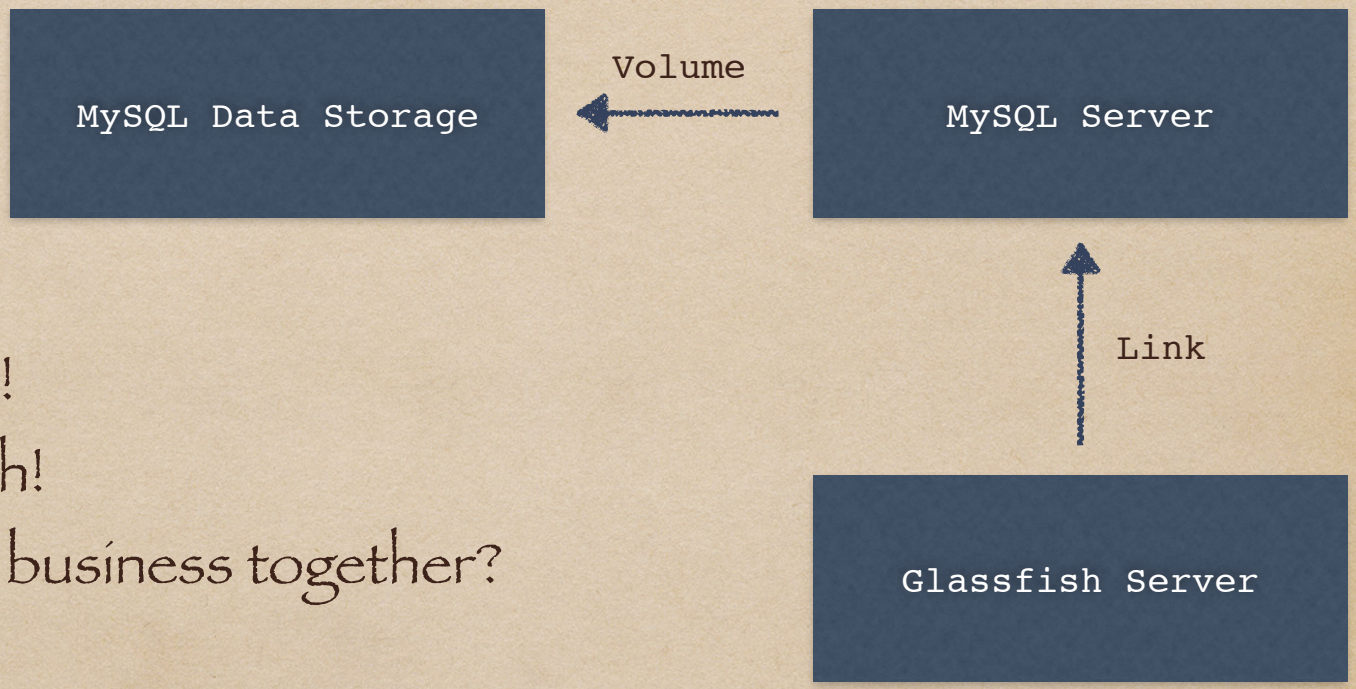
to container...

Summary

+ Creating a data container is easy.
+ Linking a data container to another container is easy too.
- It's really easy to lose data when we do not use data container (docker rm).
- Difficult to manage permissions across containers and host.

Networking

Links



G: Hi MySQL!

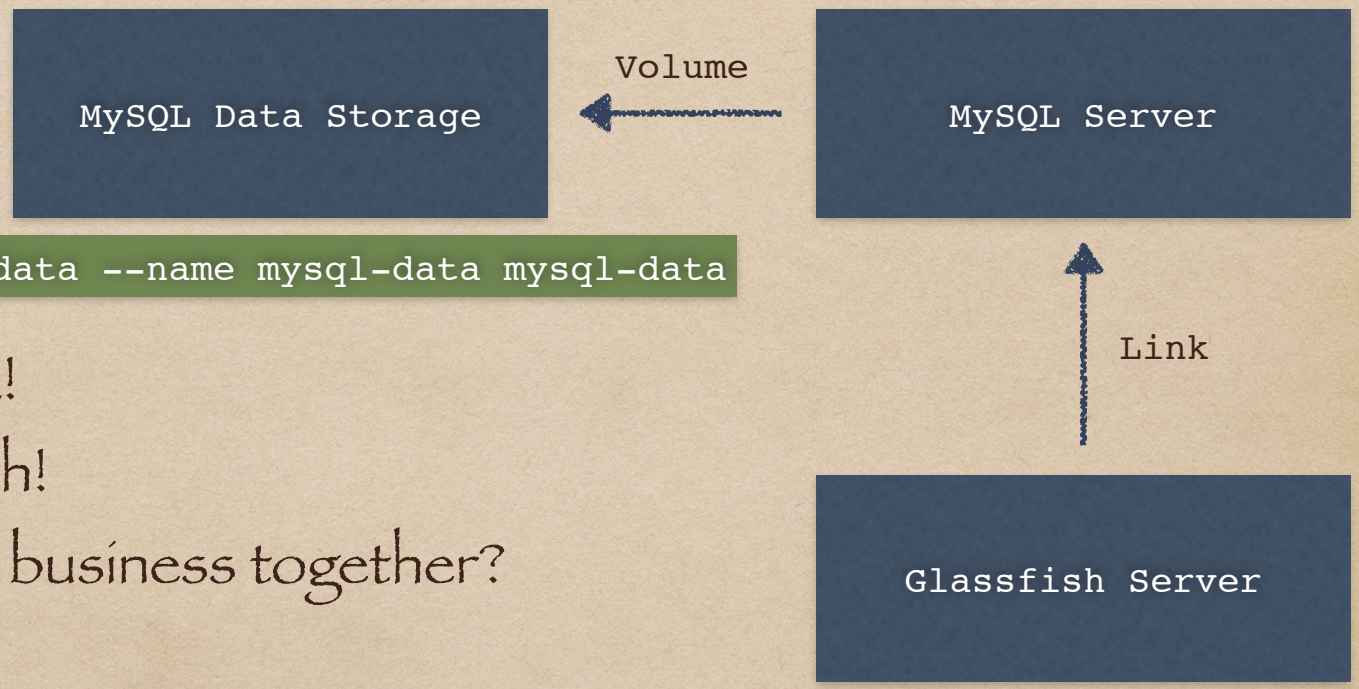
M: Hi Glassfish!

