Cloud Native and Kubernetes Yogyakarta Meetup #2
Saturday, 29 Feb 2020 @ Bima Hacker's Village Yogyakarta

Running Cloud-Native Applications on Kubernetes

Akhyar A.
DevOps Engineer
gojek GO MERCHANTS

HELLO WORLD!

I'm **Akhyar Amarullah**, some people call me **Akhy**

Currently a **DevOps Engineer** at **Gojek** in **GoMerchants Automation Team**

What I do at work:

- Continuous whatever (integration/delivery/etc)
- Write Python, Go, and sometimes shell scripts
- Manage Kubernetes clusters
- Make both devs and ops happy with toolings and automation

In the past, I've also been professionally a...
QA Engineer, Mobile App Engineer (Android), and Web Developer
~~after giving up trying to be a Graphics/Web Designer and Game Developer~~

Find @akhy on GitHub and Twitter

What is “Cloud Native” app?

What is Cloud Native app?

“Cloud-native applications are a collection of small, independent, and loosely coupled services. They are designed to deliver well recognized business value, like the ability to rapidly incorporate user feedback for continuous improvement.”
-- RedHat

“Cloud-native is an approach to building and running applications that exploits the advantages of the cloud computing delivery model. Cloud-native is about how applications are created and deployed, not where.”
-- Pivotal

Common problems

- New features took months to be released to customers
- “It works in my machine!”
  - Dev forgets to FTP upload a new PHP library/helper file he/she pasted from StackOverflow
  - The server isn’t installed with PHP-GD extension yet
  - ...
- The web page is very slooooow to open when there are a lot of users accessing it
- An unhelpful customer reported that his/her payment failed, but the developer doesn’t have any idea what exactly happened
- A developer accidentally pushed “test test123” message to users
- Website visitors see “Failed to connect to ‘cobacoba’ DB on localhost”
- etc.

Meet the “Twelve Factor App”

https://12factor.net

The Twelve-Factor Principles

1. **Codebase**: One codebase tracked in revision control, many deploys
2. **Dependencies**: Explicitly declare and isolate dependencies
3. **Config**: Store config in the environment
4. **Backing services**: Treat backing services as attached resources
5. **Build, release, run**: Strictly separate build and run stages
6. **Processes**: Execute the app as one or more stateless processes
7. **Port binding**: Export services via port binding
8. **Concurrency**: Scale out via the process model
9. **Disposability:** Maximize robustness with fast startup and graceful shutdown
10. **Dev/prod parity**: Keep development, staging, and production as similar as possible
11. **Logs**: Treat logs as event streams
12. **Admin processes**: Run admin/management tasks as one-off processes

See the full explanations on: https://12factor.net

Deploying to Kubernetes

Kubernetes makes it easy to implement and run cloud native 12-factor apps

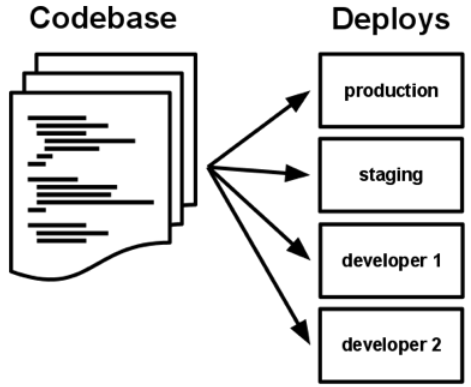
Release =
Built Code + Config

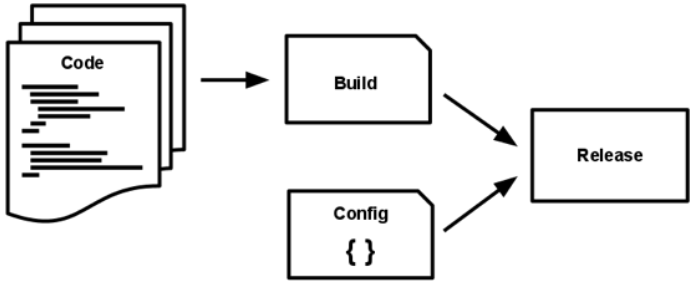
Explicitly declare and isolate dependencies (#2)
→ Lock your dependencies with package managers

One codebase tracked in revision control, many deploys (#1)

Keep development, staging, and production as similar as possible (#10)

Strictly separate build and run stages (#5)





How does Kubernetes help?

