Jump into DevOps World

22-23 September 2018

Before start

GIthub: https://github.com/Ima8/Jump_into_DevOps_world

digitalocean: http://bit.ly/do_get_100 Free $100

putty: https://www.ssh.com/ssh/putty/

Mysql workbench: https://www.mysql.com/products/workbench/

Docker CE: https://www.docker.com/get-started

node js 8.12: https://nodejs.org/en/download/

แมวไม่ว่าจะสีอะไรขอให้จับหนูได้ก็เป็นพอ

Hi, I’m MAX

คาดหวังอะไรจากคอร์สนี้ ?

“

Business is Happy

“

Everyone is Happy

“

Programmer is Happy

What is DevOps ?

“

‘DevOps’ ไม่ใช่แค่เรื่องของ ‘Automation’

“

‘DevOps’ ไม่ใช่แค่เรื่องของ ‘Automation’

‘DevOps’ ไม่ใช่เรื่องของการใช้ ‘Tools’

“

“DevOps” = “culture”

the DevOps methodology is not about technical capability.
Successful DevOps is an organizational challenge.

“

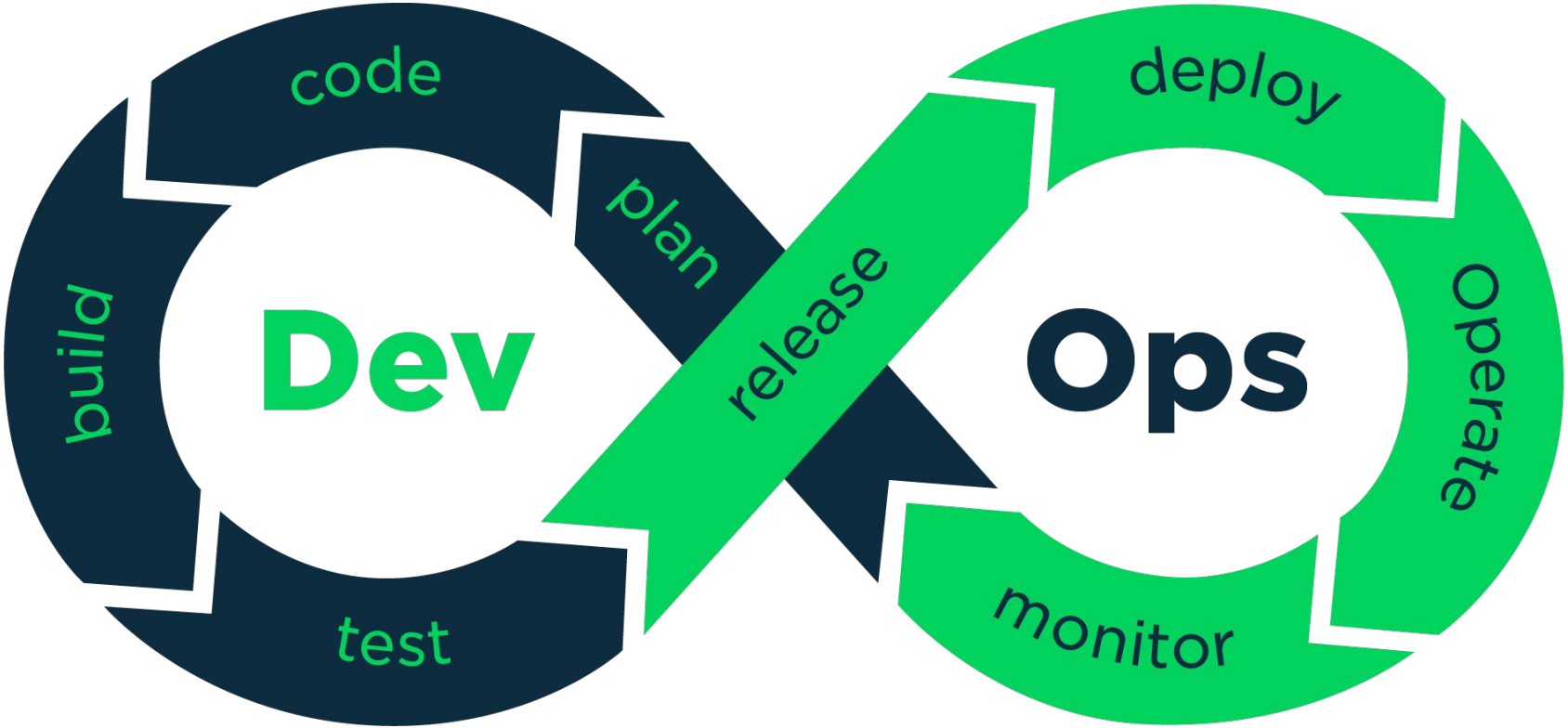
DevOps is a culture, not a role!
The whole company needs
to be doing DevOps for it to work.

Mike Dilworth

“

Same Goal

Deliver software faster and more reliably to customers

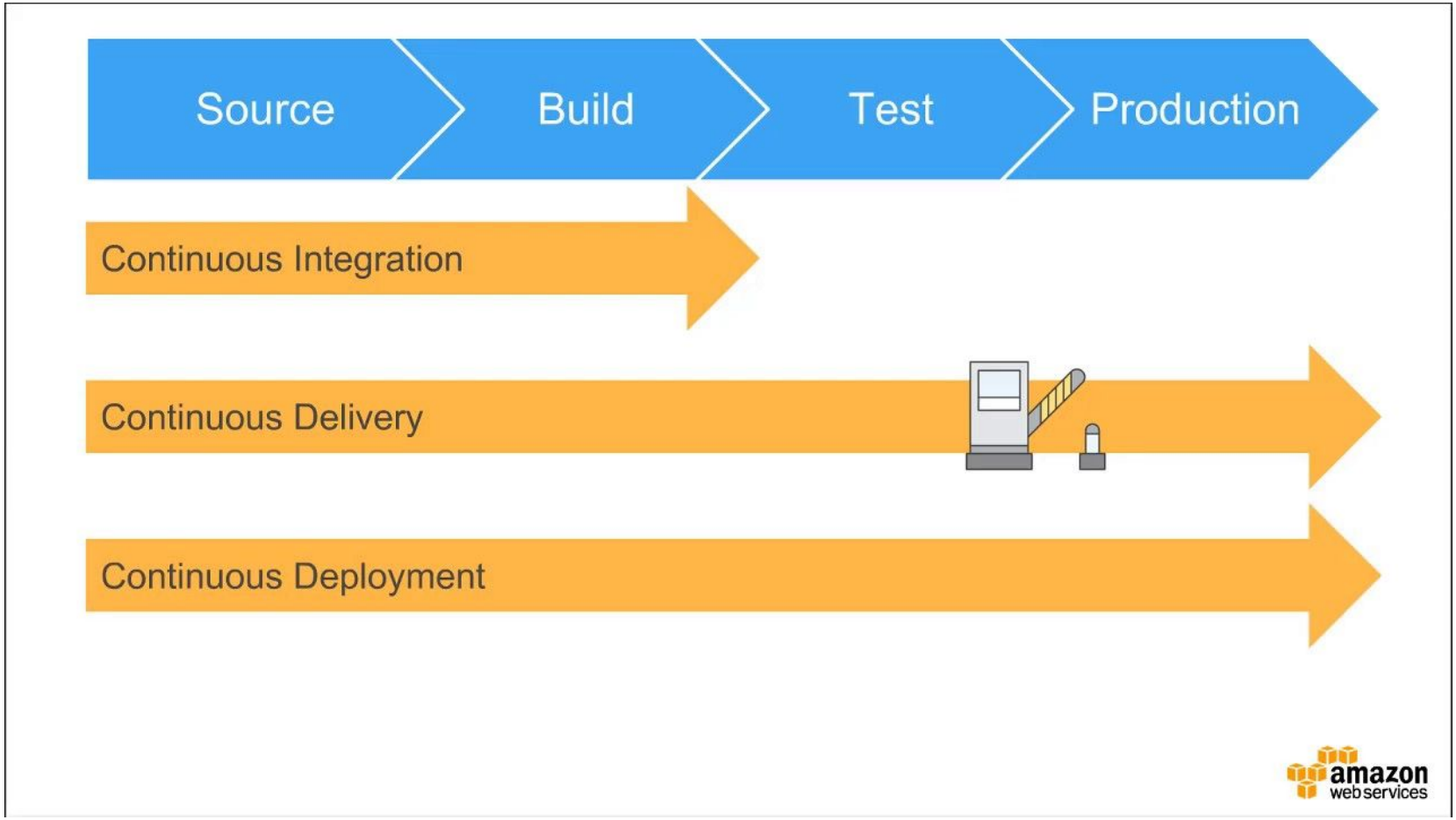

code
plan
build
Dev
test
release
deploy
Ops
Operate
monitor

What you should focus?

- High availability
- Respect production
- งานน่าเบื่อ ต้องไม่ทำเอง มีเวลากลับไปโฟกัสที่ Coding
- อะไรจะทำให้ Dev ปวดหัว ต้องไม่เกิดขึ้น!

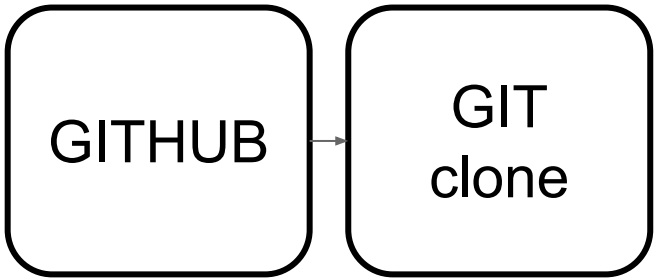
Schedule

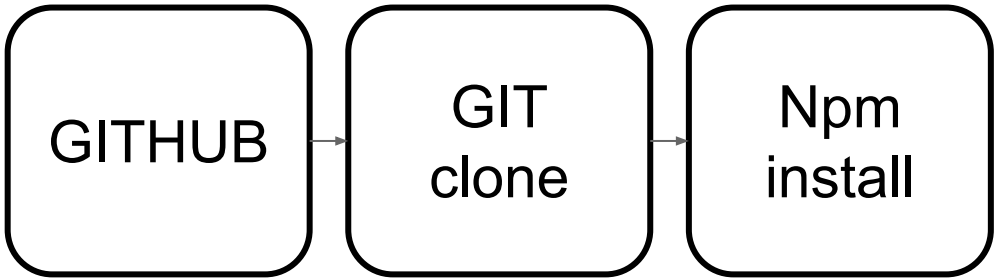
- What DevOps is
- What is CI/CD
- Docker
- Docker-compose
- Setup your MySql cluster for high availbailtiy (MariaDB Galera Cluster)
- Rancher
- CircleCI
- ELK for Logging
- API Gateway with Kong

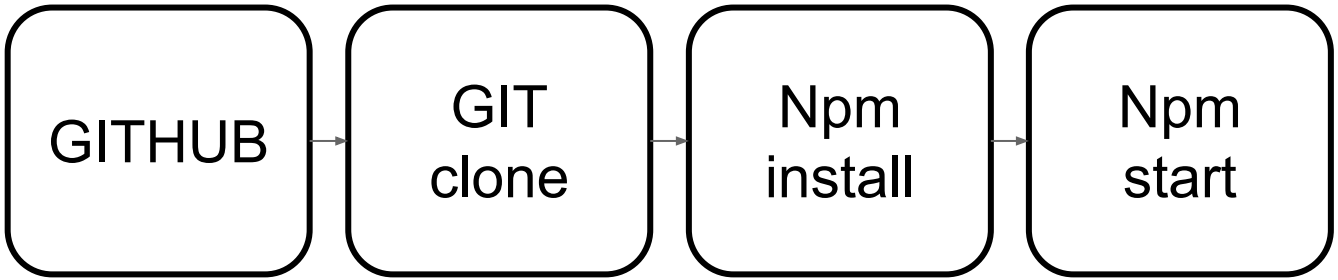

Source
Build
Test
Production
Continuous Integration
Continuous Delivery
Continuous Deployment
amazon
web services

