MIMACOM

JAVA FORUM STUTTGART 2021

1000 Ways to Run Java in the Cloud

Talk by Enrique Llerena Domínguez & Fabian Keller

About Us

Enrique Llerena Domínguez

Senior Software Engineer @mimacom
Machine Learning Enthusiast @127.0.0.1

@ellerenad
www.ienjoysoftware.dev

Fabian Keller

Head of Cloud Innovation @mimacom
Woodworker @127.0.0.1

@_fabiankeller
www.fabian-keller.de


MIMACOM

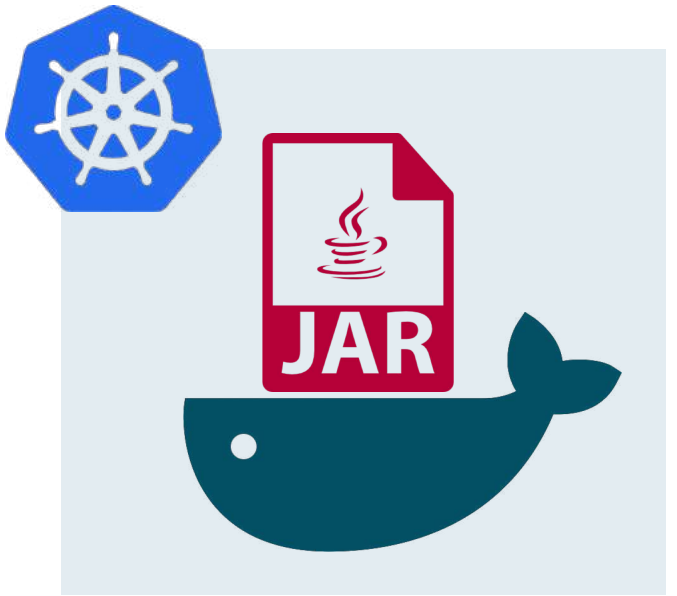
Image source: Photo by dylan nolte on Unsplash

We need to run our Java application in the cloud.