G: Can we do business together?

M: Yes!

Links

```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```




```
$> docker run -v /data --name mysql-data mysql-data
```

G: Hi MySQL!

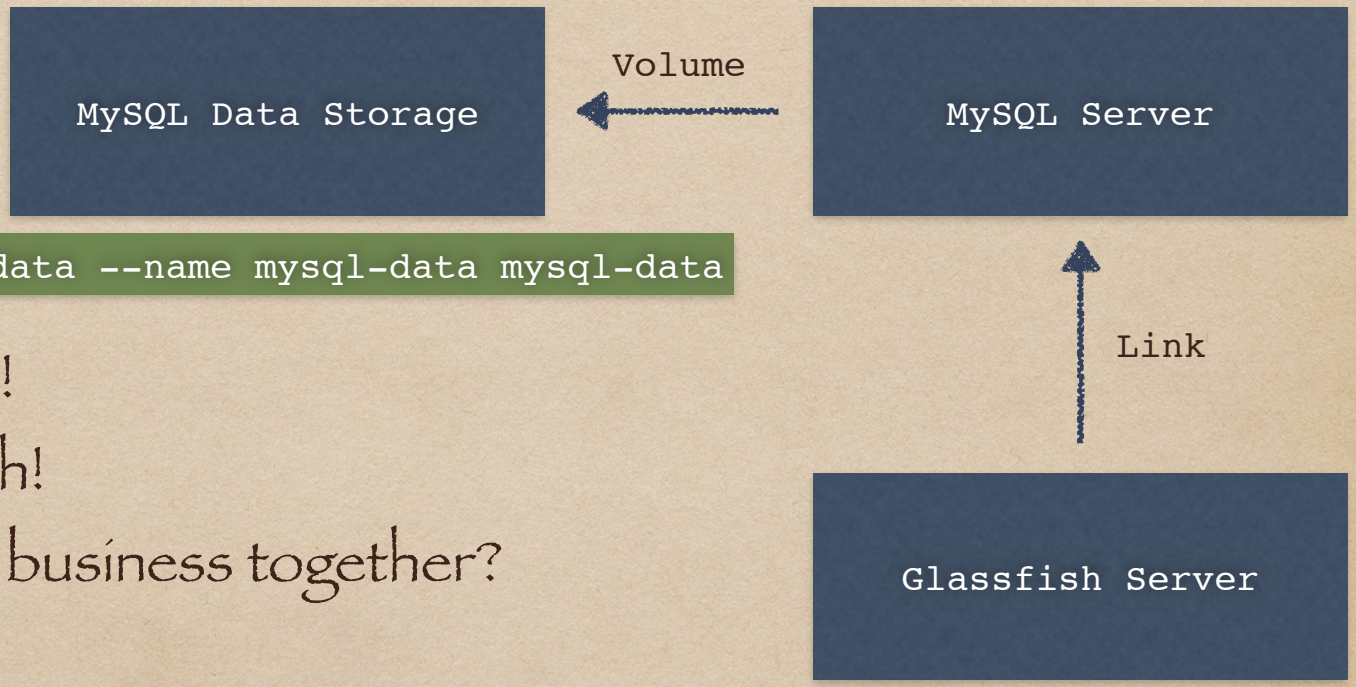
M: Hi Glassfish!

G: Can we do business together?

M: Yes!

Links

```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```




```
$> docker run -v /data --name mysql-data mysql-data
```

G: Hi MySQL!

M: Hi Glassfish!

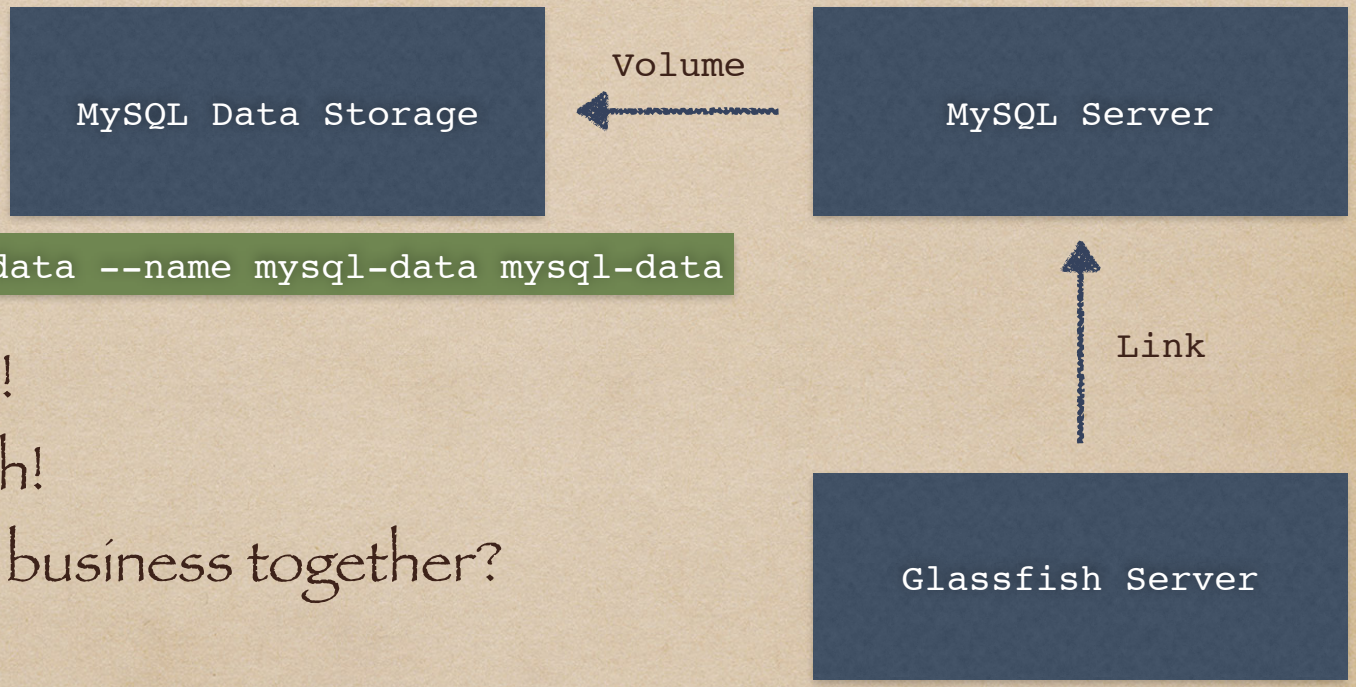
G: Can we do business together?

