How to Code, Deploy, and Operate Cloud-Native Apps Using Kubernetes

Aditya Satrya
Head of IT Development
Jabar Digital Service
https://digitalservice.jabarprov.go.id

Outline

- Cloud-Native & 12-Factor App
- Kubernetes
- 12-Factor App using Kubernetes

INDONESIA
OpenInfra Days

Cloud-Native & 12-Factor App

Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation

Cloud-Native Application

- **Operability:** Expose control of application/system lifecycle.
- **Observability:** Provide meaningful signals for observing state, health, and performance.
- **Elasticity:** Grow and shrink to fit in available resources and to meet fluctuating demand.
- **Resilience:** Fast automatic recovery from failures.
- **Agility:** Fast deployment, iteration, and reconfiguration

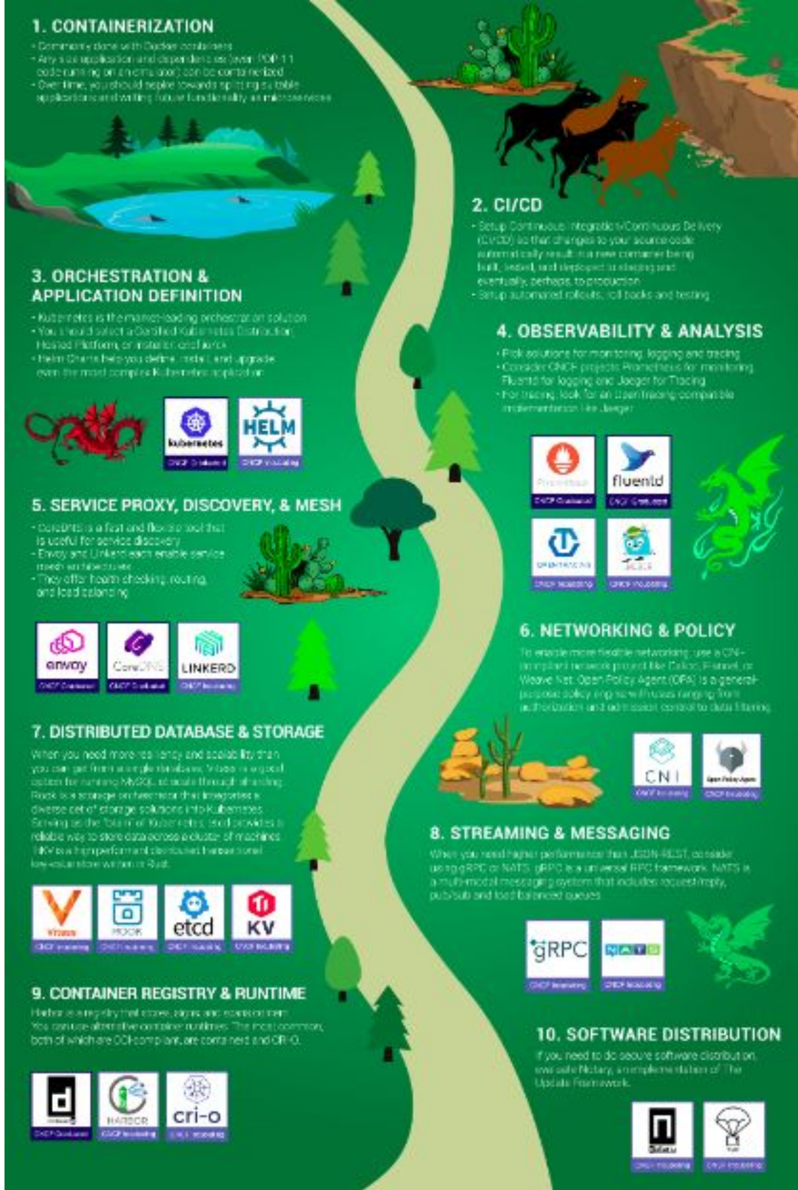
Cloud-Native Trail Map

1. **Containerization**
2. **CI/CD**
3. **Orchestration**

--below this are optional--

4. Observability
5. Service Discovery
6. Networking & Policy
7. Distributed database & storage
8. Streaming & messaging
9. Container registry
10. Software distribution

Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation



1. CONTAINERIZATION
- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

2. CI/CD
- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION & APPLICATION DEFINITION
- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: cncf.io/ck
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application

4. OBSERVABILITY & ANALYSIS
- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

5. SERVICE PROXY, DISCOVERY, & MESH
- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

6. NETWORKING & POLICY
To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general-purpose policy engine with uses ranging from authorization and admission control to data filtering

7. DISTRIBUTED DATABASE & STORAGE
When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding. Rook is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. Serving as the "brain" of Kubernetes, etcd provides a reliable way to store data across a cluster of machines. TiKV is a high performant distributed transactional key-value store written in Rust.

8. STREAMING & MESSAGING
When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues.

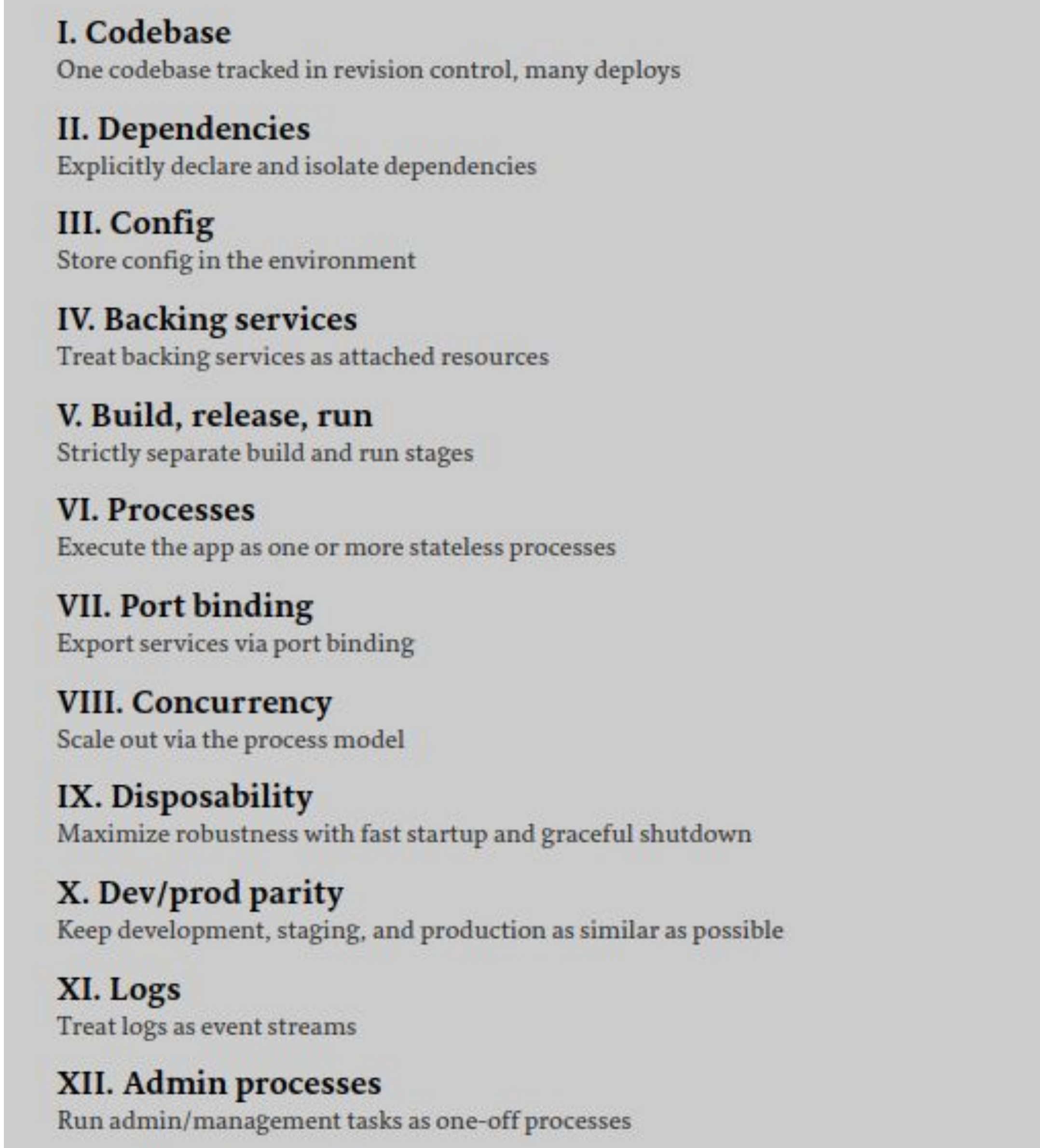
9. CONTAINER REGISTRY & RUNTIME
Harbor is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.

10. SOFTWARE DISTRIBUTION
If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

12-Factor App

- Methodology to build app optimized for the cloud (cloud-native)
- Drafted by developers at Heroku (2011)
- http://12factor.net

The Twelve Factor



I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

INDONESIA
OpenInfra Days

Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes

Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

X. Dev/prod parity
Keep development, staging, and production as similar as possible

VII. Port binding
Export services via port binding

Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes

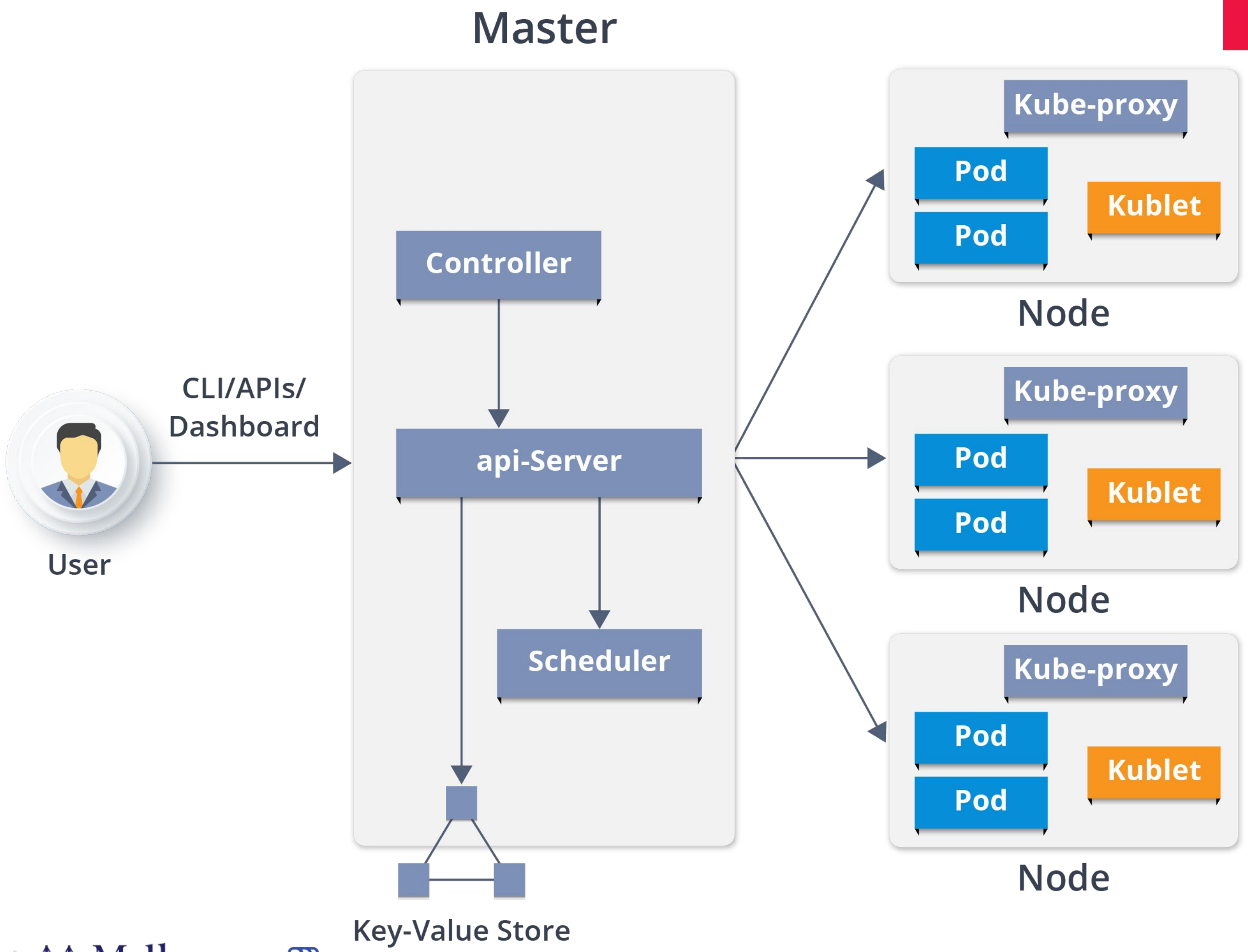
Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation

