# Deployment to Kubernetes in Bukalapak

Kubernetes and Cloud Native Bandung Meetup #3

Lo.Ka.Si Coffee & Space

16 November 2019

# Muhammad Saiful Islam

**Cloud Engineer @ Bukalapak, 2019 — now**

Site Reliability Engineer @ Bukalapak, 2018 — 2019

Software Engineering Intern @ Bukalapak, 2017

S.Tr.Kom. from Politeknik Negeri Bandung, 2014 — 2018


## Introduction to Kubernetes

**Giri Kuncoro**, 13 September 2019


## Migration Journey to Kubernetes

**Didiet Agus Pambudiono**, 24 October 2019

both are accessible from
https://github.com/cloudnative-id/kubernetes-cloud-native-bandung

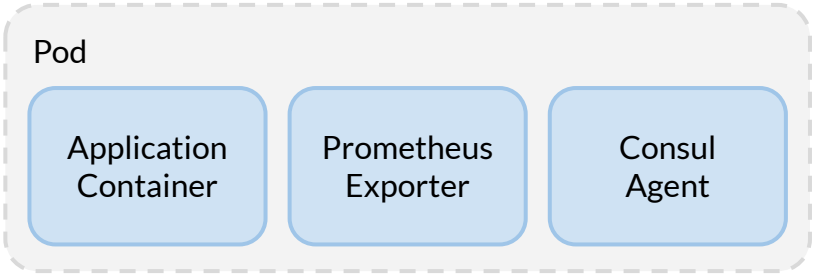
# A quick **recap**...

# Kubernetes
# Pod

Small groups of
containers and volumes

Tight coupling —
live and die together

Shared networking
between containers



# Kubernetes Deployment

**Declarative updates** for pods

Specify a **desired state**,
controller handles the rest

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lulu-api
spec:
  replicas: 2
  template:
    metadata:
      labels:
        project: lulu
        service: api
    spec:
      containers:
      - name: lulu-api
        ports:
        - containerPort: 8080
        resources:
          limits:
            memory: 2Gi
            cpu: 2
          requests:
            memory: 0.5Gi
            cpu: 1
```

# Kubernetes Service

Exposes a group of pods into a network service

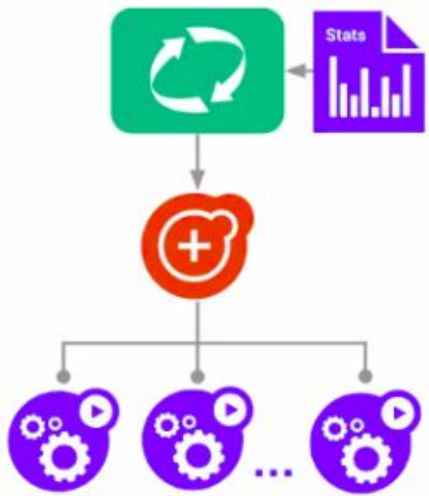
Provides load balancing and a DNS name to be accessed by other workloads