M: Yes!

```
$> docker run -dP --link /mysql-srv:mysql-srv --name gf-srv gf-srv
```

Links

```
$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv
```




```
$> docker run -v /data --name mysql-data mysql-data
```

G: Hi MySQL!

M: Hi Glassfish!

G: Can we do business together?

M: Yes!

```
$> docker run -dP --link /mysql-srv:mysql-srv --name gf-srv gf-srv
```

```
$> docker ps -a
CONTAINER ID   IMAGE              STATUS                    PORTS                     NAMES
7fc67e97e997   gf-srv:latest      Up 3 hours                0.0.0.0:49182->8080/tcp   gf-srv
b419203704cd   mysql-srv:latest   Up 3 hours                0.0.0.0:49181->3306/tcp   mysql-srv
1fcef70322a3   mysql-data:latest  Exited (0) 3 hours ago                              mysql-data
```

Links - Effect

- New environment variables
- /etc/hosts updated

Glassfish Server

Environment variables

```
MYSQL_SRV_NAME=mysql-srv
MYSQL_SRV_PORT=tcp://172.17.0.5:3306
MYSQL_SRV_PORT_3306_TCP=tcp://172.17.0.5:3306
MYSQL_SRV_PORT_3306_TCP_PROTO=tcp
MYSQL_SRV_PORT_3306_TCP_PORT=3306
MYSQL_SRV_PORT_3306_TCP_ADDR=172.17.0.5
```

/etc/hosts

```
172.17.0.43     58861e280cfa
127.0.0.1  localhost
::1  localhost ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
172.17.0.5  mysql-srv
```

Summary

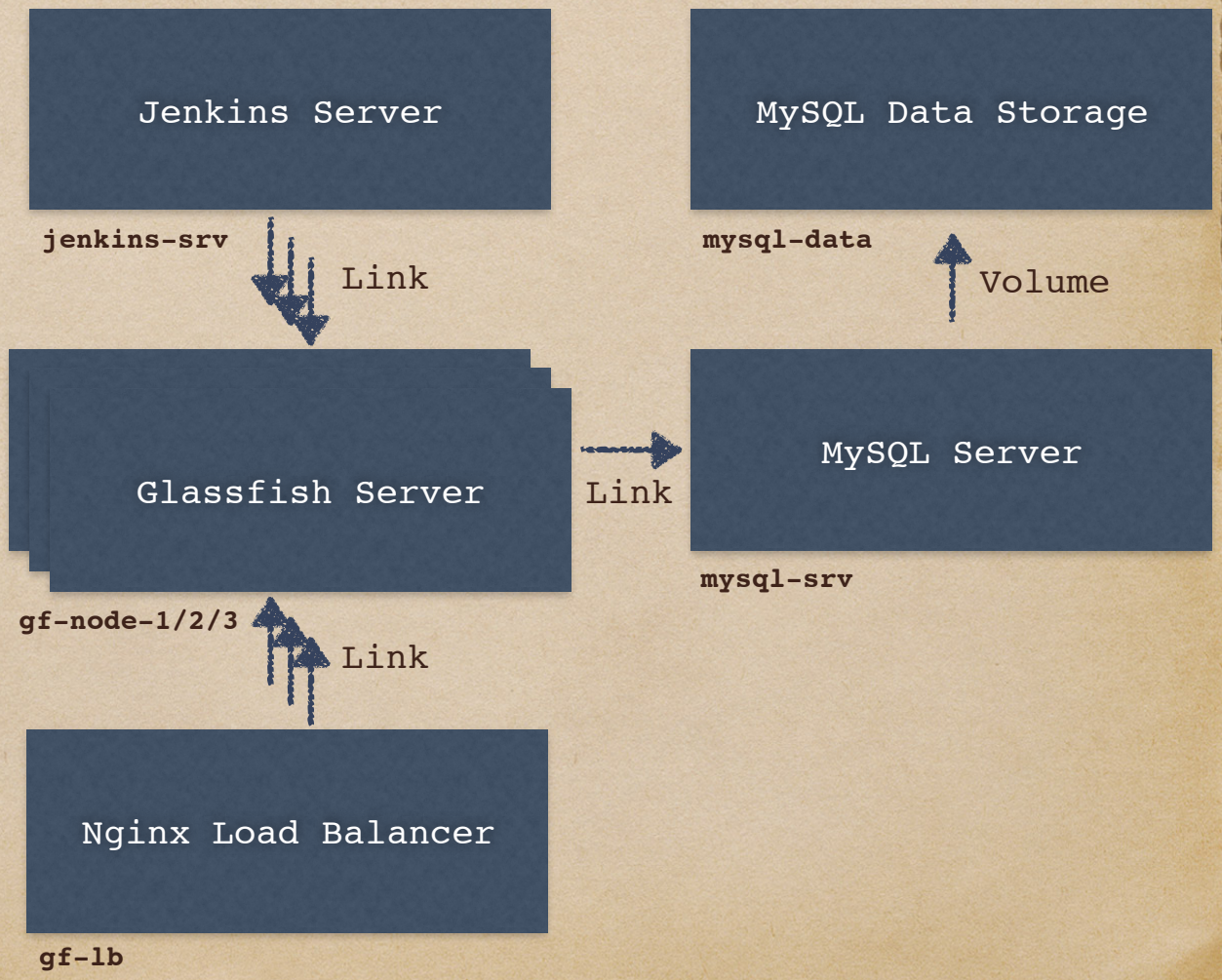
+ Linking different containers together is easy.
+ Adequate when all the containers are well defined.
- Becomes really complex to manage when the number of containers grows.
- Hardcoded at runtime. No way to dynamically add new nodes.