Source code

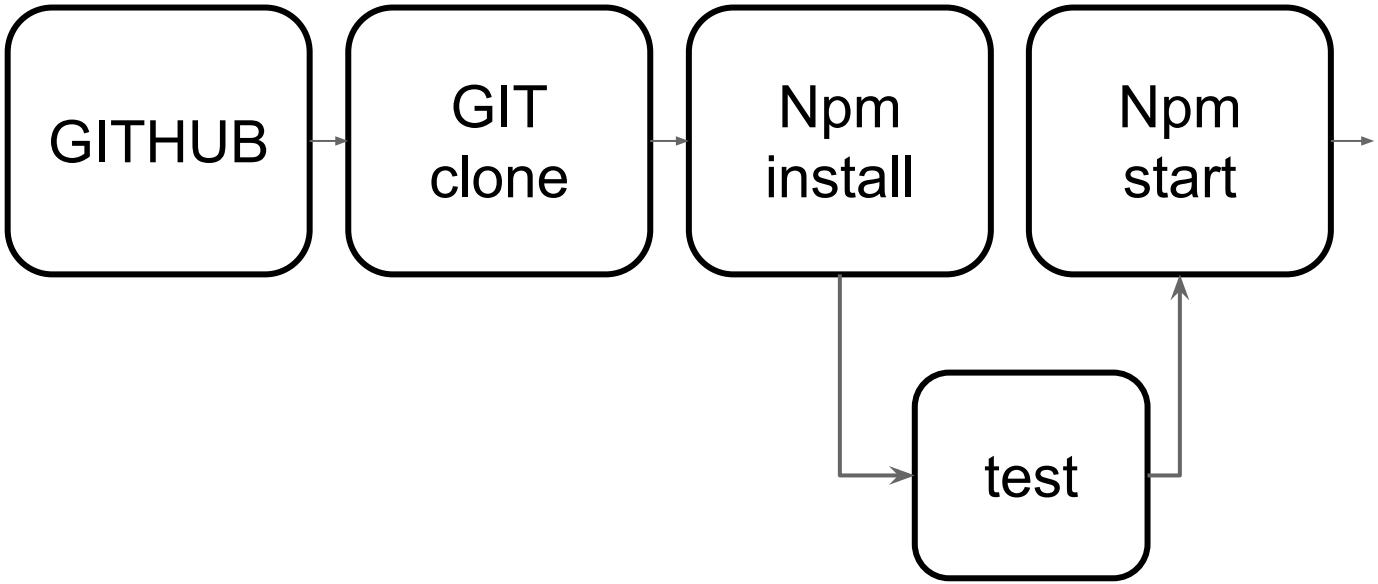
Source code

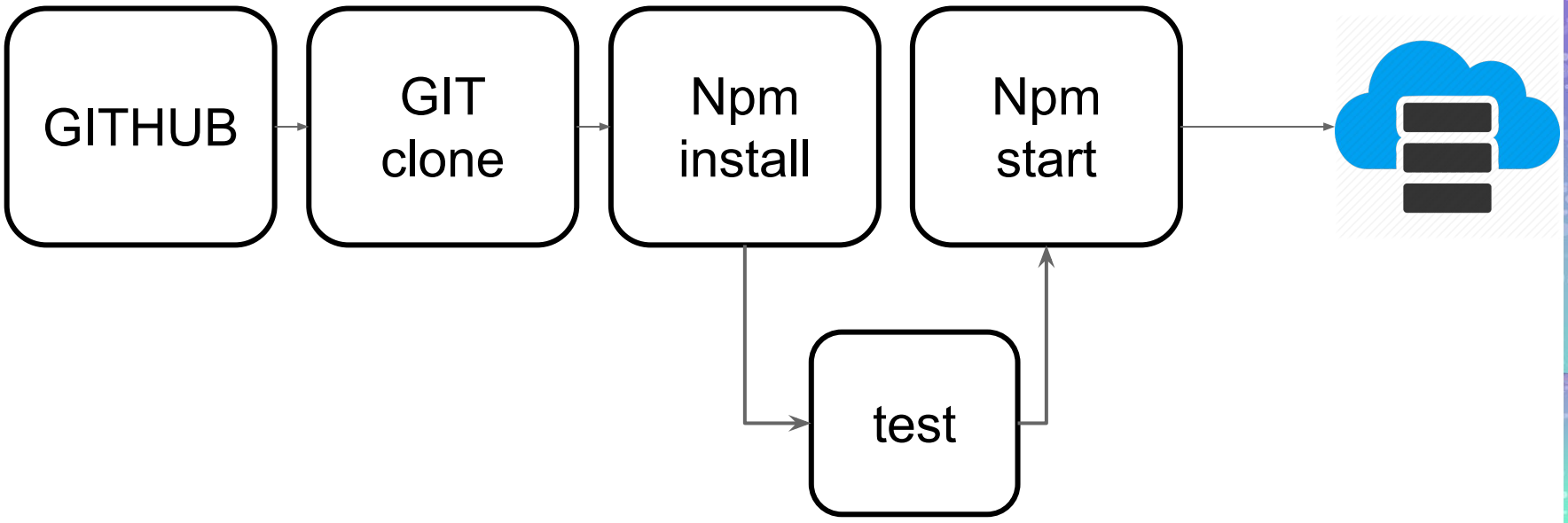

GITHUB
GIT
clone

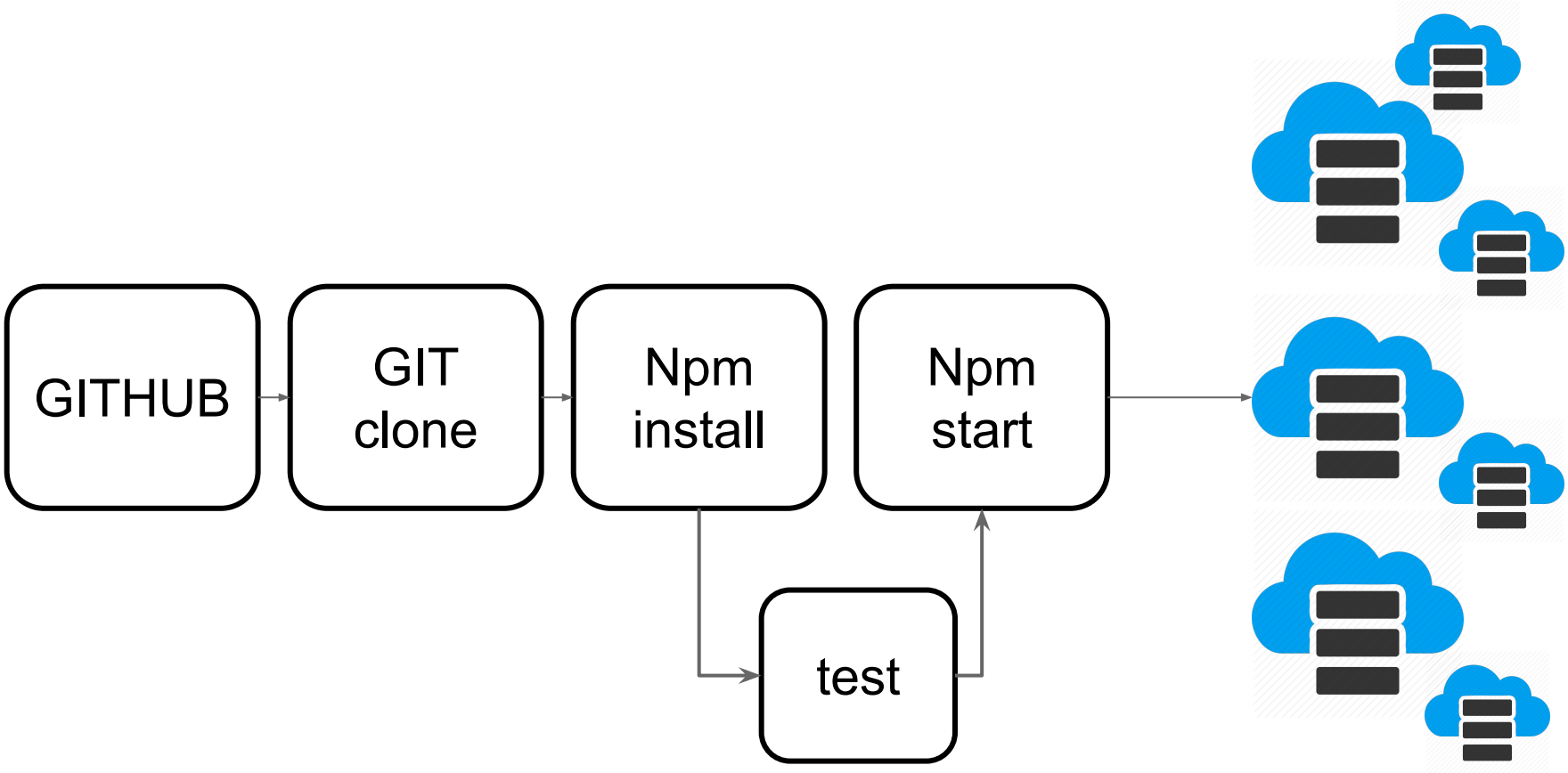

GITHUB
GIT
clone
Npm
install

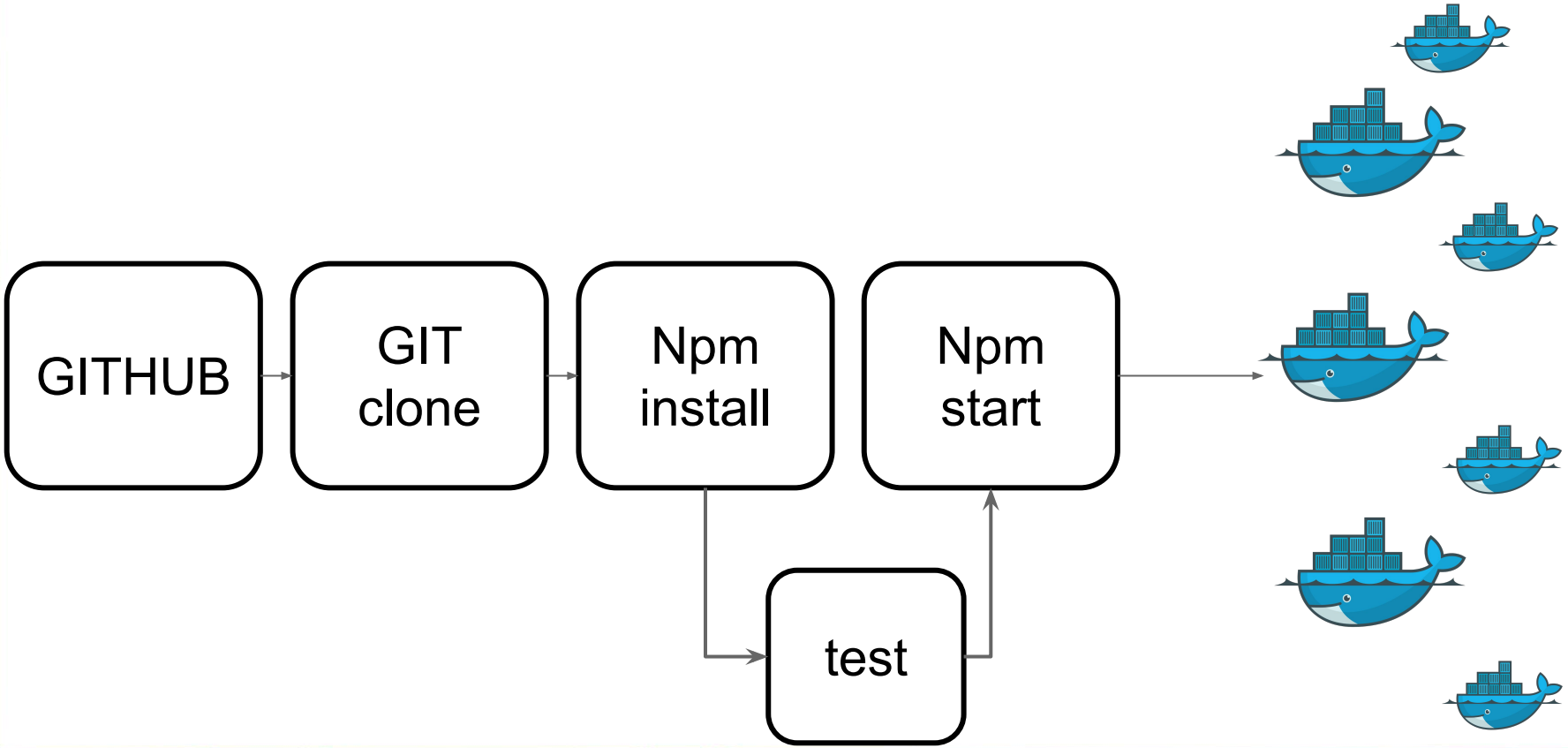

GITHUB
GIT
clone
Npm
install
Npm
start

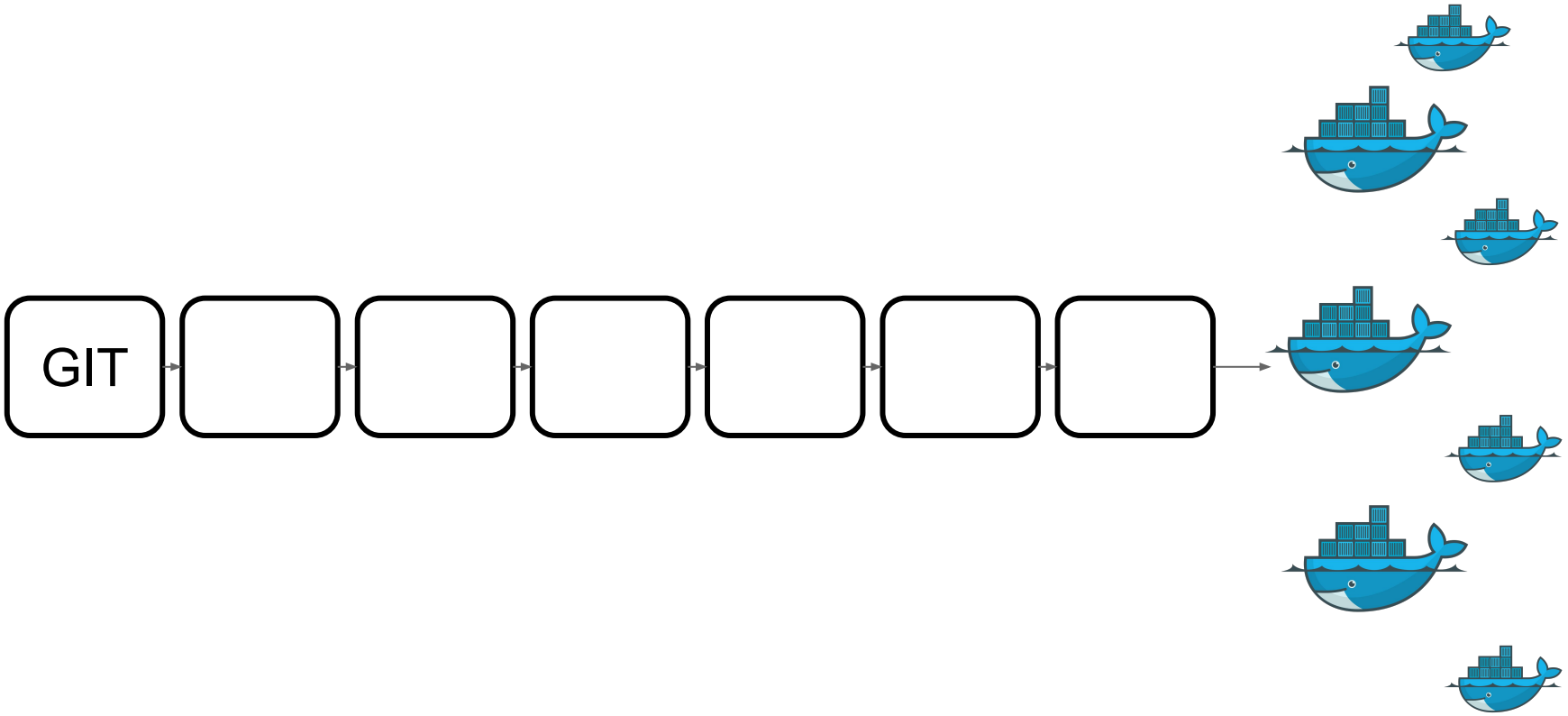

GITHUB
GIT
clone
Npm
install
Npm
start
Done?

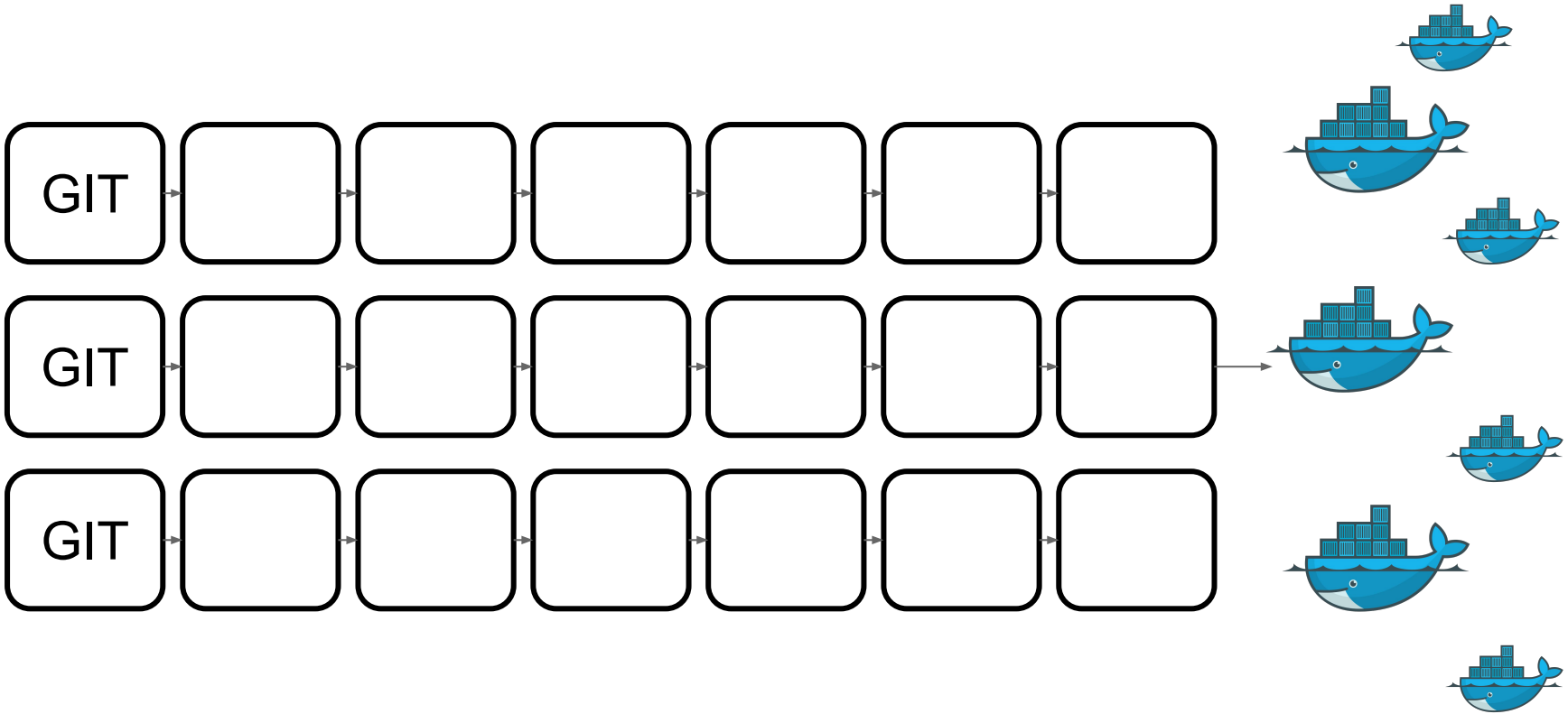

GITHUB
GIT
clone
Npm
install
Npm
start
test


GITHUB
GIT
clone
Npm
install
Npm
start
test


GITHUB
GIT
clone
Npm
install
Npm
start
test


GITHUB
GIT
clone
Npm
install
Npm
start
test


GIT


GIT
GIT
GIT

Schedule

- What DevOps is
- What is CI/CD
- Docker
- Docker-compose
- Setup your MySql cluster for high availbailtiy (MariaDB Galera Cluster)
- Rancher
- CircleCI
- ELK for Logging

