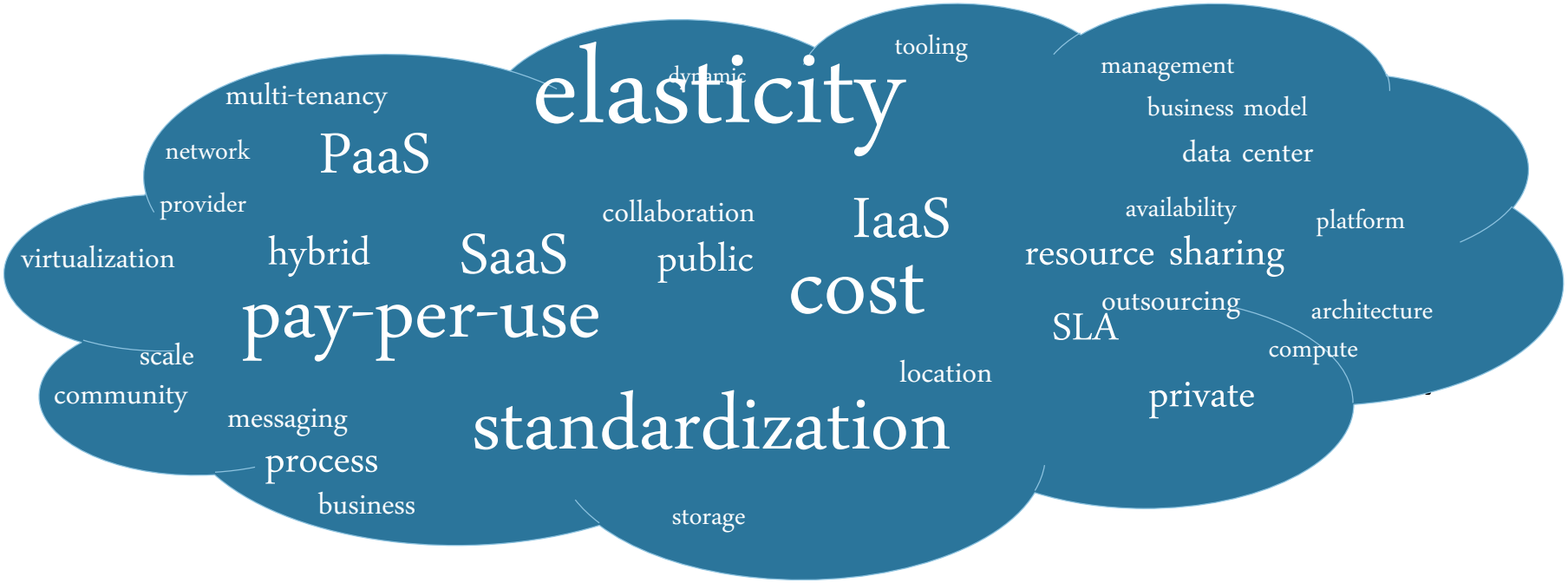
Applying Architectural Patterns for the Cloud:
Lessons Learned During Pattern Mining and Application

Ralph Retter (Daimler TSS GmbH)
ralph.retter@daimler.com

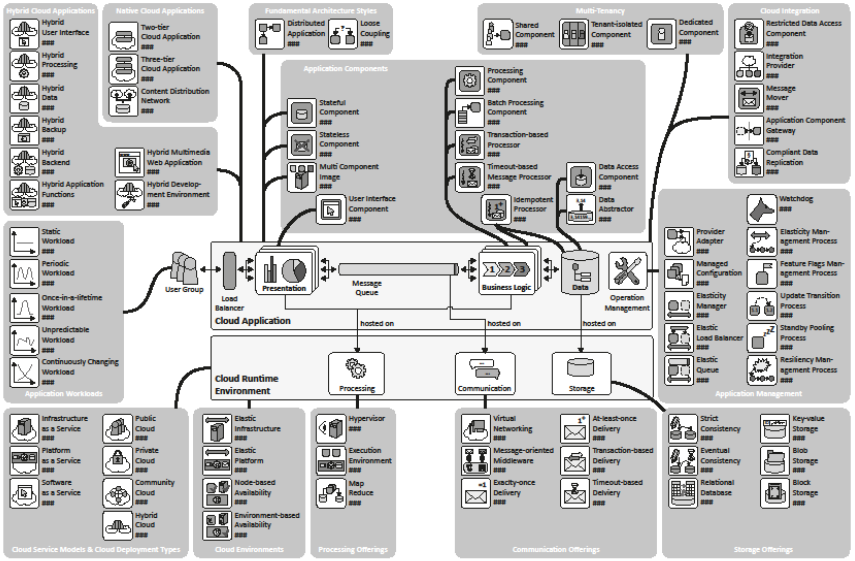
Christoph Fehling (University of Stuttgart, Germany)

Problem
We need to do Cloud Computing!



Cloud Computing Architectural Patterns
A Structured Approach

- Structures the Problem using a Pattern Language
  - Focus: Application Architecture for the cloud



patterns mined at different enterprises by different people!

The Reason

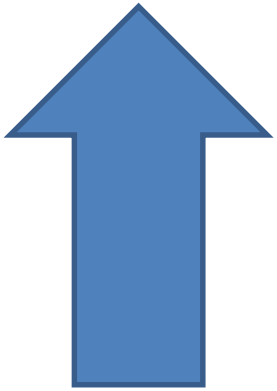
Why did we mine the patterns?

