

Deploy microservices





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...
Timeline About Friends 3,138 Photos More

When did you work at Opendream?
... 22 Pending Items

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Intro
Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

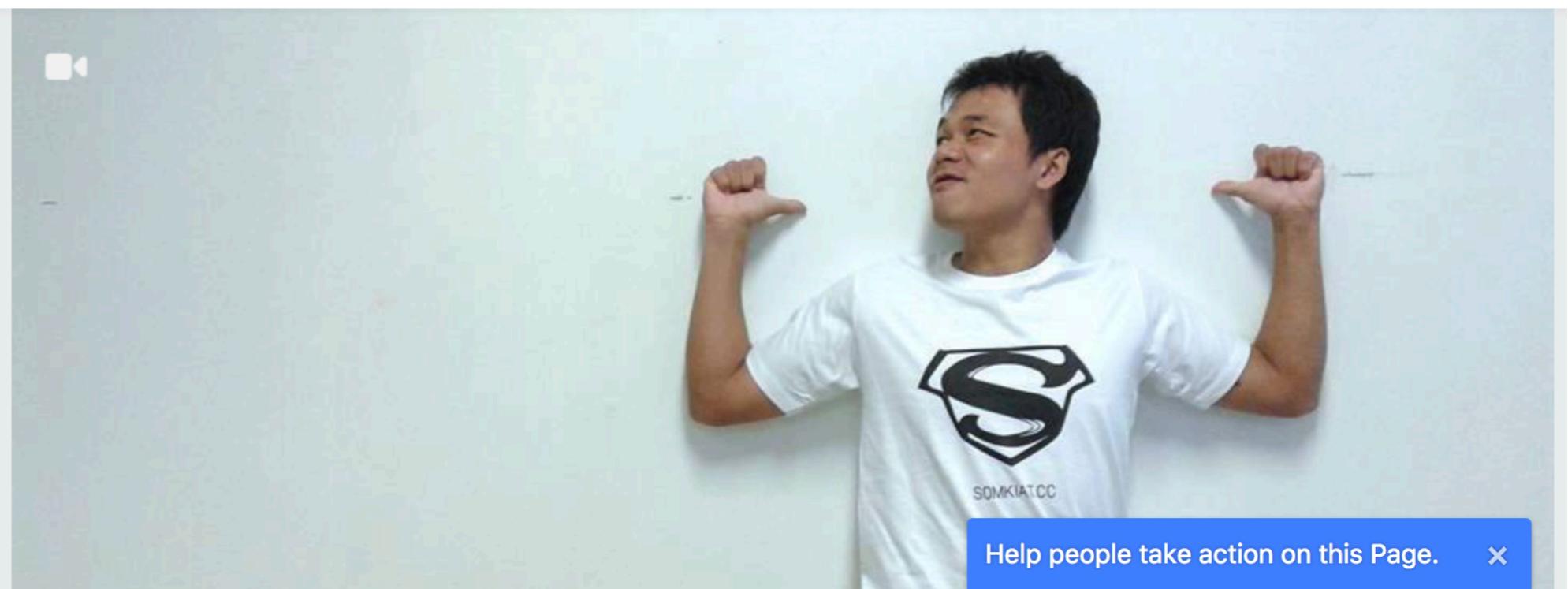
@somkiat.cc

Home

Posts

Videos

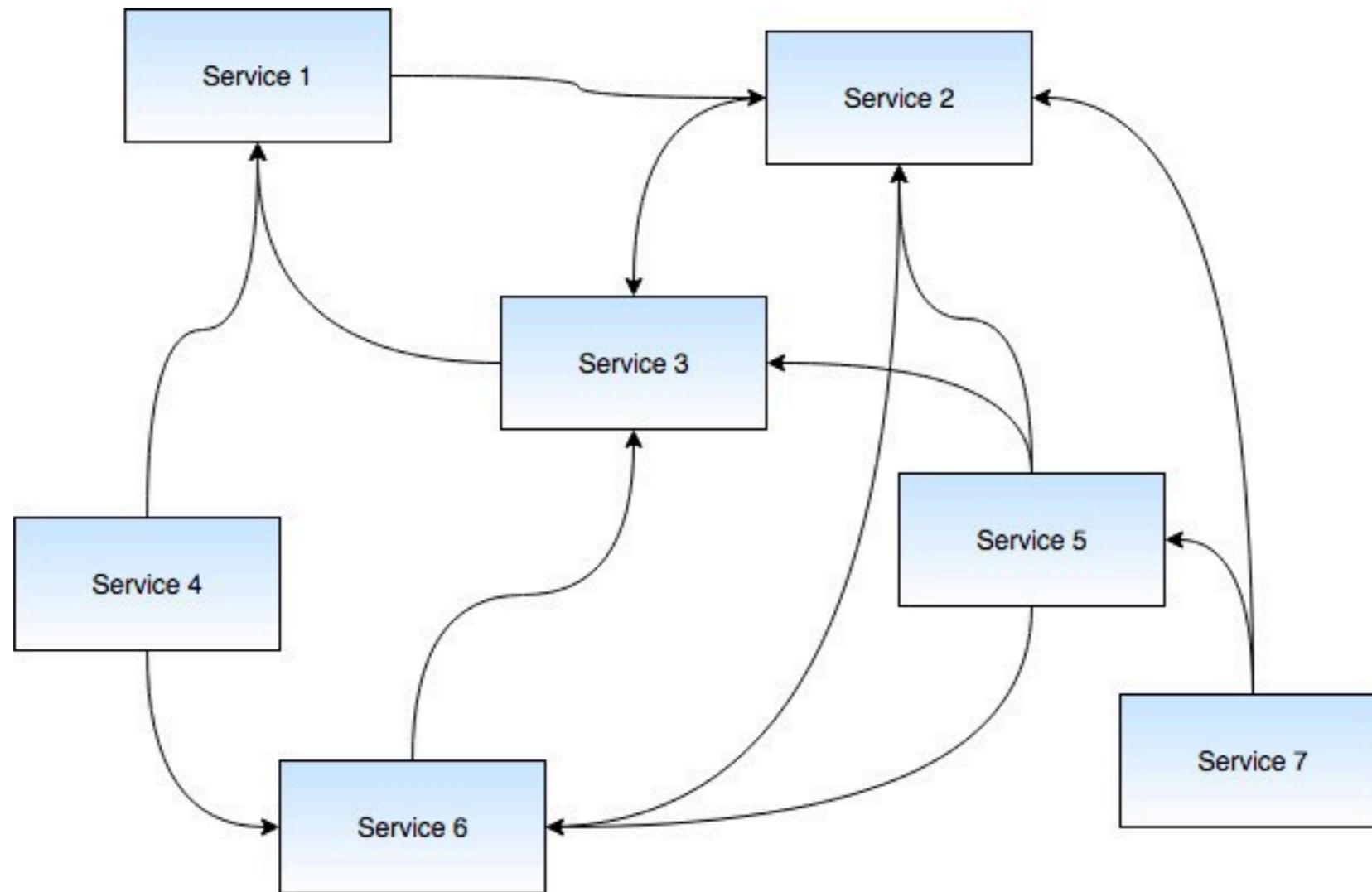
Photos



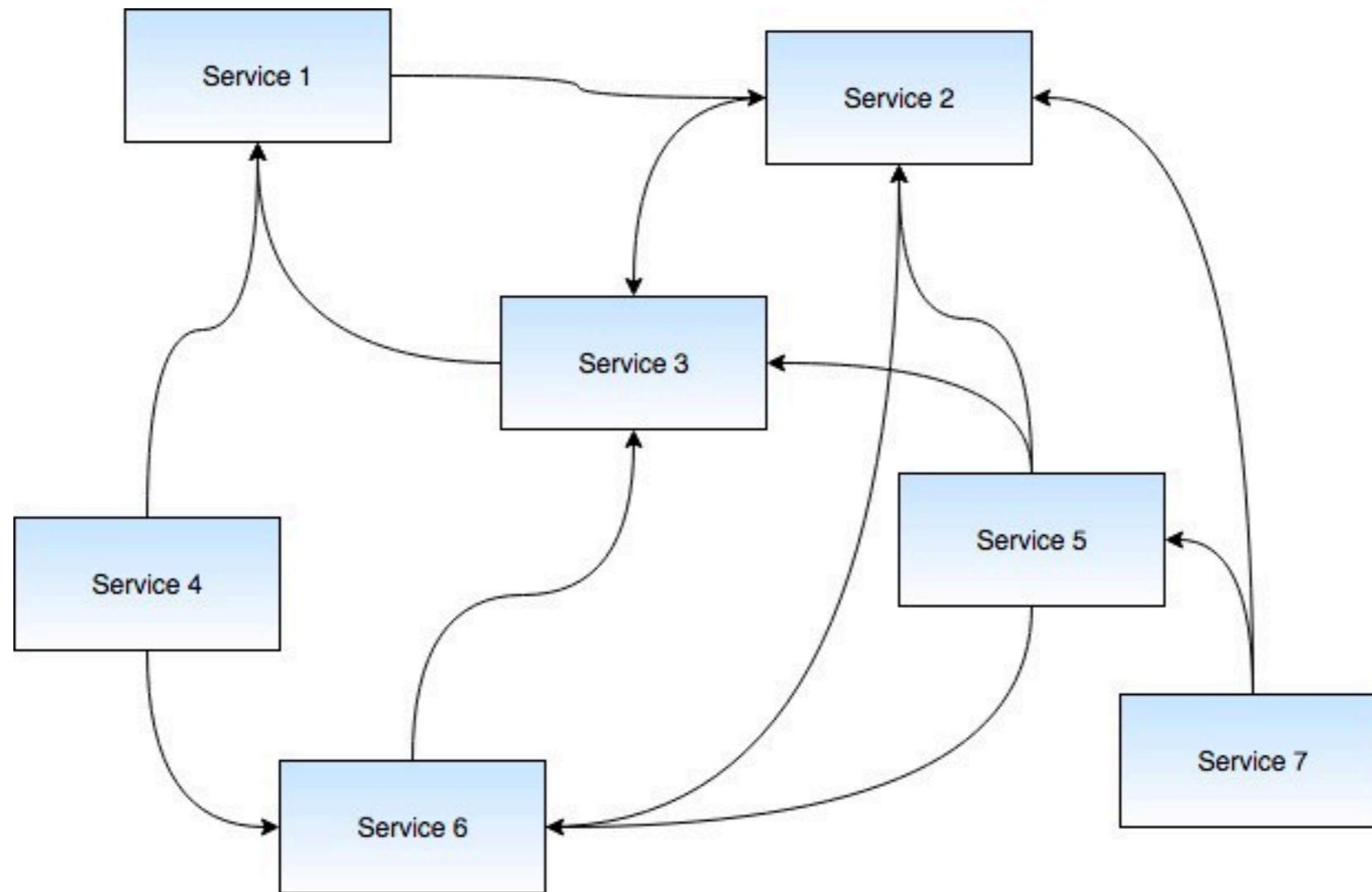
Deploy microservices



Many services !!