continuous integration and delivery platform

- Configuration
  - .yml
- Fast Builds

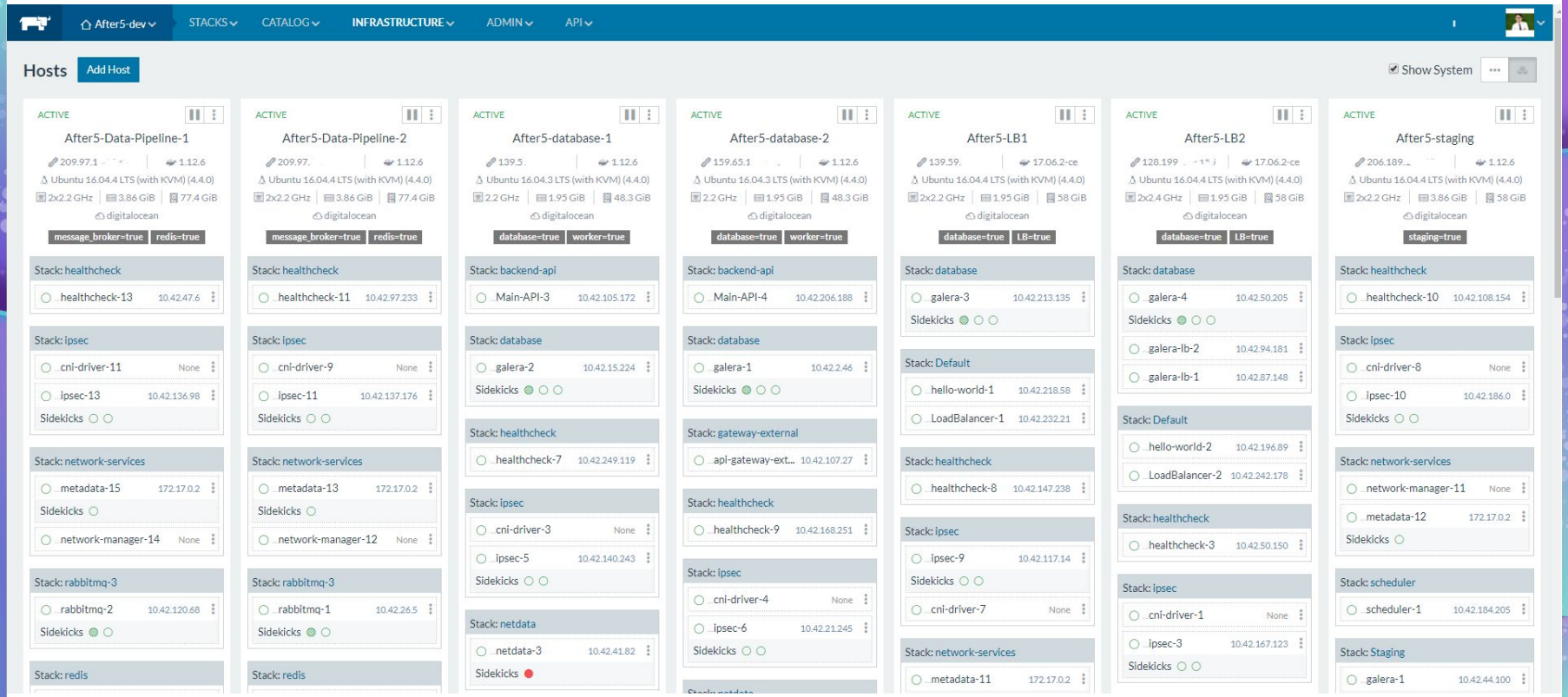
RANCHER

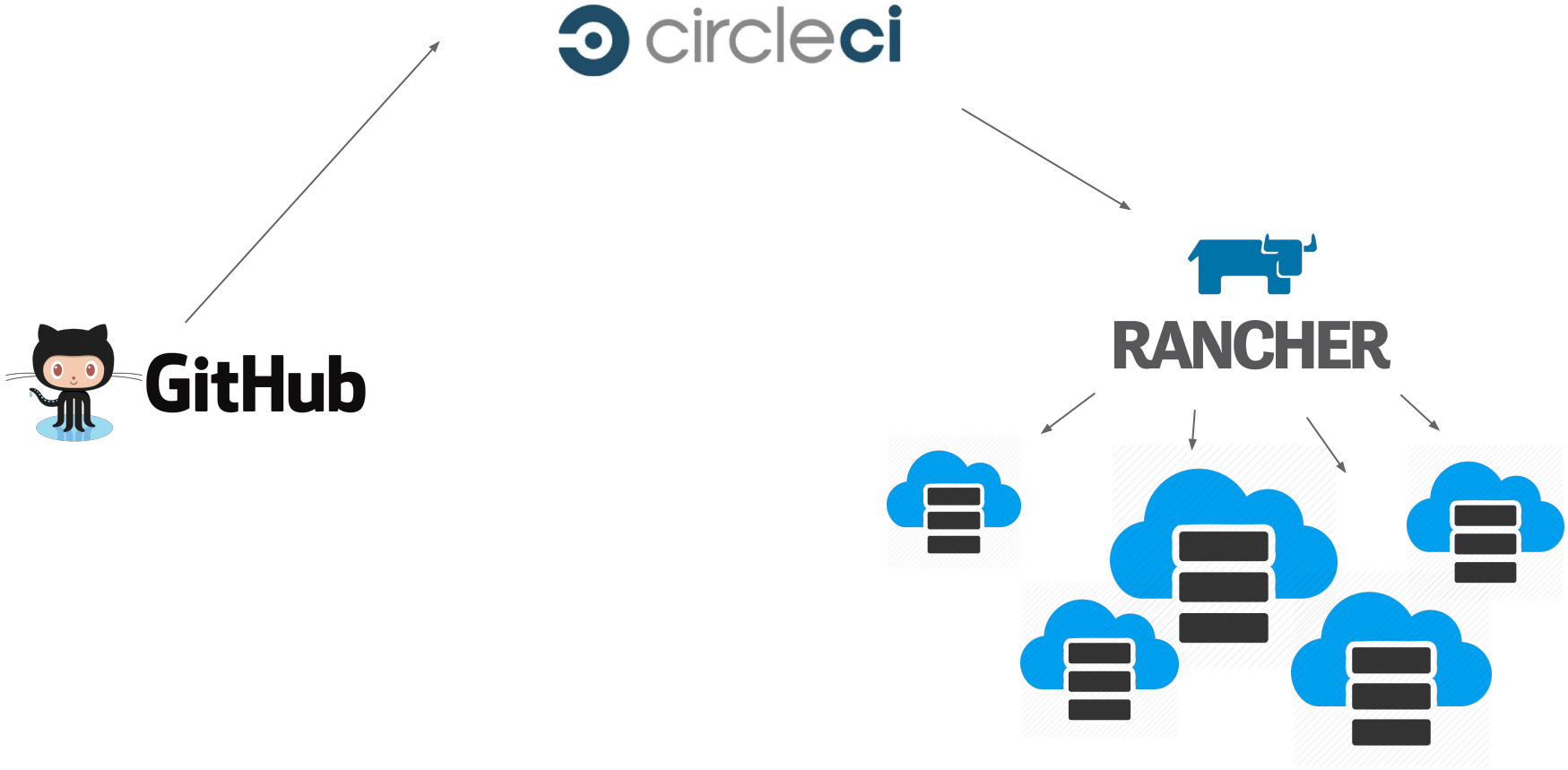
- Infrastructure orchestration
- Container orchestration and scheduling
- Application catalog
- enterprise-grade control

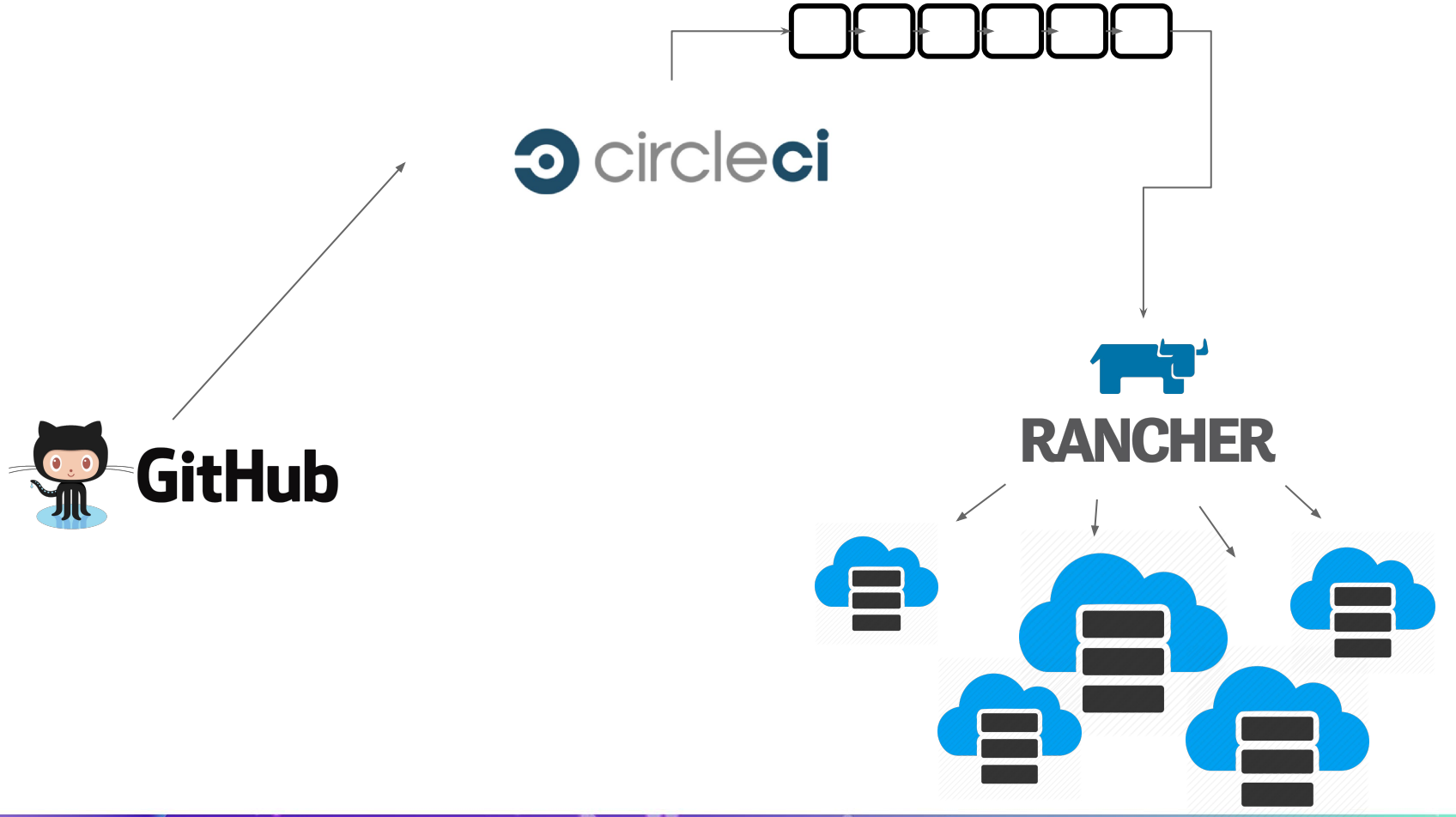
RANCHER

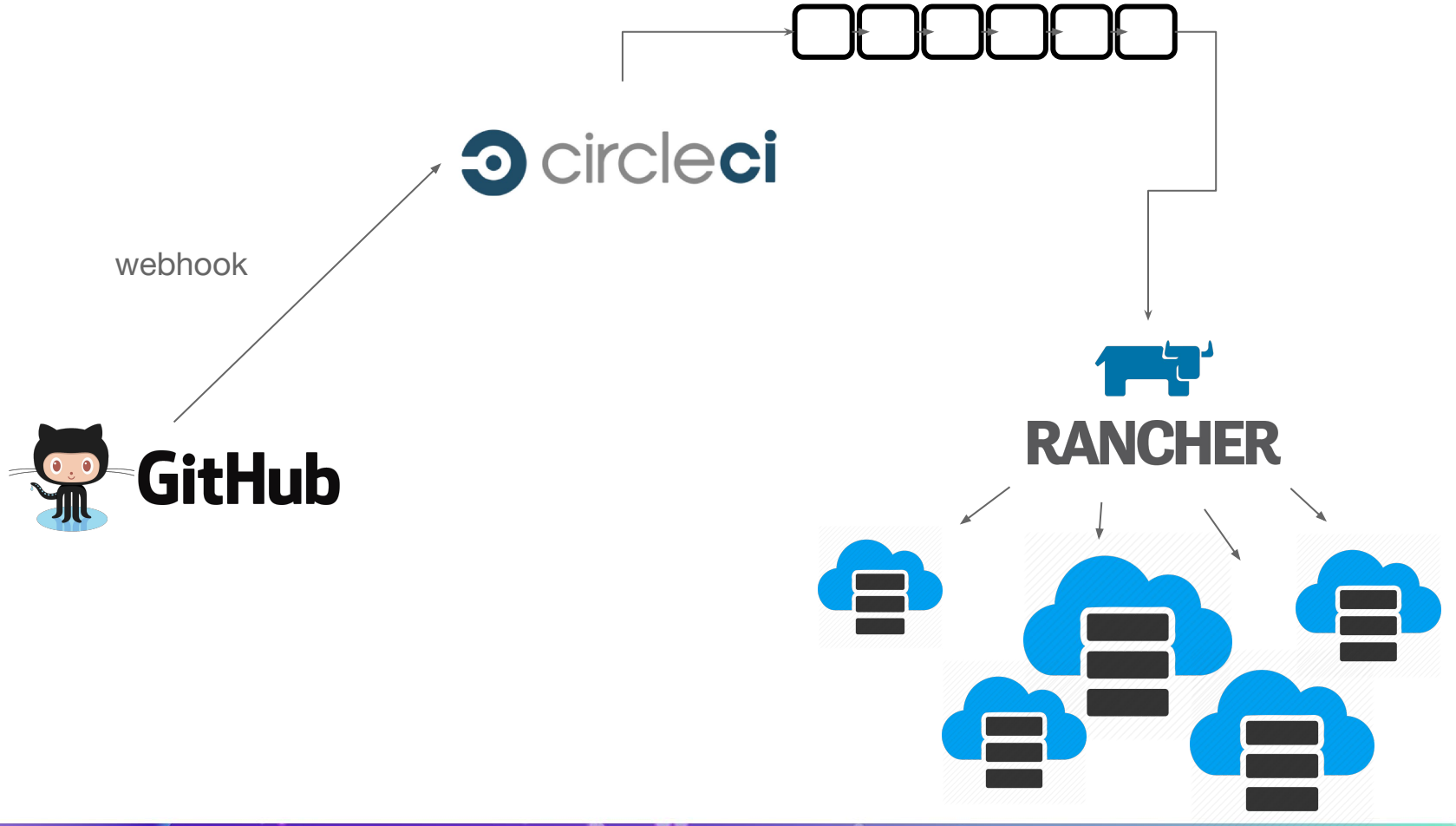
1. Rancher 1.6
    a. **Cattle** , Docker swarm, Kubernetes