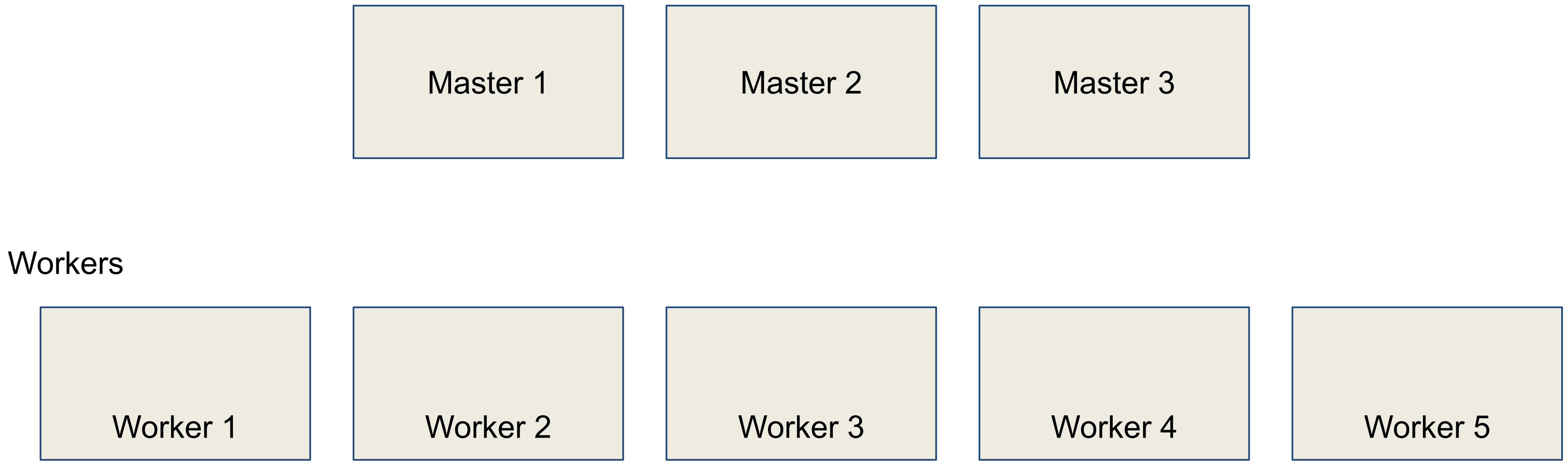
INDONESIA
OpenInfra Days

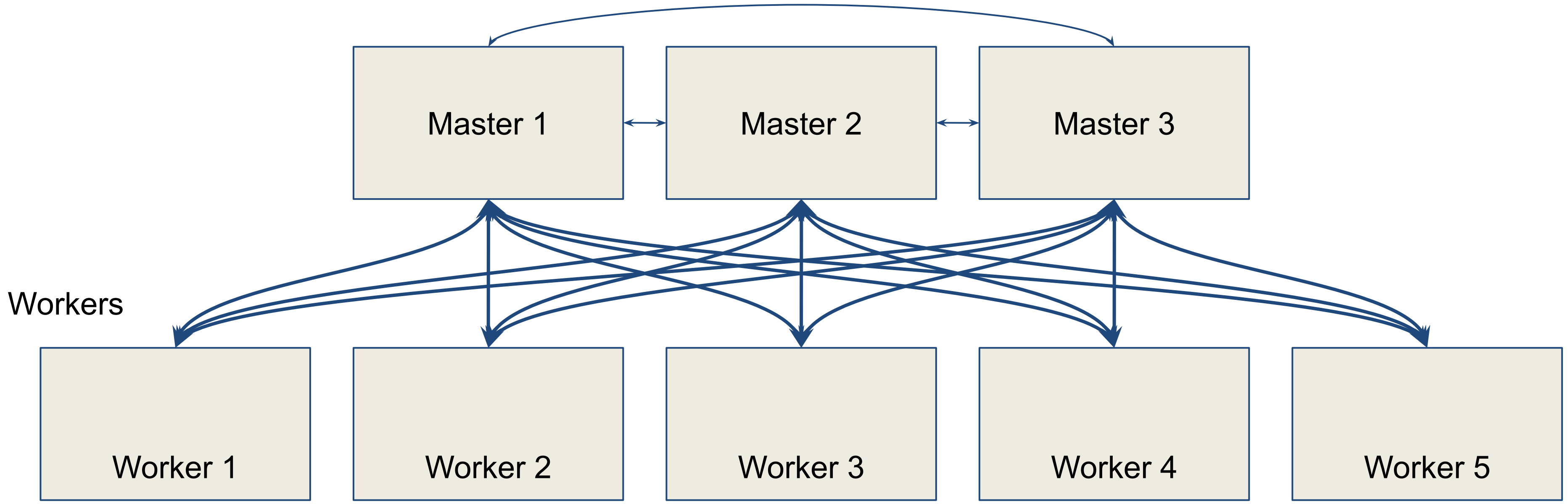
Kubernetes

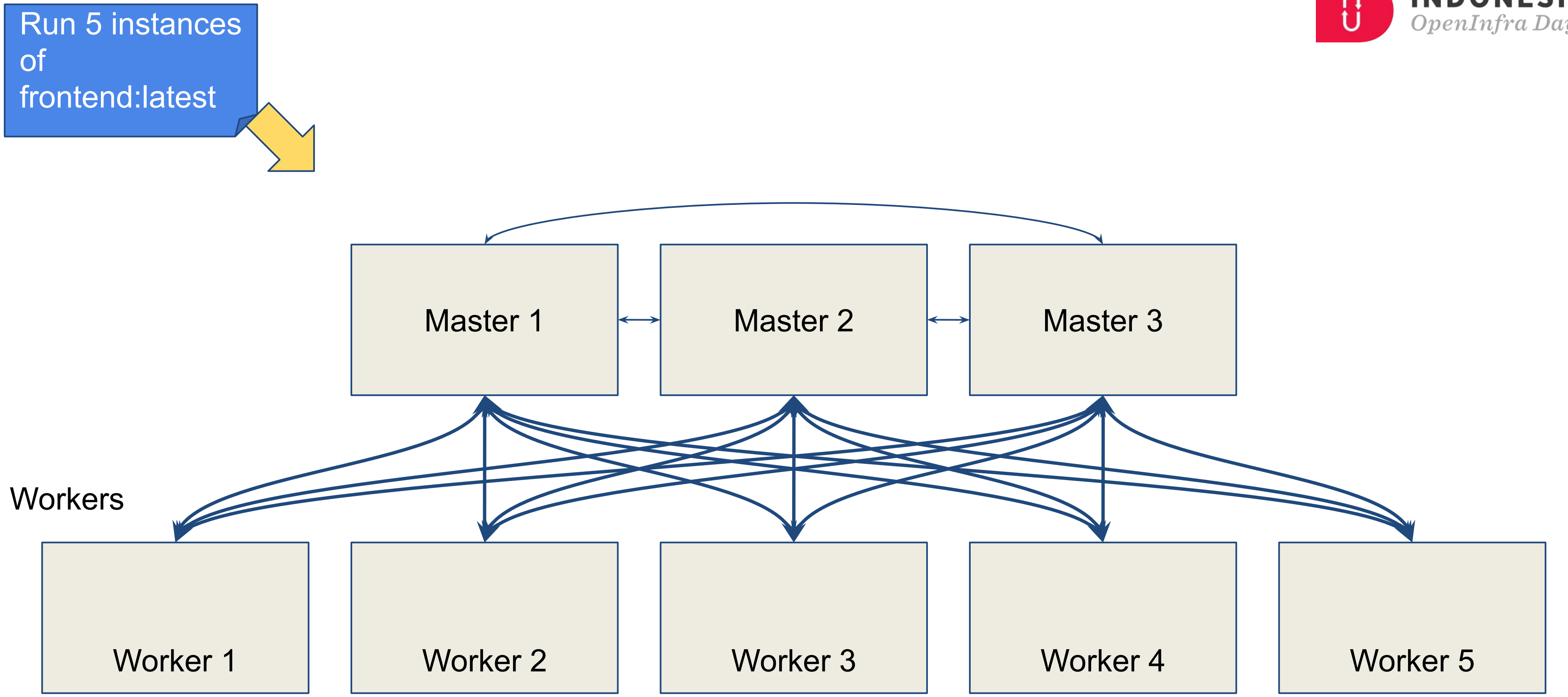
What is Kubernetes?

- Open-source system for automating:
  - deployment
  - scaling
  - management of containerized applications

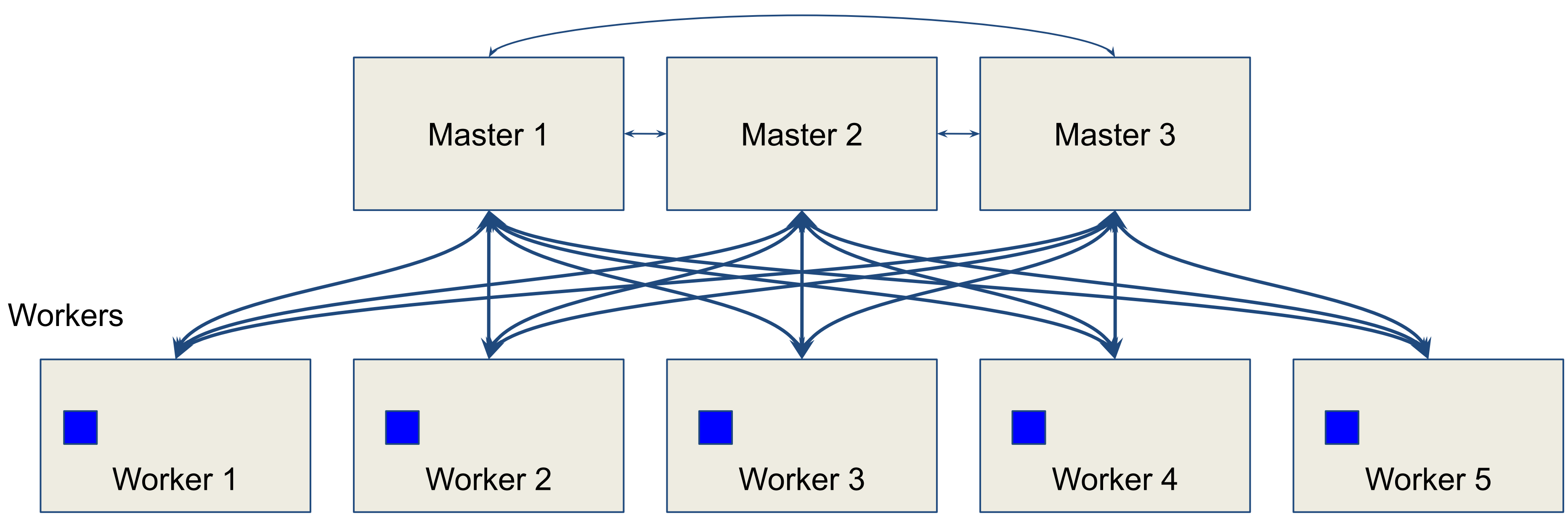

Master
Controller
api-Server
Scheduler
Key-Value Store
User
CLI/APIs/
Dashboard
Kube-proxy
Pod
Pod
Kublet
Node
Kube-proxy
Pod
Pod
Kublet
Node
Kube-proxy
Pod
Pod
Kublet
Node