How to deploy ?



How to deploy ?

Language-specific packaging format

Deploying a service as a VM

Deploying a service as a Container

Serverless deployment



Deployment strategies

Service per instance

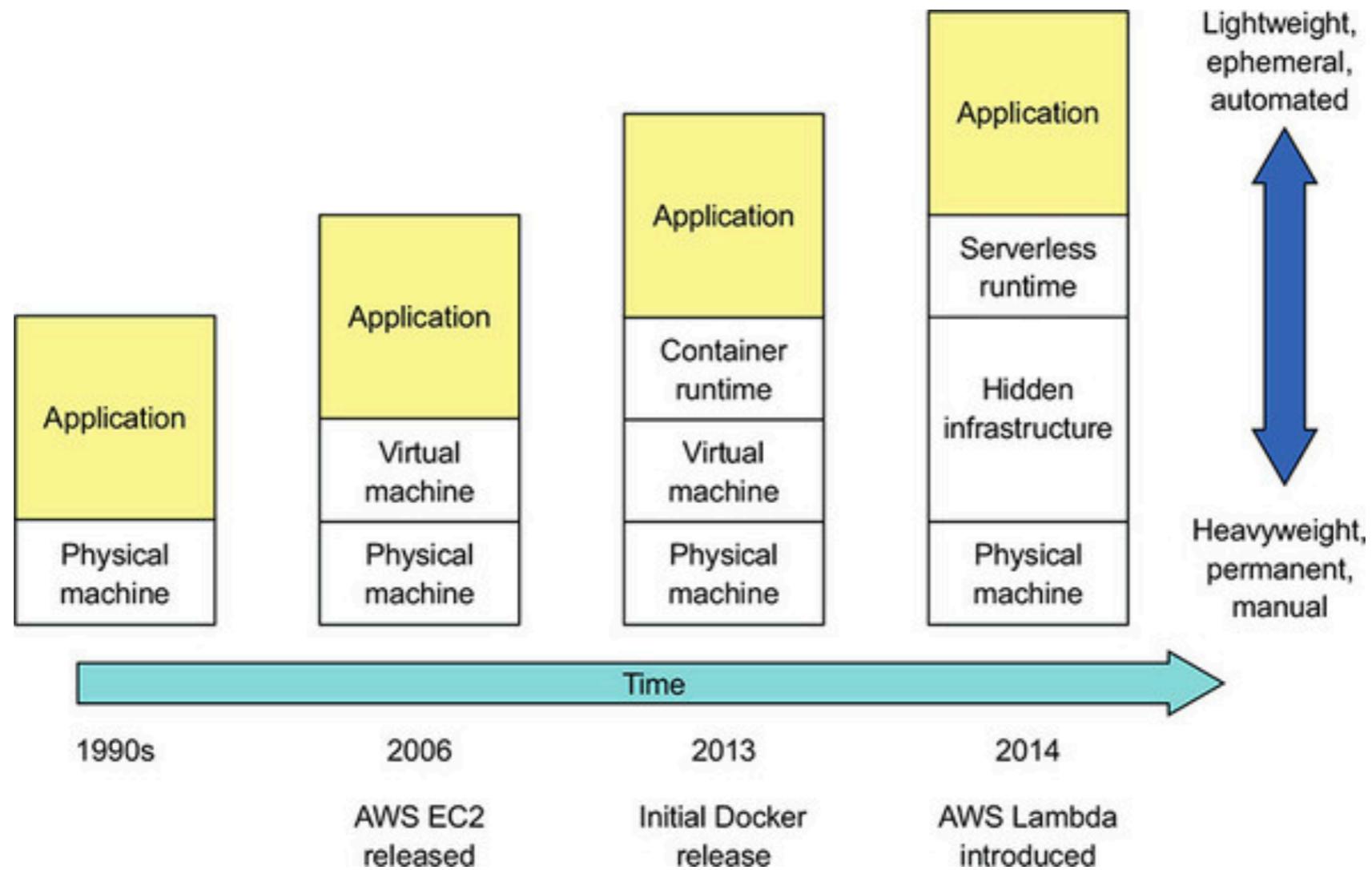
Multiple services per instance

Multiple services in multiple instances



Trend about Deployment !!

Heavyweight to Lightweight
Manual to Automated



1. Language-specific

Create executable files for deploy
Deploy files to production server
All services are process