2. Rancher 2.0
    a. Your Enterprise Kubernetes Platform


After5-dev
STACKS
CATALOG
INFRASTRUCTURE
ADMIN
API
Hosts
Add Host
Show System
ACTIVE
After5-Data-Pipeline-1
209.97.1
1.12.6
Ubuntu 16.04.4 LTS (with KVM) (4.4.0)
2x2.2 GHz
3.86 GiB
77.4 GiB
digitalocean
message_broker=true
redis=true
Stack: healthcheck
healthcheck-13 10.42.47.6
Stack: ipsec
cni-driver-11 None
ipsec-13 10.42.136.98
Sidekicks
Stack: network-services
metadata-15 172.17.0.2
Sidekicks
network-manager-14 None
Stack: rabbitmq-3
rabbitmq-2 10.42.120.68
Sidekicks
Stack: redis
ACTIVE
After5-Data-Pipeline-2
209.97.
1.12.6
Ubuntu 16.04.4 LTS (with KVM) (4.4.0)
2x2.2 GHz
3.86 GiB
77.4 GiB
digitalocean
message_broker=true
redis=true
Stack: healthcheck
healthcheck-11 10.42.97.233
Stack: ipsec
cni-driver-9 None
ipsec-11 10.42.137.176
Sidekicks
Stack: network-services
metadata-13 172.17.0.2
Sidekicks
network-manager-12 None
Stack: rabbitmq-3
rabbitmq-1 10.42.26.5
Sidekicks
Stack: redis
ACTIVE
After5-database-1
139.5
1.12.6
Ubuntu 16.04.3 LTS (with KVM) (4.4.0)
2.2 GHz
1.95 GiB
48.3 GiB
digitalocean
database=true
worker=true
Stack: backend-api
Main-API-3 10.42.105.172
Stack: database
galera-2 10.42.15.224
Sidekicks
Stack: healthcheck
healthcheck-7 10.42.249.119
Stack: ipsec
cni-driver-3 None
ipsec-5 10.42.140.243
Sidekicks
Stack: netdata
netdata-3 10.42.41.82
Sidekicks
ACTIVE
After5-database-2
159.65.1
1.12.6
Ubuntu 16.04.3 LTS (with KVM) (4.4.0)
2.2 GHz
1.95 GiB
48.3 GiB
digitalocean
database=true
worker=true
Stack: backend-api
Main-API-4 10.42.206.188
Stack: database
galera-1 10.42.2.46
Sidekicks
Stack: gateway-external
api-gateway-ext... 10.42.107.27
Stack: healthcheck
healthcheck-9 10.42.168.251
Stack: ipsec
cni-driver-4 None
ipsec-6 10.42.21.245
Sidekicks
ACTIVE
After5-LB1
139.59.
17.06.2-ce
Ubuntu 16.04.4 LTS (with KVM) (4.4.0)
2x2.2 GHz
1.95 GiB
58 GiB
digitalocean
database=true
LB=true
Stack: database
galera-3 10.42.213.135
Sidekicks
Stack: Default
hello-world-1 10.42.218.58
LoadBalancer-1 10.42.232.21
Stack: healthcheck
healthcheck-8 10.42.147.238
Stack: ipsec
ipsec-9 10.42.117.14
Sidekicks
cni-driver-7 None
Stack: network-services
metadata-11 172.17.0.2
ACTIVE
After5-LB2
128.199
17.06.2-ce
Ubuntu 16.04.4 LTS (with KVM) (4.4.0)
2x2.4 GHz
1.95 GiB
58 GiB
digitalocean
database=true
LB=true
Stack: database
galera-4 10.42.50.205
Sidekicks
galera-lb-2 10.42.94.181
galera-lb-1 10.42.87.148
Stack: Default
hello-world-2 10.42.196.89
LoadBalancer-2 10.42.242.178
Stack: healthcheck
healthcheck-3 10.42.50.150
Stack: ipsec
cni-driver-1 None
ipsec-3 10.42.167.123
Sidekicks
ACTIVE
After5-staging
206.189.
1.12.6
Ubuntu 16.04.4 LTS (with KVM) (4.4.0)
2x2.2 GHz
3.86 GiB
58 GiB
digitalocean
staging=true
Stack: healthcheck
healthcheck-10 10.42.108.154
Stack: ipsec
cni-driver-8 None
ipsec-10 10.42.186.0
Sidekicks
Stack: network-services
network-manager-11 None
metadata-12 172.17.0.2
Sidekicks
Stack: scheduler
scheduler-1 10.42.184.205
Stack: Staging
galera-1 10.42.44.100


circleci
GitHub
RANCHER

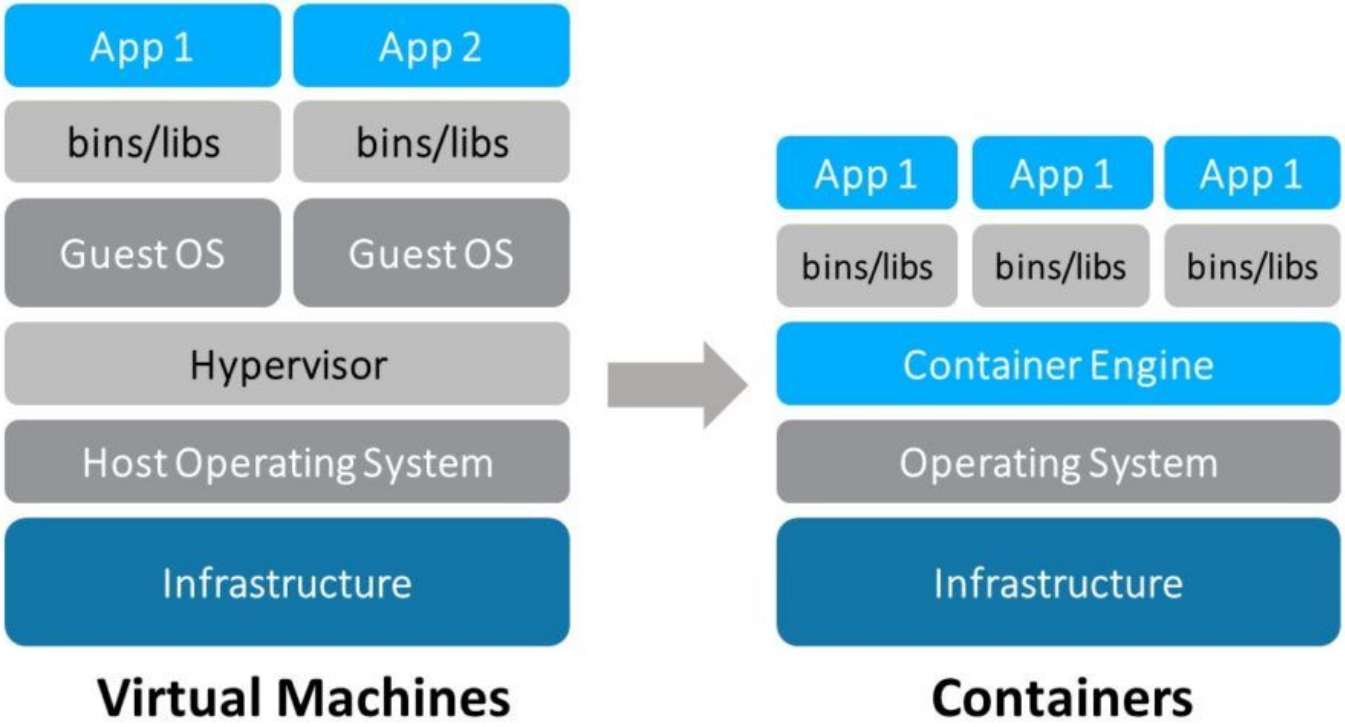

circleci
GitHub
RANCHER


circleci
webhook
GitHub
RANCHER

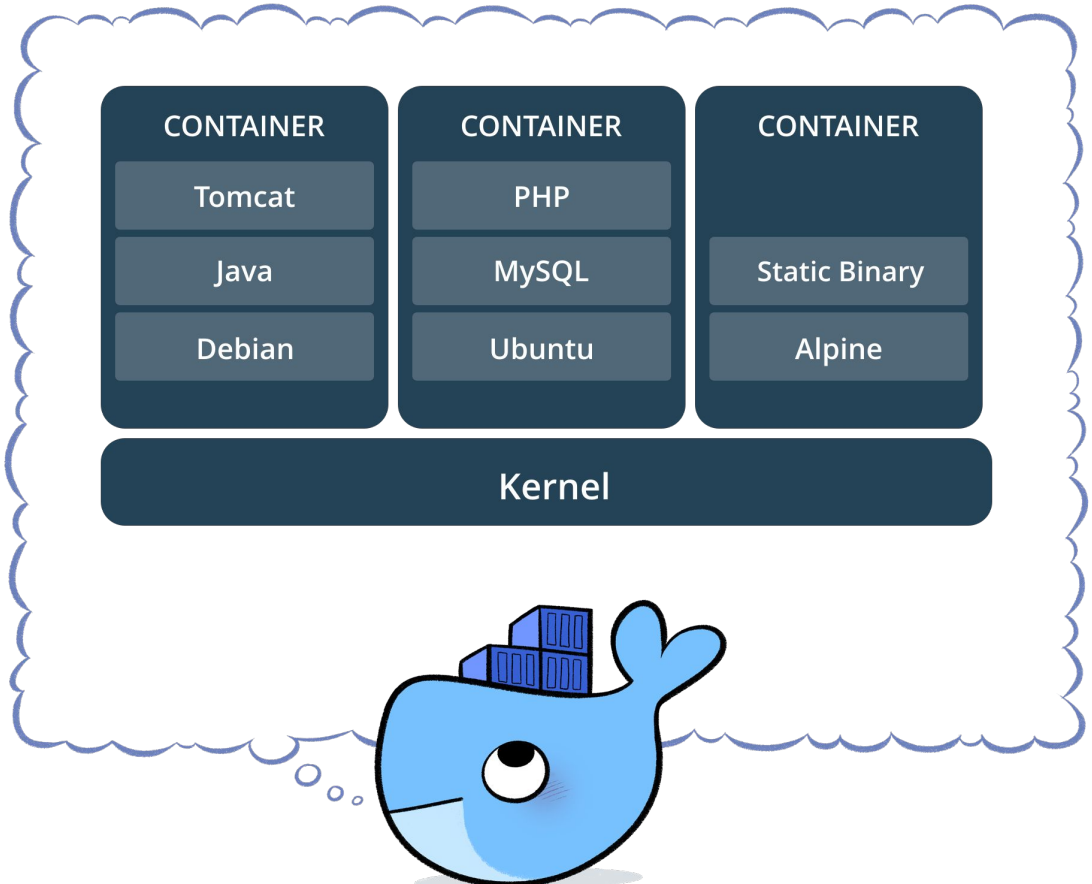
4 Steps for make your pipeline

1. Prepare your Rancher server
2. Add new server (worker)
3. Dockerfile
4. Circleci config

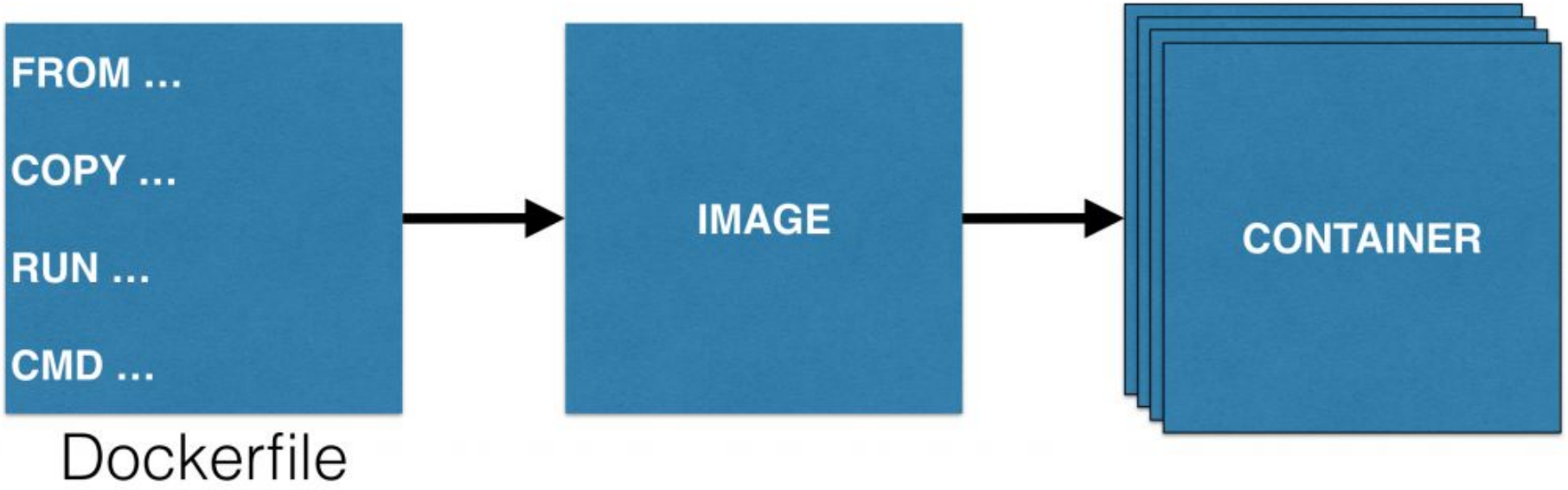
Docker vs VM


App 1
App 2
bins/libs
bins/libs
Guest OS
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App 1
App 1
App 1
bins/libs
bins/libs
bins/libs
Container Engine
Operating System
Infrastructure
Containers

Docker


CONTAINER
Tomcat
Java
Debian
CONTAINER
PHP
MySQL
Ubuntu
CONTAINER
Static Binary
Alpine
Kernel

Docker lifecycle



Docker Basic command

- docker run {image}
  - i.e. docker run ima8/demo1
- docker run -d {image}
- docker run -p 8080:80 {image}
- docker run --name demo {image}
- docker run --name demo -p 8080:80 {image}

Demo: Docker

“ docker run -p 8080:80 nginx ”

Docker: volume

- -v /app:/home/app
  - -v /app:./
- docker run /app:/home/app -d ima8/demo1

Cheat Sheet

Initialize swarm mode and listen on a specific interface
`docker swarm init --advertise-addr 10.1.0.2`

Join an existing swarm as a manager node
`docker swarm join --token <manager-token> 10.1.0.2:2377`

Join an existing swarm as a worker node
`docker swarm join --token <worker-token> 10.1.0.2:2377`

List the nodes participating in a swarm
`docker node ls`

Create a service from an image exposed on a specific port and deploy 3 instances
`docker service create --replicas 3 -p 80:80 --name web nginx`

List the services running in a swarm
`docker service ls`

Scale a service
`docker service scale web=5`

List the tasks of a service
`docker service tasks web`

RUN

```
docker run
          --rm remove container automatically after it exits
           -it connect the container to terminal
     --name web name the container
       -p 5000:80 expose port 5000 externally and map to port 80
-v ~/dev:/code create a host mapped volume inside the container
    alpine:3.4 the image from which the container is instantiated
       /bin/sh the command to run inside the container
```

Stop a running container through SIGTERM
`docker stop web`

Stop a running container through SIGKILL
`docker kill web`

Create an overlay network and specify a subnet
`docker network create --subnet 10.1.0.0/24 --gateway 10.1.0.1 -d overlay mynet`

List the networks
`docker network ls`

List the running containers
`docker ps`

Delete all running and stopped containers
`docker rm -f $(docker ps -aq)`

Create a new bash process inside the container and connect it to the terminal
`docker exec -it web bash`

Print the last 100 lines of a container's logs
`docker logs --tail 100 web`

BUILD

Build an image from the Dockerfile in the current directory and tag the image
`docker build -t myapp:1.0 .`

List all images that are locally stored with the Docker engine
`docker images`

Delete an image from the local image store
`docker rmi alpine:3.4`

SHIP

Pull an image from a registry
`docker pull alpine:3.4`

Retag a local image with a new image name and tag
`docker tag alpine:3.4 myrepo/myalpine:3.4`

Log in to a registry (the Docker Hub by default)
`docker login my.registry.com:8000`