A practical complex use case

Stage 1

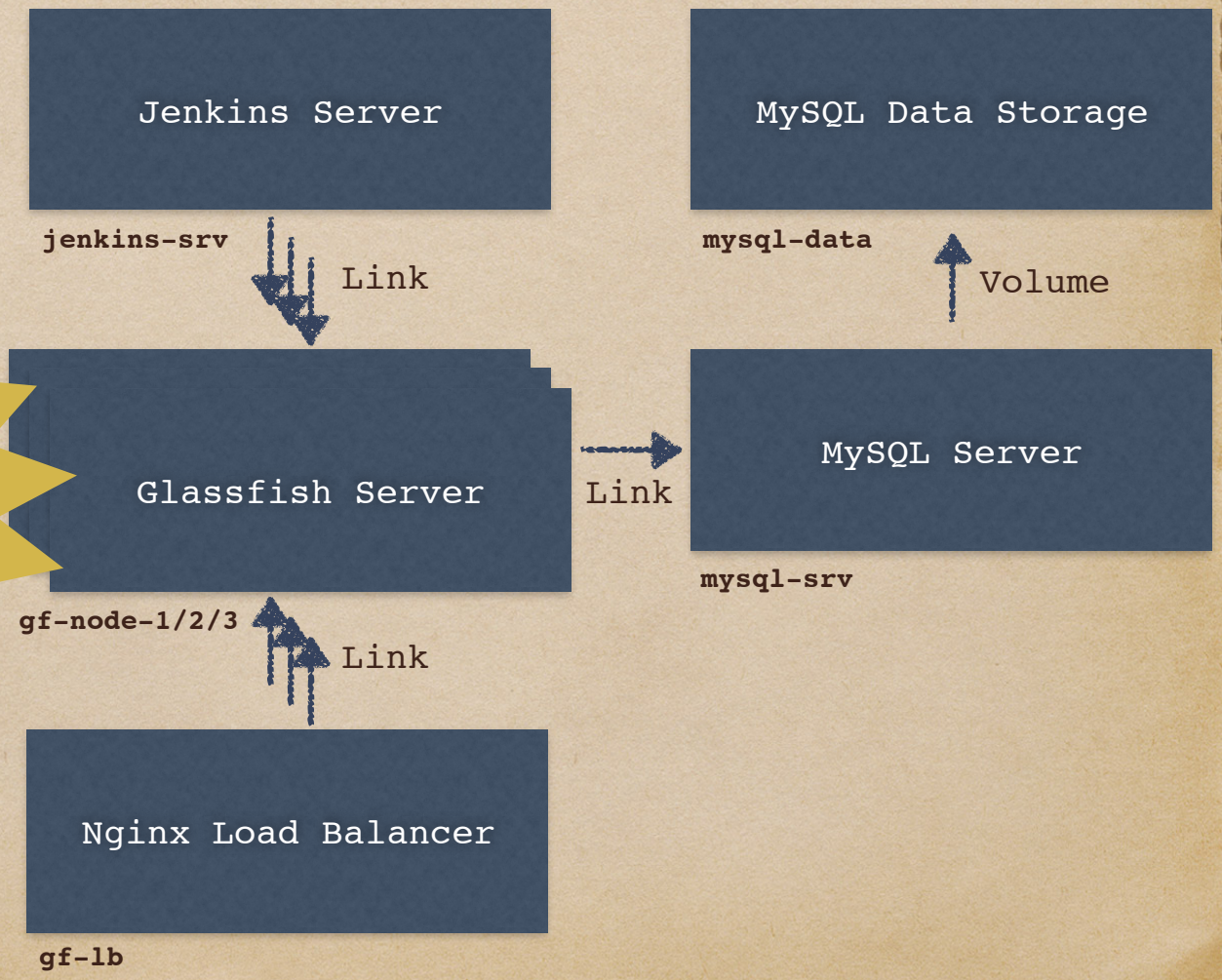
Containers statically configured

Glassfish nodes loadbalanced

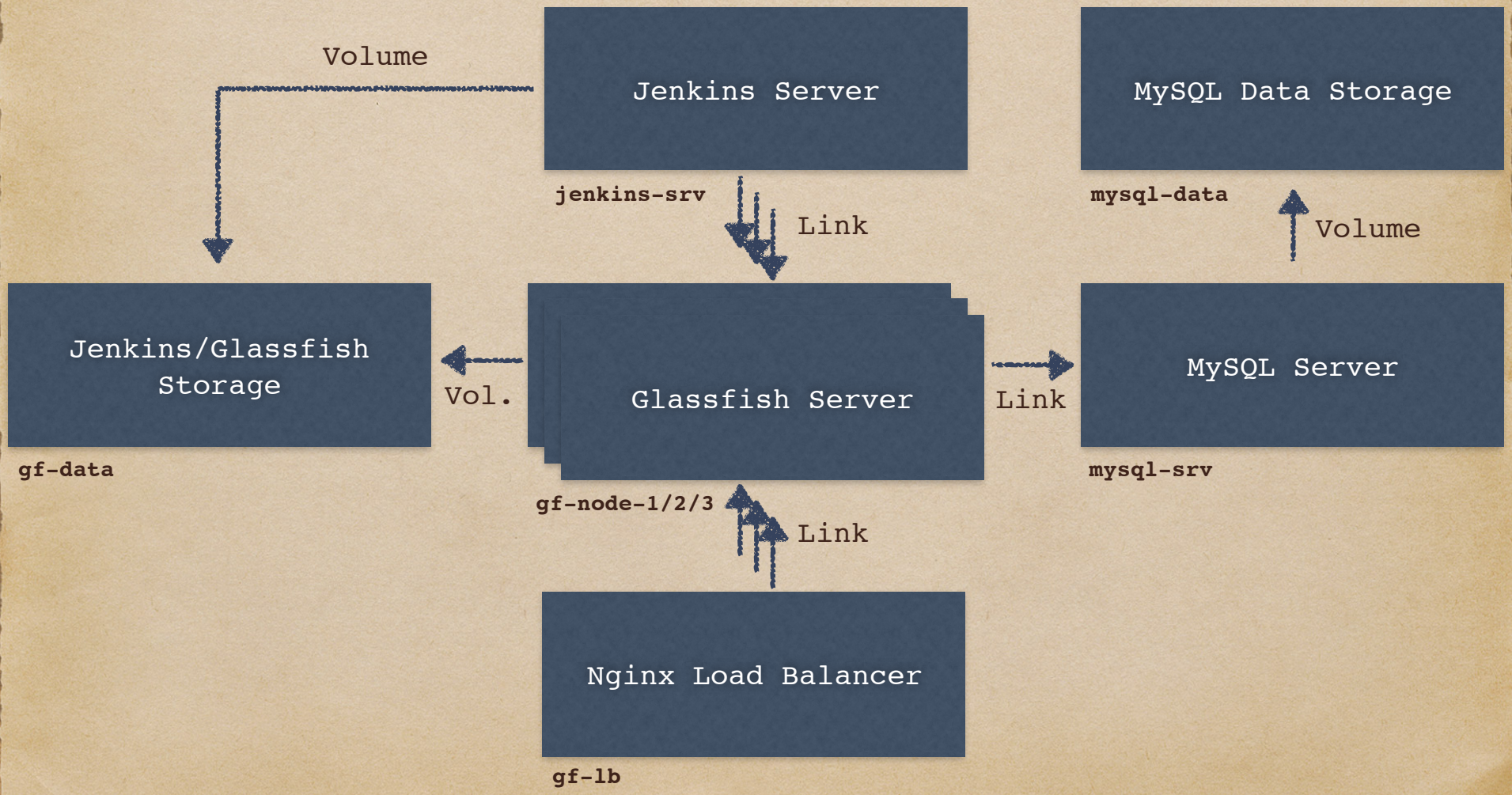


Glassfish nodes loadbalanced

Glassfish 4+ has a bug with the deployment of a file from a remote host. It's not possible to deploy a file through the network.



Glassfish nodes loadbalanced



Glassfish deployment

- Command to deploy an EAR to a remote Glassfish with the EAR present on the local file system.

```
$> asadmin --host <hostname> --port 4848 --user admin --passwordfile \
   <passwordFile> --interactive=false --echo=true --terse=false deploy \
   --name <applicationName> --force=true --isredeploy=false \
   --logreportederrors=true --type <applicationType> --upload=false \
   <pathOnLocalFileSystemToTheFileToDeploy>
```

```
$> asadmin --host gf-node-1 --port 4848 --user admin --passwordfile \
   /glassfish4/glassfish/bin/PASSWORD_FILE --interactive=false --echo=true \
   --terse=false deploy \ --name todolist-backend-ear --force=true \
   --isredeploy=false --logreportederrors=true --type ear --upload=false \
   /data/todolist-backend-ear-1.0.0-SANPSHOT.ear
```

Docker commands

```
$> docker run -v /data --name mysql-data mysql-data

$> docker run -v /data --name gf-data gf-data

$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv

$> docker run -dP --volumes-from gf-data --link /mysql-srv:mysql-srv --name gf-node-1 gf-node
$> docker run -dP --volumes-from gf-data --link /mysql-srv:mysql-srv --name gf-node-2 gf-node
$> docker run -dP --volumes-from gf-data --link /mysql-srv:mysql-srv --name gf-node-3 gf-node

$> docker run -dP --volumes-from gf-data --link /gf-node-1:gf-node-1 \
   --link /gf-node-2:gf-node-2 --link /gf-node-3:gf-node-3 --name jenkins-srv jenkins-srv

$> docker run -dP --link /gf-node-1:gf-node-1 --link /gf-node-2:gf-node-2 --link \
   /gf-node-3:gf-node-3 --name gf-lb gf-lb

$> docker ps -a
CONTAINER ID   IMAGE               STATUS                  PORTS                      NAMES
3077d3feb1fa   gf-lb:latest        Up 3 hours              0.0.0.0:49186->80/tcp      gf-lb
c6b369c6b369   jenkins-srv:latest  Up 3 hours              0.0.0.0:49185->8080/tcp    jenkins-srv
