Kubernetes at Bukalapak

An SRE perspective

7 February 2019

Christoper Hans

SRE @ Bukalapak

ABOUT ME

An avid Unix user, (ba|z)sh warrior, long-time musician and System Administrator.

Previously worked at Accenture, now bringing all the good things from consulting to Bukalapak!

History of Our Deployment Stack

History of our Deployment Stack

01 Manual Deployment

Old-fashioned &
self-managed deployment

History of our Deployment Stack

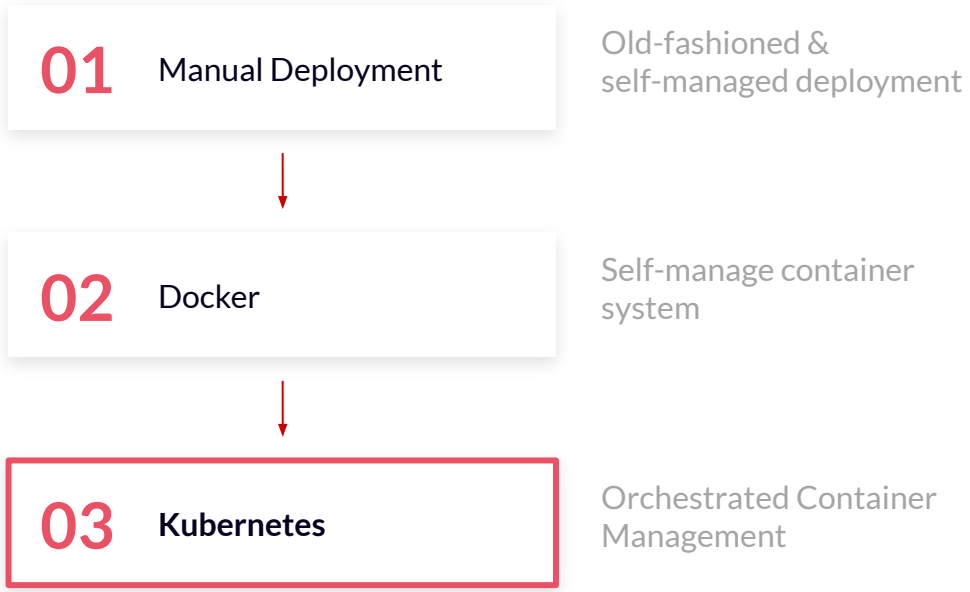
01 Manual Deployment

Old-fashioned & self-managed deployment

02 Docker

Self-manage container system

History of our Deployment Stack



Why Kubernetes?

Managed Container Clusters

- Service Discovery & Load Balancing
- Network & Storage Orchestration
- Secret & Configuration Management

Resilient

- Self Healing
- High Availability
- Stops bad deployment from progressing

Infrastructure Abstraction

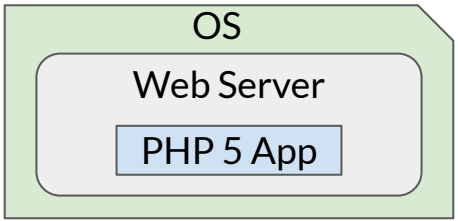
- Resource Pooling / Easier Provisioning
- Infra & Apps Decoupled
- Simple Horizontal Scaling

SRE Perspective

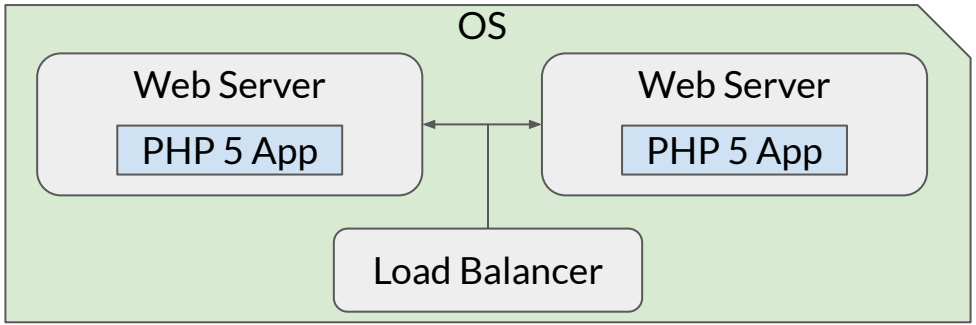
Infrastructure Abstraction

Imagine **replicating** simple PHP 5 App which only has static pages…

What steps needs to be done?

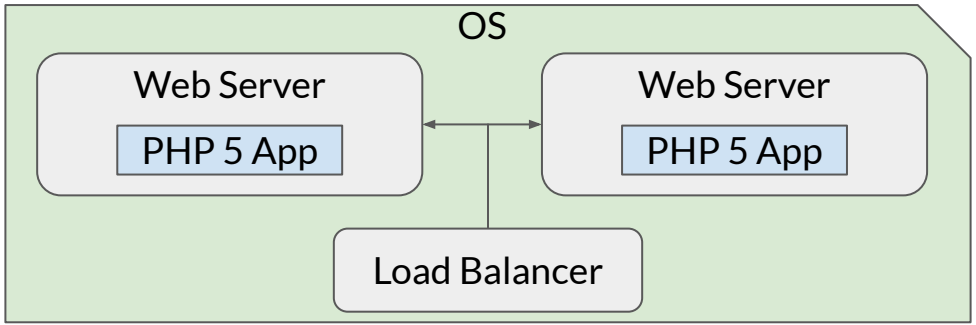


In Manual Deployment



1. Install new instance of web server
2. Install PHP5 App
3. Add Load Balancing
4. Register each instance to Load Balancer

