Cloud-Native Application Design

Zurich University of Applied Sciences

Presenter: Sandro Brunner
e-mail: brnr@zhaw.ch

ICCLab

Part of InIT - Institute of Applied Information Technology

Research Lab at ZHAW in Winterthur CH

Currently 25 Researchers

ICCLab - Research Topics

- **Research Themes**
  - Energy Efficiency in Cloud Computing
  - Infrastructure as a Service (IaaS)
  - Platform as a Service (PaaS)

- **Research Initiatives**
  - Cloud Dependability and High Availability
  - Cloud Incident Management
  - Cloud Orchestration
  - Cloud Storage
  - Cloud-Native Applications
  - Distributed Computing in the Cloud
  - Energy Aware Cloud Load Management
  - PaaS on OpenStack
  - Rating – Charging – Billing
  - Software Defined Networking for Clouds
  - Understanding Cloud Energy Consumption

Cloud-Native Application

What is a Cloud-Native Application?

Application **optimized** to run in the cloud. Takes **advantage** and **considers the drawbacks** of the cloud-environment.

Main Characteristics of a Cloud-Native Application

Scalabilty & Resilience

Also possible to get there by **migrating** an already existing application.

Motivation

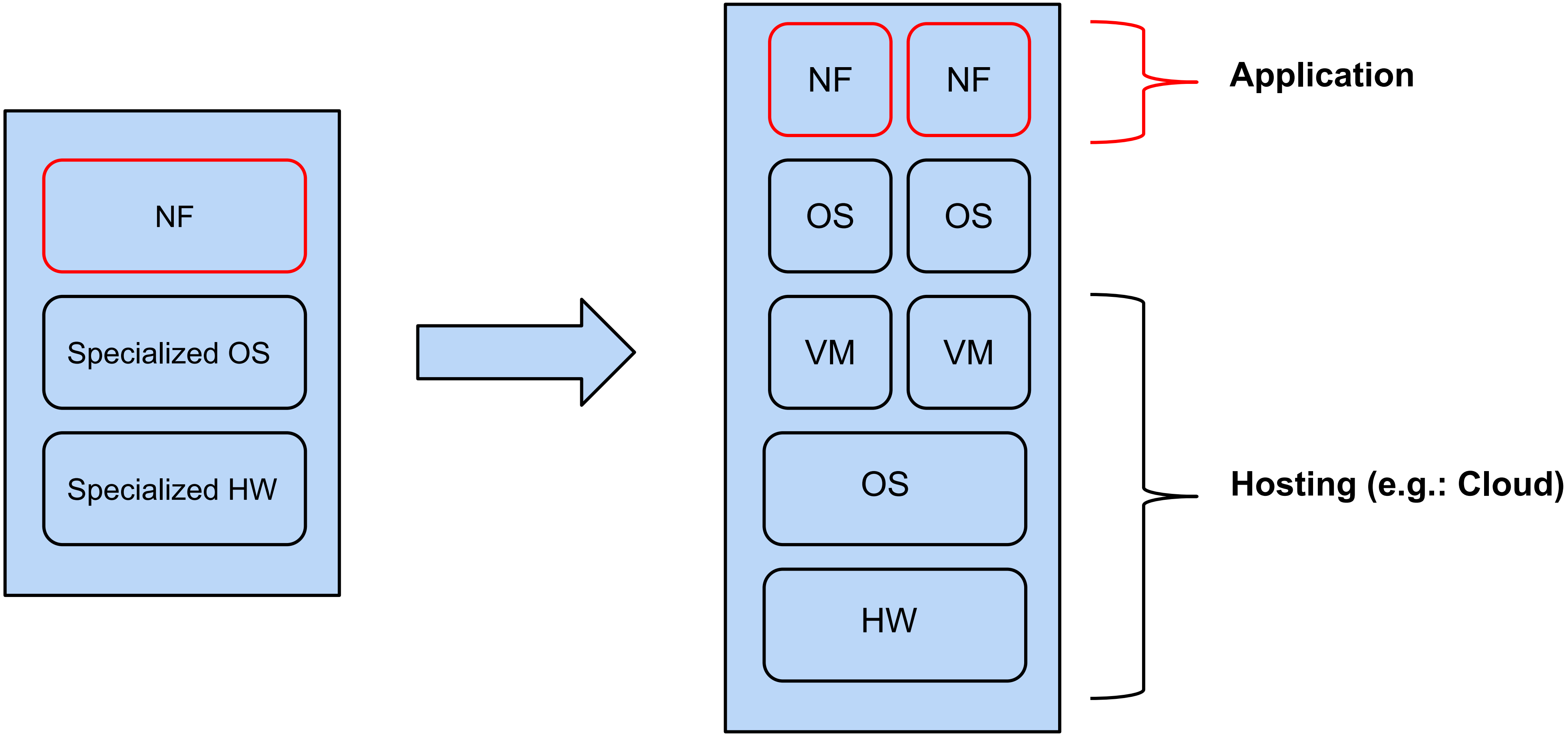
Exploiting Benefits of Cloud Computing

- Obtain IT-Resources on Demand (Compute, Storage, Network)
- Pay-as-you-go Pricing-Model → No upfront costs
- Speeding-Up Development / Deployment Cycle
- Transfer responsibility of operating infrastructure
- ...

