Lightweight Java EE for <Buzzword> Services

Coding, building, deploying, scaling: fast and efficient!

Marcus Fihlon, McPringle

May 12, 2017

Software Engineer | Agile Coach | Lecturer | Speaker | Author

Disclaimer

The following presentation has been approved for open audiences only. Hypersensitivity to occasional profanity requires covering ears.

All logos, photos etc. used in this presentation are the property of their respective copyright owners and are used here for educational purposes only. Any and all marks used throughout this presentation are trademarks of their respective owners.

The presenter is not acting on behalf of CSS Insurance, neither as an official agent nor representative. The views expressed are those solely of the presenter.

Marcus Fihlon disclaims all responsibility for any loss or damage which any person may suffer from reliance on this information or any opinion, conclusion or recommendation in this presentation whether the loss or damage is caused by any fault or negligence on the part of presenter or otherwise.

Session Material

Slides, Code, Video

http://fihlon.ch/javacro17

About Me

- **Software Engineer**
  CSS Insurance, Open Source Software
- **Agile Coach**
  CSS Insurance
- **Lecturer**
  TEKO Swiss Technical College
- **Speaker**
  Conferences, User Groups, Meetups
- **Author**
  Articles, Books

www.fihlon.ch | github.com | hackergarten.net | jug.ch

Agenda

Buzzword Bingo

Theory of Evolution

Live Coding

Wrap-up

Buzzword Bingo

Buzzword Bingo

- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other problems!
- **Docker**
  Just because it’s cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Buzzword Bingo

- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other problems!
- **Docker**
  Just because it’s cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Buzzword Bingo

- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other problems!
- **Docker**
  Just because it’s cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Buzzword Bingo

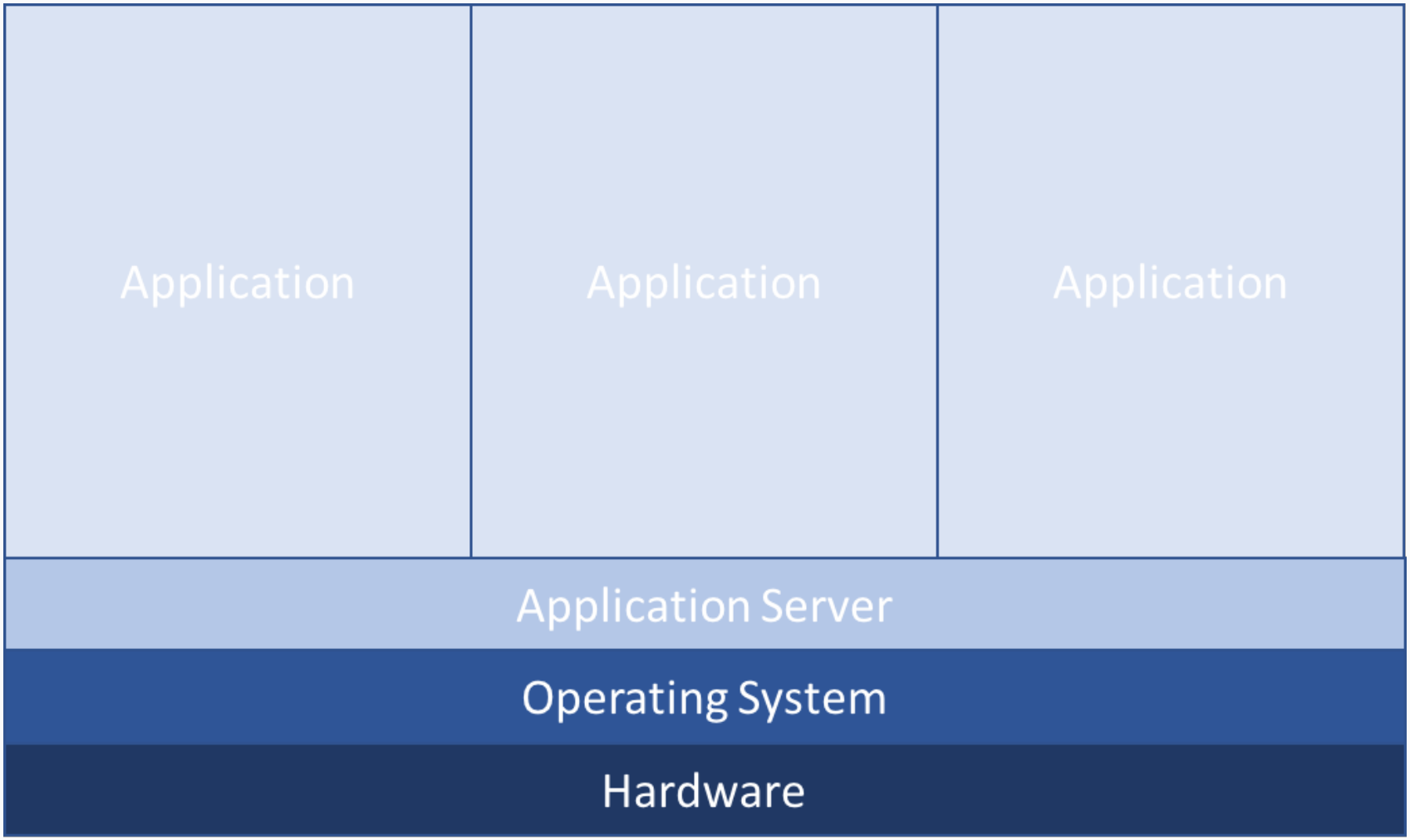
- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other problems!
- **Docker**
  Just because it’s cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Buzzword Bingo