In Manual Deployment



1. Install new instance of web server
2. Install PHP5 App
3. Add Load Balancing
4. Register each instance to Load Balancer

Not Pictured:

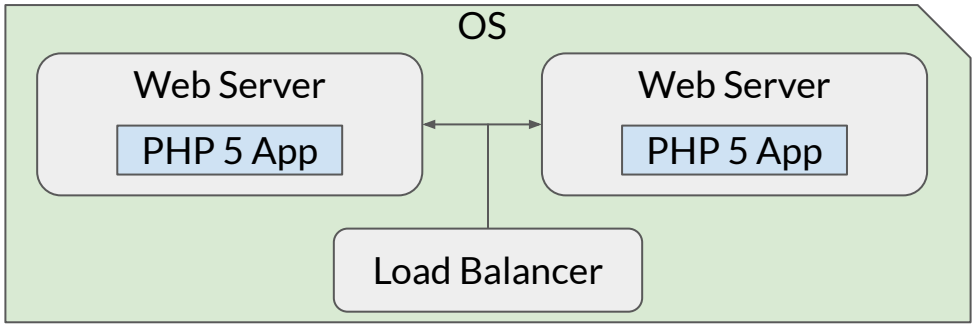
- Port **MUST NOT** conflict between both web server and Load Balancer

In Kubernetes?

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-nginx-app
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: php-nginx-app
  template:
    metadata:
      labels:
        app: php-nginx-app
    spec:
      containers:
      - name: php-nginx-app
        image: php-nginx-app:1.0.1
        ports:
        - containerPort: 80
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

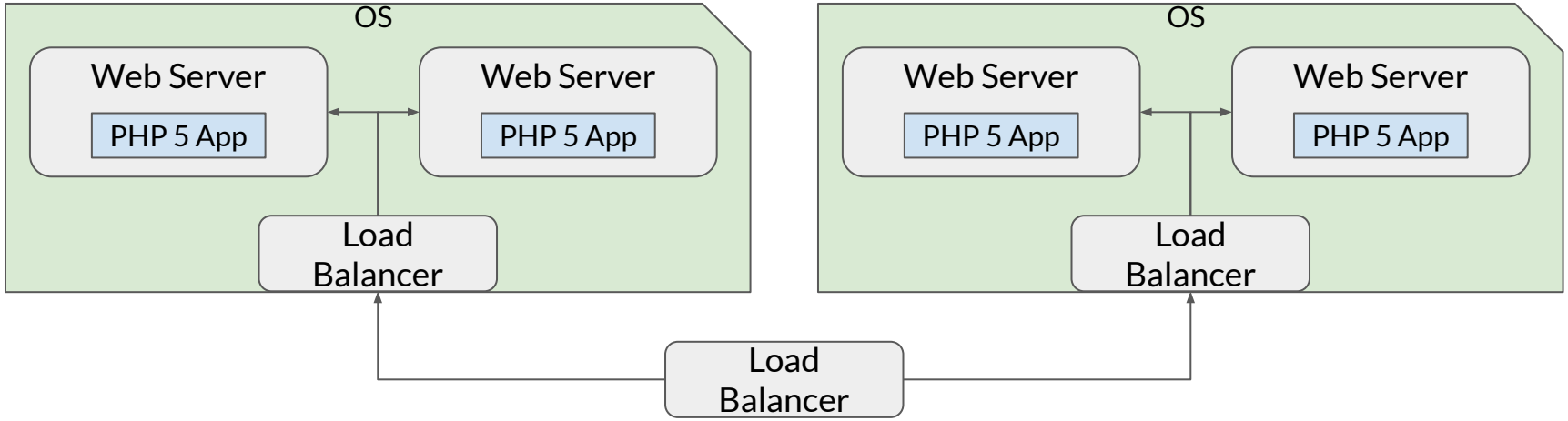
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-nginx-app
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: php-nginx-app
  template:
    metadata:
      labels:
        app: php-nginx-app
    spec:
      containers:
      - name: php-nginx-app
        image: php-nginx-app:1.0.1
        ports:
        - containerPort: 80
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

* Note: **Replica number** is only limited by **Resource (CPU, RAM)** of Kubernetes Cluster



What if our OS had 100% Resource Utilization (CPU, RAM) and we need to add more replicas?