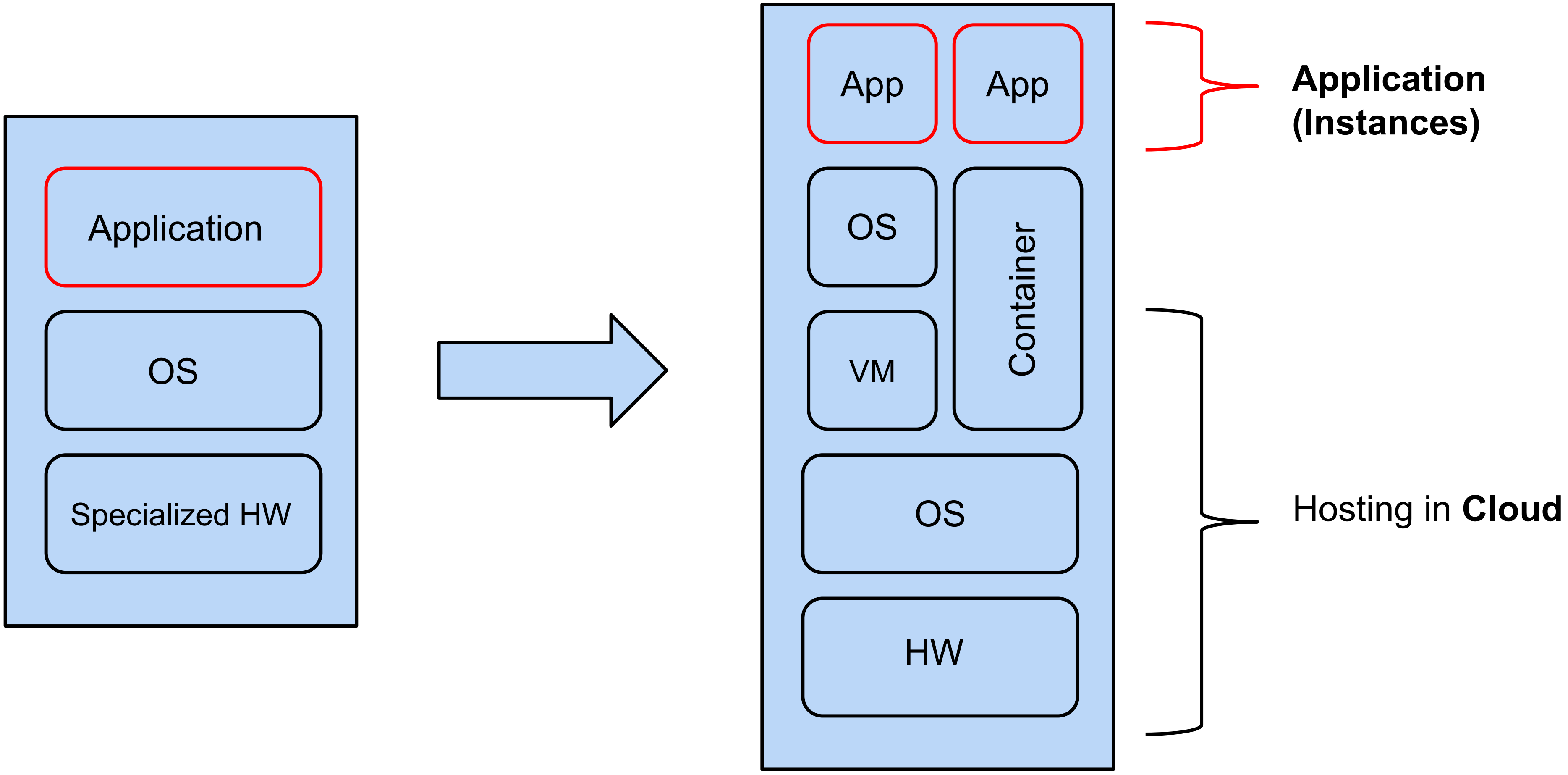
Can be boiled down to **economica**l reasons/benefits