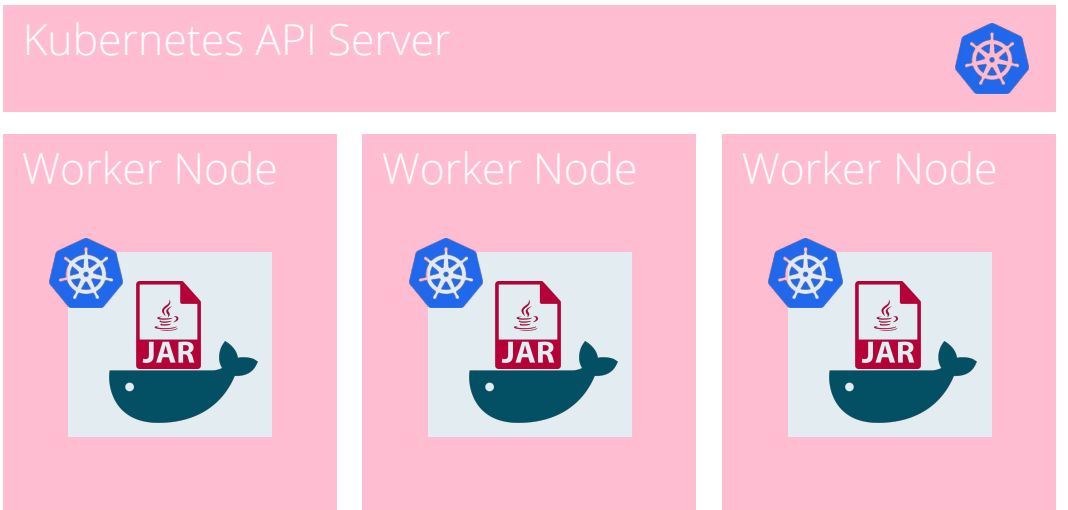
Build the app

Containerize

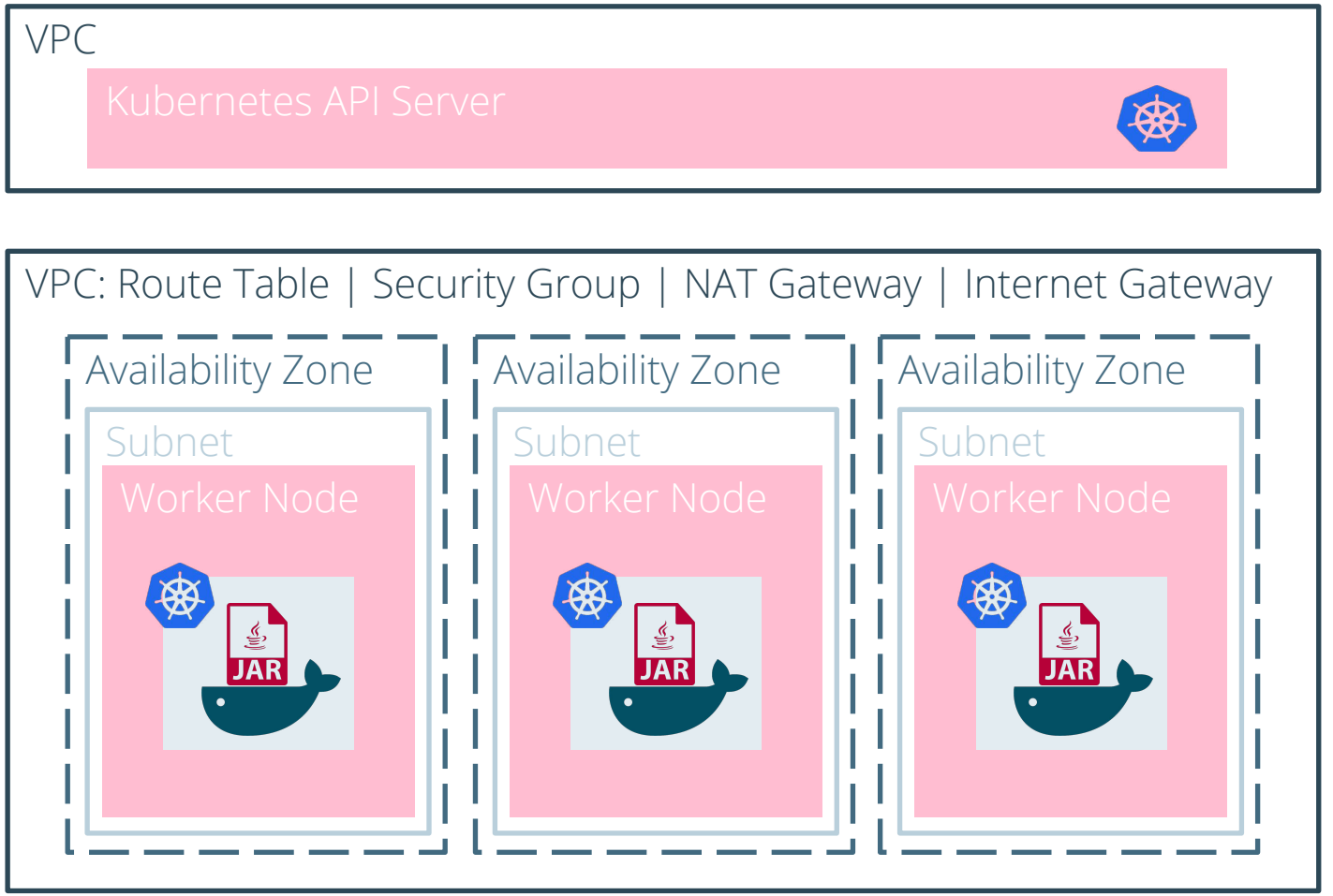
Make a Pod



Deploy the Cluster



Add a Network



Add Ingress Routing

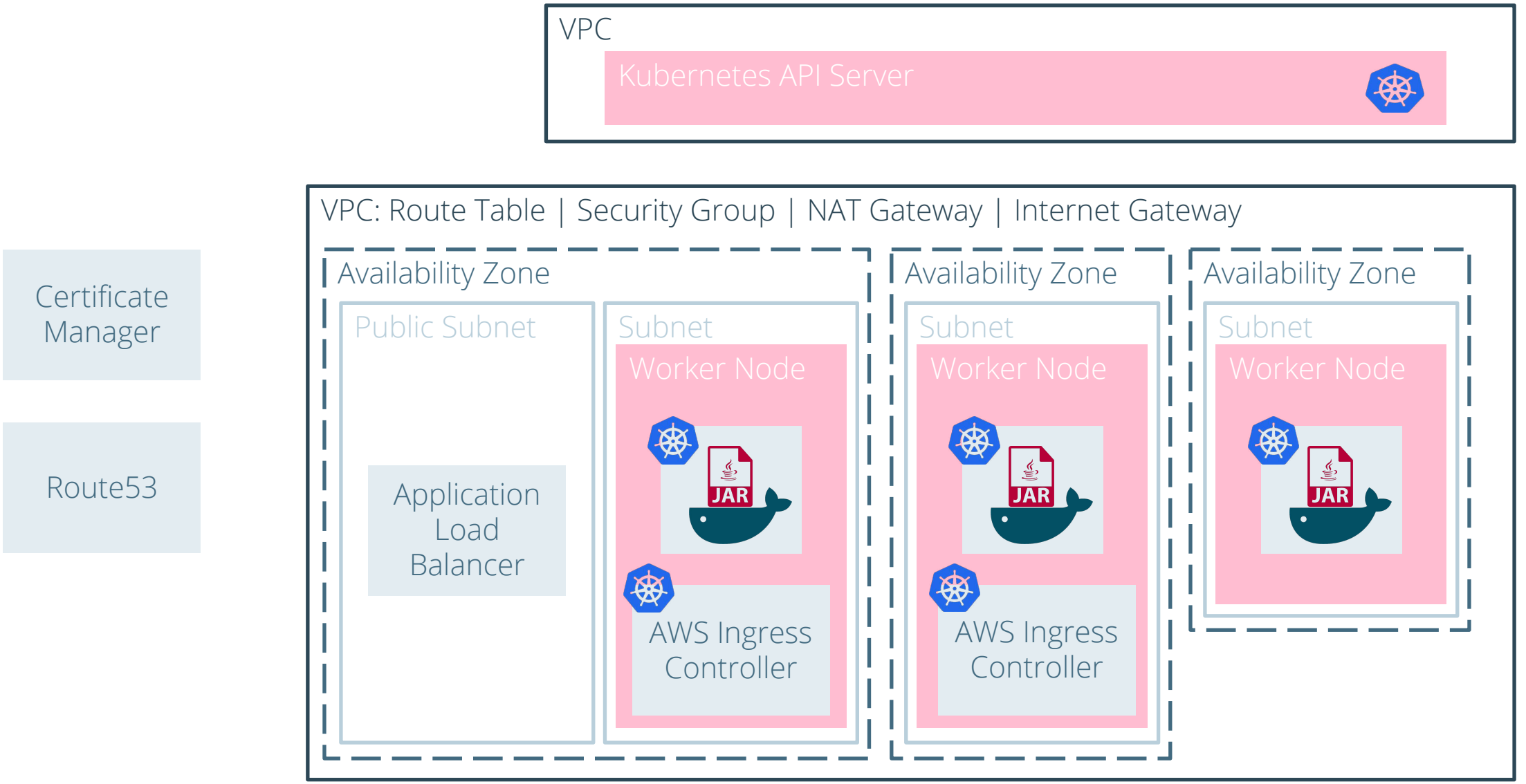


Photo by Ryan Snaadt on Unsplash

Can your application answer 1 request in less than a second?

1	request per second
60	requests per minute
3600	requests per hour
86.400	requests per day

That’s easy! Less than 80k requests per day
→ single instance deployment ☺

Goals

How do we actually run Java in the cloud nowadays?

- Different ways that are already productive – or in its way
- The challenges
- Reduce responsibilities in product teams to gain speed

Possible combinations

Base Software	Runtimes	Packaging	Compute	Deployment
Ubuntu	GraalVM	Buildpacks.io	Public Cloud VMs	Blue Green
Alpine	OpenJDK	Dockerfile	Managed container orchestrator	Canary
…	AWS Corretto	Docker Multi Stage	BYO K8s	Recreate
	Oracle JDK	Hashicorp Packer	Heroku / Cloud Foundry	Rolling
		Podman	AWS Beanstalk	
		…	Public Cloud Serverless	
			…	