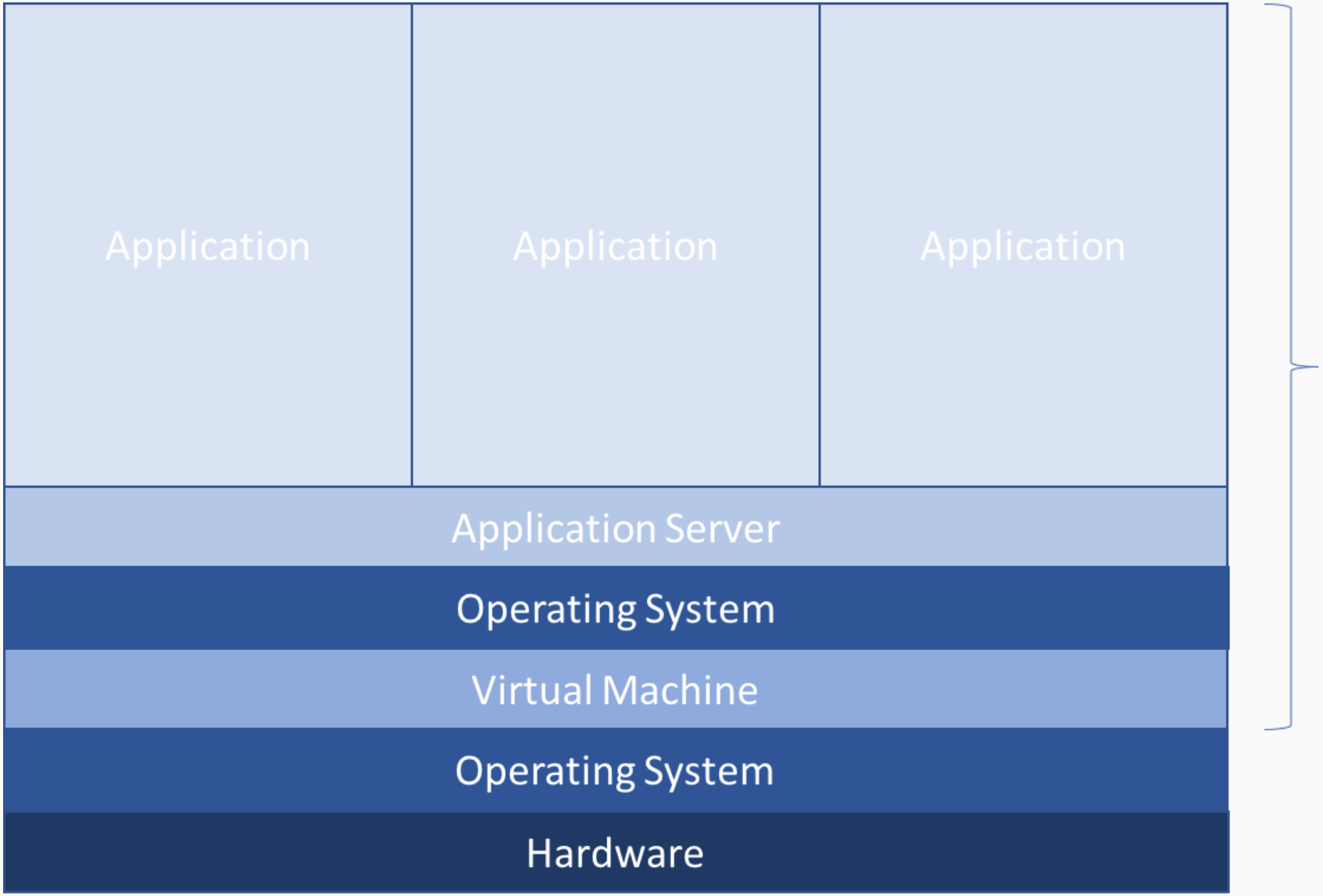
- **Monolith**
  Bad by default!
- **Microservices**
  Solve every problem!
- **Nanoservices**
  Solve all other problems!
- **Docker**
  Just because it’s cool!
- **Java EE**
  Uncool and heavy framework for bloated monoliths!