→ Reduce Costs through Technology
→ Improve Time-To-Market through Technology

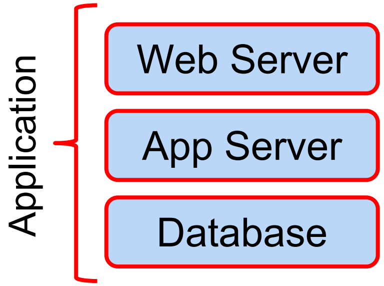
NFV vs Cloud-Native Applications I



NFV vs Cloud-Native Applications II



Designing a Cloud-Native Application



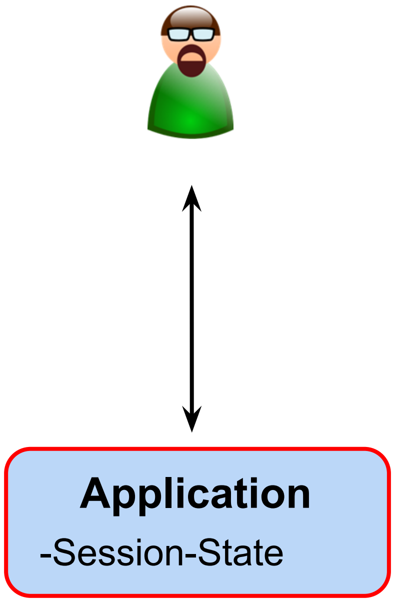
Application

Web Server

App Server

Database

Example: Simple Web-Application



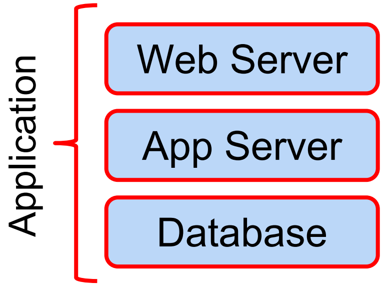
Application

-Session-State

: Welcome Back, John Doe
: Contents of Shopping Cart

Designing a Cloud-Native Application

Example: Simple Web-Application