Combinations

At least 1120 ☺

It is not just a catchy title 

Responsibilities

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Business Value Impact

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Way 1 | Lightweight

Photo by Trac Vu on Unsplash

Way 1 | Lightweight

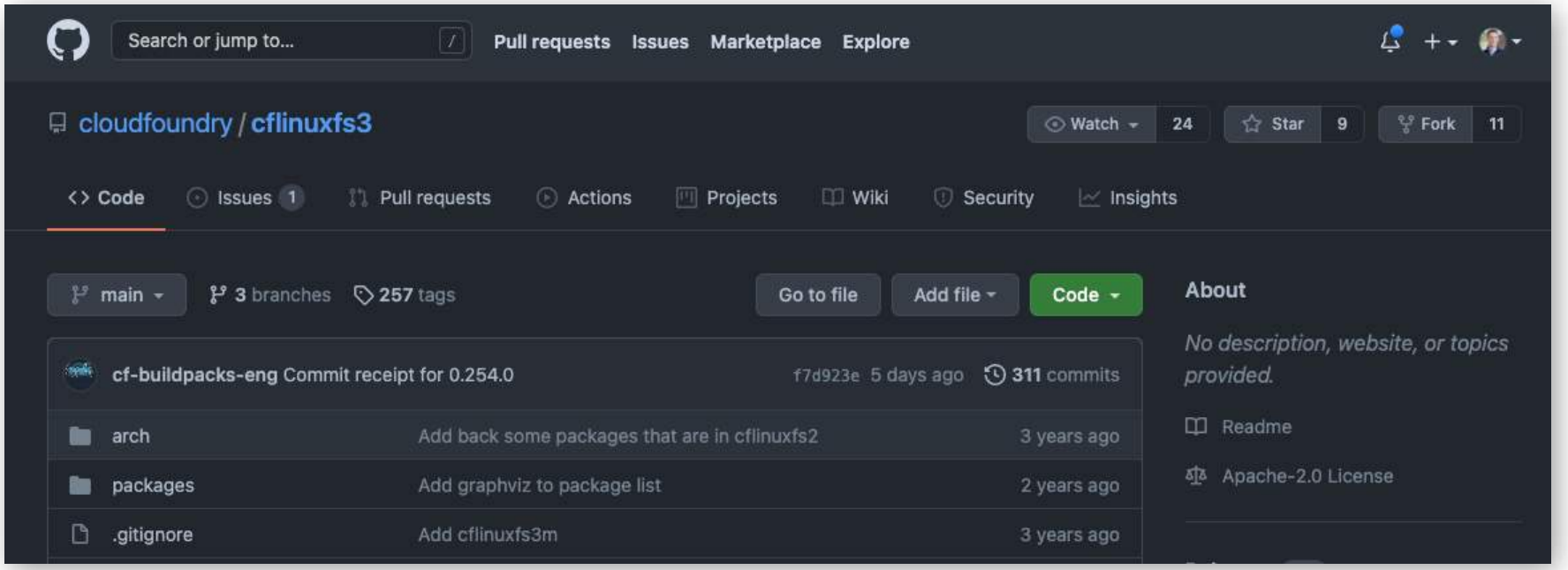
BASE SOFTWARE	RUNTIMES	PACKAGING	COMPUTE	DEPLOYMENT
Ubuntu (cloudfoundry/cflinuxfs3)	OpenJDK	Cloud Foundry Buildpack	Cloud Foundry	Blue-green

CLOUD FOUNDRY

- Been part of the platform team for 3 customers
- Seen this stack for >100 teams running thousands of applications, with ~40% built with Java.

Ubuntu
(cloudfoundry/cflinuxfs3)

- Hardened image used by Cloud Foundry by default
- Open-source: https://github.com/cloudfoundry/cflinuxfs3



RUNTIMES

OpenJDK

PACKAGING

Cloud Foundry Buildpack

Datadog Buildpack

Java Buildpack

Cloud Foundry Container Image

- Datadog Agent
- JRE
- App
- Base Image: cflinuxfs3

Buildpacks are essentially 4 scripts:

`bin/detect` whether or not to apply the buildpack to an app.
`bin/supply` provides dependencies for an app.
`bin/finalize` prepares the app for launch.
`bin/release` indicates how the app should be executed.

Cloud Native Buildpacks

Jar to Container Image With a Single Command

Buildpacks.io

	Cloud Native Buildpacks	Dockerfile	source-to-image (s2i)	Jib	ko
Advanced Caching	Yes	No	Yes	No	No
Auto-detection	Yes	No	Yes	Yes	Yes
Bill-of-Materials	Yes	No	No	No	No
Modular / Pluggable	Yes	No	No	N/A †	N/A †
Multi-language	Yes	Yes	Yes	No	No
Multi-process	Yes	No	No	No	No
Minimal app image	Yes	Yes ♦	Yes ‡	Yes	Yes
Rebasing	Yes	No	No	No	No
Reproducibility	Yes	No	No	Yes	Yes
Reusability	Yes	No	Yes	N/A †	N/A †
	○ Azure ○ CircleCI	○ Amazon ECS	○ OpenShift	○ Gradle ○ Maven	

```
# build container
pack build sample-app \
  --path samples/apps/java-maven \
  --builder cnbs/sample-builder:bionic

# and run it
docker run --rm -p 8080:8080 sample-app
```

COMPUTE

Cloud Foundry

DEPLOYMENT

Blue-green

- Highly opinionated PaaS
- Huge infrastructure footprint
- But scales to hundreds of teams

- All it takes to production is a:
  `cf push -f manifest.yaml`

```yaml
# manifest.yaml
---
applications:
- name: machine-pricing
  instances: 3
  routes:
  - route: machine-pricing.prod.machin.es
  buildpacks:
    - java_buildpack
    - datadog_buildpack
  path: /machine-pricing/target/mp-2.5.3.jar
  memory: 4GB
  services:
    - pricing-db
- name: spare-parts-pricing
  instances: 2
  routes:
  - route: spare-parts-pricing.prod.machin.es
  buildpacks:
    - java_buildpack
    - datadog_buildpack
  path: /spare-parts-pricing/target/spp-1.0.3.jar
  memory: 3GB
```

Way 1 | Responsibilities

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Dev Team Responsibilities

What went well?

Hardly any infrastructure effort in teams

Supports various languages

Very secure platform

Challenges

- Highly opinionated, so only solves 80% of the use cases
- Making data services available to Cloud Foundry is tricky
- Platform footprint is huge and therefore not suitable for small teams

Way 1 Alternative Technologies

Alternative technology stacks with similar responsibilities

Azure App Service

- Used in multiple projects
- Simple to use, simple to understand
- Rather expensive when running a landscape with multiple services
- Supports most languages and even docker images