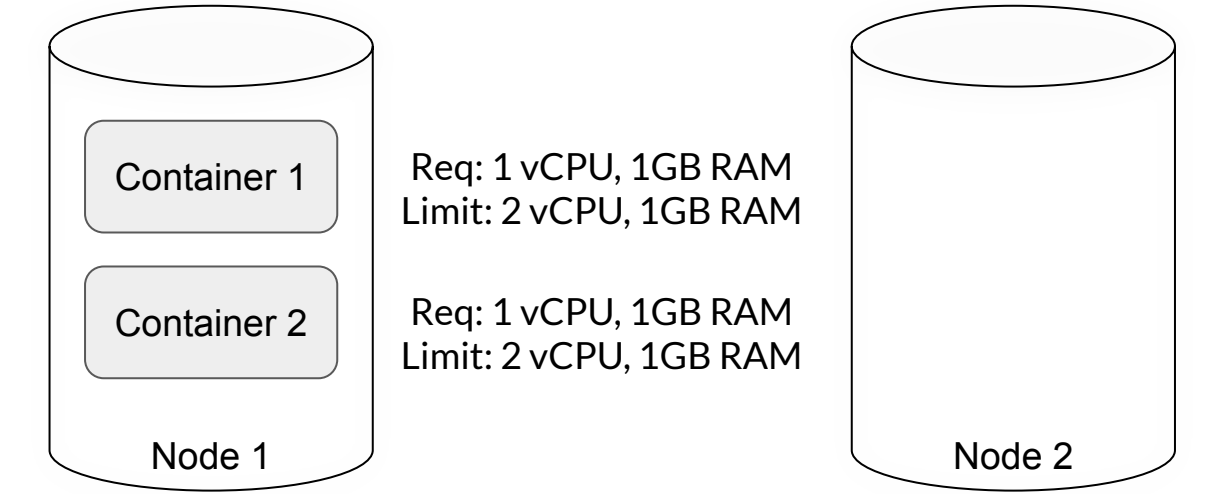
In Manual Deployment



1. Create new OS
2. Install new instances of web server
3. Install PHP5 App
4. Create Load Balancer for new OS
5. Create main Load Balancer

In Kubernetes?

No Actions needed!