Push an image to a registry
`docker push myrepo/myalpine:3.4`

Workshop: Docker I

“ run nginx website with docker and edit the content ”

nginx path: /usr/share/nginx/html

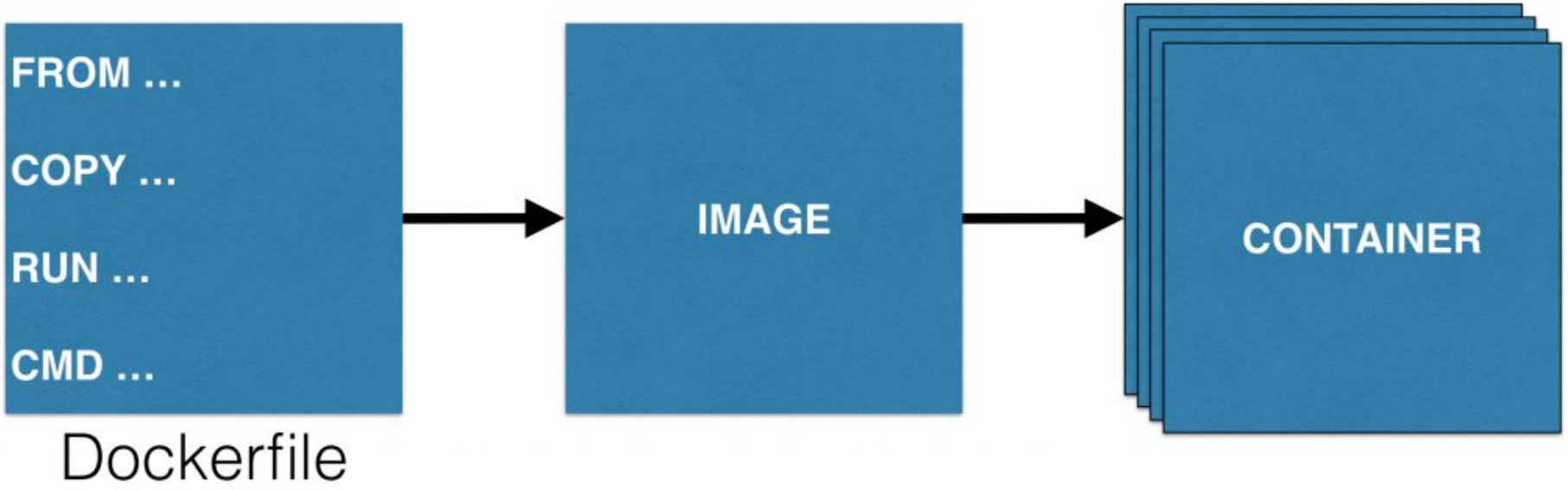
Workshop: Docker I

“ run nginx website with docker and edit the content ”

nginx path: /usr/share/nginx/html

docker run -v ~/nginx:/usr/share/nginx/html -p 80:80 -d nginx

Docker lifecycle

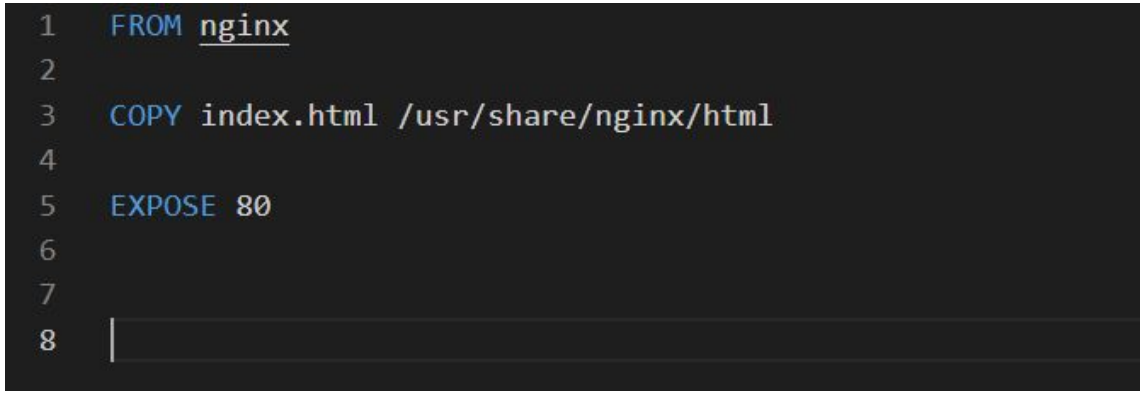


Docker Basic command

- docker build .
  - docker build -t ima8/demo1 .
  - docker build -f Dockerfile.staging .
  - docker build -t ima8/demo1 -f Dockerfile.staging .
- docker run -p 80:80 ima8/demo1 .
- docker images

Workshop: Dockerfile I

“ Make your own Dockerfile ”



Workshop: Dockerfile II

“ Make your docker file and CMD ”

https://github.com/Ima8/example-nodejs-express

Docker File

```
1  FROM node:8.11-alpine          # Base image
2
3
4  RUN mkdir -p /app
5
6
7  COPY / /app
8
9
10 WORKDIR /app
11
12
13 CMD ["node", "/app/server.js"]    # run command
14
15
16 EXPOSE 3000                  # Post
17
```

Guideline

- docker build -t ima8/demo2 .
- docker run -p 8000:3000 ima8/demo2

Dockerfile

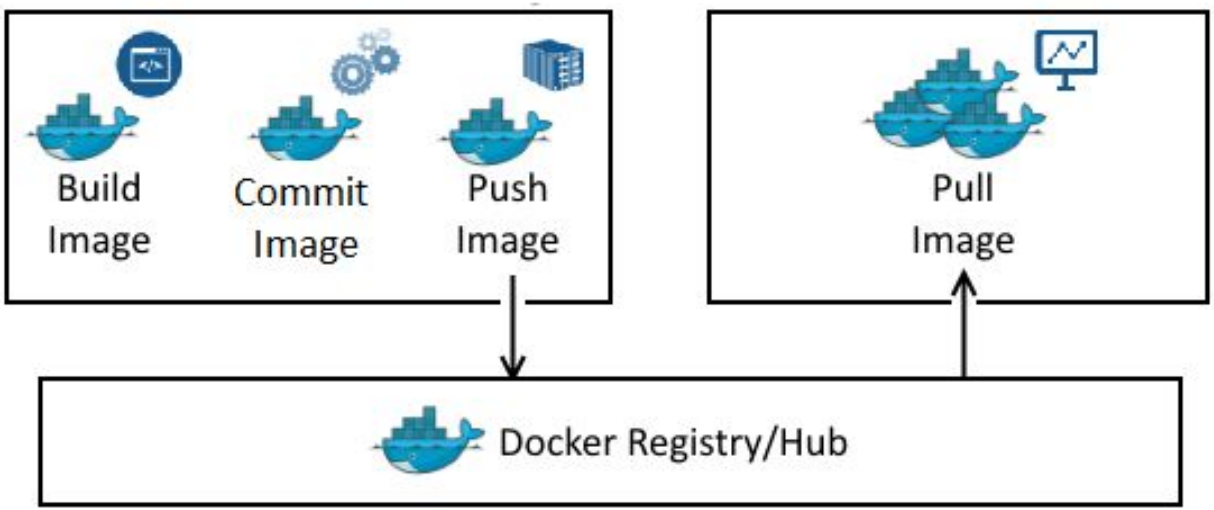
```
1   FROM node:8.11-alpine
2
3   RUN mkdir -p /app
4
5   COPY thisisapp/ /app
6
7   WORKDIR /app
8
9   RUN npm install
10
11  CMD ["node","/app/bin/www"]
12
13  EXPOSE 3000
14
```

Workshop: Dockerfile III

“ Make your docker file and run it with volume ”

github.com/Ima8/node-express-multer-image-upload

Docker Registry



Docker Registry

- Register at https://hub.docker.com/
- docker login
- docker build -t {username}/{name} .
  - i.e. docker build -t ima8/demo .
  - docker tag {image_name} {username}/{name}

- docker push {username}/{name}

Docker Registry

- docker pull {username}/{name}
- docker run {username}/{name}

Workshop: Docker Registry I

“ ให้เอา Workshop docker file ก่อนหน้า
ขึ้น Docker Registry ”

Docker Private Registry

- Gitlab.com
- Self Hosted (gitlab omnibus)

Gitlab Docker Private Registry

```
# docker login my.gitlab.com

# docler build -t my.gitlab.com/api/foo .

# docker pull my.gitlab.com/api/foo
```

Gitlab Registry

```yaml
version: '3.6'
services:
  gitlab:
    image: 'gitlab/gitlab-ce:rc'
    restart: always
    hostname: 'gitlab'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'https://gitlab.local'
        nginx['ssl_certificate'] = "/etc/gitlab/ssl/gitlab.crt"
        nginx['ssl_certificate_key'] = "/etc/gitlab/ssl/gitlab.key"
        registry_external_url 'https://registry.local'
        gitlab_rails['registry_host'] = "registry"
    ports:
      - '80:80'
      - '443:443'
      - '22:22'
    volumes:
      - './config:/etc/gitlab'
      - './ssl:/etc/gitlab/ssl'
      - './logs:/var/log/gitlab'
      - './data:/var/opt/gitlab'

  registry:
    image: registry:2
    restart: always
    hostname: registry
    volumes:
      - ./registry:/var/lib/registry
```

Self Signed Certificate for Gitlab

```
# mkdir -p ssl

# openssl genrsa -out ssl/gitlab.key 2048

# openssl req -new -key ssl/gitlab.key -out ssl/gitlab.csr

# openssl x509 -req -days 365 \

   -in ssl/gitlab.csr \

   -signkey ssl/gitlab.key \

   -out ssl/gitlab.crt
```

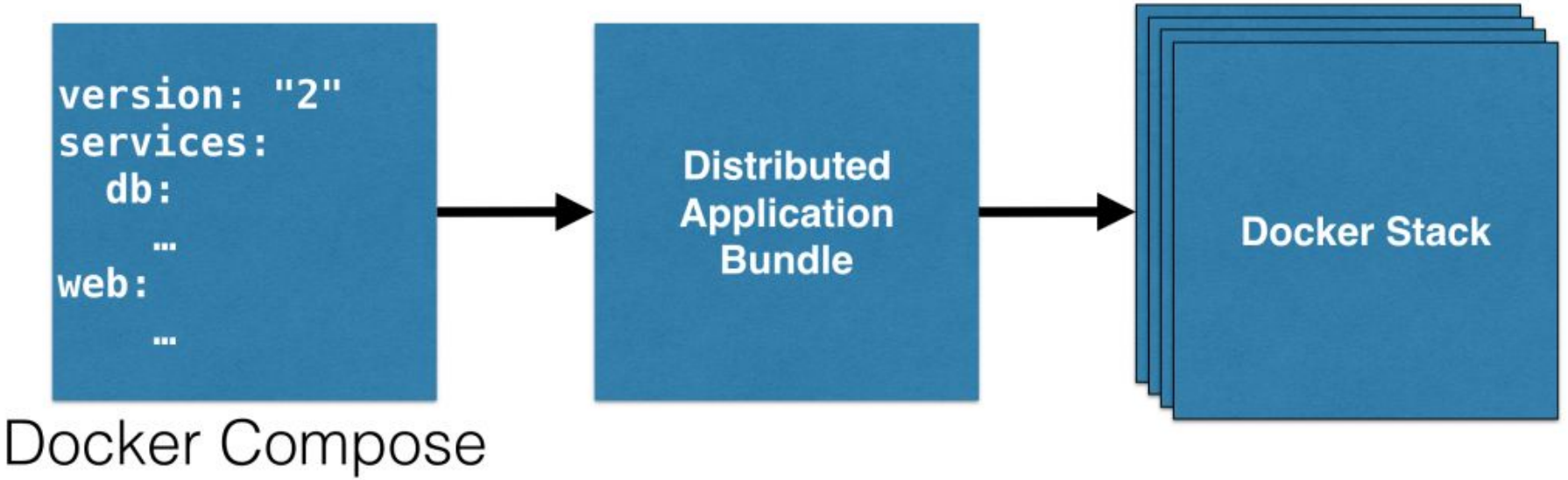
Self Signed Certificate for Registry

```
# openssl genrsa -out ssl/registry.key 2048

# openssl req -new -key ssl/registry.key \

    -out ssl/registry.csr

# openssl x509 -req -days 365 \

    -in ssl/registry.csr \

    -signkey ssl/registry.key \

    -out ssl/registry.crt
```

Schedule

- What DevOps is
- What is CI/CD
- Docker
- Docker-compose
- Setup your MySql cluster for high availbailtiy (MariaDB Galera Cluster)
- Rancher
- CircleCI
- ELK for Logging

Docker compose



Docker Registry

https://hub.docker.com/_/wordpress/

Docker compose

```yaml
version: '2'

services:
  wp:
    image: wordpress
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_PASSWORD=mypass
    volumes:
      - /home/user/workplace/app:/var/www/html
    ports:
      - 80:80

```

Docker compose

```yaml
version: '2'

services:
  wp:
    image: wordpress
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_PASSWORD=mypass
    volumes:
      - ./app:/var/www/html
    ports:
      - 80:80
```

Docker compose

```yaml
version: '2'

services:
  wp:
    image: wordpress
    environment:
      - WORDPRESS_DB_HOST=db
      - WORDPRESS_DB_PASSWORD=mypass
    volumes:
      - /home/user/workplace/app:/var/www/html
    ports:
      - 80:80

```

```
docker-compose start
docker-compose stop

docker-compose pause
docker-compose unpause

docker-compose ps
docker-compose up
docker-compose down
```

Docker compose

docker-compose up vs docker-compose start

- docker-compose start = Starts existing
- docker-compose up = build , create , start

Docker compose

- docker-compose up
  - docker-compose up -d
  - docker-compose -f {docker-compose.yml} up

- docker-compose down
- docker-compose logs
  - docker-compose logs -f
    - follow the logs

Docker compose

- docker-compose up --force-recreate
- docker-compose down && docker-compose build --no-cache && docker-compose up

Workshop: Docker-compose I

“ run example wordpress project “

path: /ch_docker-compose/1_wordpress

wordpress docker-compose

Workshop: Docker-compose II

“ run your own dockerfile with docker-compose “

```yaml
version: '2'

services:
  webserver:
    build: .
    ports:
      - 8080:80
```

Workshop: Docker-compose III

“ run your own image with docker-compose “

4 Steps for make your pipeline

1. Prepare your Rancher server
2. Add new server (worker)
3. Dockerfile
4. Circleci config

Prepare your Rancher ecosystem

1. **Rancher node**

2. **Database node (mysql)**

Prepare your Rancher ecosystem

Rancher node

1. SINGLE CONTAINER
2. SINGLE CONTAINER - EXTERNAL DATABASE
3. SINGLE CONTAINER - BIND MOUNT MYSQL VOLUME
4. Full Active/Active HA

Prepare your Rancher ecosystem

https://github.com/Ima8/Jump_into_DevOps_world

Jump_into_DevOps_world/ch_setup_rancher/

- **docker-compose up -d**
- **docker-compose logs -f**

Prepare your Rancher ecosystem

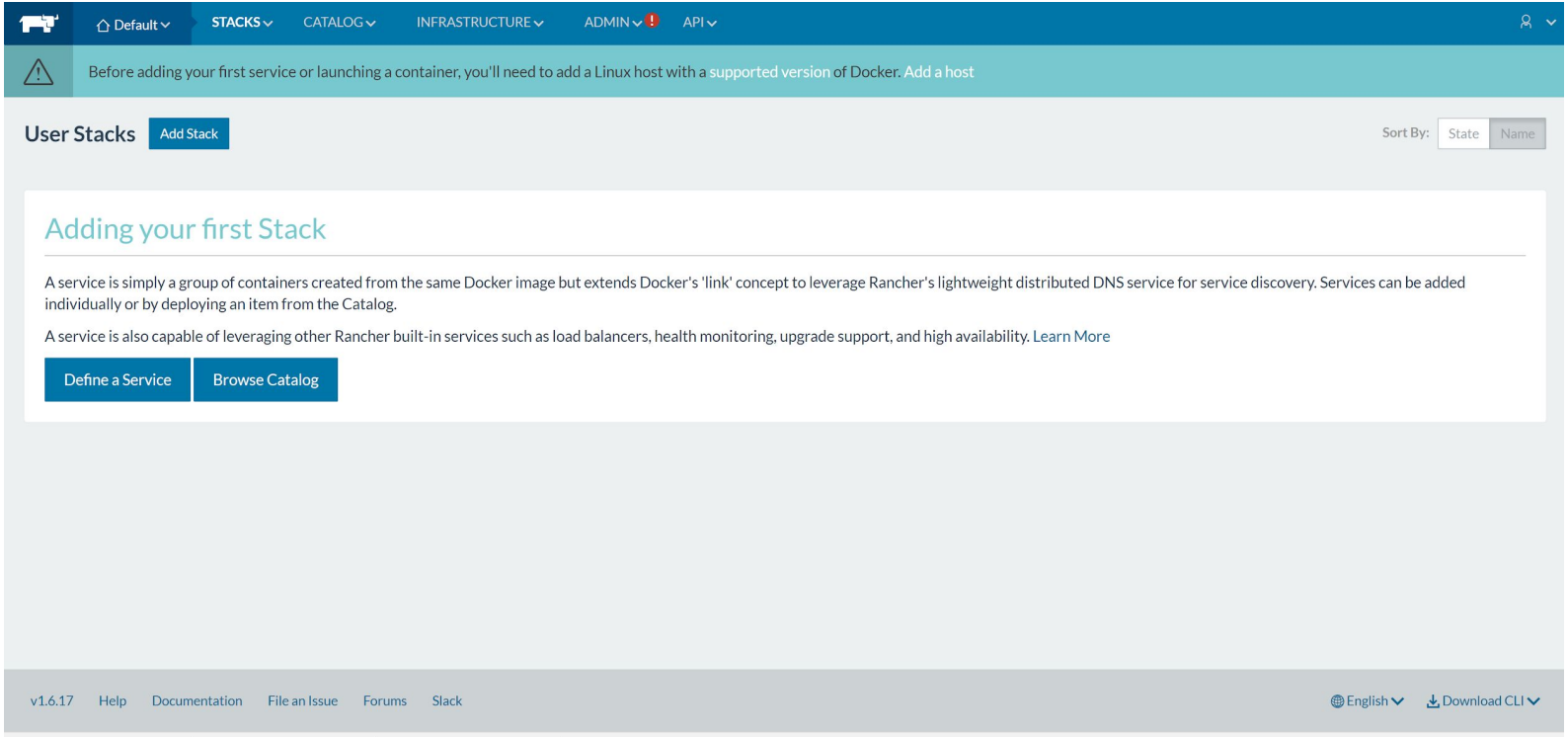
```
mysql:
  image: mysql:5.7
  restart: always
  volumes:
    - ./var/lib/mysql:/var/lib/mysql
  ports:
    - "3306:3306"
  environment:
    - MYSQL_ROOT_PASSWORD=your_root_password!
    - MYSQL_DATABASE=rancher
    - MYSQL_USER=rancher
    - MYSQL_PASSWORD=your_password!
  healthcheck:
      test: "/usr/bin/mysql --user=rancher --password=your_password! --execute \"SHOW DATABASES;\""
      interval: 5s
      timeout: 60s
      retries: 10
```