- It forces you to use Docker (or other container technologies)
- It doesn’t care what language your apps written in, as long as they’re containerized, it can run them in the same way
- Its “Deployment” helps you to adopt stateless model
- Containerization makes disposability easy
- It’s fast to scale up and down as traffic changes
- Flexible, a lot of toolings around it allows us to adopt any workflow that suit organization style/culture
- Efficiently use available resources by automatically scheduling pods to nodes
- Easy to use centralized logs

Dockerfile

```
FROM python:3.5
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r
requirements.txt
COPY . ./
CMD ["python", "helloworld.py"]
```

requirements.txt

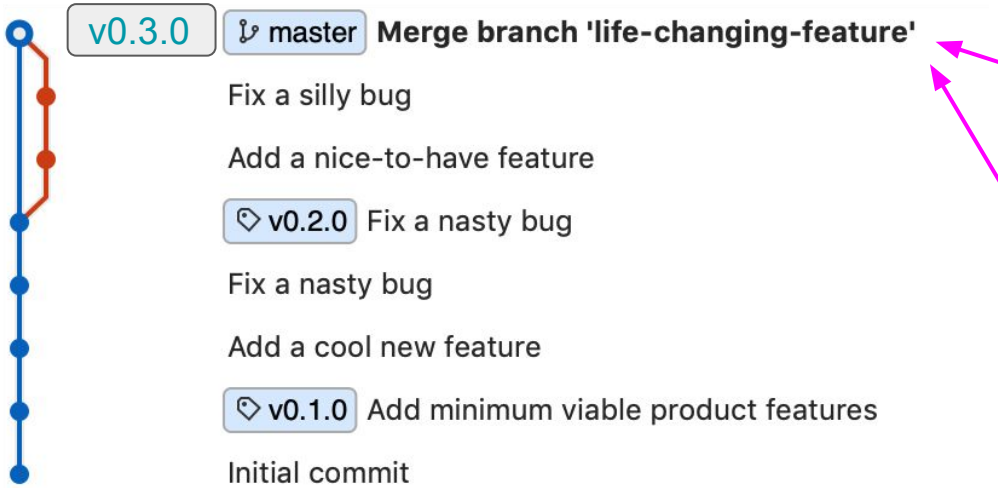
```
Flask==1.1.1
python-dotenv==0.10.3
python-ldap==3.2.0
prometheus_client==0.7.1
```

Track and Deploy

STAGING (helloworld-stg.env)

PRODUCTION (helloworld-prod.env)

Promote to PROD



STAGING (helloworld-stg.env)

PRODUCTION (helloworld-prod.env)

Environment Files

 # helloworld-stg.env

```
DB_URL=mysql://db-stg:3306/helloworld
DB_USERNAME=staging-user
LOG_LEVEL=debug
LOG_FORMAT=json
NEW_RELIC_APP_NAME=helloworld-stg
NEW_RELIC_APP_ENABLED=true
```

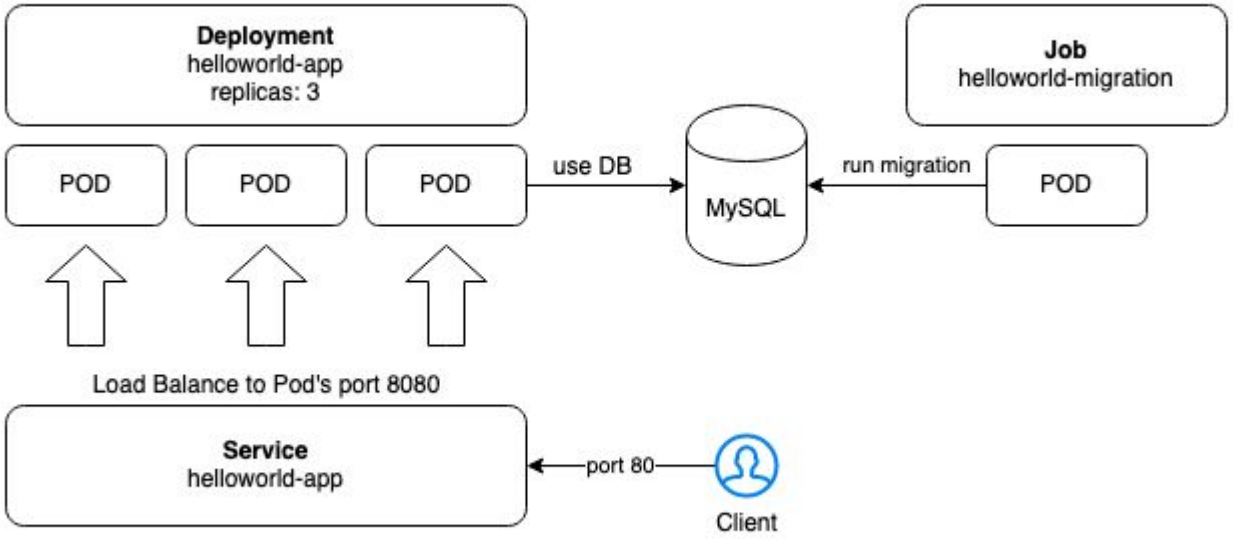
 # helloworld-prd.env

```
DB_URL=mysql://db-prd:3306/helloworld
DB_USERNAME=prod-user
LOG_LEVEL=info
LOG_FORMAT=json
NEW_RELIC_APP_NAME=helloworld
NEW_RELIC_APP_ENABLED=true
```