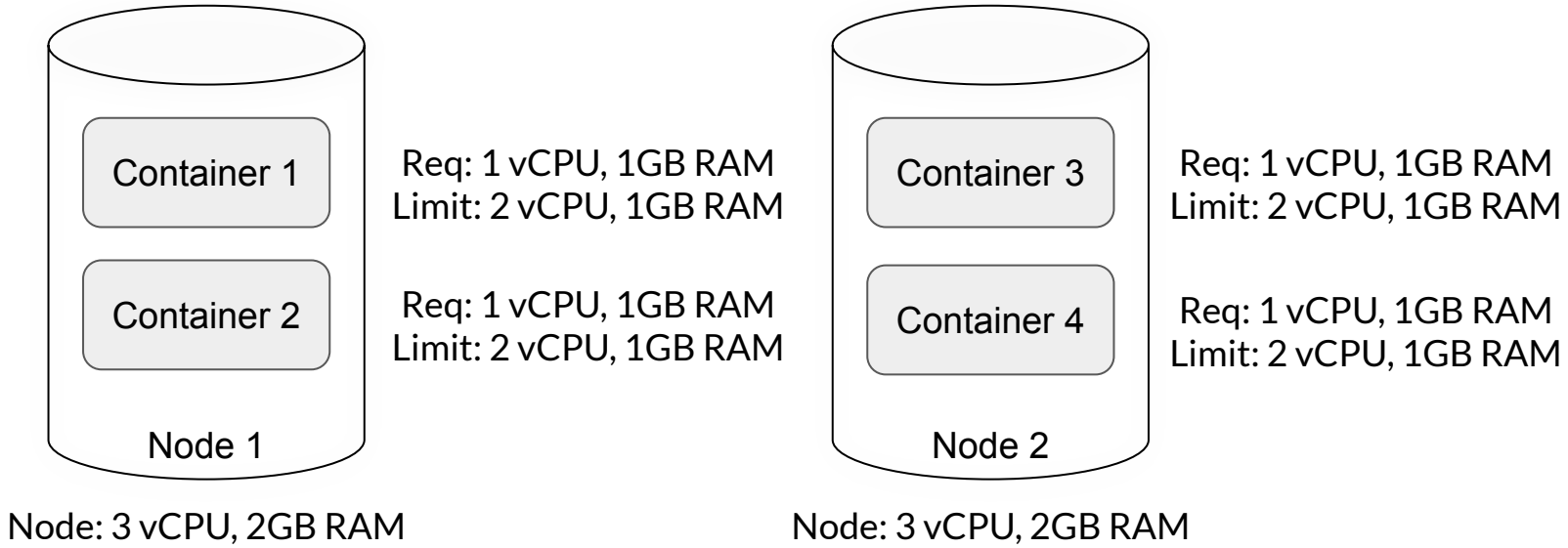


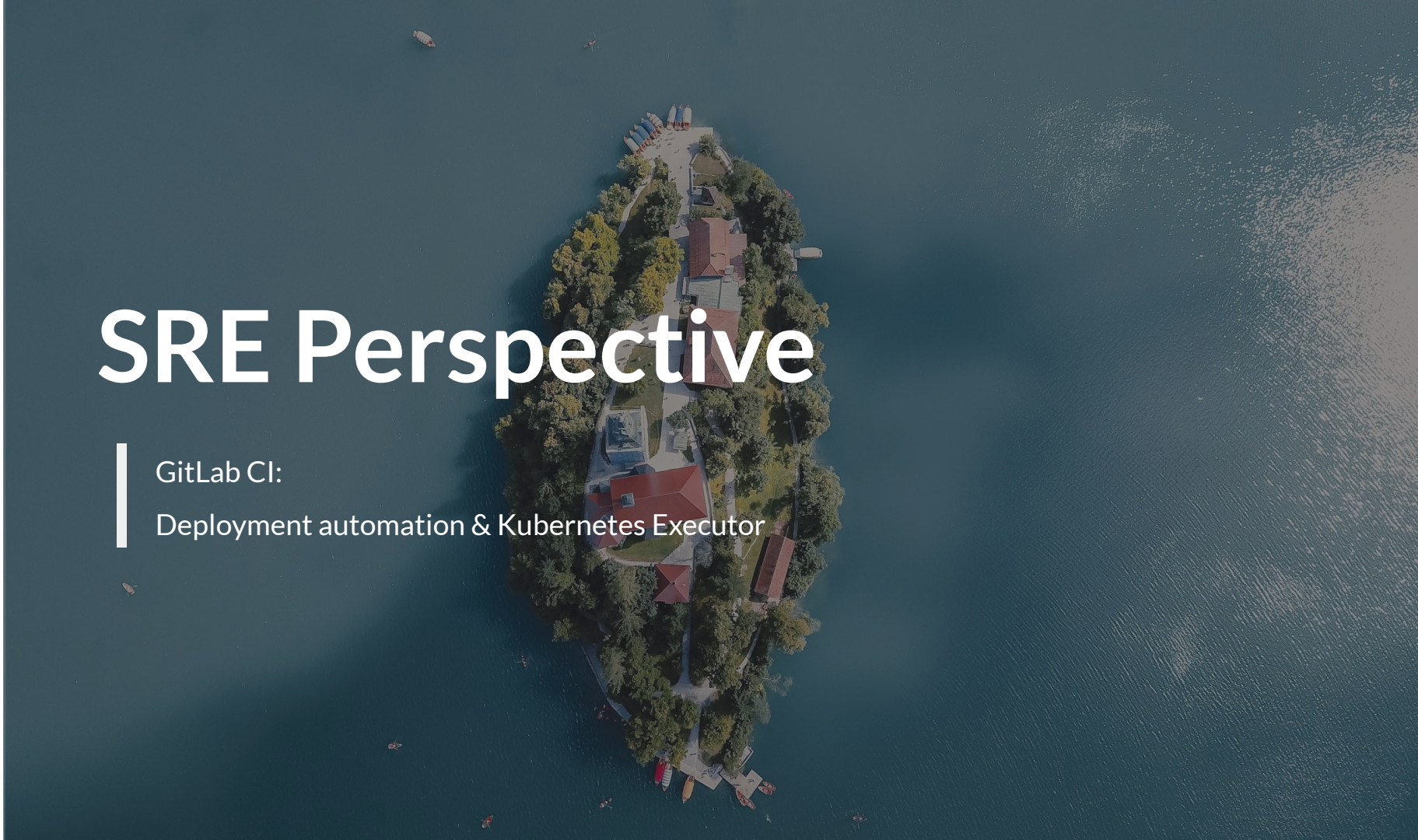
Node: 3 vCPU, 2GB RAM

Node: 3 vCPU, 2GB RAM

In Kubernetes?

No Actions needed!

