Create a production-ready application

Sponsor by

Create a production-ready application

เฮ็ดเวิร์คช๊อปให้แอปพร้อมไช้งานบนโปรดัคชั่น

Sponsor by

Jumpbox®

Tech Passion | Sharing | Society

Jumpbox Academy

- ACADEMY
- TECH COMMUNITY
- SOCIAL IMPACT

> **We help ignite your tech passion.**
>
> **Navigate your Learning paths.**
>
> **‘til impacts the society.**
>
> *Jumpbox*

Jumpbox®

JoJo Jumpbox

Cloud Native

Cloud Native

for Product

Jumpbox®

The Jar

The Jar

Thailand’s local tool that can store every people

The Jar

Thailand’s local tool that can store every people

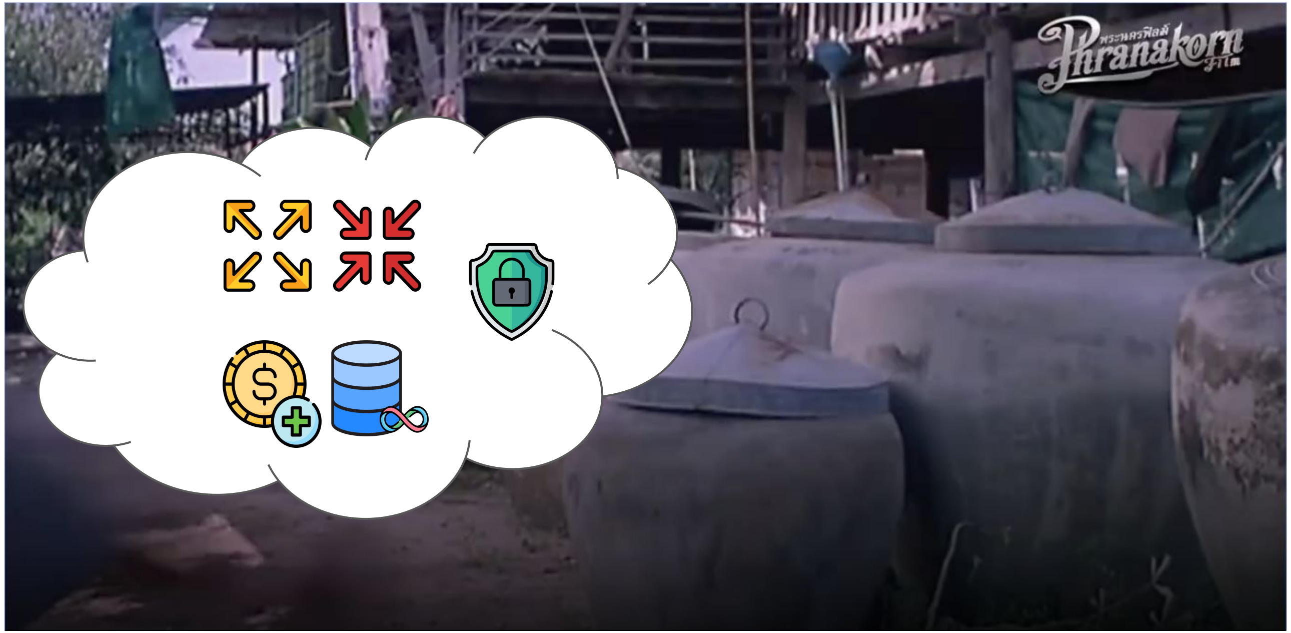
who run away from ผีปอบ

Adapt Thai Local Tools with Cloud Native

Implement **Jar** + **Cloud Native**

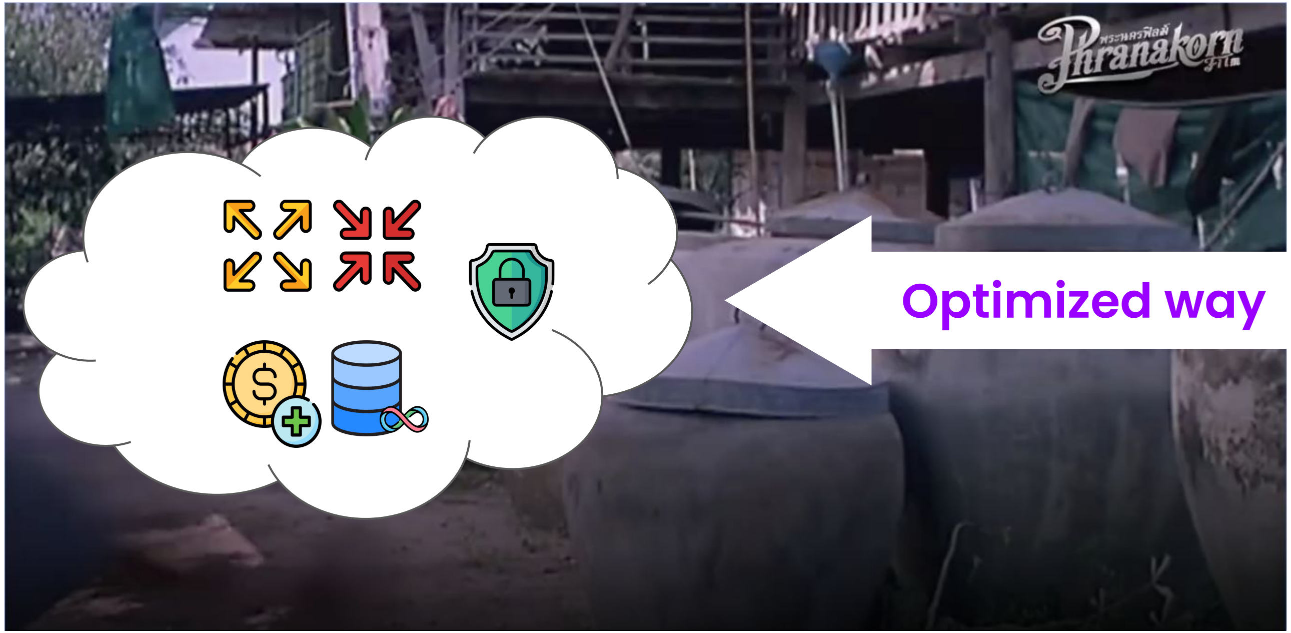
Adapt **Thai Local Tools** with Cloud Native