d94f346c9404   gf-node-3:latest    Up 3 hours              0.0.0.0:49184->8080/tcp    gf-node-3
ac0a87c6b369   gf-node-2:latest    Up 3 hours              0.0.0.0:49183->8080/tcp    gf-node-2
7fc67e97e997   gf-node-1:latest    Up 3 hours              0.0.0.0:49182->8080/tcp    gf-node-1
b419203704cd   mysql-srv:latest    Up 3 hours              0.0.0.0:49181->3306/tcp    mysql-srv
6b536651c078   gf-data:latest      Exited (0) 3 hours ago                             gf-data
1fcef70322a3   mysql-data:latest   Exited (0) 3 hours ago                             mysql-data
```

Docker commands

```
$> docker run -v /data --name mysql-data mysql-data

$> docker run -v /data --name gf-data gf-data

$> docker run -dP --volumes-from mysql-data --

$> docker run -d  umes-from gf-da                                  de-1 gf-node
$> docker run -  mes-from gf-d                                     de-2 gf-node
$> docker run  es-from                                             -node

$> docker ru  -from
   --link

$> docke  de
   /gf-

$> do
CONT
307
c6  rs                                                             nkins-srv
d9  rs                                                             -node-3
ac0a87c6b369   gf-node-2:latest    Up 3 hours                  0/tcp   -node-2
7fc67e97e997   gf-node-1:latest    Up 3 hours   0.0:491  080/tcp   gf-node-1
b419203704cd   mysql-srv:latest    Up 3 hours   0.0.0.0:49181->3306/tcp   mysql-srv
6b536651c078   gf-data:latest      Exited (0) 3 hours ago                 gf-data
1fcef70322a3   mysql-data:latest   Exited (0) 3 hours ago                 mysql-data
```

Each time we want to add a new Glassfish node to the load balancer, we have to do:

1. Stop the load balancer
2. Start the new Glassfish node
3. Update the load balancer configuration
4. Start the load balancer again

Nginx load balancer configuration

One Glassfish node added

Nginx config file

```
upstream application {
  server gf-node-1:8080;
  server gf-node-2:8080;
  server gf-node-3:8080;
  keepalive 64;
}

server {
  listen 80;
  server_name localhost;

  location / {
    ...

    proxy_pass http://application;
    proxy_redirect off;
  }
}
```