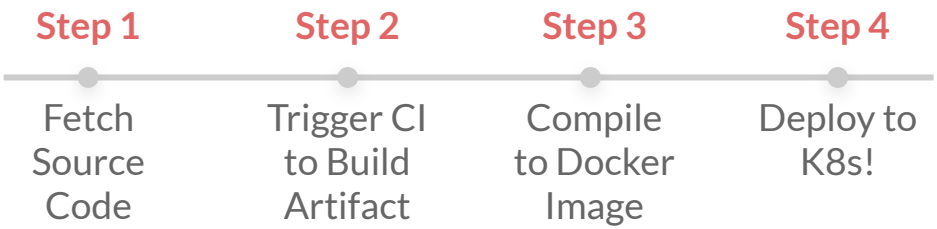


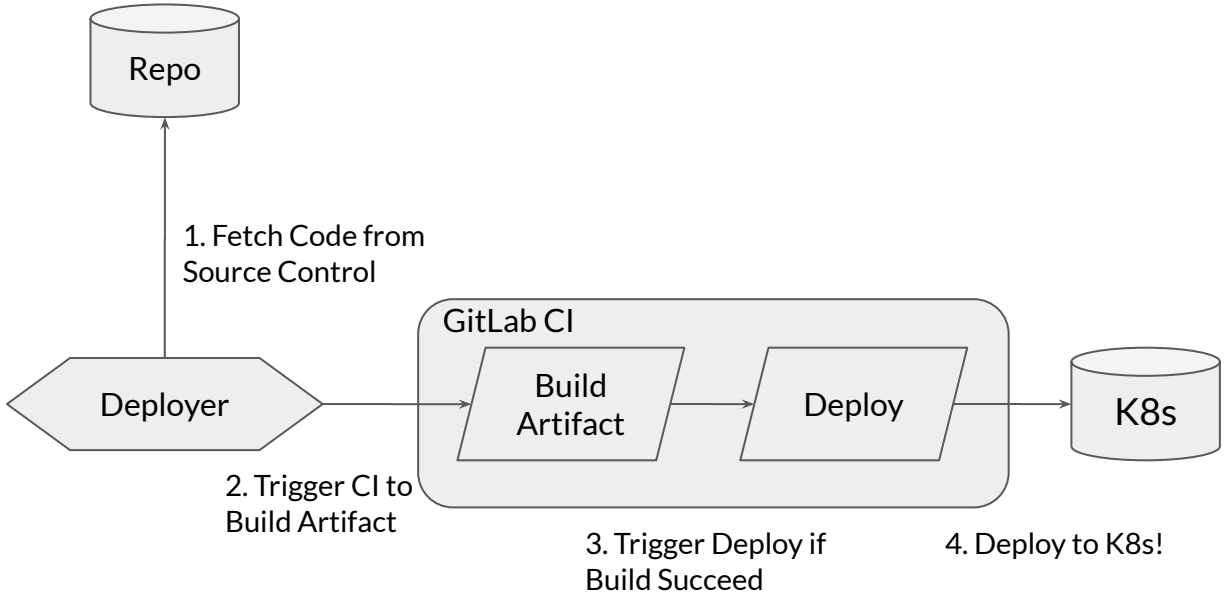


SRE Perspective

GitLab CI:

Deployment automation & Kubernetes Executor


Step 1
Step 2
Step 3
Step 4
Fetch Source Code
Trigger CI to Build Artifact
Compile to Docker Image
Deploy to K8s!


Repo
1. Fetch Code from Source Control
GitLab CI
Deployer
Build Artifact
Deploy
K8s
2. Trigger CI to Build Artifact
3. Trigger Deploy if Build Succeed
4. Deploy to K8s!