```
apiVersion: v1
kind: Service
metadata:
  name: lulu-api
spec:
  selector:
    project: lulu
    service: api
  ports:
  - protocol: TCP
    port: 8080
  type: ClusterIP
```

# Horizontal Pod Autoscaler

Automatically add (or remove) pods when needed

Based on CPU/memory metrics with custom metrics support



**>70**

**Ruby** services

**>150**

**Go** services

**>150**

**Java/NodeJS/Python/Elixir/**

other services

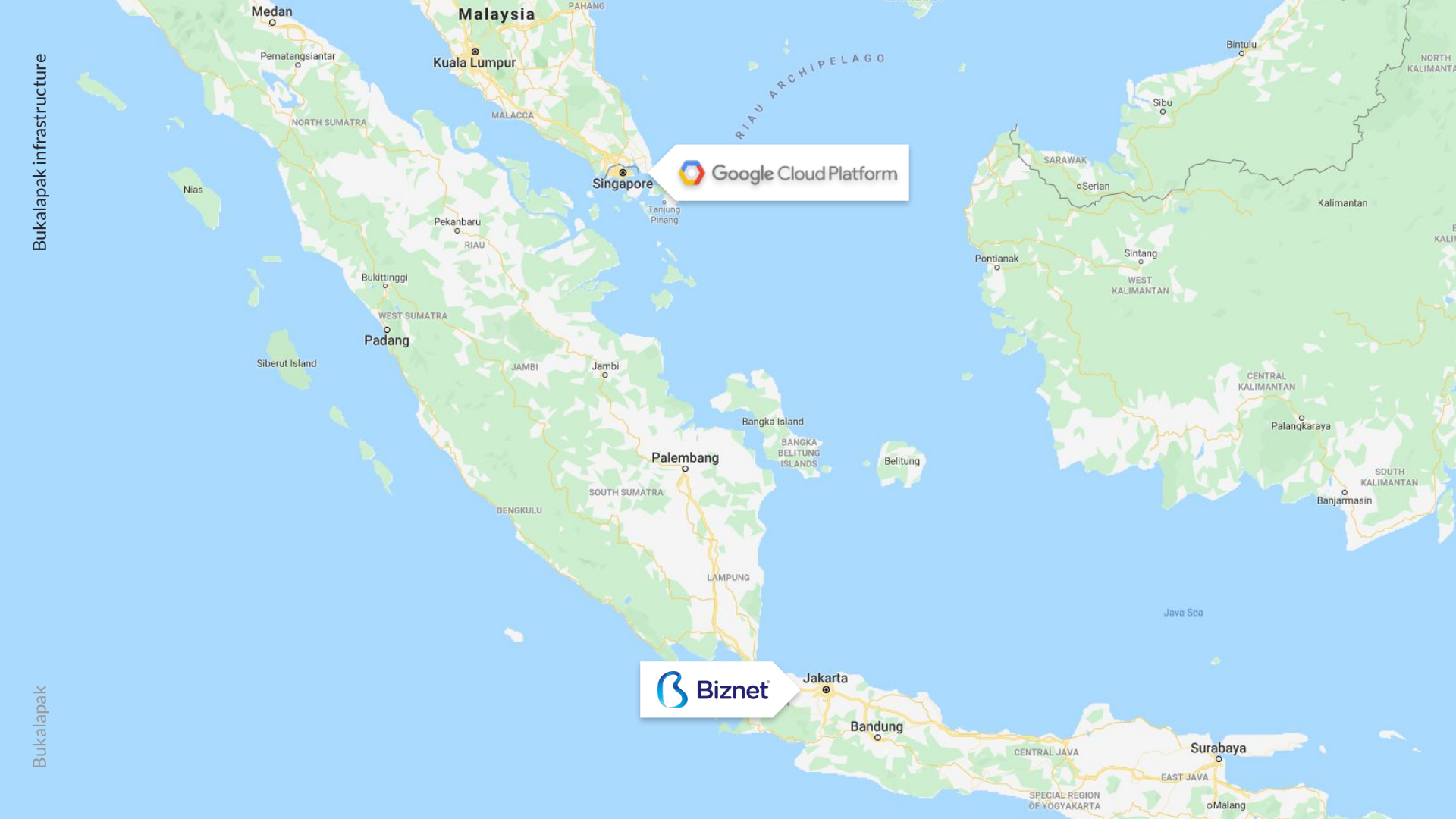
**>1.700**

deployments

**>20.000**

pods running

Bukalapak infrastructure

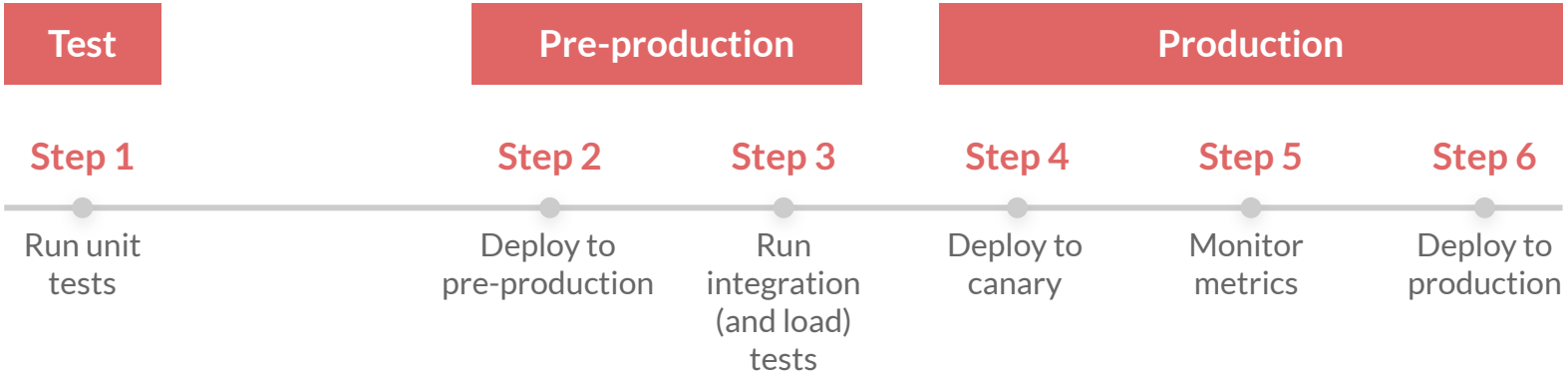


So, how do we **deploy**
our microservices to our clusters?

Test

Pre-production

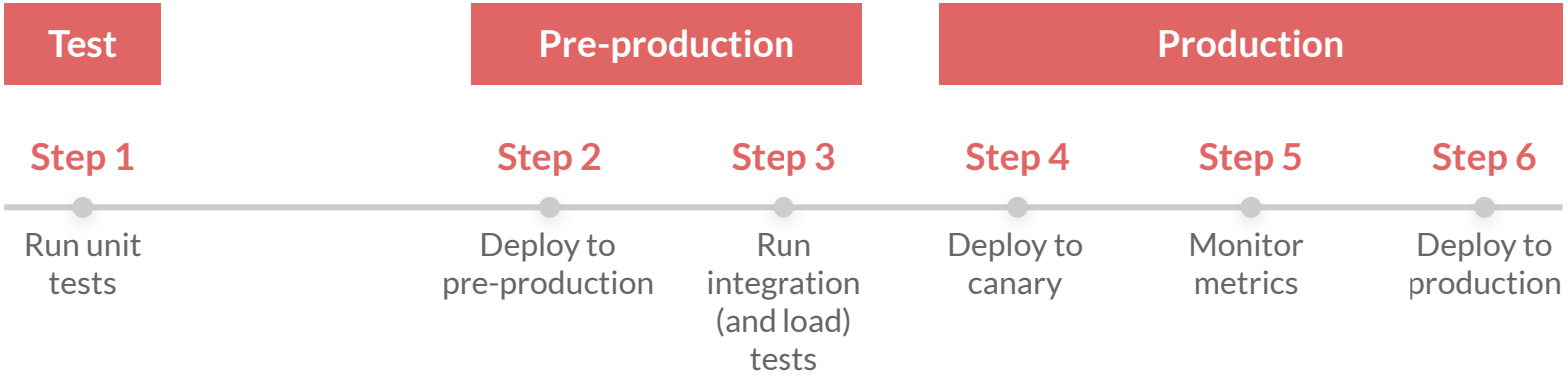
Production


Test
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Run unit tests
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production

```
stages:
- test

- before-build-preproduction
- build-preproduction
- before-deploy-preproduction
- deploy-preproduction
- after-deploy-preproduction