We need to do Cloud!
Typical Resulting Questions

- "Which cloud infrastructure (provider) is the right one for our enterprise?“
- "Is this application suitable for the cloud?"
- "Why isn't it as easy to deploy an application in our data center as it is to deploy a sample application in my favorite public cloud?“

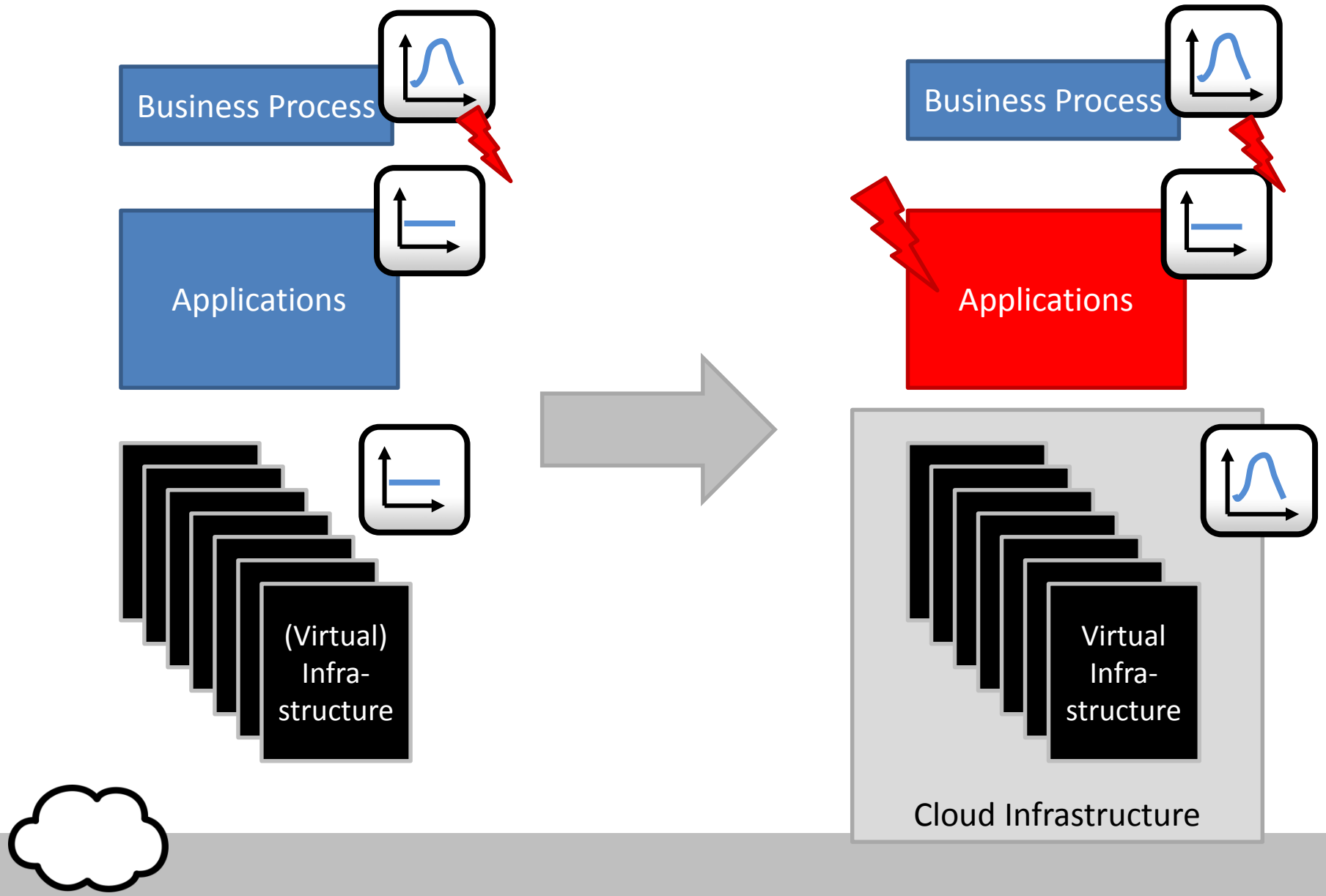
What Happens Next…

- Business Process
- Application
- Platform
- **Infrastructure**

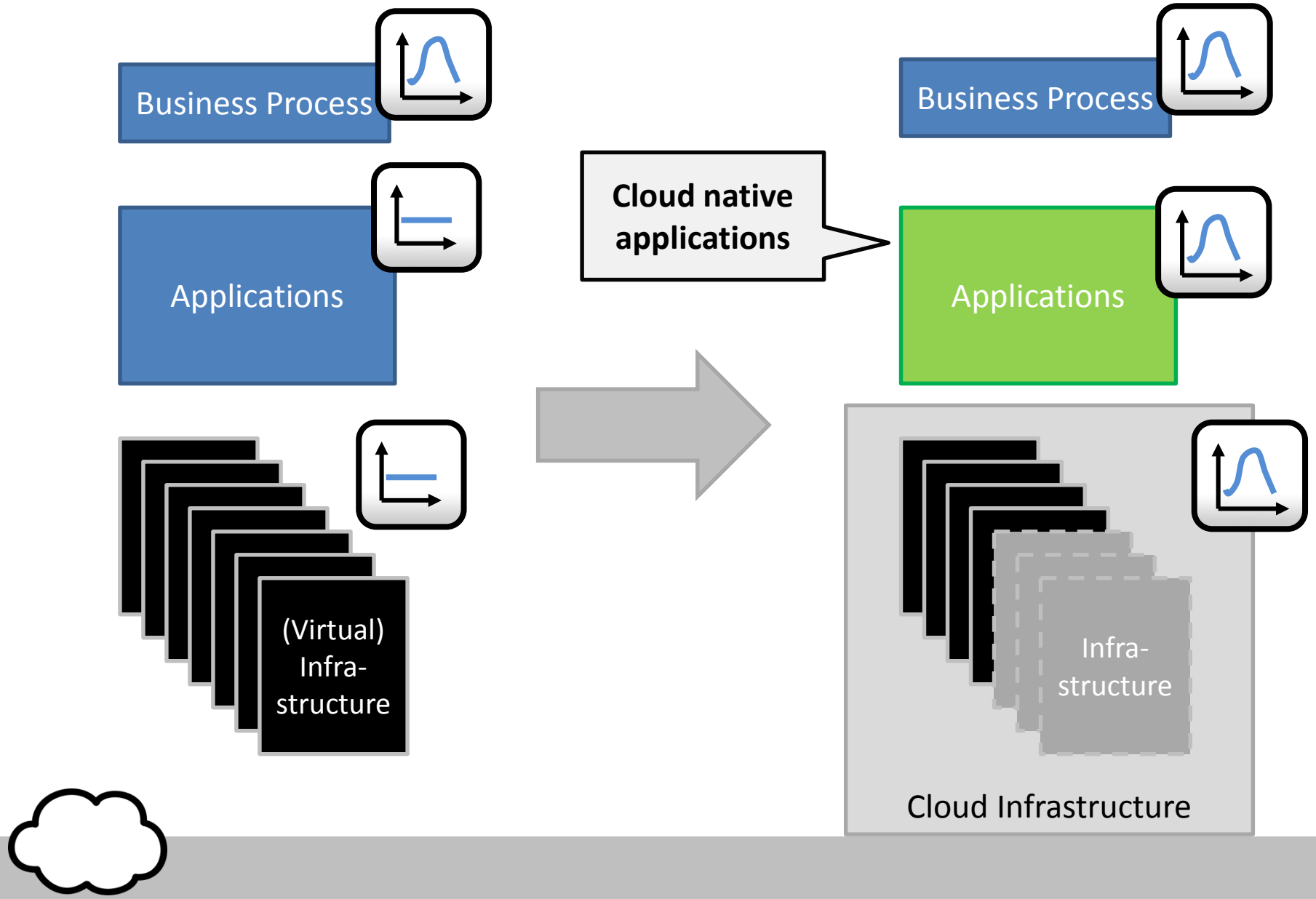


bottom up approach

Typical Result of Bottom Up



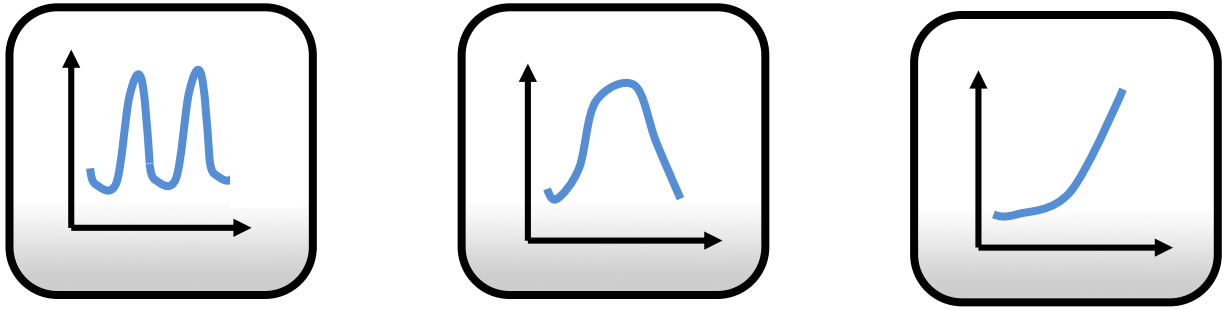
What You Really Want



The Approach

What we learned by mining and applying the patterns, and what they are good for!

Real Requirements != We need to do Cloud!



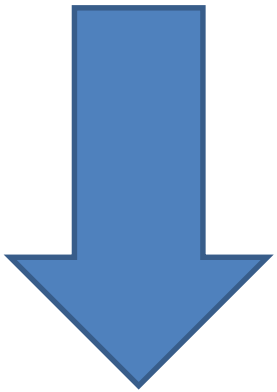
- Example requirements:
  - Deal with dynamic load patterns without provisioning for peak-load
    - ➔ save money!