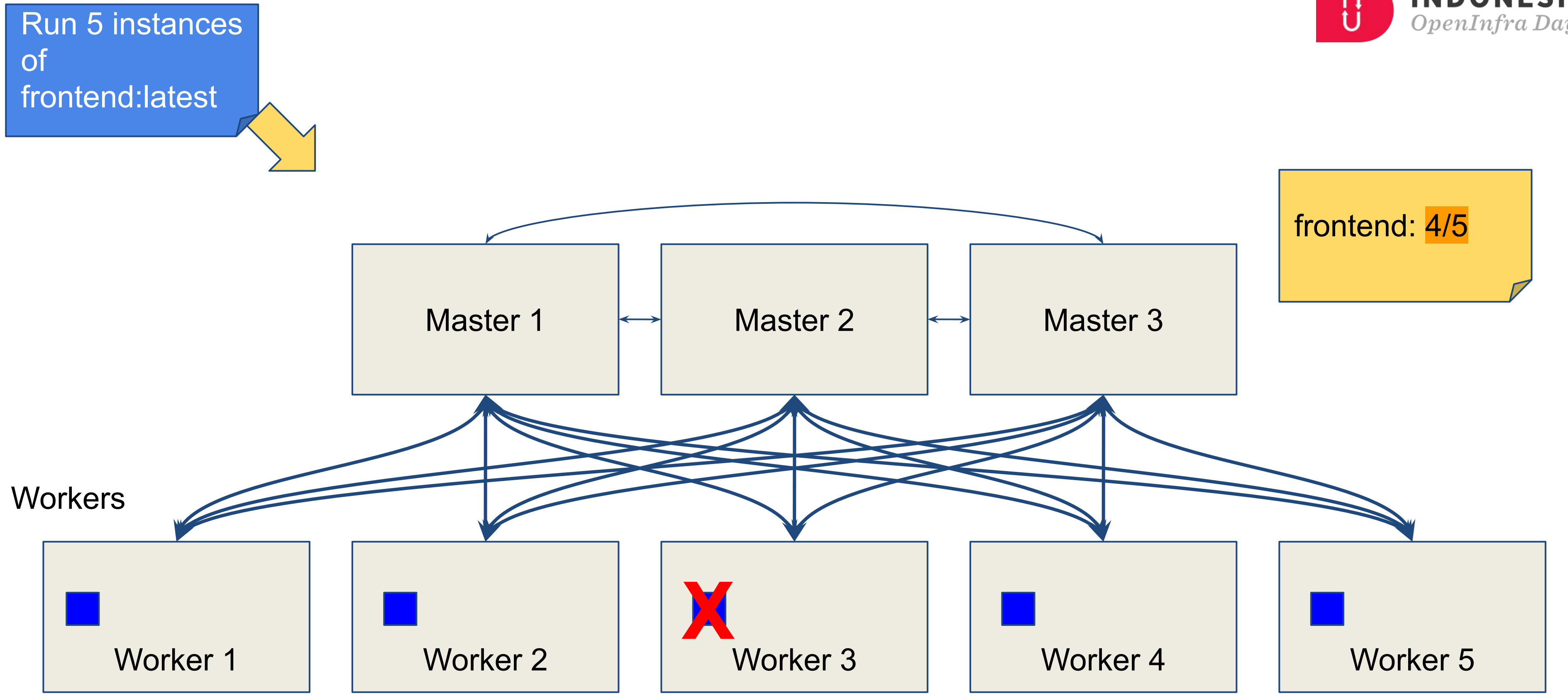

Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Run 5 instances of frontend:latest
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

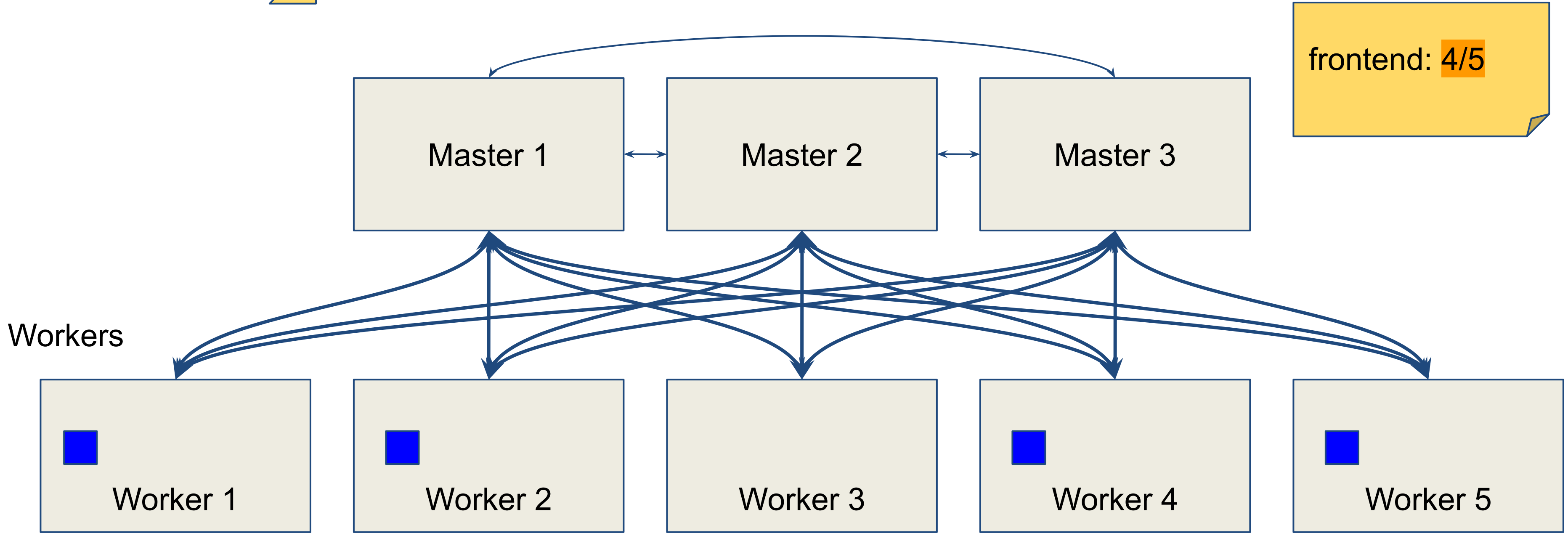
Run 5 instances of frontend:latest

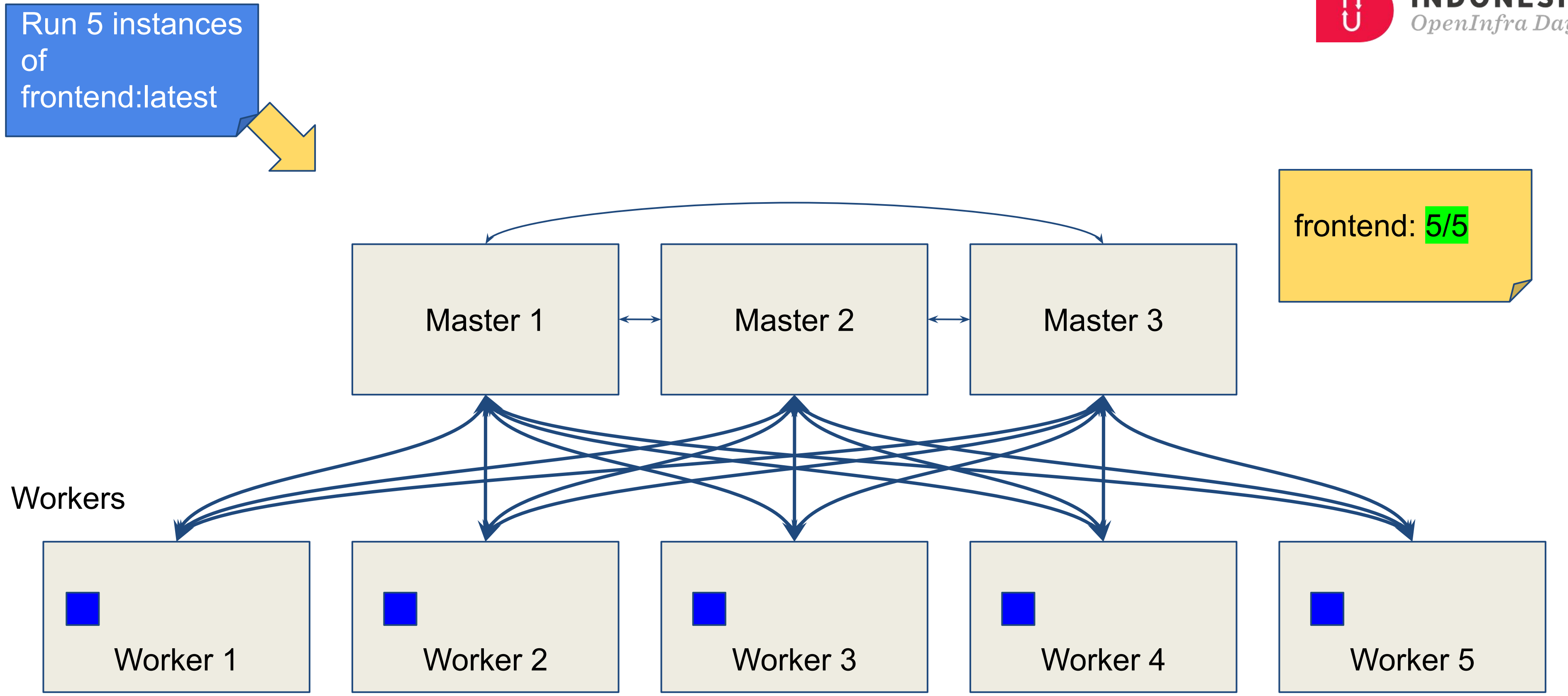


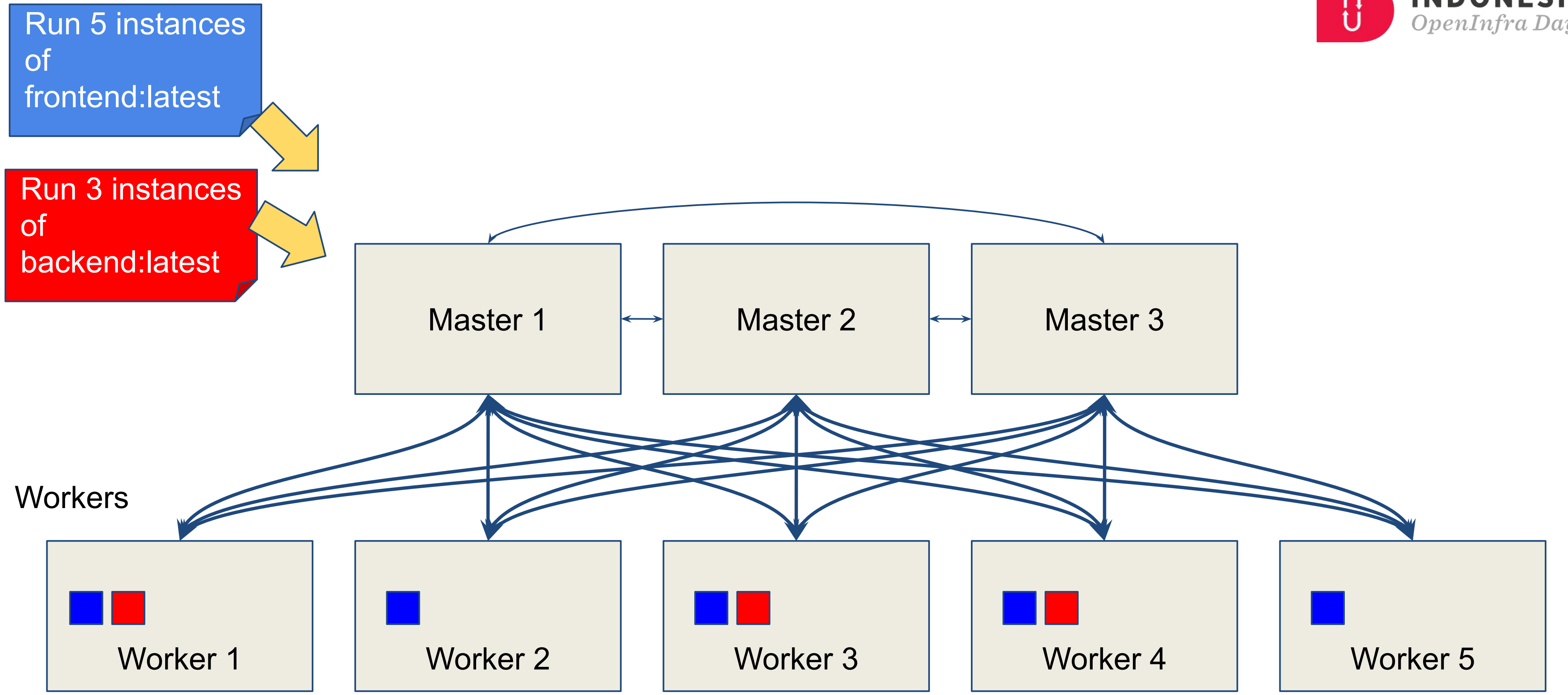

Run 5 instances of frontend:latest
frontend: 4/5
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