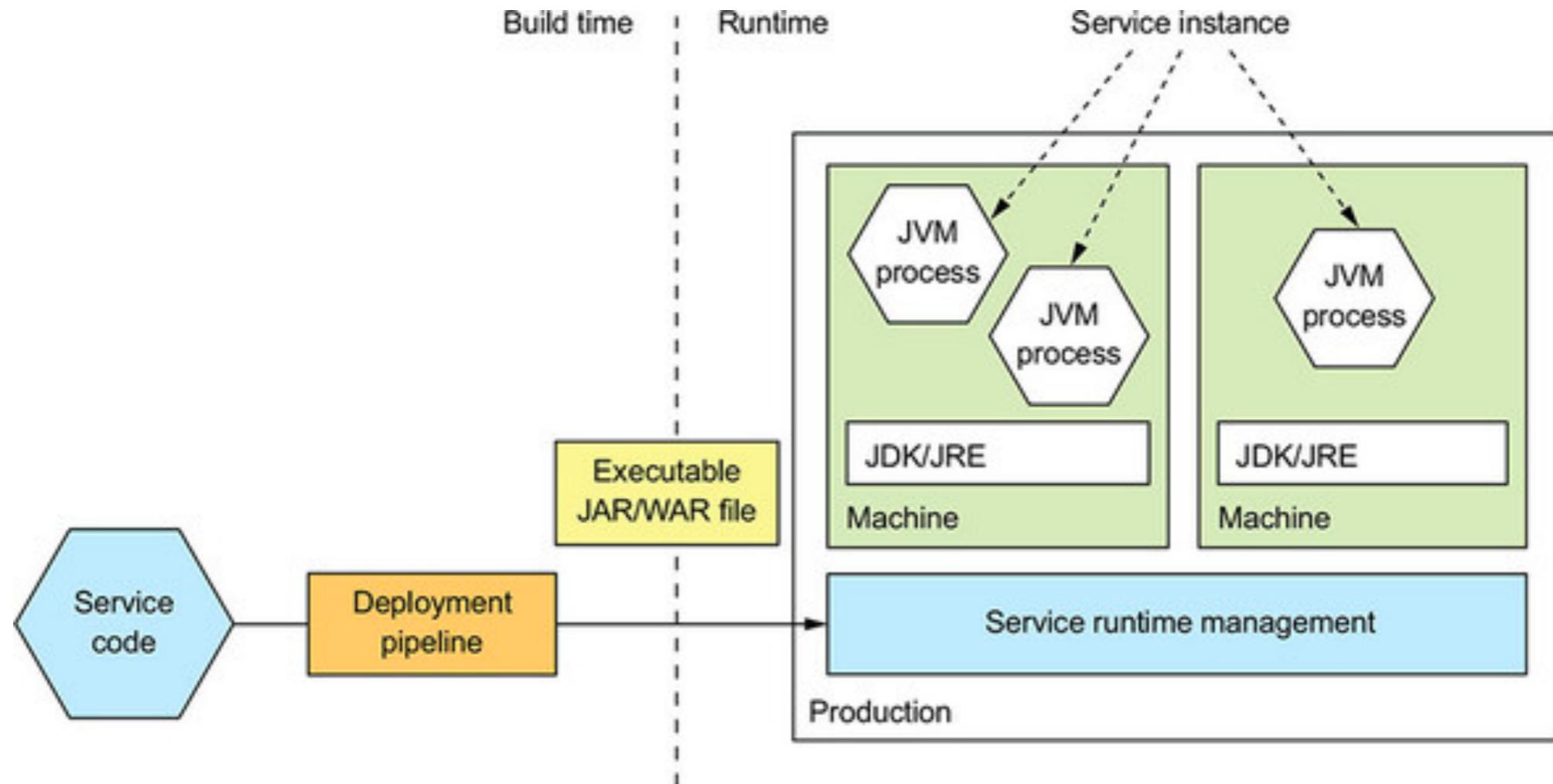


Executable files

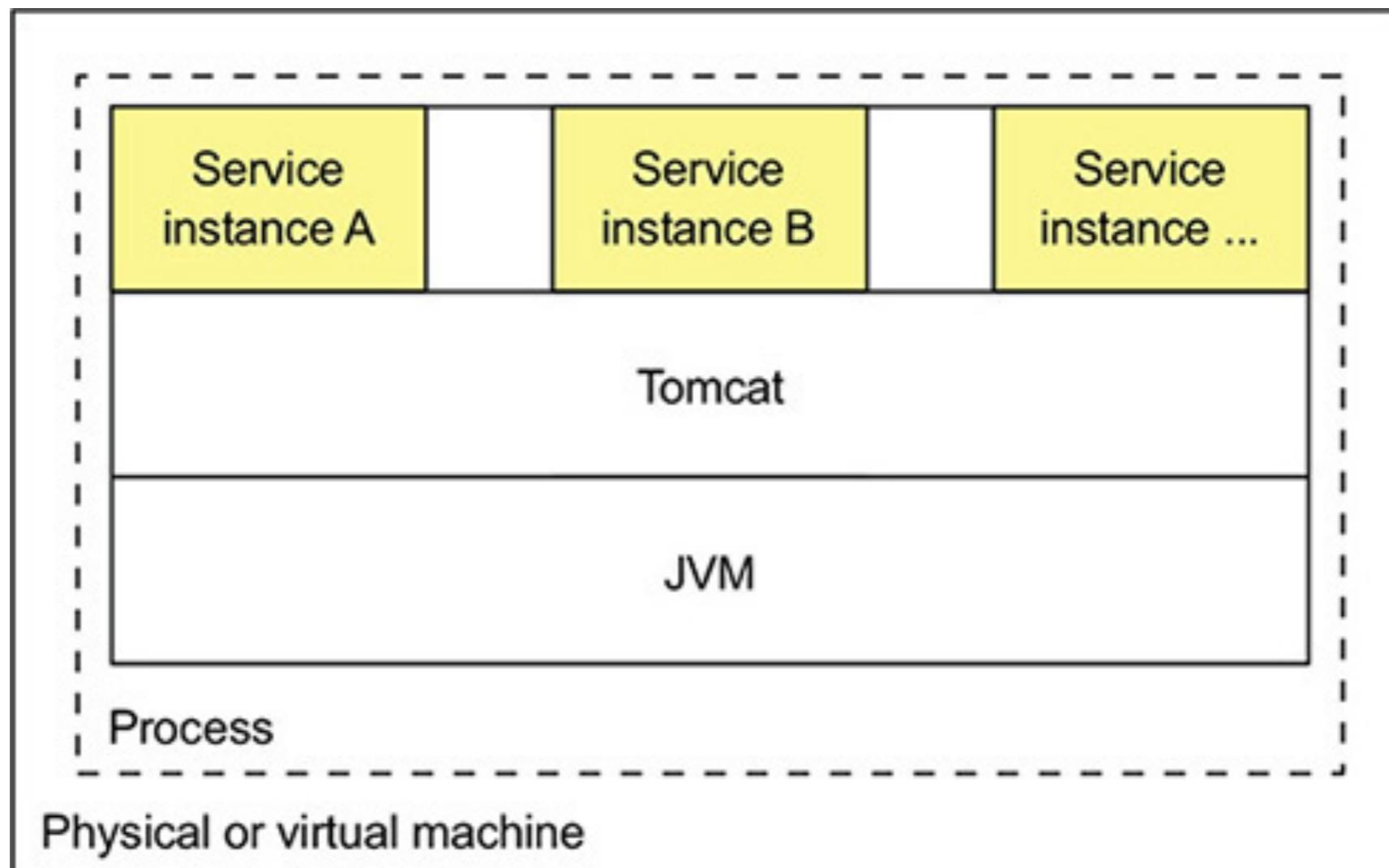
Programming Language	Executable file format
Java	JAR/WAR/EAR
Golang	Binary
NodeJS, JavaScript	Directory of source code



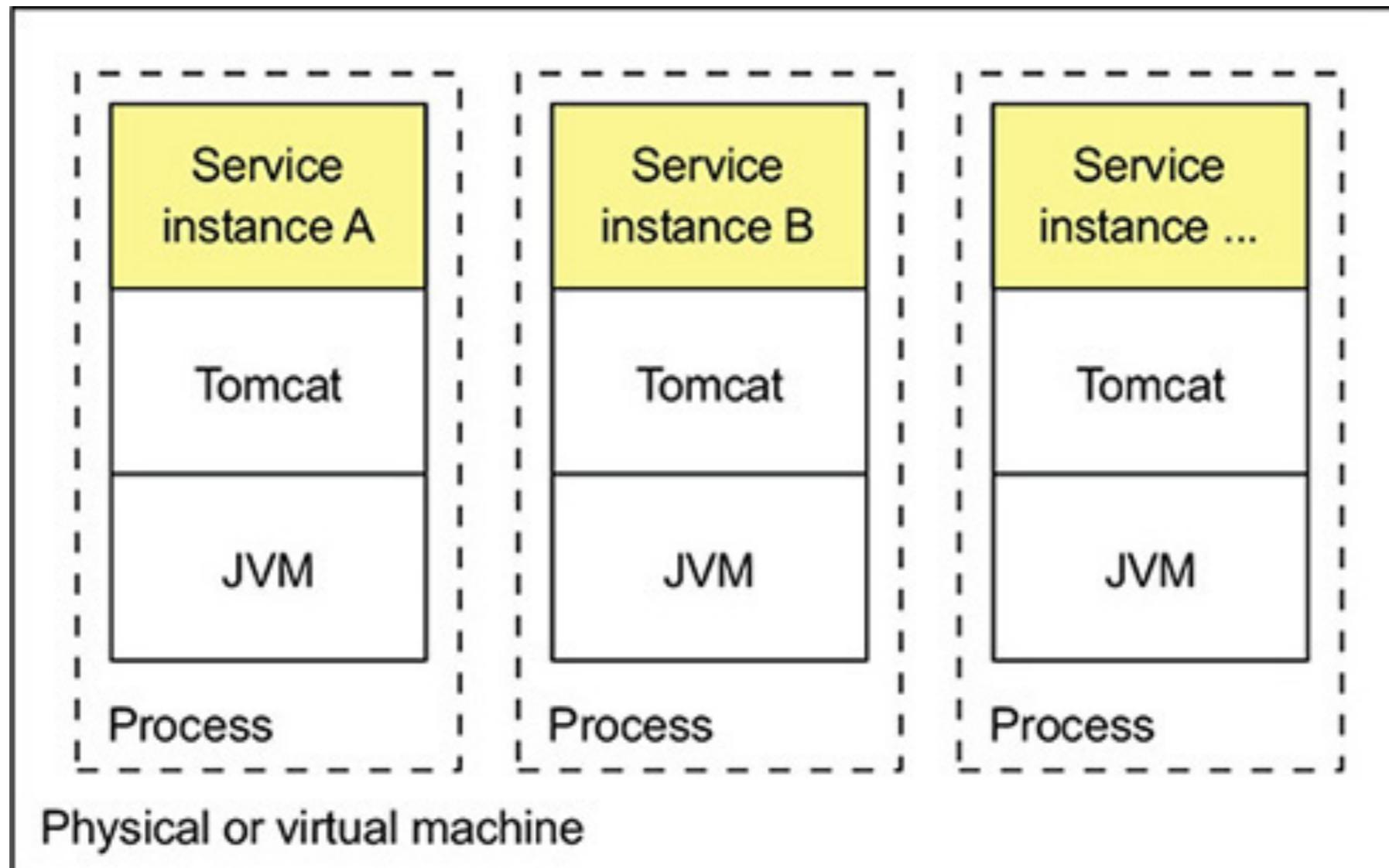
Example :: Deploy Java



Deploy multiple services per web server



Deploy multiple services per machine



Benefits

Fast deployment
Efficient resource utilisation



Drawbacks

Lack of encapsulation tech stack
Can't limit resources per service
Lack of isolation