  - Make application deployments easier and faster …
    - ➔ save time through standardization!
  - …

Better Questions
(Requirement Driven)

1. Which of my business processes benefit from cloud properties of underlying applications?
   - dynamic load patterns
   - Pay-per-use
   - Self service required?
2. Which applications drive these business processes and can they deal with:
   - Resource sharing / pooling
   - Elasticity as a result of requirement for dynamic load patterns and pay per use?
3. What (Cloud) Infrastructure and platforms are needed to support these applications?

Top Down Approach

- **Business Process**
- Application
- Platform
- Infrastructure



top down

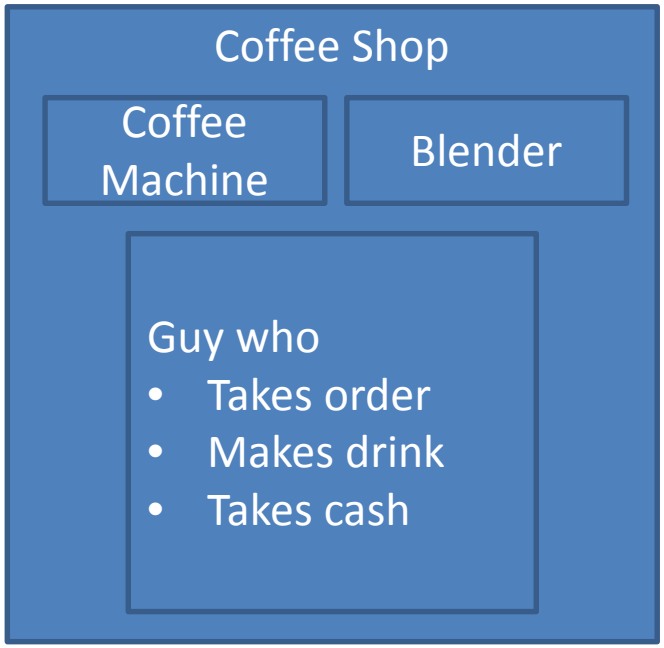
How To Use the Patterns

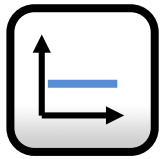
illustrative example - the coffee shop

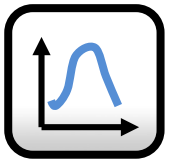
Requirements of Illustrative Example:

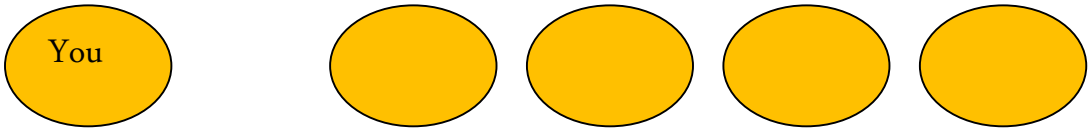
- Functional
  - Make coffee-related specialities and sell them to walk-in customers
- Non-functional requirements
  - Deal with varying amount of simultaneous customers according to the time of the day
  - Maximize order throughput!
  - Keep lines as short as possible

Traditional Small Coffee Shop Architecture

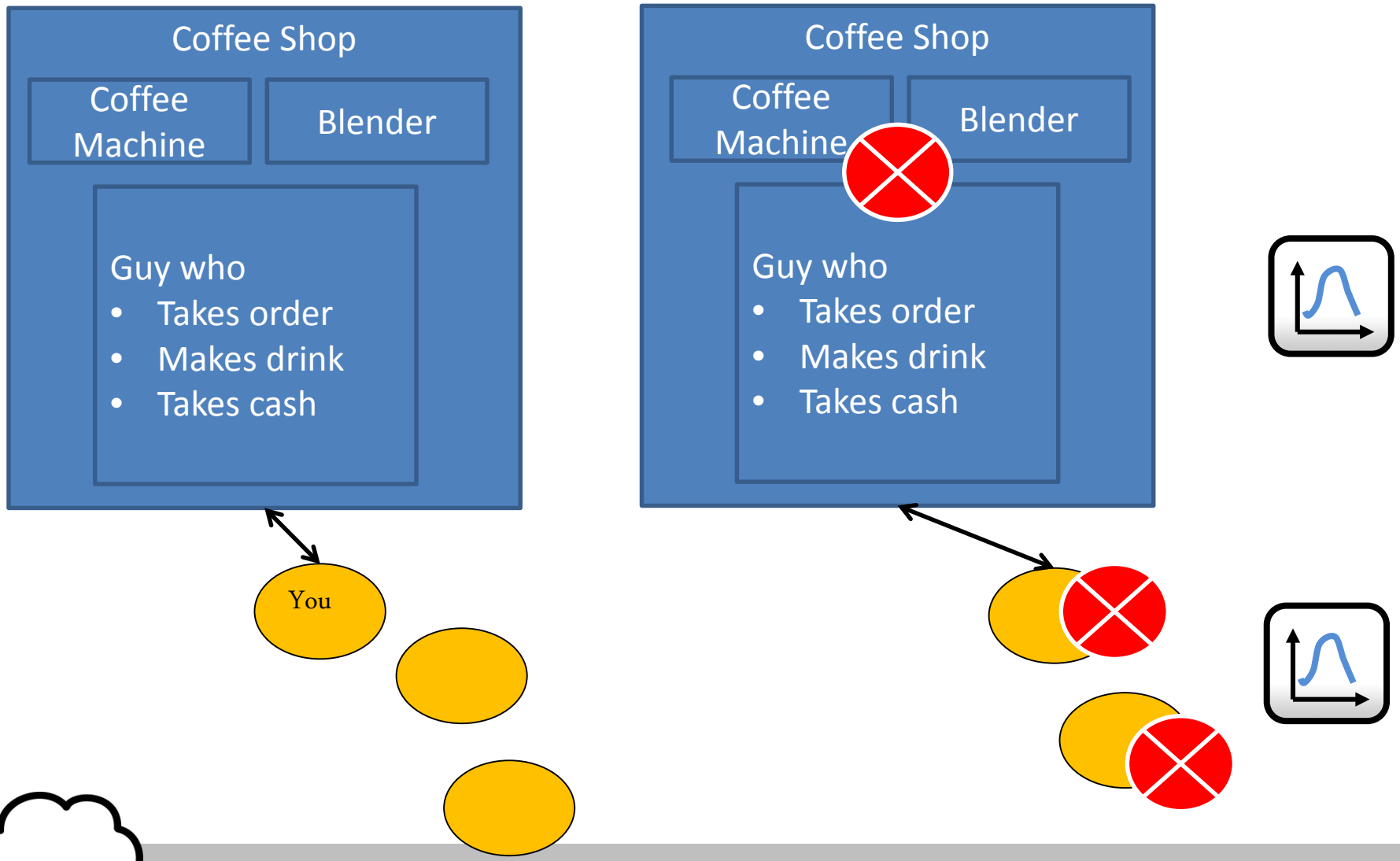




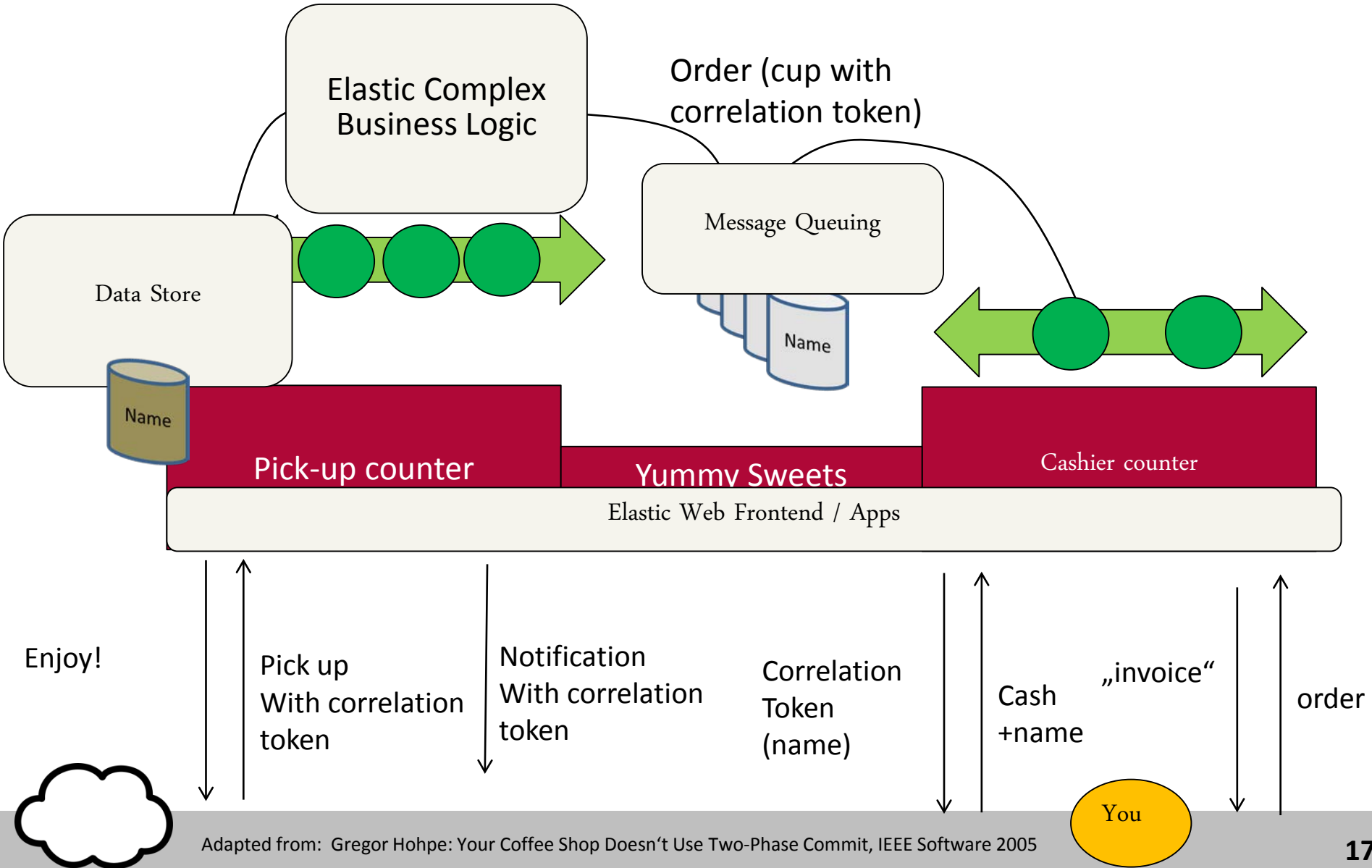




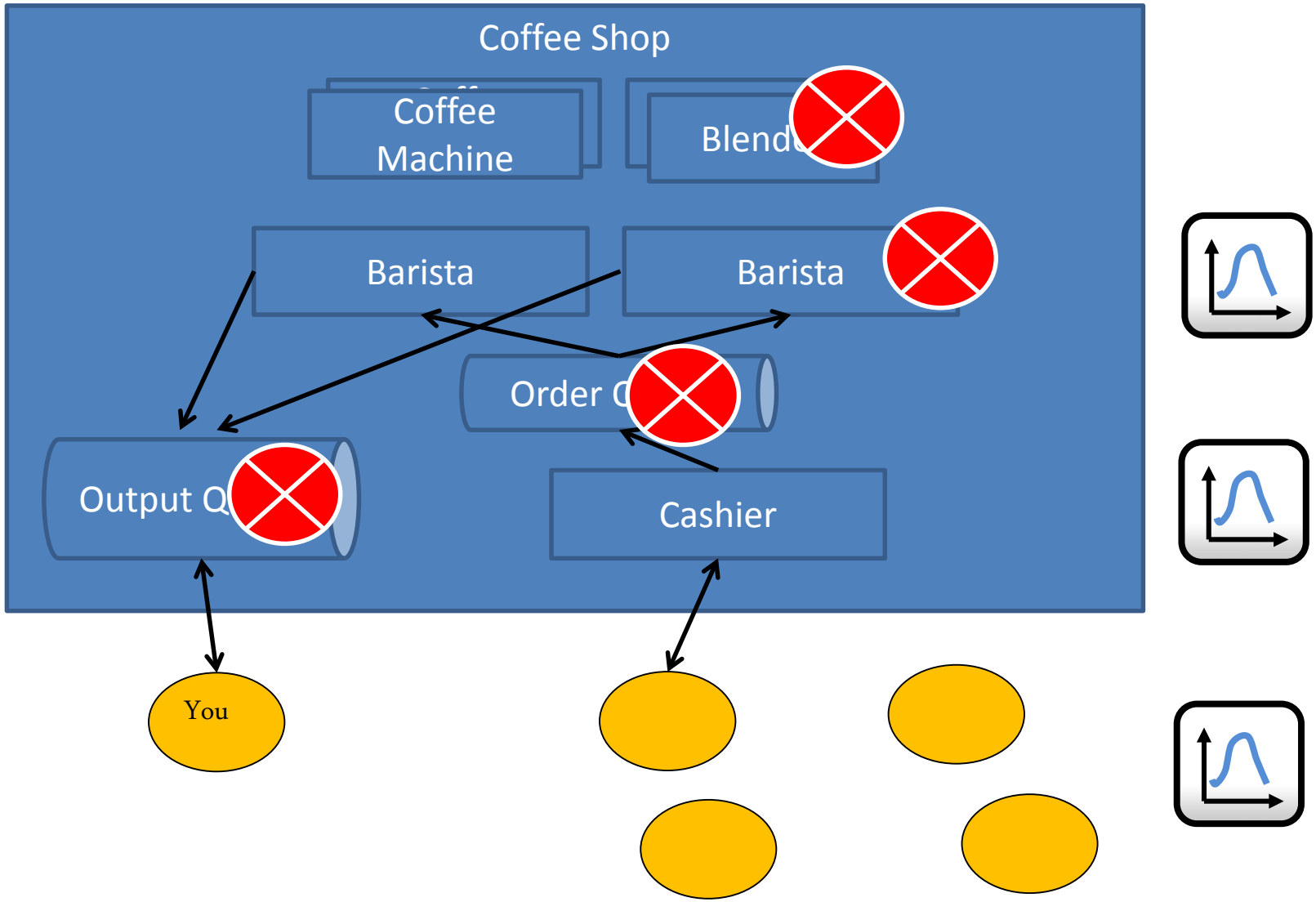
Cloud-Based Bigger Coffee Shop Architecture