- before-build-canary
- build-canary
- before-deploy-canary
- deploy-canary

- before-deploy-production
- deploy-production
```

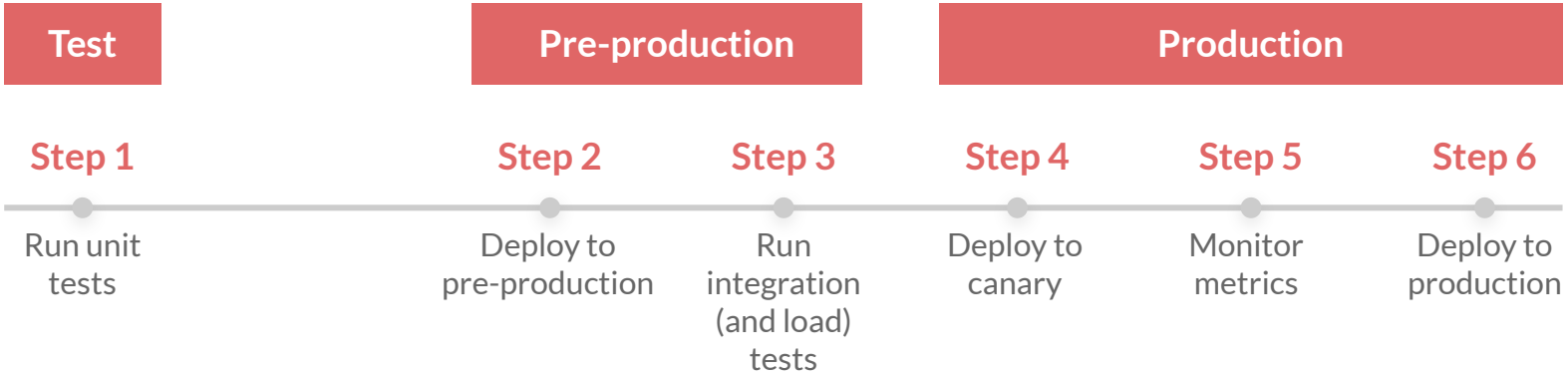
```
test:
  stage: test
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  script:
  - make test
  - GIT_BRANCH=$CI_COMMIT_REF_NAME goveralls -coverprofile=coverage.out
-endpoint=$COVERALLS_ENDPOINT

lint:
  stage: test
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  script:
  - curl -sfL https://install.goreleaser.com/github.com/golangci/golangci-lint.sh | sh -s --
-b $(go env GOPATH)/bin v1.16.0
  - golangci-lint --version
  - golangci-lint run
```

```yaml
compile-preproduction:
  stage: before-build-preproduction
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  only:
    refs:
    - master
    - /^release.+/
  script:
  - make compile
  artifacts:
    paths:
    - deploy/_output

build-image-preproduction:
  stage: build-preproduction
  image: $CI_REGISTRY/bukalapak/gudang/docker:1.0.0
  only:
    refs:
    - master
    - /^release.+/
  script:
  - export VERSION=${CI_COMMIT_TAG:-$CI_COMMIT_SHA}-preproduction
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" "$CI_REGISTRY"
  - make build-image push
```

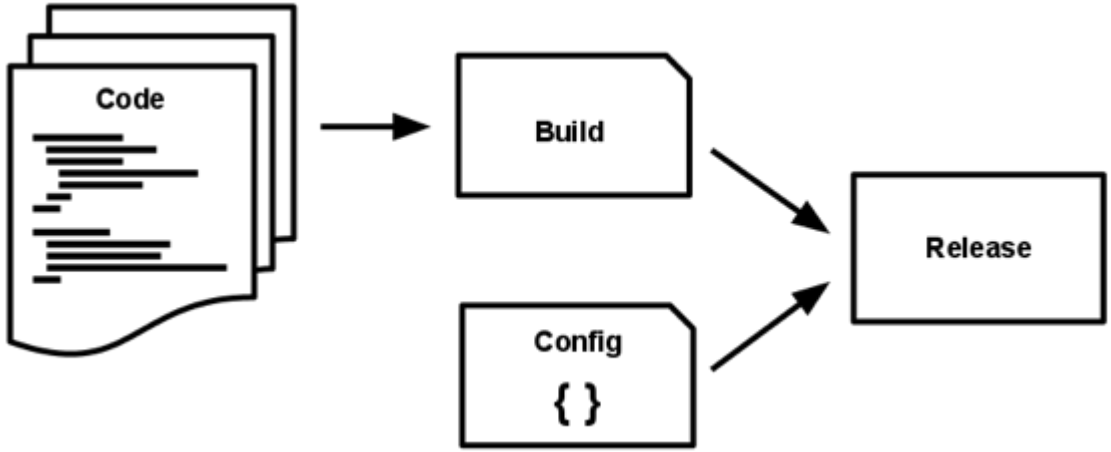

Test
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Run unit tests
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production

**Database**
Host: `10.80.47.5`
User: `lulu_rw`
Pass: `-zH_FsinJx38KLyy`

**Database**
Host: `10.162.4.93`
User: `lulu_rw`
Pass: `RpiGmD*om2jqAjJq`



Should we build it
**for each environment**?




https://12factor.net/config