Nginx config file

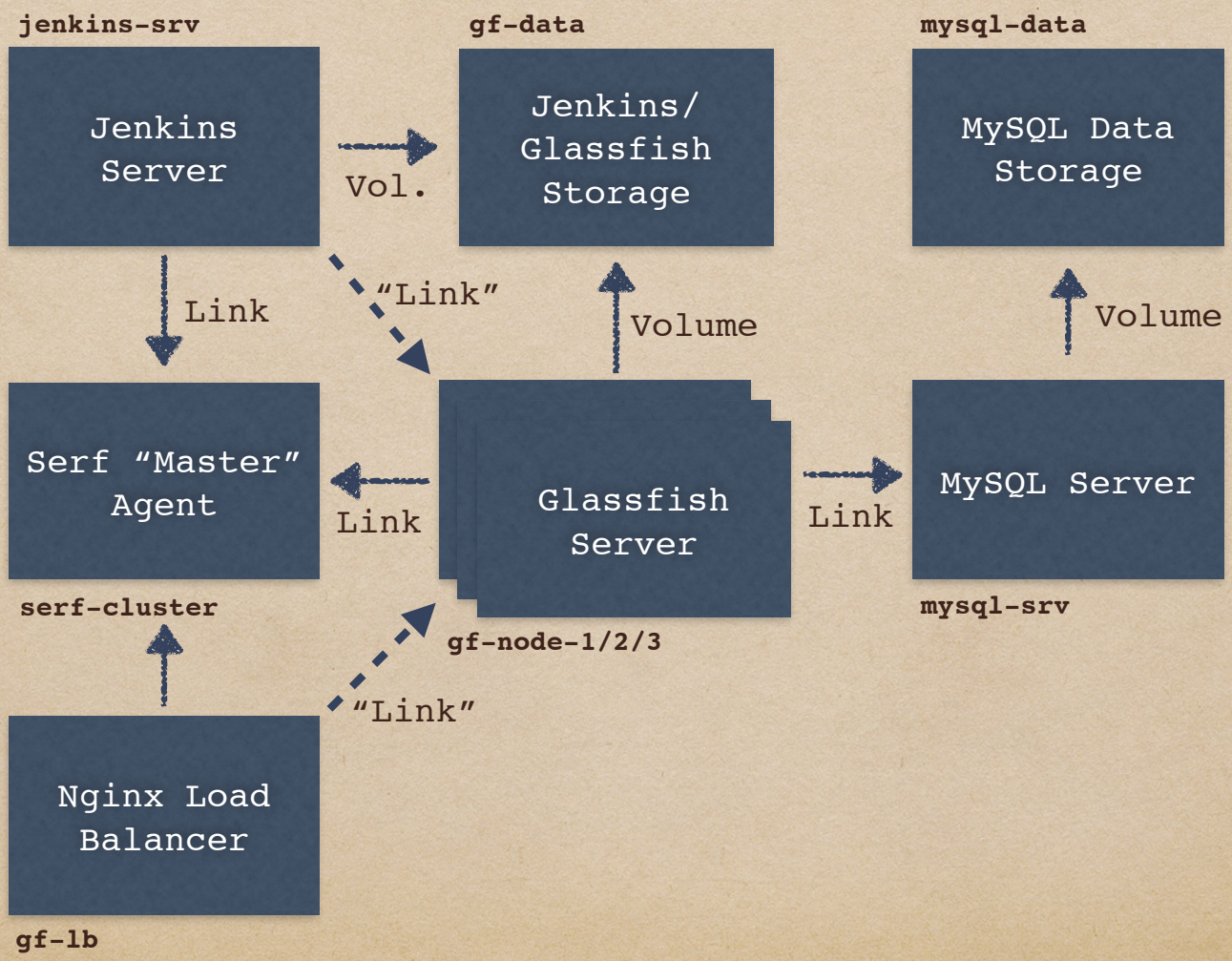
```
upstream application {
  server gf-node-1:8080;
  server gf-node-2:8080;
  server gf-node-3:8080;
  server gf-node-4:8080;
  keepalive 64;
}

server {
  listen 80;
  server_name localhost;

  location / {
    ...

    proxy_pass http://application;
    proxy_redirect off;
  }
}
```

A practical complex use case

Stage 2

Containers dynamically reconfigured

Orchestration tool



Docker commands

```
$> docker run -v /data --name mysql-data mysql-data

$> docker run -v /data --name gf-data gf-data

$> docker run -dP --volumes-from mysql-data --name mysql-srv mysql-srv

$> docker run -dP --name serf-cluster serf-cluster

$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --name jenkins-srv jenkins-srv

$> docker run -dP --link /serf-cluster:serf-cluster --name gf-lb gf-lb

$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-1 gf-node
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-2 gf-node
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node

$> docker ps -a
CONTAINER ID   IMAGE               STATUS                  PORTS                      NAMES
d94f346c9404   gf-node-3:latest    Up 3 hours              0.0.0.0:49187->8080/tcp    gf-node-3
ac0a87c6b369   gf-node-2:latest    Up 3 hours              0.0.0.0:49186->8080/tcp    gf-node-2
7fc67e97e997   gf-node-1:latest    Up 3 hours              0.0.0.0:49185->8080/tcp    gf-node-1
3077d3feb1fa   gf-lb:latest        Up 3 hours              0.0.0.0:49184->80/tcp      gf-lb
c6b369c6b369   jenkins-srv:latest  Up 3 hours              0.0.0.0:49183->8080/tcp    jenkins-srv
6b536651c078   serf-cluster        Up 3 hours              0.0.0.0:49182->7946/tcp    serf-cluster
b419203704cd   mysql-srv:latest    Up 3 hours              0.0.0.0:49181->3306/tcp    mysql-srv
6b536651c078   gf-data:latest      Exited (0) 3 hours ago                             gf-data
1fcef70322a3   mysql-data:latest   Exited (0) 3 hours ago                             mysql-data
```

Effect on Jenkins when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

member-join.sh

```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    touch /glassfishNodes/$GFNAME
    echo "$GFIP $GFNAME" >> /etc/hosts
  fi
done
```

Effect on Jenkins when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

member-join.sh

```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    touch /glassfishNodes/$GFNAME
    echo "$GFIP $GFNAME" >> /etc/hosts
  fi
done
```

```
$@jenkins> ls /glassfishNodes
gf-node-1 gf-node-2 gf-node-3
```

Effect on Jenkins when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

member-join.sh

```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    touch /glassfishNodes/$GFNAME
    echo "$GFIP $GFNAME" >> /etc/hosts
  fi
done
```

```
$@jenkins> ls /glassfishNodes
gf-node-1 gf-node-2 gf-node-3
```

/etc/hosts

```
172.17.0.43    58861e280cfa
127.0.0.1 localhost
::1  localhost ip6-localhost ip6-loopback
fe00::0   ip6-localnet
ff00::0   ip6-mcastprefix
ff02::1   ip6-allnodes
ff02::2   ip6-allrouters
172.17.0.3  gf-node-1
172.17.0.4  gf-node-2
172.17.0.5  gf-node-3
```