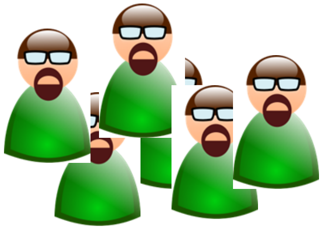


Application

Web Server

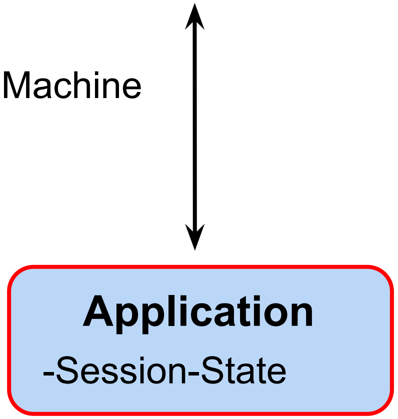
App Server

Database



Too much load for current configuration

→ Naive Solution: Scale Up / Get Bigger Machine

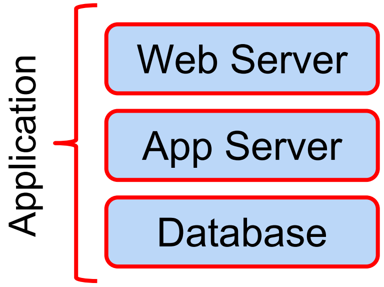


Application

-Session-State

: Welcome Back, John Doe
: Contents of Shopping Cart

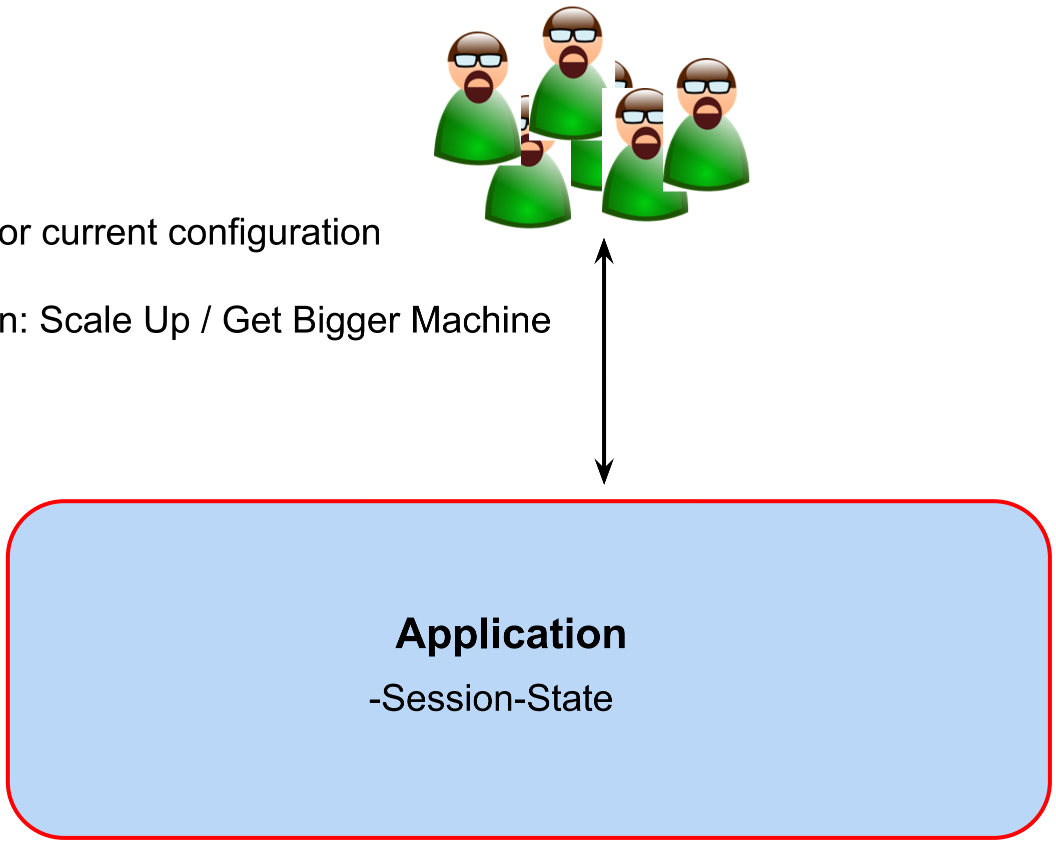
Designing a Cloud-Native Application



Example: Simple Web-Application

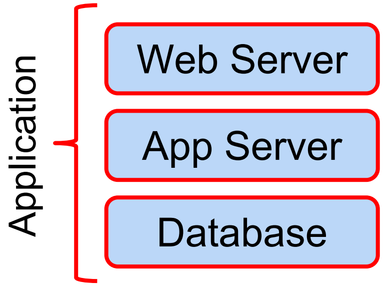
Too much load for current configuration