```
// Copyright 2017, Google, Inc.
// Licensed under the Apache License, Version 2.0 (the "License")
var http = require('http');
var server = http.createServer(function (request, response) {
  const language = 'English';
  const API_KEY = '123-456-789';
  response.write(`Language: ${language}\n`);
  response.write(`API Key: ${API_KEY}\n`);
  response.end(`\n`);
});
server.listen(3000);
```

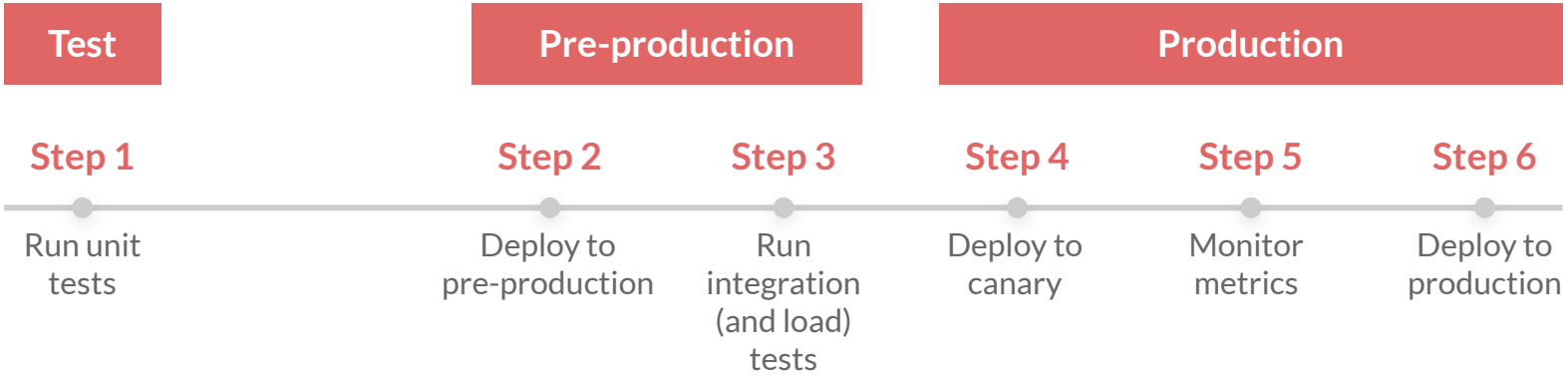
https://medium.com/google-cloud/kubernetes-configmaps-and-secrets-68d061f7ab5b

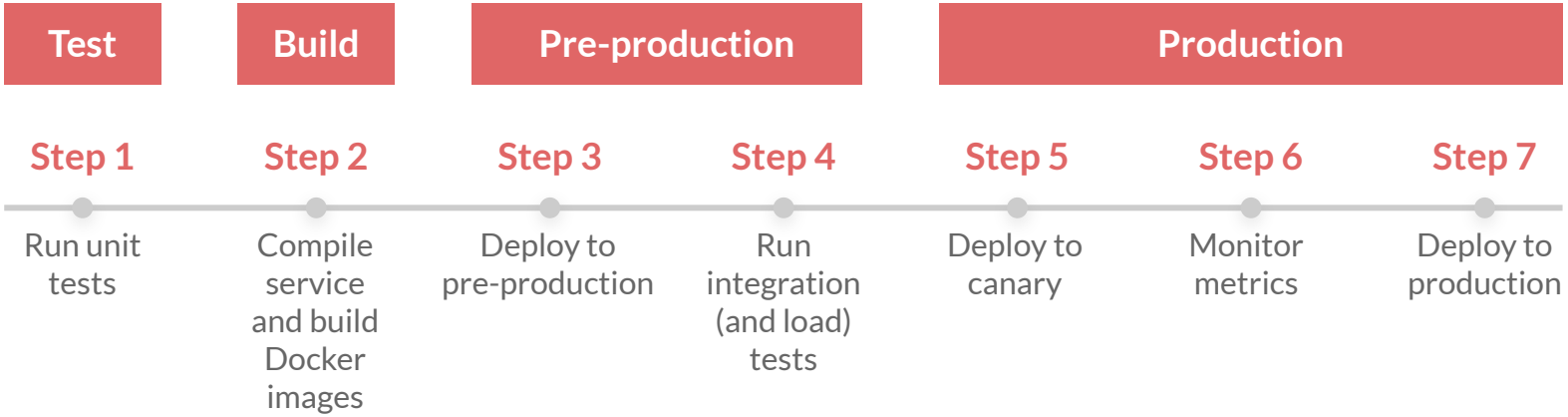
```
// Copyright 2017, Google, Inc.
// Licensed under the Apache License, Version 2.0 (the "License")
var http = require('http');
var server = http.createServer(function (request, response) {
  const language = 'English';
  const API_KEY = '123-456-789';
  response.write(`Language: ${language}\n`);
  response.write(`API Key: ${API_KEY}\n`);
  response.end(`\n`);
});
server.listen(3000);
```

https://medium.com/google-cloud/kubernetes-configmaps-and-secrets-68d061f7ab5b

```
// Copyright 2017, Google, Inc.
// Licensed under the Apache License, Version 2.0 (the "License")
var http = require('http');
var server = http.createServer(function (request, response) {
  const language = process.env.LANGUAGE;
  const API_KEY = process.env.API_KEY;
  response.write(`Language: ${language}\n`);
  response.write(`API Key: ${API_KEY}\n`);
  response.end(`\n`);
});
server.listen(3000);
```

https://medium.com/google-cloud/kubernetes-configmaps-and-secrets-68d061f7ab5b


Test
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Run unit tests
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production


Test
Build
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Run unit tests
Compile service and build Docker images
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production

```
stages:
- test

- before-build-preproduction
- build-preproduction
- before-deploy-preproduction
- deploy-preproduction
- after-deploy-preproduction

- before-build-canary
- build-canary
- before-deploy-canary
- deploy-canary

- before-deploy-production
- deploy-production
```

```
stages:
- test

- before-build-preproduction
- build-preproduction
- before-deploy-preproduction
- deploy-preproduction
- after-deploy-preproduction

- before-build-canary
- build-canary
- before-deploy-canary
- deploy-canary

- before-deploy-production
- deploy-production
```

```
stages:
- test

- before-build
- build

- before-deploy-preproduction
- deploy-preproduction
- after-deploy-preproduction

- before-deploy-canary
- deploy-canary

- before-deploy-production
- deploy-production
```

```yaml
compile-preproduction:
  stage: before-build-preproduction
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  only:
    refs:
    - master
    - /^release.+/
  script:
  - make compile
  artifacts:
    paths:
    - deploy/_output

build-image-preproduction:
  stage: build-preproduction
  image: $CI_REGISTRY/bukalapak/gudang/docker:1.0.0
  only:
    refs:
    - master
    - /^release.+/
  script:
  - export VERSION=${CI_COMMIT_TAG:-$CI_COMMIT_SHA}-preproduction
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" "$CI_REGISTRY"
  - make build-image push
```

```
compile:
  stage: before-build
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  only:
    refs:
    - master
    - /^release.+/
  script:
  - make compile
  artifacts:
    paths:
    - deploy/_output

build-image:
  stage: build
  image: $CI_REGISTRY/bukalapak/gudang/docker:1.0.0
  only:
    refs:
    - master
    - /^release.+/
  script:
  - export VERSION=${CI_COMMIT_TAG:-$CI_COMMIT_SHA}
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" "$CI_REGISTRY"
  - make build-image push
```

How do we **deploy and link** the configuration **at runtime**?

# Kubernetes ConfigMap

Bind environment variables
to Pods' containers
at runtime

Separate configuration
from code

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: lulu-config
data:
  DATABASE_HOST: 10.80.47.5
  DATABASE_USERNAME: lulu_rw
  DATABASE_PASSWORD: -zH_FsinJx38KLyy
```