AWS Elastic Beanstalk

- Not used in production at any of our customers 🤷‍♂️

Way 773 | Mediumweight

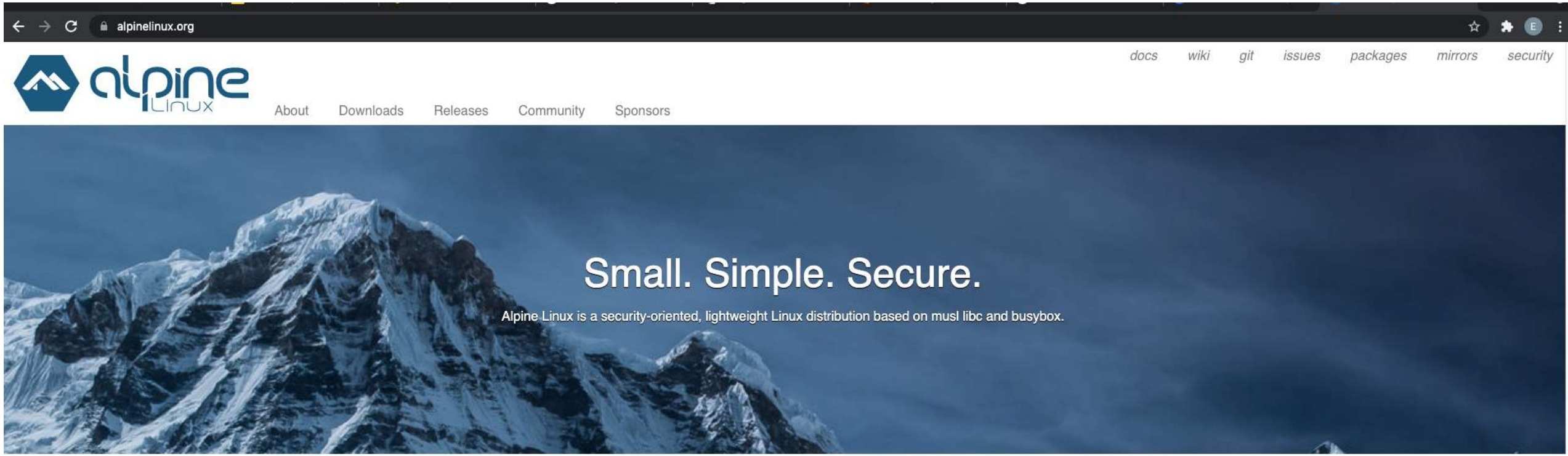
Photo by John McArthur on Unsplash

Way 773 | Medium Complexity

BASE SOFTWARE	RUNTIMES	PACKAGING	COMPUTE	DEPLOYMENT
Alpine	OpenJDK	Docker Multi Stage	Managed container orchestrator	Blue-green

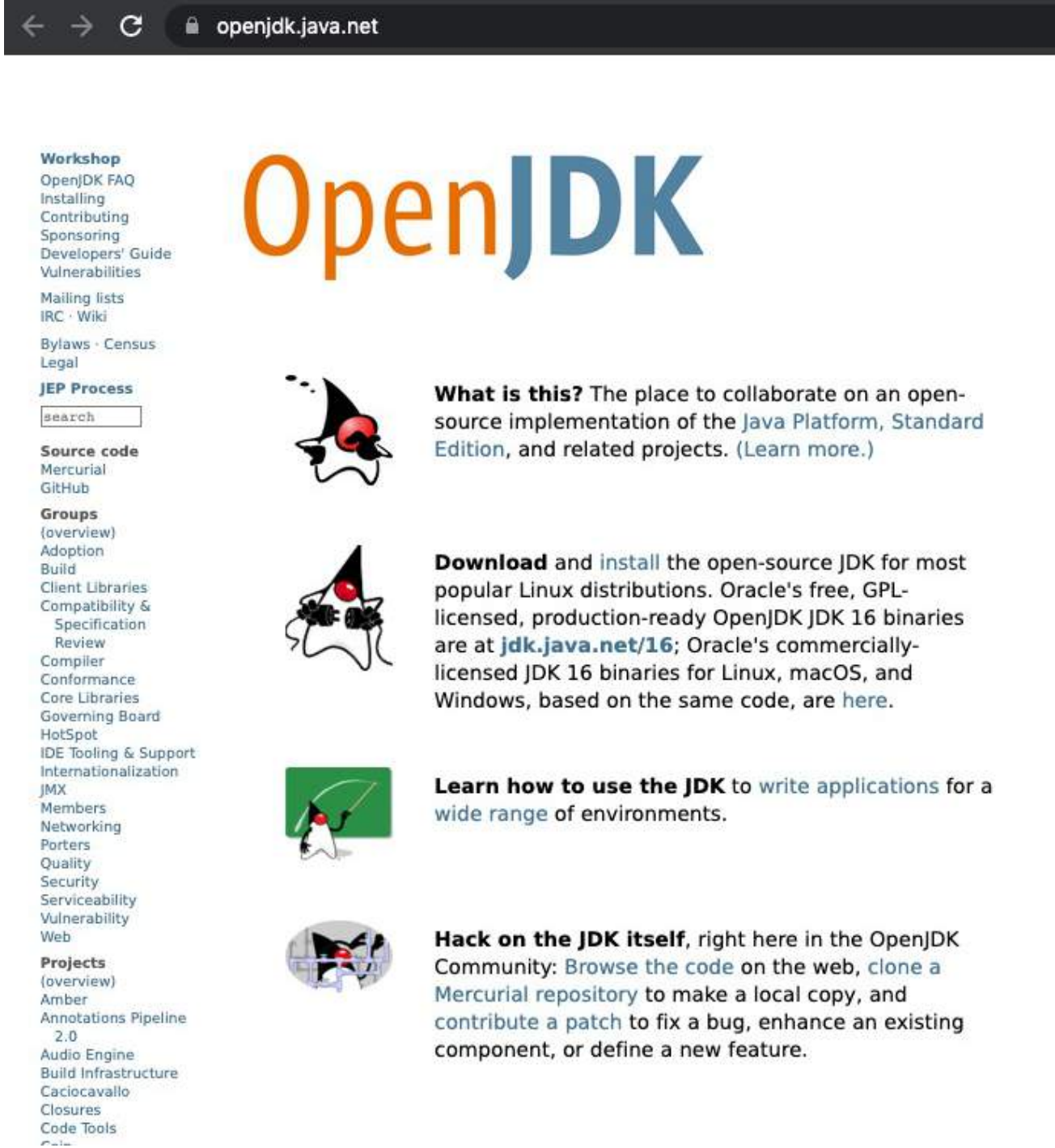
- Java application supporting customer facing operations
- Around 100k requests per day