→ Naive Solution: Scale Up / Get Bigger Machine



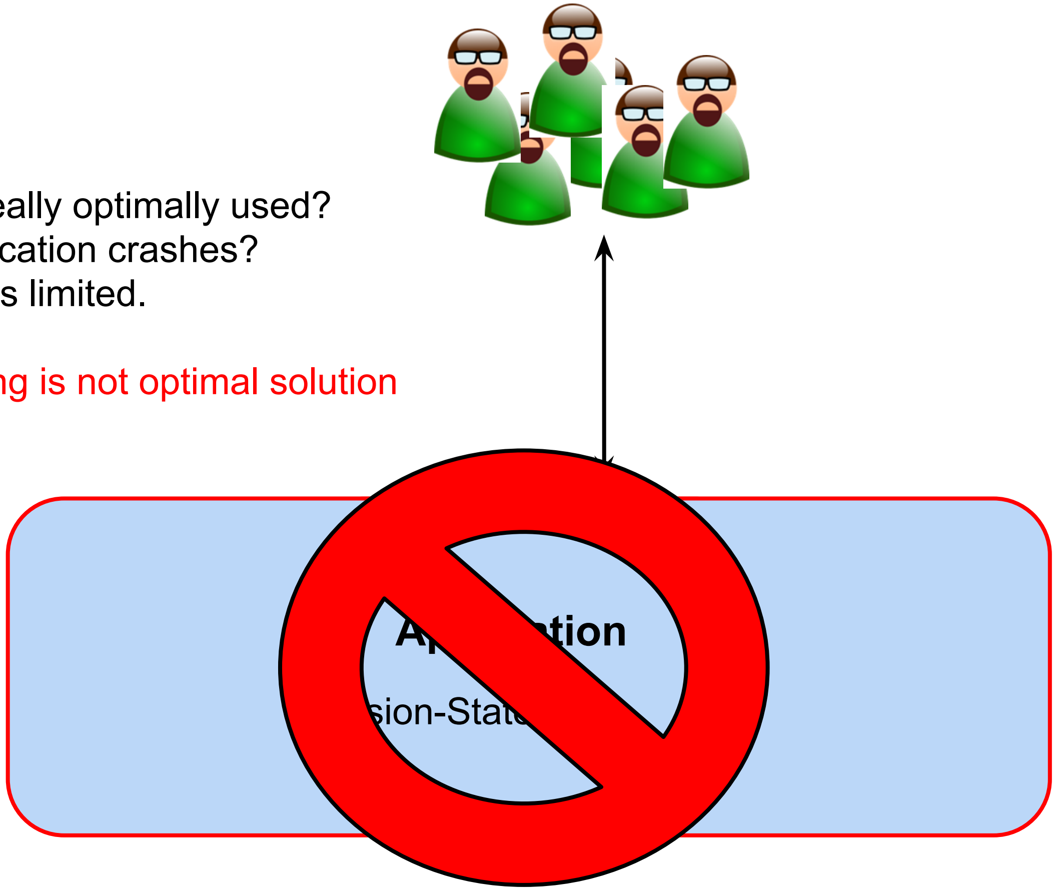
Designing a Cloud-Native Application

Example: Simple Web-Application



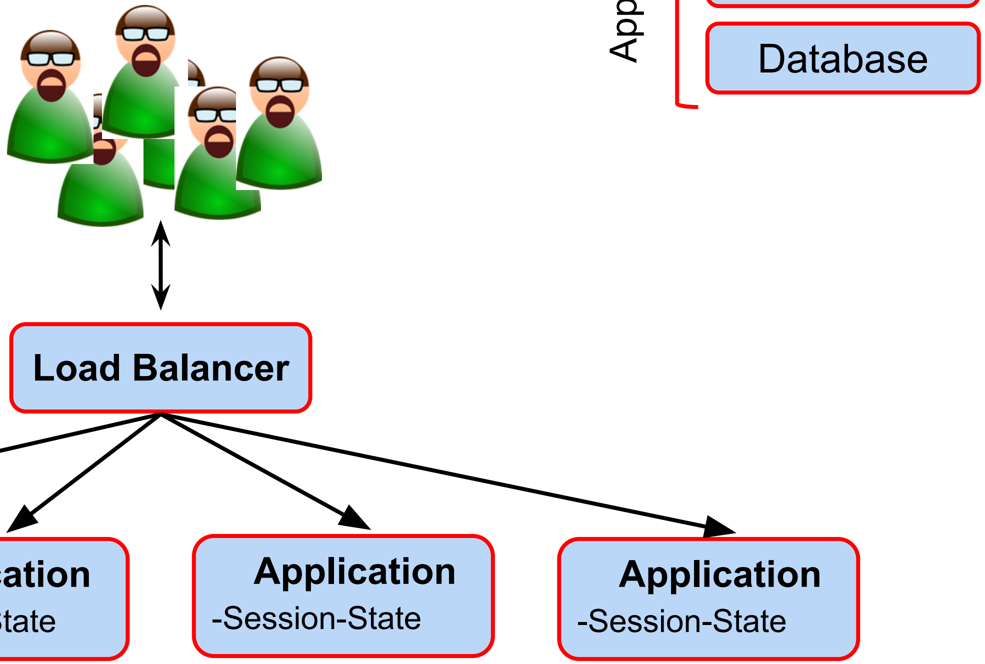
Are resources really optimally used?
What if the application crashes?
Vertical scaling is limited.

→ Vertical scaling is not optimal solution



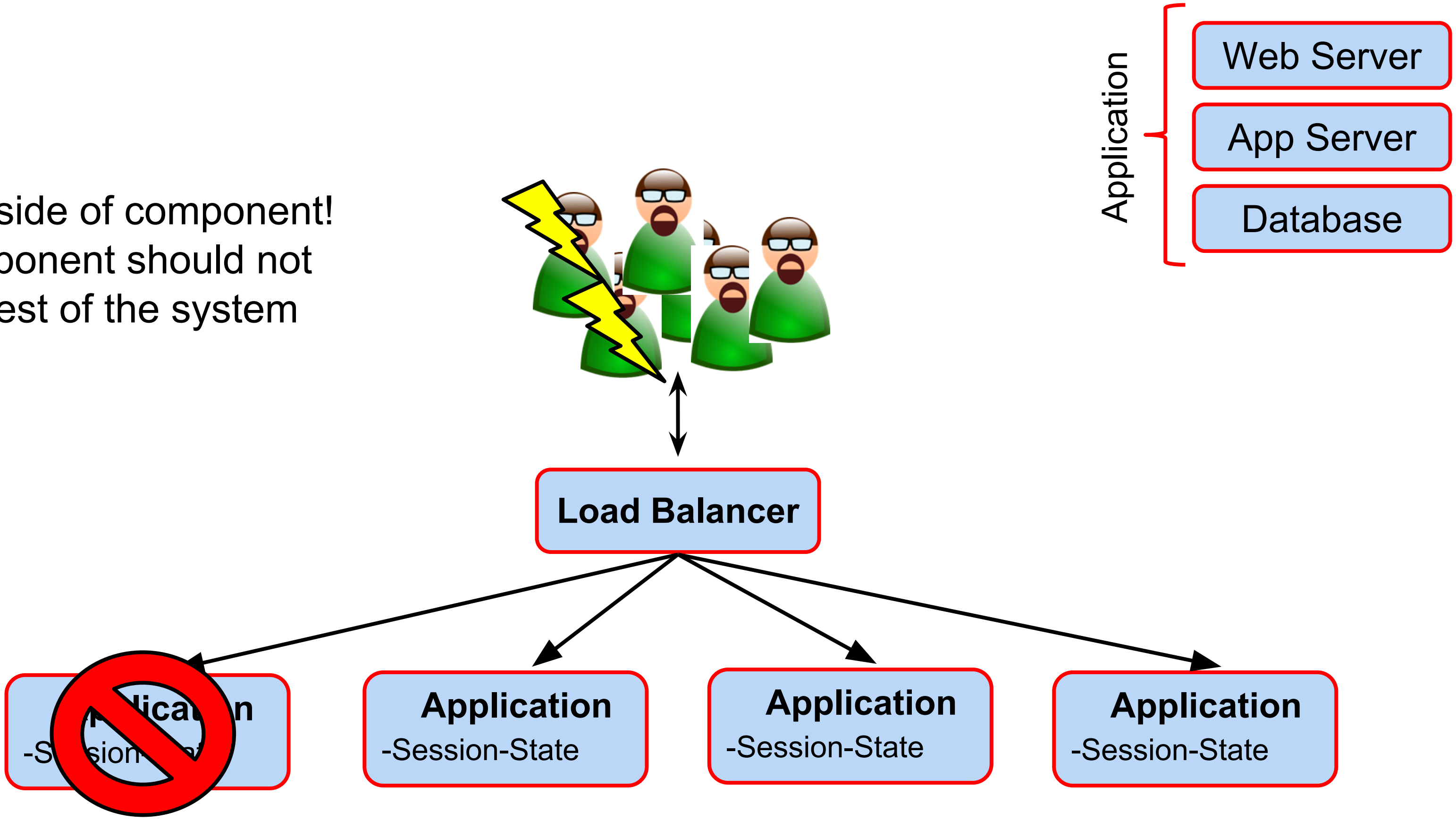
Designing a Cloud-Native Application

Better Solution: **Horizontal Scaling**
→ Resources used more efficiently
→ No over- or underprovisioning of resources