Implement **Jar** + **Cloud Native**



Adapt Thai Local Tools with Cloud Native

Implement **Jar** + **Cloud Native**



Cloud Native

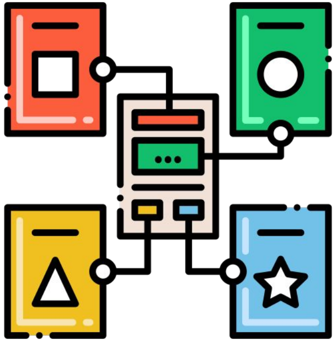
for Application

Jumpbox®

Cloud Native for Application

Jumpbox®

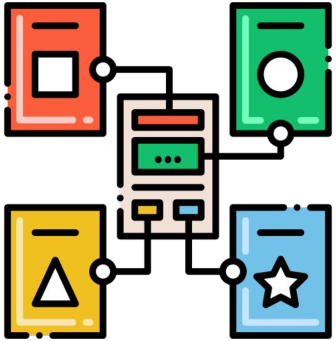
Cloud Native for Application



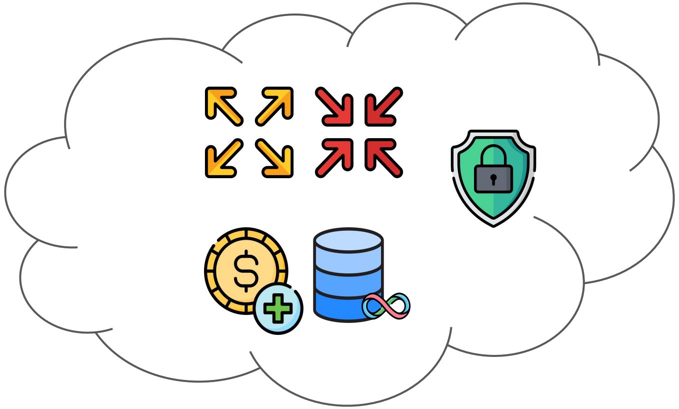
Application

Jumpbox®

Cloud Native for Application



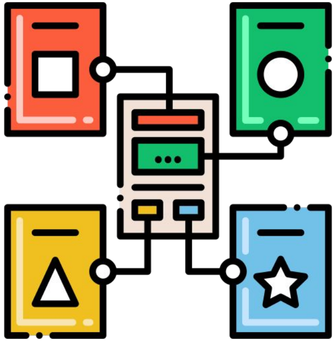
Application



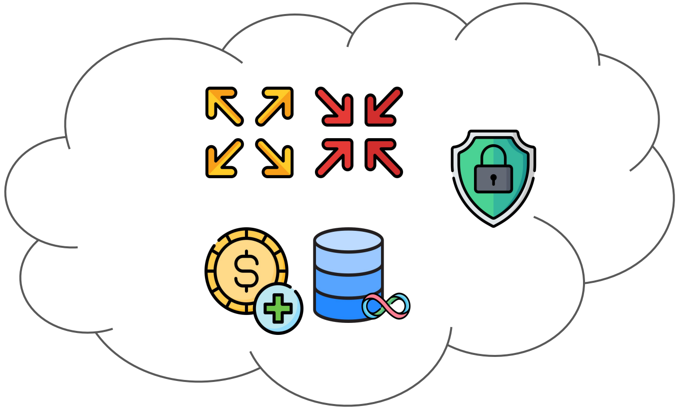
Cloud Attribute

Jumpbox®

Cloud Native for Application



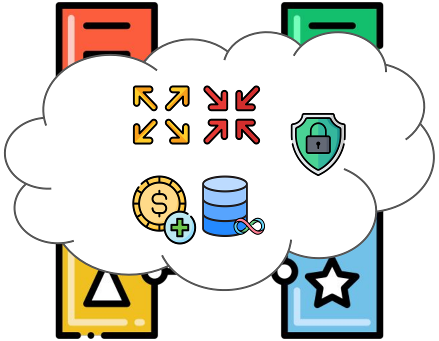
Application



Cloud Attribute

Jumpbox®

Cloud Native for Application



Cloud Native Application

Thank you

Jumpbox®

Thank you

Good Night

Jumpbox®

Just Kidding

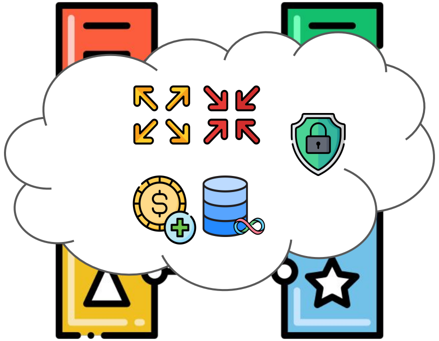
Let’s Get back to

Let’s Get back to

Cloud Native for Application

Cloud Native for Application

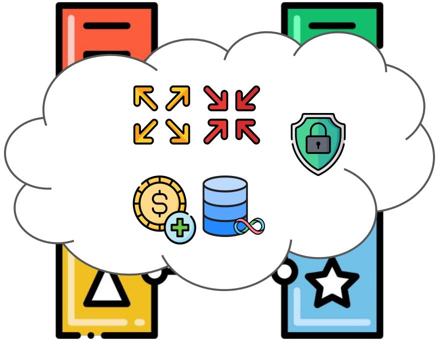
Cloud Native for Application



Cloud Native Application

Jumpbox®

Cloud Native for Application

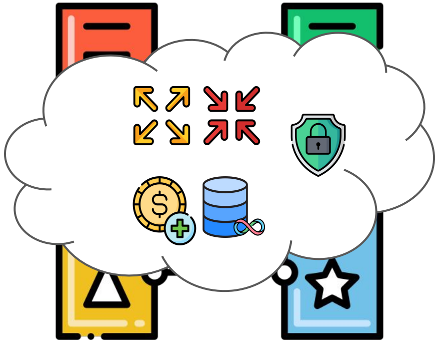


Cloud Native Application

Containerization

by container hardening way

Cloud Native for Application



Cloud Native Application

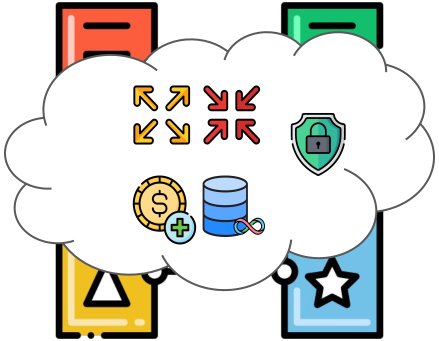
Containerization

by container hardening way

Live Reload Configuration

by adjust centralized configuration

Cloud Native for Application



Cloud Native Application

Containerization

by container hardening way

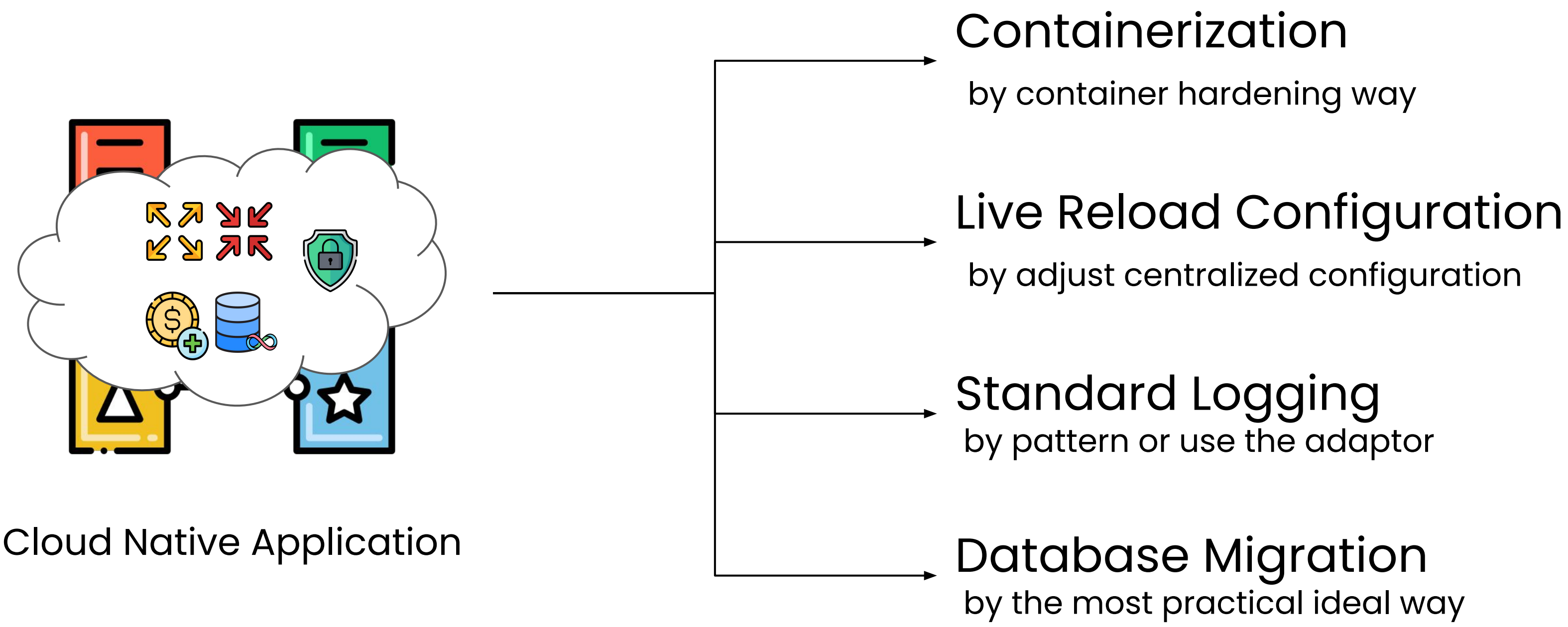
Live Reload Configuration

by adjust centralized configuration

Standard Logging

by pattern or use the adaptor

Cloud Native for Application



Cloud Native Application

Containerization

by container hardening way

Live Reload Configuration

by adjust centralized configuration

Standard Logging

by pattern or use the adaptor

Database Migration

by the most practical ideal way

Cloud Native for Application

Containerization

Disclaimer

by the most practical ideal way

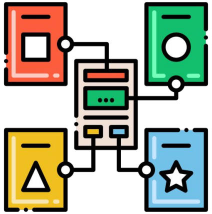
Containerization

Containerization

With Distroless image

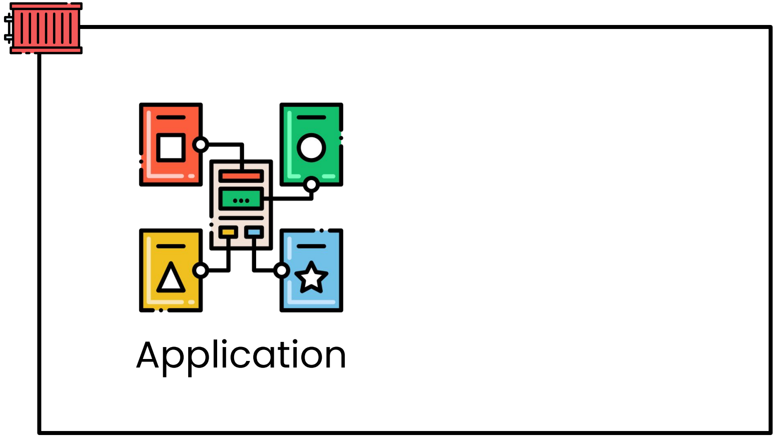
Containerization with Distroless

Containerization with Distroless

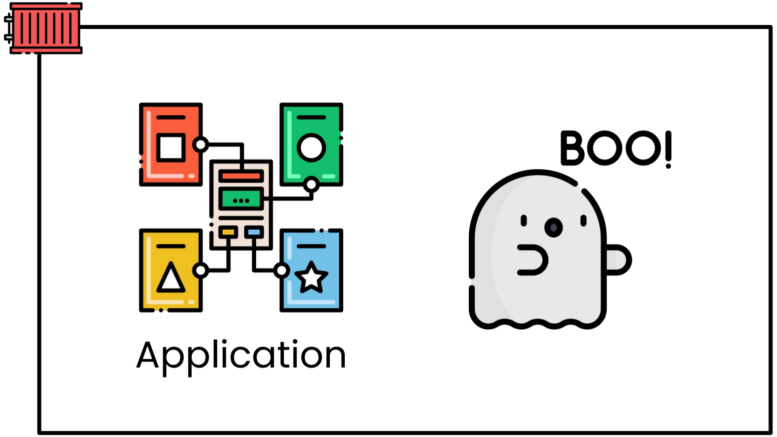


Application

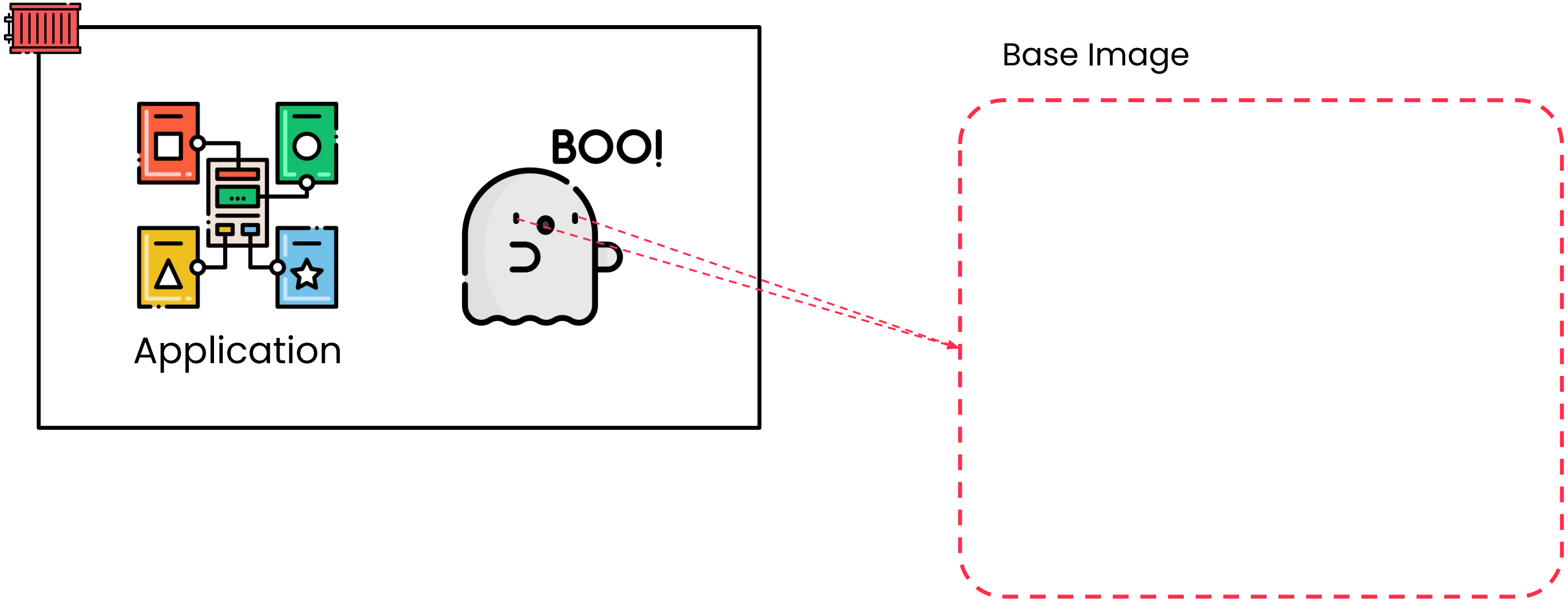
Containerization with Distroless



Containerization with Distroless

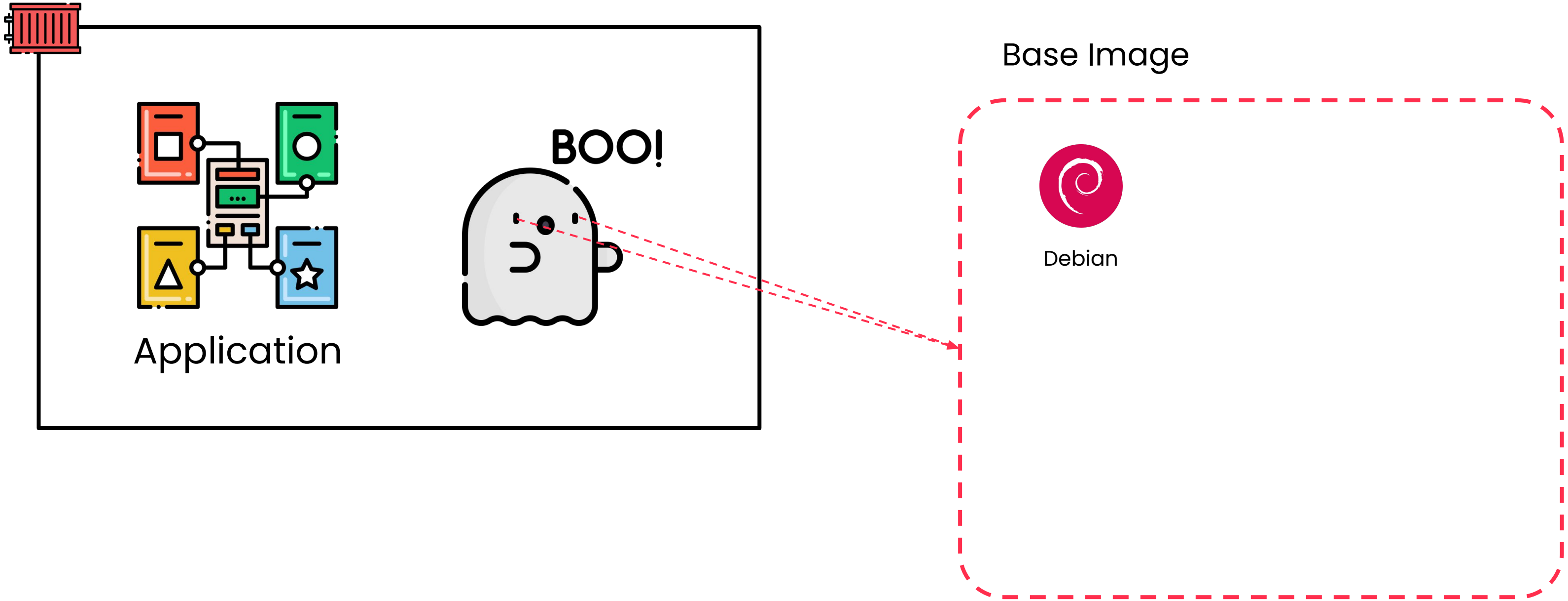


Containerization with Distroless



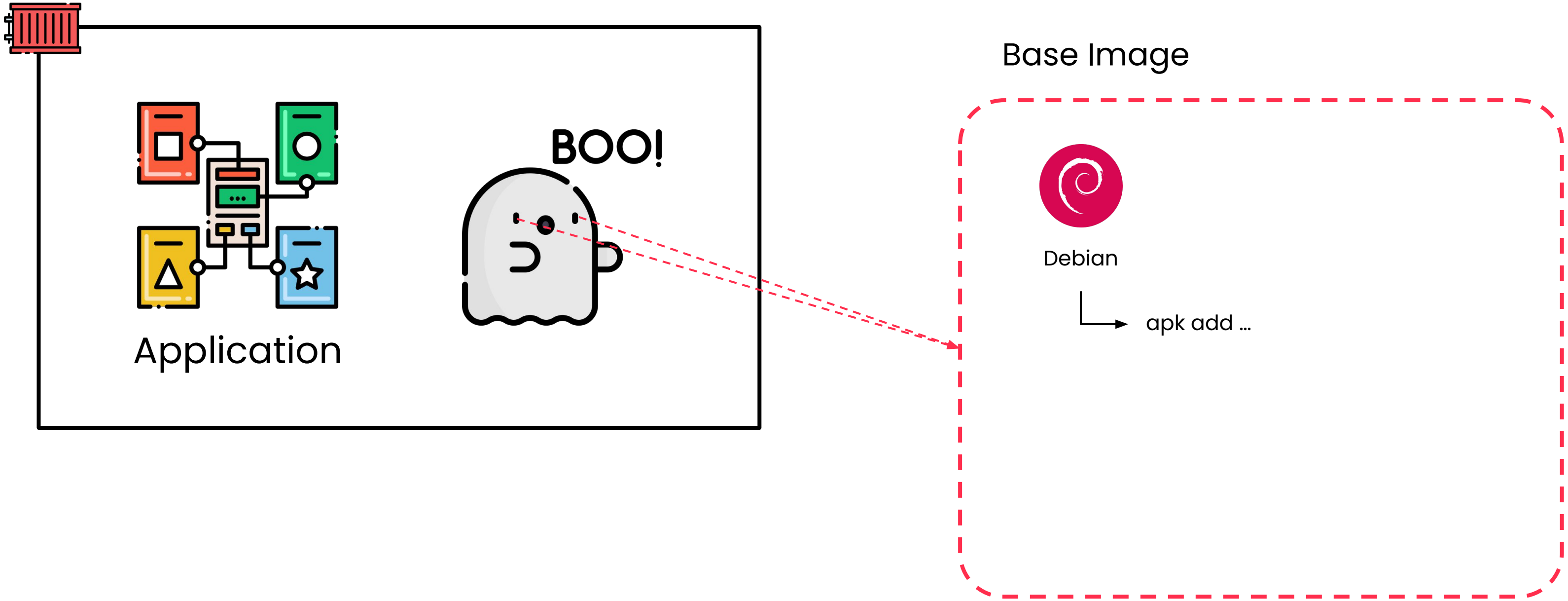
Jumpbox®

Containerization with Distroless

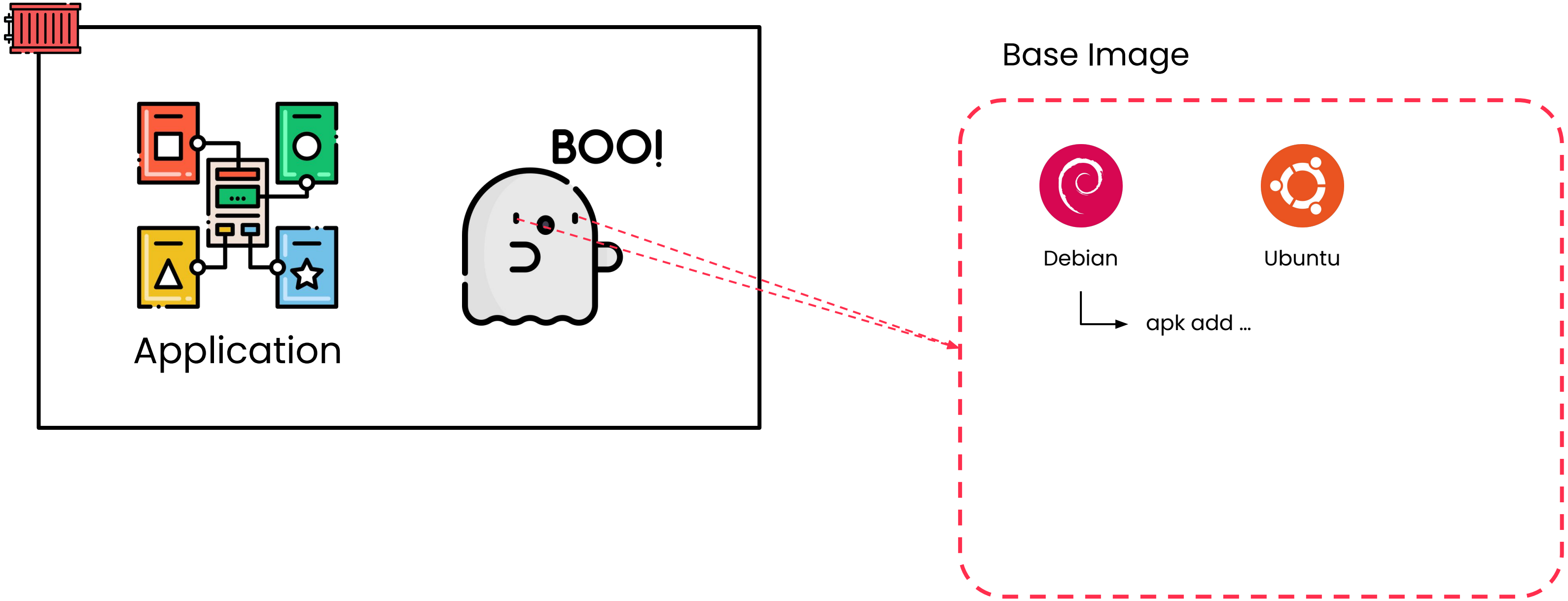


Jumpbox®

Containerization with Distroless

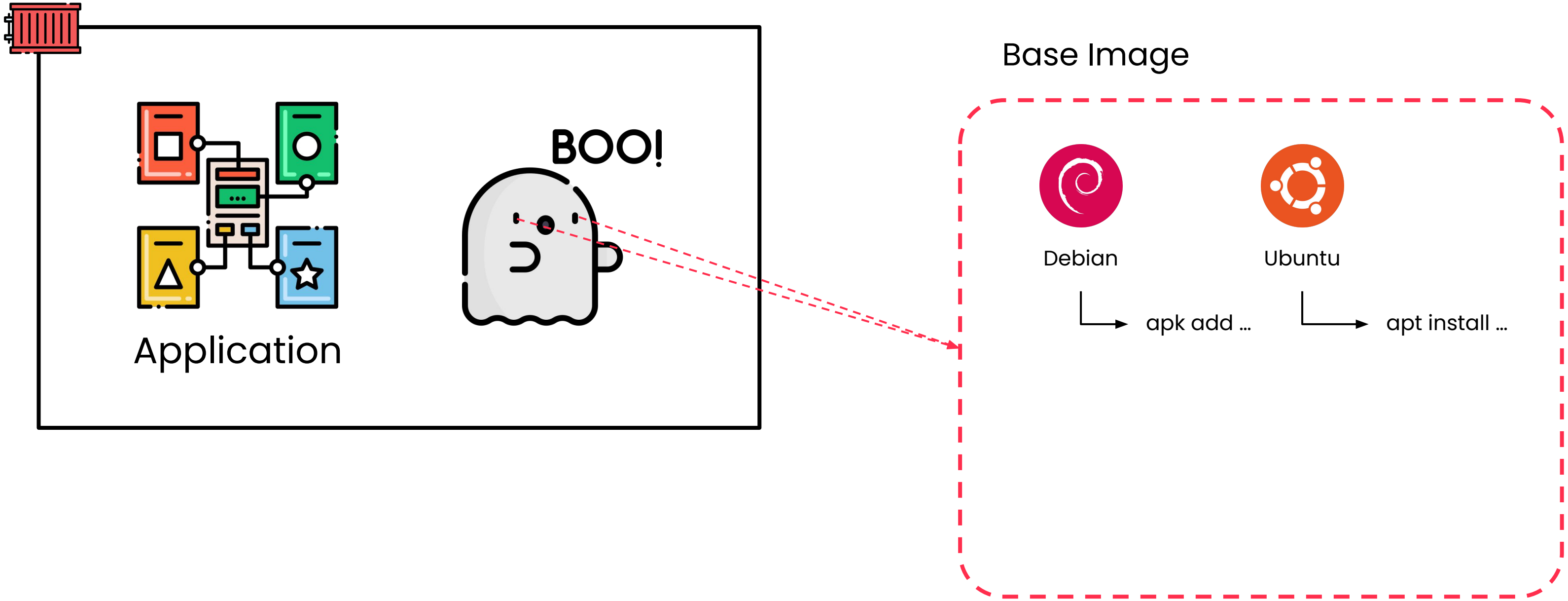


Containerization with Distroless



Jumpbox®

Containerization with Distroless



Application

Base Image

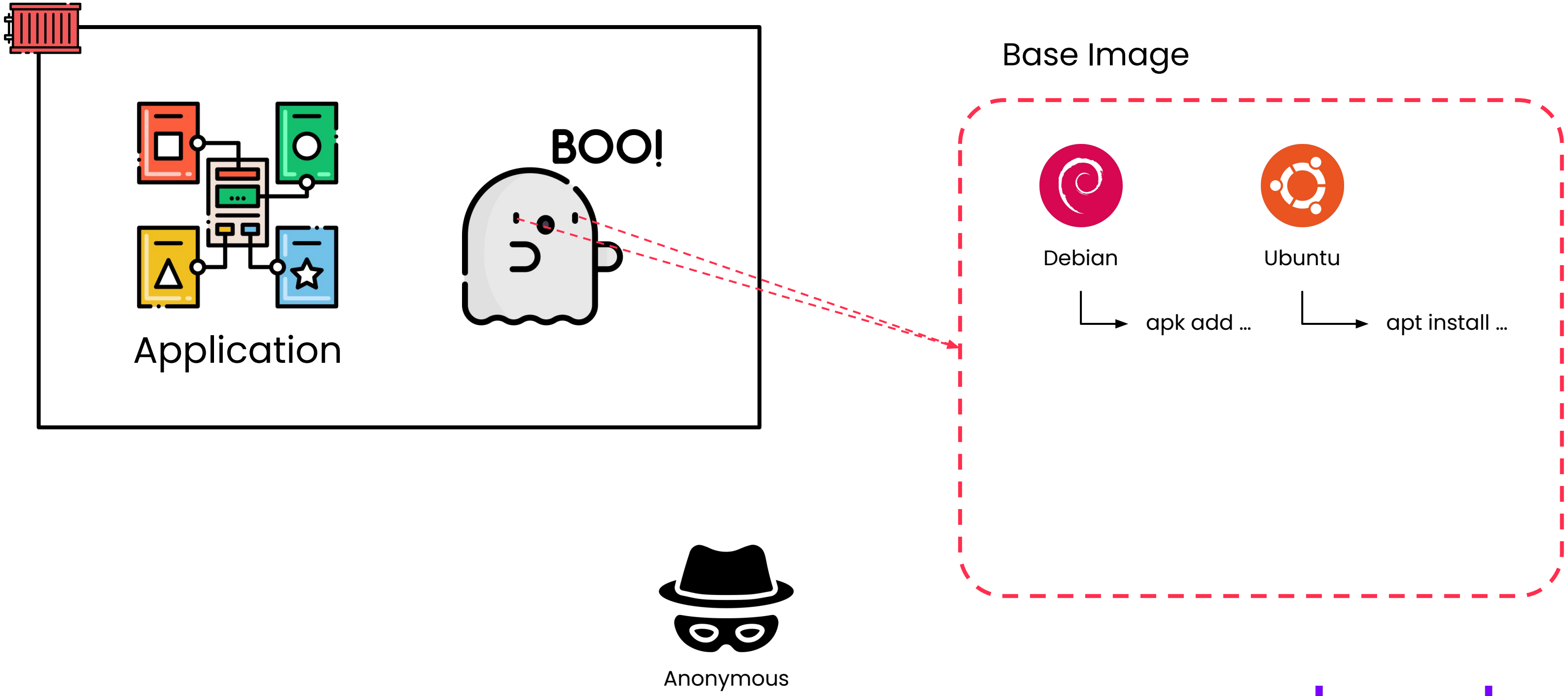
Debian

└→ apk add ...

Ubuntu

└→ apt install ...

Containerization with Distroless



Application

BOO!

Base Image

Debian

Ubuntu

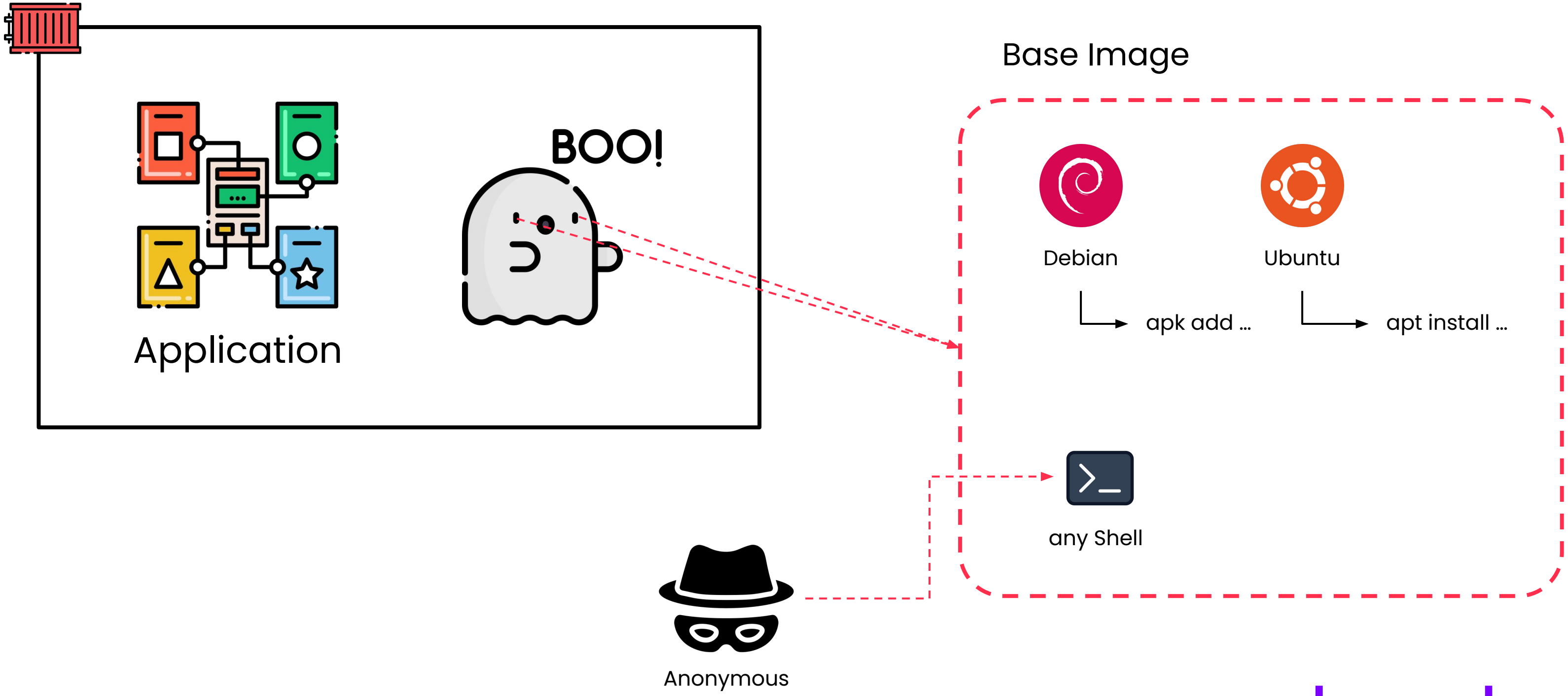
apk add ...

apt install ...