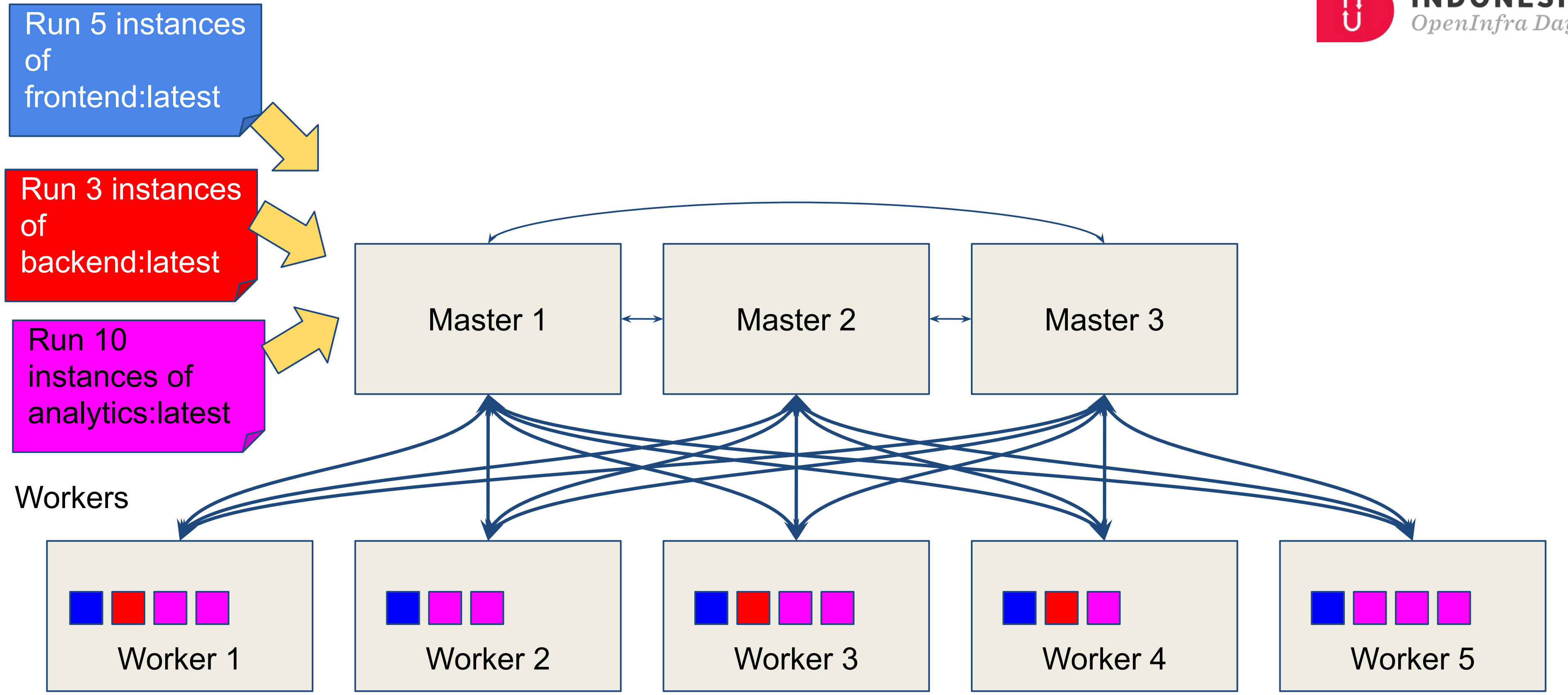
Run 5 instances of frontend:latest

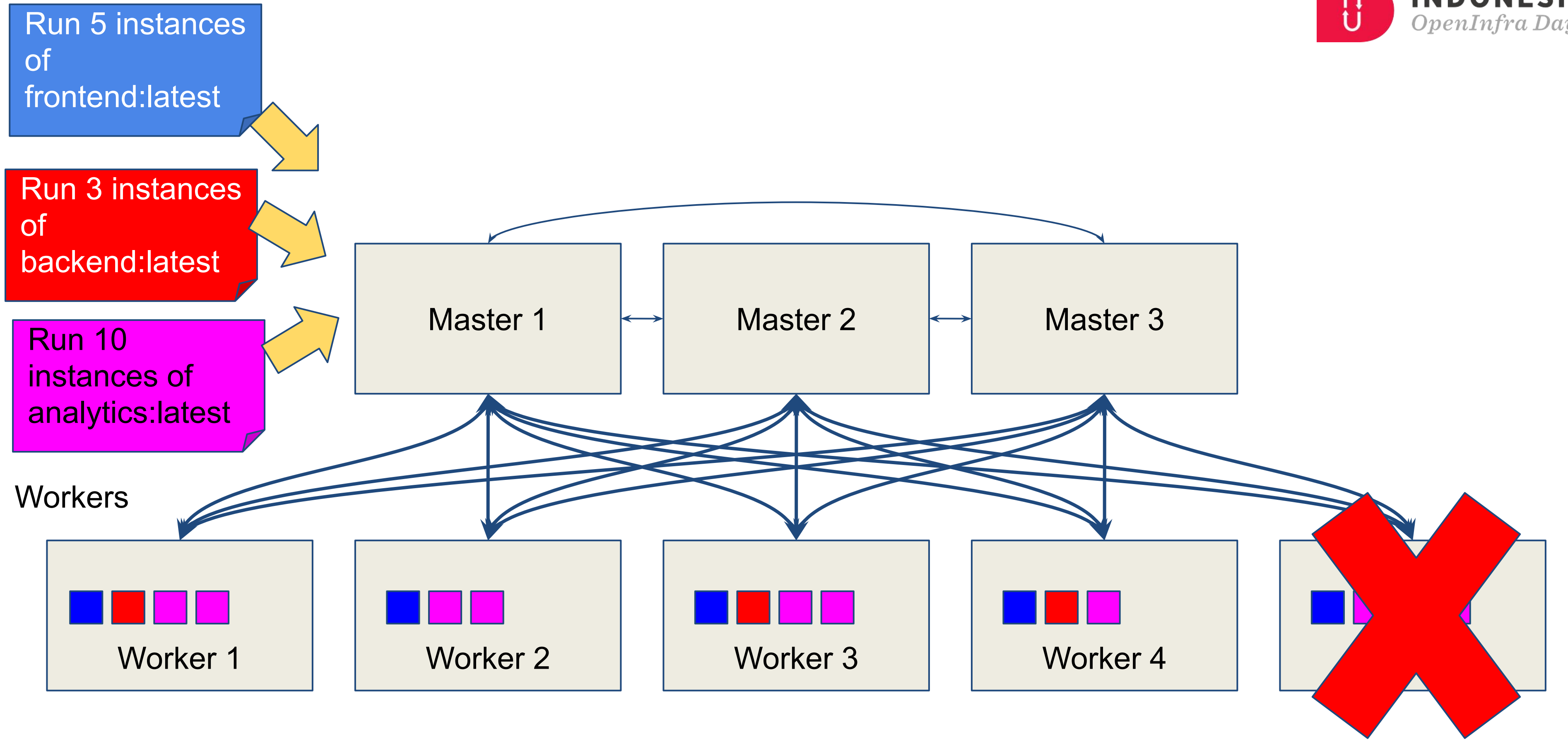
frontend: 4/5

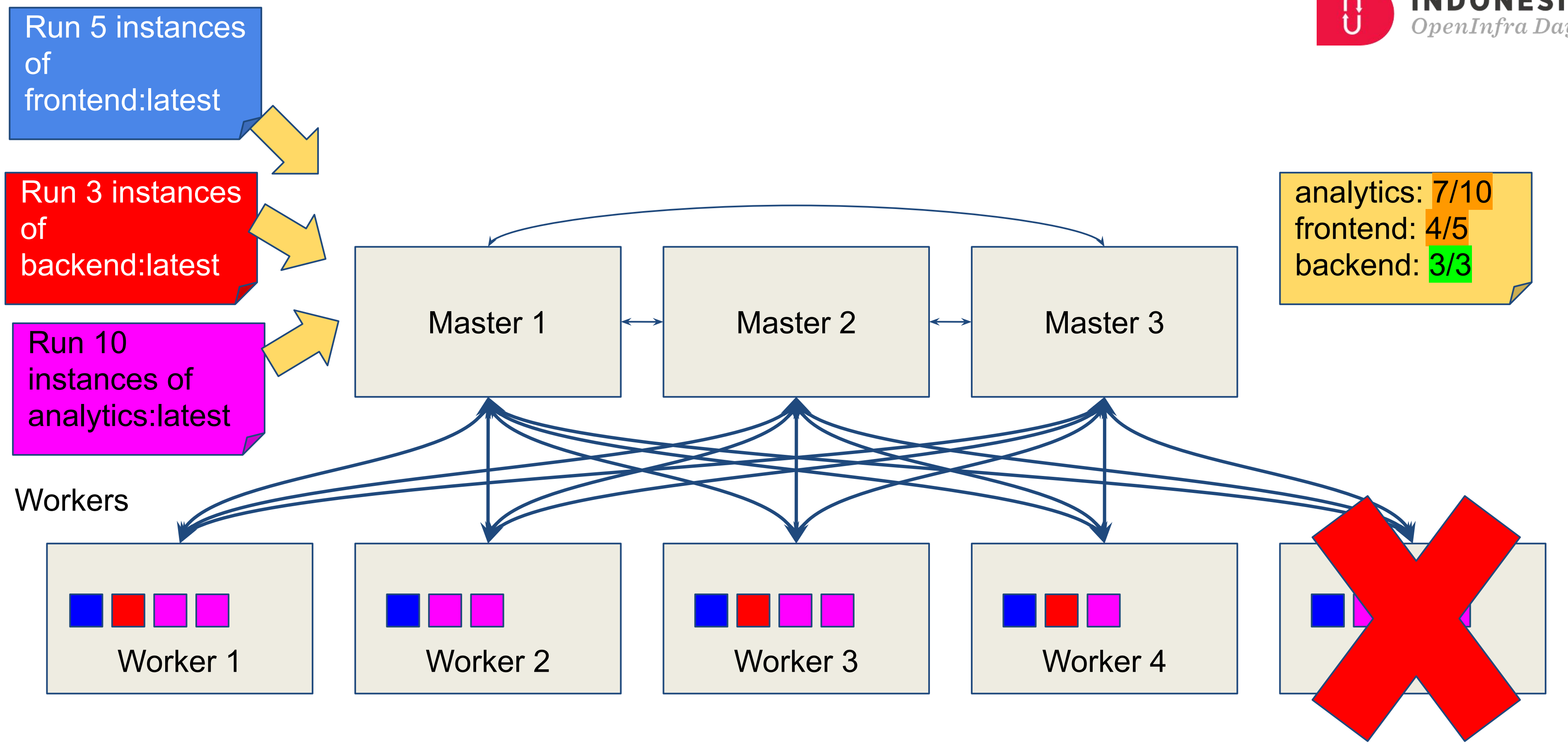


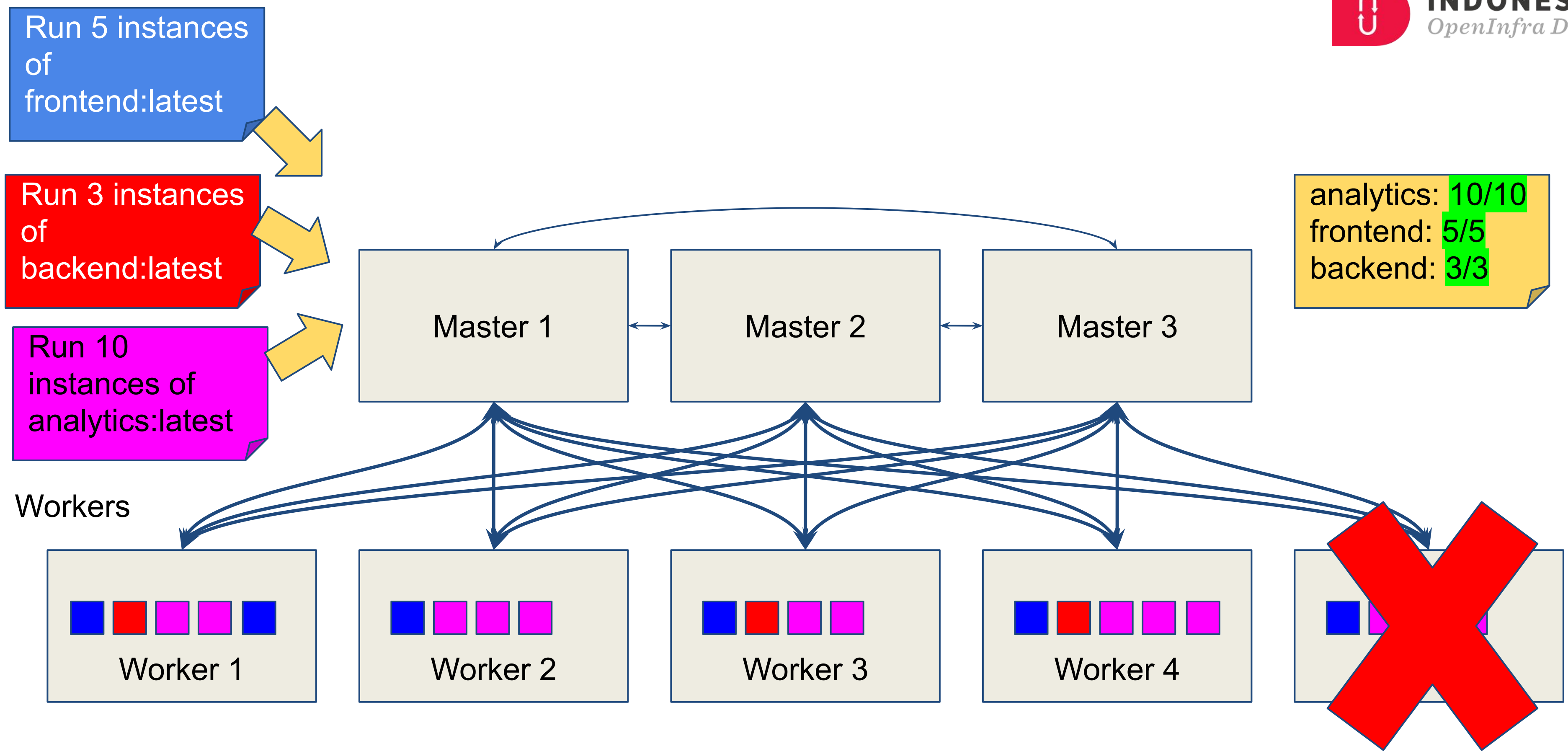

Run 5 instances of frontend:latest
frontend: 5/5
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