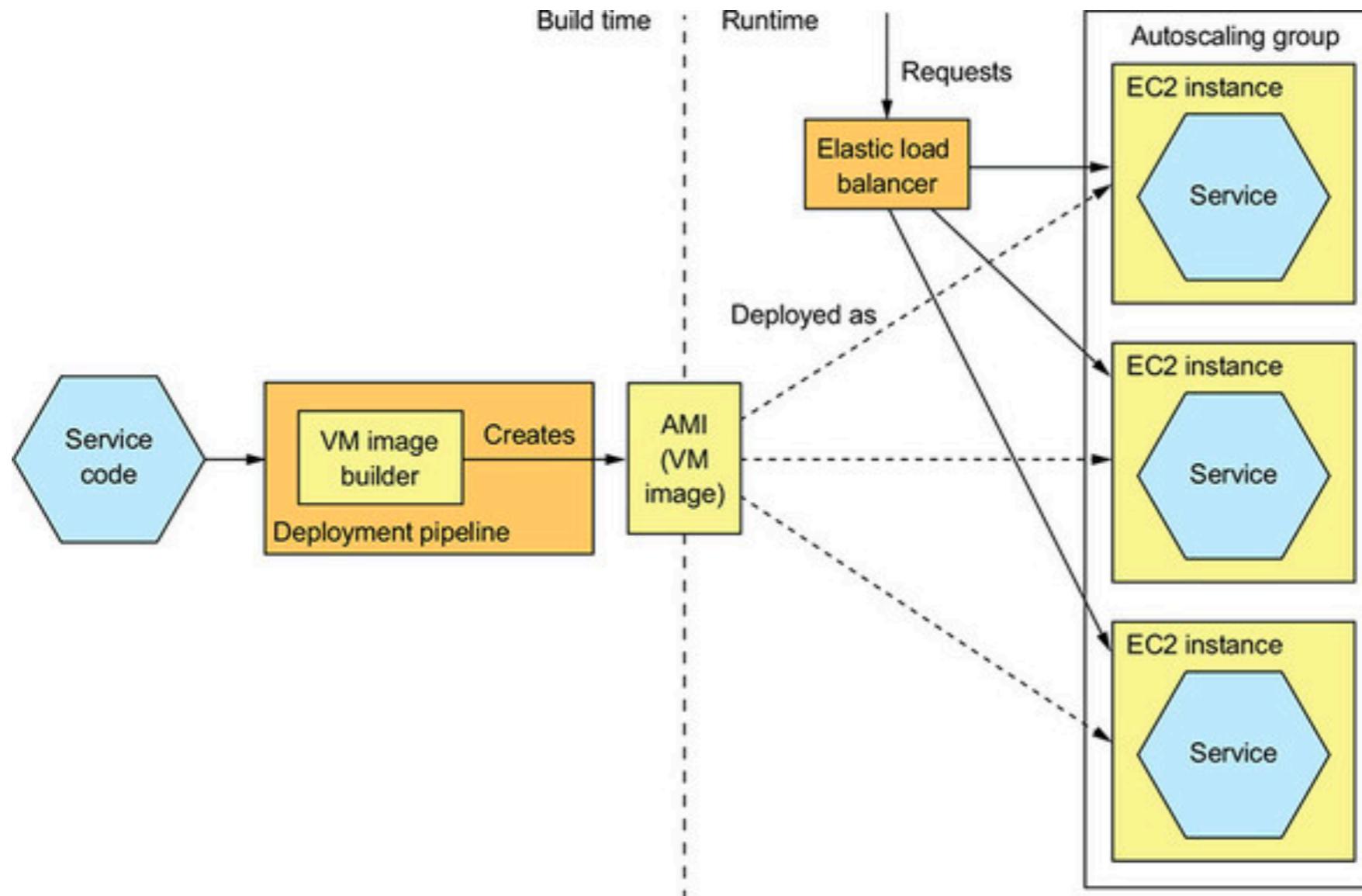


2. Deploy as a Virtual Machine

Create VM on machine
Deploying a service on VM



Example :: Deploy Java



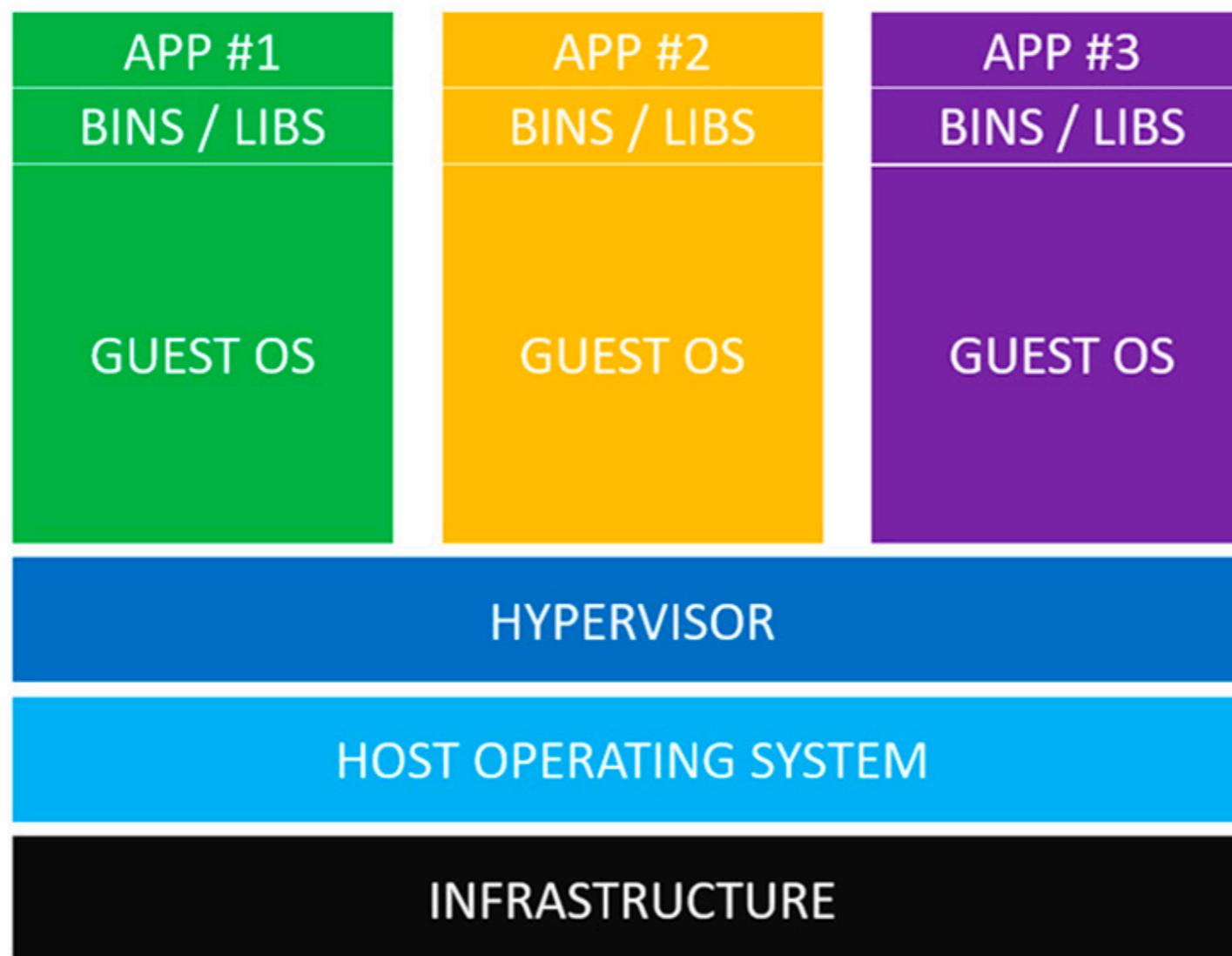
Benefits

Encapsulation tech stack
Isolated service instances
Use mature cloud infrastructure