Jenkins Job Glassfish deployment step

```
GFNODES=`ls /glassfishNodes`
for gfnode in $GFNODES; do
  asadmin --host $gfnode --port 4848 --user admin --passwordfile \
  /glassfish4/glassfish/bin/PASSWORD_FILE --interactive=false --echo=true \
  --terse=false deploy \ --name todolist-backend-ear --force=true \
  --isredeploy=false --logreportederrors=true --type ear --upload=false \
  /data/todolist-backend-ear-1.0.0-SANPSHOT.ear
done
```

Jenkins Job Glassfish deployment step

```
GFNODES=`ls /glassfishNodes`
for gfnode in $GFNODES; do
  asadmin --host $gfnode --port 4848 --user admin --passwordfile \
  /glassfish4/glassfish/bin/PASSWORD_FILE --interactive=false --echo=true \
  --terse=false deploy \ --name todolist-backend-ear --force=true \
  --isredeploy=false --logreportederrors=true --type ear --upload=false \
  /data/todolist-backend-ear-1.0.0-SANPSHOT.ear
done
```

The job is now dynamically reconfigured as there is a loop to get the Glassfish node name. As the /etc/hosts is up to date, the node name is resolved to its corresponding address IP.

Effect on Nginx Load Balancer when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

Dockerfile

```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    SERVER_STRING="server ${GFIP}:8080;"

    sed -i "/upstream/a\  $SERVER_STRING" \
      /etc/nginx/sites-enabled/default
    echo $SERVER_STRING >> /nginx/servers

   nginx -s reload
  fi
done
```

Effect on Nginx Load Balancer when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

Dockerfile

```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    SERVER_STRING="server ${GFIP}:8080;"

    sed -i "/upstream/a\  $SERVER_STRING" \
      /etc/nginx/sites-enabled/default
    echo $SERVER_STRING >> /nginx/servers

   nginx -s reload
  fi
done
```

Nginx config file

```
upstream application {
  server gf-node-3:8080;
  server gf-node-2:8080;
  server gf-node-1:8080;
 keepalive 64;
}

...
```

Effect on Nginx Load Balancer when a Glassfish node is added

```
$> docker run -dP --volumes-from gf-data --link /serf-cluster:serf-cluster \
   --link /mysql-srv:mysql-srv --name gf-node-3 gf-node
```

Dockerfile

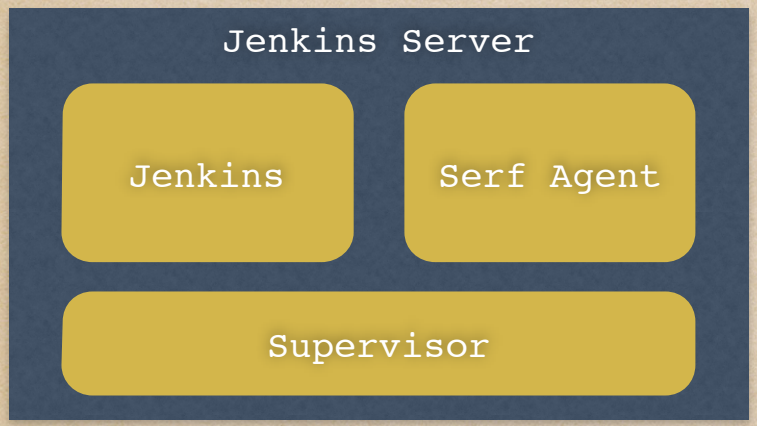
```
#!/bin/bash

while read -a values; do
  GFNAME=${values[0]}

  if [[ $GFNAME == gf-node-* ]]; then
    GFIP=${values[1]}

    SERVER_STRING="server ${GFIP}:8080;"

    sed -i "/upstream/a\  $SERVER_STRING" \
      /etc/nginx/sites-enabled/default
    echo $SERVER_STRING >> /nginx/servers

   nginx -s reload
  fi
done
```

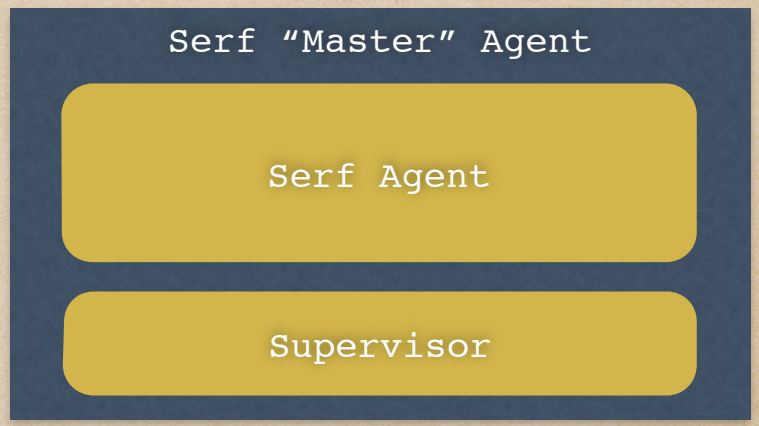
Nginx config file

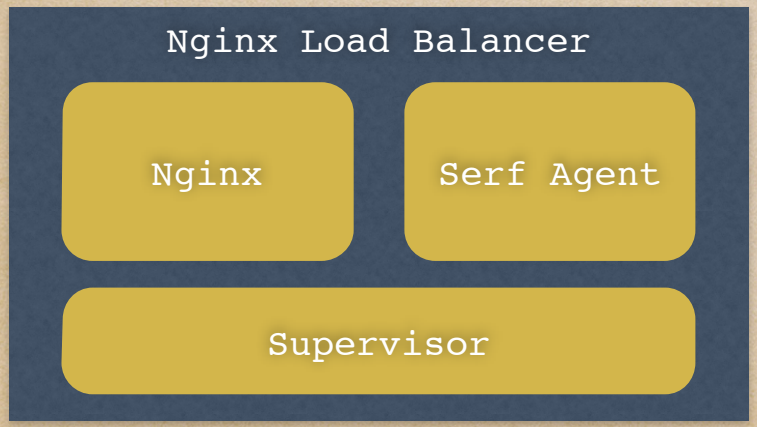
```
upstream application {
  server gf-node-3:8080;
  server gf-node-2:8080;
  server gf-node-1:8080;
 keepalive 64;
}

...
```

