VIETNAM WEB SUMMIT

bizweb
Bán hàng dễ dàng hơn

Building Bizweb Microservices with Docker

Nguyễn Minh Khôi
CTO of DKT Technology
dkt.com.vn

Bizweb Tech Stack

Programming Languages

C#

Frameworks & Libraries

.NET MVC

spring

NETFLIX OSS

Databases

Microsoft SQL Server

redis

mongoDB

Others

elastic

docker

Message Queues

kafka

RabbitMQ™

Cloud Services

amazon webservices™ | S3, EC2, Route53

Web Servers

NGINX

IIS

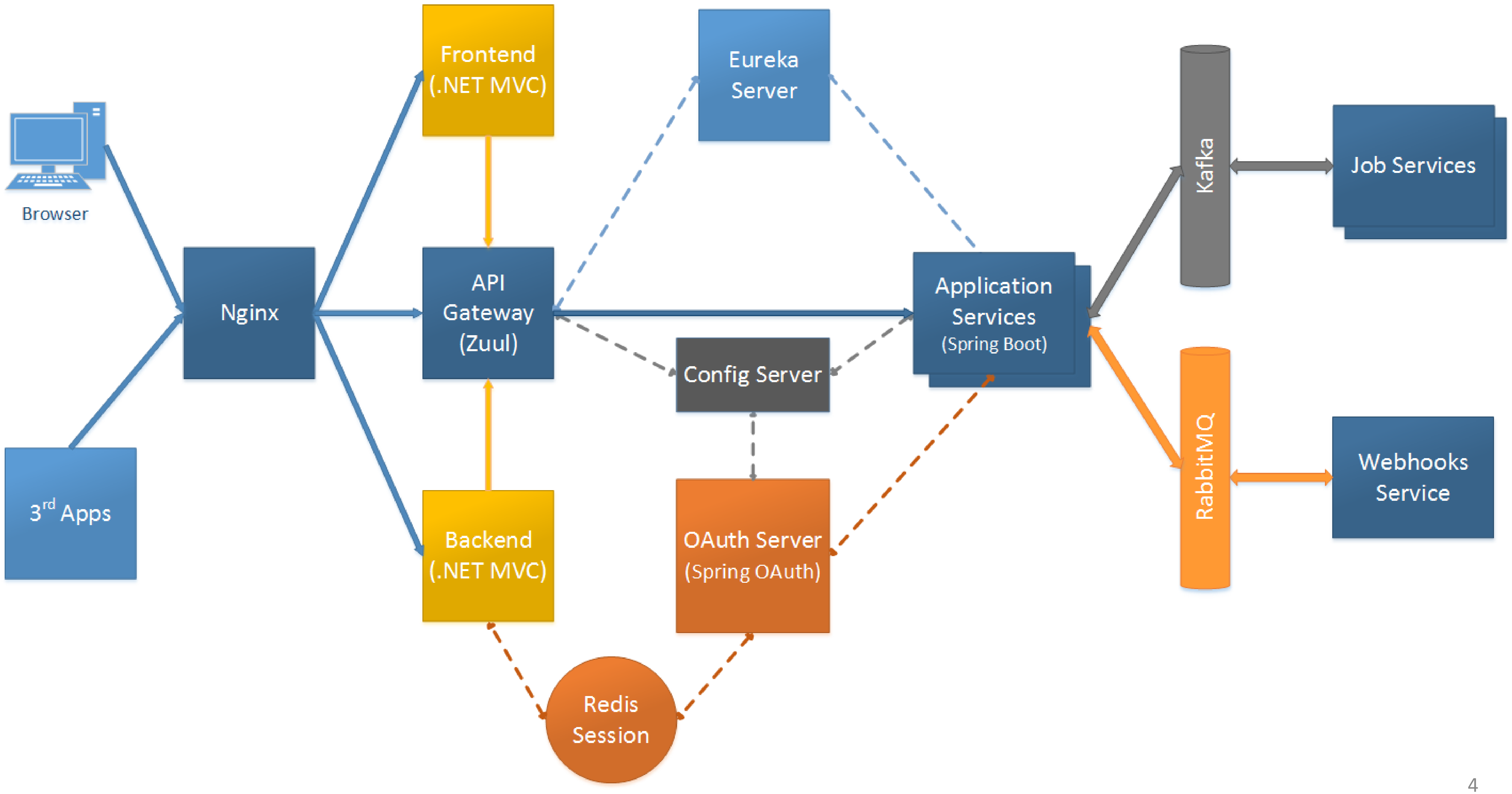
jetty://

traffic server™

Bizweb Microservices Components

- API Gateway: Zuul
- Service Discovery: Eureka (Server), Ribbon (Client)
- Centralized Configuration: Spring Cloud Config
- API Security: Spring Security & Spring Security OAuth
- REST API: Spring Boot
- Job Service: Kafka & Spring Boot