Anonymous

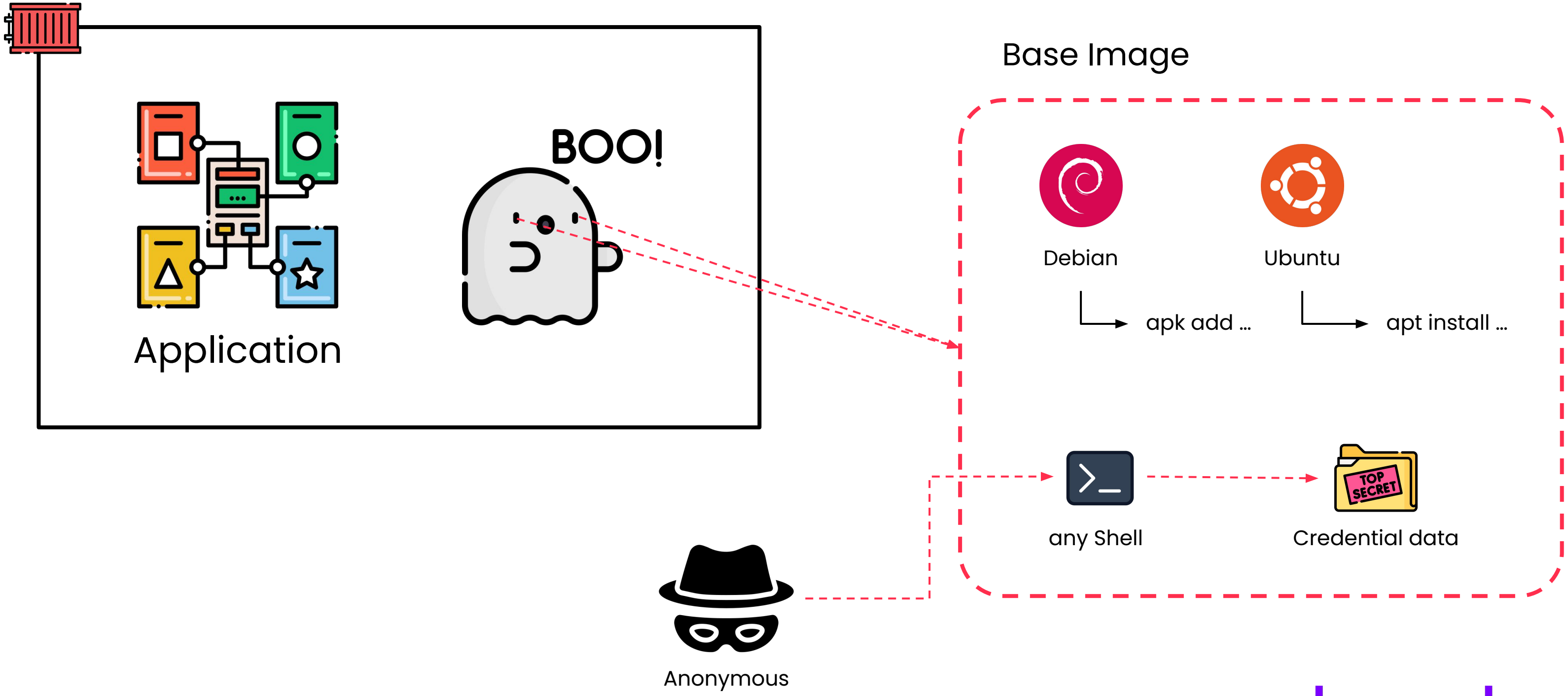
Jumpbox®

Containerization with Distroless



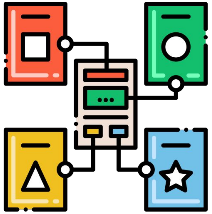
Jumpbox®

Containerization with Distroless



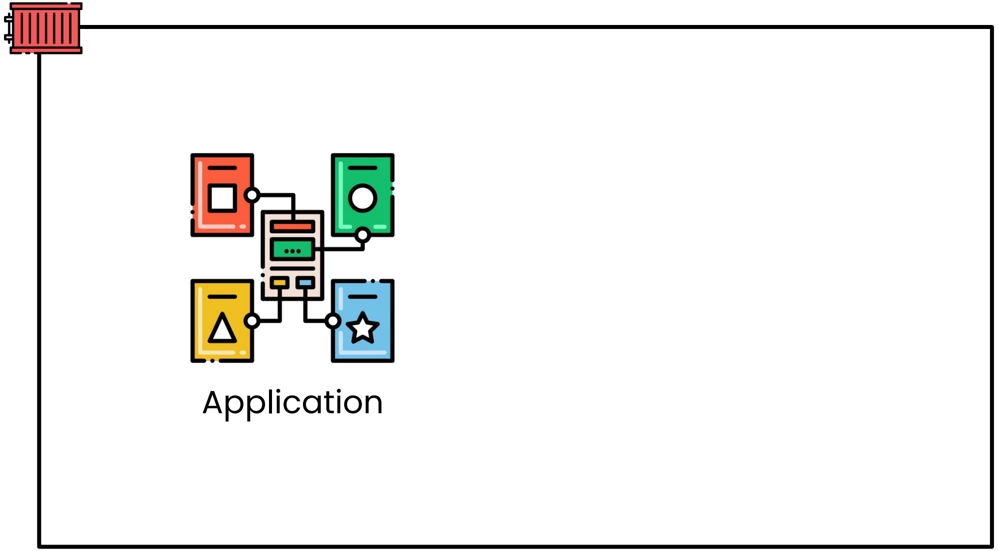
Jumpbox®

Containerization with Distroless

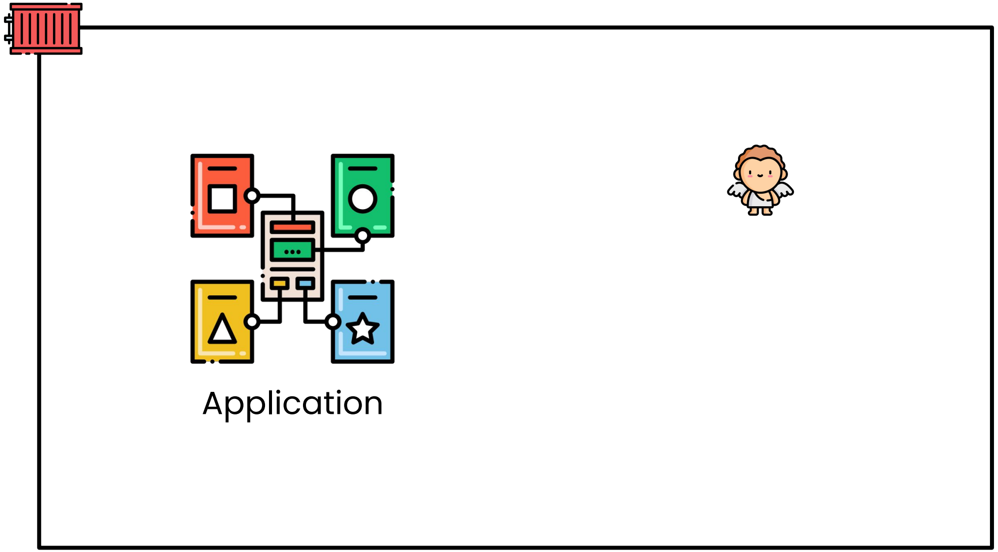


Application

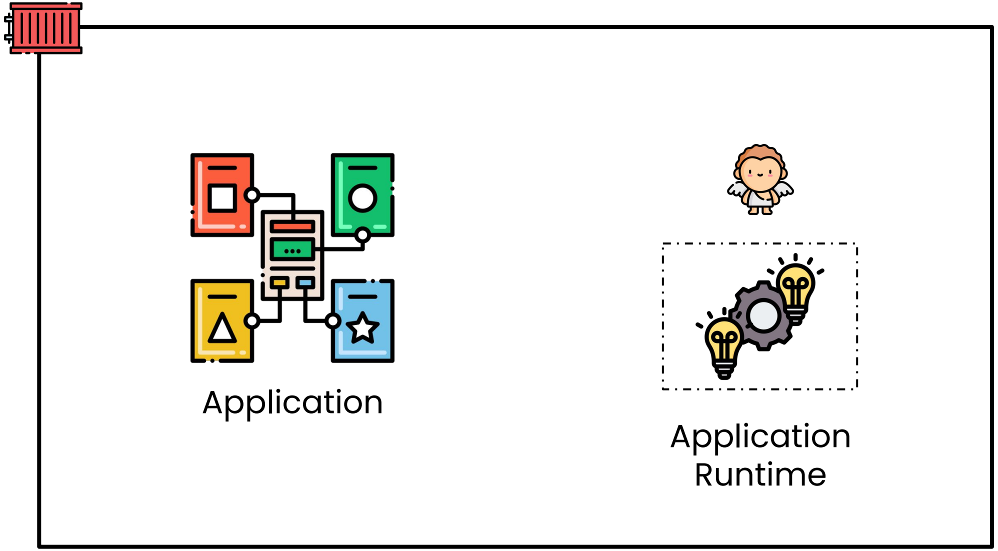
Containerization with Distroless



Containerization with Distroless

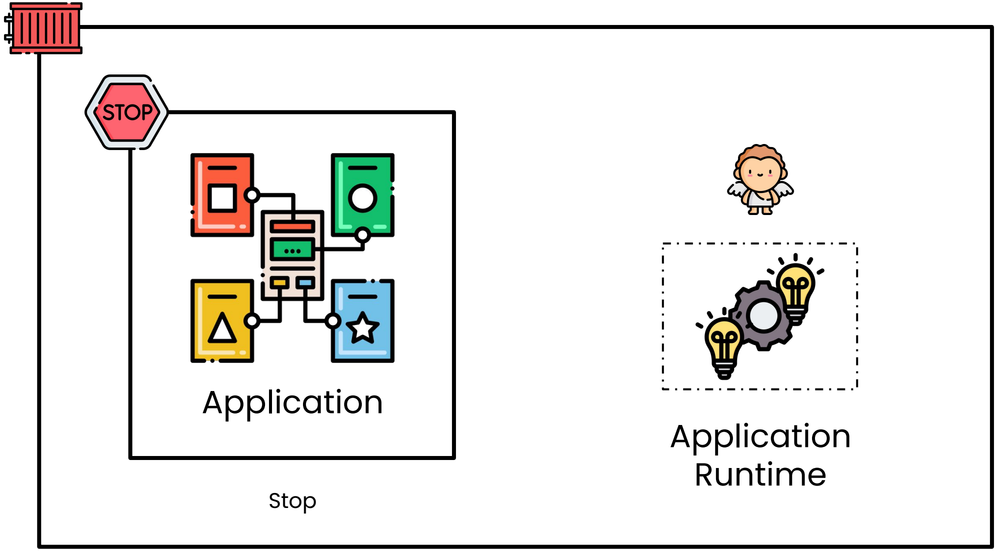


Containerization with Distroless



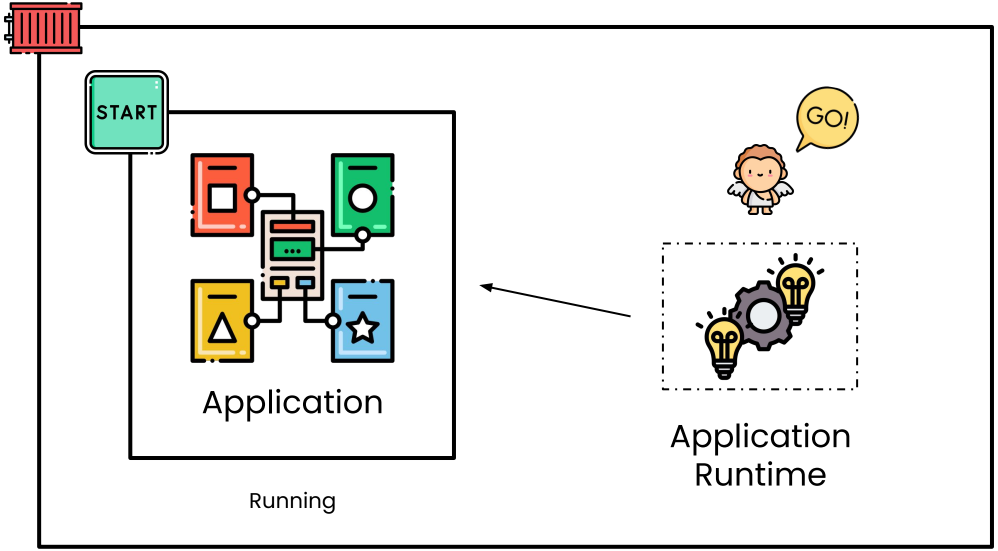
Jumpbox®

Containerization with Distroless

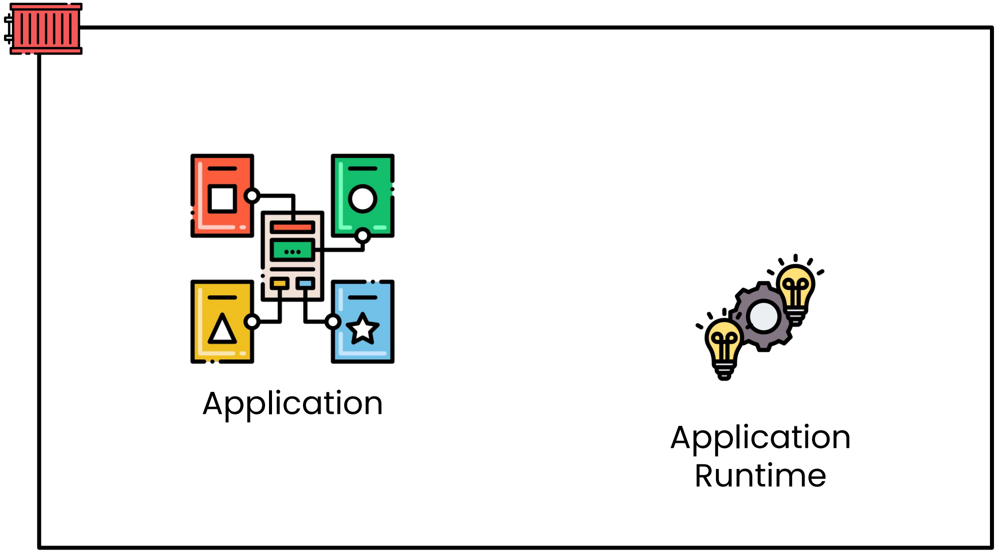


Jumpbox®

Containerization with Distroless



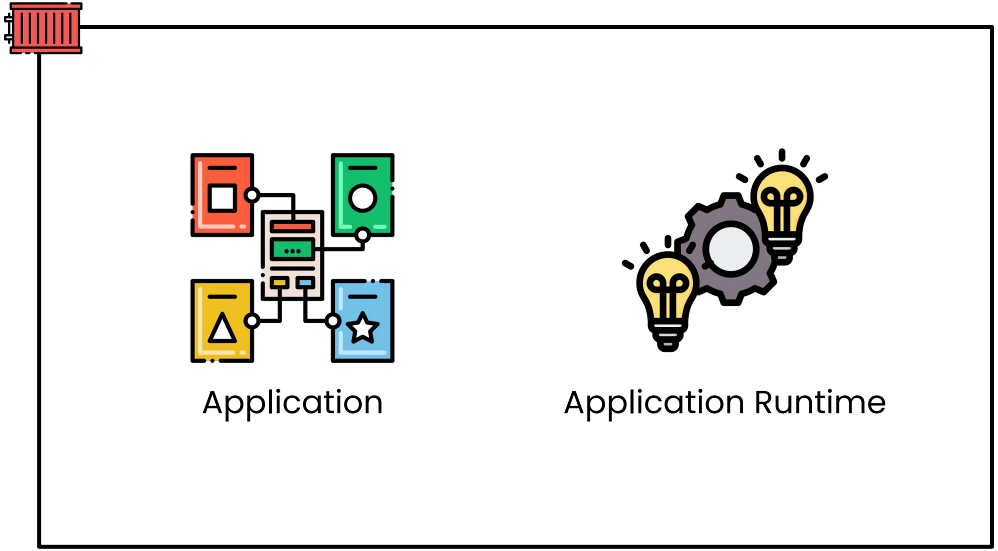
Containerization with Distroless



Application

Application Runtime

Containerization with Distroless



Application

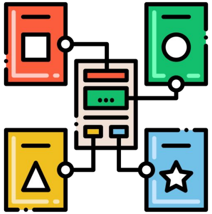
Application Runtime

Live Reload

Live Reload

When configuration is changed

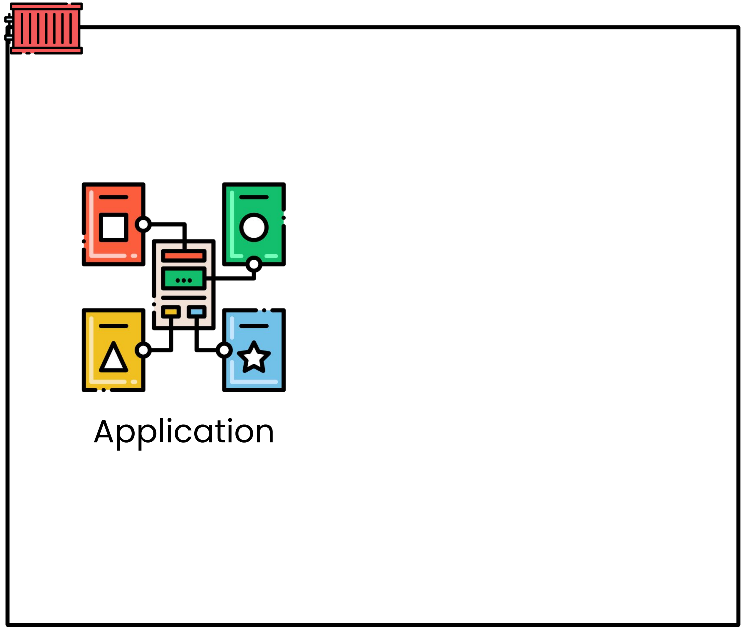
Live Reload



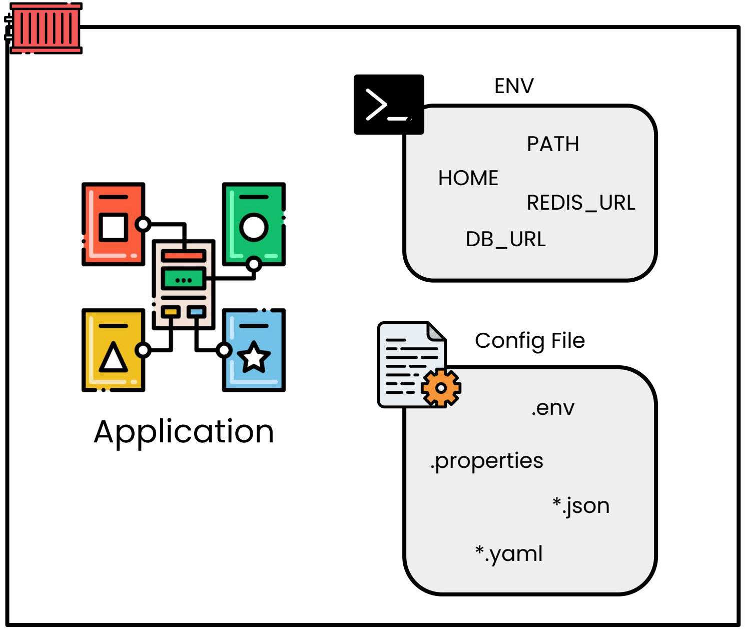
Application

Jumpbox®

Live Reload

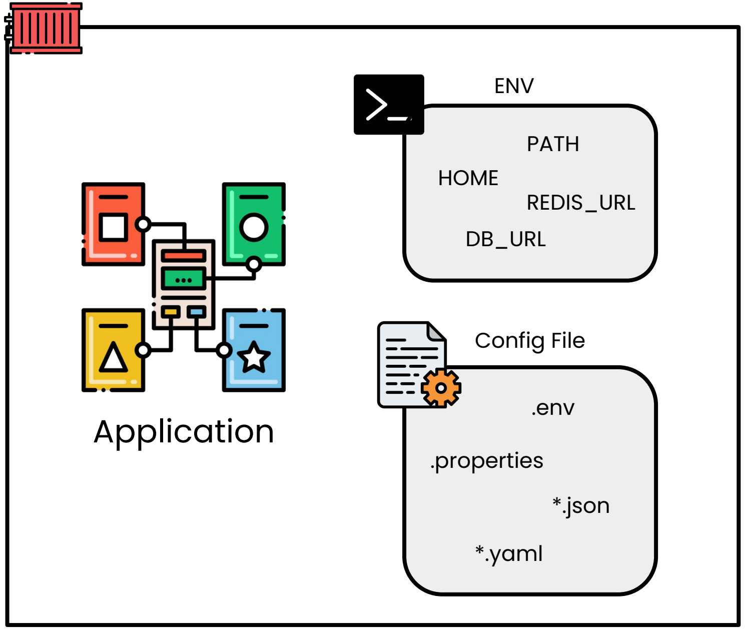


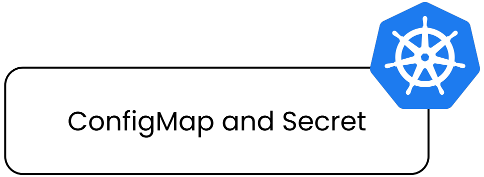
Live Reload



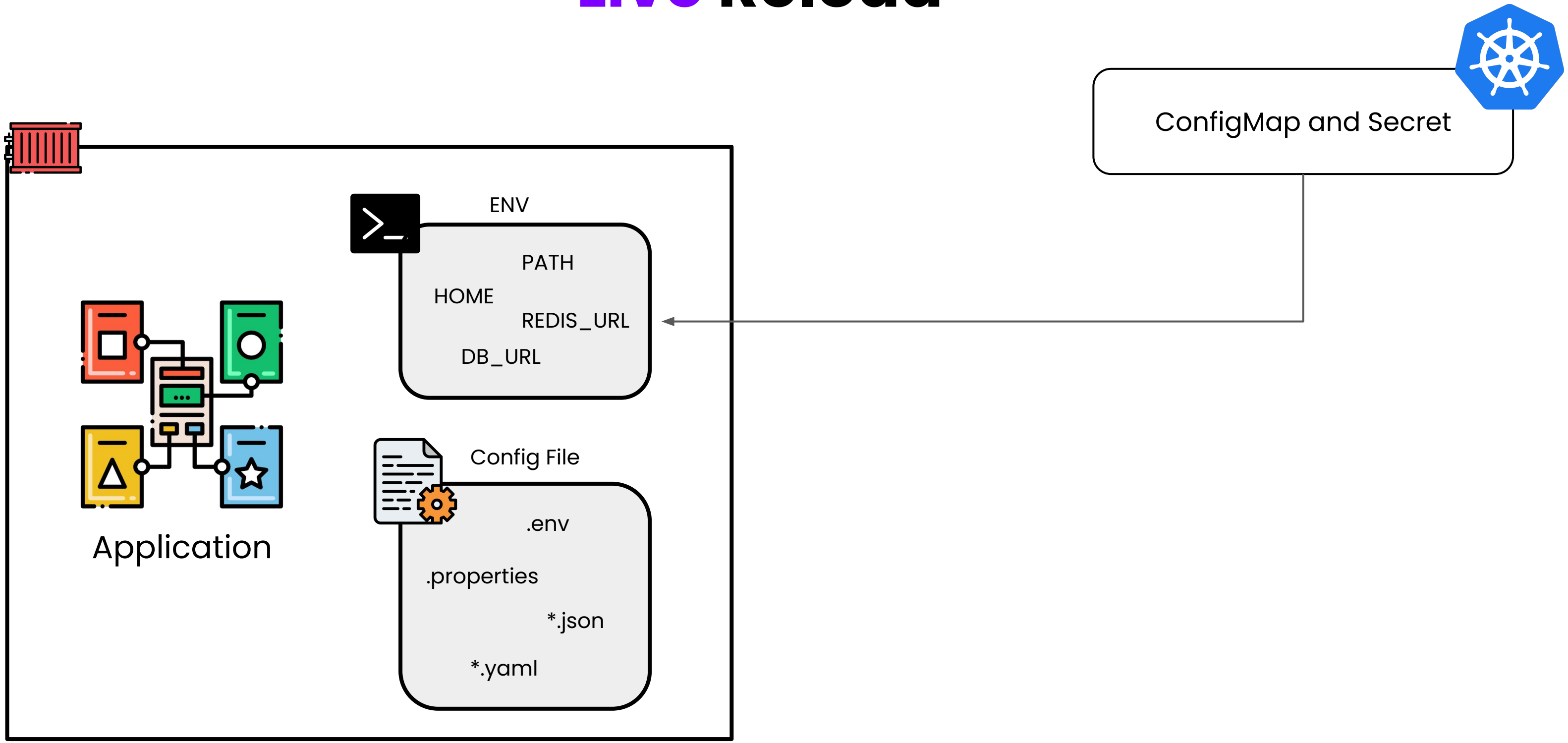
Jumpbox®

Live Reload


Application
ENV
PATH
HOME
REDIS_URL
DB_URL
Config File
.env
.properties
*.json
*.yaml

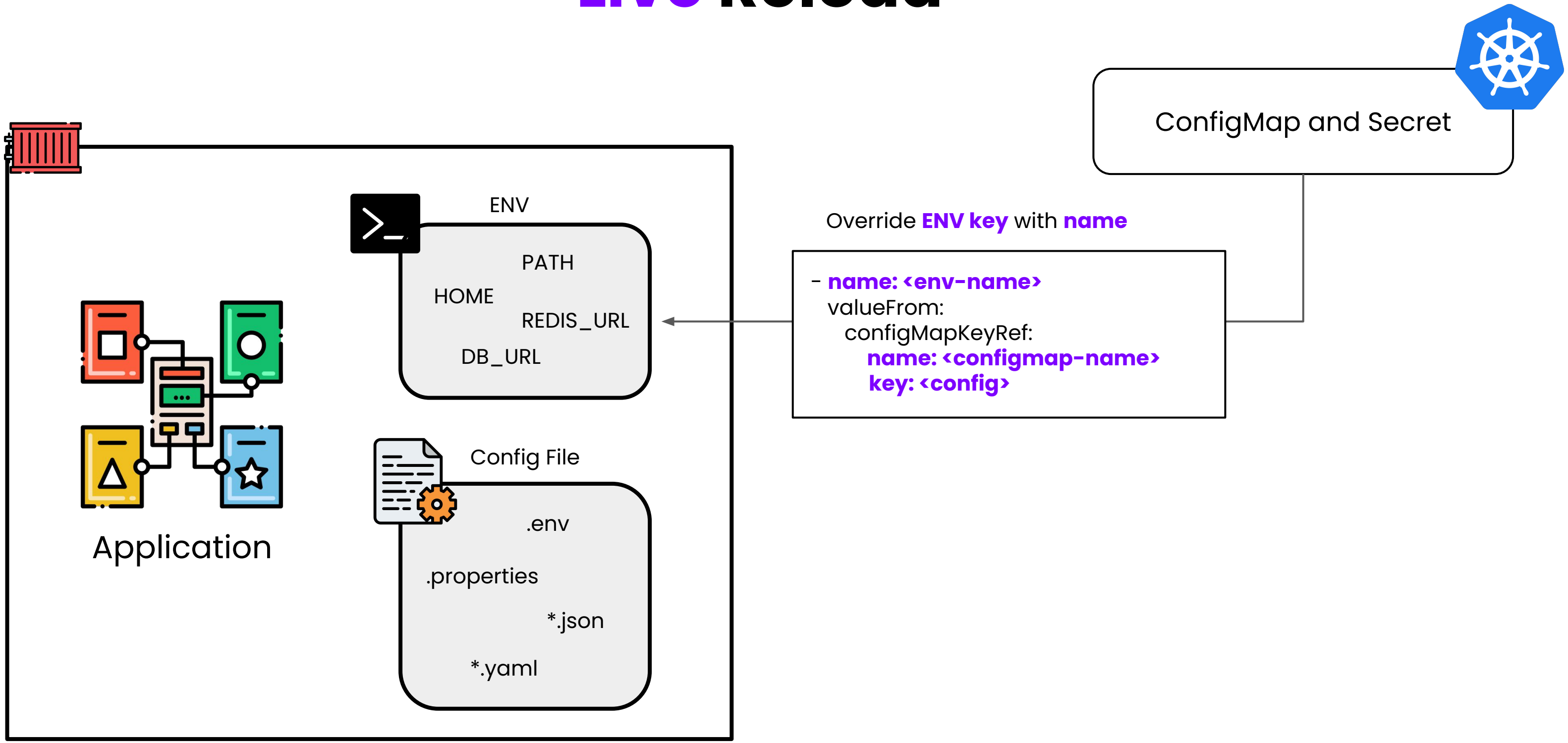

ConfigMap and Secret

Live Reload



ConfigMap and Secret
Application
ENV
PATH
HOME
REDIS_URL
DB_URL
Config File
.env
.properties
*.json
*.yaml

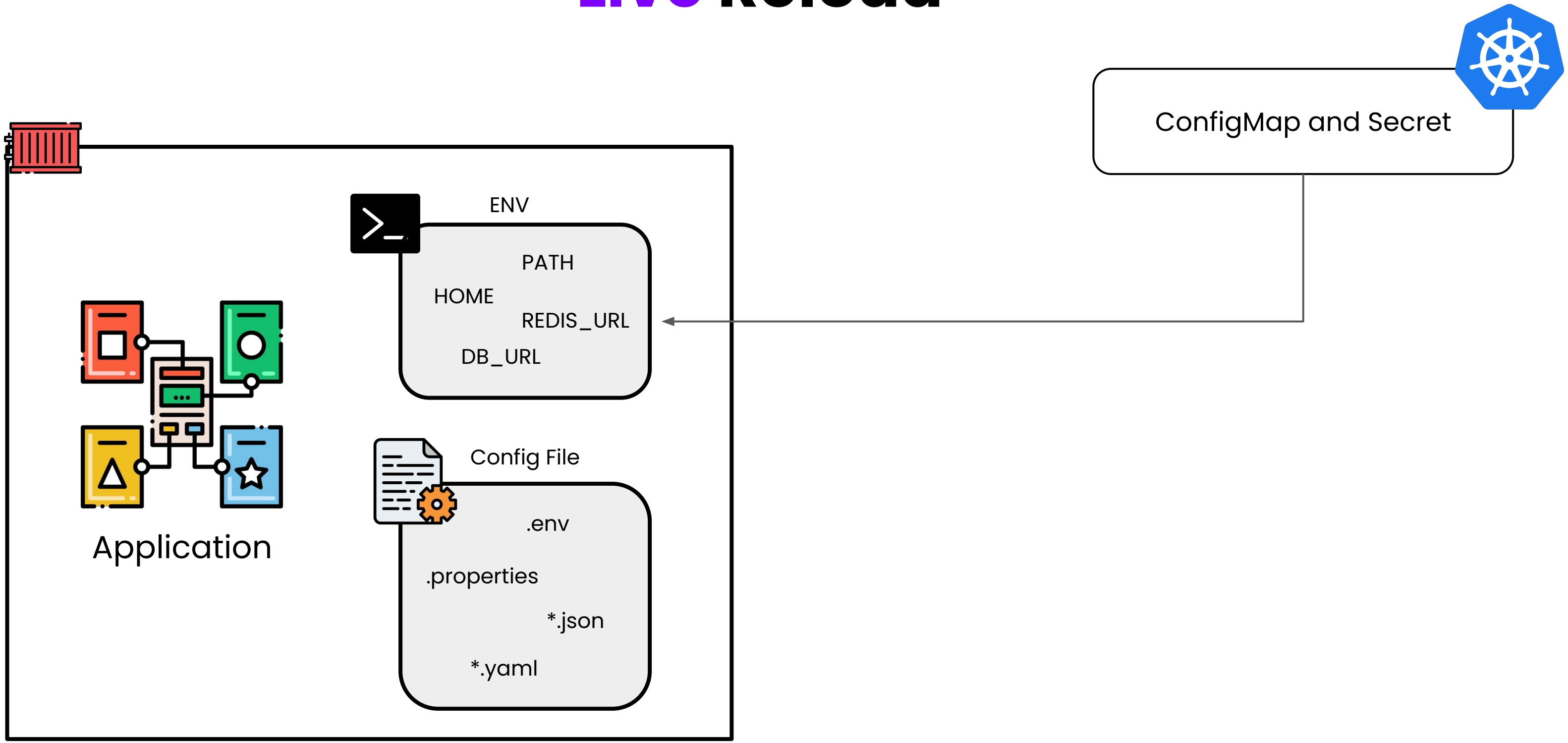
Live Reload



Override **ENV key** with **name**

```
- name: <env-name>
  valueFrom:
    configMapKeyRef:
      name: <configmap-name>
      key: <config>
```

Live Reload



ConfigMap and Secret

Application

ENV

PATH

HOME

REDIS_URL

DB_URL

Config File

.env

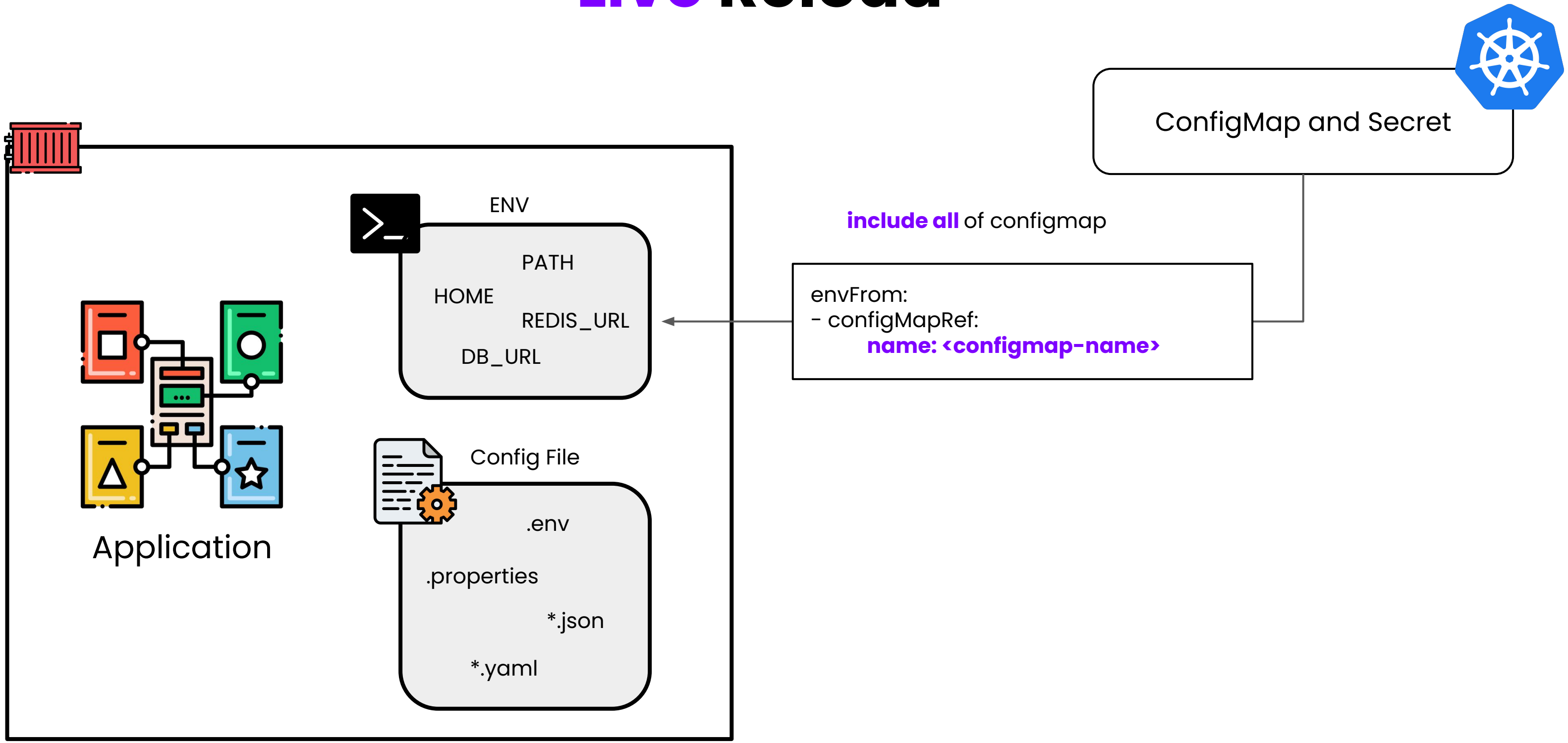
.properties

*.json

*.yaml

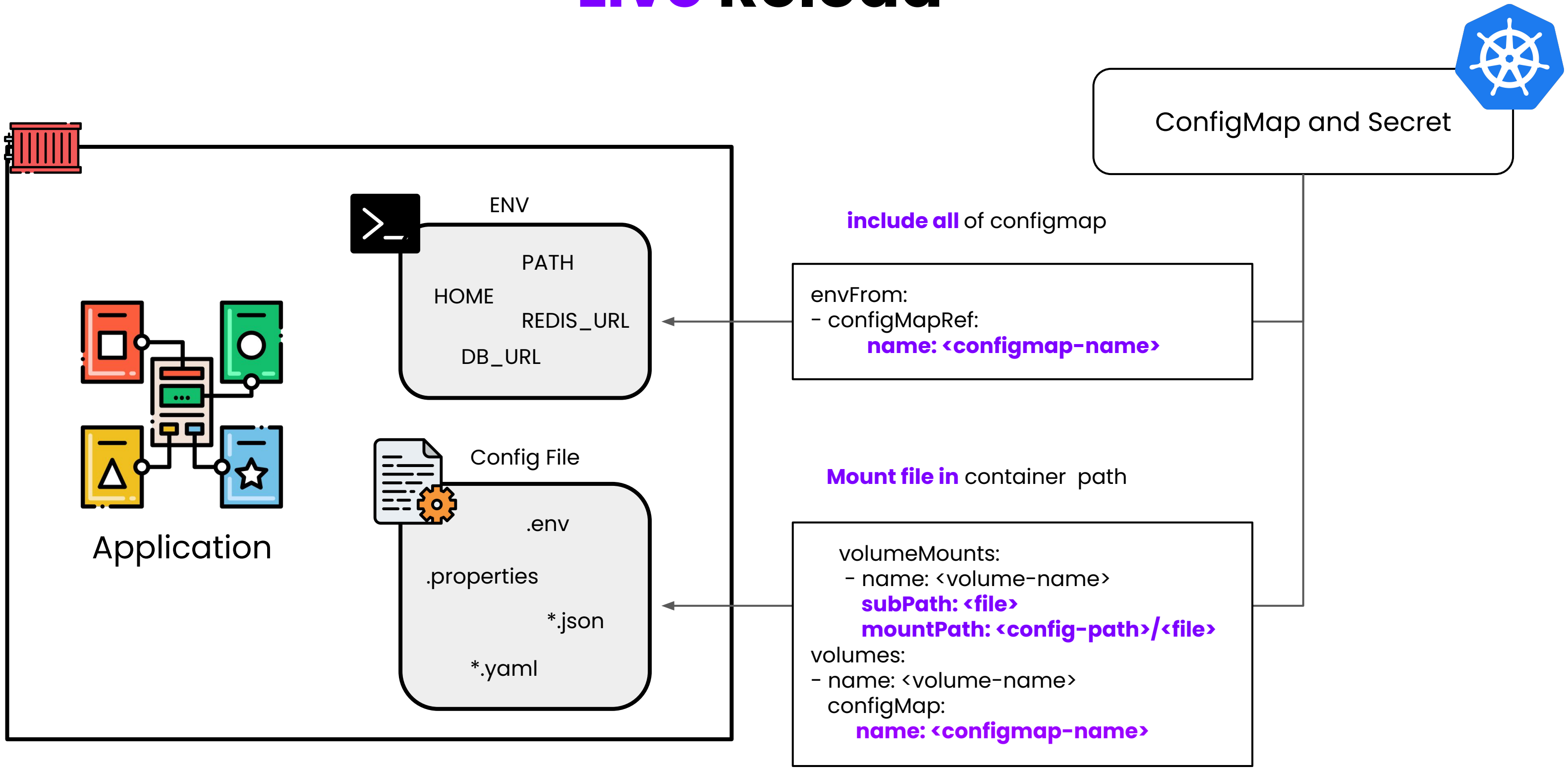
Jumpbox®

Live Reload



```
envFrom:
- configMapRef:
      name: <configmap-name>
```

Live Reload



include all of configmap

```
envFrom:
- configMapRef:
      name: <configmap-name>
```

Mount file in container path

```
  volumeMounts:
   - name: <volume-name>
     subPath: <file>
     mountPath: <config-path>/<file>
volumes:
- name: <volume-name>
  configMap:
     name: <configmap-name>
```

Summary of Live Reload

Summary of Live Reload

ConfigMap & Secret

Summary of Live Reload

ConfigMap & Secret

- Environment Variable (override **key** with **name**)

Summary of Live Reload

ConfigMap & Secret

- Environment Variable (override **key** with **name**)
- **All key-value** as Environment

Summary of **Live** Reload

ConfigMap & Secret

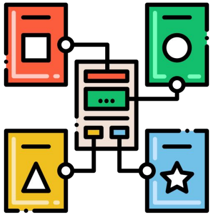
- Environment Variable (override **key** with **name**)
- **All key-value** as Environment
- All key-value **as file(s)**