Prepare your Rancher ecosystem

```yaml
version: "2.1"
services:
  rancher:
    image: rancher/server:stable
    restart: always
    ports:
      - "8080:8080"
    links:
      - mysql
    depends_on:
      mysql:
        condition: service_healthy
    environment:
      - CATTLE_DB_CATTLE_MYSQL_HOST=mysql ## Your public ip
      - CATTLE_DB_CATTLE_MYSQL_PORT=3306
      - CATTLE_DB_CATTLE_MYSQL_NAME=rancher
      - CATTLE_DB_CATTLE_USERNAME=rancher
      - CATTLE_DB_CATTLE_PASSWORD=your_password!
  mysql:
```

Workshop: rancher I

“ setup rancher on cloud server “


Default
STACKS
CATALOG
INFRASTRUCTURE
ADMIN
API
Before adding your first service or launching a container, you'll need to add a Linux host with a supported version of Docker. Add a host
User Stacks
Add Stack
Sort By:
State
Name
Adding your first Stack
A service is simply a group of containers created from the same Docker image but extends Docker's 'link' concept to leverage Rancher's lightweight distributed DNS service for service discovery. Services can be added individually or by deploying an item from the Catalog.
A service is also capable of leveraging other Rancher built-in services such as load balancers, health monitoring, upgrade support, and high availability. Learn More
Define a Service
Browse Catalog
v1.6.17
Help
Documentation
File an Issue
Forums
Slack
English
Download CLI

0.0.0.0:8080

Lunch

Schedule

- What DevOps is
- What is CI/CD
- Docker
- Docker-compose
- Setup your MySql cluster for high availbailtiy (MariaDB Galera Cluster)
- Rancher
- CircleCI
- ELK for Logging

MariaDB Galera Cluster

MariaDB Galera Cluster is a synchronous multi-master cluster for MariaDB. It is available on Linux only, and only supports the XtraDB/InnoDB storage

MariaDB Galera Cluster

- Synchronous replication
- Active-active multi-master topology
- Read and write to any cluster node
- Automatic membership control, failed nodes drop from the cluster
- Automatic node joining
- True parallel replication, on row level
- Direct client connections, native MariaDB look & feel

MariaDB Galera Cluster

- at least 3 node per cluster (recommand 5)
- memory at least 1Gb per node
- private networking if possible

MariaDB Galera Cluster

Same step for every node

- sudo apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 0xF1656F24C74CD1D8
- sudo add-apt-repository 'deb [arch=amd64,i386,ppc64el] http://nyc2.mirrors.digitalocean.com/mariadb/repo/10.3/ubuntu xenial main'
- sudo apt-get update
- sudo apt-get install mariadb-server
- sudo apt-get install rsync

MariaDB Galera Cluster

Same step for first node

- sudo mysql_secure_installation
- sudo systemctl stop mysql

/etc/apt/sources.list.d/galera.list

```
deb http://releases.galeracluster.com/mysql-wsrep-5.6/ubuntu xenial main
deb http://releases.galeracluster.com/galera-3/ubuntu xenial main
```

Galera Cluster for MySQL

Same step for every node

- sudo nano /etc/apt/preferences.d/galera.pref
-

/etc/apt/preferences.d/galera.pref

```
# Prefer Codership repository
Package: *
Pin: origin releases.galeracluster.com
Pin-Priority: 1001
```

MariaDB Galera Cluster

Same step for every node

- sudo nano /etc/mysql/conf.d/galera.cnf

/etc/mysql/conf.d/galera.cnf on the first node

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://first_ip,second_ip,third_ip"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="this_node_ip"
wsrep_node_name="this_node_name"
```

MariaDB Galera Cluster

Same step for every node

/etc/mysql/conf.d/galera.cnf on the first node

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

# Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

# Galera Cluster Configuration
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://first_ip,second_ip,third_ip"

# Galera Synchronization Configuration
wsrep_sst_method=rsync

# Galera Node Configuration
wsrep_node_address="this_node_ip"
wsrep_node_name="this_node_name"
```

MariaDB Galera Cluster

First node on Cluster

- sudo galera_new_cluster
- tail -f /var/log/syslog
- mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"

other node on clsuter

- sudo systemctl start mysql
- mysql -u root -p -e "SHOW STATUS LIKE 'wsrep_cluster_size'"
-

MariaDB Galera Cluster

ใครอยากเสริมอะไรมั้ย

Workshop: setup Galera

Tip: MariaDB Galera Cluster

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

max_connections=1024
key_buffer_size= 20M
table_open_cache = 256
max_allowed_packet = 32M
sort_buffer_size = 20M
read_buffer_size = 20M
thread_concurrency = 4

# INNODB #
innodb-flush-method            = O_DIRECT
innodb-log-files-in-group      = 2
innodb-log-file-size           = 512M
innodb-flush-log-at-trx-commit = 1
innodb-file-per-table          = 1
innodb-buffer-pool-size        = 2G
```

Tip: MariaDB Galera Cluster

```
# change auto_increment_increment and auto_increment_offset automatically
wsrep_auto_increment_control=1

# enable "strictly synchronous" semantics for read operations
wsrep_causal_reads=0

# retry autoinc insert, which failed for duplicate key error
wsrep_drupal_282555_workaround=1

wsrep_retry_autocommit=3
```

MariaDB Galera Cluster: รับมือกับพญานาค

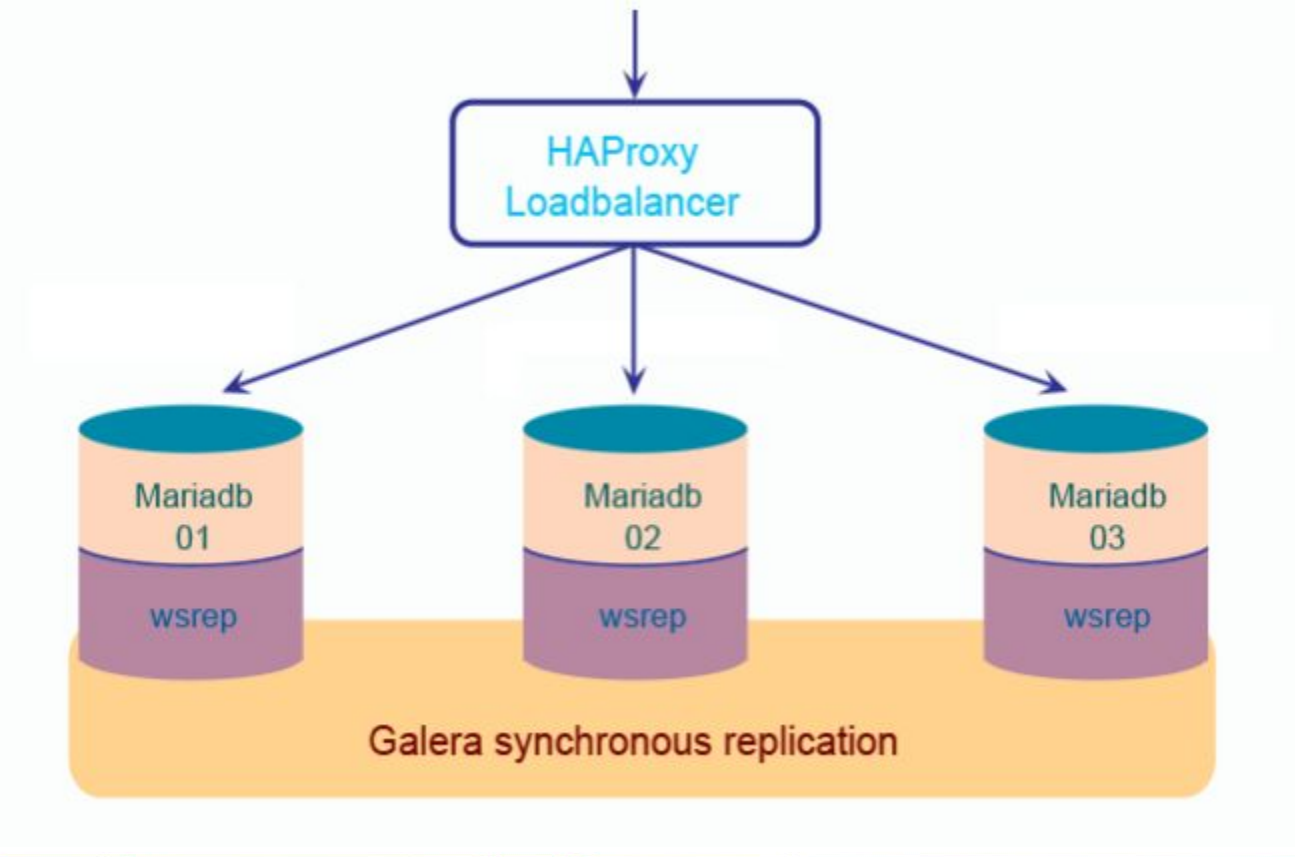
Can’t Auth after grant user access

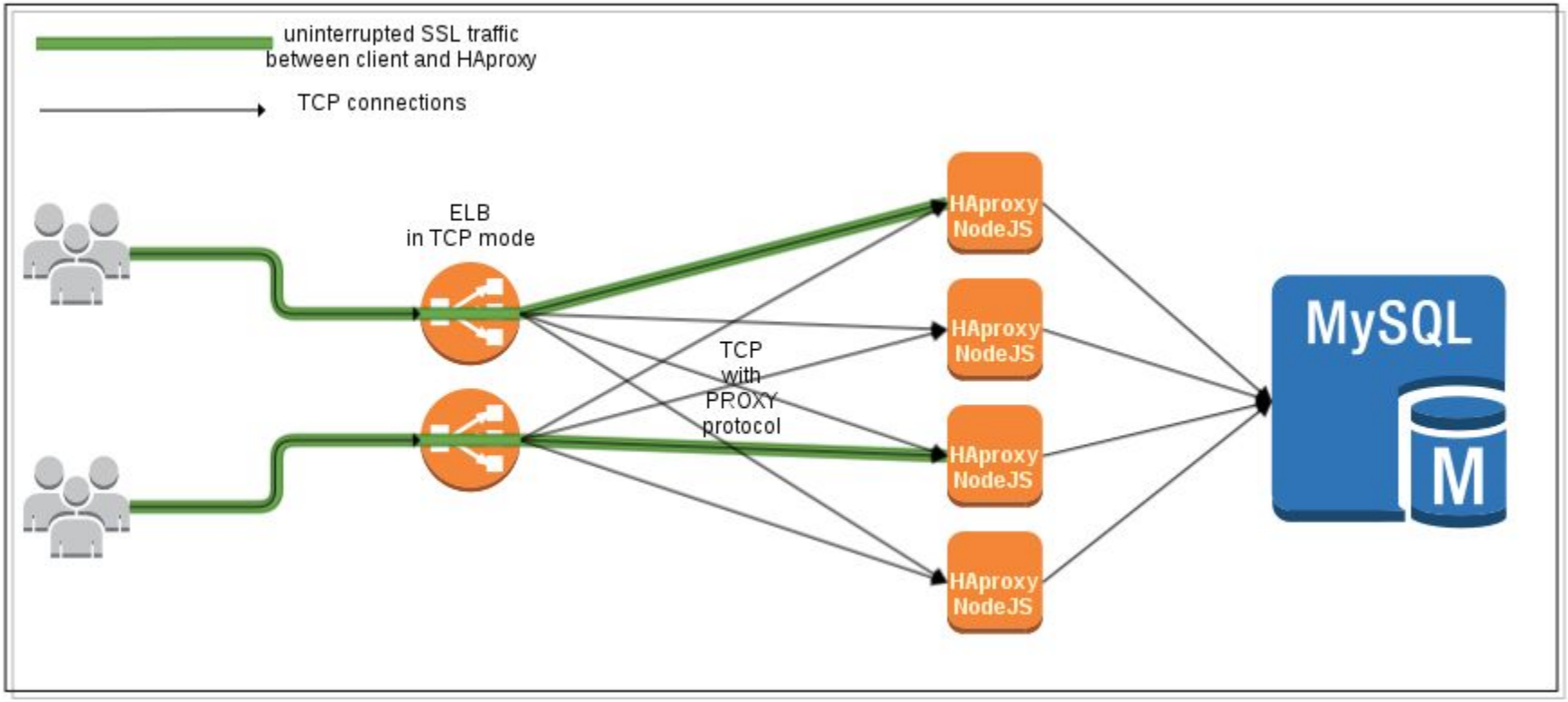
MariaDB Galera Cluster: Reset password

```
/etc/init.d/mysql stop

mysqld_safe --skip-grant-tables &

mysql -uroot

use mysql;

flush privileges;

select * from user;

UPDATE user SET Password =
PASSWORD(‘thisispassword’)
WHERE User = 'user_a';

GRANT ALL PRIVILEGES ON *.* TO
'user_a' WITH GRANT OPTION;

flush privileges;

systemctl start mysql
```

MariaDB Galera Cluster: Reset password

repeat every node in cluster


HAProxy
Loadbalancer
Mariadb
01
wsrep
Mariadb
02
wsrep
Mariadb
03
wsrep
Galera synchronous replication


uninterrupted SSL traffic
between client and HAproxy
TCP connections
ELB
in TCP mode
TCP
with
PROXY
protocol
HAproxy
NodeJS
HAproxy
NodeJS
HAproxy
NodeJS
HAproxy
NodeJS
MySQL
M

Load balance Galera Cluster with NGINX

```
stream {
        upstream galera {
                least_conn;
                server 10.1.1.10:3306;
                server 10.1.1.20:3306;
                server 10.1.1.30:3306;
                zone tcp_mem 64k;
        }