Drawbacks

Less efficient resources utilisation



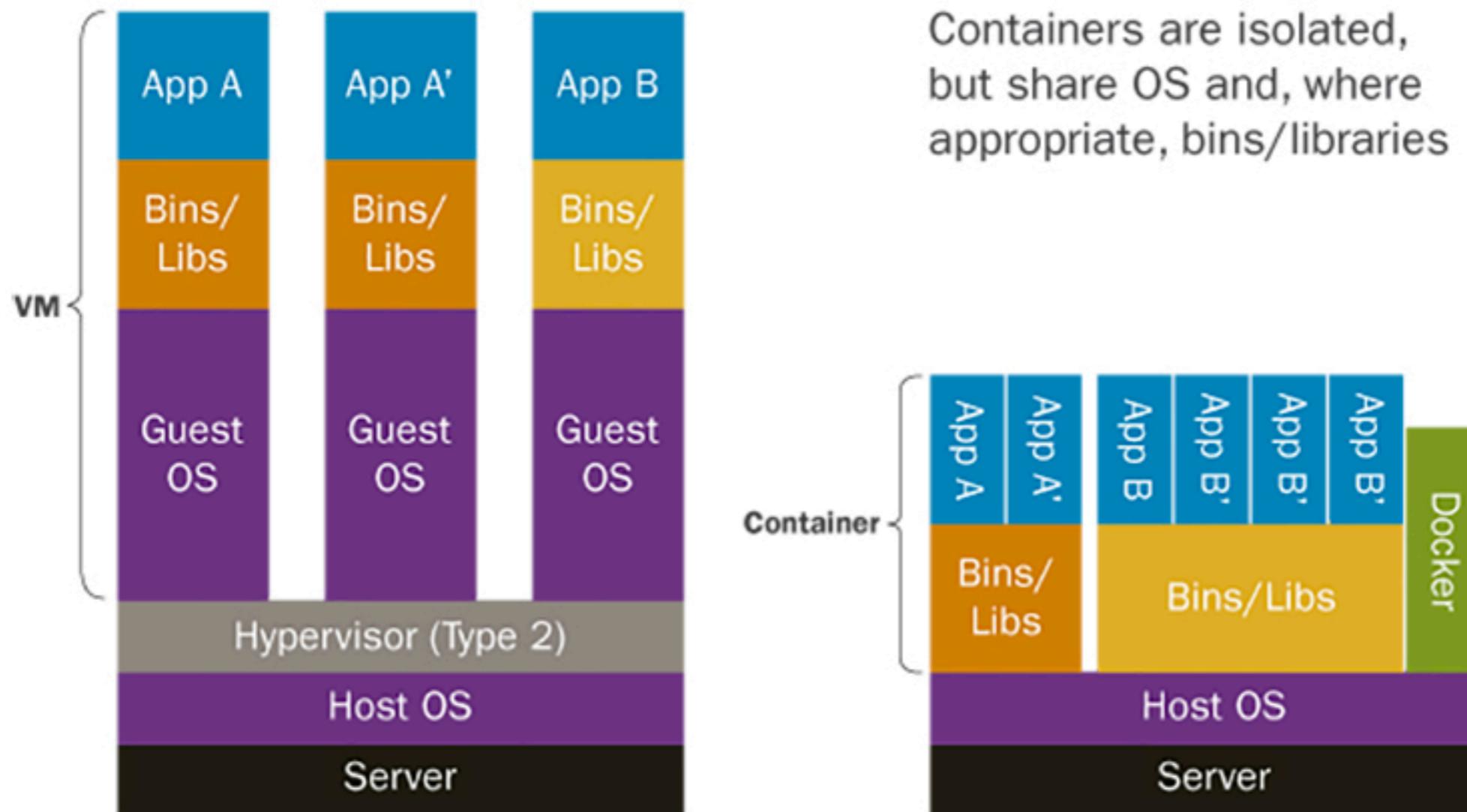
Drawbacks

Less efficient resources utilisation
Slow deployment
System administration overhead



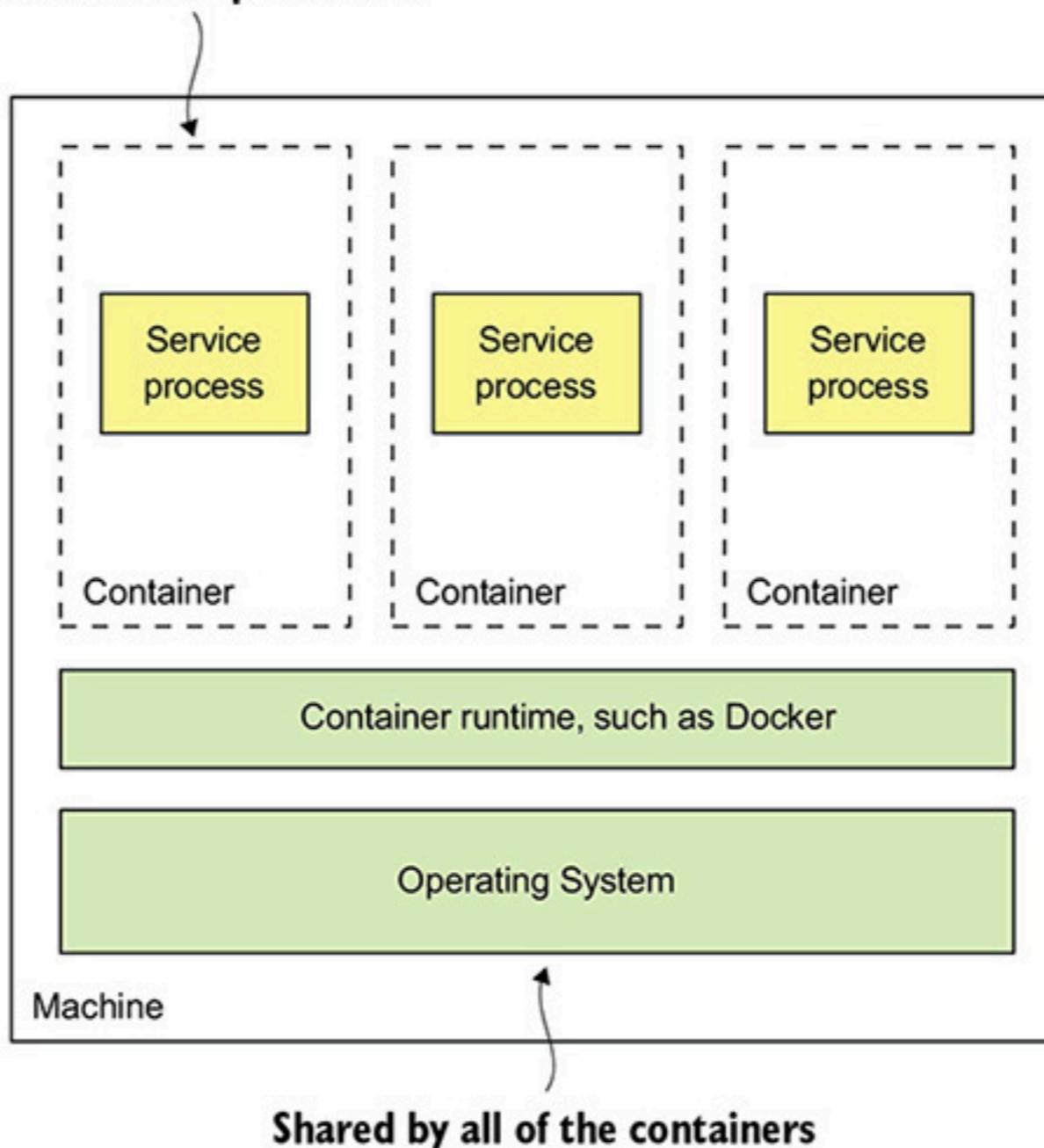
3. Deploy as a Container

Containers vs. VMs

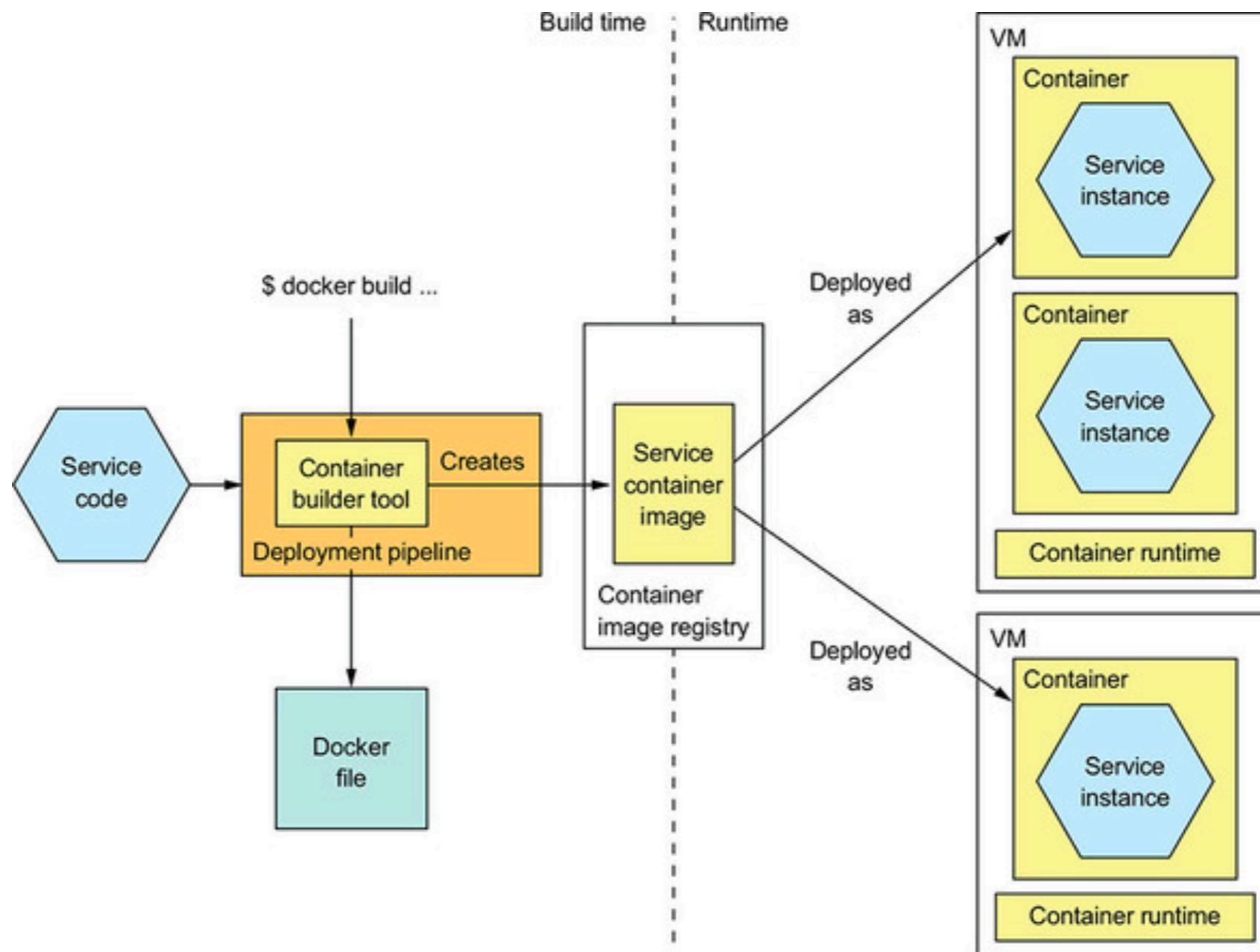


Deploy as a Container

Each container is a sandbox
that isolates the processes.