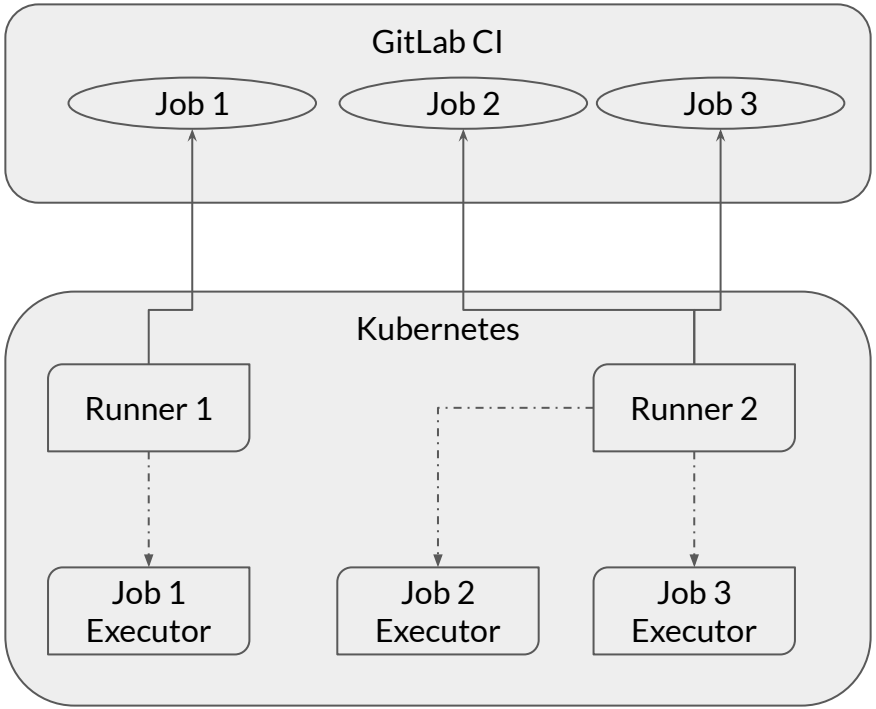
GitLab CI Runner Kubernetes Executor

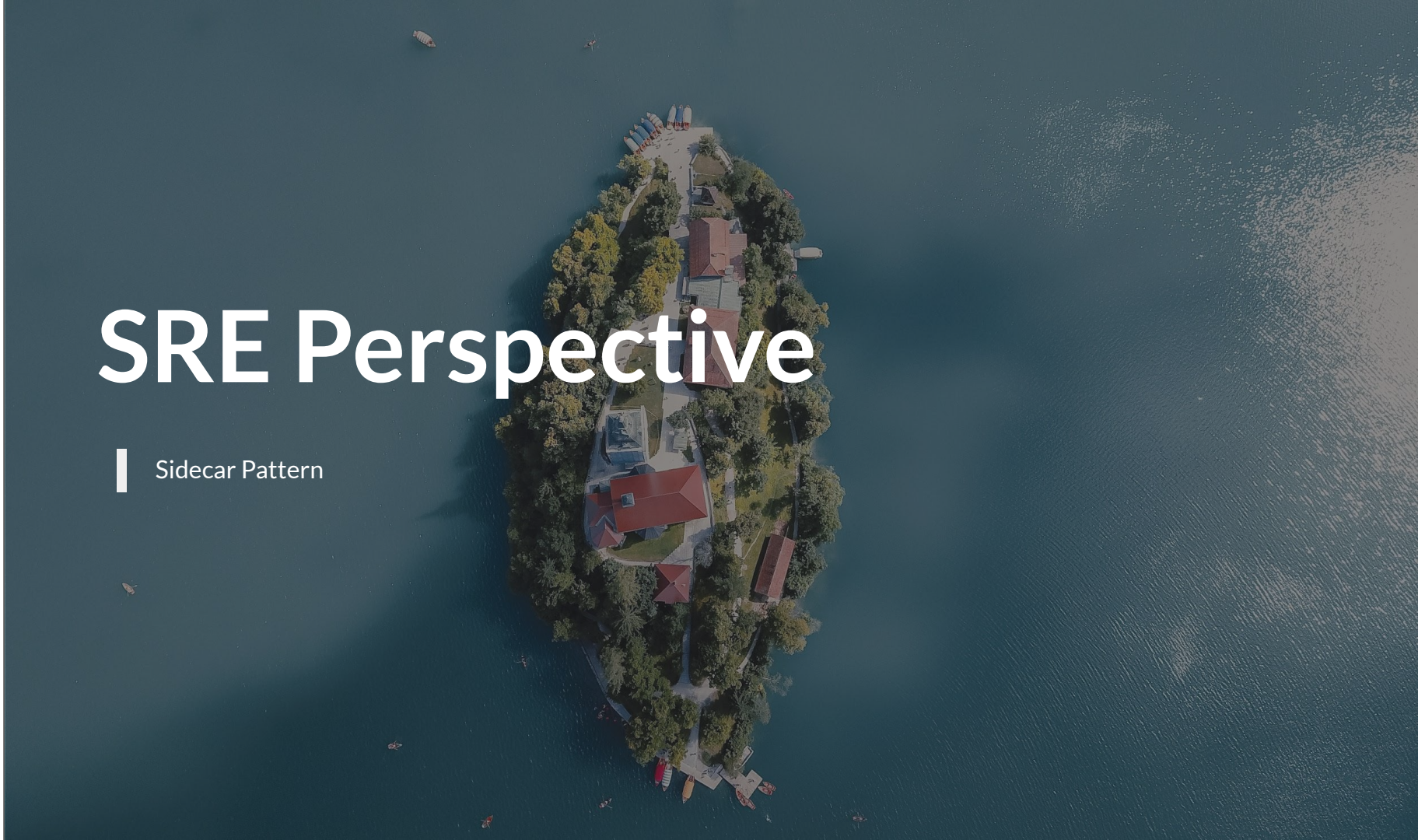
Horizontal Scaling

Adding Resource to GitLab Runner when using Kubernetes Executor is as