        server {
                listen 13306;
                proxy_pass galera;
                proxy_connect_timeout 1s;
        }
}
```

Schedule

- What DevOps is
- What is CI/CD
- Docker
- Docker-compose
- Setup your MySql cluster for high availbailtiy (MariaDB Galera Cluster)
- Rancher
- CircleCI
- ELK for Logging

4 Steps for make your pipeline

1. Prepare your Rancher server
2. Add new server (worker)
3. Dockerfile
4. Circleci config

Prepare your Rancher server

1. Rancher node

2. Database node

Workshop: rancher II

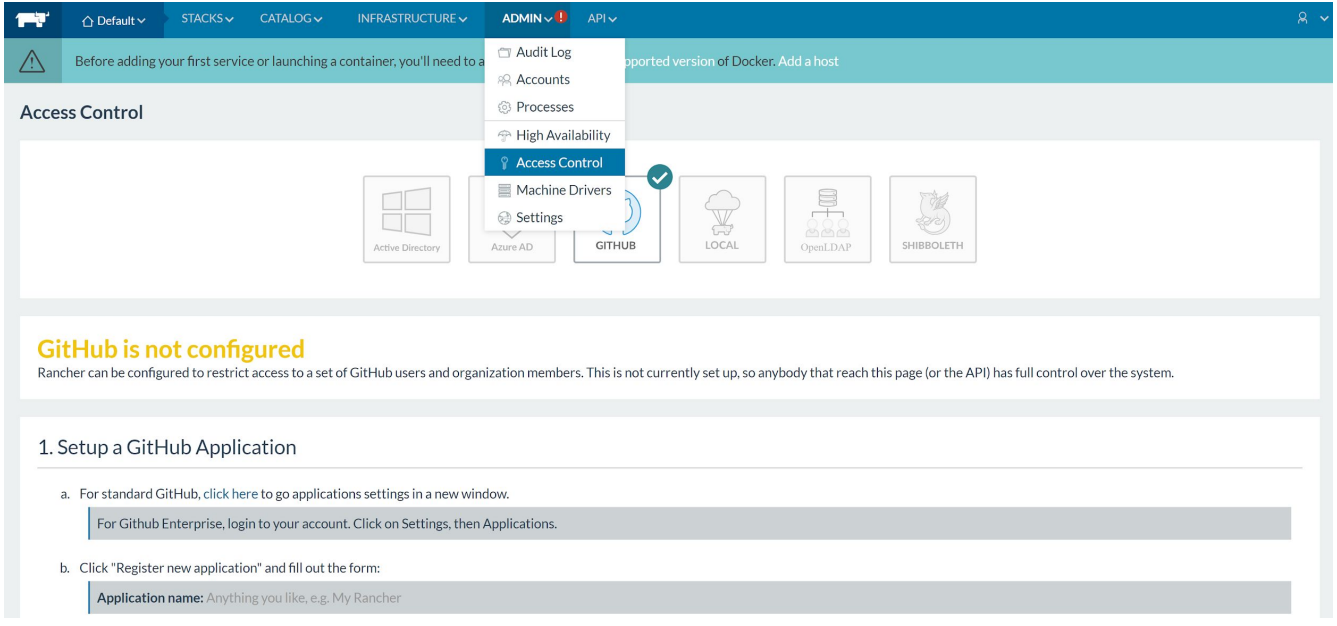
“ setup rancher with Galera Cluster“

4 Steps for make your pipeline

1. Prepare your Rancher server
2. Add new server (worker)
3. Dockerfile
4. Circleci config

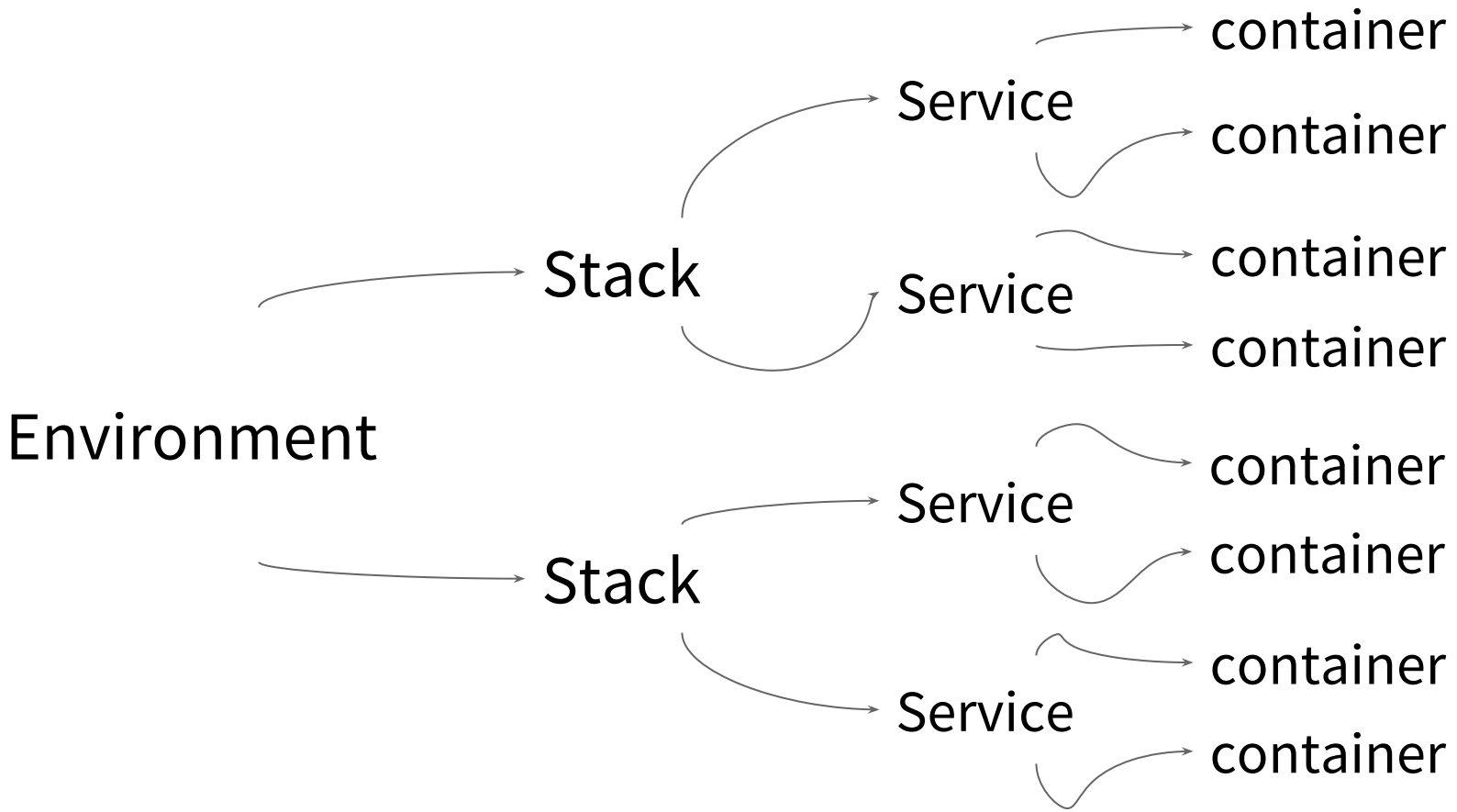
Rancher first time

1. **Set Access Control!!!**



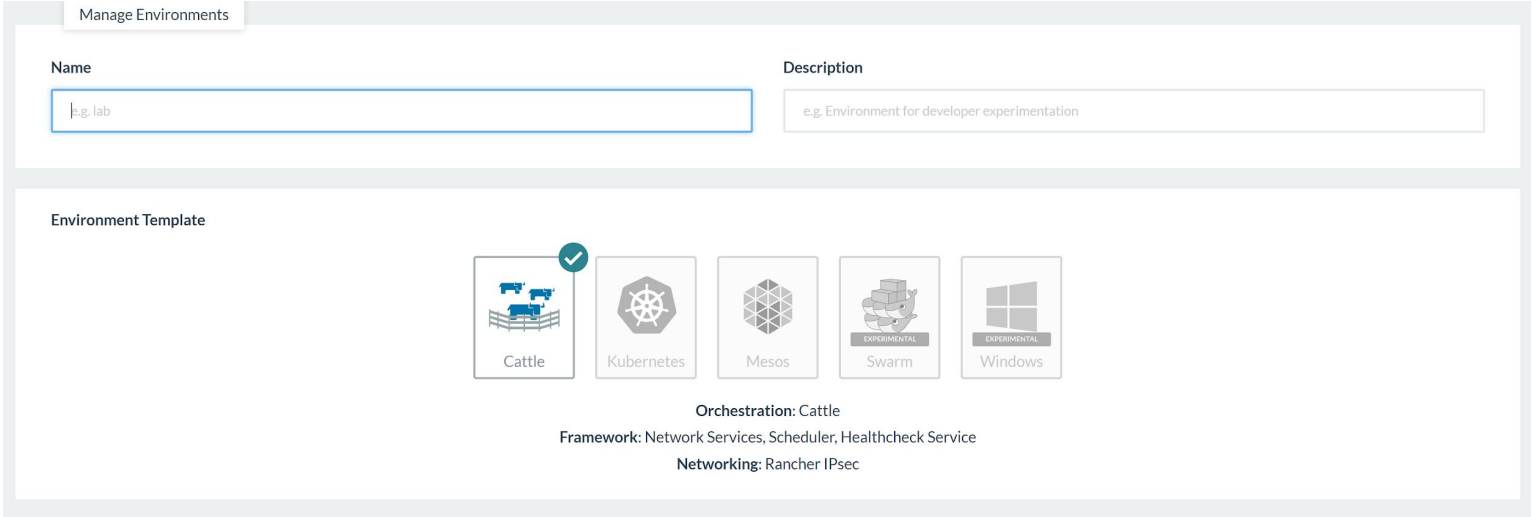
Rancher first time

1. Environment
2. Stack
3. Service


Environment
Stack
Stack
Service
Service
Service
Service
container
container
container
container
container
container
container
container

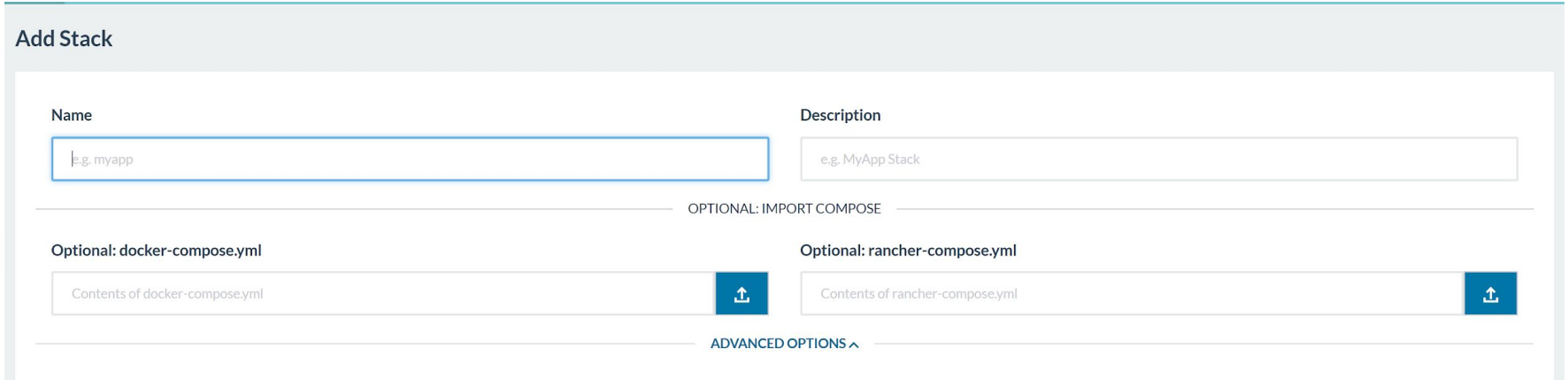
Rancher first time

1. Environment



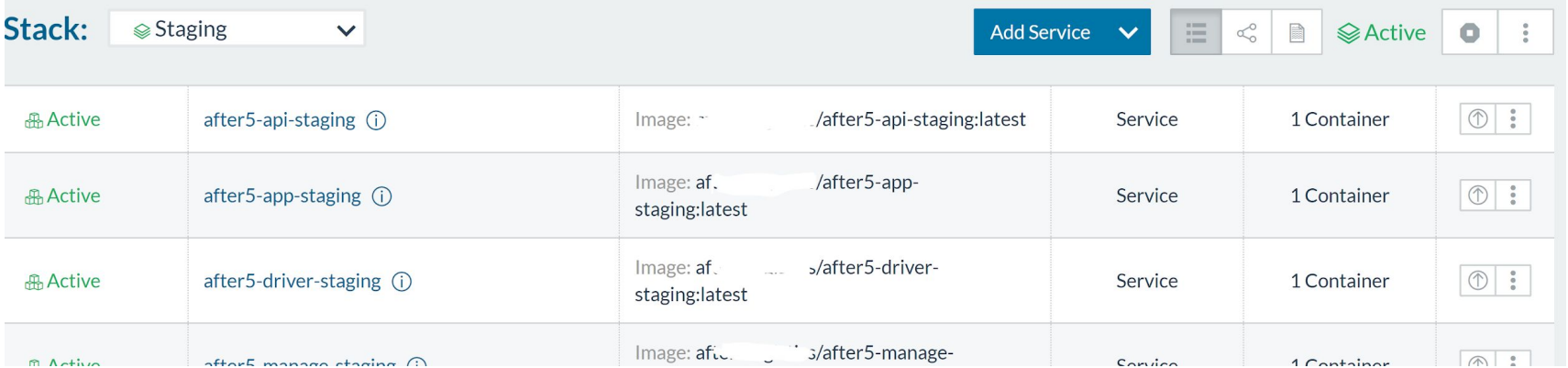
Rancher first time

2. Stack



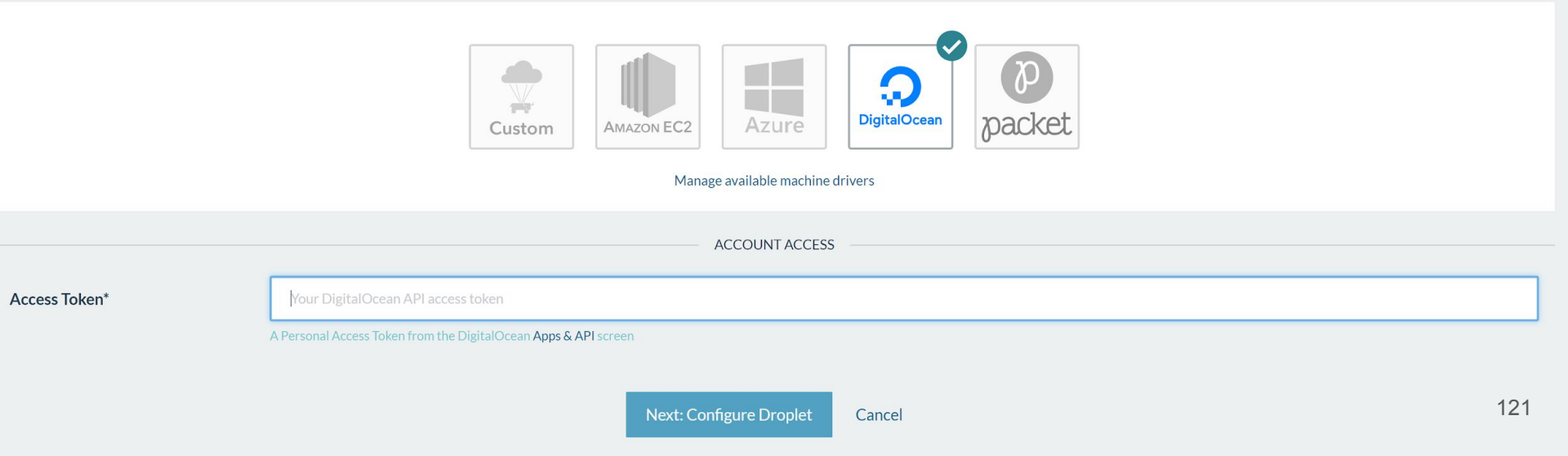
Rancher first time

3. Service



2. Add new server (worker)

1. Get access_token
2. Click Click Click



Demo: rancher

“ Create new server “

Workshop: rancher III

“ Create new 2 server for worker “

Manual run rancher client

crontab -e

@reboot sudo docker run --rm --privileged -v /var/run/docker.sock:/var/run/docker.sock -v /var/lib/rancher:/var/lib/rancher rancher/agent:v1.2.10 http://111.222.333.444:8080/v1/scripts/ABCDDFDFDFEFEFEFEFEFBE0DDCD65C:151312312384000000:jhYGgwgwgb5Jh1pKeB9G7KkEhi0g

Demo: rancher l

“ Deploy example app “

Demo: rancher I

- image: nginx
- port 80

Workshop: rancher IV

“ Deploy your own image with rancher “

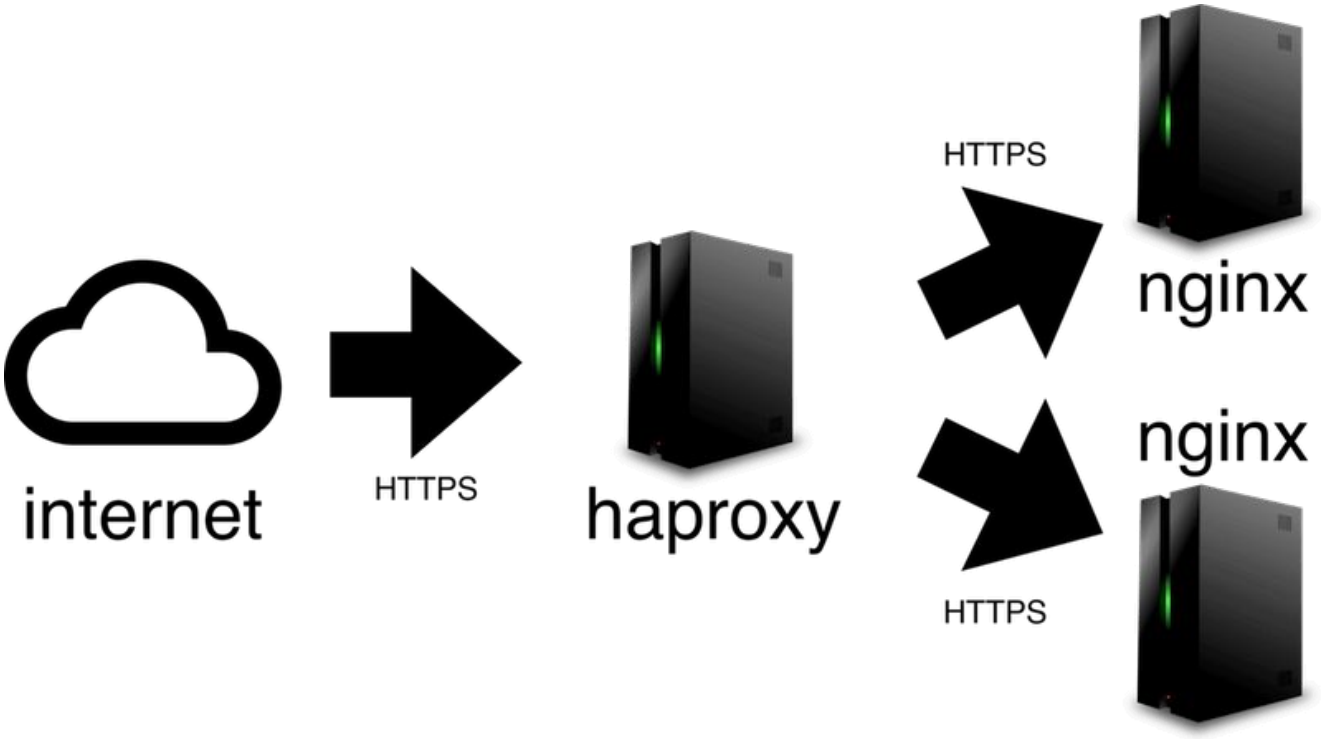
Workshop: rancher V

“ Deploy your own docker-compose with rancher “

Workshop: rancher VI

“ Deploy Mutil Containner“

Load Balancing with Haproxy

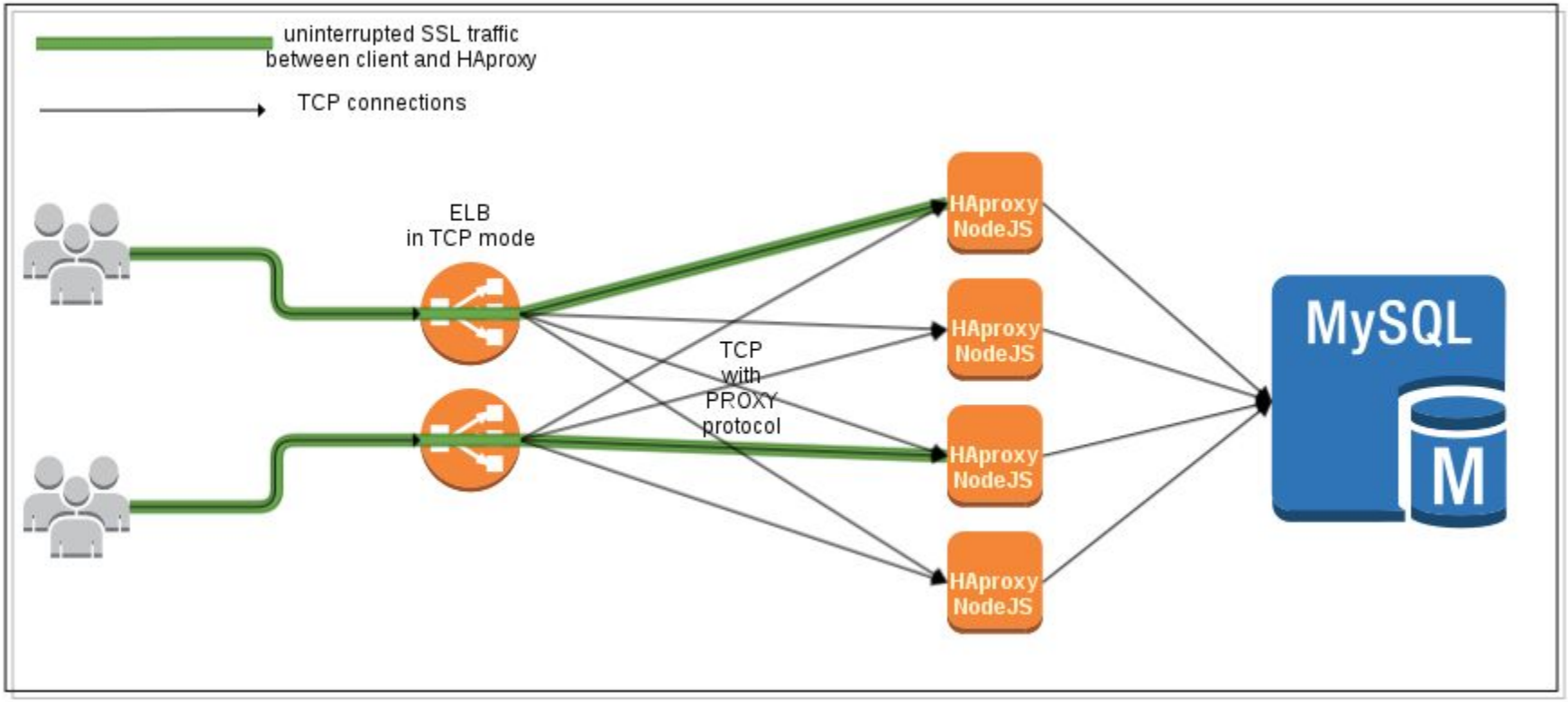


Demo: Load Balancing

“ Create load balancing for example “

Workshop: Load Balancing

“ Create load balancing for your app “


uninterrupted SSL traffic
between client and HAproxy
TCP connections
ELB
in TCP mode
TCP
with
PROXY
protocol
HAproxy
NodeJS
HAproxy
NodeJS
HAproxy
NodeJS
HAproxy
NodeJS
MySQL
M

4 Steps for make your pipeline

1. Prepare your Rancher server
2. Add new server (worker)
3. Dockerfile
4. Circleci config

Break

circleci

Create a Pipeline for Deploying

circleci configuration