Alpine



https://wiki.alpinelinux.org/wiki/How_to_get_regular_stuff_working

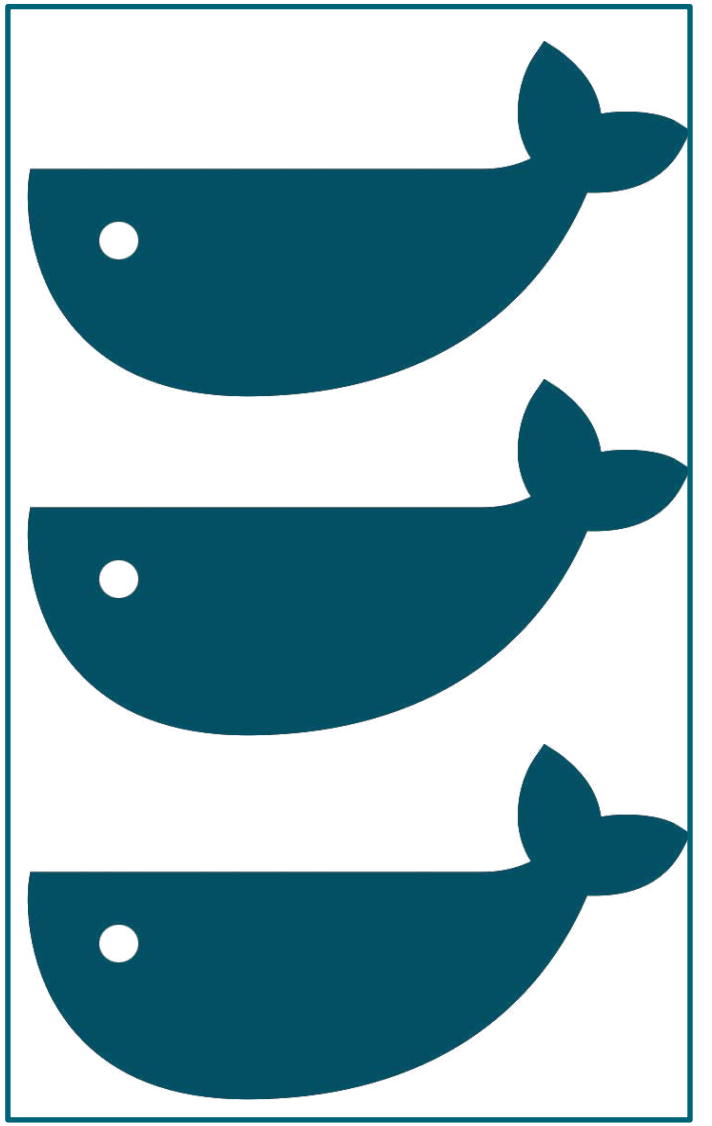
OpenJDK



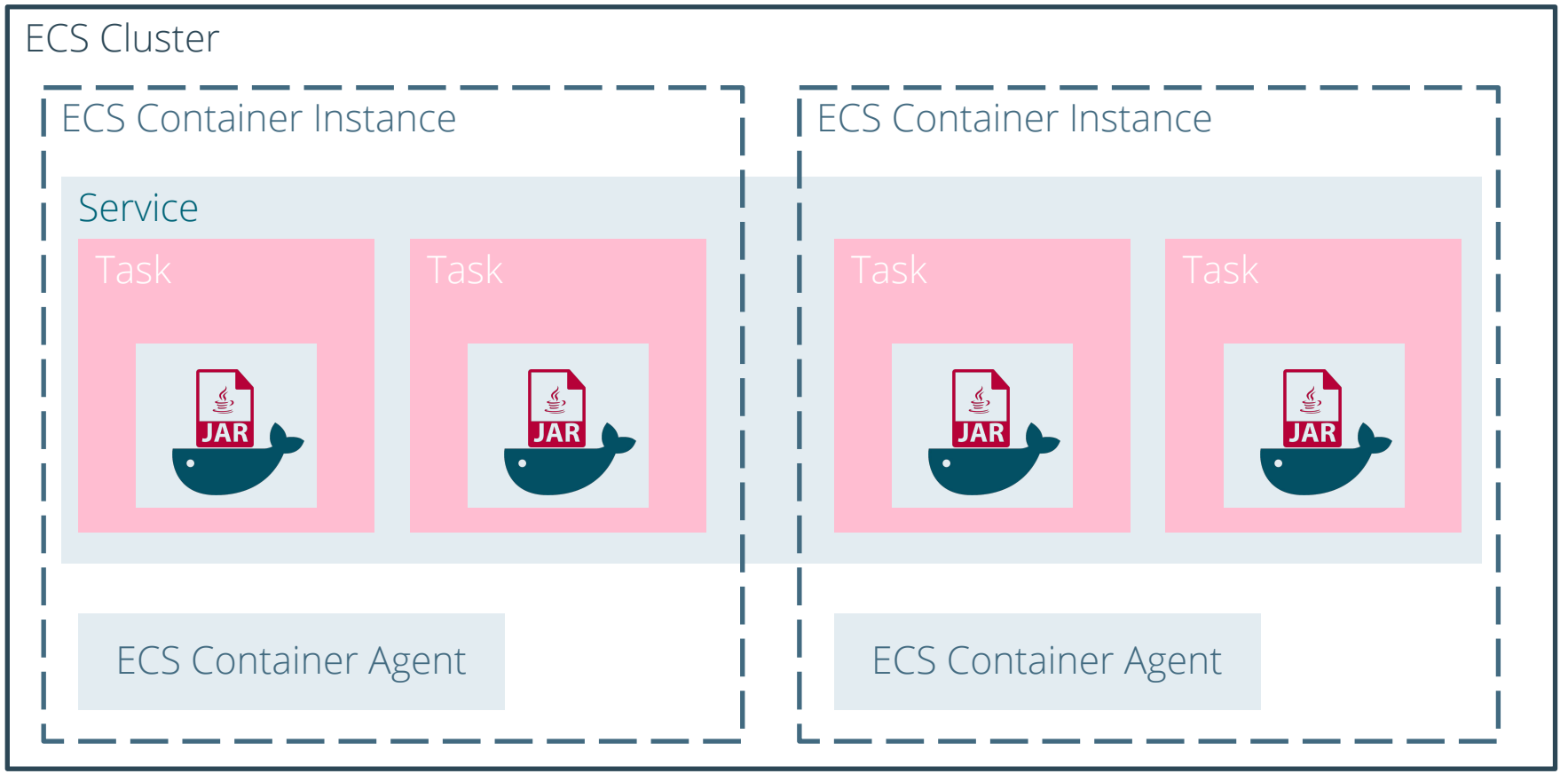
Docker Multi Stage

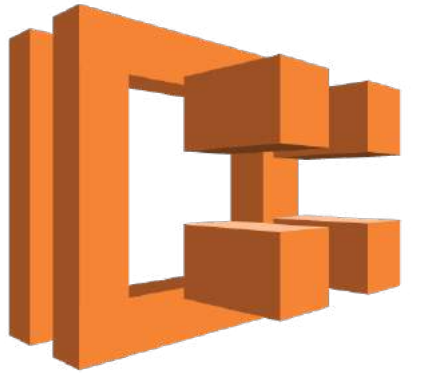
```
# syntax=docker/dockerfile:1
FROM golang:1.16 AS builder
WORKDIR /go/src/github.com/alexellis/href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go    ./
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /go/src/github.com/alexellis/href-counter/app ./
CMD ["./app"]
```