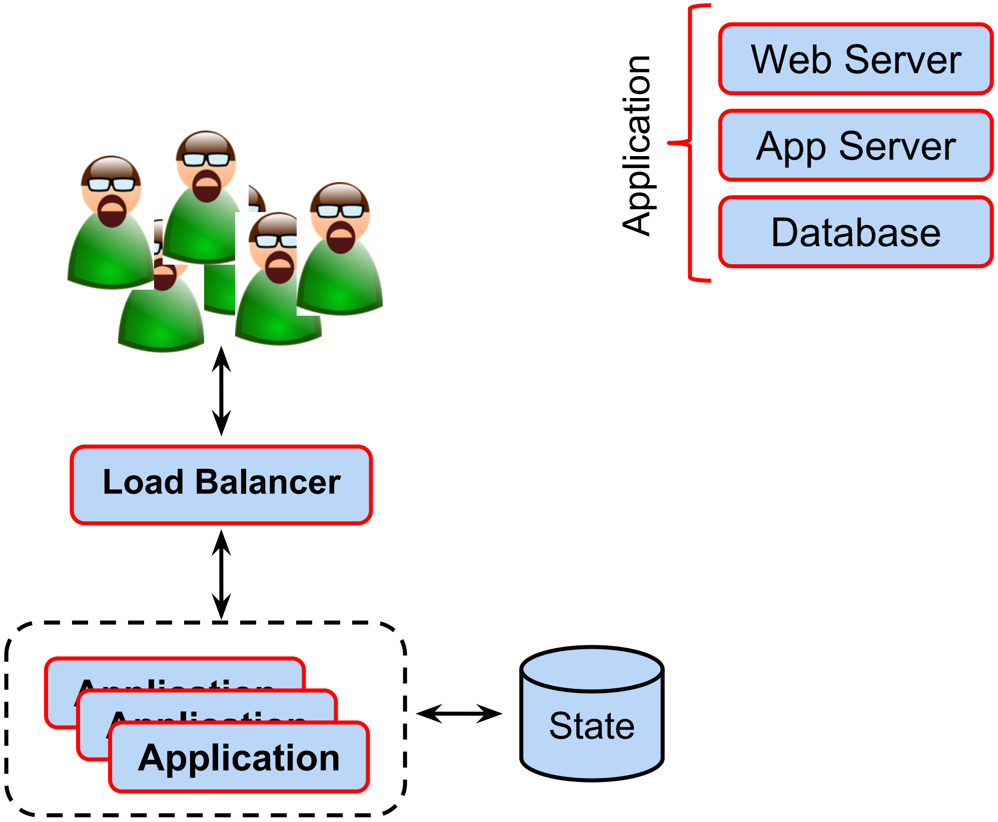


Designing a Cloud-Native Application

Beware!
→ Save state outside of component!
→ Failure of component should not
influence the rest of the system



Designing a Cloud-Native Application



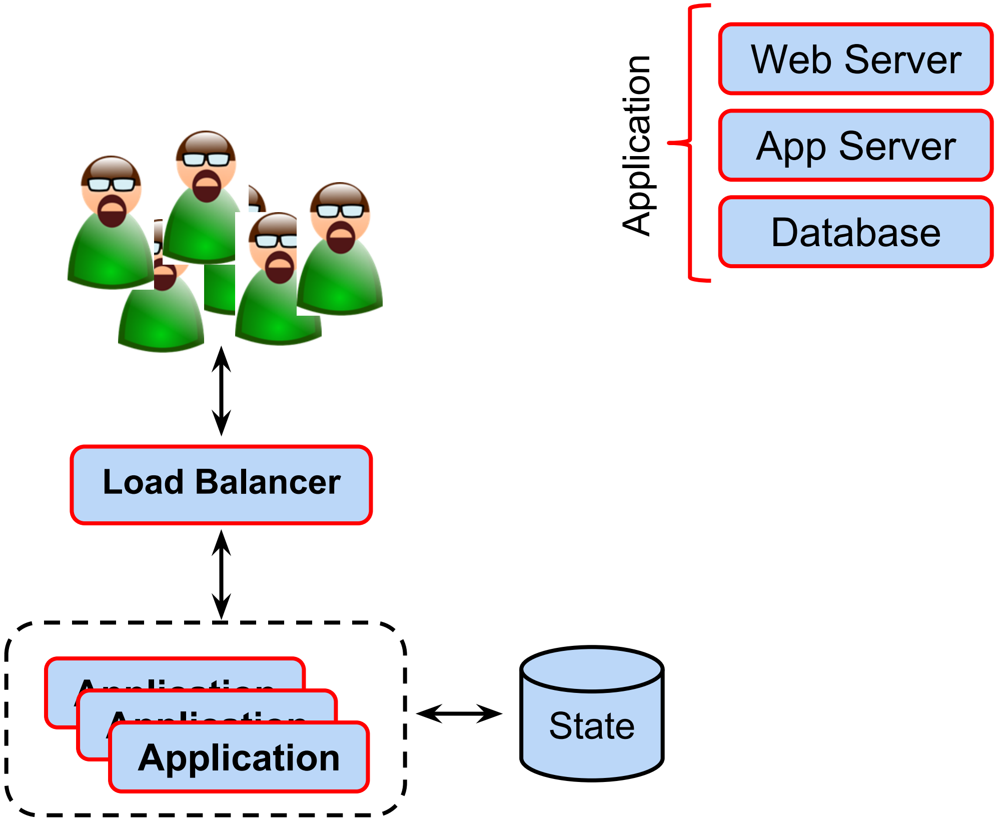
Designing a Cloud-Native Application

Next Step: Automate Scaling

- Need to know “what’s going on”