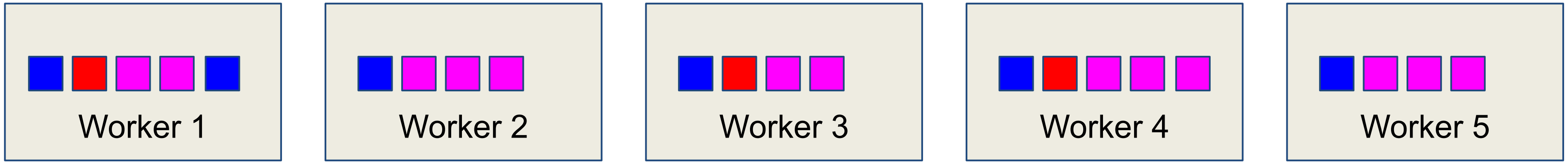

Run 5 instances of frontend:latest
Run 3 instances of backend:latest
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

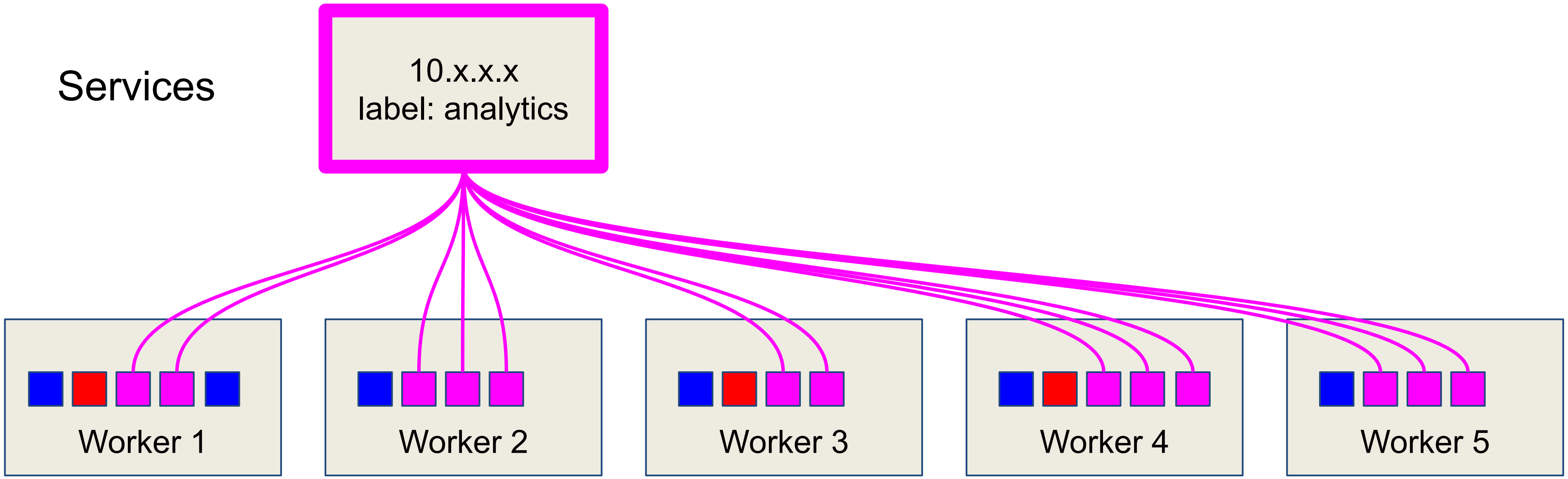

Run 5 instances of frontend:latest
Run 3 instances of backend:latest
Run 10 instances of analytics:latest
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

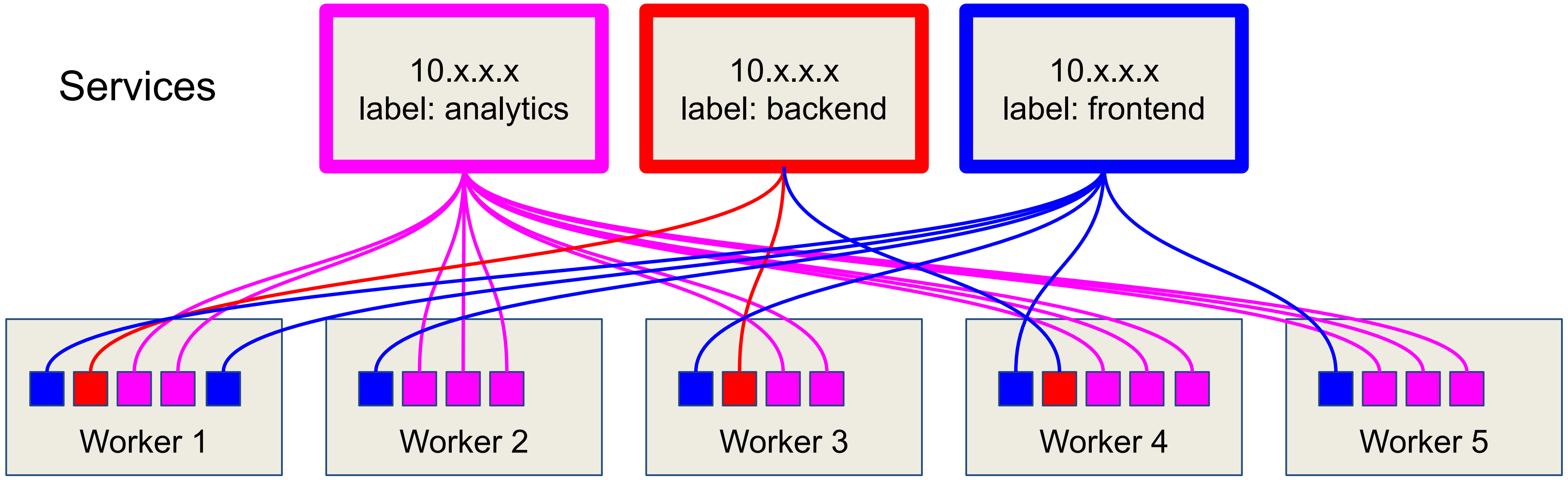

Run 5 instances of frontend:latest
Run 3 instances of backend:latest
Run 10 instances of analytics:latest
Master 1
Master 2
Master 3
Workers
Worker 1
Worker 2
Worker 3
Worker 4

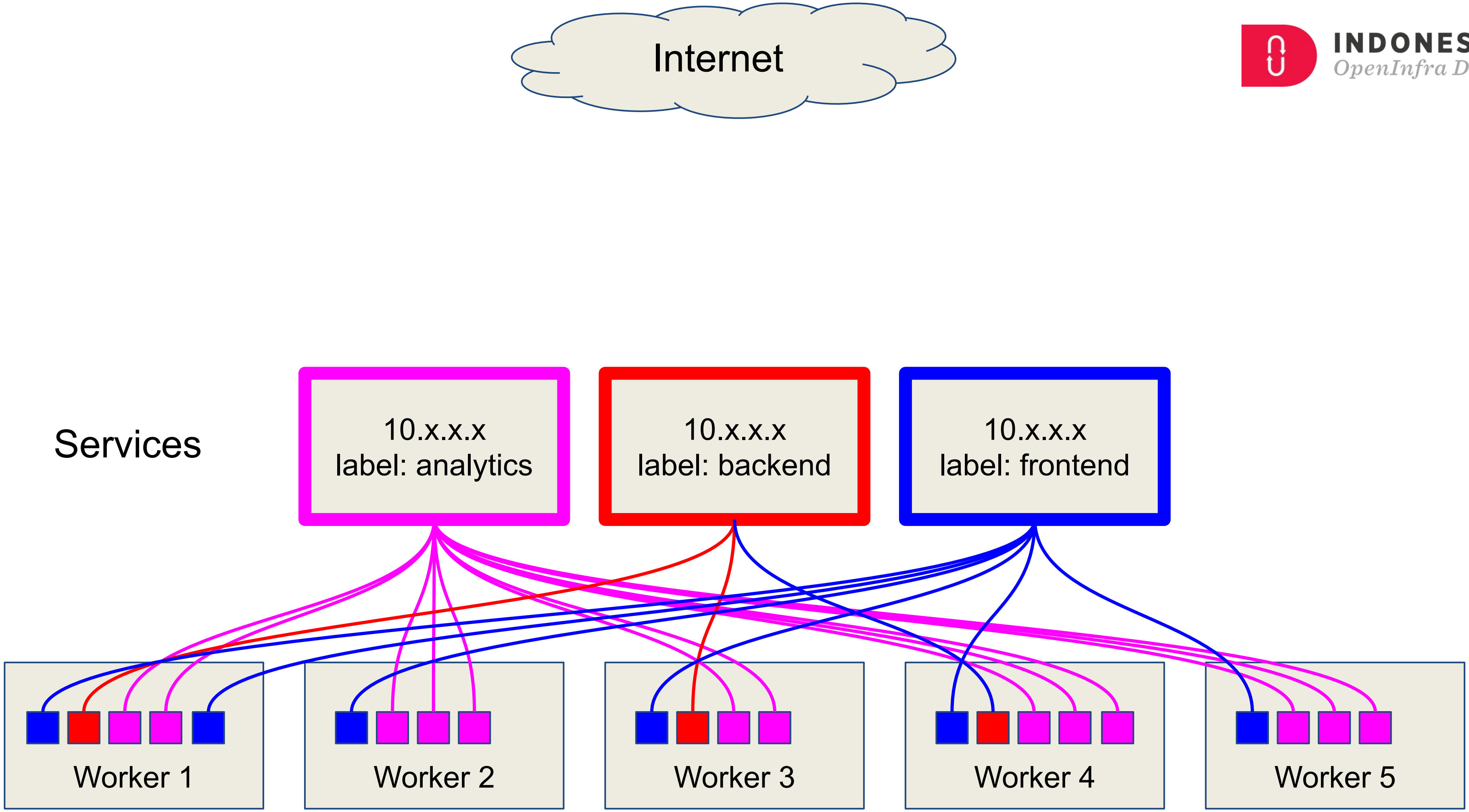

Run 5 instances of frontend:latest
Run 3 instances of backend:latest
Run 10 instances of analytics:latest
Master 1
Master 2
Master 3
analytics: 7/10
frontend: 4/5
backend: 3/3
Workers
Worker 1
Worker 2
Worker 3
Worker 4