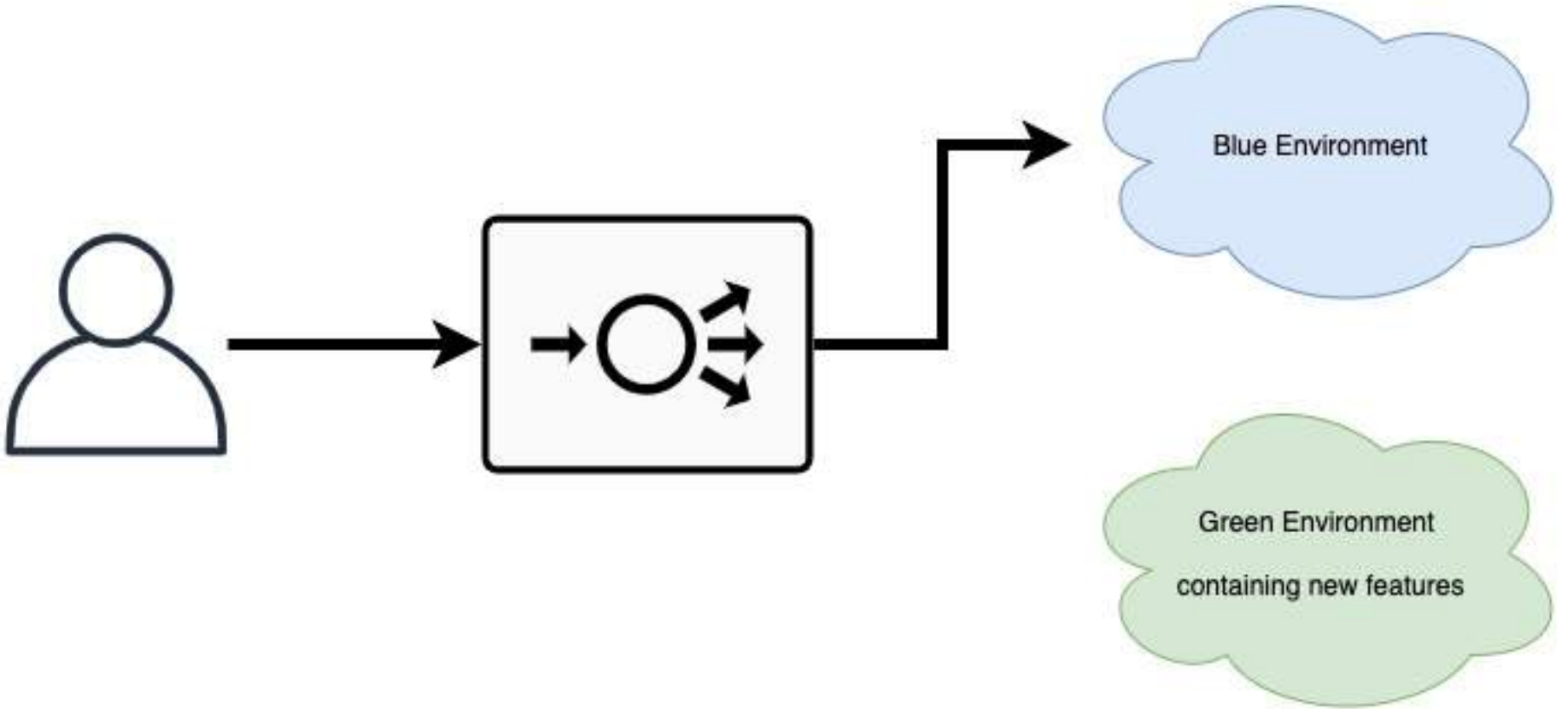
Managed container orchestrator



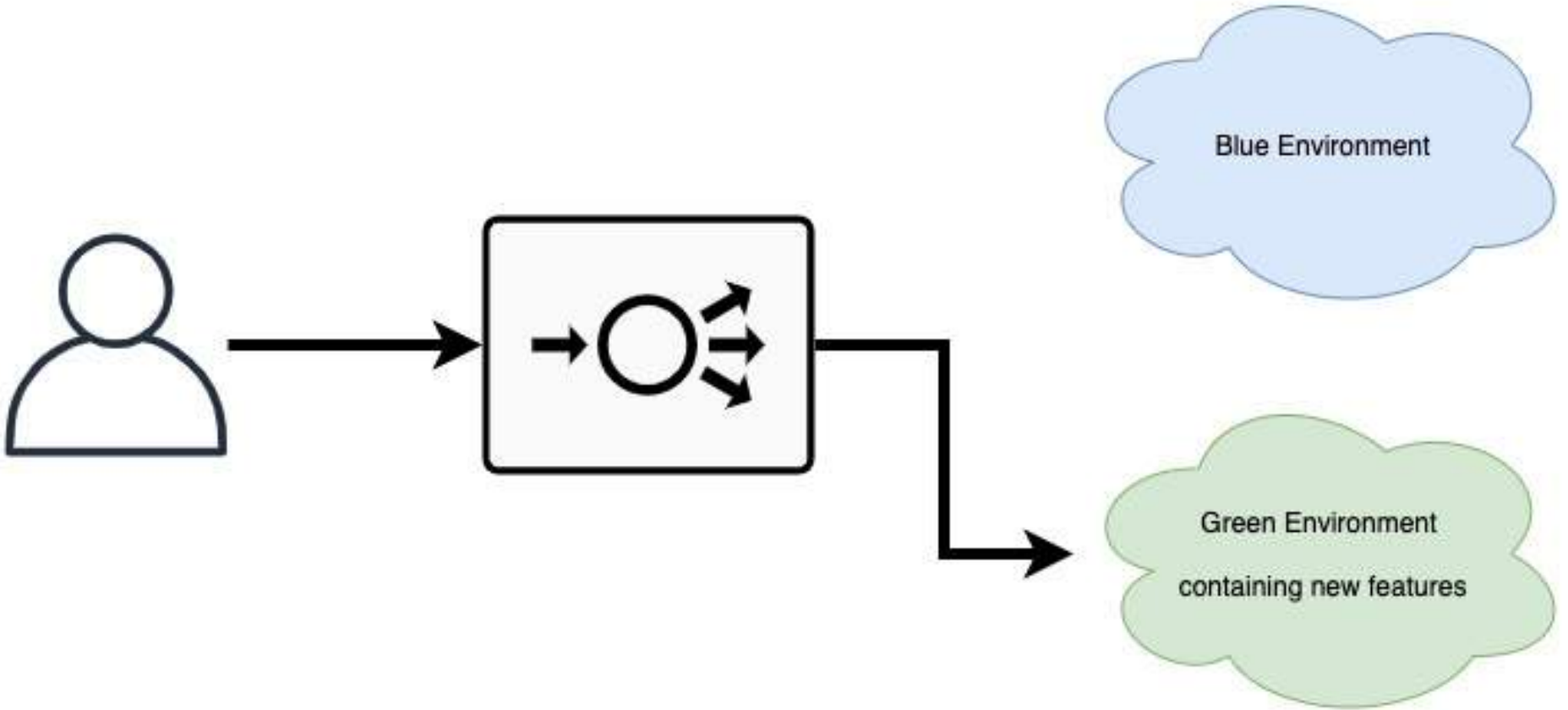


AWS ECS

Blue-green



Blue-green



Way 773 | Responsibilities

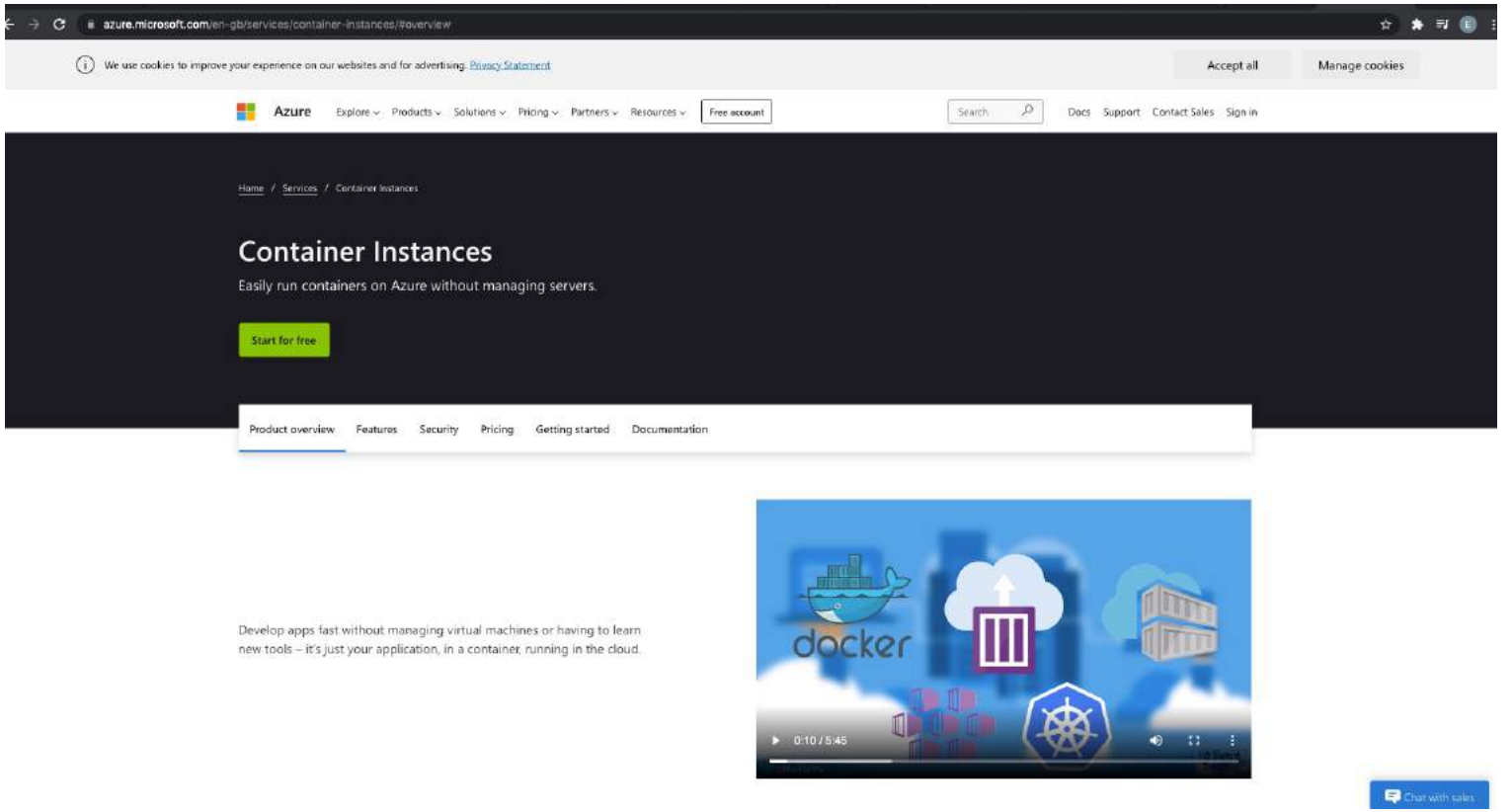
CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Dev Team Responsibilities

Way 773 Alternative Technologies

Alternative technology stacks with similar responsibilities

Azure Container Instances



What went well?

Easy to deploy

Easy to scale

There are ways to break into the containers