# Kubernetes ConfigMap

Bind environment variables
to Pods' containers
at runtime

Separate configuration
from code

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: lulu-config
data:
  DATABASE_HOST: 10.80.47.5
  DATABASE_USERNAME: lulu_rw
  DATABASE_PASSWORD: zH_FsinJx38KLyy
```

# Kubernetes Secret

Store **sensitive data** in clusters

**Give control** over how sensitive data is used

Reduce the risk of exposing the data to **unauthorized users**

```
apiVersion: v1
kind: Secret
metadata:
  name: lulu-secret
data:
  DATABASE_PASSWORD: LXpIX0ZzaW5KeDM4S0x5eQ==
```

# Link them all

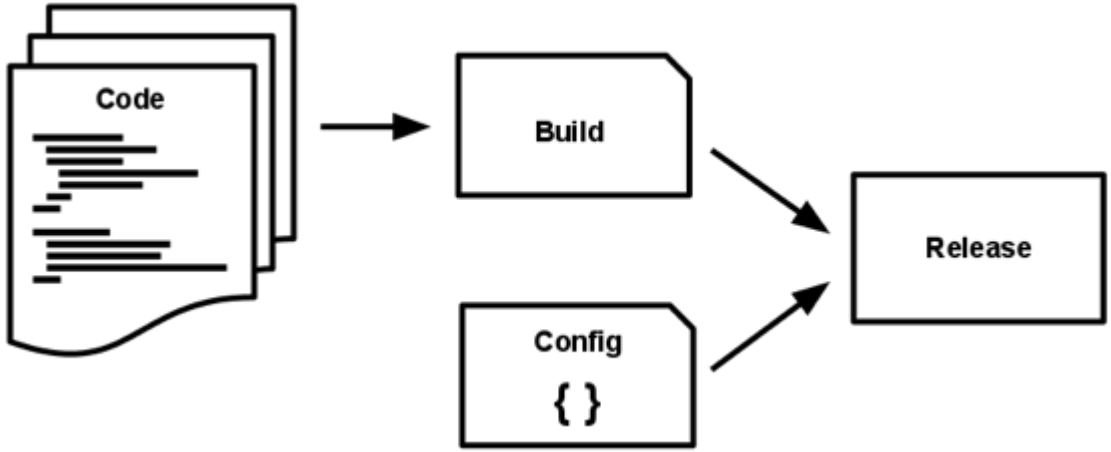
Store sensitive data
in clusters

Give control over
how sensitive data is used

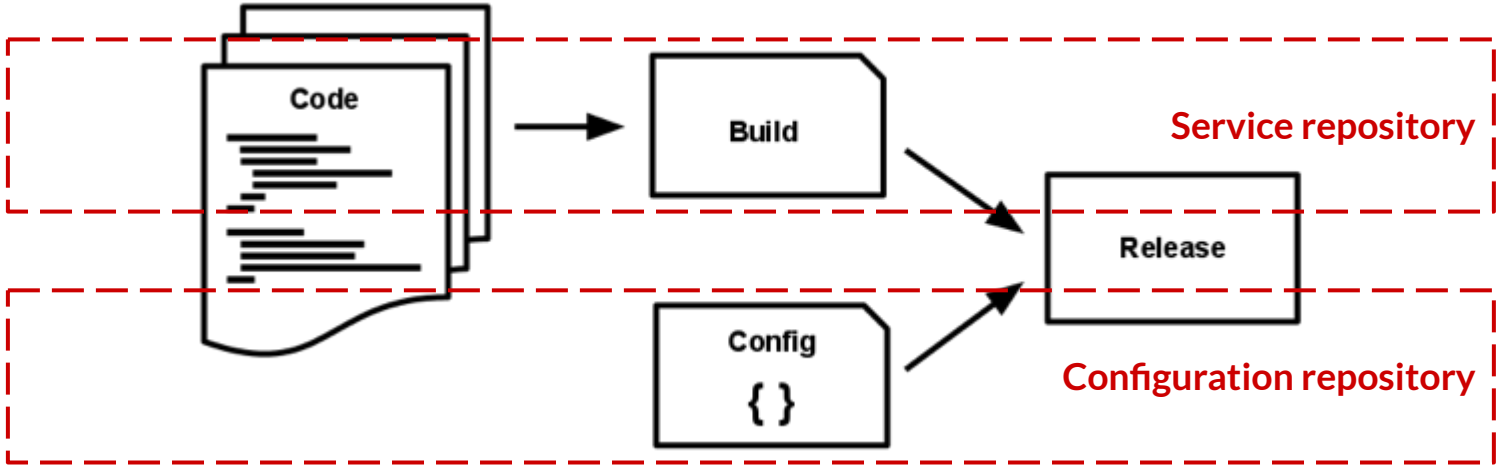
Reduce the risk
of exposing the data
to unauthorized users

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lulu-api
spec:
  template:
    spec:
      containers:
      - name: lulu-api
        envFrom:
        - configMapRef:
            name: lulu-config
        - secretRef:
            name: lulu-secret
```