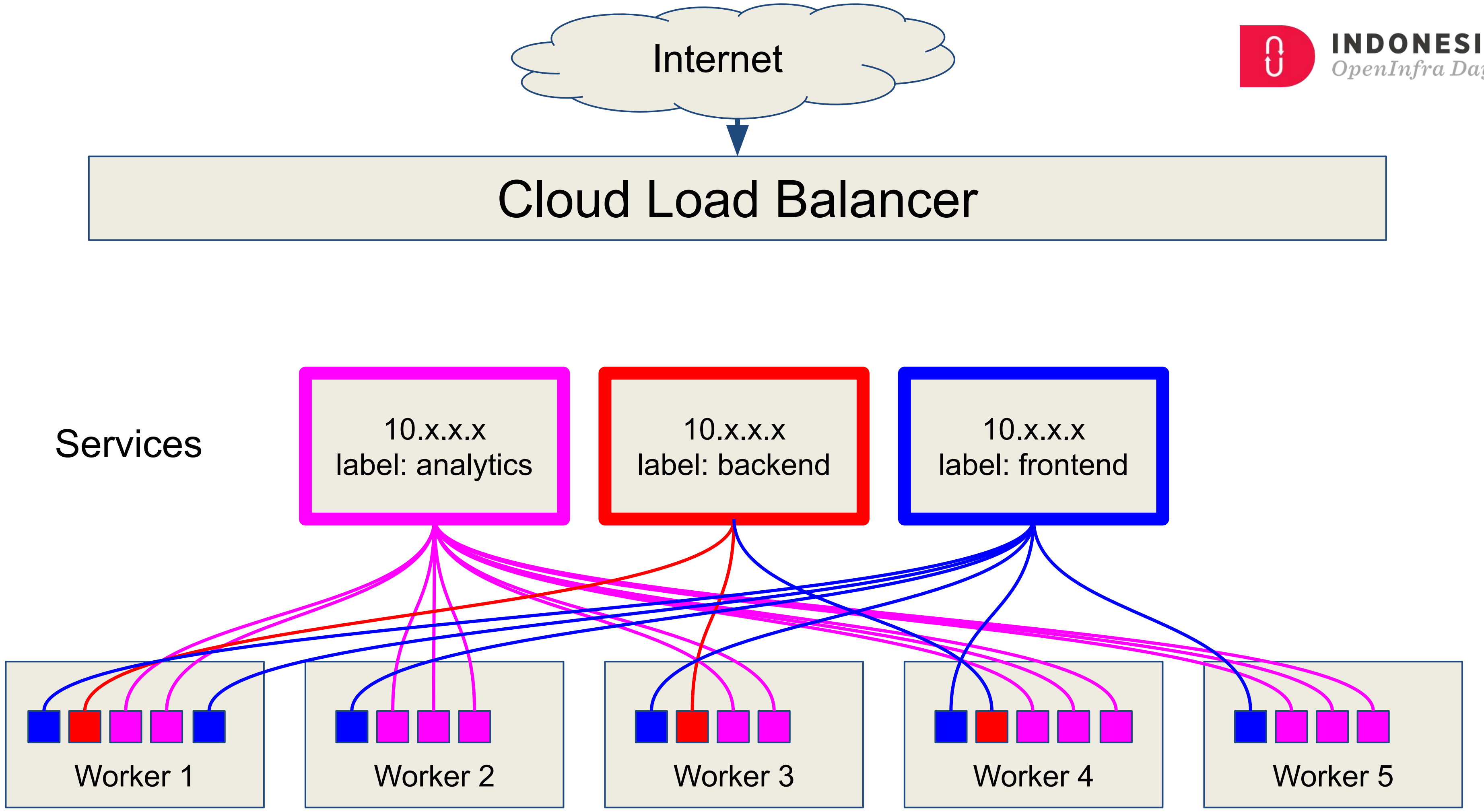

Run 5 instances of frontend:latest
Run 3 instances of backend:latest
Run 10 instances of analytics:latest
Master 1
Master 2
Master 3
analytics: 10/10
frontend: 5/5
backend: 3/3
Workers
Worker 1
Worker 2
Worker 3
Worker 4

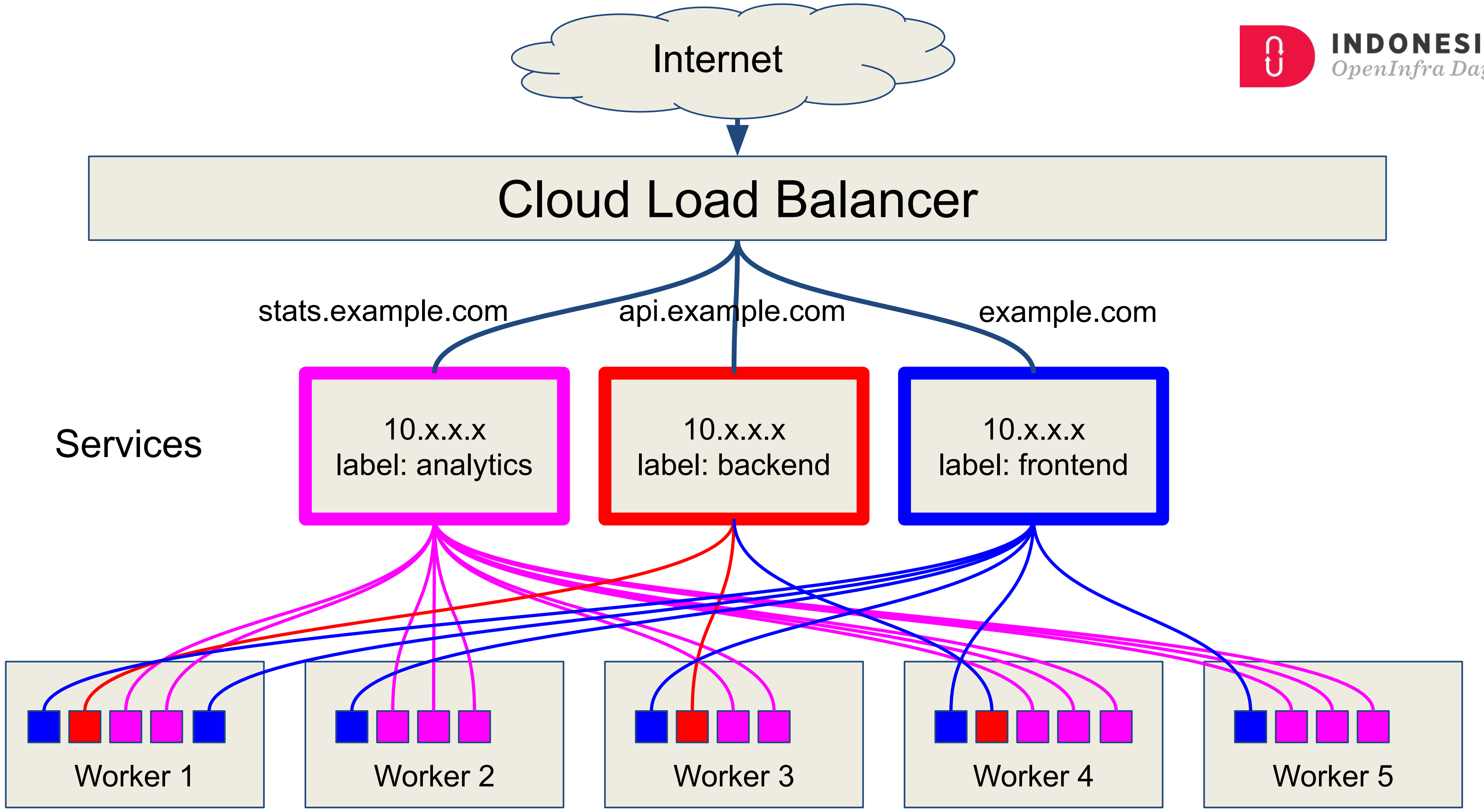

Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Services
10.x.x.x
label: analytics
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Services
10.x.x.x
label: analytics
10.x.x.x
label: backend
10.x.x.x
label: frontend
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Internet
Services
10.x.x.x
label: analytics
10.x.x.x
label: backend
10.x.x.x
label: frontend
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Internet
Cloud Load Balancer
Services
10.x.x.x
label: analytics
10.x.x.x
label: backend
10.x.x.x
label: frontend
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5


Internet
Cloud Load Balancer
stats.example.com
api.example.com
example.com
Services
10.x.x.x
label: analytics
10.x.x.x
label: backend
10.x.x.x
label: frontend
Worker 1
Worker 2
Worker 3
Worker 4
Worker 5

INDONESIA
OpenInfra Days

12-Factor App Using Kubernetes

Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation

Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes

Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

X. Dev/prod parity
Keep development, staging, and production as similar as possible

VII. Port binding
Export services via port binding

Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes

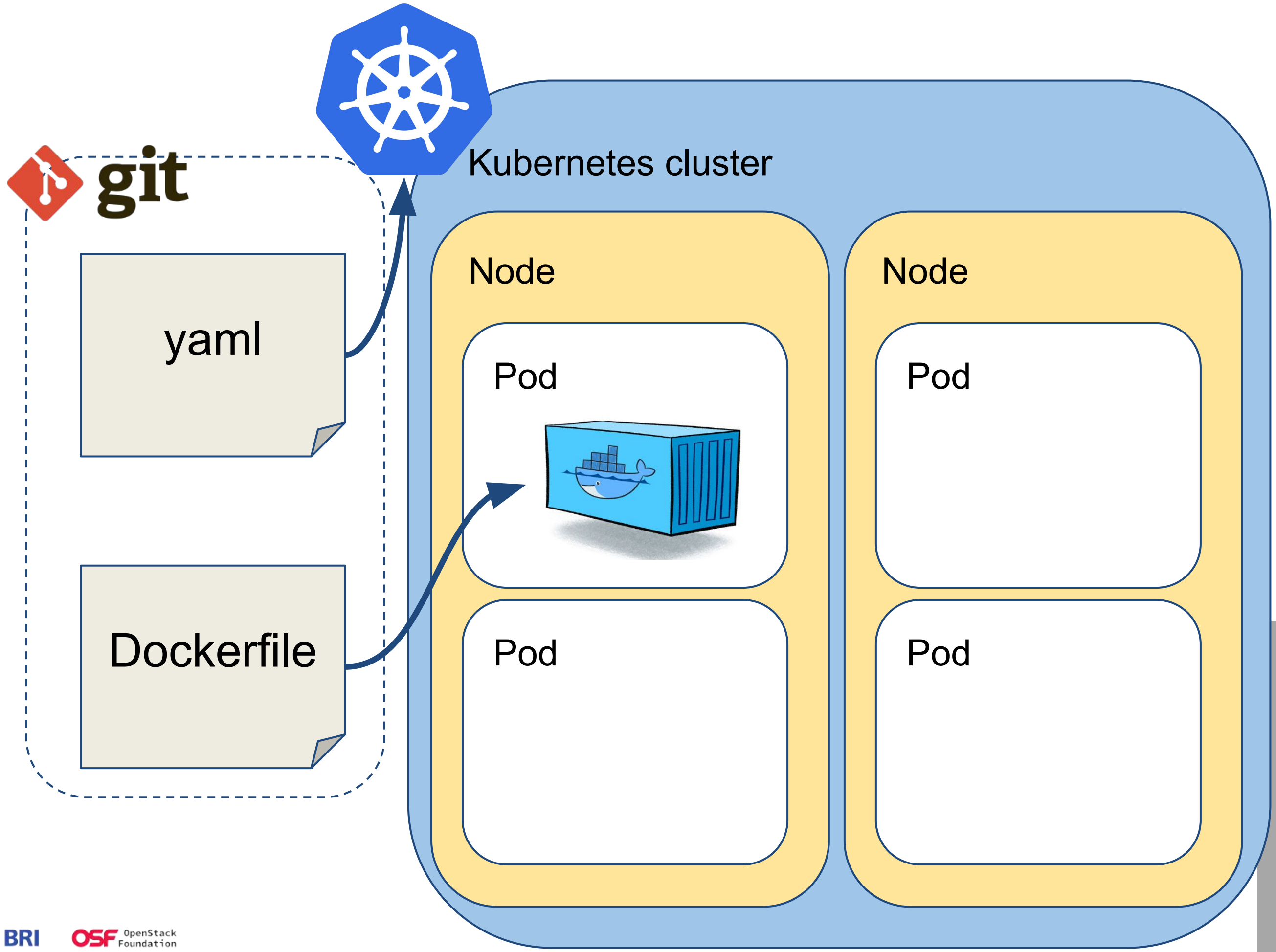
Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes



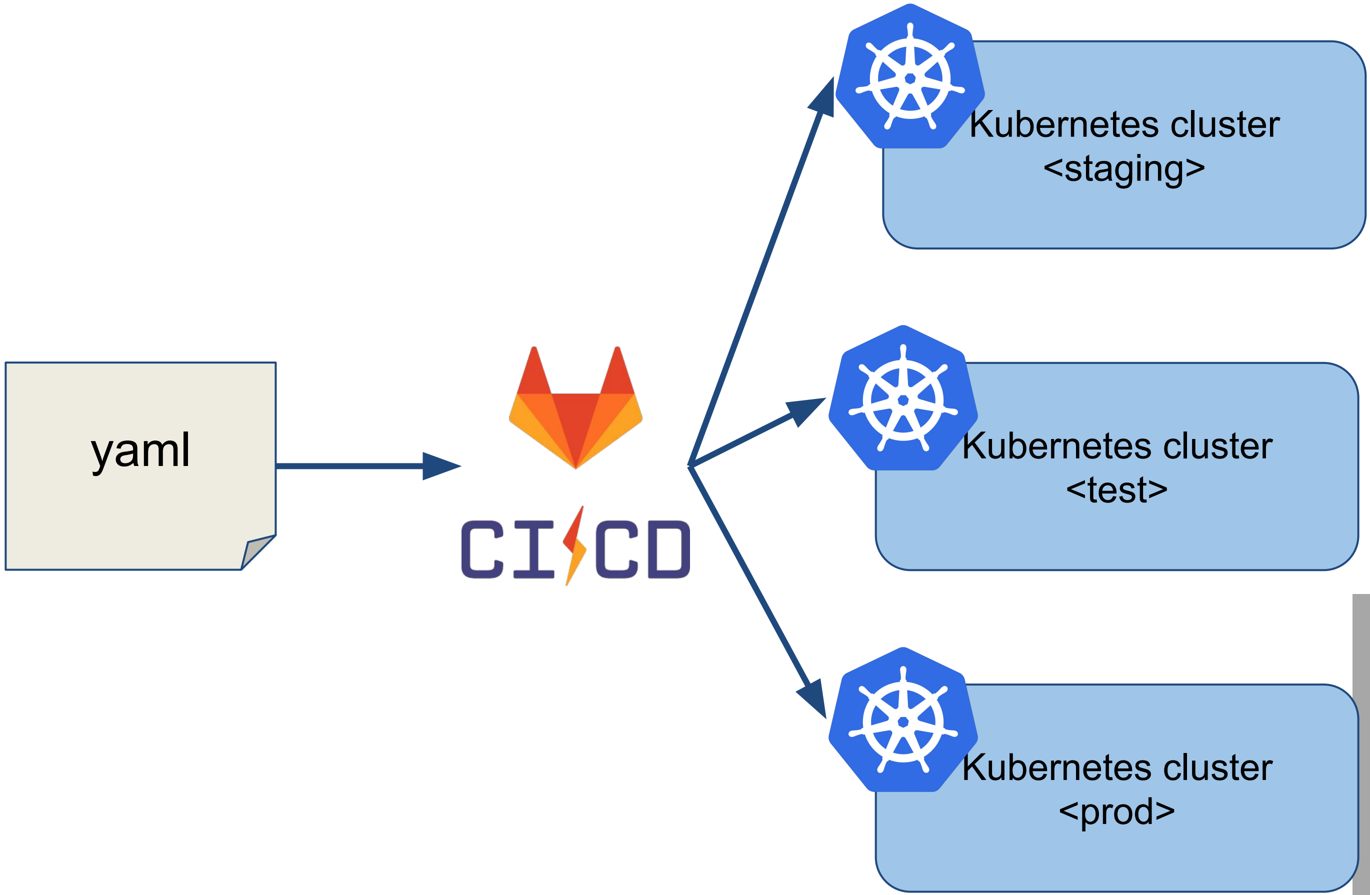
Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes


yaml
CI/CD
Kubernetes cluster <staging>
Kubernetes cluster <test>
Kubernetes cluster <prod>

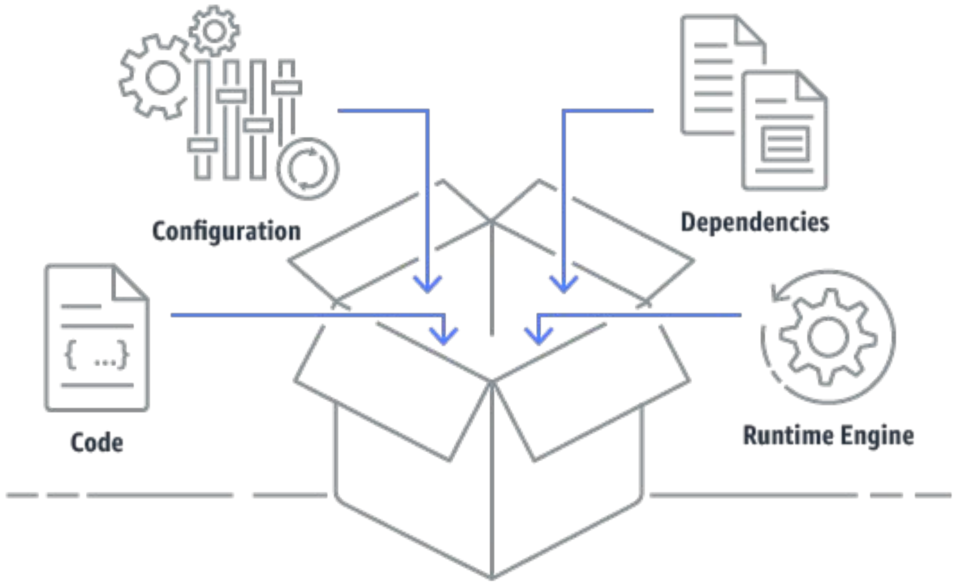
Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes



Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

VI. Processes
Execute the app as one or more stateless processes

Application code:

```
fmt.Fprintf(w, "ENV: %s\n", os.Getenv("ENV"))
fmt.Fprintf(w, "DB_HOST: %s\n", os.Getenv("DB_HOST"))
fmt.Fprintf(w, "DB_PORT: %s\n", os.Getenv("DB_PORT"))
fmt.Fprintf(w, "DB_USER: %s\n", os.Getenv("DB_USER"))
fmt.Fprintf(w, "DB_PASSWORD: %s\n", os.Getenv("DB_PASSWORD"))
```

k8s yaml:

```
containers:
        - name: demo-app
          image: asatrya/alpine-k8s-pod-lb-demo
          env:
            - name: DB_HOST
              valueFrom:
                  configMapKeyRef:
                    name: demo-configmap
                    key: DB_HOST
            - name: DB_PORT
```

Code

I. One Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

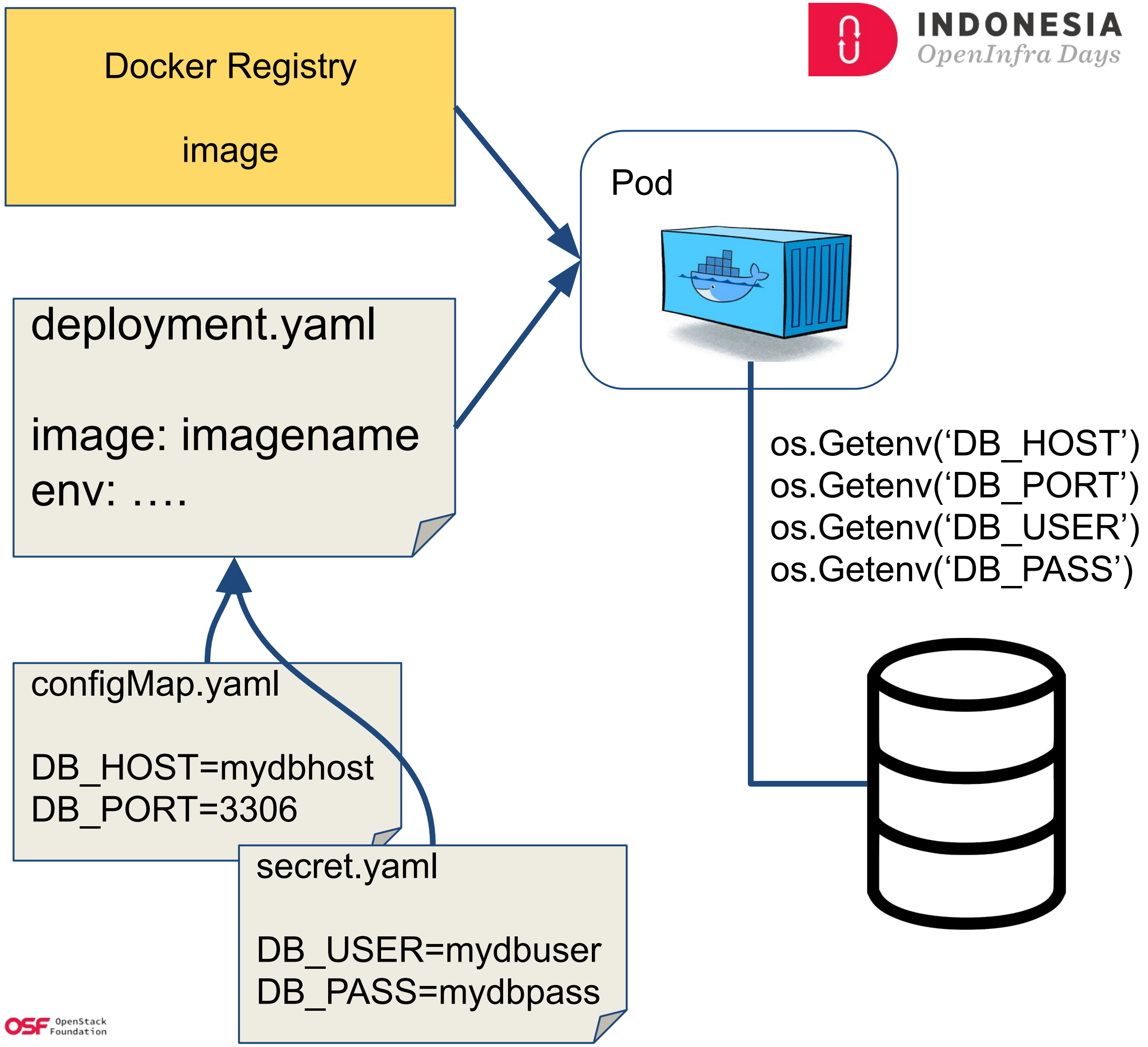
VI. Processes
Execute the app as one or more stateless processes

- Share nothing
- Do not write persistent data to node memory/filesystem

Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages
X. Dev/prod parity
Keep development, staging, and production as similar as possible
VII. Port binding
Export services via port binding



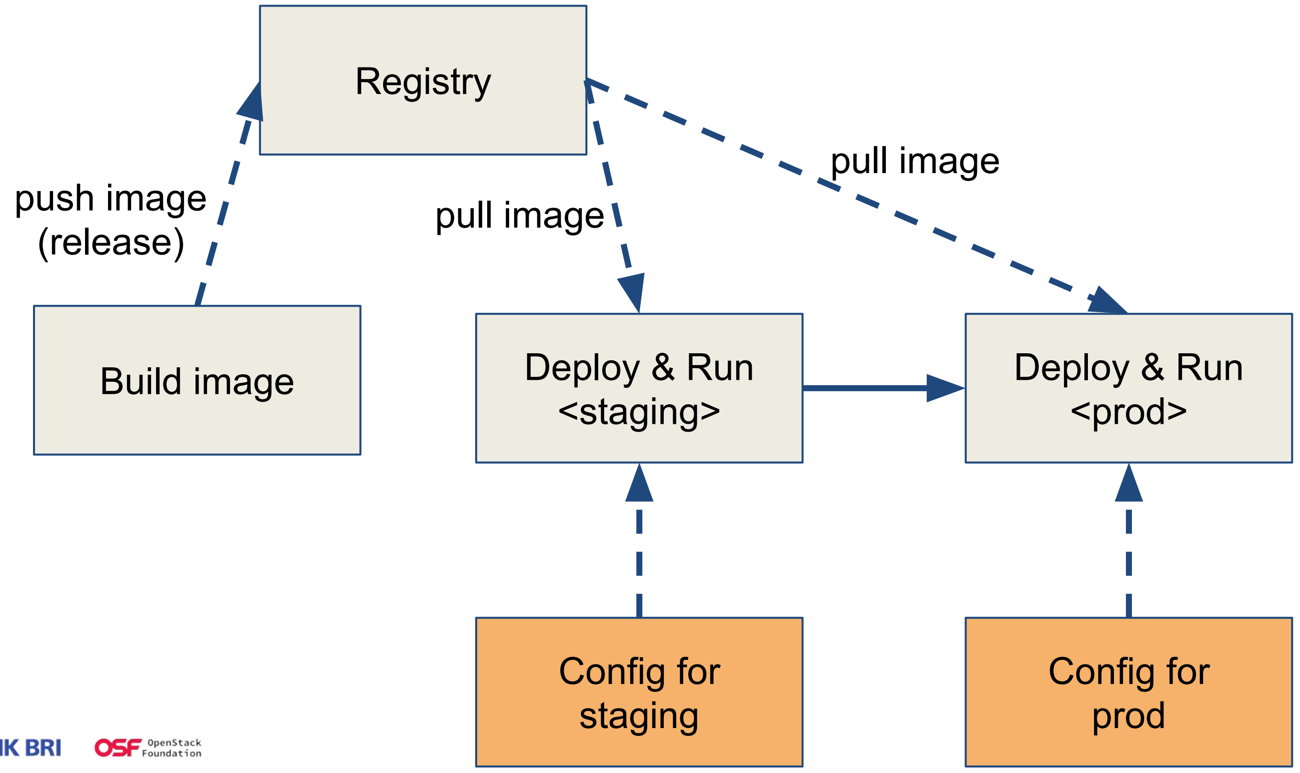
Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