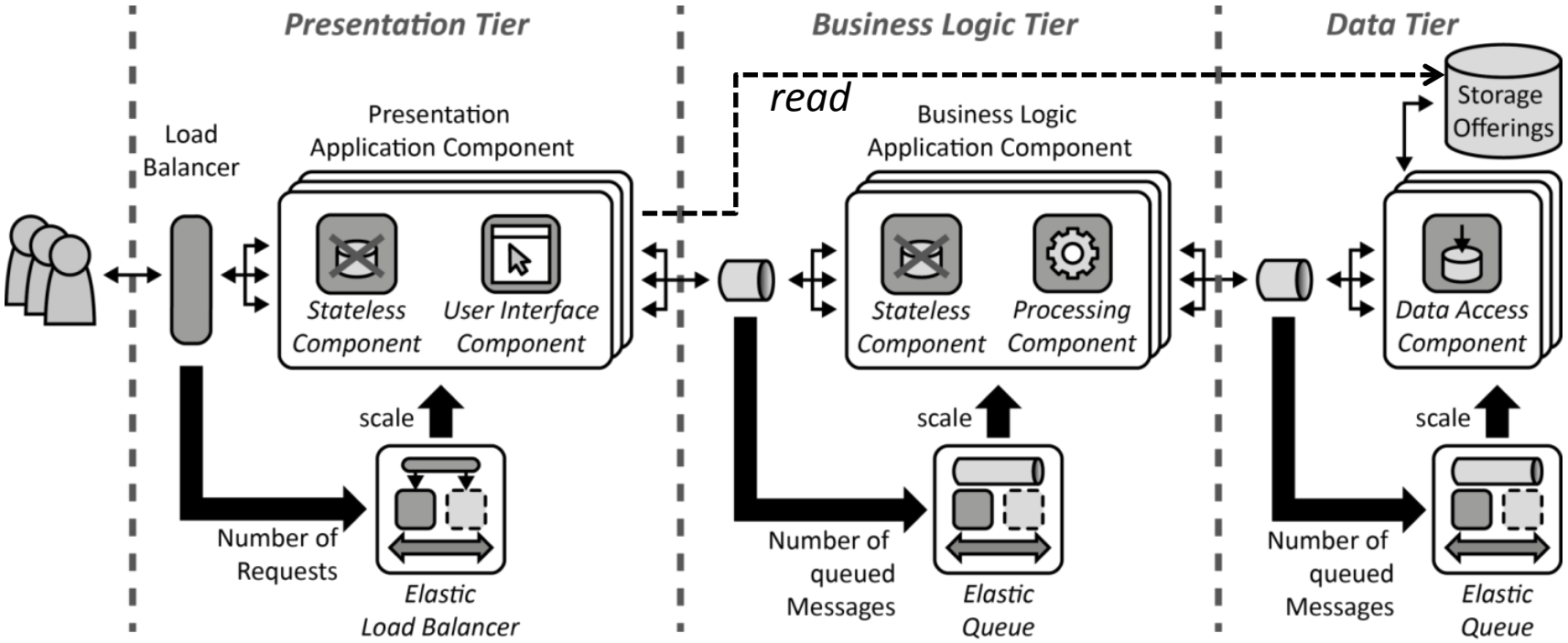
Example: Cloud Native Application
Coffee-Shop Architecture



Cloud-Native Bigger Coffee Shop Architecture



Example Pattern: 3-Tier Cloud Native Application



Required Services @ Cloud Provider

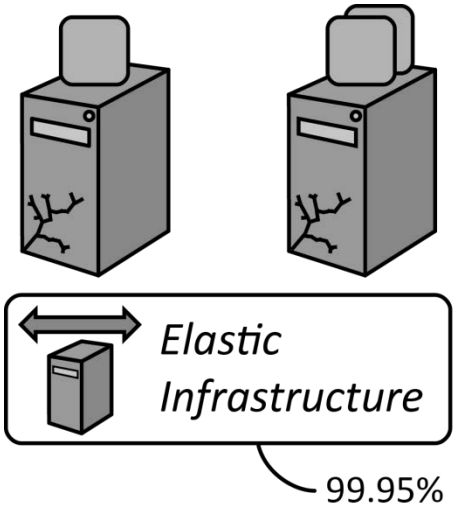
Elastic Infrastructure ###
Elastic Platform ###
Node-based Availability ###
Environment-based Availability ###