    Resource Usage, Response Times, …

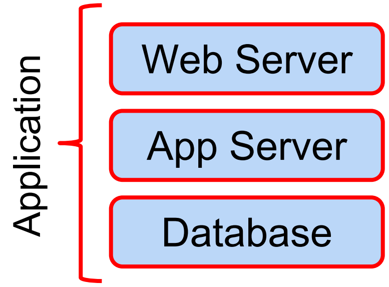
- Need to be able to take actions accordingly

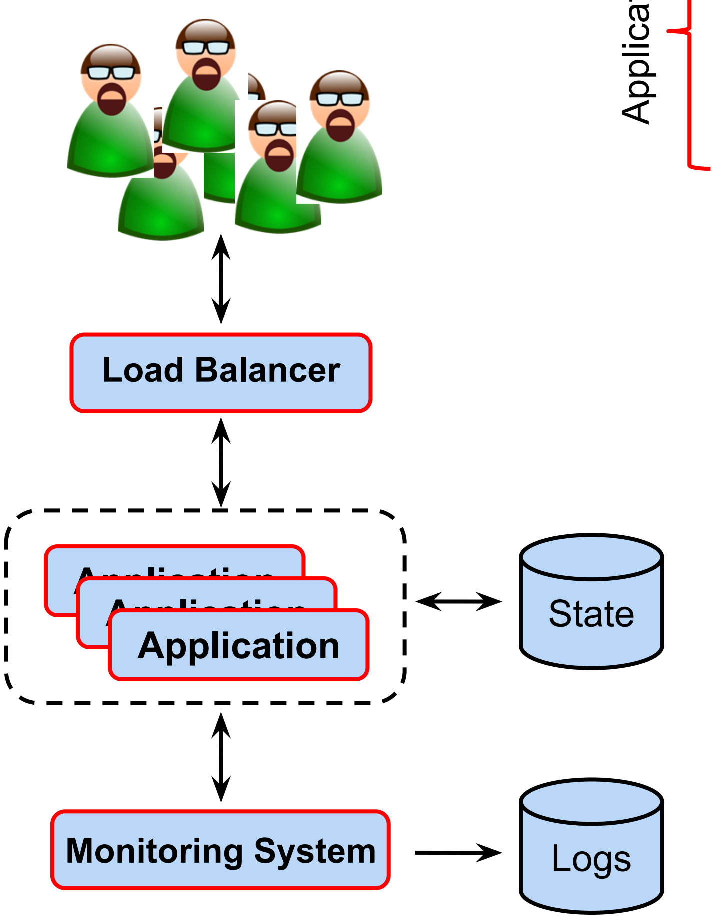


Designing a Cloud-Native Application

Next Step: Automate Scaling
- Need to know “what’s going on”
  Resource Usage, Response Times, …
- Need to be able to take actions accordingly

→ **Monitoring System:**
- Monitor Systems + Applications
- Collect / Aggregate Logs





Designing a Cloud-Native Application

Next Step: Automate Scaling

- Need to know “what’s going on”