# How do we **manage** the configuration?




https://12factor.net/config




https://12factor.net/config

# Minerva

Stores service **configuration files** and **manifest templates**

Ensures **separation** between code and configuration

Provides CI/CD **pipeline and tooling**

```
minerva
└── services
    └── lulu
        ├── config
        │   ├── kubeconfig.yaml
        │   ├── kubelize
        │   ├── preproduction
        │   │   └── web
        │   └── production
        │       └── web
        └── manifest
            └── web
                ├── deployment.yml
                ├── hpa.yml
                ├── service.yml
                ├── resourcequota.yml
                └── ingress.yml
```

```
apiVersion: v1
clusters:
- name: preproduction
  cluster:
    certificate-authority-data: <certificate-for-your-cluster>
    server: https://<cluster-endpoint>
- name: production
  cluster:
    certificate-authority-data: <certificate-for-your-cluster>
    server: https://<cluster-endpoint>
```

```yaml
projectName: sample-examples
services:
- name: web
  port: "8888"
  language: go
  environments:
  ...
  minCpu: "1"
  maxCpu: "2"
  minMem: "0.5"
  maxMem: "2"
  averageUtilization: 65
  rollingUpdate:
    maxUnavailable: 10%
    maxSurge: 25%
  healthCheck:
    delay: "10"
    healthz: healthz
    ready: healthz
    period: 4
    timeout: "2"
    liveFailThreshold: 1
    liveSuccThreshold: 1
    readFailThreshold: 1
    readSuccThreshold: 2
```

```yaml
environments:
- name: canary
  replica: "1"
- name: production
  replica: "5"
  minReplicas: "5"
  maxReplicas: "20"
  quotaRequestsCpu: 25
  quotaLimitsCpu: 50
  quotaRequestsMem: 12.5
  quotaLimitsMem: 50
- name: preproduction
  replica: "1"
  minReplicas: "1"
  maxReplicas: "5"
  quotaRequestsCpu: 6.25
  quotaLimitsCpu: 12.5
  quotaRequestsMem: 3.125
  quotaLimitsMem: 12.5
```

```
ENV=preproduction
DATABASE_HOST=10.80.47.5
DATABASE_USERNAME=lulu_rw
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{.config.projectName}}-{{.service.name}}-{{.environment.name}}
  namespace: {{.config.projectName}}
  ...
spec:
  replicas: {{.environment.replica}}
  ...
  template:
    metadata: ...
    spec:
      ...
      containers:
      - name: {{.config.projectName}}-{{.service.name}}
        image: {{.variable.CI_REGISTRY}}/bukalapak/{{.config.projectName}}/{{.service.name}}:{{.variable.VERSION}}
        ...
        envFrom:
          - configMapRef:
              name: {{.config.projectName}}-{{.service.name}}-{{or .variable.SUFFIX .environment.name}}
              optional: false
          - secretRef:
              name: {{.config.projectName}}-{{.service.name}}-{{or .variable.SUFFIX .environment.name}}
              optional: false
```

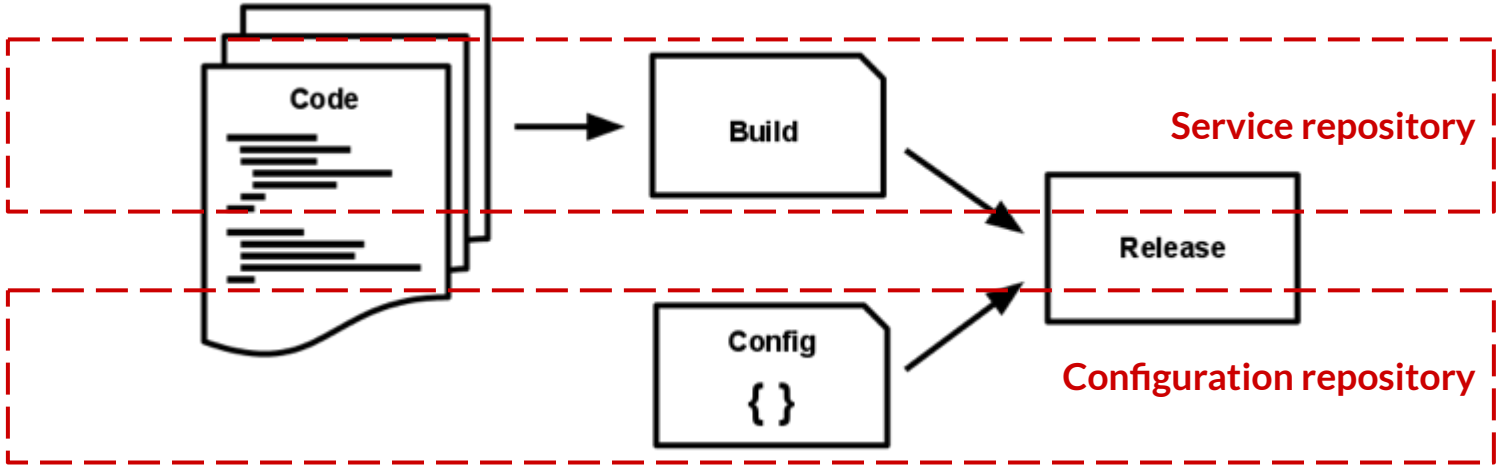

# Hashicorp Vault

Stores **arbitrary key/value secrets**

**Encrypts** the secrets prior to writing them to persistent storage

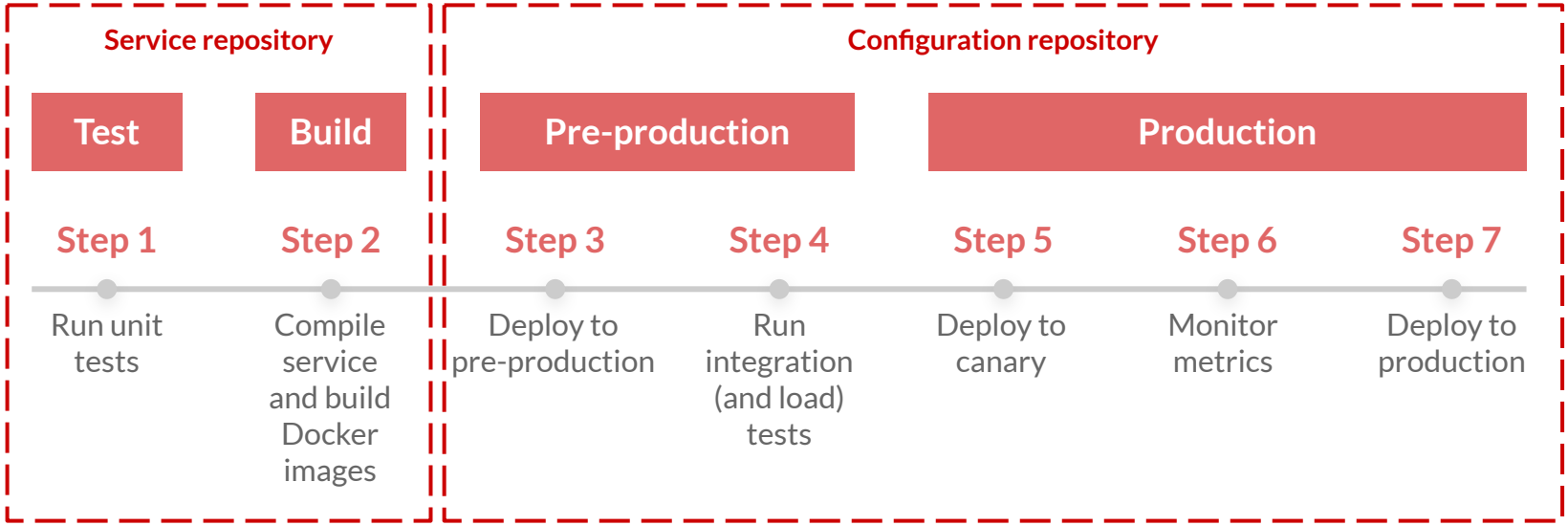
Controls **access** to tokens, passwords, ...