X. Dev/prod parity
Keep development, staging, and production as similar as possible

VII. Port binding
Export services via port binding



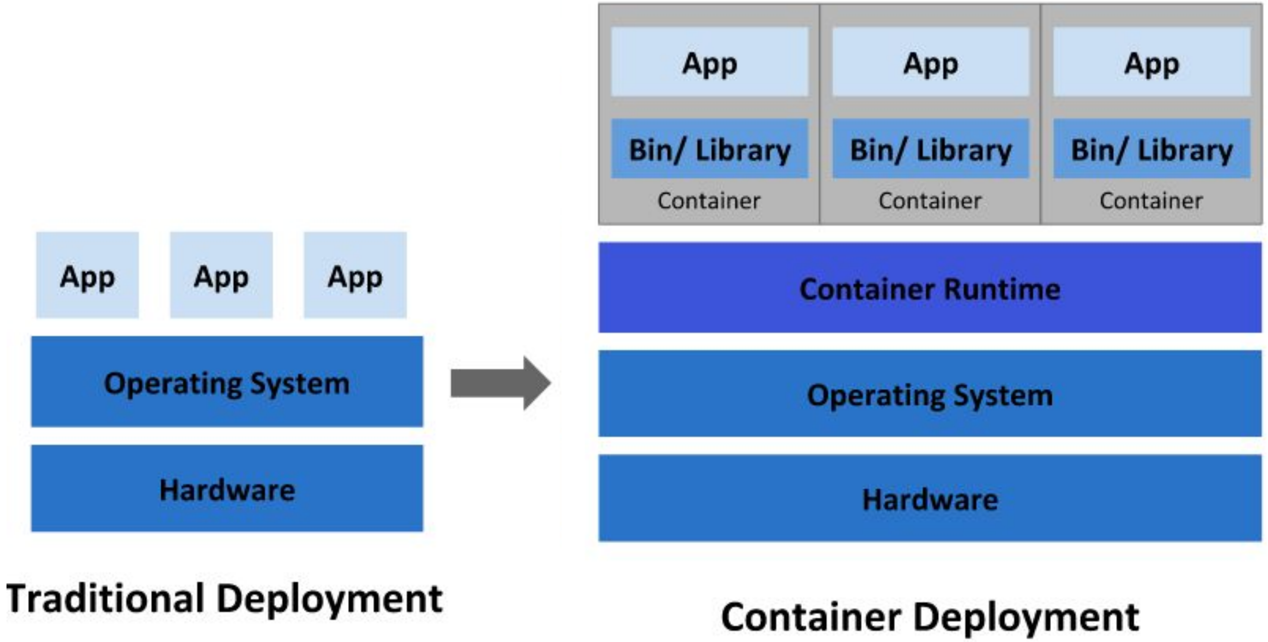
Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

X. Dev/prod parity
Keep development, staging, and production as similar as possible

VII. Port binding
Export services via port binding



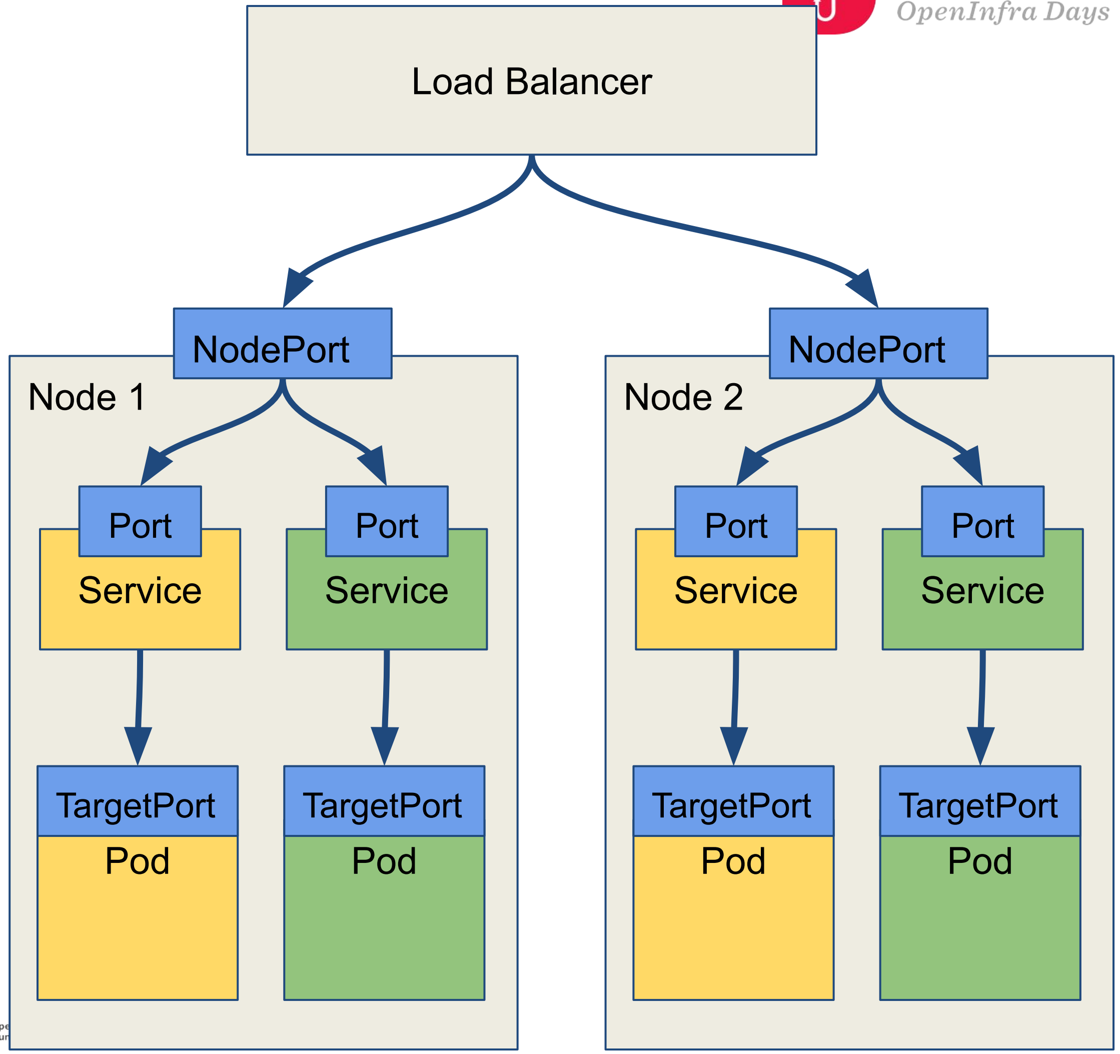
Deploy

IV. Backing services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

X. Dev/prod parity
Keep development, staging, and production as similar as possible

VII. Port binding
Export services via port binding



Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes

- Manual Scaling
  - kubectl scale
- Autoscaling
  - based on CPU utilization
  - based on custom metrics

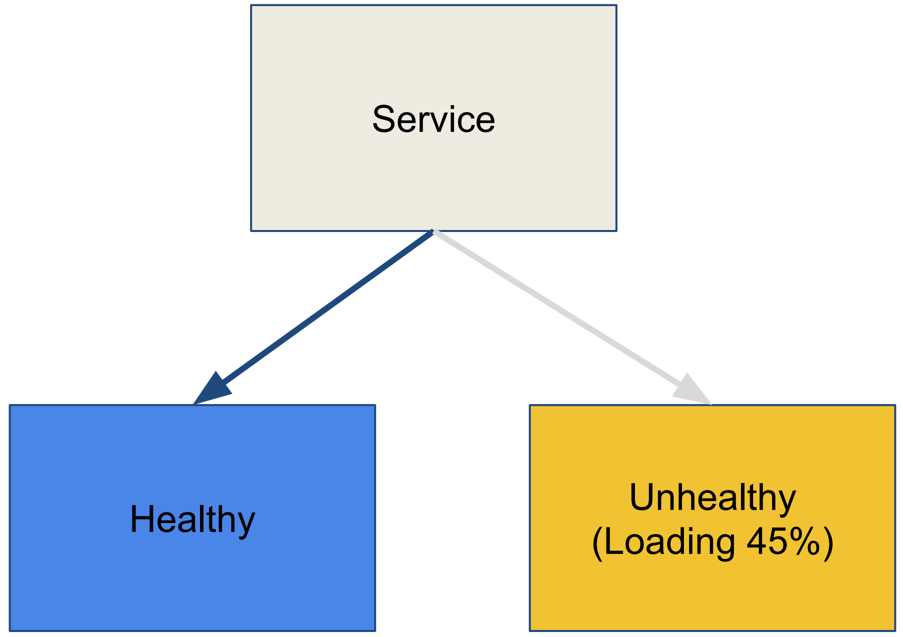
Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes



Liveness PASS
Readiness PASS

Liveness PASS
Readiness FAIL

INDONESIA
OpenInfra Days

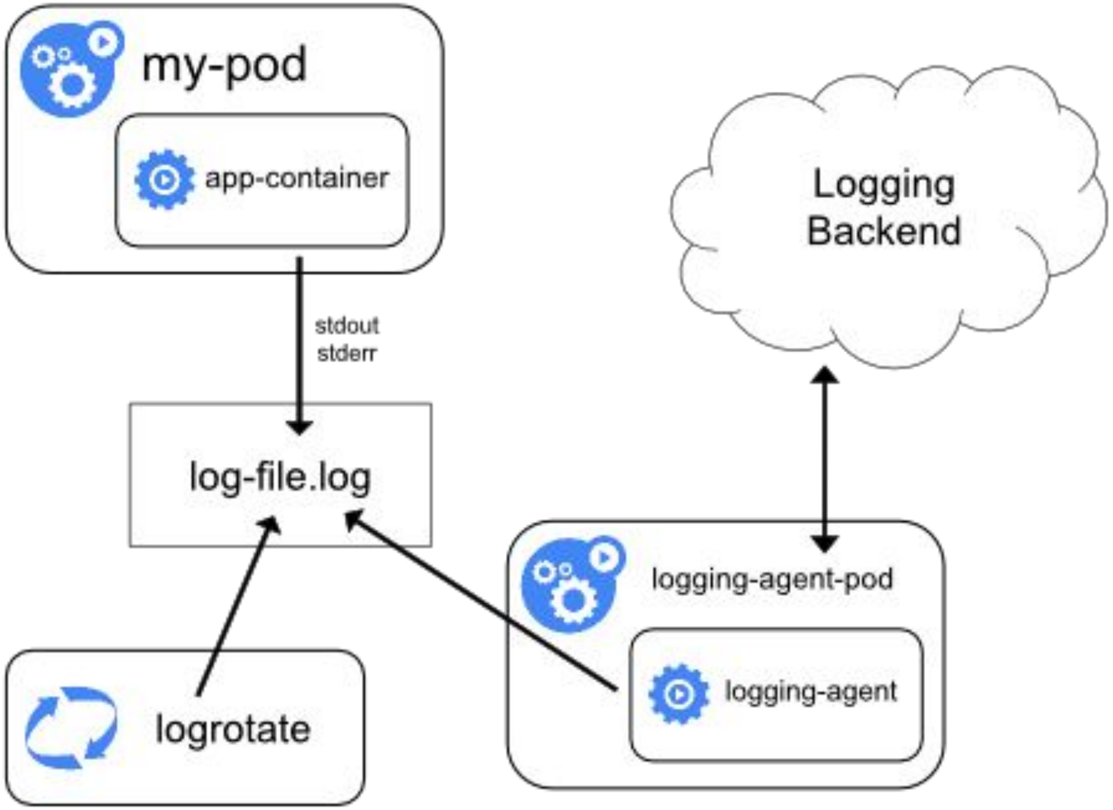
Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes



Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation

Operate

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin tasks as one-off processes

- CronJob
- Job

Demo

You can access demo source code at https://github.com/asatrya/k8s-12-factor-demo

Note: Read README first.

Summary

- Code: optimize for automation
- Deploy: portability
- Operate: scalability, resiliency

INDONESIA
OpenInfra Days

Thank you!

Biznet GioCloud Biznet Mellanox TECHNOLOGIES BANK BRI OSF OpenStack Foundation