  Resource Usage, Response Times, …

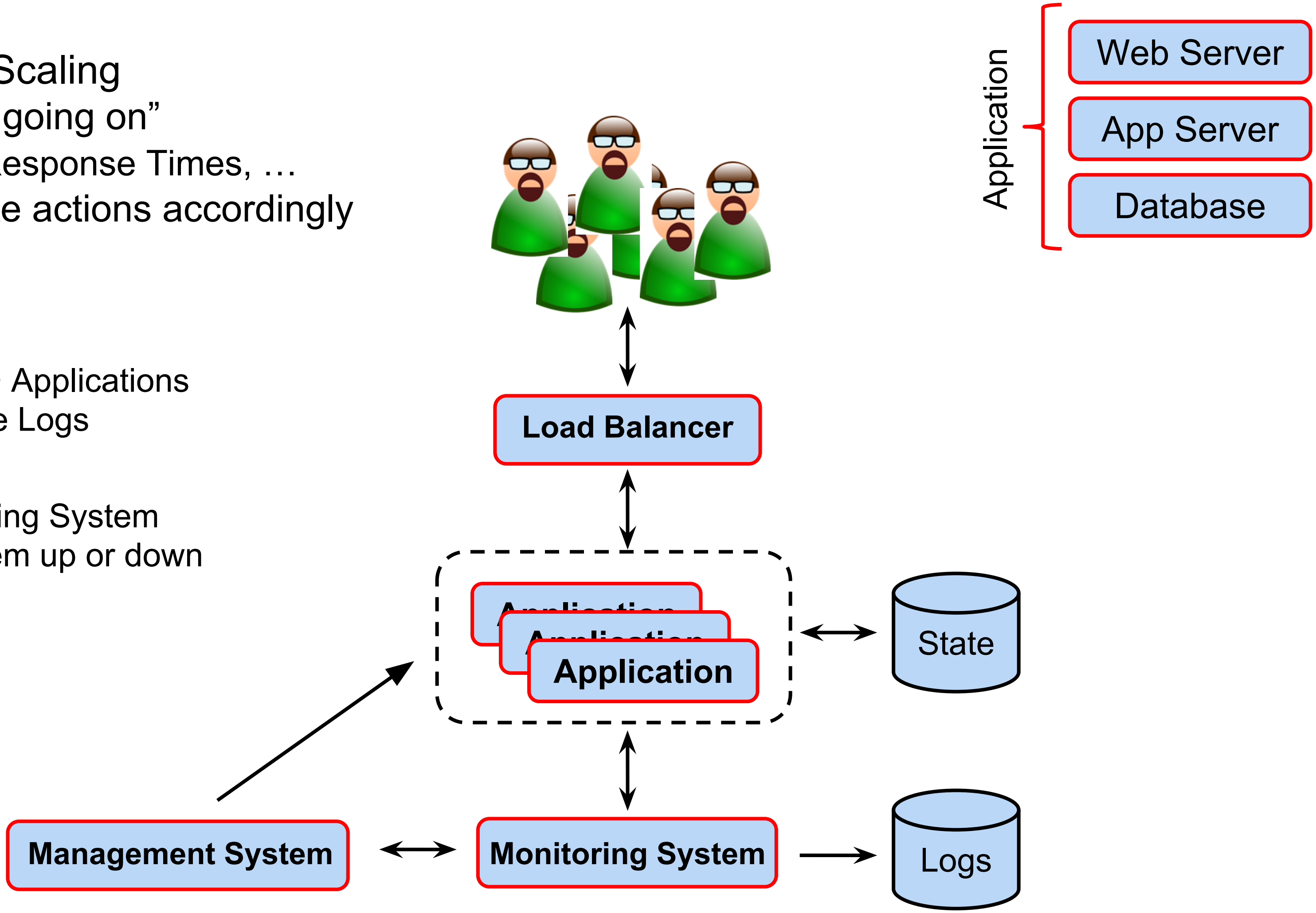
- Need to be able to take actions accordingly

→ **Monitoring System:**

- Monitor Systems + Applications
- Collect / Aggregate Logs

→ **Management System**:

- Input from Monitoring System
- Able to scale system up or down



Summary Cloud-Native Applications

Cloud-Native Applications should be:

Scalable: Run as economically efficient as possible
Resilient: Expect Failure / Infrastructure uses Commodity Hardware

Components of Cloud-Native Applications should be:

Stateless: Outage of a single component should not compromise the whole system
Scalable & Resilient

Cloud-Native Applications are:

A composition of a variety of services (Application, Monitoring, Management)
Distributed Systems
Complex

How to Build Cloud-Native Applications

Loads of problems already encountered and solved

Design Patterns for Cloud-Native Applications
→ Circuit Breaker, Valet Key, Bulkhead, Retry

Services offered by Cloud Vendor (Amazon, Google, Microsoft)

Open Source Libraries / Frameworks:
→ Netflix OSS – e.g.: Hystrix, Ribbon, Chaos Monkey, etc.
→ Twitter – Zipkin, Snowflake, Finagle, Mesos
→ Spring Cloud

Open Source Tools
→ Caches, Key-Value Stores, Webserver, Load-Balancer, Messaging/Queuing Systems, Service Registries, Configuration Management, Monitoring/Log Data Collection & Analysis, Load/Performance Tester

Questions

?

Links

ICCLab:

- http://blog.zhaw.ch/icclab/

Cloud-Native Applications Initiative:

- http://blog.zhaw.ch/icclab/category/research-approach/themes/cloud-native-applications/

ZHAW InIT

- http://init.zhaw.ch/en/engineering/institutes-centres/institute-of-applied-information-technology.html

Links II

Additional Resources

Book: Cloud Design Patterns

Libraries: Netflix OSS, Twitter Open Source, Spring Cloud

Caches / Key-Value Stores: Memcached, redis, etcd, Apache Zookeeper

DBs: Druid, Apache Cassandra, InfluxDB

Webserver / Proxys: Apache HTTP Server, nginx, HAProxy

Messaging/Queuing Systems: RabbitMQ, Apache Kafka, Queues.IO, beanstalkd, ejabberd

Configuration Management Tools: cdist, Chef, Puppet

Monitoring / Log Data Collection & Analysis: Zabbix, nagios, New Relic, Loggly, fluentd, logplex, Elasticsearch, logstash, kibana, Sensu

Load/Performance Tester: loader.io, Jmeter, stress, Tsung, httperf

Various: Hystrix, Graphite, Jenkins, CloudFlare, Varnish, PgBouncer, Gearman, Quartz