https://www.vaultproject.io




https://12factor.net/config

```yaml
include:
- project: 'bukalapak/minerva'
  ref: v1.0.0
  file: 'shared/pipeline-template/.gitlab-ci-general.yaml'

test: ...

lint: ...

compile: ...

build: ...
```


Service repository
Configuration repository
Test
Build
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Run unit tests
Compile service and build Docker images
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production

```yaml
preproduction-manifest:
  extends: .generate-and-validate-manifest
  stage: before-preproduction-deployment
  variables:
    ENV: preproduction

preproduction-configuration:
  extends: .configuration-deployment
  stage: before-preproduction-deployment
  variables:
    ENV: preproduction
```

```yaml
deploy-preproduction:
  extends: .deploy-template
  stage: preproduction-deployment
  variables:
    ENV: preproduction
    FILEPATH: deploy/_output/$ENV/manifest
  environment:
    name: preproduction
  artifacts:
    paths:
    - ${FILEPATH}/.rollout_check.list

preproduction-done:
  extends: .releaselog-done
  stage: after-preproduction-deployment
  variables:
    NOTIFICATION_ENV: preproduction
```

Then … we need to **rollback**.

```
compile:
  stage: before-build
  image: $CI_REGISTRY/bukalapak/gudang/golang:2.4.1
  only:
    refs:
    - master
    - /^release.+/
  script:
  - make compile
  artifacts:
    expire_in: 1 day
    paths:
    - deploy/_output

build-image:
  stage: build
  image: $CI_REGISTRY/bukalapak/gudang/docker:1.0.0
  only:
    refs:
    - master
    - /^release.+/
  script:
  - export VERSION=${CI_COMMIT_TAG:-$CI_COMMIT_SHA}
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" "$CI_REGISTRY"
  - make build-image push
```

How to link between
code and config **version**?

# peru

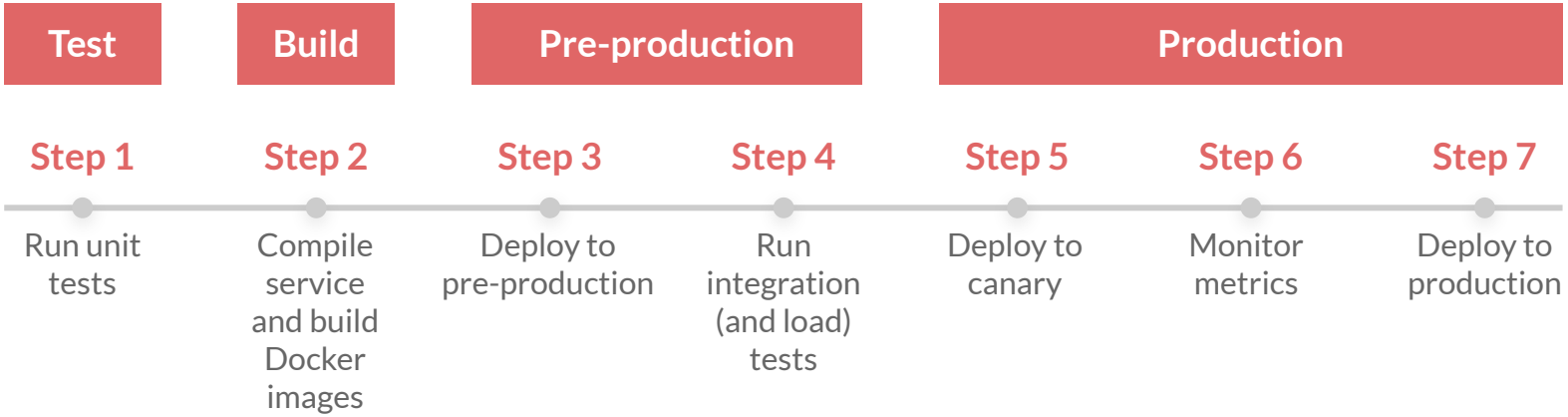
Includes code from another **repository**