easy as adding new Kubernetes nodes!

No Extra Node Needed!

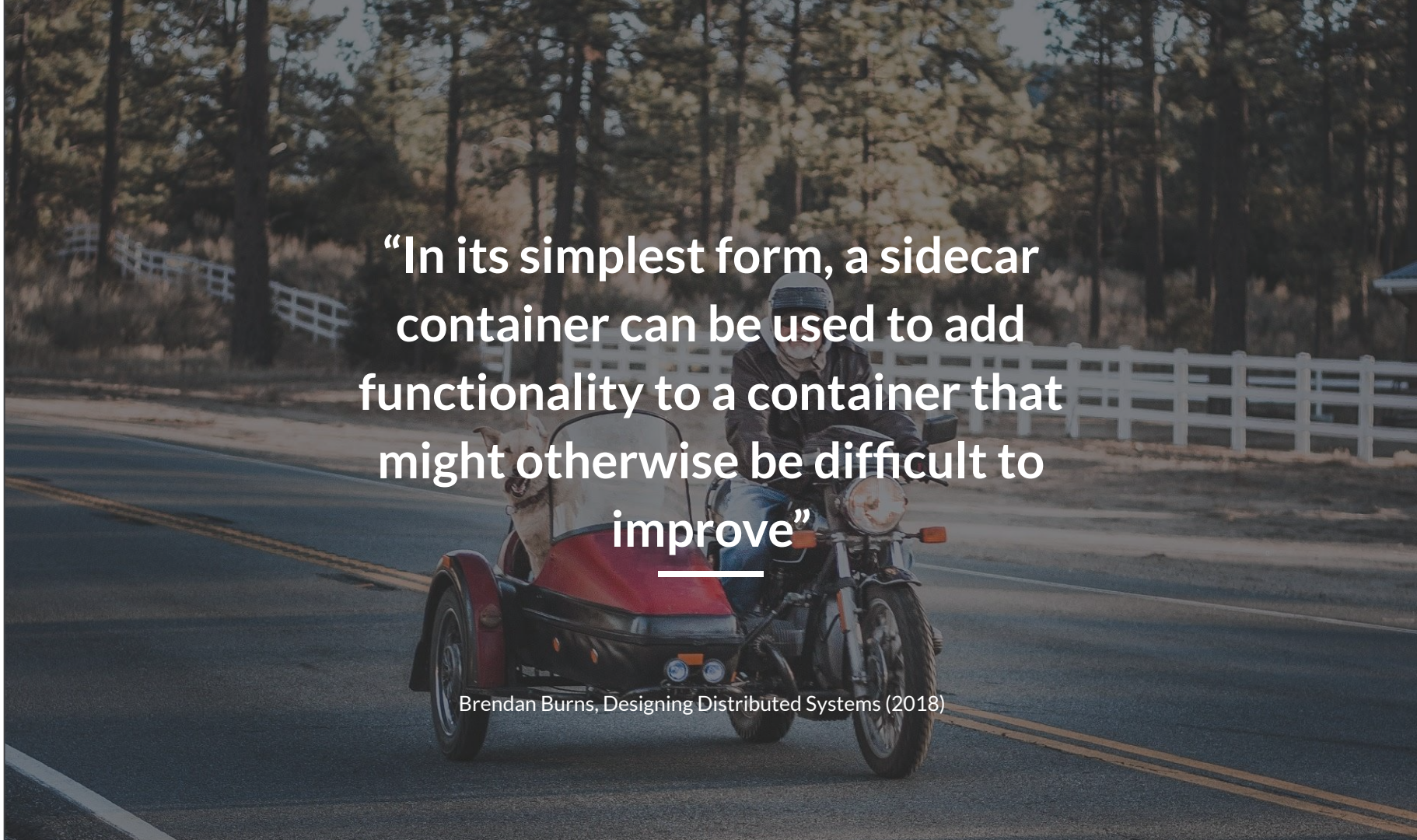
No specialized node only for GitLab Runner - Use current Kubernetes Cluster as long as resource are available!