Standard Logging

Team Convention & Adapter Pattern

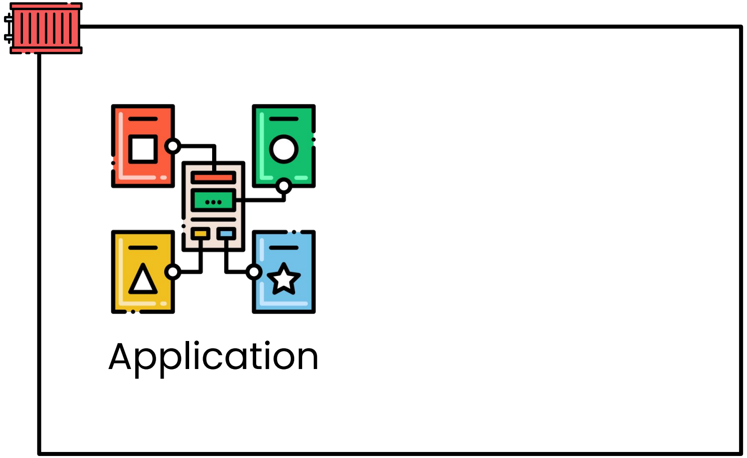
Jumpbox®

Standard Logging

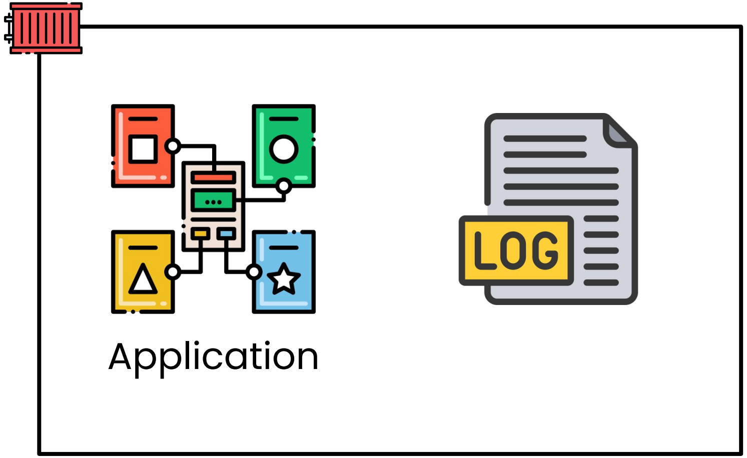


Application

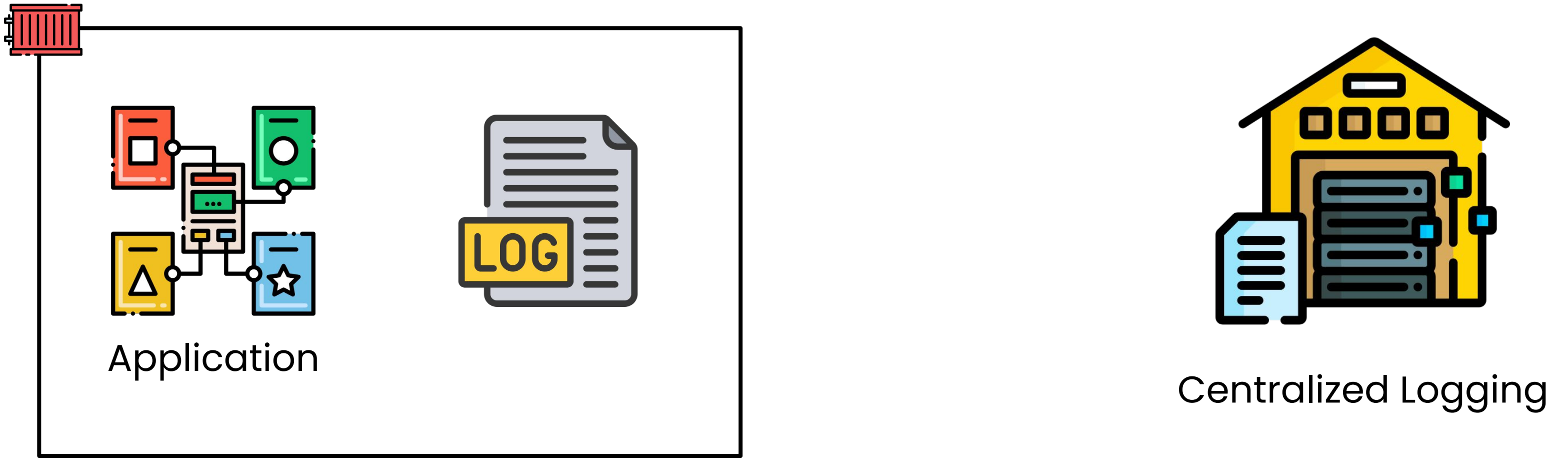
Standard Logging



Standard Logging



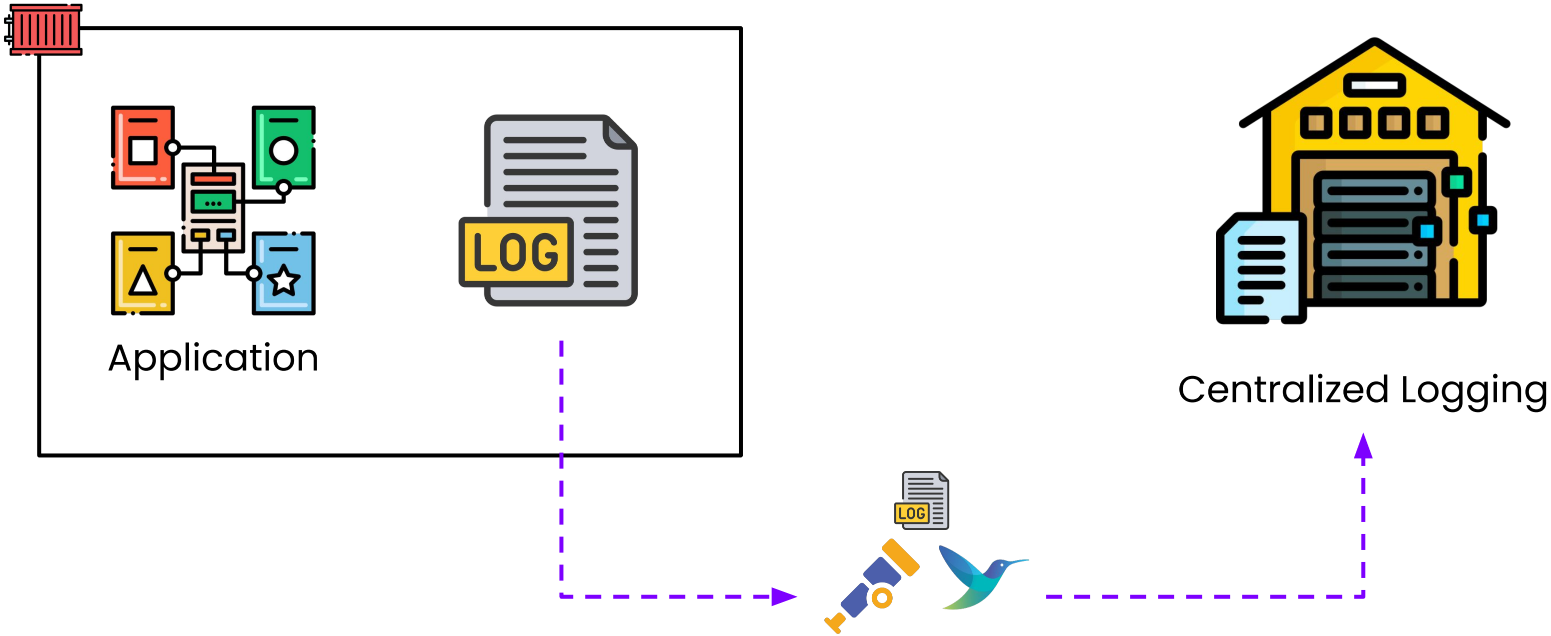
Standard Logging



Application

Centralized Logging

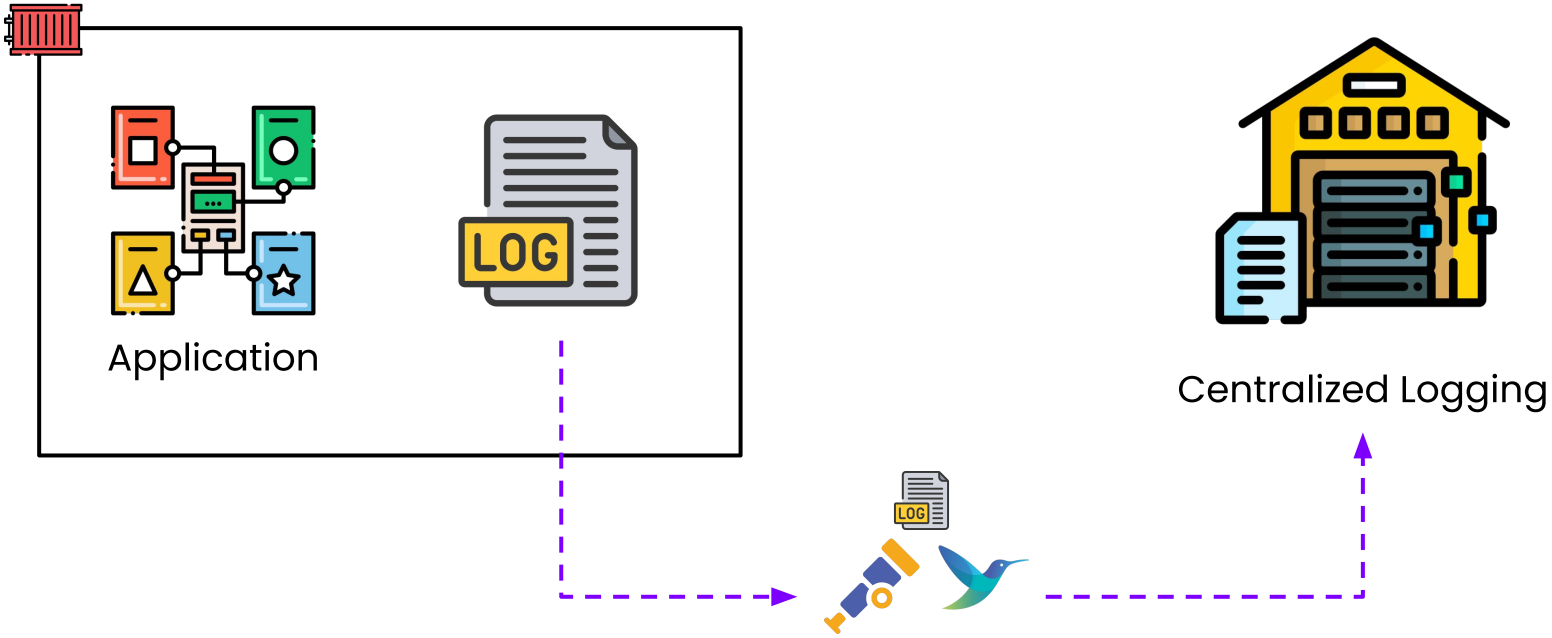
Standard Logging



Jumpbox®

Standard Logging

Stdout with **JSON**

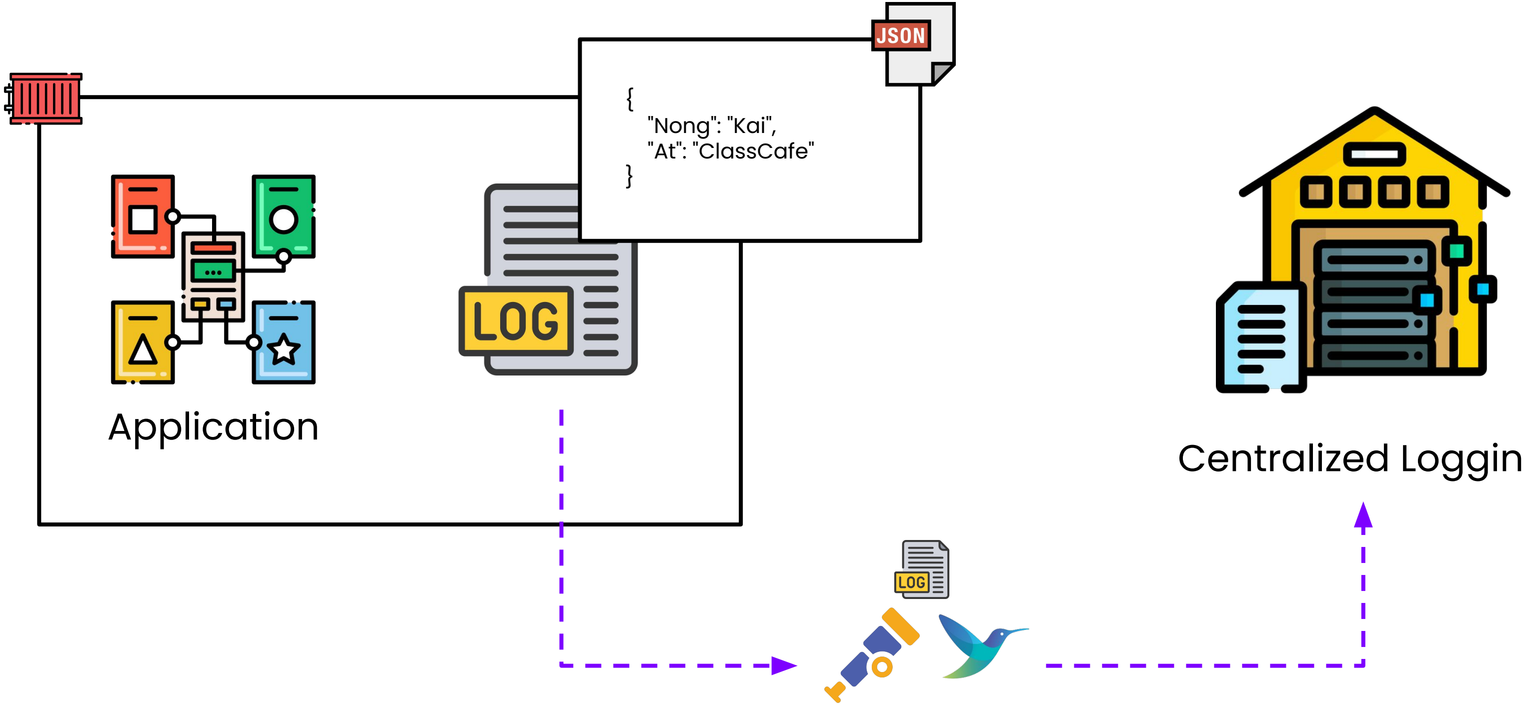


Application

Centralized Logging

Standard Logging

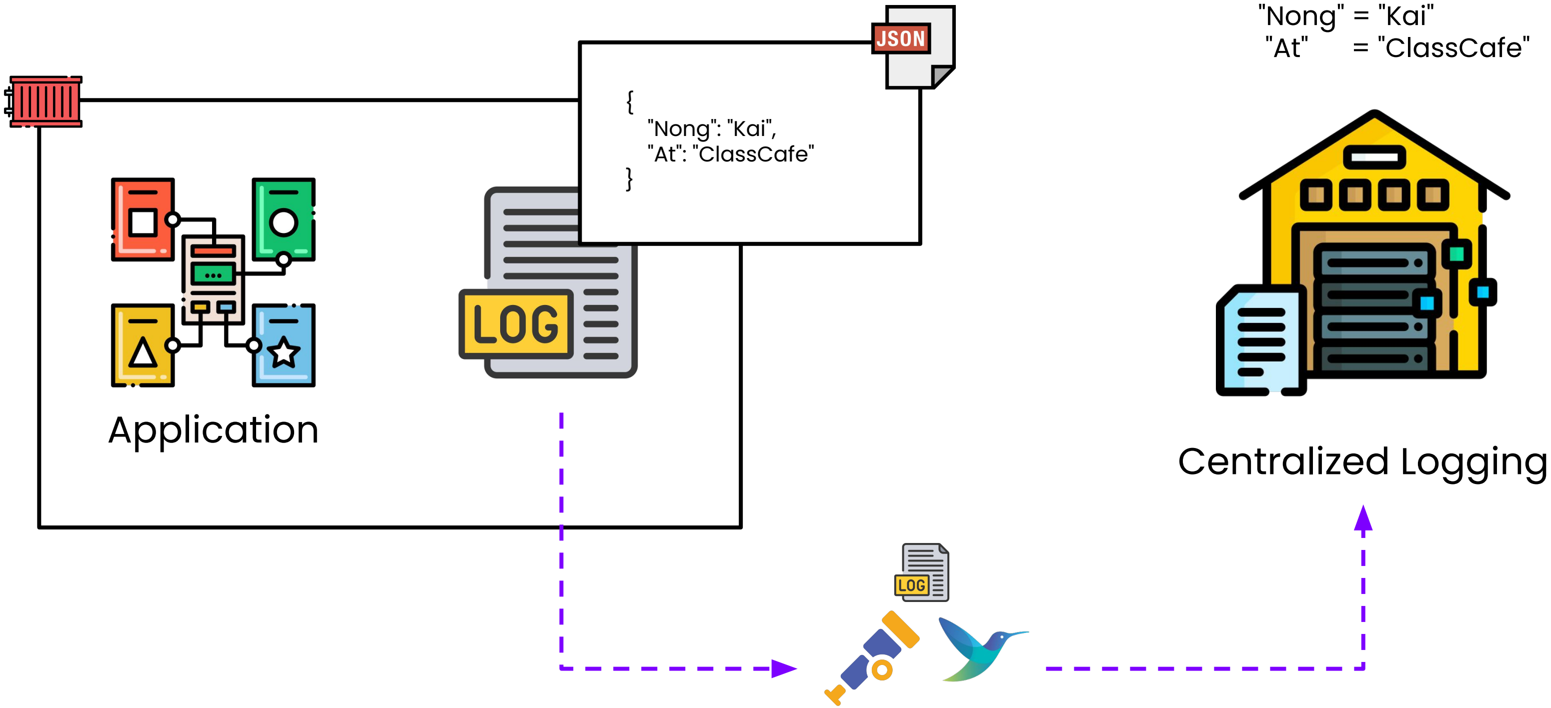
Stdout with JSON



```
{
    "Nong": "Kai",
    "At": "ClassCafe"
}
```

Standard Logging

Stdout with **JSON**



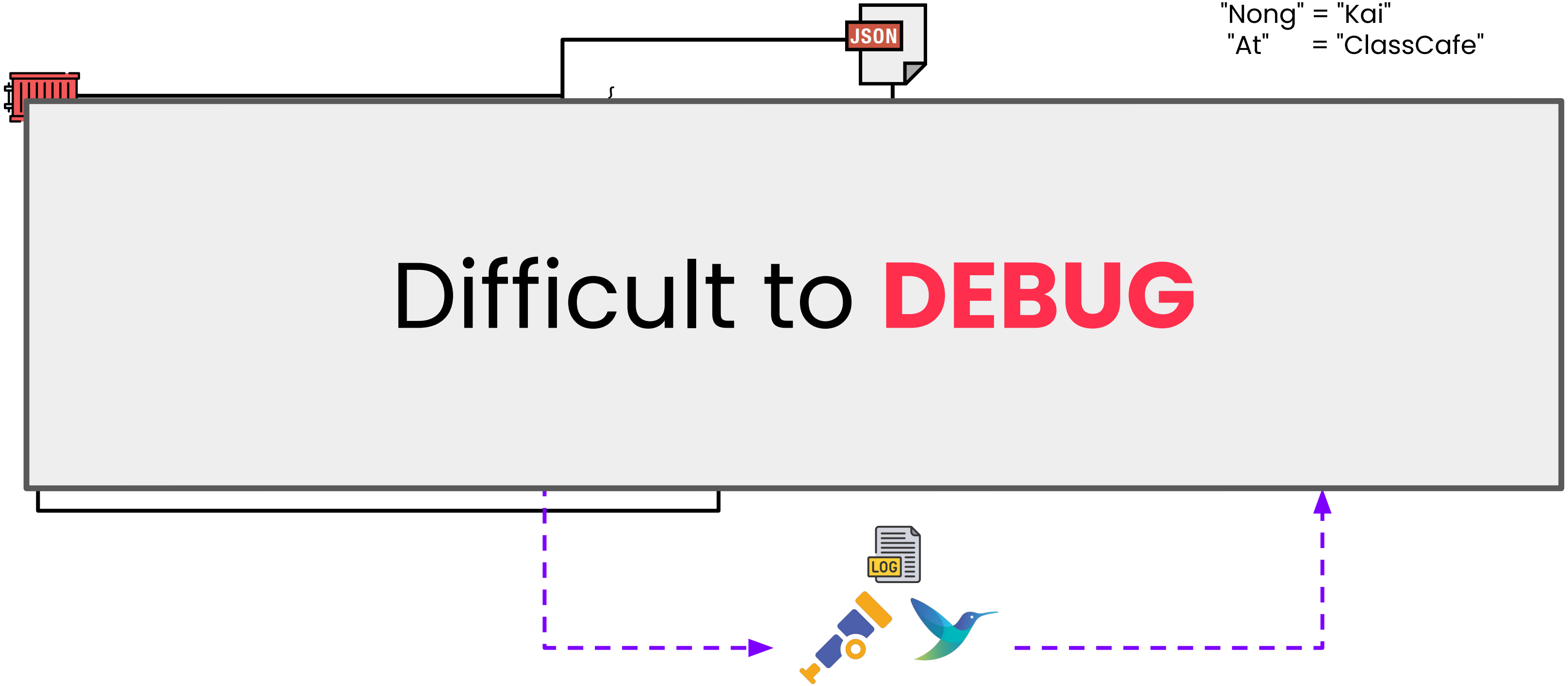
```
{
  "Nong": "Kai",
  "At": "ClassCafe"
}
```

```
"Nong" = "Kai"
 "At"     = "ClassCafe"
```

Centralized Logging

Standard **Logging**

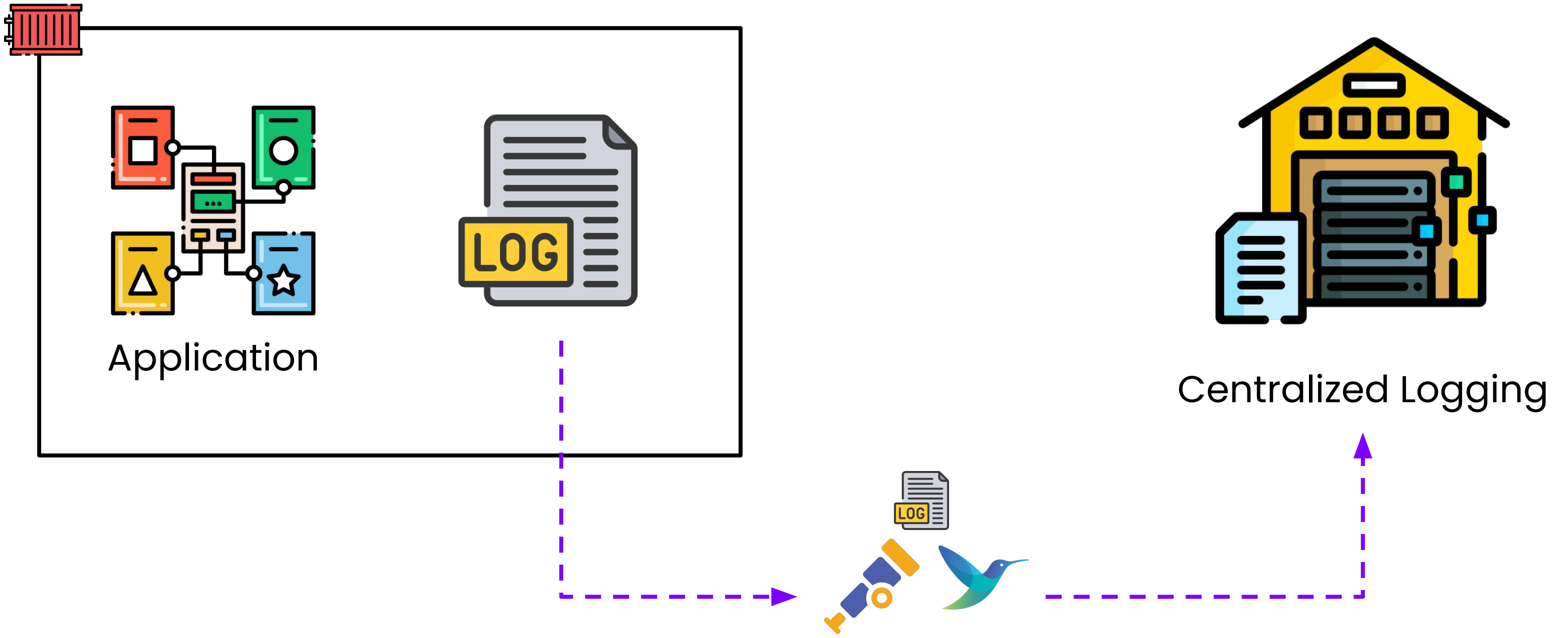
Stdout with **JSON**



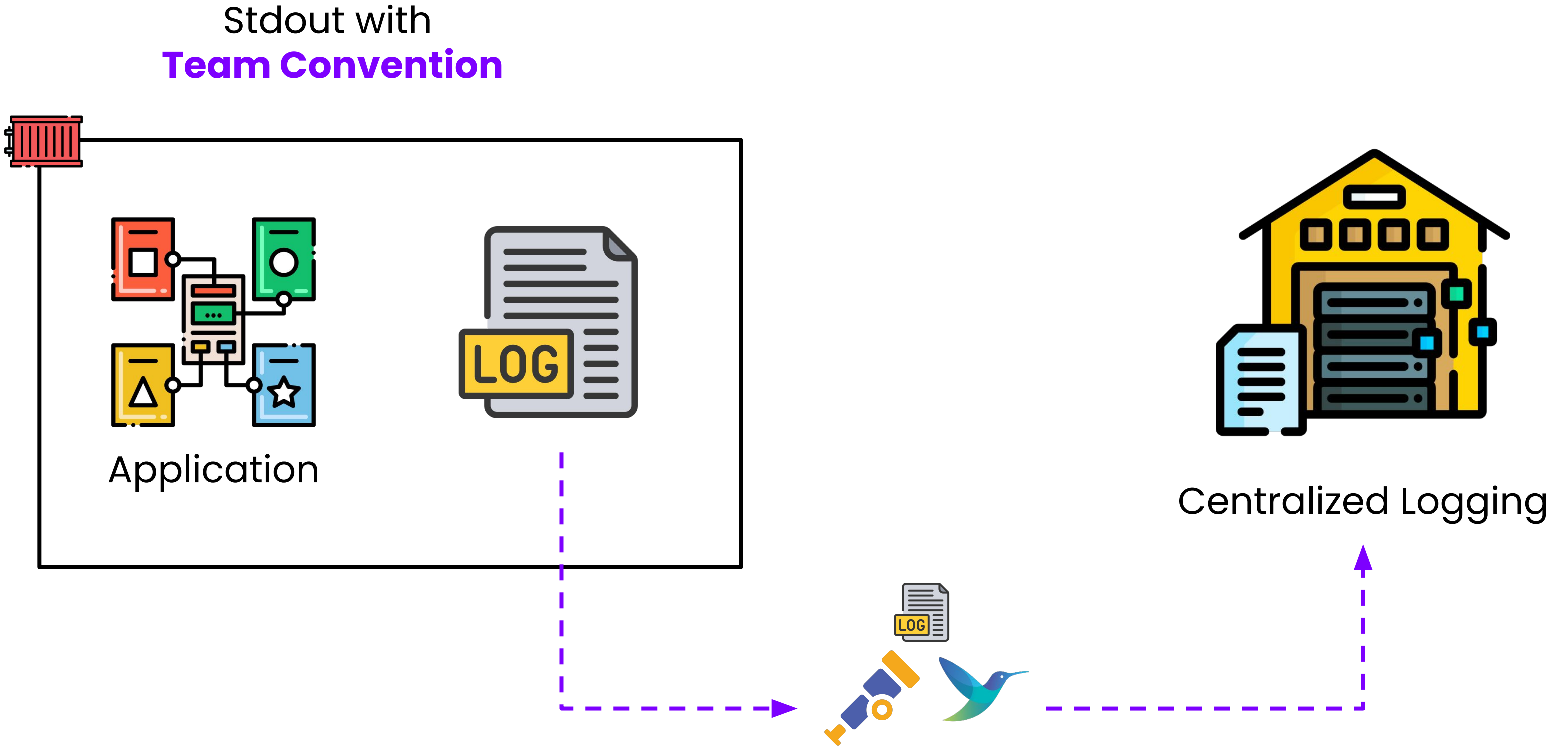
"Nong" = "Kai"
"At" = "ClassCafe"