Challenges

No way to test the configuration without deploying

A lot of manual configurations: security group, IAM policies, load balancers

Way 3 | Heavyweight

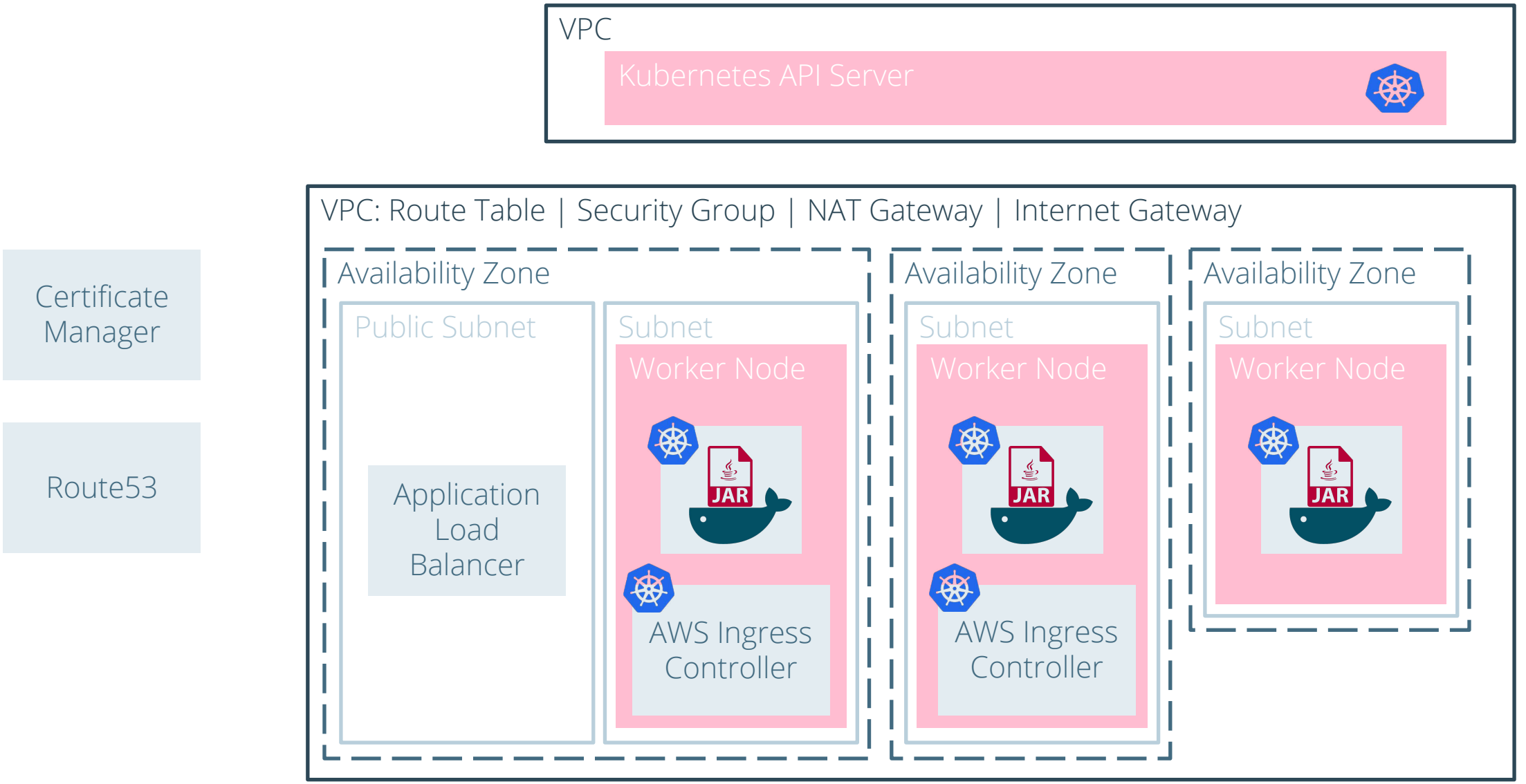
Way 1087 | Heavy Complexity

BASE SOFTWARE	RUNTIMES	PACKAGING	COMPUTE	DEPLOYMENT
Ubuntu	OpenJDK	Docker Multi Stage	BYO K8s	Rolling

kubernetes

- Kubernetes is still young and the ecosystem is still evolving
- Totally hyped and must-use platform since 2019
- FOMO for all not using K8s