Bizweb Microservices Architecture



Problems

- Take times to deploy on new servers:
  - Install Java
  - Copy fat .jar file (~75-100MB) using FTP/SCP
  - Make script to run as a Linux service
- Take times to update services:
  - 20 microservices + job services
  - Manual update on multiple hosts
  - Manual scale & choose server to deploy
- Quite hard to monitor these microservices

Solved with Docker & Jenkins

Simple Dockerfile for all services

```
FROM frolvlad/alpine-oraclejdk8:slim
ADD lib lib
ADD product.jar app.jar
RUN sh -c 'touch /app.jar'
ENTRYPOINT ["java","-Xmx128m","-Xms128m","-
Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

Docker Swarm Mode

- Built-In Orchestration
- Easy to start

```
docker swarm init
docker services create --name product product:1.2.0
```

- Secure by default

```
docker swarm join --token [manager_token|worker_token]
```

- Easy to scale

```
docker service scale product=10
```

- Rolling updates

```
docker service update --update-delay 1m --update-parallelism 2
--image product:1.2.1 product
```

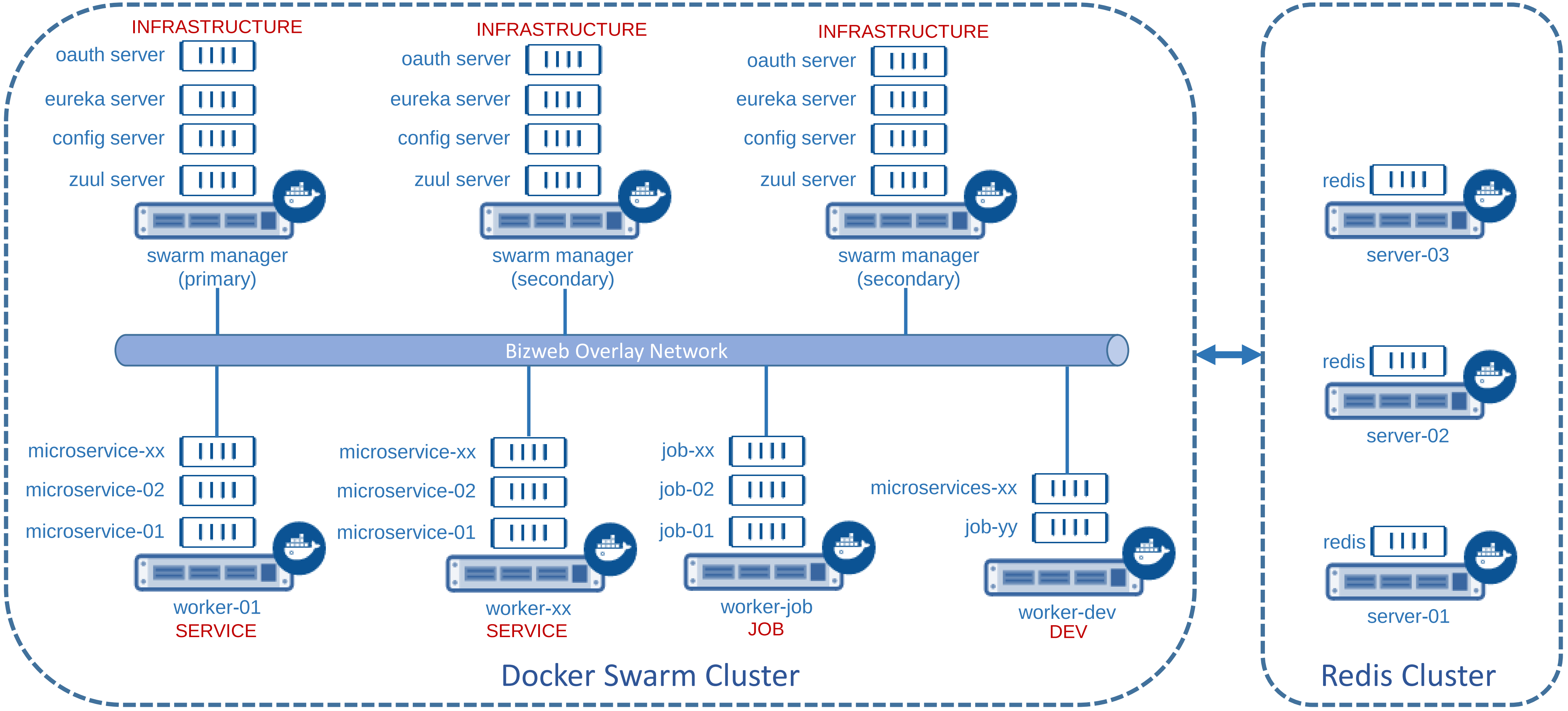
Combine Netflix OSS with Docker Swarm

- Zuul for API Gateway
  - Only handle requests from outside
- Eureka for Service Discovery
- Ribbon Client for direct call between microservices:
  - Client load balancer
  - Caching (reduce request to Eureka)
- Docker Swarm:
  - Manage microservices
  - Deploy, scale, update microservices

Docker Swarm Deployment

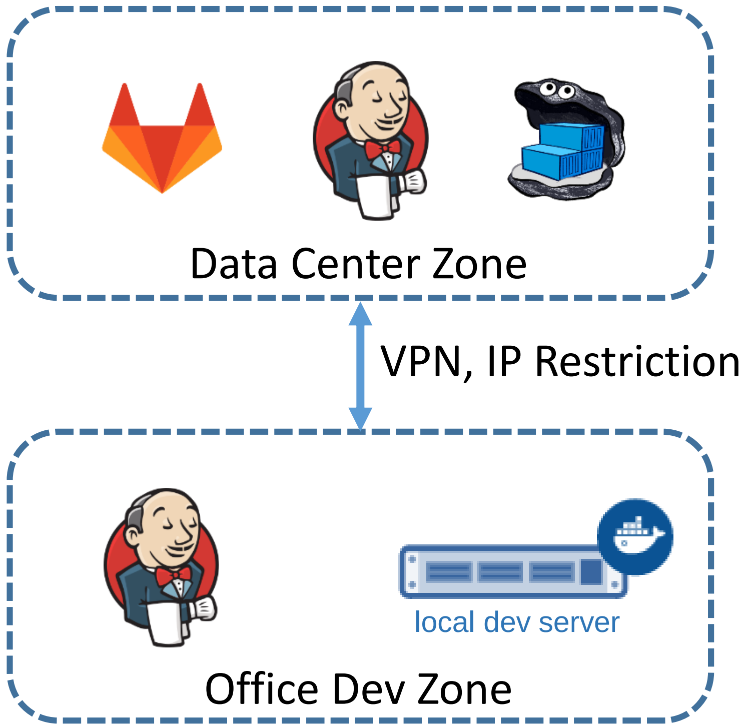
- 3 manager nodes on 3 different physical machines
- Workers on Physical & Virtual Machines
- Using overlay network:
  - Communicate with Eureka Server
  - Direct call between microservices
- Label for services & environment:
  - INFRASTRUCTURE: running Zuul, Eureka, Config, OAuth Service
  - SERVICE: running microservices
  - JOB: running Job services
  - DEV: running all containers of dev environment

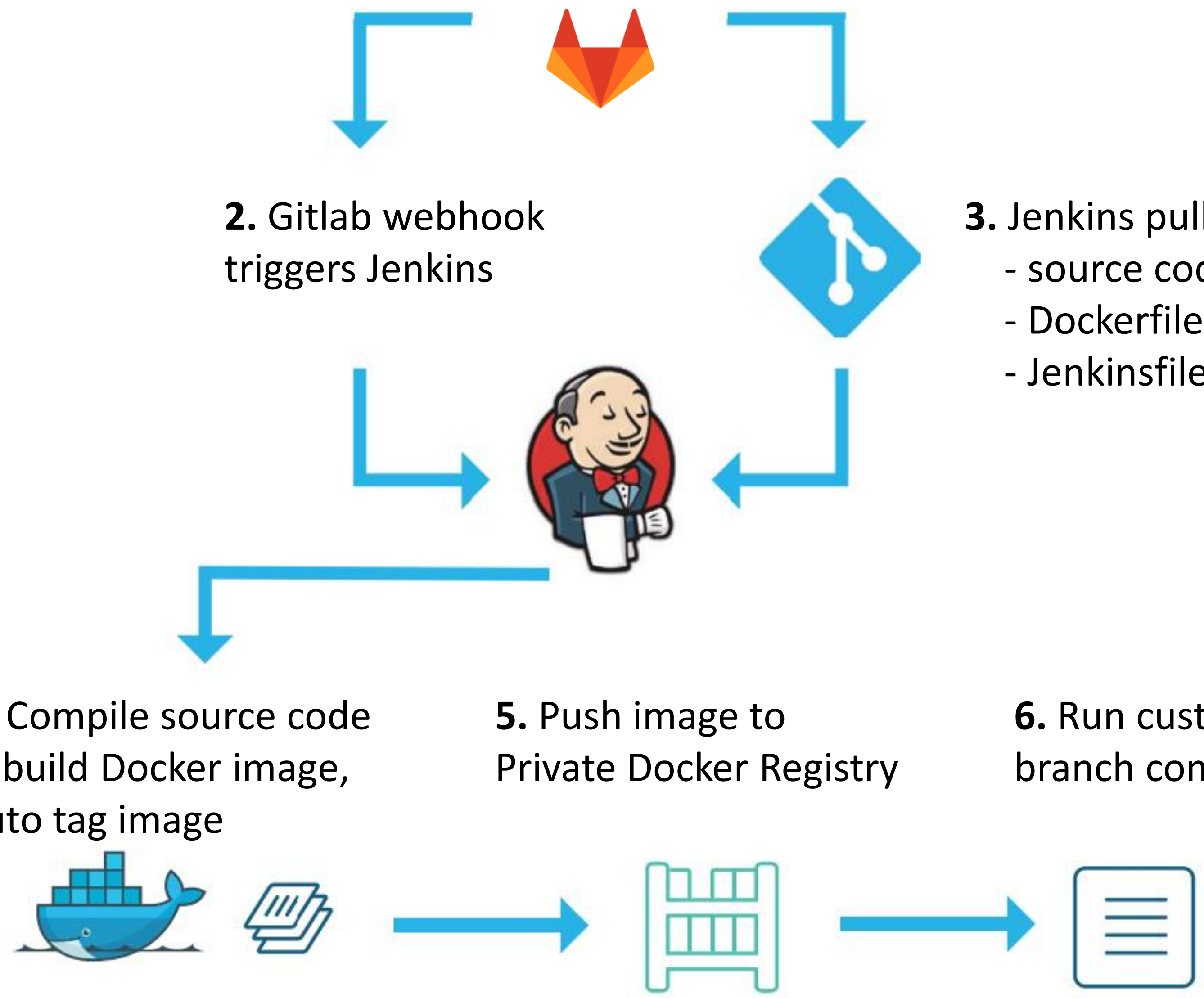
Docker Swarm Cluster



CI with Jenkins & Docker

- Environment Prerequisites:
  - Gitlab 8.12 (support webhooks)
  - Jenkins 2
  - Docker Registry 2.0
- Run on Docker





1. Developer pushes to Gitlab repo

2. Gitlab webhook triggers Jenkins

3. Jenkins pull code from Gitlab:
- source code
- Dockerfile
- Jenkinsfile

4. Compile source code & build Docker image, auto tag image

5. Push image to Private Docker Registry

6. Run custom script based on branch committed

CI with Jenkins & Docker

- Using Spotify docker-maven-plugin:
  - Save space by caching java libraries image layer (~75MB)
  - Reduce network traffic & deploy time (only 200-700KB transferred)
- Docker image auto tag:

  `{git_commit_short_code}-{branch}` -> `4b4a71ef-dev`
- Custom script based on branch committed:
  - dev: trigger another Jenkins Server to update service
  - live: manual update

References

- http://www.slideshare.net/HanoiItlc/itlc-hn-14-bizweb-microservices-architecture
- https://docs.docker.com/engine/swarm/
- https://www.docker.com/use-cases/cicd

Contact

- Nguyễn Minh Khôi – DKT Technology
- Email: khoinm@dkt.com.vn
- Facebook: https://fb.com/khoinguyen84

Thank you!
Q&A

VIETNAM
WEB
SUMMIT