Difficult to **DEBUG**

Standard Logging



Standard Logging



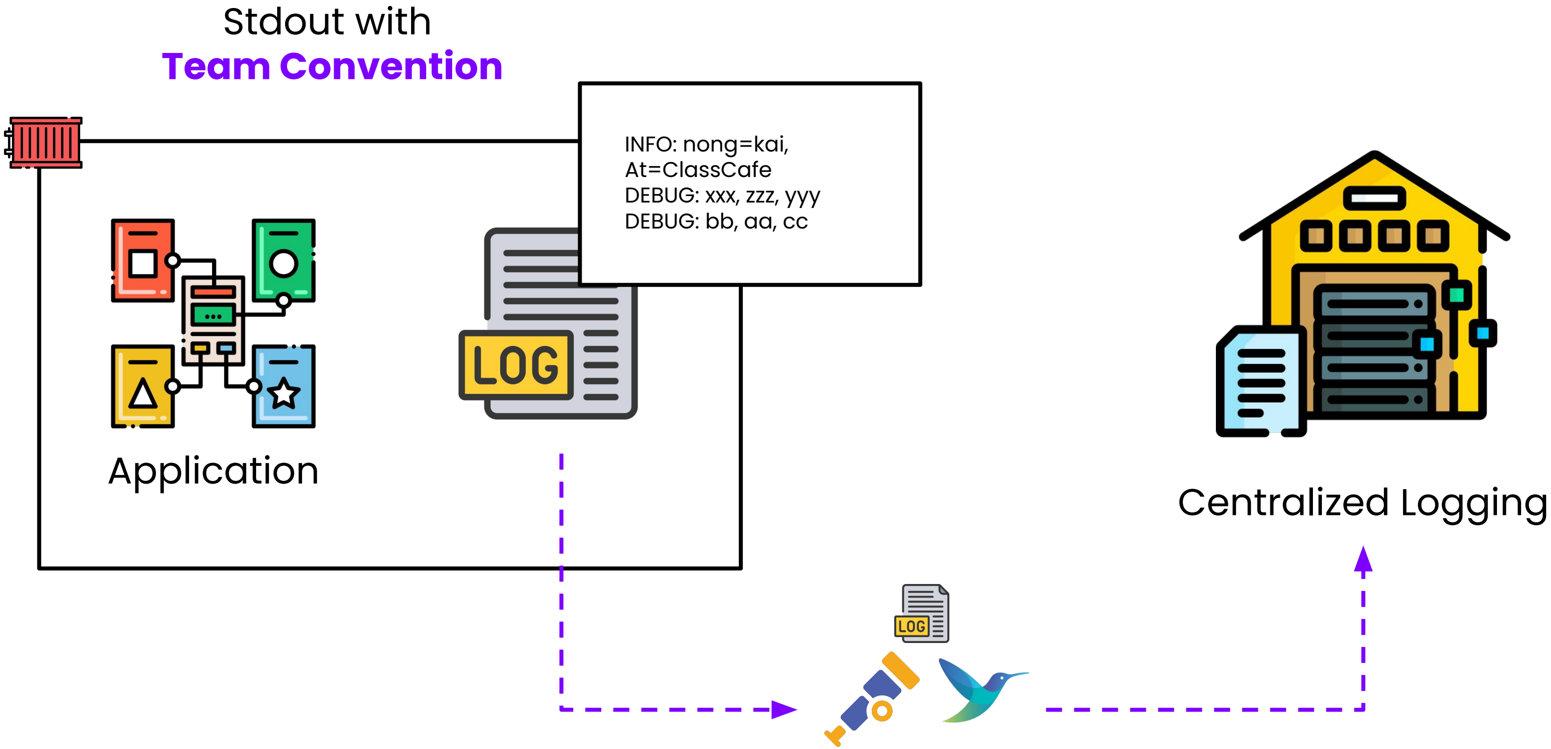
Stdout with
Team Convention

Application

Centralized Logging

Jumpbox®

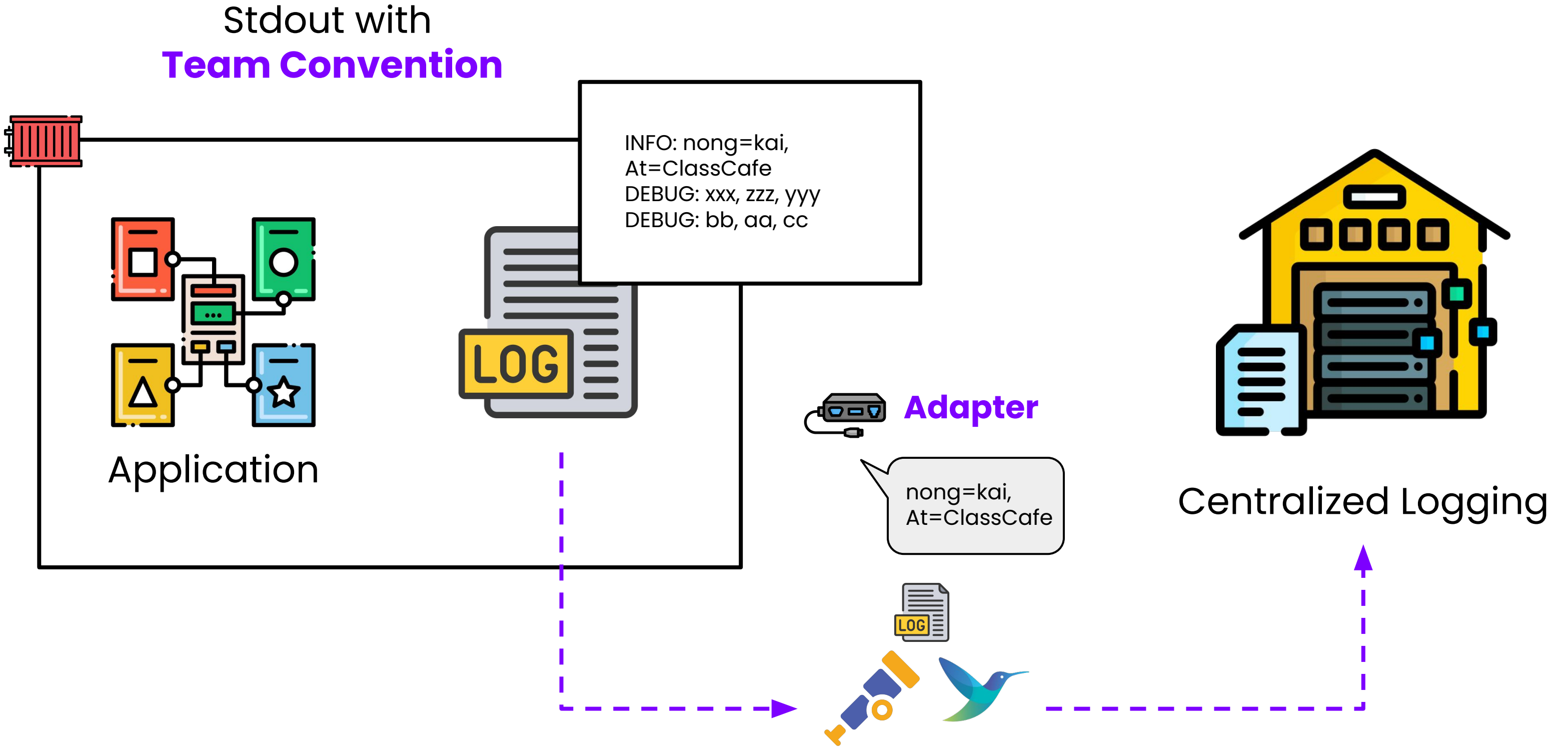
Standard Logging



Stdout with
Team Convention

```
INFO: nong=kai,
At=ClassCafe
DEBUG: xxx, zzz, yyy
DEBUG: bb, aa, cc
```

Standard Logging



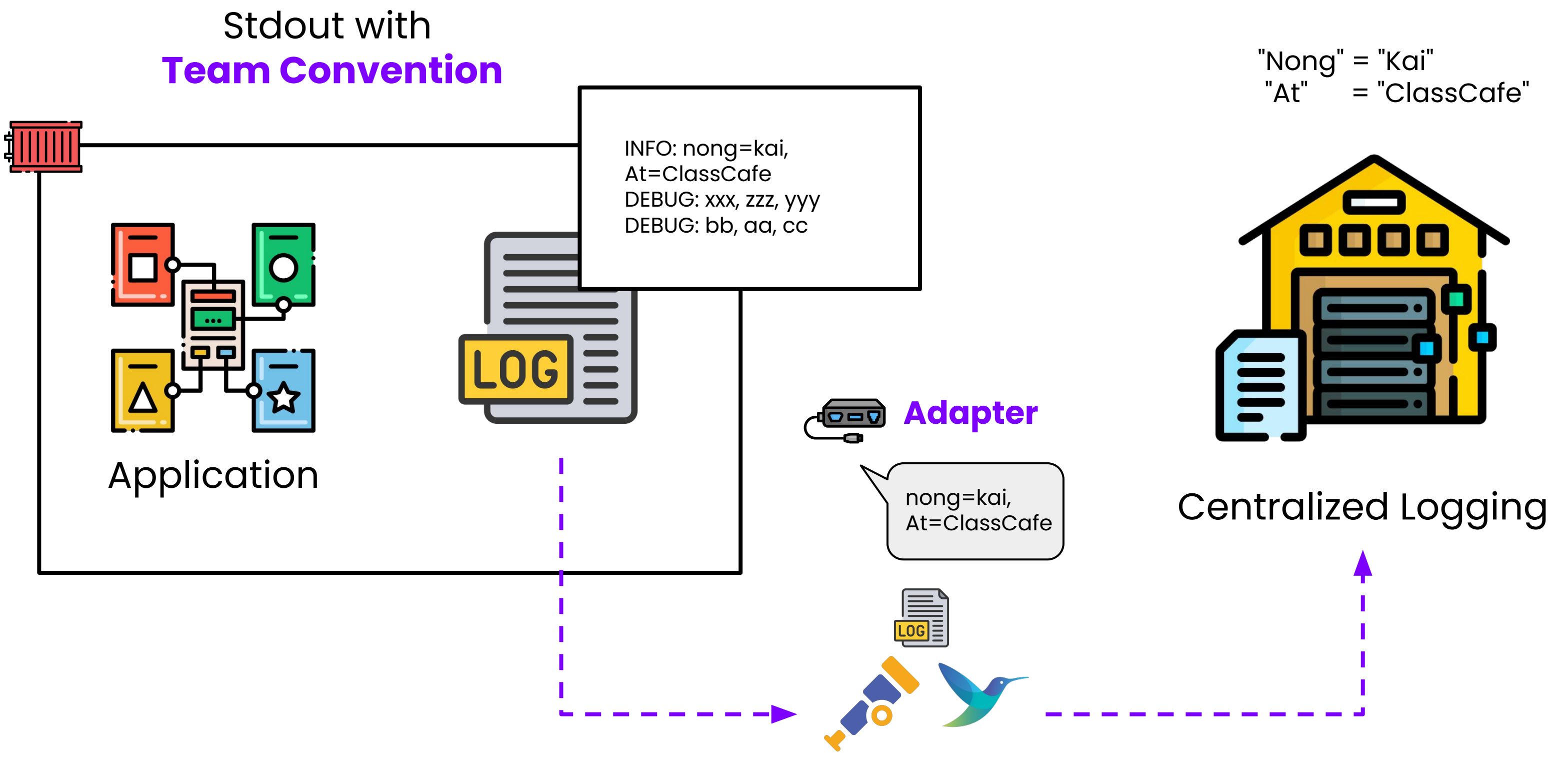
Stdout with
Team Convention

```
INFO: nong=kai,
At=ClassCafe
DEBUG: xxx, zzz, yyy
DEBUG: bb, aa, cc
```

```
nong=kai,
At=ClassCafe
```

Jumpbox®

Standard **Logging**



```
INFO: nong=kai,
At=ClassCafe
DEBUG: xxx, zzz, yyy
DEBUG: bb, aa, cc
```

```
nong=kai,
At=ClassCafe
```

Database Migration

When your service is running on Kubernetes

Jumpbox®

Database Migration

1. Service Startup

Database Migration

1. Service Startup
2. initContainers

Database Migration

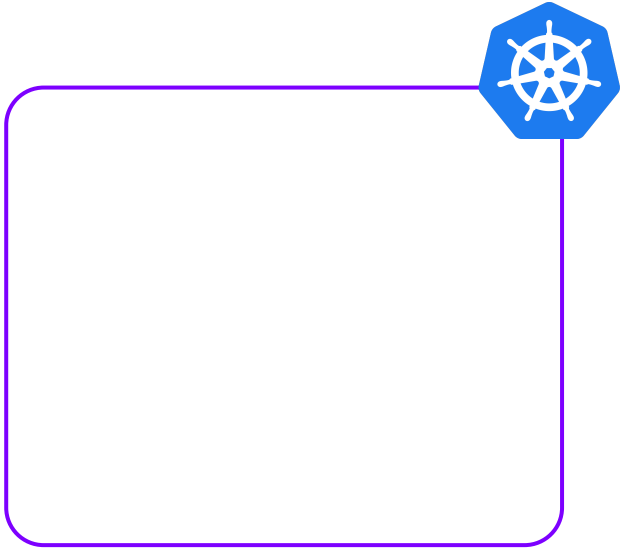
1. Service Startup
2. initContainers
3. Kubernetes Job

Database Migration

1. Service Startup
2. initContainers
3. Kubernetes Job
4. CD Pipeline Before Deployment Is Triggered

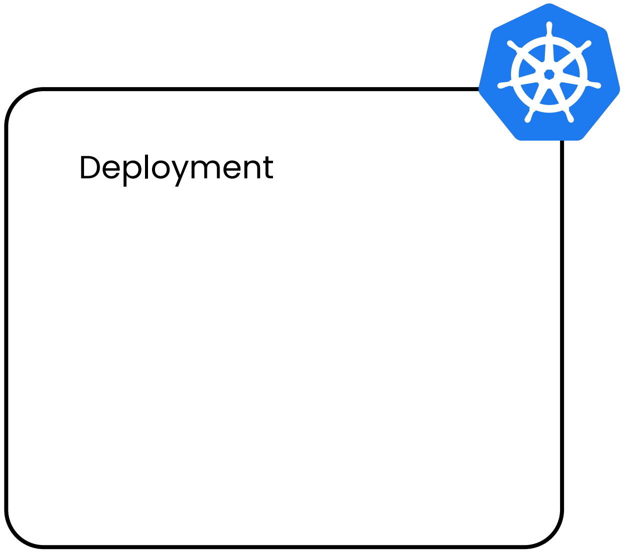
2. Service Startup

1. Service Startup



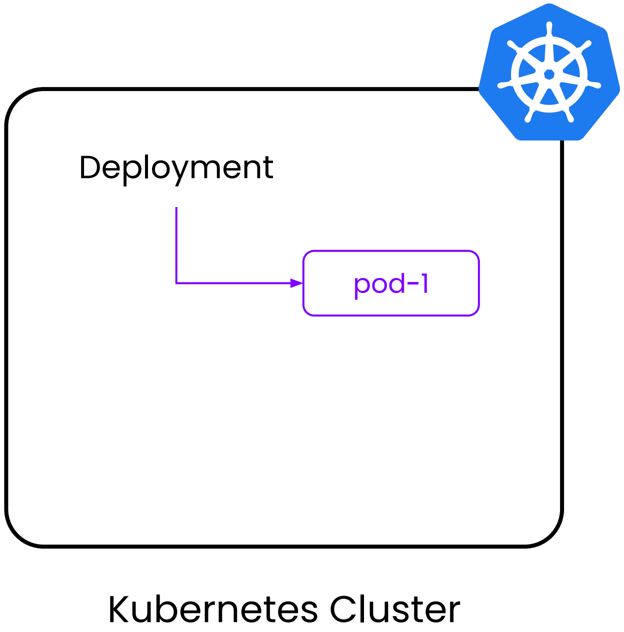
Kubernetes Cluster

1. Service Startup

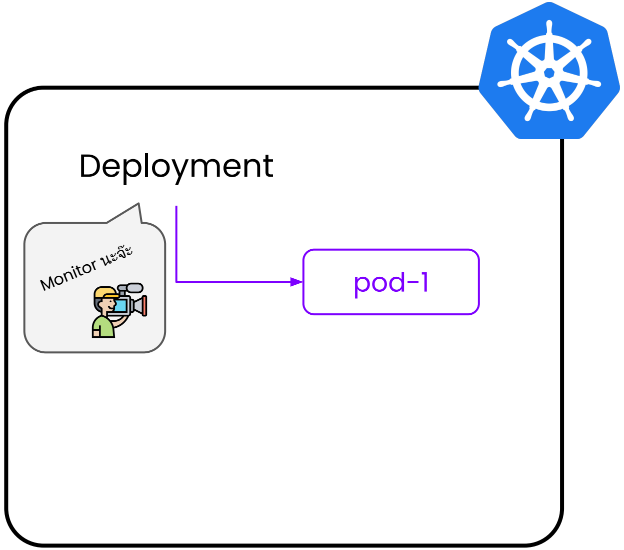


Kubernetes Cluster

1. Service Startup



1. Service Startup



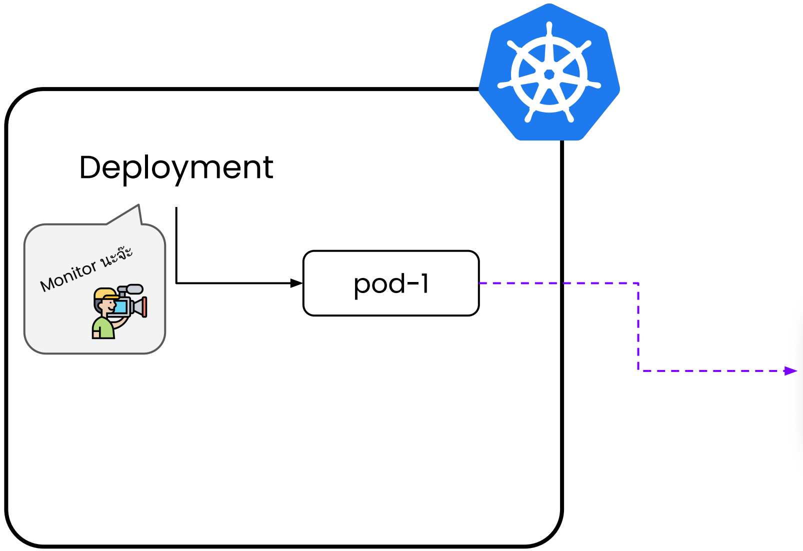
Deployment

Monitor นะจ๊ะ

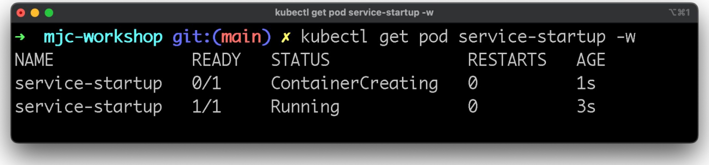
pod-1

Kubernetes Cluster

1. Service Startup



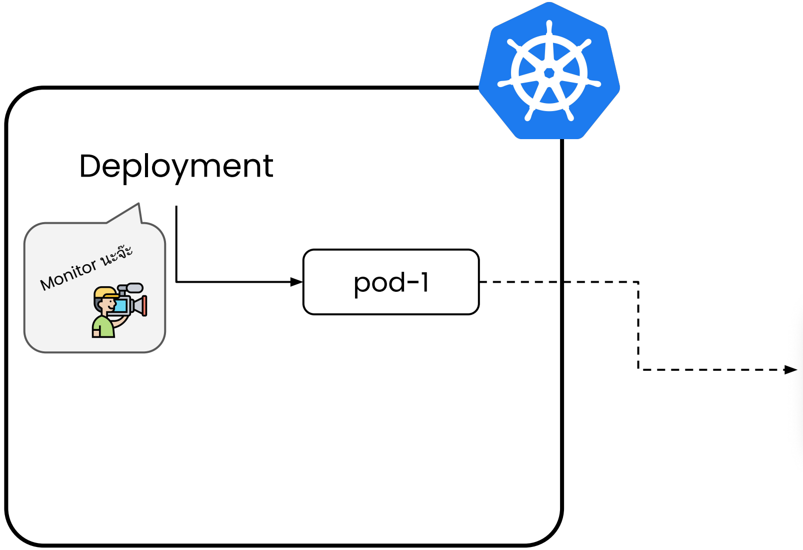
Kubernetes Cluster



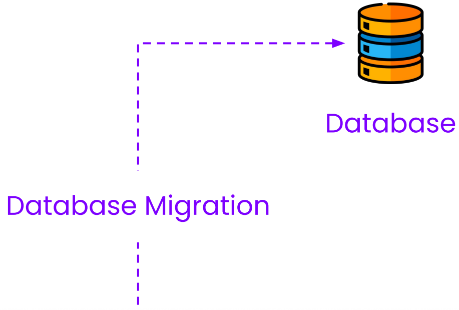
```
→ mjc-workshop git:(main) ✗ kubectl get pod service-startup -w
NAME              READY   STATUS              RESTARTS   AGE
service-startup   0/1     ContainerCreating   0          1s
service-startup   1/1     Running             0          3s
```

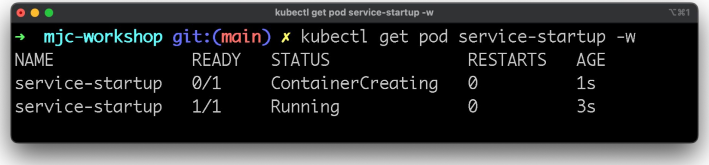
Service Startup

1. Service Startup



Kubernetes Cluster

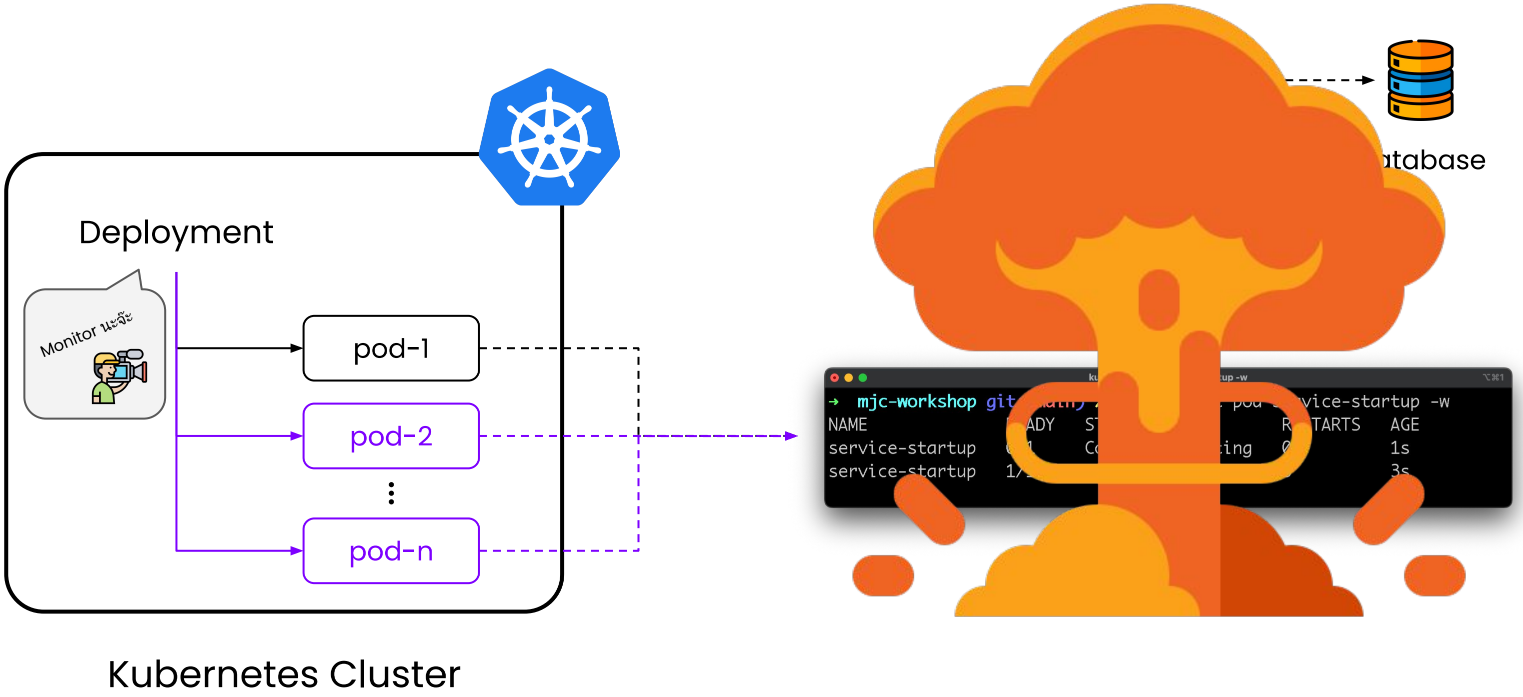




```
➜  mjc-workshop git:(main) ✗ kubectl get pod service-startup -w
NAME              READY   STATUS              RESTARTS   AGE
service-startup   0/1     ContainerCreating   0          1s
service-startup   1/1     Running             0          3s
```

Service Startup

1. Service Startup



```
→  mjc-workshop git         )             pod     vice-startup -w
NAME                READY   ST                    R  TARTS    AGE
service-startup     0 1     Co               ing   0          1s
service-startup     1/                                        3s
```

Summary of Service Startup

Summary of Service Startup

Pros

Summary of Service Startup

Pros

- The setup is **easy**

Summary of Service Startup

Pros

- The setup is **easy**
- **No** additional **configuration** is needed

Summary of Service Startup

Pros

- The setup is **easy**
- **No** additional **configuration** is needed

Limitation

Summary of **Service Startup**

Pros

- The setup is **easy**
- **No** additional **configuration** is needed

Limitation

- Running **more than one pod** of your service, you can run into difficulties

Summary of **Service Startup**

Pros

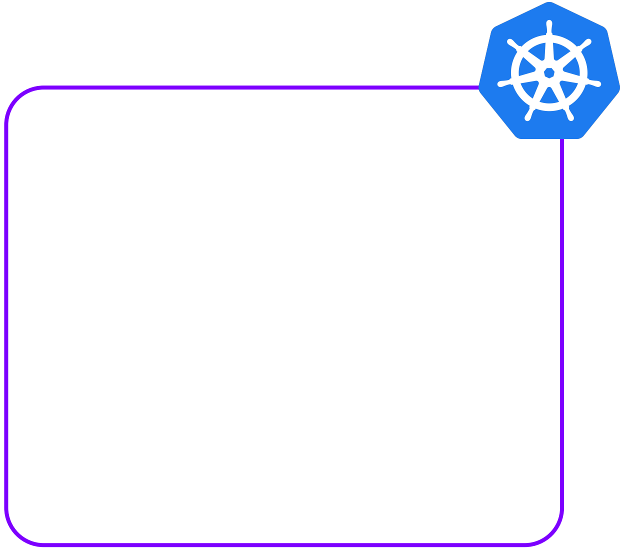
- The setup is **easy**
- **No** additional **configuration** is needed

Limitation

- Running **more than one pod** of your service, you can run into difficulties
- Big migration might exceeds **start-up time limit**