Virtual Networking ###
Message-oriented Middleware ###
Exaclty-once Delivery ###
At-least-once Delivery ###
Transaction-based Delivery ###
Timeout-based Delivery ###

Communication Offerings

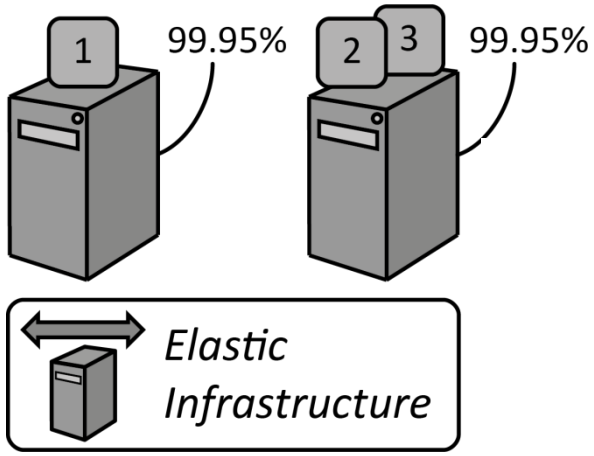
Strict Consistency ###
Eventual Consistency ###
Relational Database ###
Key-value Storage ###
Blob Storage ###
Block Storage ###

Requirements on Cloud Platform / Infrastructure
Requirements resulting from Application Design



Environment-based availability

vs.