 # local.env

```
DB_URL=mysql://localhost:3066/helloworld-db-final-revisi-terakhir-edit-3
DB_USERNAME=akhy
LOG_LEVEL=debug
LOG_FORMAT=text
NEW_RELIC_APP_ENABLED=false
```

Overview

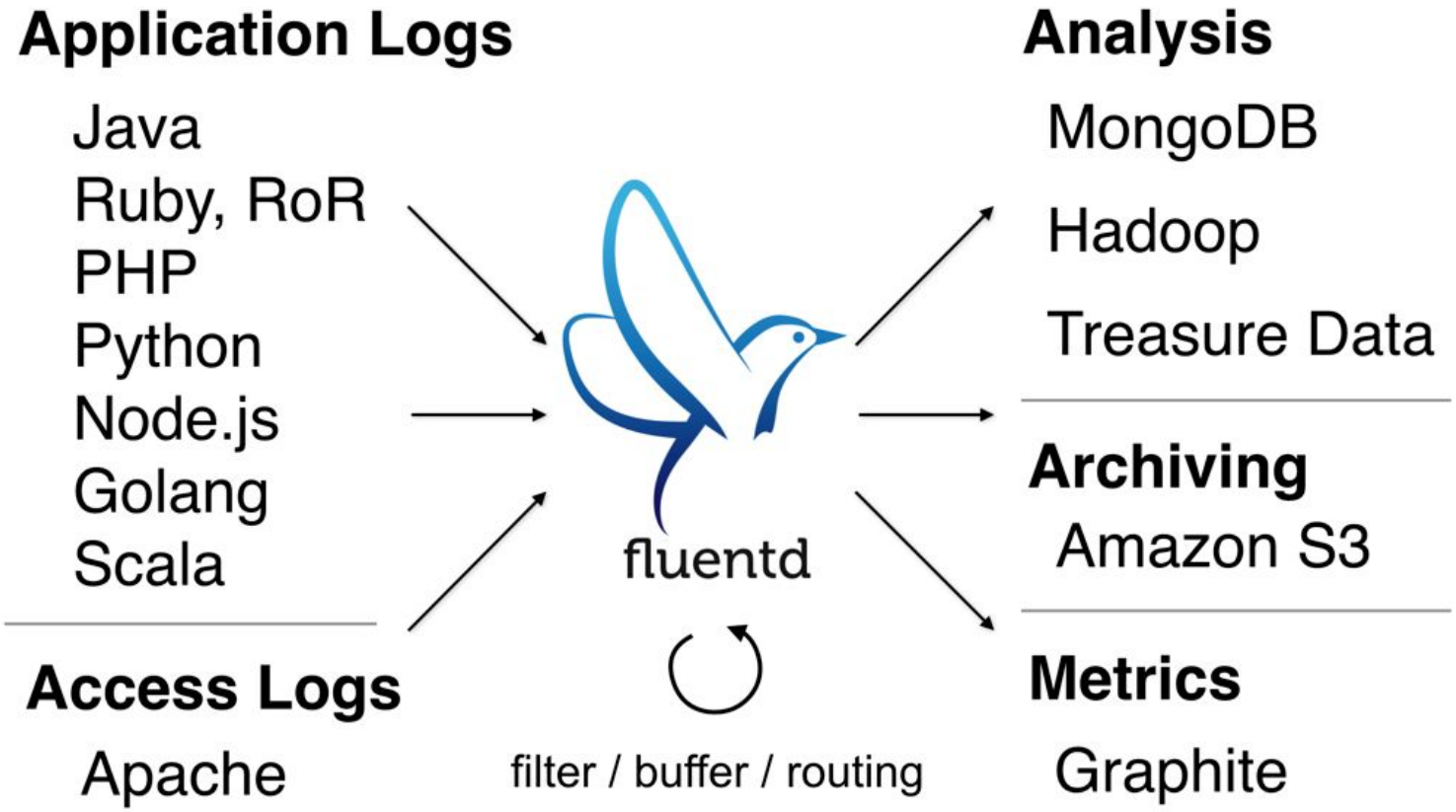

Deployment
helloworld-app
replicas: 3
Job
helloworld-migration
POD
POD
POD
use DB
MySQL
run migration
POD
Load Balance to Pod's port 8080
Service
helloworld-app
port 80
Client

> kubectl apply

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
 name: helloworld
spec:
 replicas: 3
 template:
   spec:
     containers:
     - name: helloworld
       image: akhy/helloworld:v0.2.0
       command:
         - helloworld
         - --port=8080
       envFrom:
         configMapRef:
           name: helloworld-stg
       ports:
         - name: http
           containerPort: 8080
       livenessProbe:
         httpGet:
           path: /ping
           port: http
```

```yaml
apiVersion: v1
kind: Service
metadata:
 name: helloworld
spec:
 selector:
   app: helloworld
 ports:
   - name: http
     port: 80
     targetPort: http
```

Centralized Logging



Tips on migrating your app to run on Kubernetes

Ensure it’s stateless:

- Don’t write logs to files → Write to STDOUT, it’s a universal interface ;)
- Don’t write session/cookie to files → Use external database/redis
- Don’t write states to files → Use external database

Expose configurations via environment variables, DON’T HARDCODE THEM

Have a “/ping” or “/health” endpoint for healthcheck/probe

References and learning resources

https://kubernetes.io

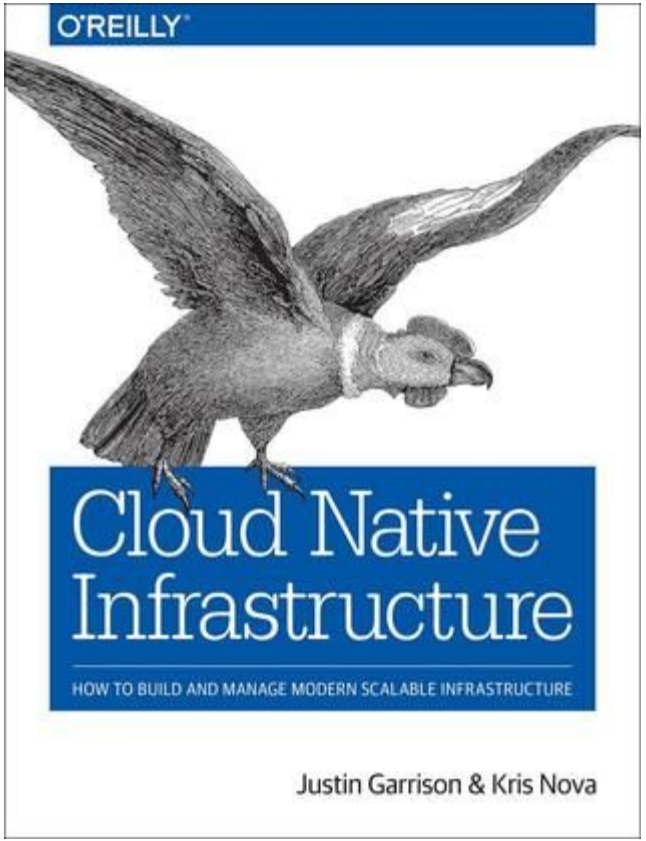
https://12factor.net

https://www.redhat.com/en/topics/cloud-native-apps

https://pivotal.io/cloud-native

https://medium.com/ibm-cloud/kubernetes-12-factor-apps-555a9a308caf

https://ramitsurana.github.io/awesome-kubernetes



Thank You!

https://github.com/**akhy**
https://id.linkedin.com/in/**akhyar**