Example :: Deploy Java



Deploy as a container

Create image

Push image to Container image registry

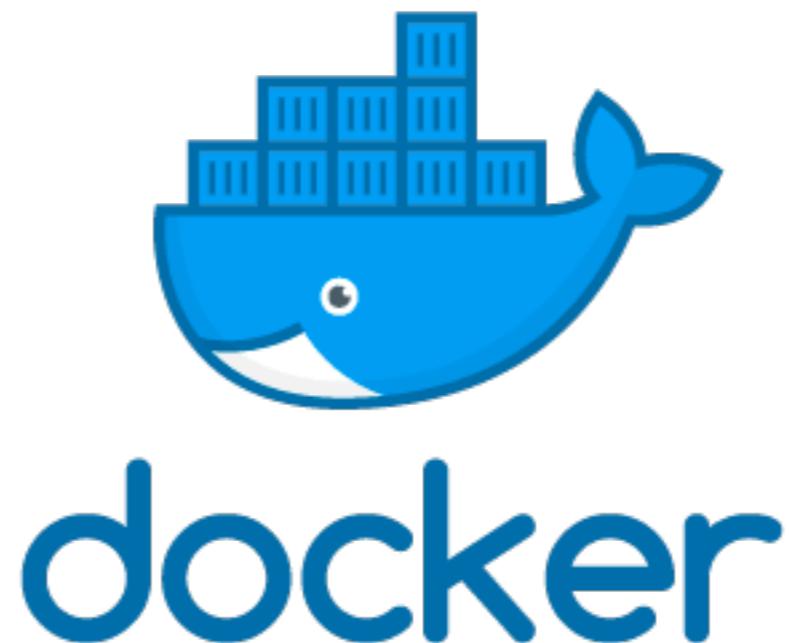
Deploy service as a container



Deploy as a container

Create image

Push image to Container image registry
Deploy service as a container



Benefits

Encapsulation tech stack
Isolated service instances

Service instance's resources are constrained



Drawbacks

System administration overhead
Infrastructure for containers

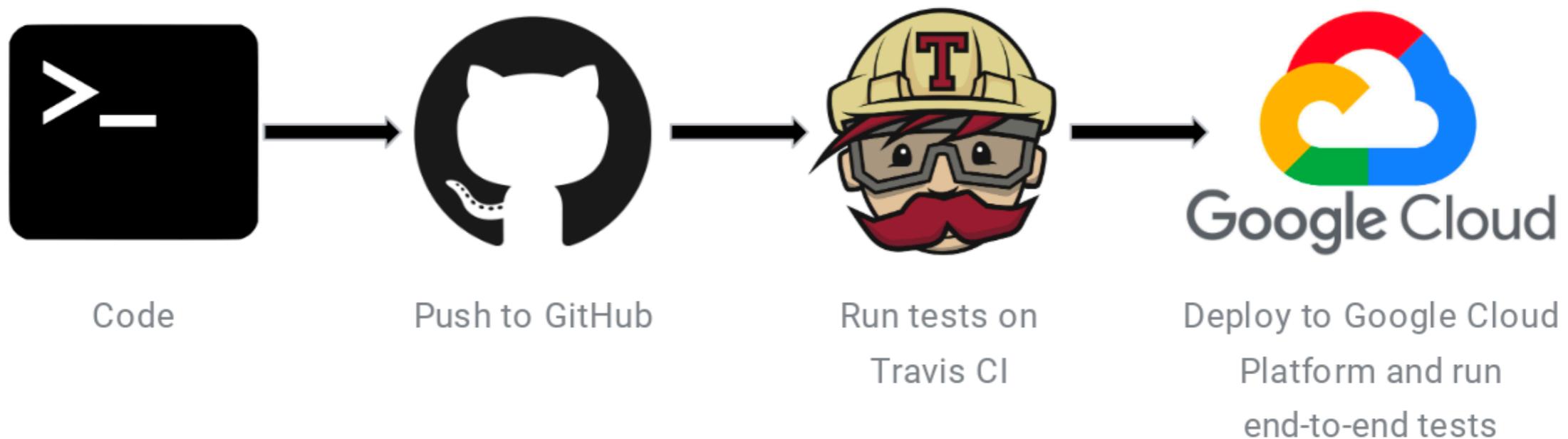


Demo

Deploy to Google Cloud



Deploy to Google Cloud



<https://github.com/up1/workshop-deploy-microservice-java>



Demo project



Steps

Working with Docker
Auto-build with Travis CI
Deploy to Google Cloud



Step 1 :: Working with Docker



Using Play with Docker

<https://labs.play-with-docker.com/>



Using Play with Docker

Add new instance

The screenshot shows the Play with Docker web interface. On the left, there's a sidebar with a timer at 03:59:27, a 'CLOSE SESSION' button, and a 'Instances' section. Below 'Instances' is a '+ ADD NEW INSTANCE' button. A list of existing instances shows one entry: '192.168.0.13 node1'. On the right, a detailed view of a new instance is shown. The instance name is 'bilnljc_bilnlmjc3o4000faub60'. It has an IP of 192.168.0.13, 0.76% memory usage (30.35MiB / 3.906GiB), and 0.57% CPU usage. The SSH link is 'ssh ip172-18-0-13-bilnljc3o4000faub50@direct.labs.play'. Below the instance details are 'DELETE' and 'EDITOR' buttons. The terminal window shows a root shell on the instance:

```
$ #####  
#          WARNING!!!! #  
# This is a sandbox environment. Using personal credentials #  
# is HIGHLY! discouraged. Any consequences of doing so are #  
# completely the user's responsibilites. #  
# #  
# The PWD team.  
#####  
[node1] (local) root@192.168.0.13 ~
```



1. Clone project

```
$git clone https://github.com/up1/workshop-deploy-microservice-java.git
```



2. Build docker image

\$cd catalog

\$docker image build -t catalog_service:1.0 .

\$cd product

\$docker image build -t product_service:1.0 .



3. Push docker image

\$docker login

\$docker image push catalog_service:1.0 .

\$docker image push product_service:1.0 .



3. Push docker image

The screenshot shows the Docker Hub interface. At the top, there is a search bar with the placeholder "Search for great content (e.g., mysql)". Below the search bar, there are navigation links for "Explore", "Repositories", and "Organizations". On the left, there is a dropdown menu showing the user's name "somkiat" and a search bar for "Search by repository name...". To the right of these is a blue button labeled "Create Repository +". The main content area displays two repository cards:

- somkiat / product_service**
Updated 7 minutes ago
0 stars, 2 downloads, PUBLIC
- somkiat / catalog_service**
Updated 7 minutes ago
0 stars, 2 downloads, PUBLIC