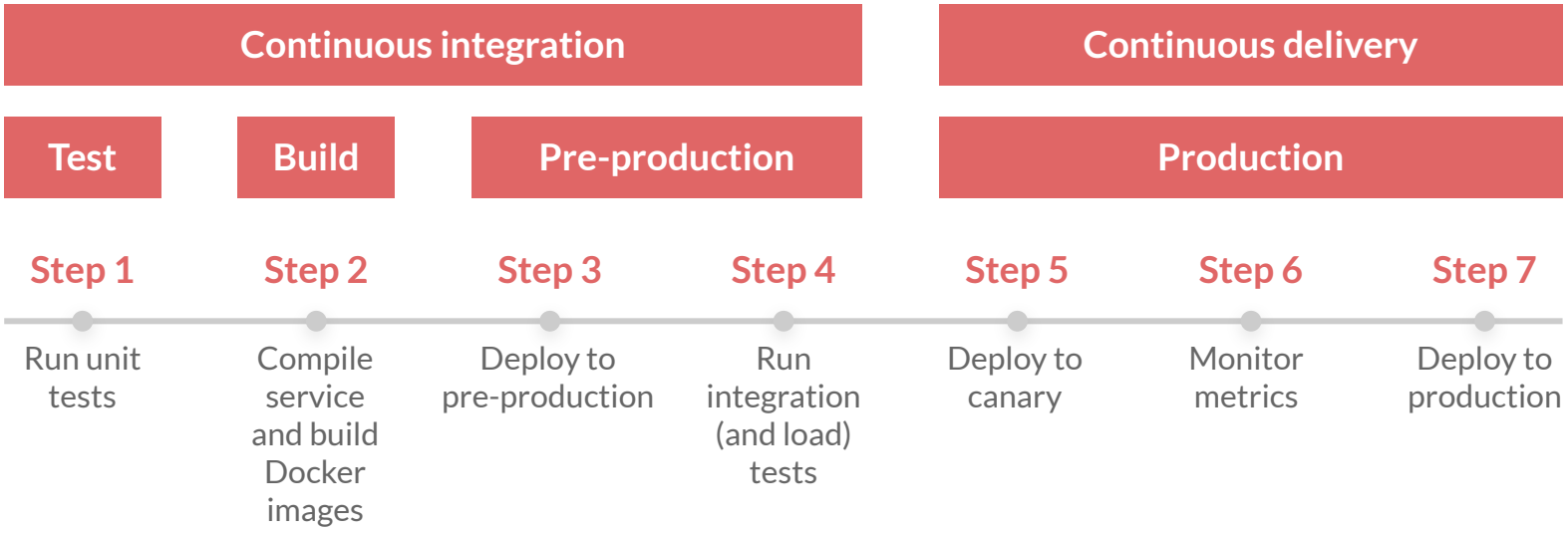
Helps track **exact version** of dependencies

so the history is always **reproducible**

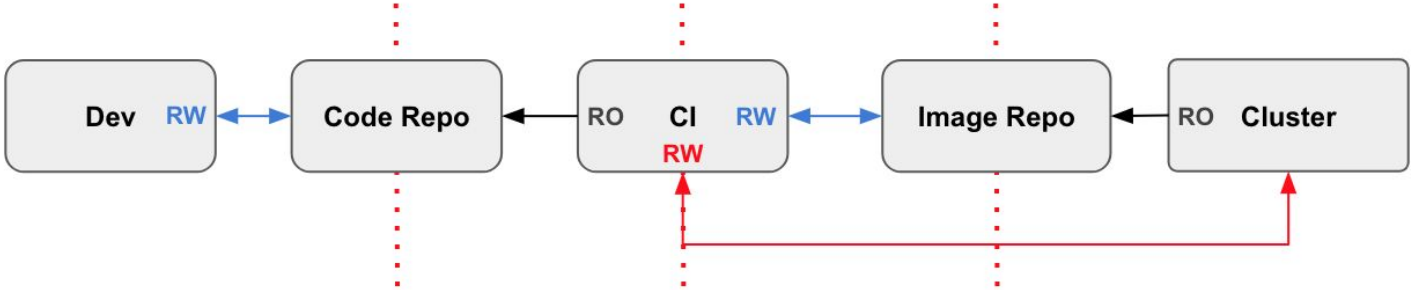
https://github.com/buildinspace/peru

```
imports:
  minerva: minerva/

git module minerva:
  url: https://gitlab.bl.internal/bukalapak/minerva.git
  rev: 2f9f7ef5aaa60d55279ffd255d42556f30e3507a
```

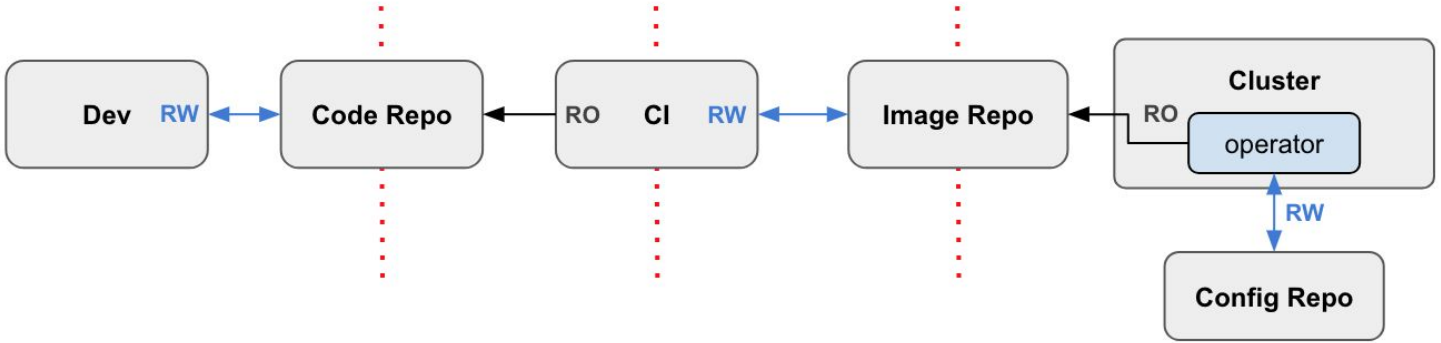

Test
Build
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Run unit tests
Compile service and build Docker images
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production


Continuous integration
Continuous delivery
Test
Build
Pre-production
Production
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Run unit tests
Compile service and build Docker images
Deploy to pre-production
Run integration (and load) tests
Deploy to canary
Monitor metrics
Deploy to production

# Is it ... **GitOps** already?




https://www.weave.works/technologies/gitops/




https://www.weave.works/technologies/gitops/

# Want to improve that?

**Scratch your own itch; let's work together ;)**

# Thank you!

**Muhammad Saiful Islam**

@saifulwebid on the Internet

saiful.islam@bukalapak.com