Theory of Evolution

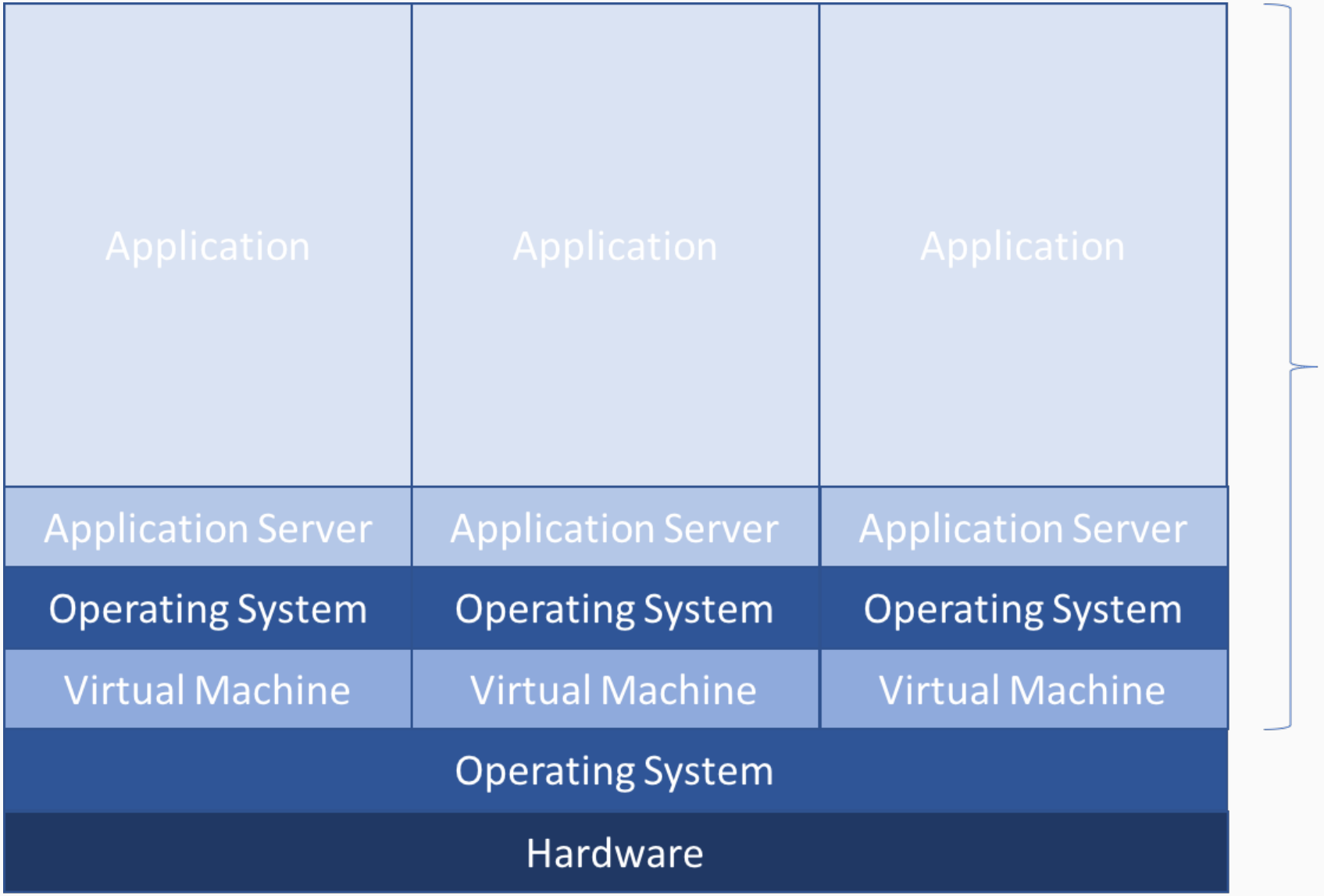
Expensive Resources



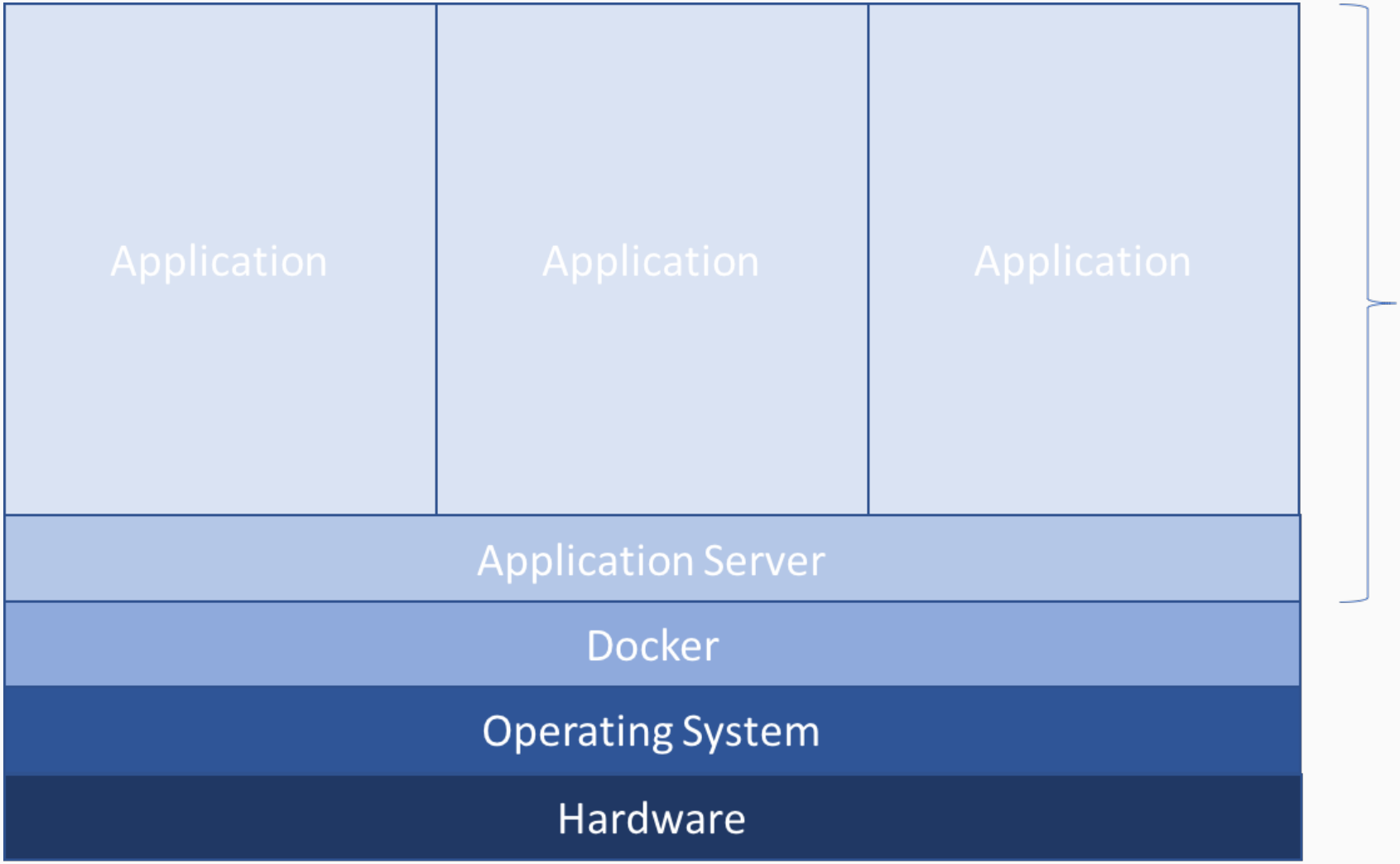
Add Virtualization



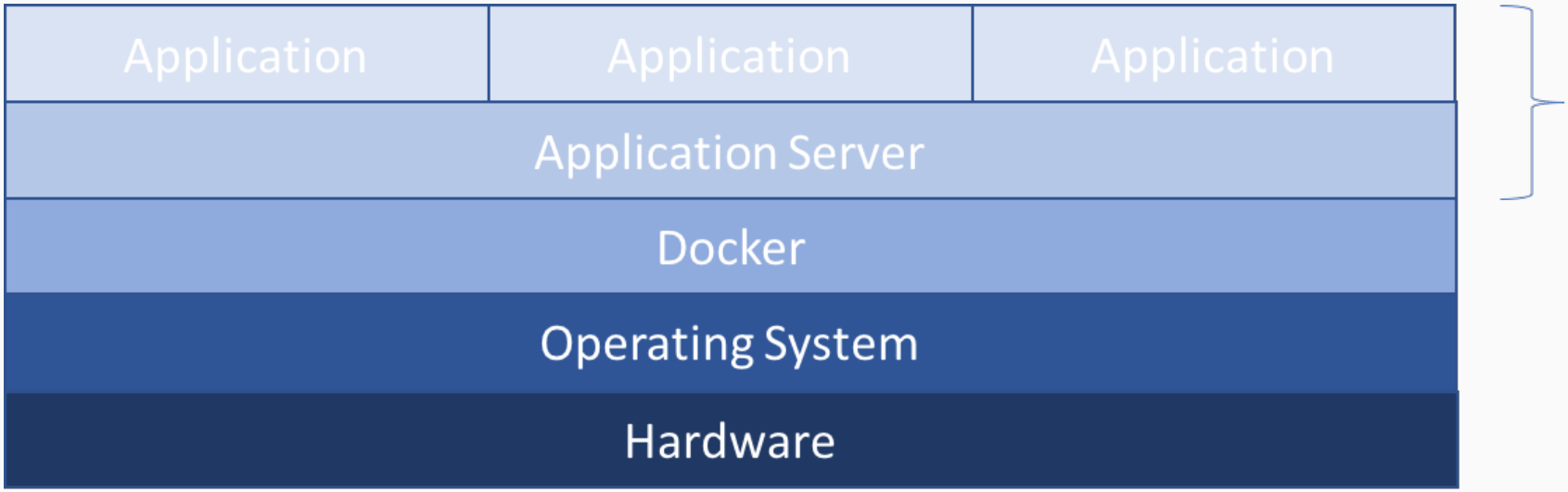
One VM per application



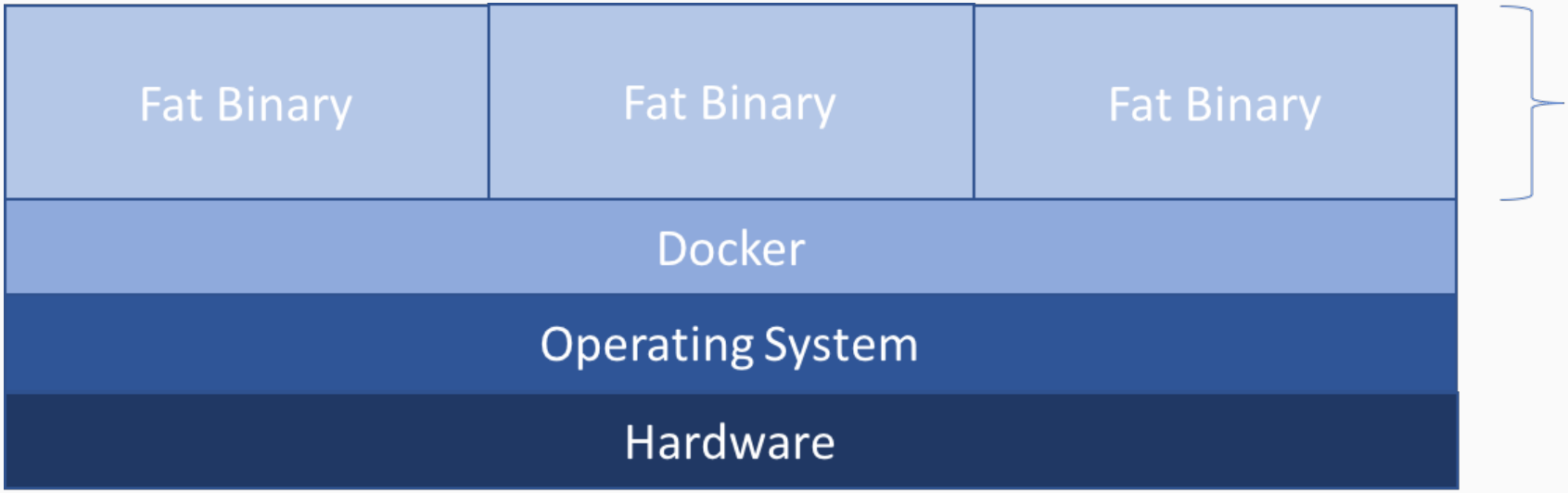
Shared Resources with Docker



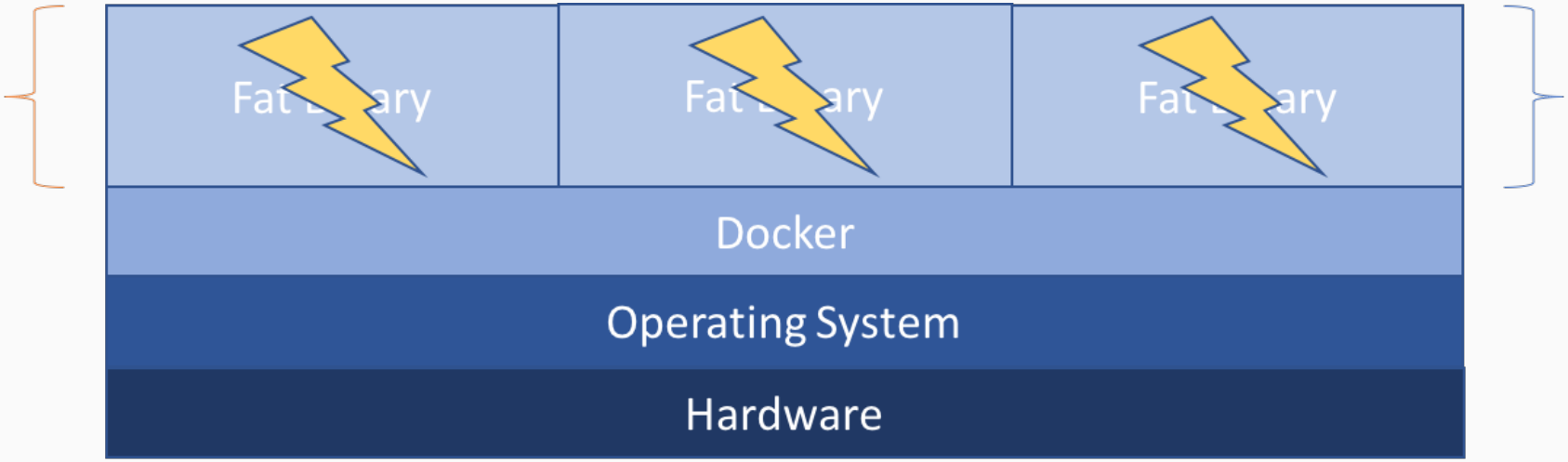
Microservice Hype



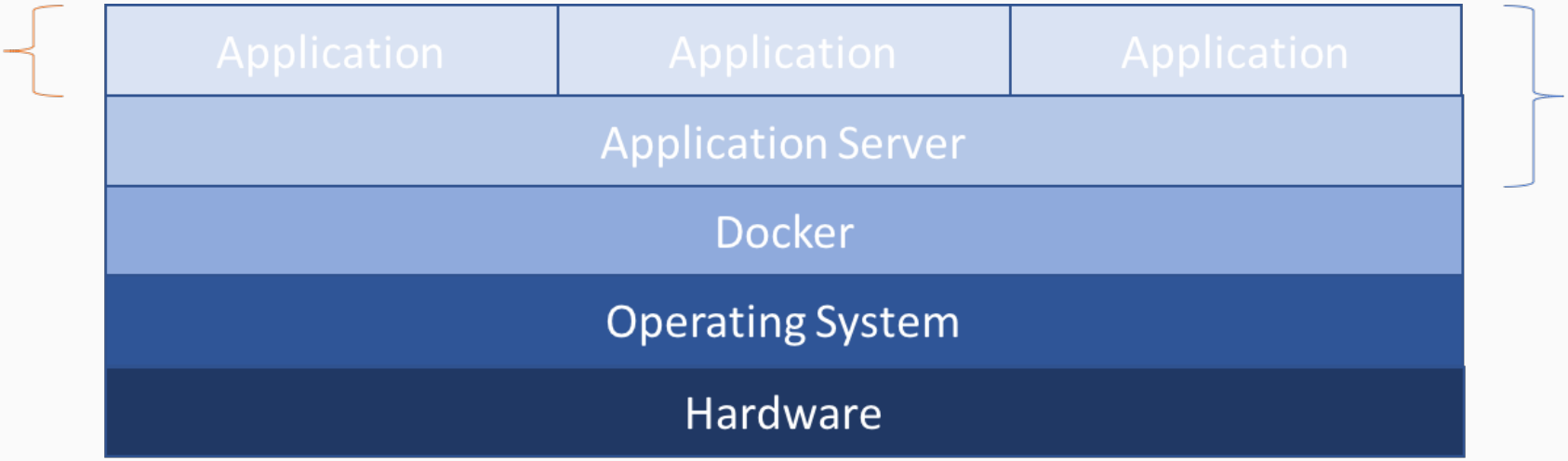
Fat Binaries...



...long build times



Short build times



Live Coding

Prepare the Swarm

- List all available machines
  ```
  docker-machine ls
  ```