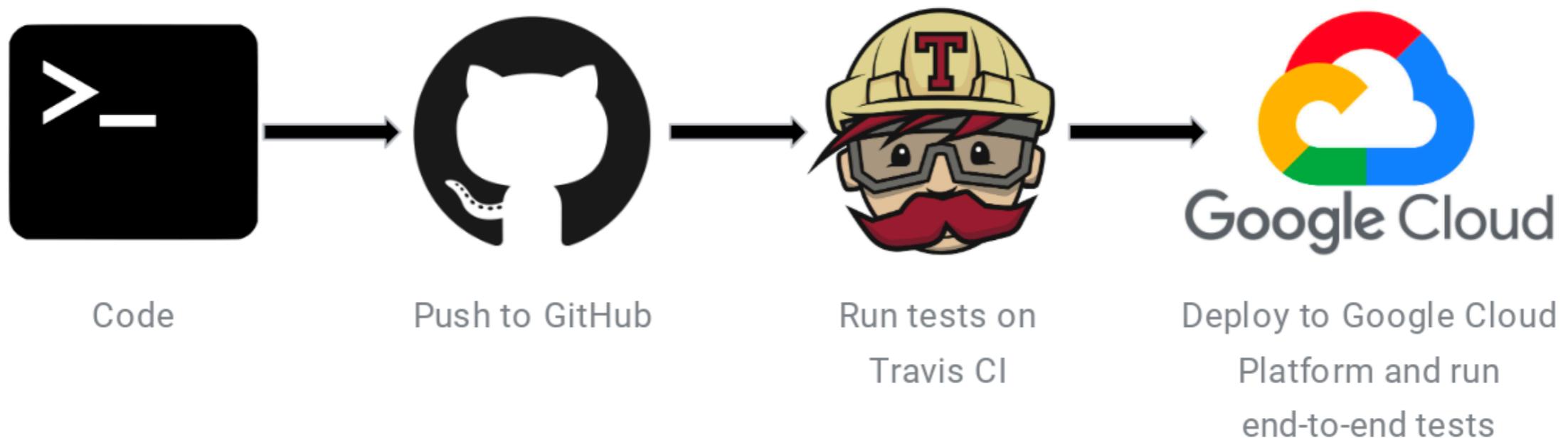


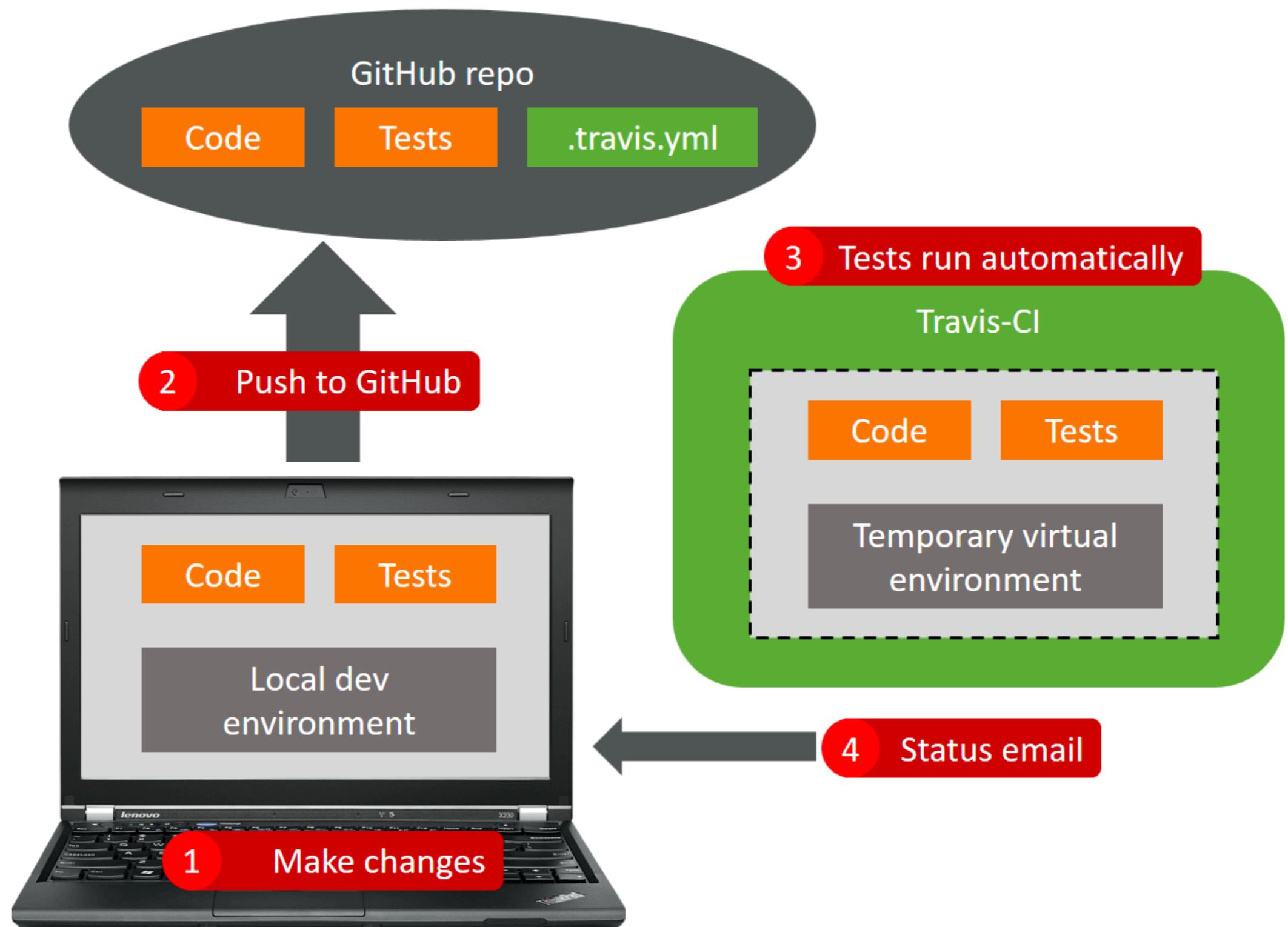
4. Deploy/create container

```
$docker-compose up -d  
$docker-compose ps
```



Step 2 :: Build with Travis CI





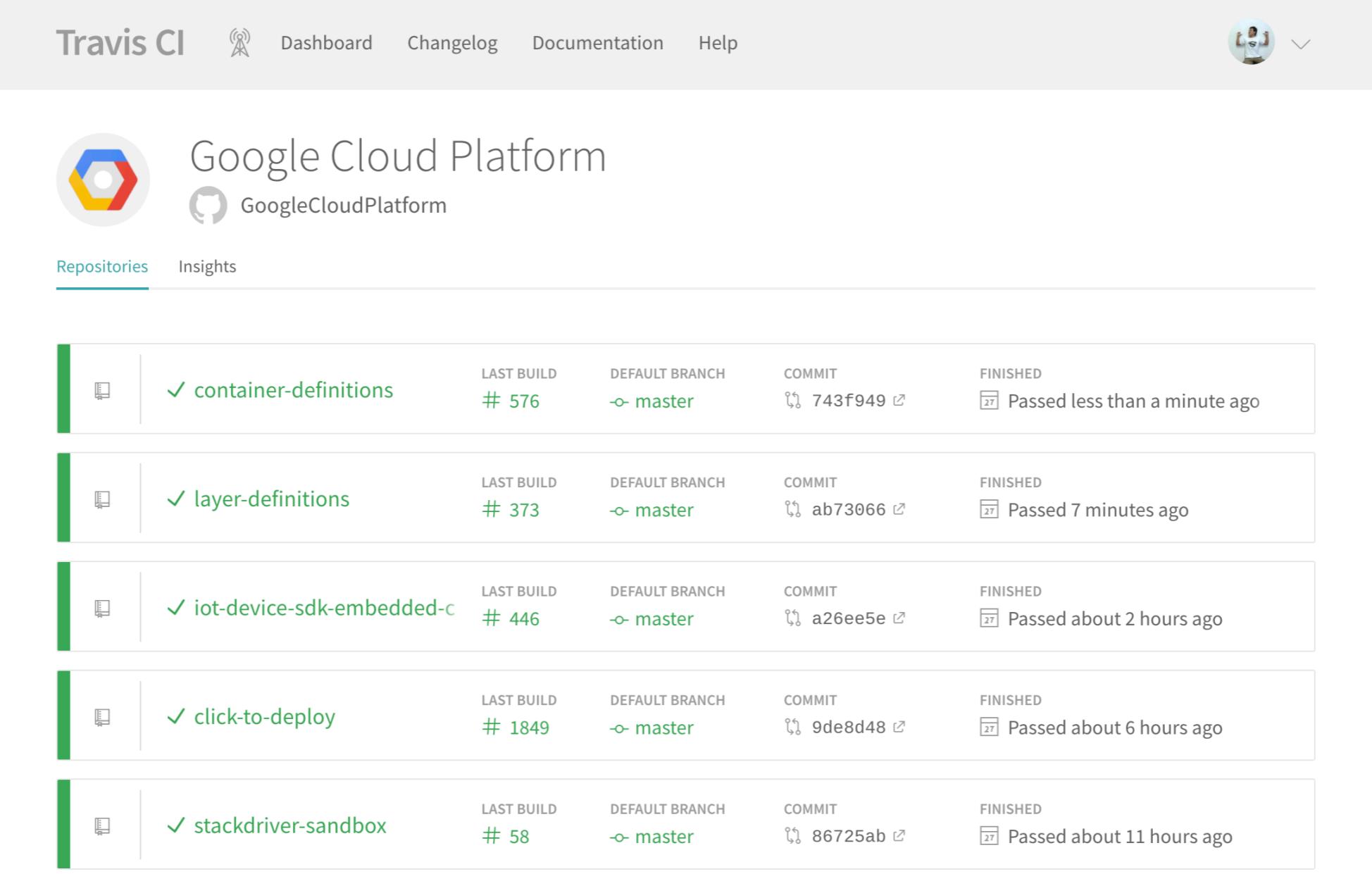
Create file .travis.yml in project

```
language: java
jdk: oraclejdk8
services:
  - docker
```



Setting GitHub + travis ci

<https://travis-ci.com/GoogleCloudPlatform>



The screenshot shows the Travis CI dashboard for the Google Cloud Platform repository. The top navigation bar includes links for Dashboard, Changelog, Documentation, Help, and a user profile icon. The main header displays the repository name "Google Cloud Platform" and owner "GoogleCloudPlatform". Below the header, there are two tabs: "Repositories" (selected) and "Insights". The main content area lists five repositories with their build status, last build number, default branch, commit hash, and finish time:

Repository	Last Build	Default Branch	Commit	Finished
container-definitions	# 576	→ master	743f949	Passed less than a minute ago
layer-definitions	# 373	→ master	ab73066	Passed 7 minutes ago
iot-device-sdk-embedded-c	# 446	→ master	a26ee5e	Passed about 2 hours ago
click-to-deploy	# 1849	→ master	9de8d48	Passed about 6 hours ago
stackdriver-sandbox	# 58	→ master	86725ab	Passed about 11 hours ago



Setting GitHub + travis ci

<https://travis-ci.com/GoogleCloudPlatform>

The screenshot shows the Travis CI dashboard for the user 'Somkiat Puisungnoen' (@up1). The dashboard includes sections for 'MY ACCOUNT' (Sync account), 'ORGANIZATIONS' (ITForge, it-kmitl-2018), and 'MISSING AN ORGANIZATION?' (Review and add your authorized organizations). A prominent feature is the 'GitHub Apps Integration' section, which encourages activating the integration to start testing and deploying on Travis CI. It mentions that the integration supports both private and open source repositories and provides enhanced security. A green 'Activate' button is visible. The top navigation bar includes links for Dashboard, Changelog, Documentation, Help, and a user profile icon.