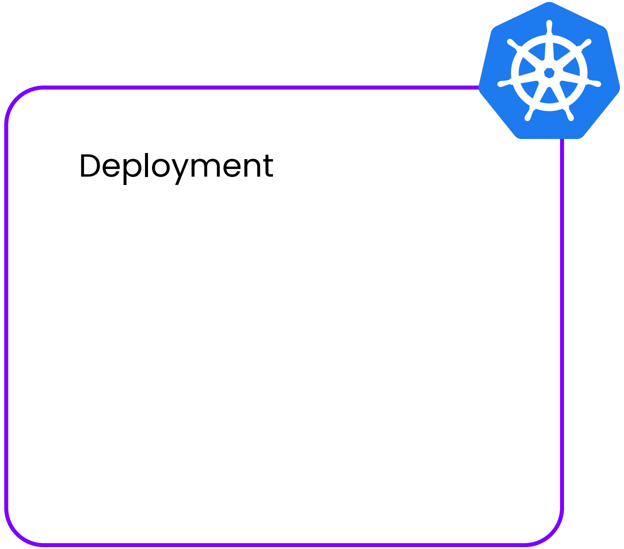
2. initContainers

2. initContainers



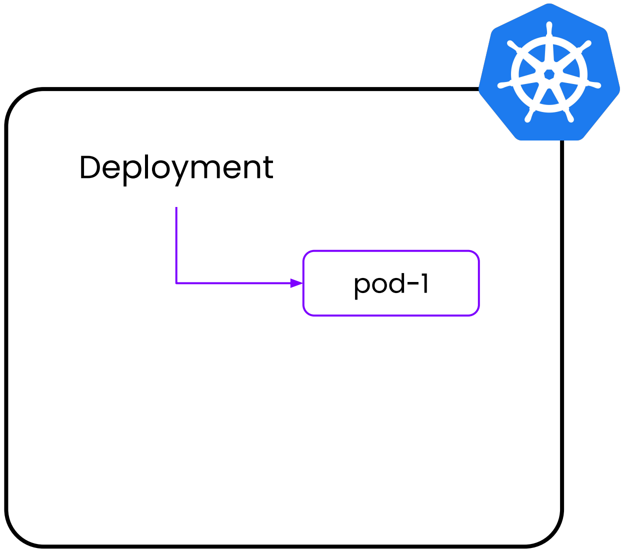
Kubernetes Cluster

2. initContainers



Kubernetes Cluster

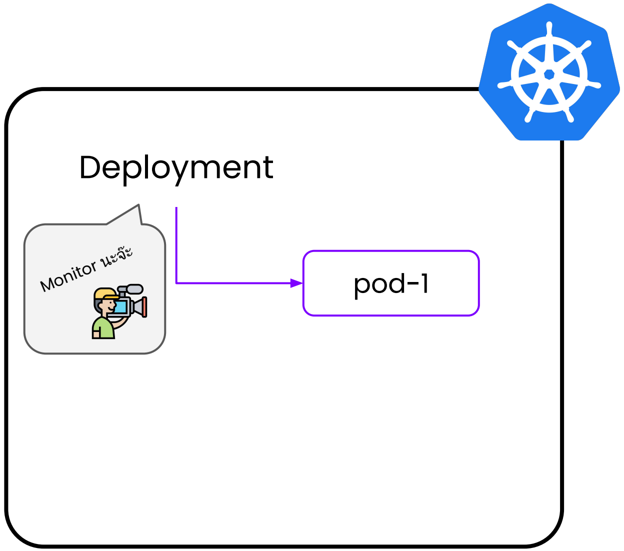
2. initContainers



Kubernetes Cluster

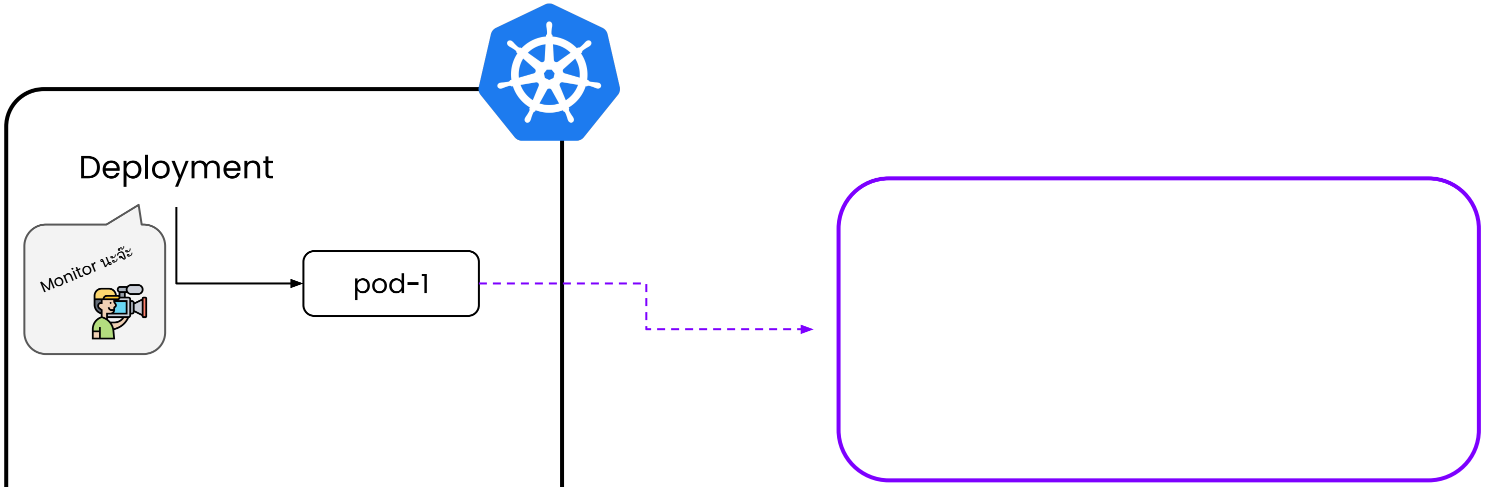
Jumpbox®

2. initContainers



Kubernetes Cluster

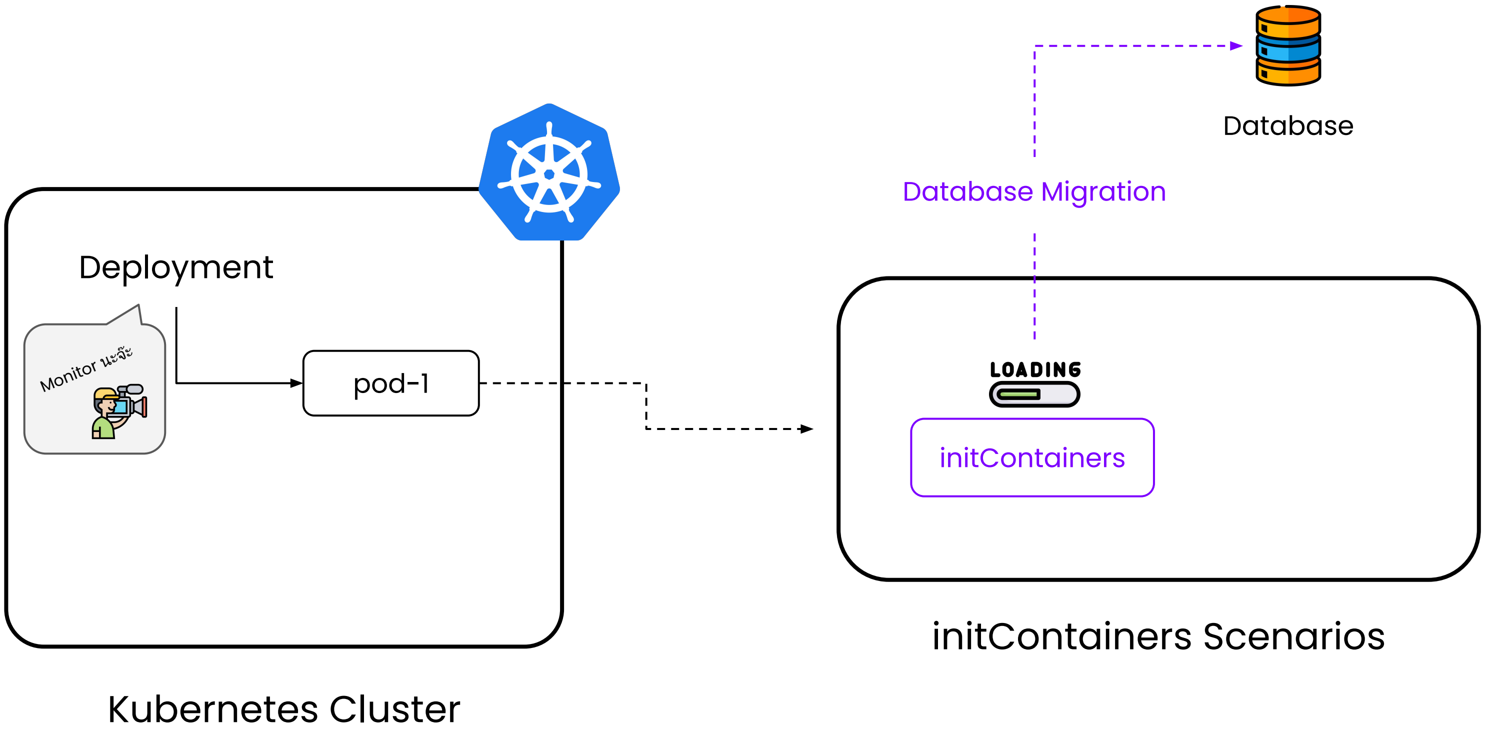
2. initContainers



Kubernetes Cluster

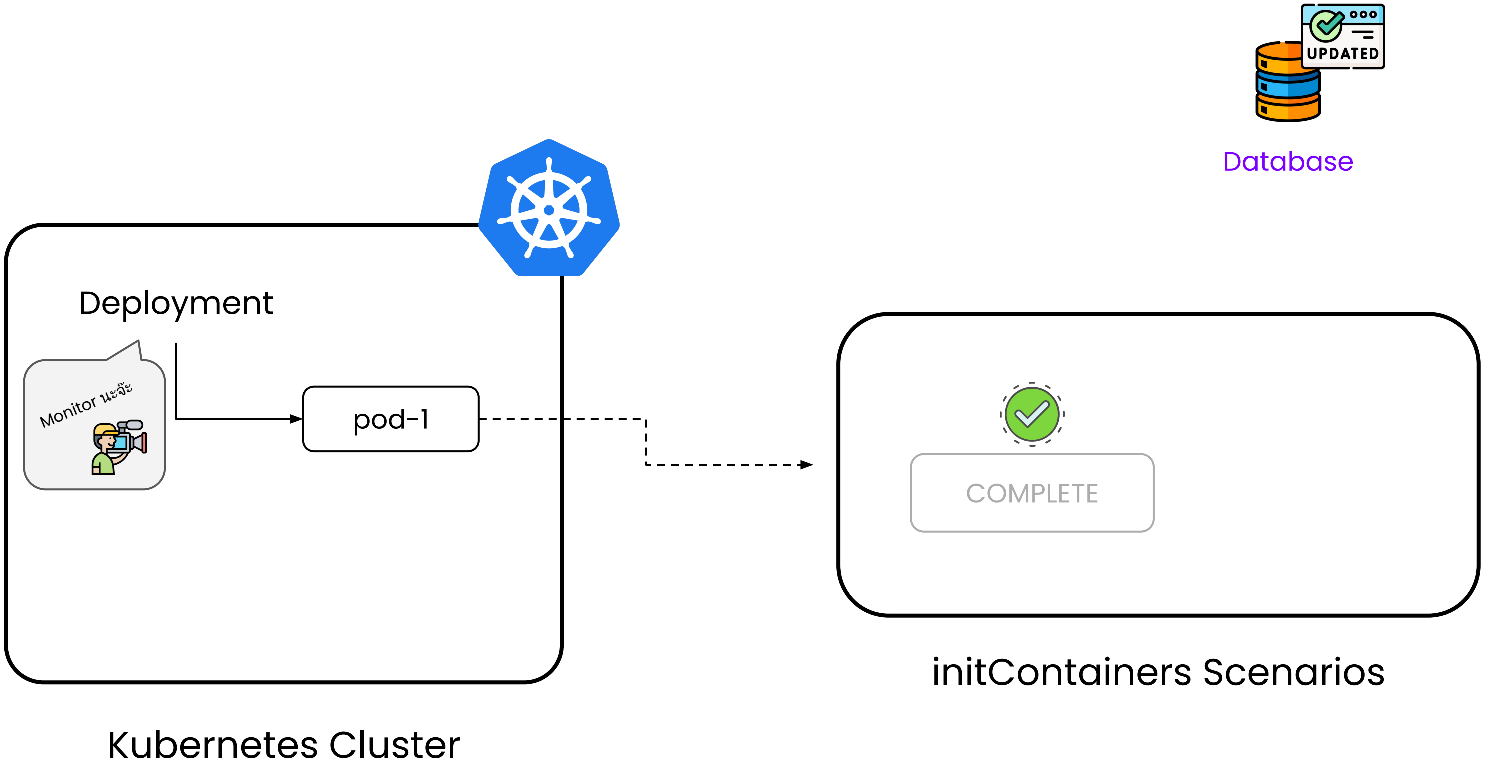
initContainers Scenarios

2. initContainers

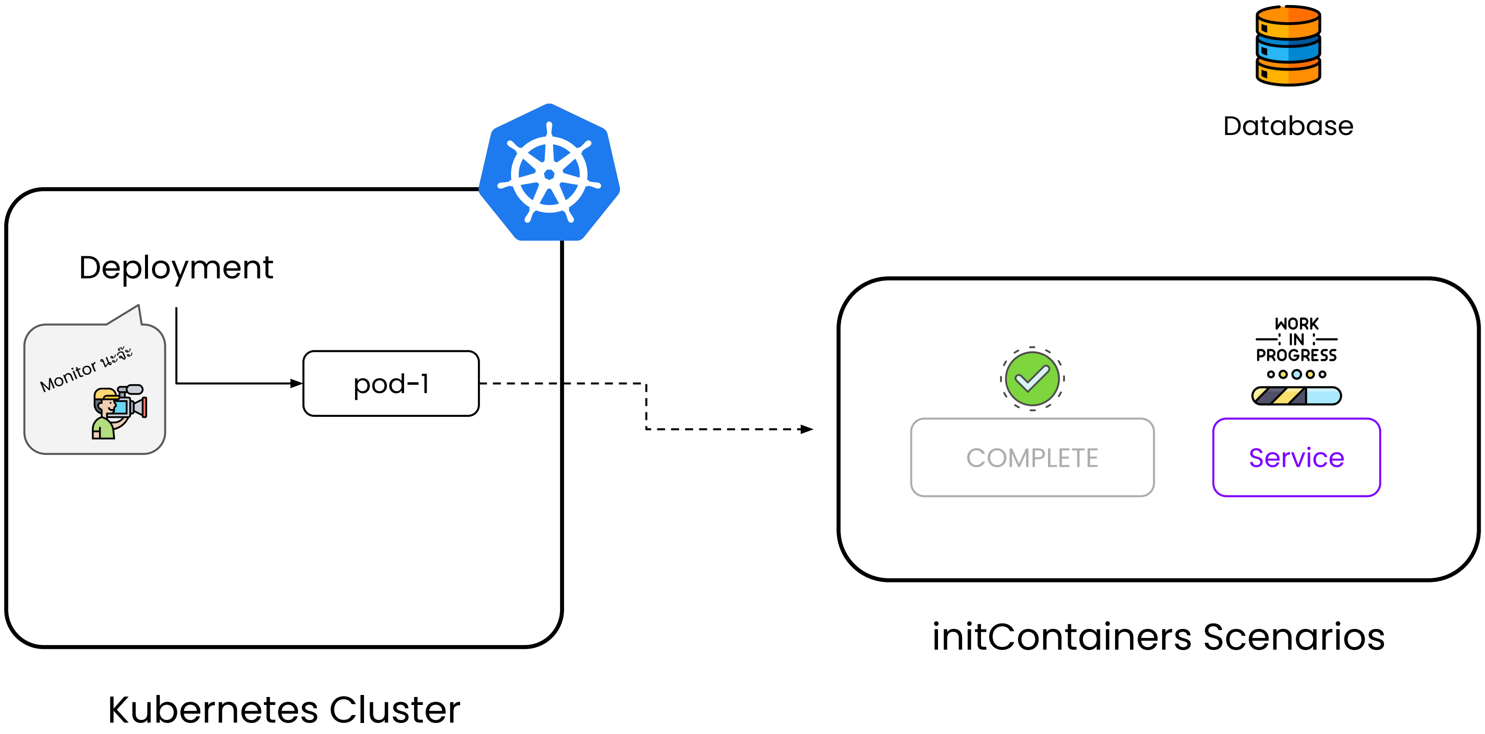


Jumpbox®

2. initContainers



2. initContainers



Database

Deployment

pod-1

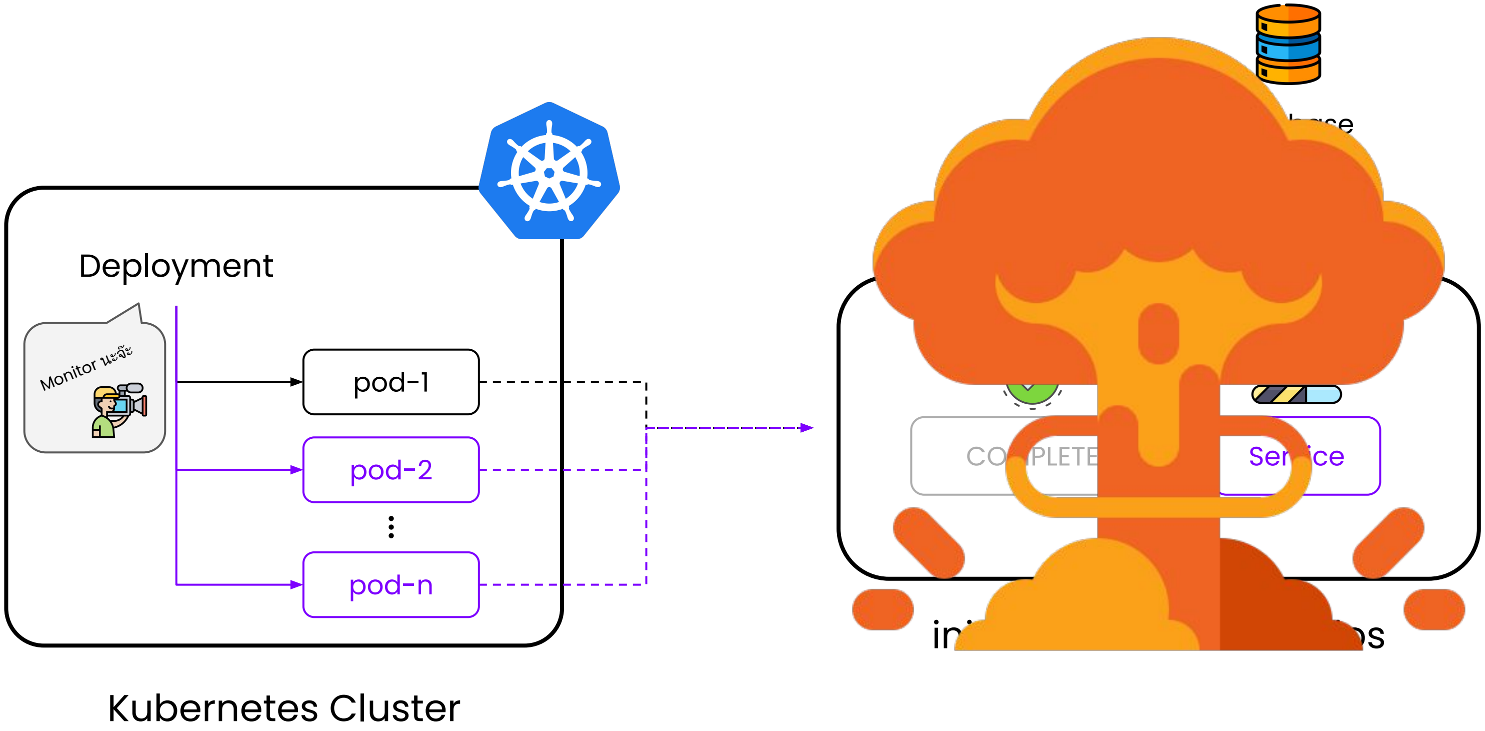
COMPLETE

Service

initContainers Scenarios

Kubernetes Cluster

2. initContainers



Summary of initContainers

Summary of initContainers

Pros

Summary of initContainers

Pros

- InitContainers are executed once **before** the **start of each pod**.

Summary of initContainers

Pros

- InitContainers are executed once **before** the **start of each pod**.
- **Separating** the service from the database migration logic

Summary of **initContainers**

Pros

- InitContainers are executed once **before** the **start of each pod.**
- **Separating** the service from the database migration logic

Limitation

Summary of **initContainers**

Pros

- InitContainers are executed once **before** the **start of each pod**.
- **Separating** the service from the database migration logic

Limitation

- Running **more than one pod** of your service, you can run into difficulties.

Summary of initContainers

Pros

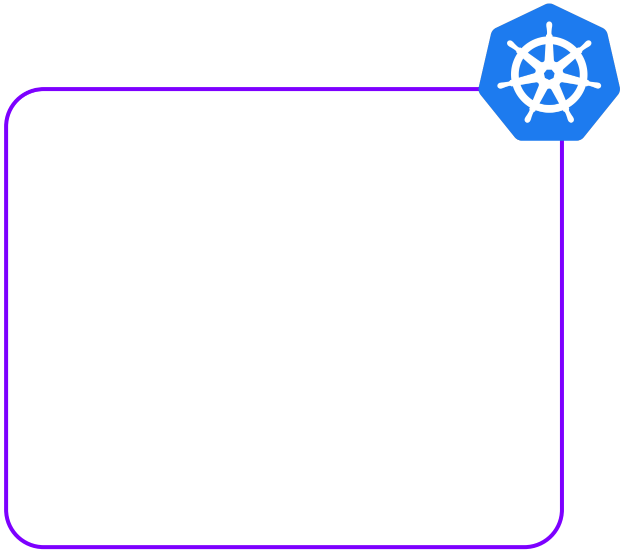
- InitContainers are executed once **before** the **start of each pod.**
- **Separating** the service from the database migration logic

Limitation

- Running **more than one pod** of your service, you can run into difficulties.
- Big migration might exceeds **start-up time limit**.