GitLab CI
Job 1
Job 2
Job 3
Kubernetes
Runner 1
Runner 2
Job 1
Executor
Job 2
Executor
Job 3
Executor



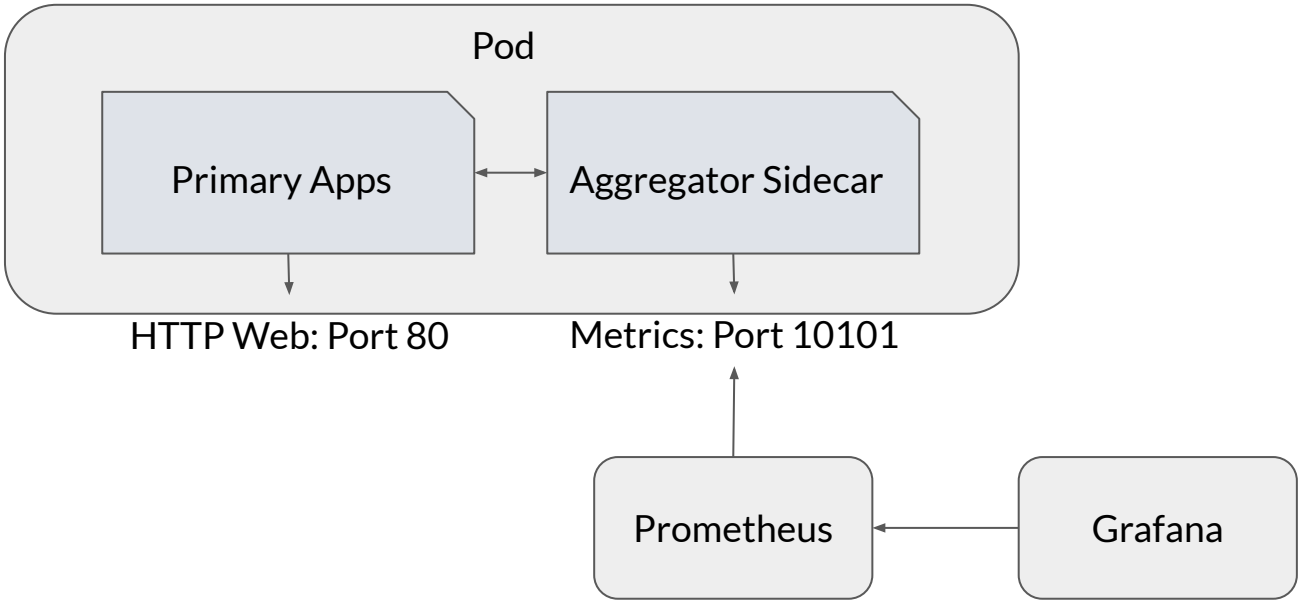
SRE Perspective

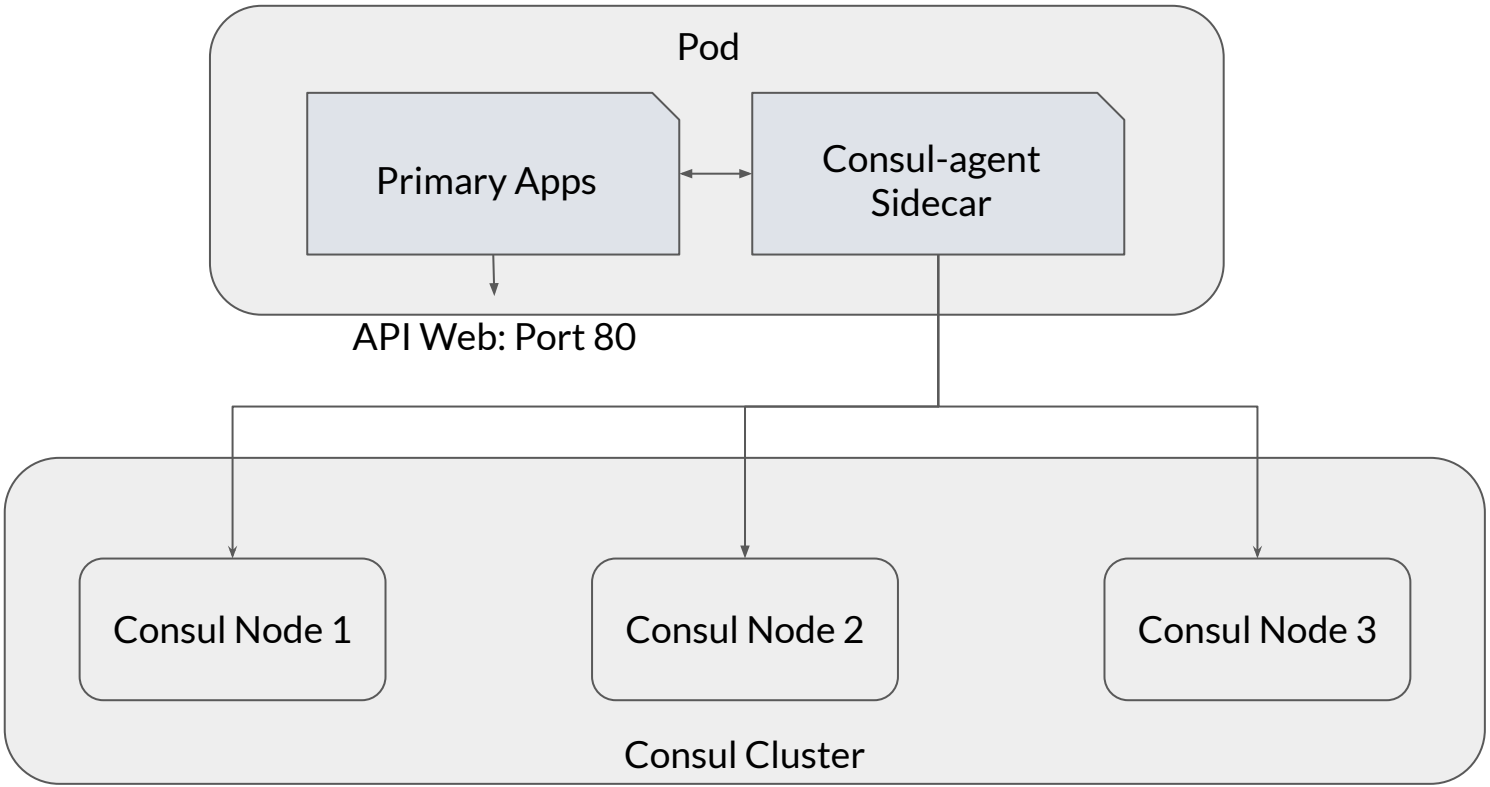
Sidecar Pattern



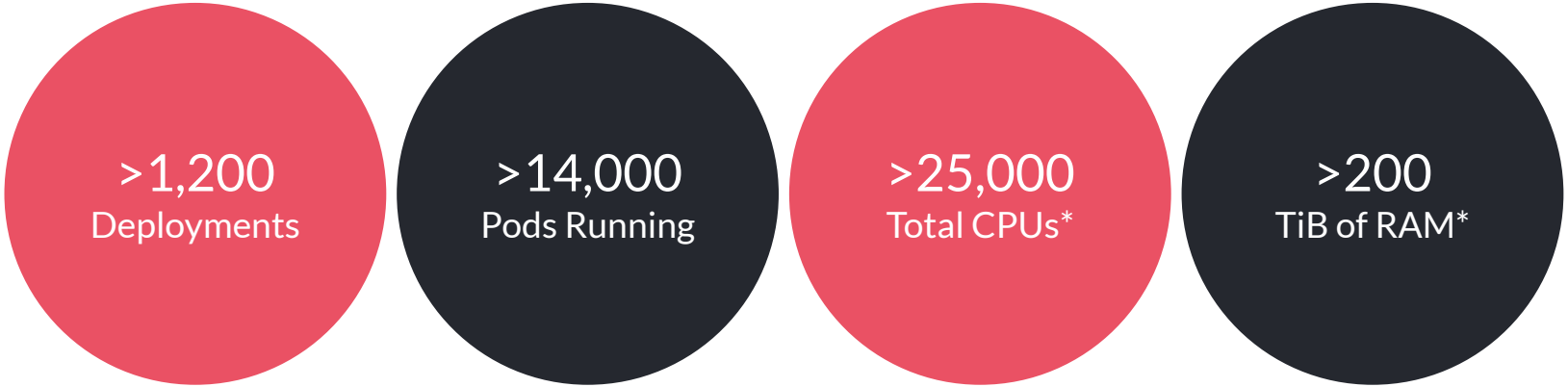
“In its simplest form, a sidecar container can be used to add functionality to a container that might otherwise be difficult to improve”

Brendan Burns, Designing Distributed Systems (2018)


Pod
Primary Apps
Aggregator Sidecar
HTTP Web: Port 80
Metrics: Port 10101
Prometheus
Grafana


Pod
Primary Apps
Consul-agent
Sidecar
API Web: Port 80
Consul Node 1
Consul Node 2
Consul Node 3
Consul Cluster

Bukalapak Kubernetes by the numbers!



* Excluding our Database, Redis and Elasticsearch Resource

Thank you

We’re Hiring!

Send Your Resume to
christoper.hans@bukalapak.com