Native EKS | AKS



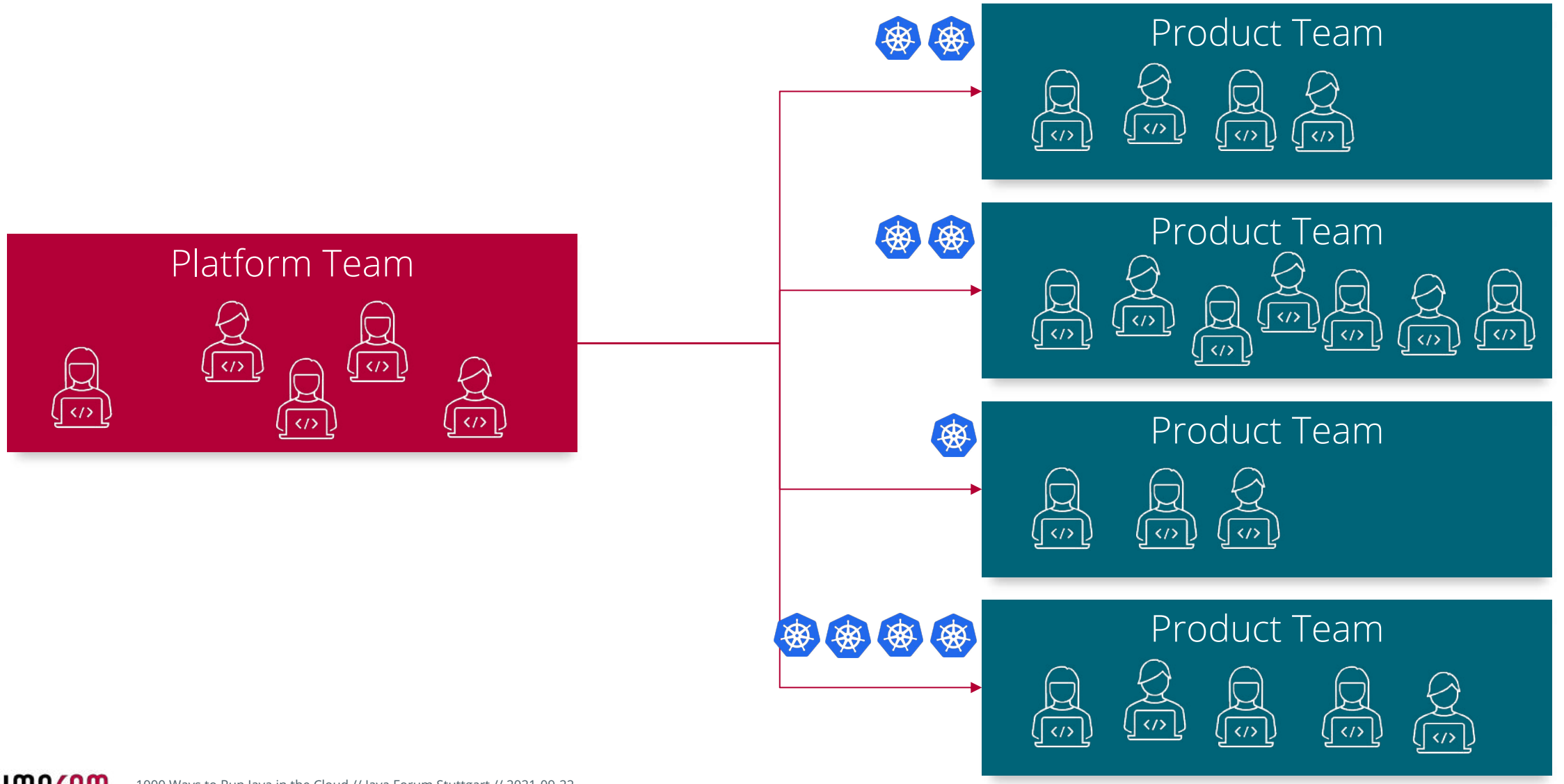
Way 1087 | Responsibilities

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Dev Team Responsibilities

Photo by Jonas Verstuyft on Unsplash

Kubernetes as a Service



Platform Team

What the platform team does

- Builds on existing software:

- Provides isolated K8s clusters on demand
- Enables self-service provisioning of data services
- Solves cross-cutting concerns: routing, monitoring, logging, etc.

- Using Kubernetes in platform teams with 2 customers
- >40 teams working on this
- ~50% product teams building Java apps

Way 1143 | Responsibilities

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Dev Team Responsibilities

Way 1143 | Responsibilities

CODING	DEPLOYMENT	PACKAGING	COMPUTE	ROUTING	OPERATIONS	SECURITY
App	CI/CD	Image Building	Network Setup	Ingress	Monitoring	Certificates In-Transit
	Deployment Strategy	Hardening	Compute Orchestration	Service Discovery	Log Aggregation	Encryption At-Rest
		OS	Infrastructure as Code		Backup & Recovery	Dev Access Control
						Security Concept

Platform Team Responsibilities

What went well?

- Consistent experience across all teams
- Reduced responsibilities significantly for dev teams
- Using Kubernetes this way is actually fun

Challenges

Opinionated stack. Does not work for all use cases.

Integration of native cloud services still challenging (though great support with crossplane.io)

Building and running the platform is a lot of effort.

Conclusion

- There is not a single perfect way to run java in the cloud, but there are a lot of wrong ones.
- When it is about delivering business value, reducing the load on the infrastructure is worth it.

How to Write Your ADR

Make that decision which infrastructure to take

1. Take the list of responsibilites and identify which ones you can live with
2. PoC: Try to manually deploy an example with the infrastructure stack you chose
3. Learn and update your expectations on your responsibilities.
4. Decide.

Image source: Photo by dylan nolte on Unsplash

Scan the QR code to participate in the raffle!

Join us at Gather Town!

Link in the comments ;)

Thanks!

Enrique Llerena Domínguez

Senior Software Engineer @mimacom
Robo Arm Engineer @127.0.0.1

@ellerenad
www.ienjoysoftware.dev

Fabian Keller

Head of Cloud Innovation @mimacom
Woodworker @127.0.0.1

@_fabiankeller
www.fabian-keller.de