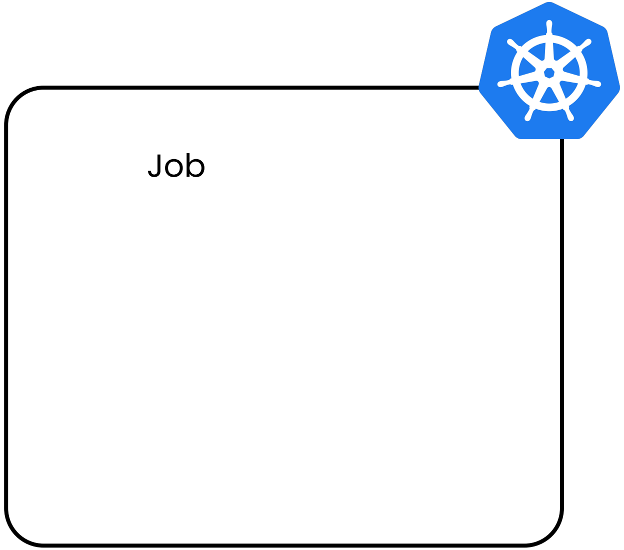
3. Kubernetes Job

3. Kubernetes Job



Kubernetes Cluster

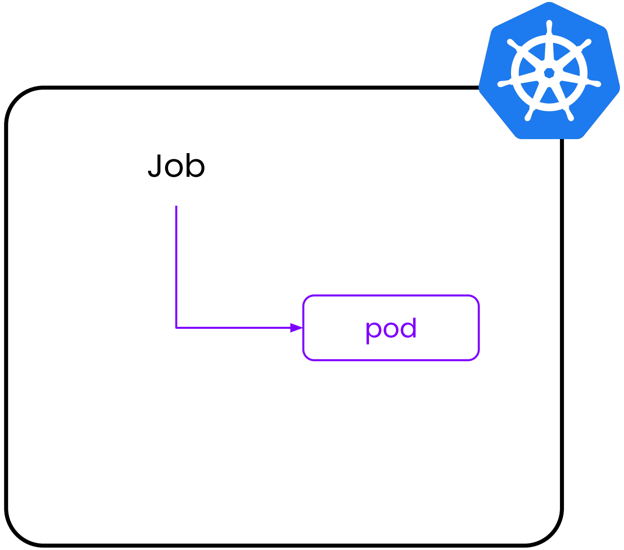
3. Kubernetes Job



Kubernetes Cluster

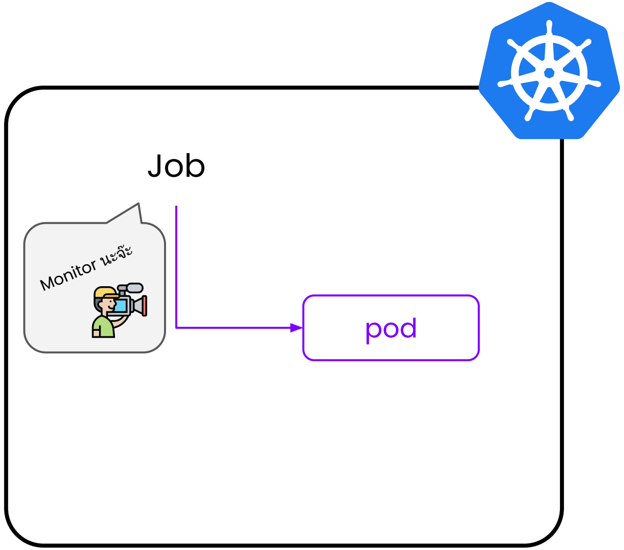
Jumpbox®

3. Kubernetes Job



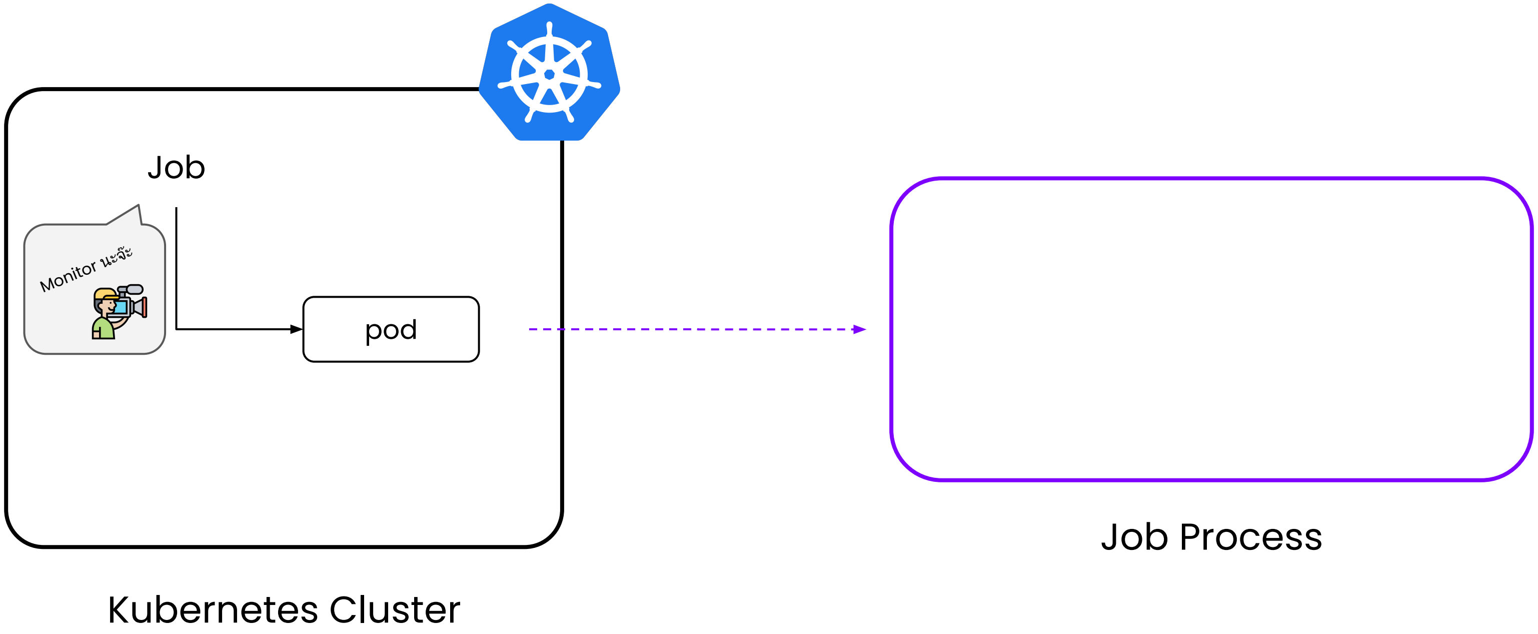
Kubernetes Cluster

3. Kubernetes Job



Kubernetes Cluster

3. Kubernetes Job



Job

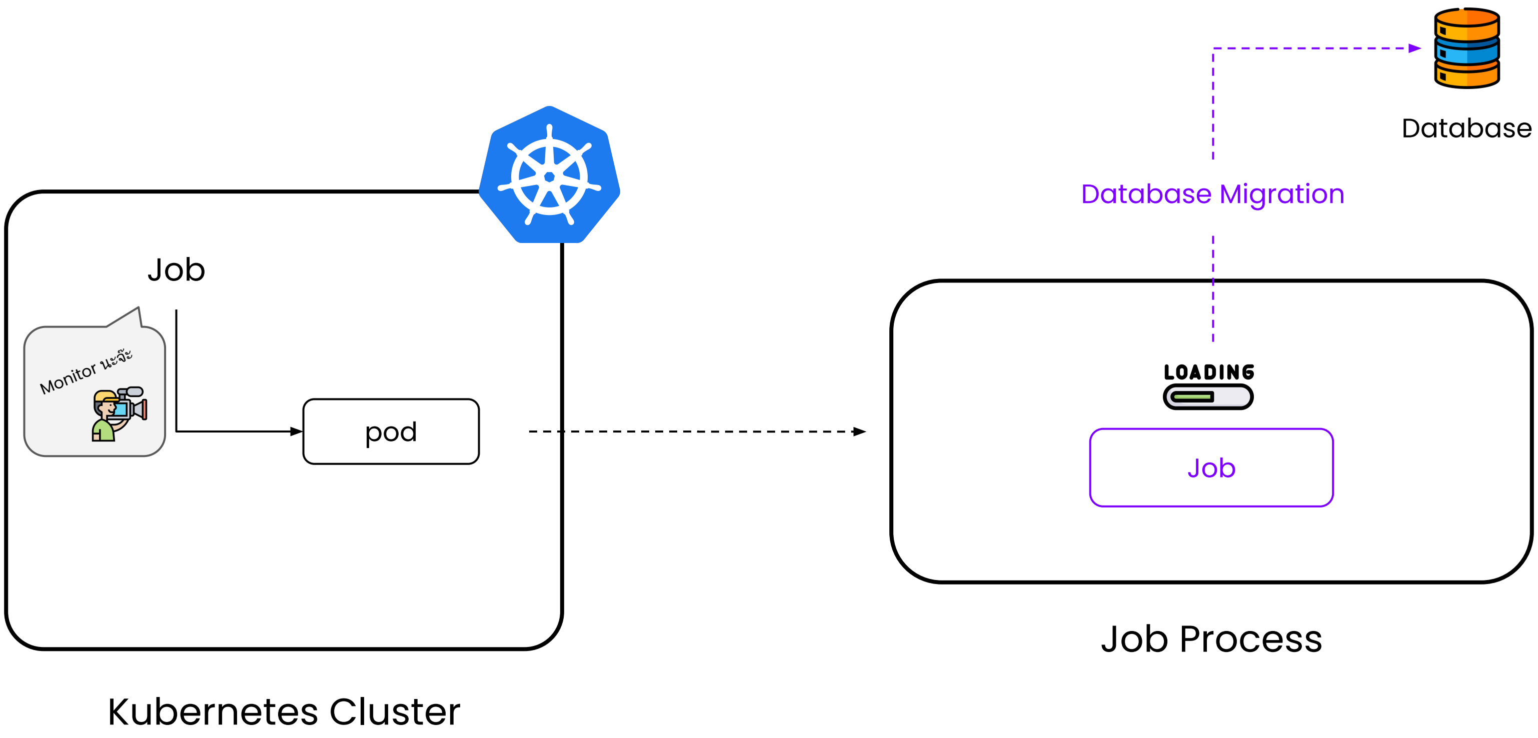
Monitor นะจ๊ะ

pod

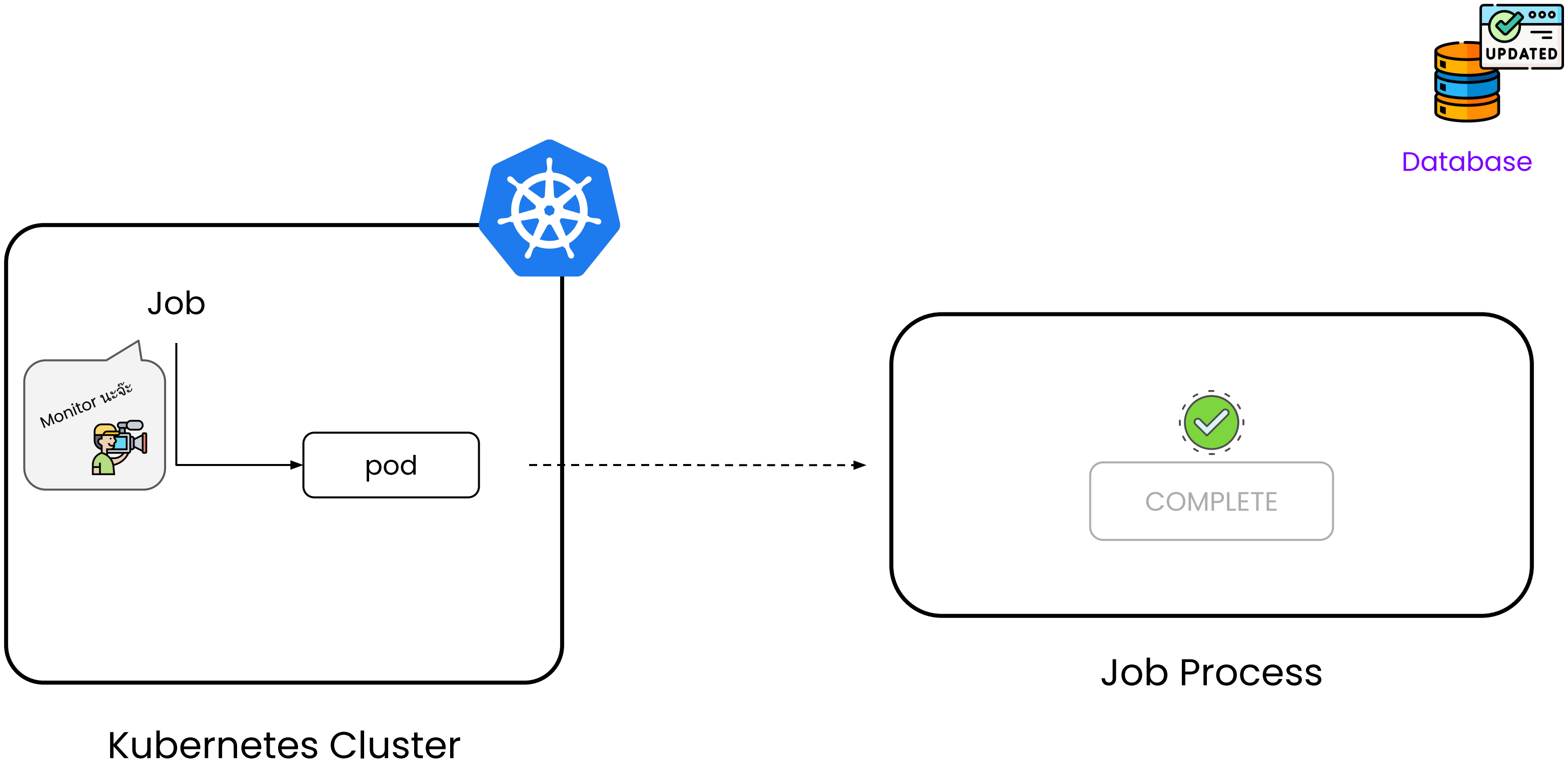
Kubernetes Cluster

Job Process

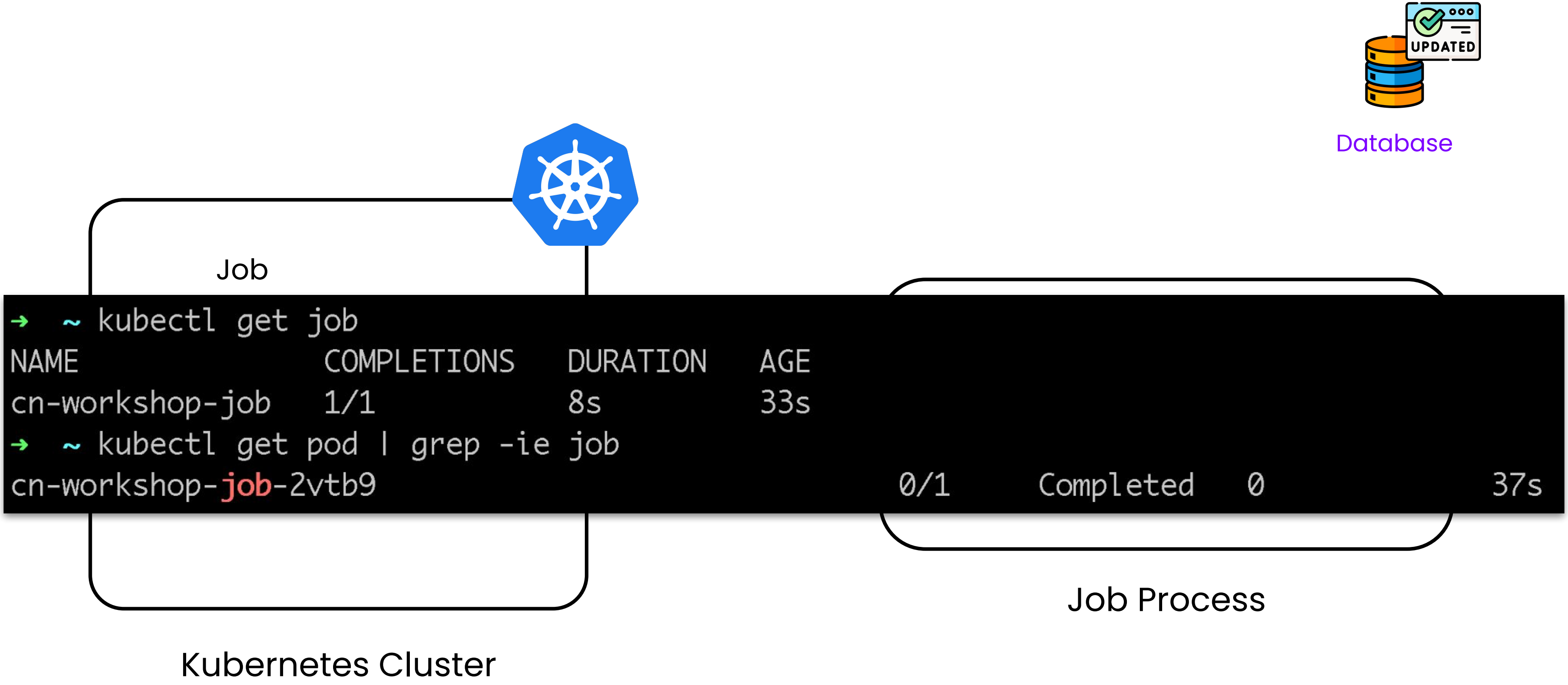
3. Kubernetes Job



3. Kubernetes Job



3. Kubernetes Job



```
➜  ~ kubectl get job
NAME              COMPLETIONS   DURATION   AGE
cn-workshop-job   1/1           8s         33s
➜  ~ kubectl get pod | grep -ie job
cn-workshop-job-2vtb9                              0/1     Completed   0              37s
```

Summary of Kubernetes Job

Summary of Kubernetes Job

Pros

Summary of Kubernetes Job

Pros

- **Separating** service from the database migration logic

Summary of Kubernetes Job

Pros

- **Separating** service from the database migration logic

Limitation

Summary of Kubernetes Job

Pros

- **Separating** service from the database migration logic

Limitation

- Kubernetes provides no native functionality to **wait for a Job to be executed** before starting pods.

4. CD Pipeline

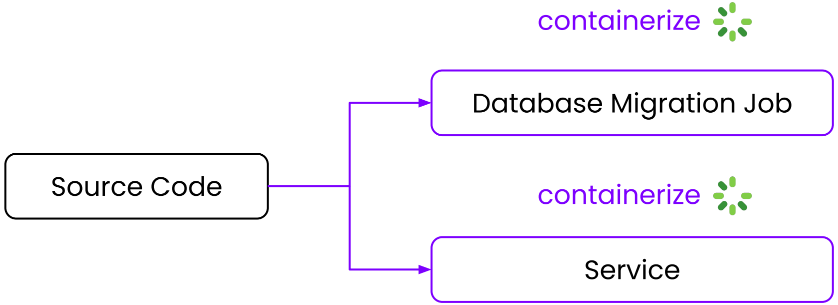
Jumpbox®

4. CD Pipeline

4. CD Pipeline

Source Code

4. CD Pipeline



Source Code

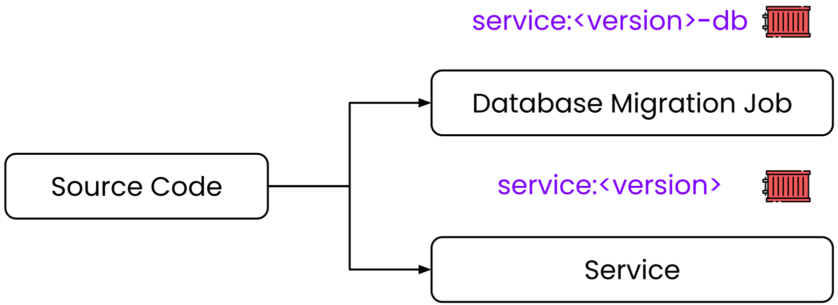
containerize

Database Migration Job

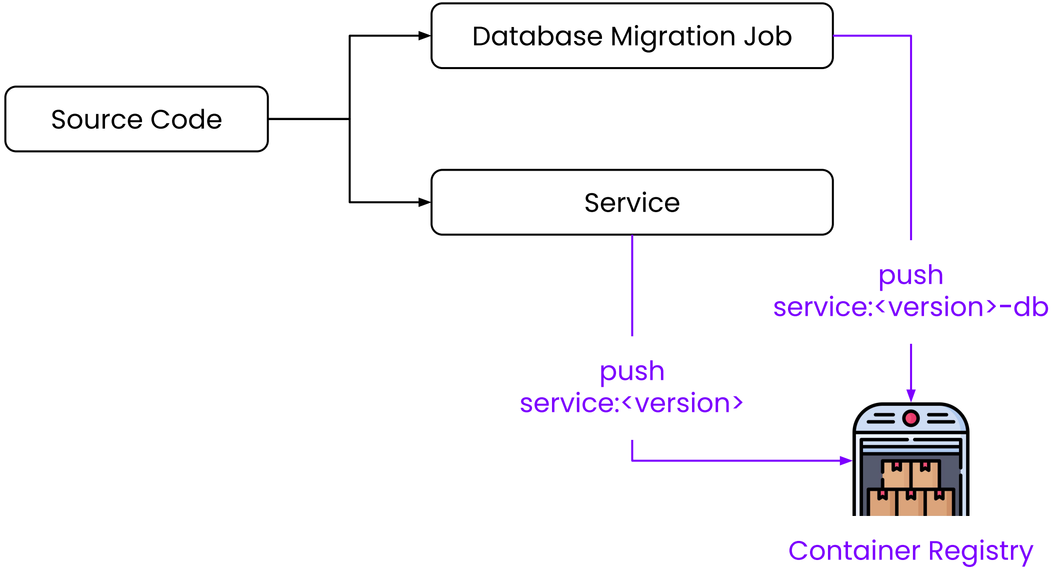
containerize

Service

4. CD Pipeline

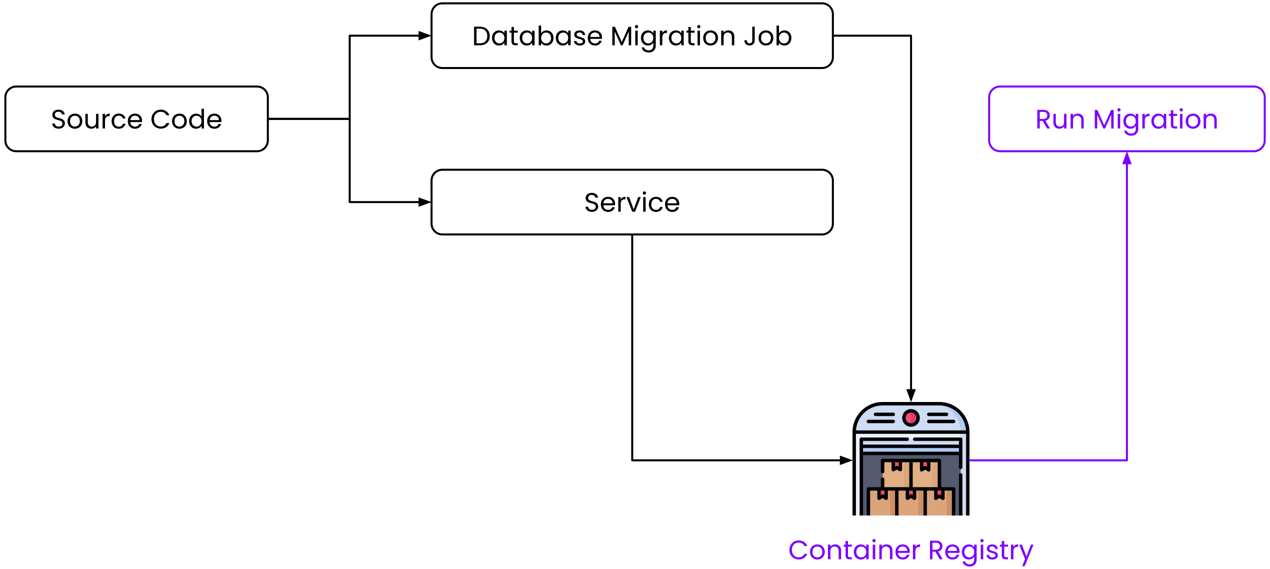


4. CD Pipeline



Jumpbox®

4. CD Pipeline


Source Code

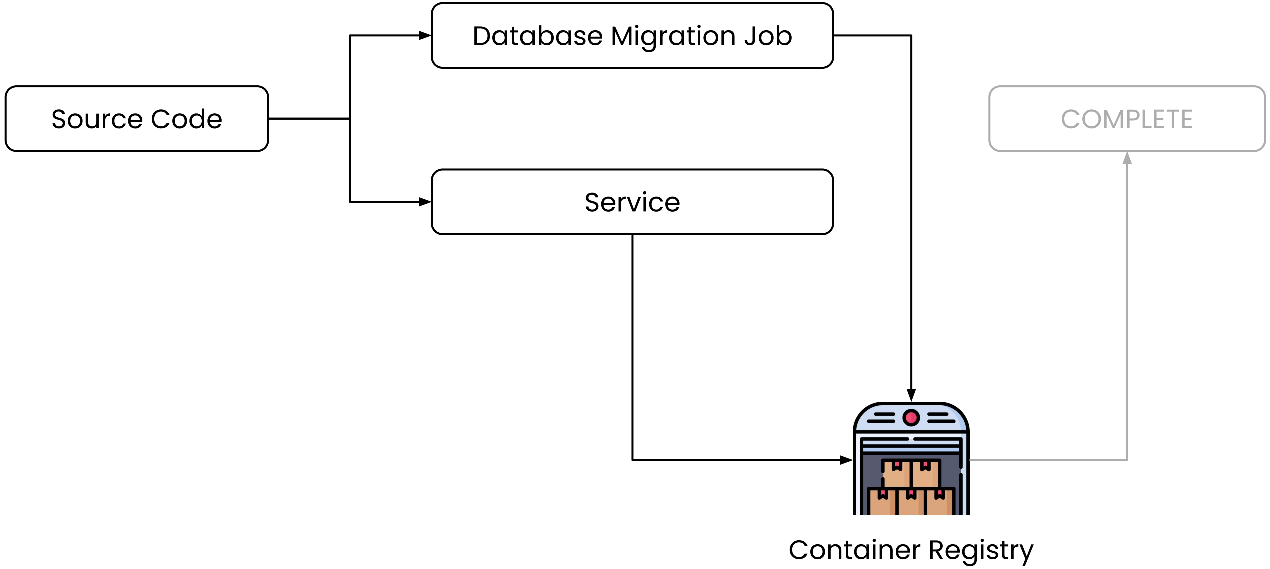
Database Migration Job

Service

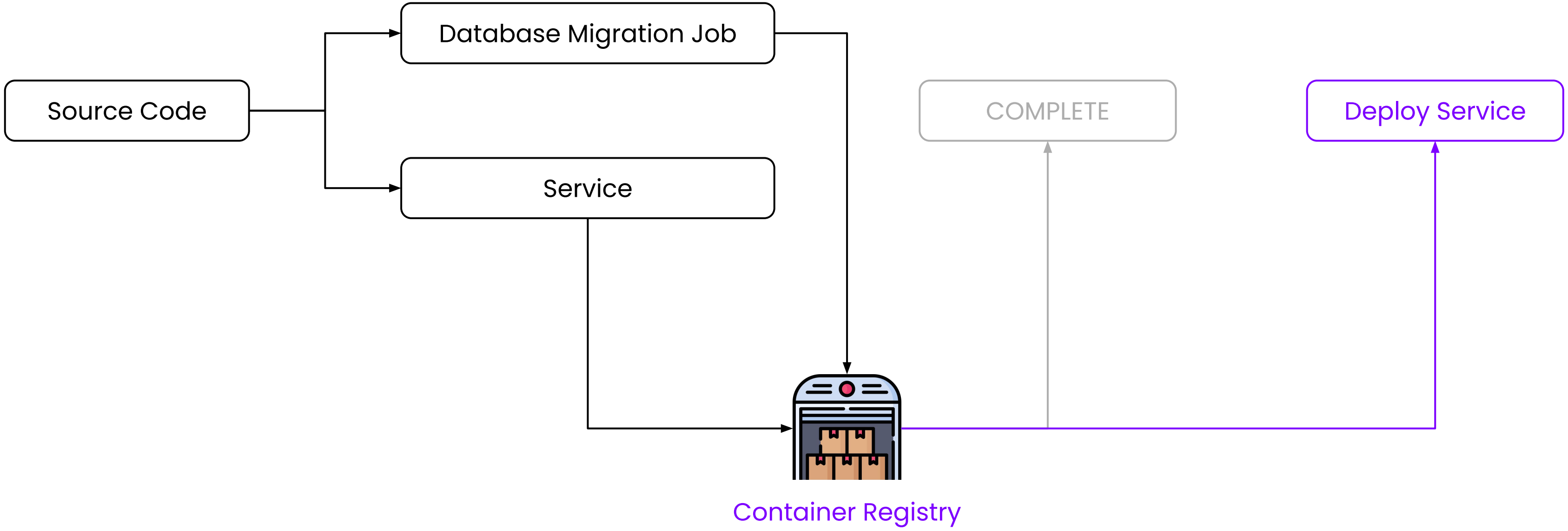
Run Migration

Container Registry

4. CD Pipeline



4. CD Pipeline



Jumpbox®

Summary of CD Pipeline

Jumpbox®

Summary of CD Pipeline

Pros

Jumpbox®

Summary of CD Pipeline

Pros

- A common solution

Summary of CD Pipeline

Pros

- A common solution

Limitation

Summary of CD Pipeline

Pros

- A common solution

Limitation

- CD pipeline needs to have the **access** to the **database secrets**.

Summary of CD Pipeline

Pros

- A common solution

Limitation

- CD pipeline needs to have the **access** to the **database secrets**.
- Database secrets must store into **two locations** instead of one centralized secret store.