```yaml
version: 2
jobs:
  build:
    docker:
      - image: circleci/node:8.11.1-stretch
    working_directory: ~/repo
    branches:
      only:
        - master
        - staging
    steps:
      - checkout
```

circleci configuration

```
version: 2
jobs:
  build:
    docker:
      - image: circleci/node
    working_directory: ~/rep
    branches:
      only:
        - master
        - staging
    steps:
      - checkout
```

Build: Job name

- At least it should be job name ‘build’

docker:

image:

- tell circle to pull docker image for this ENV.

branches:

- Select branches to action/build

steps:

- what you want circleci to do

circleci configuration

```
steps:
  - checkout
  - run: yarn install
  - setup_remote_docker
  - run: docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
  - run:
      name: "Build docker image and push to docker hub"
      command: |
        docker build -t ima8/codemania111:latest .
        docker push ima8/codemania111:latest
```

Demo: Circleci

“ Overview Circleci and import project “

Authorize CircleCI

CircleCI by **circleci**
wants to access your **gasileer** account

Personal user data
Email addresses (read-only)

Repositories
Public and private

Authorize circleci

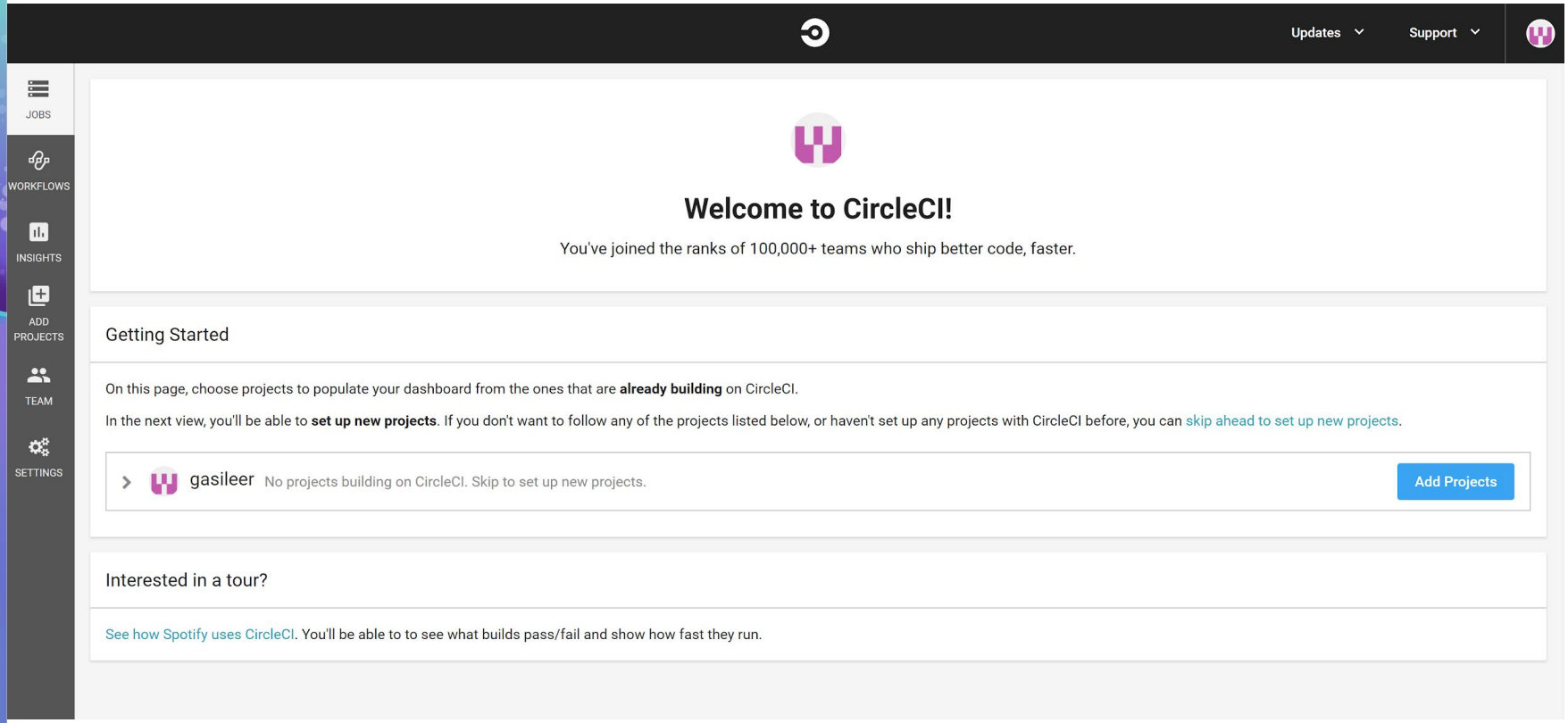
Authorizing will redirect to
https://circleci.com

Not owned or operated by GitHub

Created **7 years ago**

More than 1K GitHub users

Learn more about OAuth


Updates
Support
JOBS
WORKFLOWS
INSIGHTS
ADD PROJECTS
TEAM
SETTINGS
Welcome to CircleCI!
You've joined the ranks of 100,000+ teams who ship better code, faster.
Getting Started
On this page, choose projects to populate your dashboard from the ones that are already building on CircleCI.
In the next view, you'll be able to set up new projects. If you don't want to follow any of the projects listed below, or haven't set up any projects with CircleCI before, you can skip ahead to set up new projects.
gasileer
No projects building on CircleCI. Skip to set up new projects.
Add Projects
Interested in a tour?
See how Spotify uses CircleCI. You'll be able to to see what builds pass/fail and show how fast they run.

Demo: Circleci II

“ Run the first build: just npm install “

Workshop: Circleci I

“ Link your github and Run the first build“

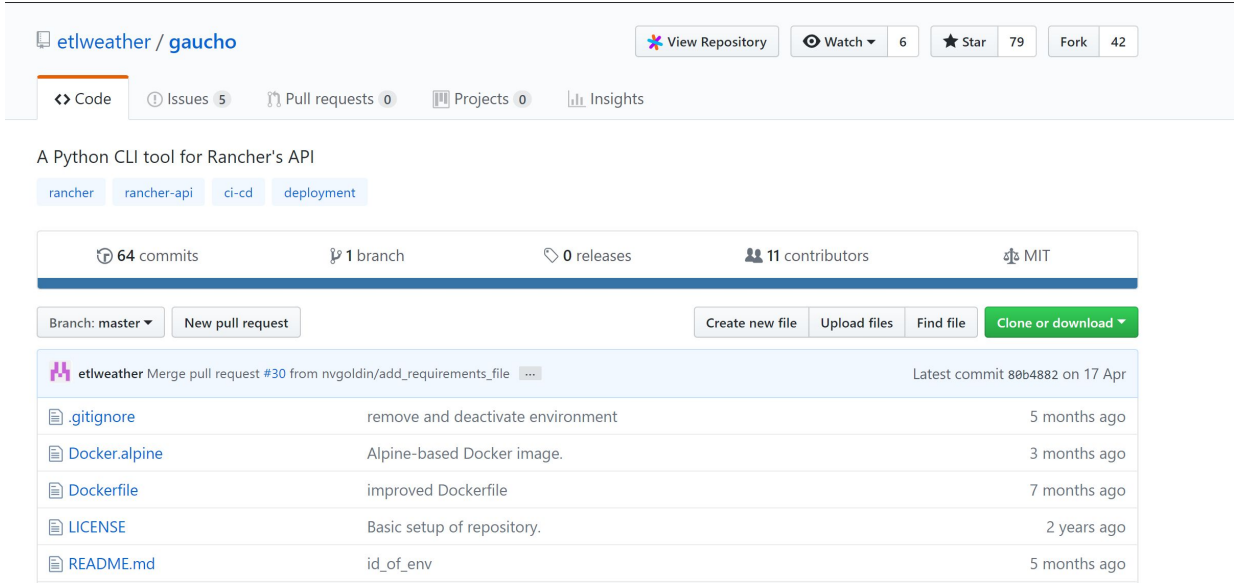
workshop: Circleci II

“ Build docker and push to docker hub “

```yaml
version: 2
jobs:
  build:
    docker:
      # specify the version you desire here
      - image: circleci/node:8.11.1-stretch
    working_directory: ~/repo
    steps:
      - checkout
      - restore_cache:
          keys:
            - v1-dependencies-{{ checksum "package.json" }}
            # fallback to using the latest cache if no exact match is found
            - v1-dependencies-
      - run: npm install
      - save_cache:
          paths:
            - node_modules
          key: v1-dependencies-{{ checksum "package.json" }}
      - run: npm test
      - setup_remote_docker
      - run: docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
      - run:
          name: "Build docker image and push to docker hub"
          command: |
              cp Dockerfile.production Dockerfile
              docker build -t ima8/example-nodejs-circle:latest .
              docker build -t ima8/example-nodejs-circle:${CIRCLE_SHA1} .
              docker push ima8/example-nodejs-circle:latest
              docker push ima8/example-nodejs-circle:${CIRCLE_SHA1}
      - run: echo Done
```

circleci configuration

https://github.com/etlweather/gaucho



circleci configuration

```
- run:
    name: "Call to rancher to deploy"
    command: |
        docker run --rm -it \
          -e CATTLE_ACCESS_KEY="$CATTLE_ACCESS_KEY" \
          -e CATTLE_SECRET_KEY="$CATTLE_SECRET_KEY" \
          -e CATTLE_URL="$CATTLE_URL" \
          etlweather/gaucho upgrade $RANCHER_EXAMPLE_NODEJS \
          --imageUuid 'docker:ima8/example-nodejs-circle:latest' \
          --batch_size 3 --start_first \
          --auto_complete --timeout 600 \
          /
```

Workshop: Circleci III

“ Deploy your app to rancher“

circleci configuration: cache

```
steps:
  - checkout
  - restore_cache:
      keys:
        - v1-dependencies-{{ checksum "package.json" }}
        - v1-dependencies-
  - run: cd example_worker
  - run: yarn install
  - save_cache:
      paths:
        - node_modules
      key: v1-dependencies-{{ checksum "package.json" }}
```

Workshop: Circleci IV

“ Deploy your app to rancher with cache“

circleci with production and staging

```
- setup_remote_docker
- run: docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
- run:
    name: "Build docker image and push to docker hub"
    command: |
      if [ "${CIRCLE_BRANCH}" == "master" ]; then
        docker build -t ima8/code_mania_demo:latest .
        docker build -t ima8/code_mania_demo:${CIRCLE_SHA1} .
        docker push ima8/code_mania_demo:latest
        docker push ima8/code_mania_demo:${CIRCLE_SHA1}
      elif [ "${CIRCLE_BRANCH}" == "staging" ]; then
        docker build -t ima8/code_mania_demo_staging:latest .
        docker push ima8/code_mania_demo_staging:latest
      else
        echo "This is ${CIRCLE_BRANCH}"
      fi
```

Workshop: Circleci V

“ Deploy your app to rancher with selected branch“

Workshop: Circleci VI

“ Add run test before deploy ”