- Create a new machine as a manager
  ```
  docker-machine create -d virtualbox mgr
  ```
- Create a new machine as a worker
  ```
  docker-machine create -d virtualbox node01
  ```
- Login to the manager
  ```
  docker-machine ssh mgr
  ```
- Initialize the swarm
  ```
  docker swarm init \
  --advertise-addr 192.168.99.100
  ```

Prepare the Visualizer

- Start the visualizer service

```
docker run -it -d -p 8080:8080 \
-e HOST=192.168.99.100 -v \
/var/run/docker.sock:/var/run/docker.sock \
manomarks/visualizer
```

- Open the visualizer service in a web browser

```
http://192.168.99.100:8080/
```

Join the Swarm

- Ask the manager node for the token to join the swarm
  `docker swarm join-token worker`
- Login to the worker node
  `docker-machine ssh node01`
- Join the worker to the swarm
  `docker swarm join --token <token> \`
  `192.168.99.100:237`

Verify the Swarm

- Login to the manager
  `docker-machine ssh mgr`
- List all nodes
  `docker node ls`
- Show more detailed information
  `docker info`

Create a Network

- Login to the manager
  `docker-machine ssh mgr`
- Create a new overlay network
  `docker network create -d overlay javacro`
- List all networks
  `docker network ls`

Deploy Services

- Login to the manager
  ```
  docker-machine ssh mgr
  ```
- Deploy the time service
  ```
  docker service create --network javacro \
  --name timeservice mcpringle/timeservice
  ```
- Deploy the hello service
  ```
  docker service create --network javacro \
  -p 8181:8080 \
  --name helloservice mcpringle/helloservice
  ```
- List all services
  ```
  docker service ls
  ```

Testing, Scaling and Draining

- Testing our services

```
curl \
http://192.168.99.100:8181/api/hello/JavaCro
```

- Scaling services up and down

```
docker service update \
--replicas 3 helloservice
```

- Draining the manager node

```
docker node update --availability drain mgr
```

Wrap-up

Conclusion

With Java EE and Docker you get...

- easy to understand code
- fast build times
- reproducable deployments
- easy scaling of your services

You are fast and efficient because you have your focus on your business code. **You create business value!**

Thank You! Questions?

Slides, Code, Video

http://fihlon.ch/javacro17