The Most Practical Ideal way

The Most Practical Ideal way

Kubernetes Job + CD Pipeline

Workshop

Simple REST Application with Rust language

Jumpbox®

Simple REST Application with RUST

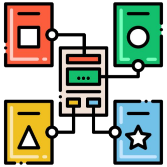
Jumpbox®

Simple **REST Application** with RUST

Scenario 1: **REST**

Simple REST Application with RUST

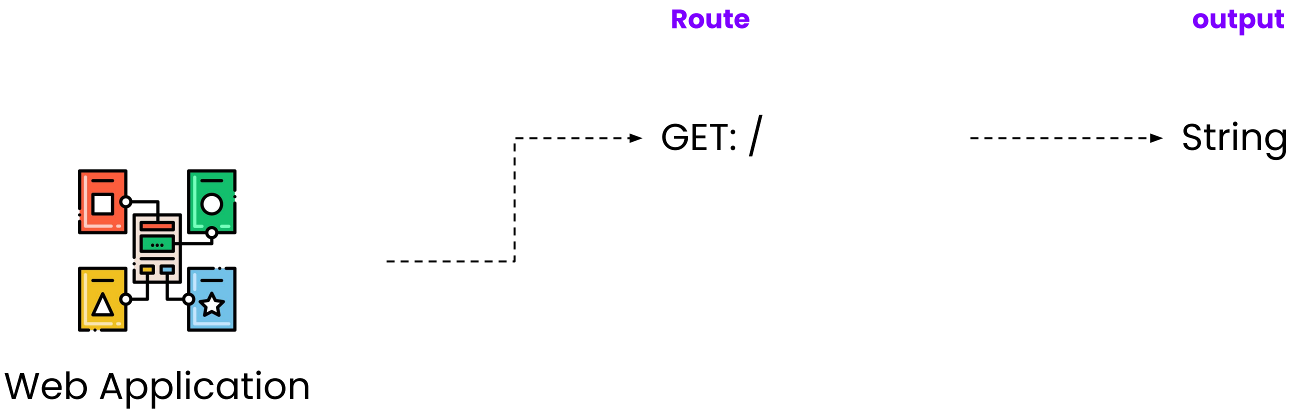
Scenario 1: REST



Web Application

Simple REST Application with RUST

Scenario 1: REST



Route

output

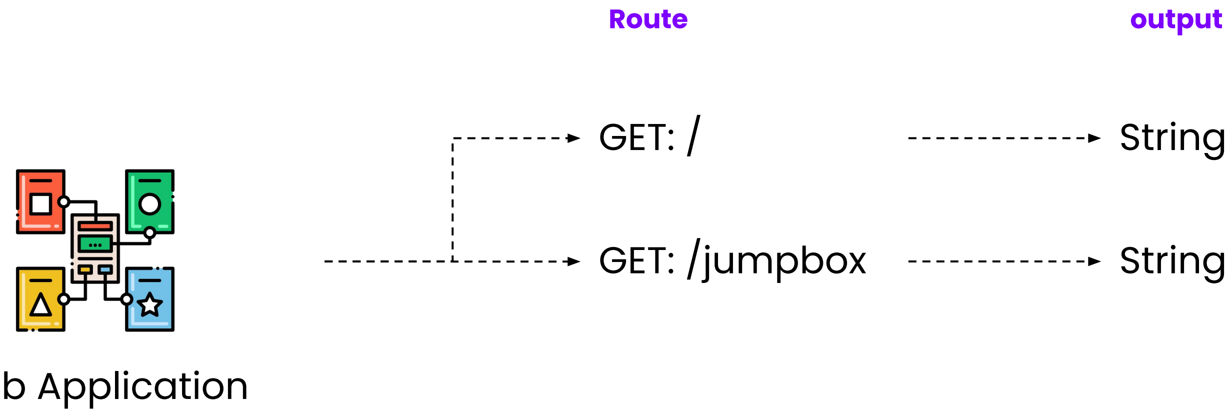
GET: /

String

Web Application

Simple REST Application with RUST

Scenario 1: REST



Route

output

GET: /

String

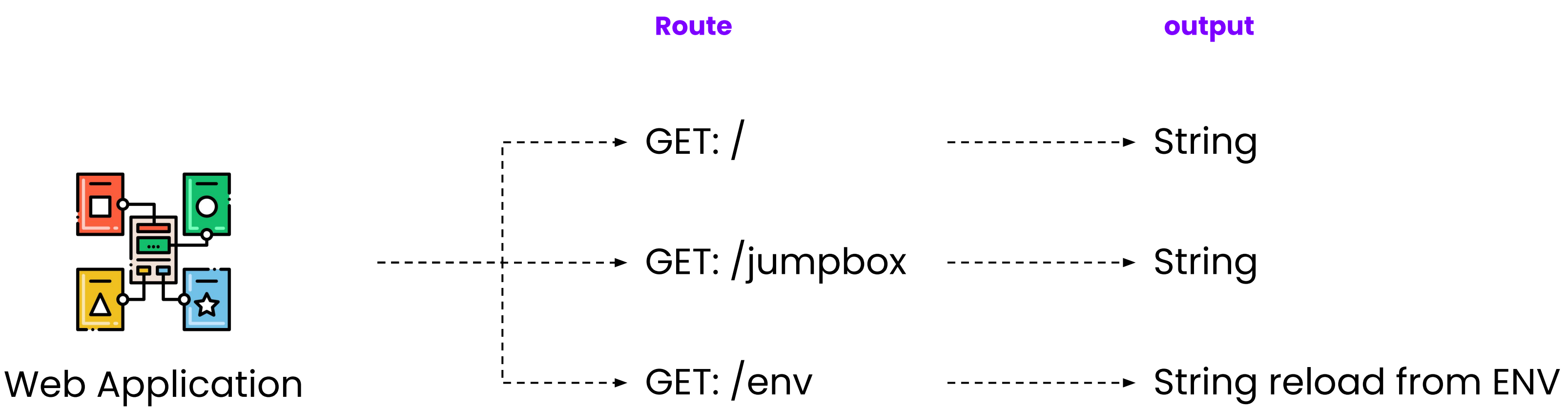
GET: /jumpbox

String

Web Application

Simple REST Application with RUST

Scenario 1: REST



Web Application

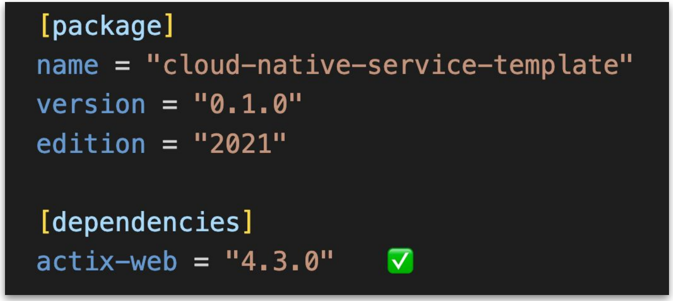
Route	output
GET: /	String
GET: /jumpbox	String
GET: /env	String reload from ENV

Simple REST Application with RUST

Scenario 1: REST with GET: /

Simple REST Application with RUST

Scenario 1: REST with Define route



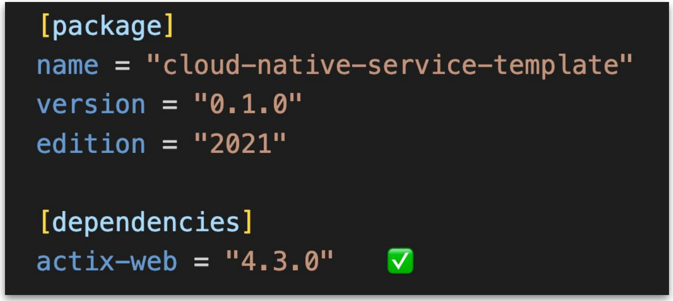
```toml
[package]
name = "cloud-native-service-template"
version = "0.1.0"
edition = "2021"

[dependencies]
actix-web = "4.3.0"    ✅
```

Install **Web**
dependencies

Jumpbox®

Simple REST Application with RUST

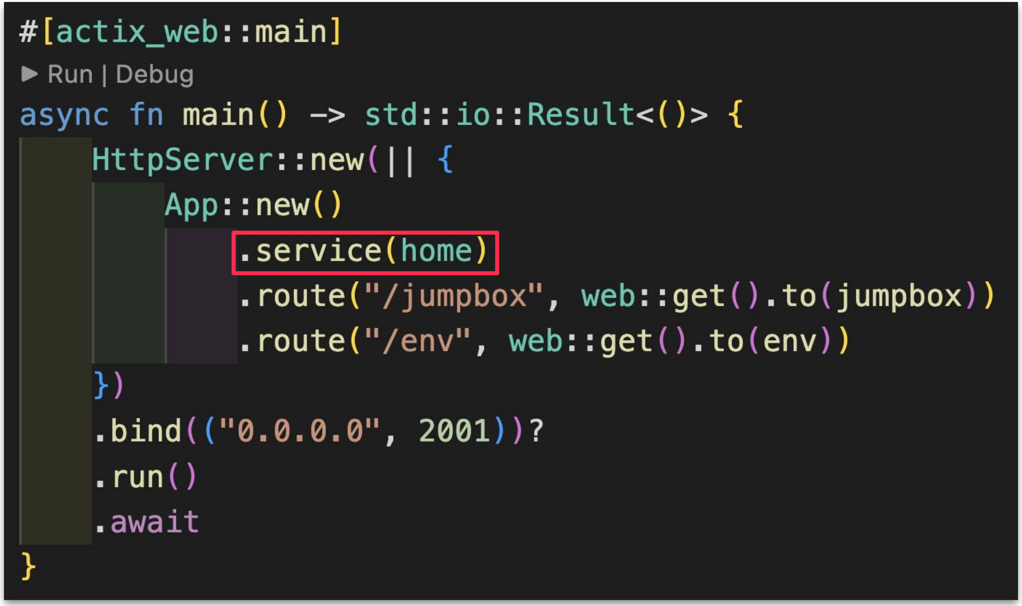
Scenario 1: REST with Define route



```
[package]
name = "cloud-native-service-template"
version = "0.1.0"
edition = "2021"

[dependencies]
actix-web = "4.3.0"    ✅
```

Install **Web** dependencies

Coding



```
#[actix_web::main]
▶ Run | Debug
async fn main() -> std::io::Result<()> {
    HttpServer::new(|| {
        App::new()
            .service(home)
            .route("/jumpbox", web::get().to(jumpbox))
            .route("/env", web::get().to(env))
    })
    .bind(("0.0.0.0", 2001))?
    .run()
    .await
}
```

Jumpbox®

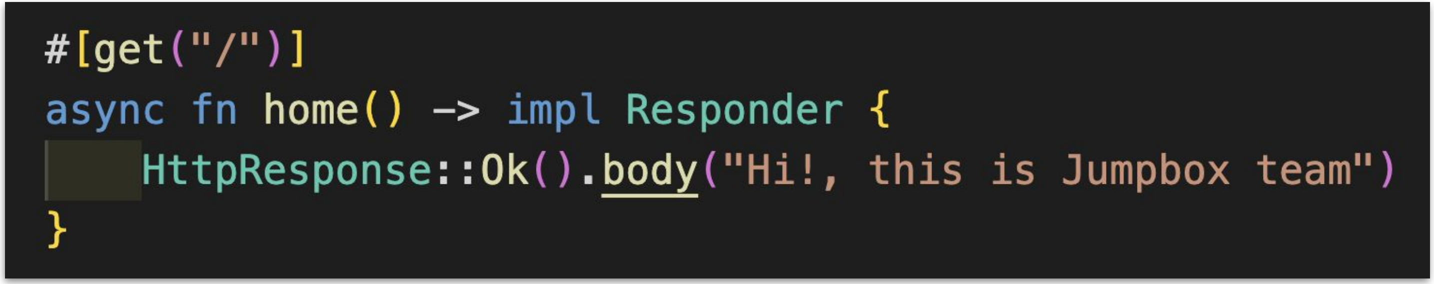
Simple **REST Application** with RUST

Scenario 1: REST with **GET: /**

Jumpbox®

Simple REST Application with RUST

Scenario 1: REST with GET: /



```
#[get("/")]
async fn home() -> impl Responder {
    HttpResponse::Ok().body("Hi!, this is Jumpbox team")
}
```

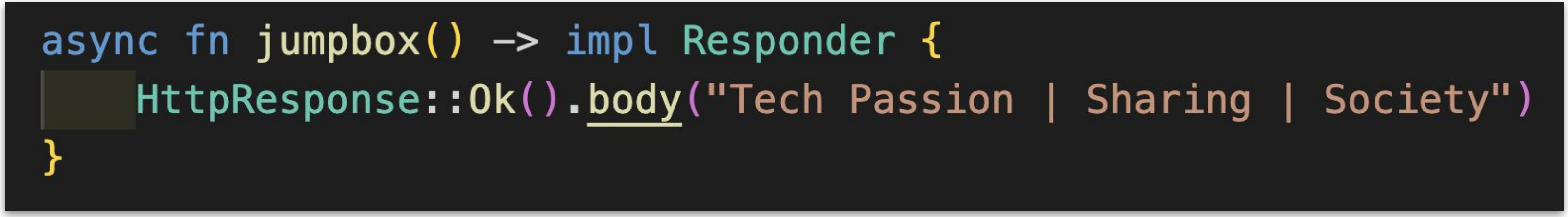
Implement **home** function

Simple **REST Application** with RUST

Scenario 1: REST with **GET: /jumpbox**

Simple **REST Application** with RUST

Scenario 1: REST with GET: /jumpbox



```
async fn jumpbox() -> impl Responder {
    HttpResponse::Ok().body("Tech Passion | Sharing | Society")
}
```

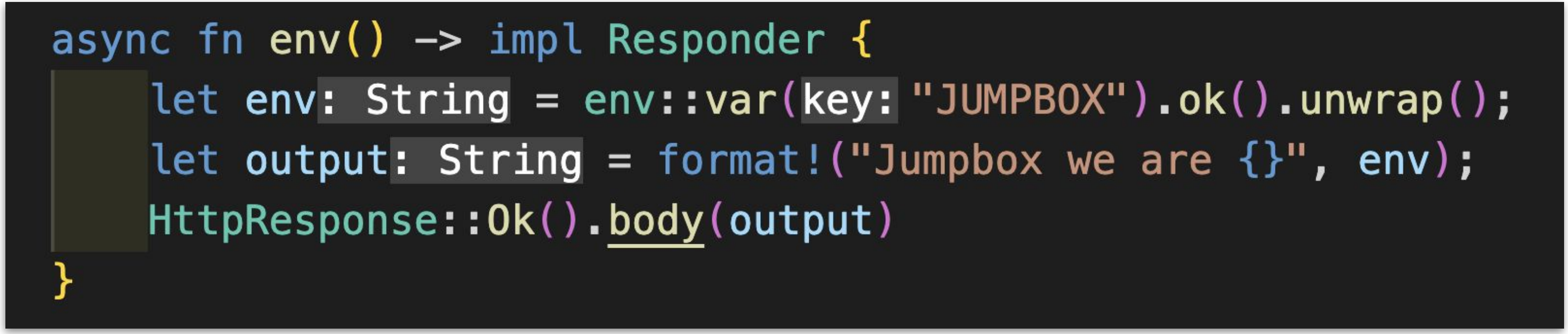
Implement **jumpbox** path function

Simple REST Application with RUST

Scenario 1: REST with GET: /env

Simple REST Application with RUST

Scenario 1: REST with GET: /env

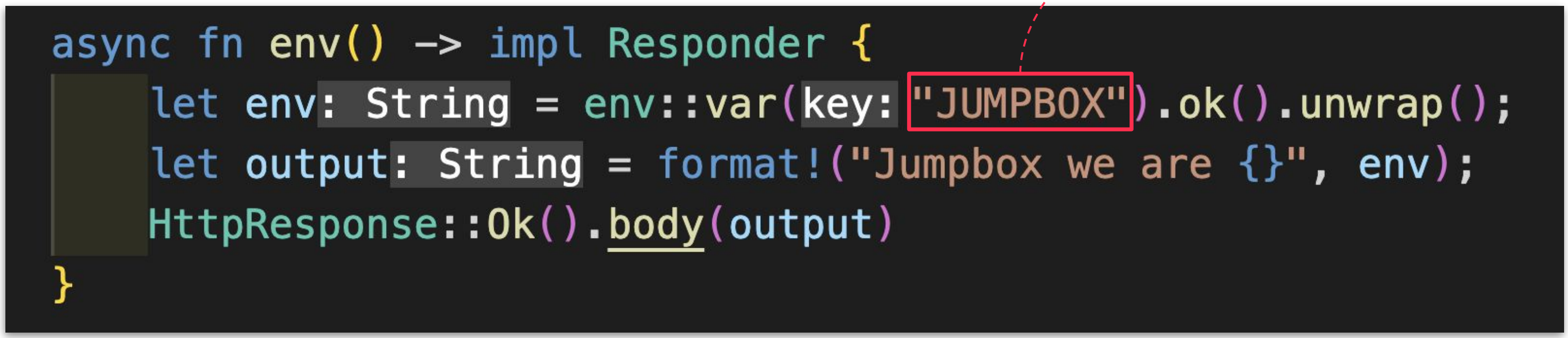


```
async fn env() -> impl Responder {
    let env: String = env::var(key: "JUMPBOX").ok().unwrap();
    let output: String = format!("Jumpbox we are {}", env);
    HttpResponse::Ok().body(output)
}
```

Implement **env** path function

Simple **REST Application** with RUST

Scenario 1: REST with **GET: /env**



Define what you want

```
async fn env() -> impl Responder {
    let env: String = env::var(key: "JUMPBOX").ok().unwrap();
    let output: String = format!("Jumpbox we are {}", env);
    HttpResponse::Ok().body(output)
}
```

Implement **env** path function

Jumpbox®

Real Demo

with cargo CLI

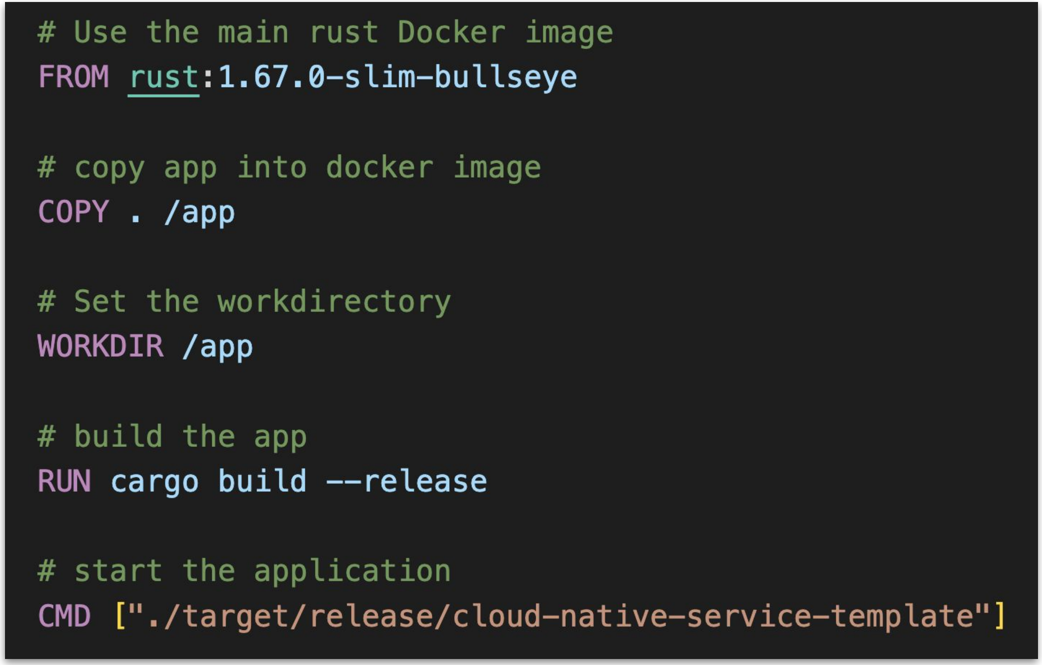
Jumpbox®

Simple REST Application with RUST

Scenarios 2: Containerization

Simple REST Application with RUST

Scenarios 2: Containerization



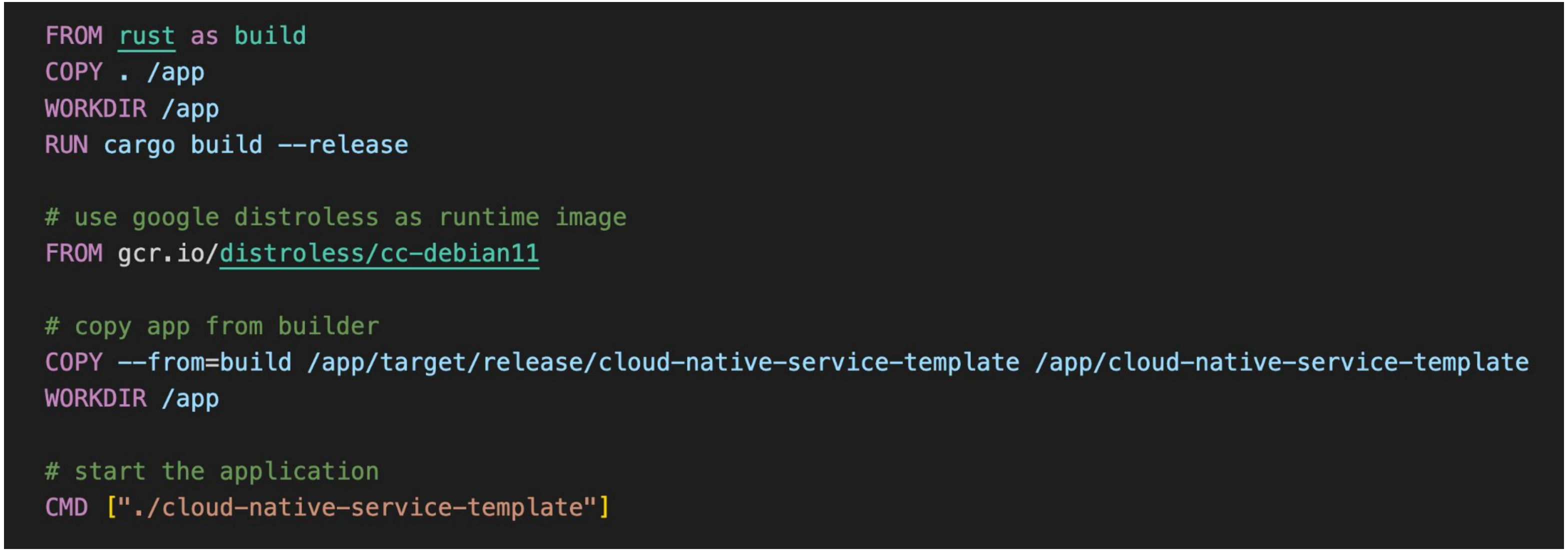
```dockerfile
# Use the main rust Docker image
FROM rust:1.67.0-slim-bullseye

# copy app into docker image
COPY . /app

# Set the workdirectory
WORKDIR /app

# build the app
RUN cargo build --release

# start the application
CMD ["./target/release/cloud-native-service-template"]
```

Simple REST Application with RUST

Scenarios 2: Containerization with Distroless image

Simple REST Application with RUST

Scenarios 2: Containerization with Distroless image



```
FROM rust as build
COPY . /app
WORKDIR /app
RUN cargo build --release

# use google distroless as runtime image
FROM gcr.io/distroless/cc-debian11

# copy app from builder
COPY --from=build /app/target/release/cloud-native-service-template /app/cloud-native-service-template
WORKDIR /app

# start the application
CMD ["./cloud-native-service-template"]
```

Real Demo

with Docker

Real Demo

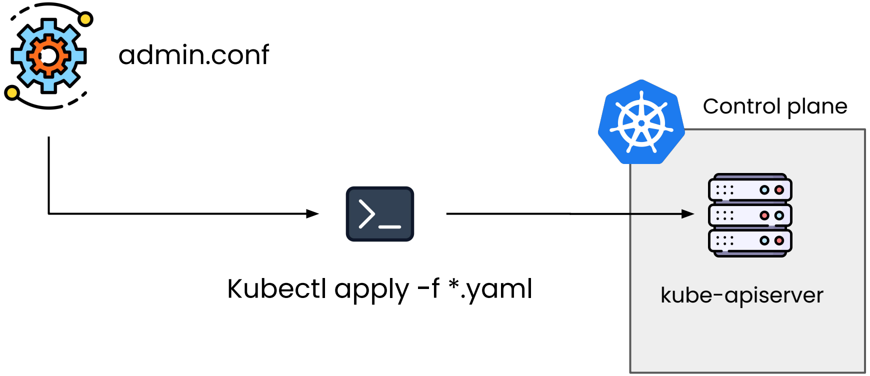
Kubernetes on Proen Cloud

Kubernetes on Proen Cloud

admin.conf

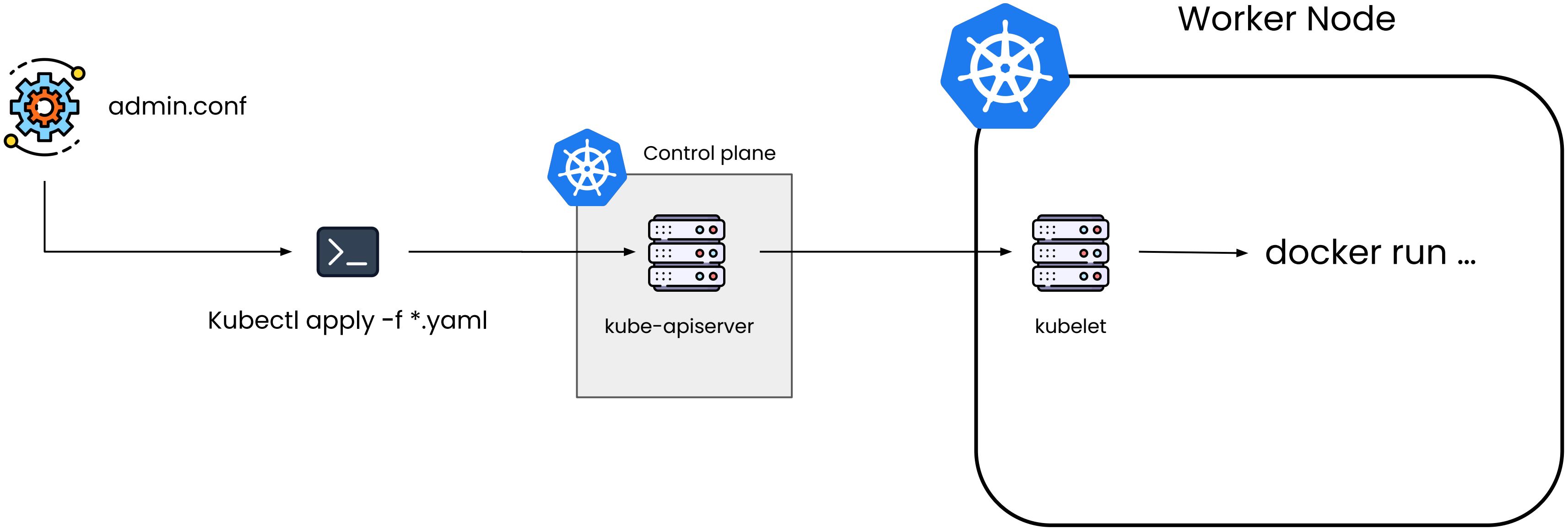
Jumpbox®

Kubernetes on Proen Cloud



Jumpbox®

Kubernetes on Proen Cloud



Jumpbox®

Kubernetes on Proen Cloud

Kubernetes on Proen Cloud

Scenarios:

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /
   b. GET: /jumpbox

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /
   b. GET: /jumpbox
   c. GET: /env

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /
   b. GET: /jumpbox
   c. GET: /env
4. Modify Configmap **ENV: JUMPBOX** to new value

Kubernetes on Proen Cloud

Scenarios:

1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /
   b. GET: /jumpbox
   c. GET: /env
4. Modify Configmap **ENV: JUMPBOX** to new value
5. Restart RUST Service POD

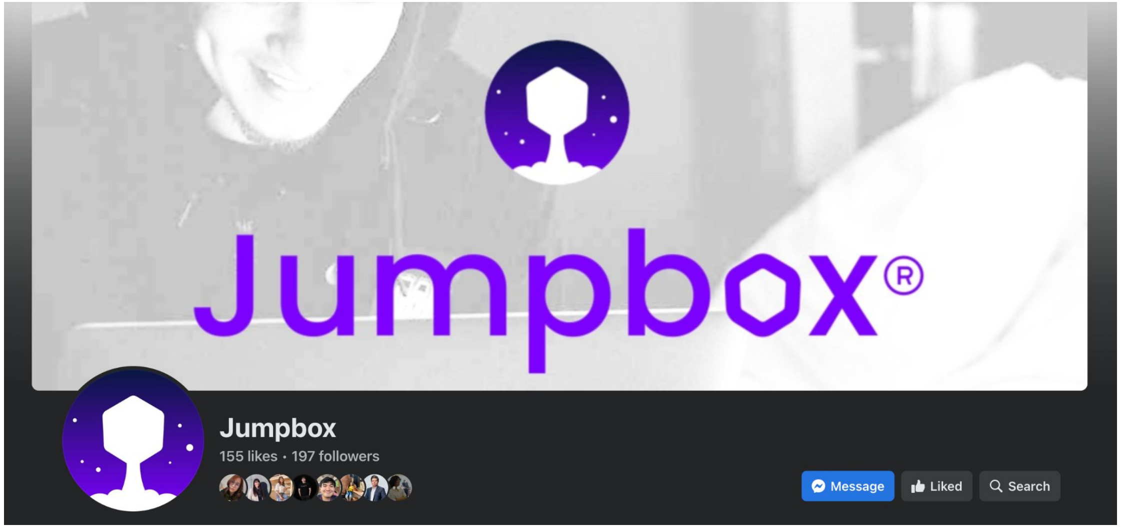
Kubernetes on Proen Cloud

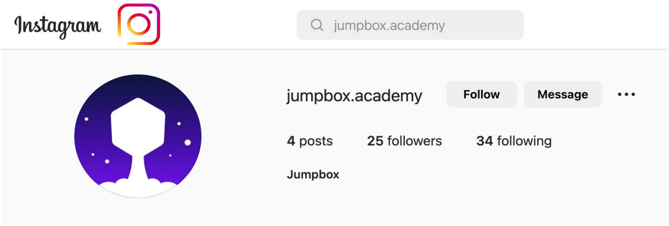
Scenarios:

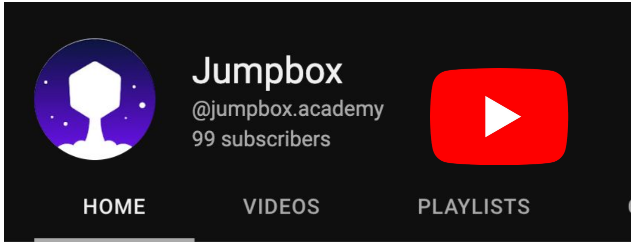
1. Apply Kubernetes manifest
2. Test endpoint by execute to **Temporary POD (nginx)** and then curl to **RUST service**
3. Test curl with
   a. GET: /
   b. GET: /jumpbox
   c. GET: /env
4. Modify Configmap **ENV: JUMPBOX** to new value
5. Restart RUST Service POD
6. Test curl GET: /env for getting new value from RUST service

Q&A

ท่านใดอยากได้ slide ฝาก **Like FB fan page Jumpbox** แล้ว Capture screen ส่งมาที่ inbox fanpage เราจะส่ง pdf ให้







Jumpbox®