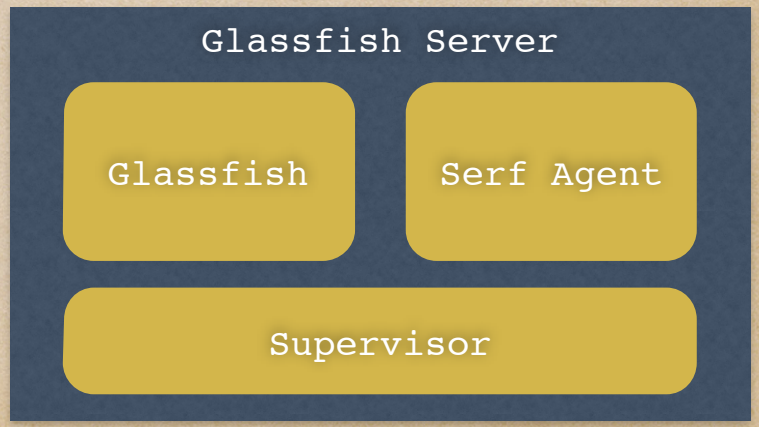
```
$@gf-lb> cat /nginx/servers
gf-node-1
gf-node-2
gf-node-3
```

Dig into details

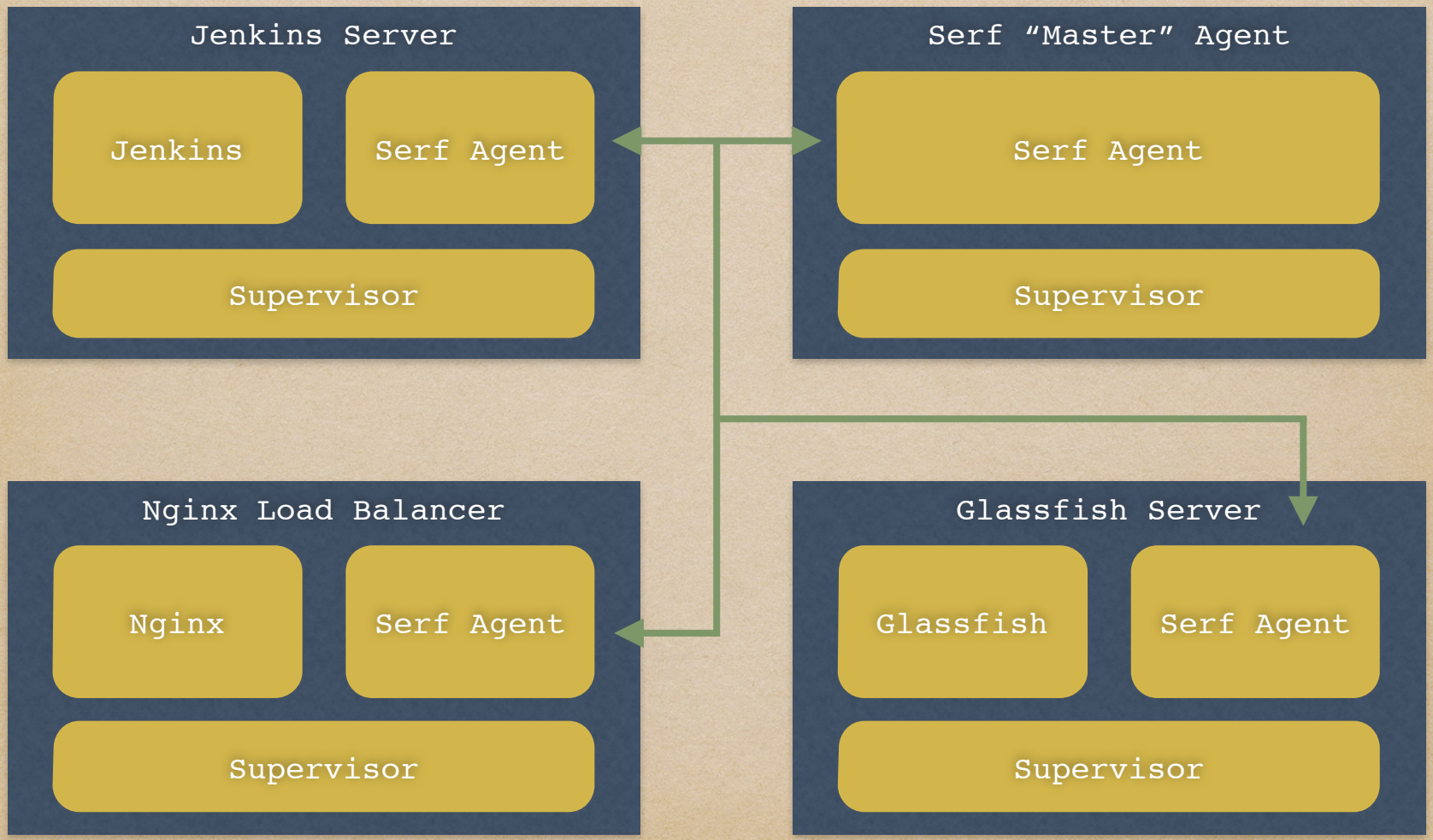








Dig into details



Docker Setup

Dockerfile

```
...

EXPOSE 8080 4848

VOLUME /supervisor

ENTRYPOINT ["/usr/bin/supervisord", "-n"]
```

start-serf.sh

```
#!/bin/bash

exec serf agent -tag role=gf-node \
-node gf-node-1 -join serf-cluster:7946
```

Supervisor

```
[program:gf-node]
command=.../glassfish/bin/asadmin start-domain domain1
user=glassfish
numprocs=1
autostart=true
autorestart=true
stdout_logfile=/supervisor/%(program_name)s.log
stderr_logfile=/supervisor/%(program_name)s.log

[program:serf-gf-node-agent]
command=/serfScripts/start-serf.sh
numprocs=1
autostart=true
autorestart=true
stdout_logfile=/supervisor/%(program_name)s.log
stderr_logfile=/supervisor/%(program_name)s.log
```

References

- Dockerfile
  - https://docs.docker.com/reference/builder/
- Docker command line
  - https://docs.docker.com/reference/commandline/cli/
- Serf
  - https://serfdom.io/
- Supervisor
  - http://supervisord.org/