Travis CI

Dashboard Changelog Documentation Help

MY ACCOUNT

Somkiat Puisungnoen

Sync account

ORGANIZATIONS

ITForge

it-kmitl-2018

MISSING AN ORGANIZATION?
Review and add your authorized organizations.

Somkiat Puisungnoen
@up1

Repositories Settings Subscription

GitHub Apps Integration

Activate the GitHub Apps integration to start testing and deploying on Travis CI.

The GitHub Apps integration supports both private and open source repositories, while providing enhanced security when interacting with GitHub.

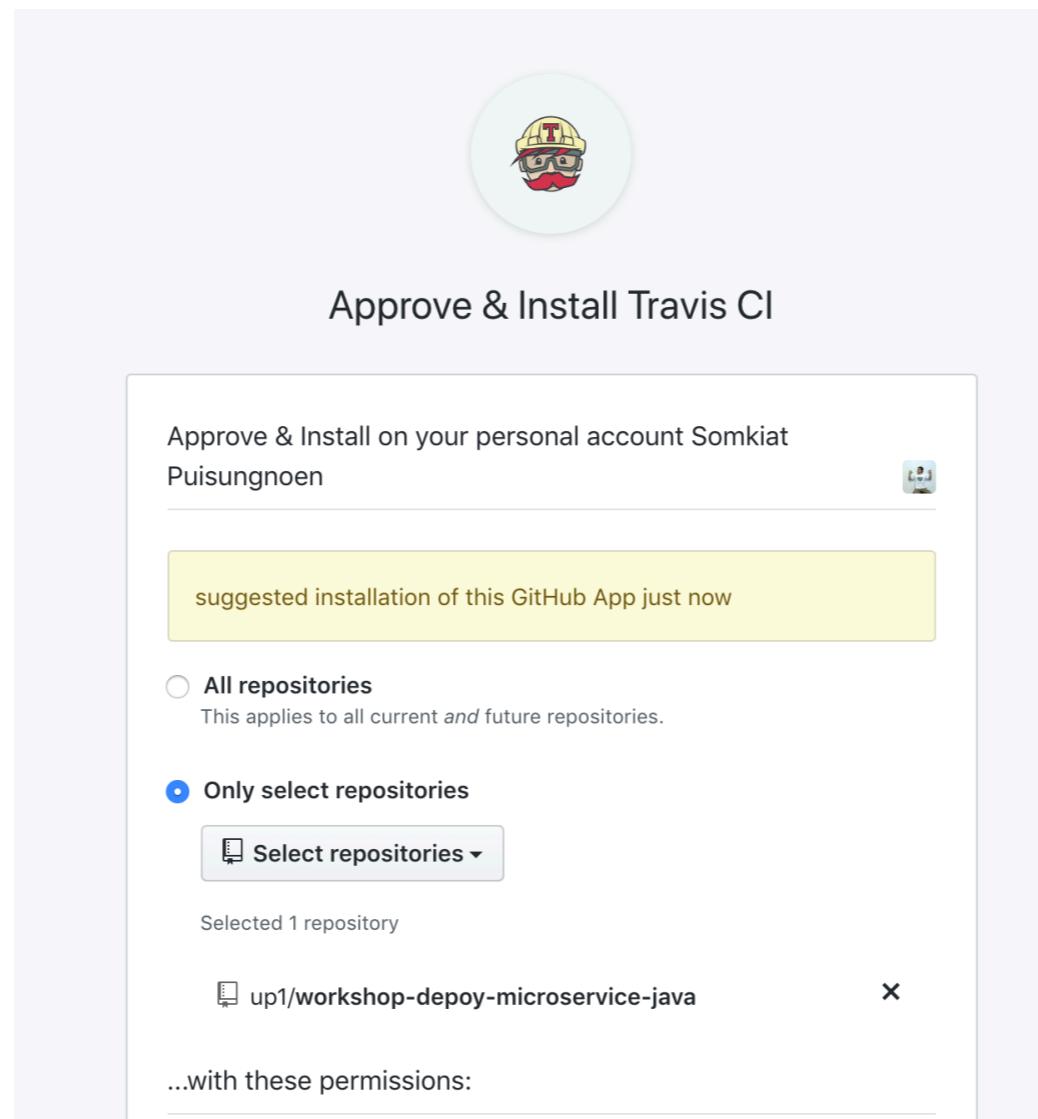
Activate

We are only able to migrate accounts that have 50 or fewer repositories using the Legacy Services Integration. Please refer to our documentation on how to migrate your account.



Setting GitHub + travis ci

<https://travis-ci.com/GoogleCloudPlatform>



Result

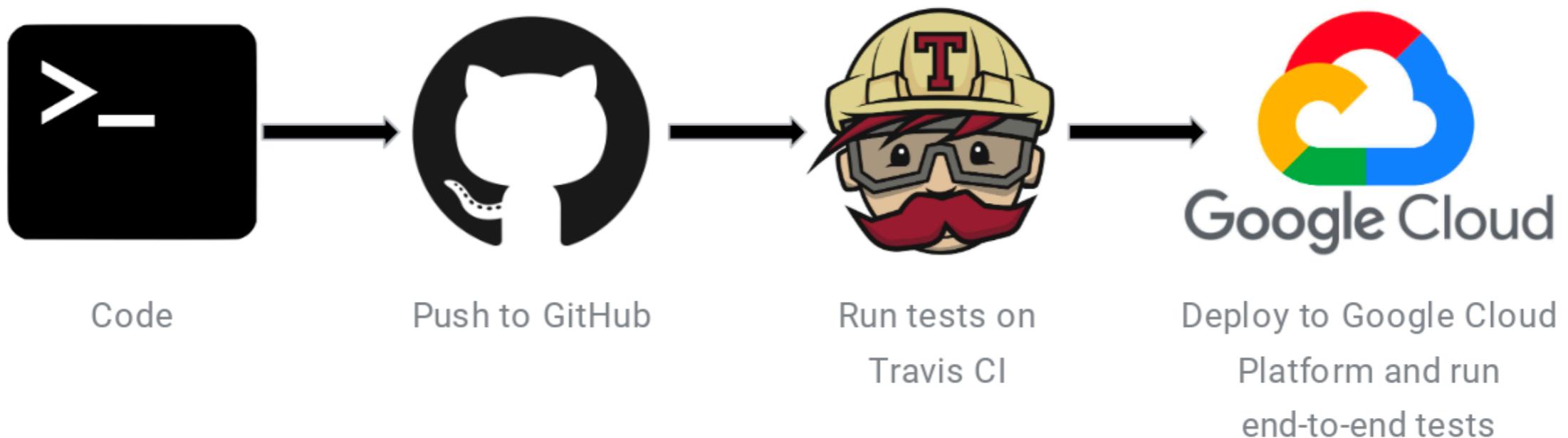
<https://travis-ci.com/up1/workshop-deploy-microservice-java>

The screenshot shows the Travis CI dashboard for the repository `up1 / workshop-deploy-microservice-java`. The build status is **build passing**. The build log shows the following worker information:

```
1 Worker information
6 PERLLIB
7   PERLIO_DEBUG
8   JAVA_TOOL_OPTIONS
9   SHELLOPTS
10  GLOBIGNORE
11  PS4
12  BASH_ENV
```



Step 3 :: Deploy to Google Cloud Engine



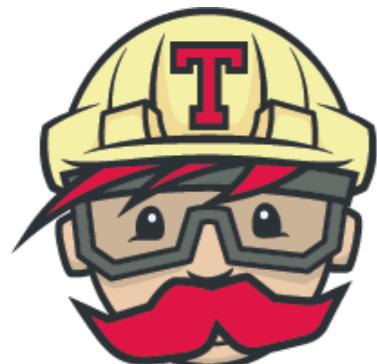
Homework

Run test

Run test/code coverage

Push docker image to Container image registry

Deploy to Google Cloud



Travis CI



circleci