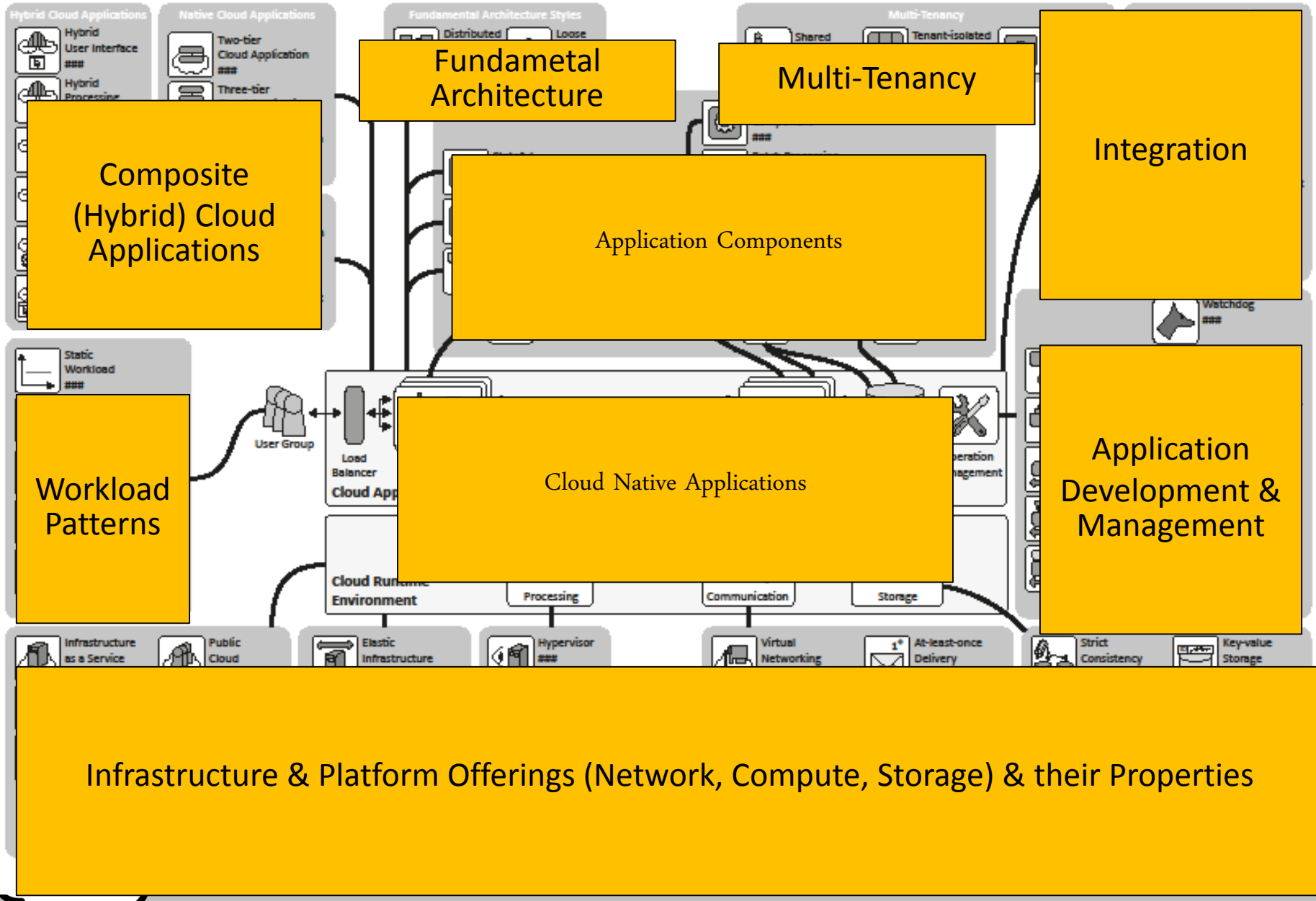


Node-based availability

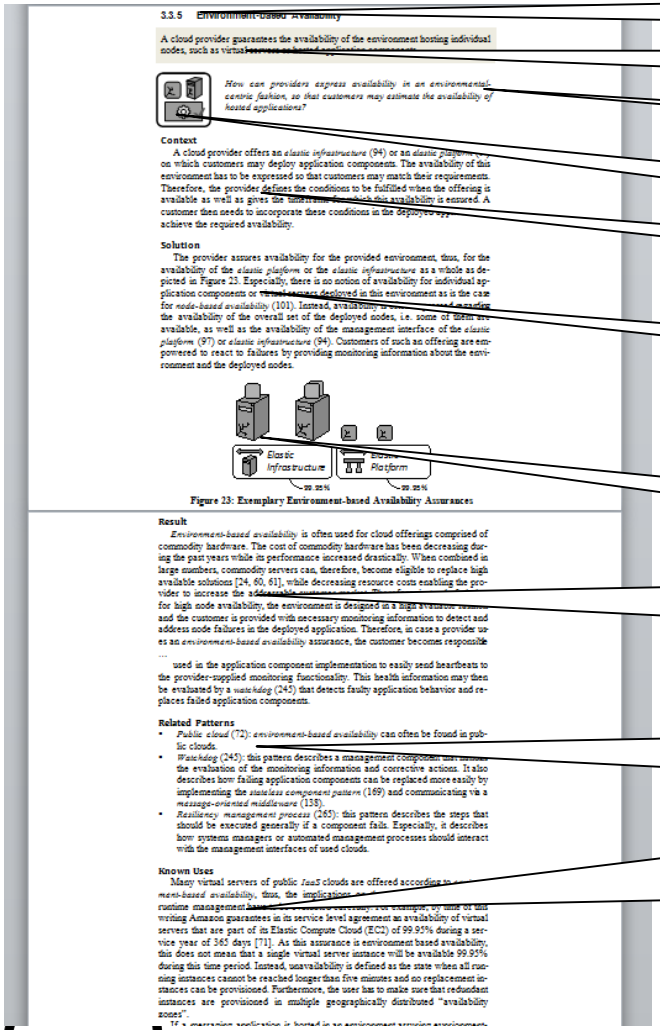
Some More Lessons Learned

- Moving to virtualized Machines is not Cloud! Even if you automate it!
- Consider environment-based availability and dynamic horizontal scaling
  - Use highly available messaging and storage platform offerings
    - Make sure you are aware of the CAP Theorem and it‘s implications
      - Decision to trade towards higher availability and eventual consistency has impact on business processes!
- Modularization!
  - Make sure you can scale different parts of your system according to their workload
  - Seperate short-running transactions with the user from long-running transactions in backend
    - Use asynchronous messaging and compensation-based transaction models in backend
- Resource Sharing!

Top-Down Approach + Pattern Map



Pattern Format



3.3.5 Environment-based Availability

A cloud provider guarantees the availability of the environment hosting individual nodes, such as virtual

How can providers express availability in an environmental-centric fashion, so that customers may estimate the availability of hosted applications?

Context

A cloud provider offers an *elastic infrastructure* (94) or an *elastic platform* on which customers may deploy application components. The availability of this environment has to be expressed so that customers may match their requirements. Therefore, the provider defines the conditions to be fulfilled when the offering is available as well as gives the time frame in which this availability is ensured. A customer then needs to incorporate these conditions in the deployed app to achieve the required availability.

Solution

The provider assures availability for the provided environment, thus, for the availability of the *elastic platform* or the *elastic infrastructure* as a whole as depicted in Figure 23. Especially, there is no notion of availability for individual application components or the instances deployed in this environment as is the case for *node-based availability* (101). Instead, availability is defined as the ratio between the availability of the overall set of the deployed nodes, i.e. some of them are available, as well as the availability of the management interface of the *elastic platform* (97) or *elastic infrastructure* (94). Customers of such an offering are empowered to react to failures by providing monitoring information about the environment and the deployed nodes.

Figure 23: Exemplary Environment-based Availability Assurances

Result

Environment-based availability is often used for cloud offerings comprised of commodity hardware. The cost of commodity hardware has been decreasing during the past years while its performance increased drastically. When combined in large numbers, commodity servers can, therefore, become eligible to replace high available solutions [24, 60, 61], while decreasing resource costs enabling the provider to increase the

for high node availability, the environment is designed in a high available

and the customer is provided with necessary monitoring information to detect and address node failures in the deployed application. Therefore, in case a provider uses an *environment-based availability* assurance, the customer becomes responsible

...

used in the application component implementation to easily send heartbeats to the provider-supplied monitoring functionality. This health information may then be evaluated by a *watchdog* (245) that detects faulty application behavior and replaces failed application components.

Related Patterns

- *Public cloud* (72): *environment-based availability* can often be found in public clouds.
- *Watchdog* (245): this pattern describes a management component for the evaluation of the monitoring information and corrective actions. It also describes how failing application components can be replaced more easily by implementing the *stateless component pattern* (169) and communicating via a *message-oriented middleware* (138).
- *Resiliency management process* (265): this pattern describes the steps that should be executed generally if a component fails. Especially, it describes how systems managers or automated management processes should interact with the management interfaces of used clouds.

Known Uses

Many virtual servers of public *IaaS* clouds are offered according to

ment-based availability, thus, the implications

runtime management

writing Amazon guarantees in its service level agreement an availability of virtual servers that are part of its Elastic Compute Cloud (EC2) of 99.95% during a service year of 365 days [71]. As this assurance is environment based availability, this does not mean that a single virtual server instance will be available 99.95% during this time period. Instead, unavailability is defined as the state when all running instances cannot be reached longer than five minutes and no replacement instances can be provisioned. Furthermore, the user has to make sure that redundant instances are provisioned in multiple geographically distributed "availability zones".

If a

Title = Unique Name

Intent = Purpose and Goal

Driving Question

Icon to use in Diagrams

Context: When is this pattern applicable

Solution: Brief escription how problem is solved

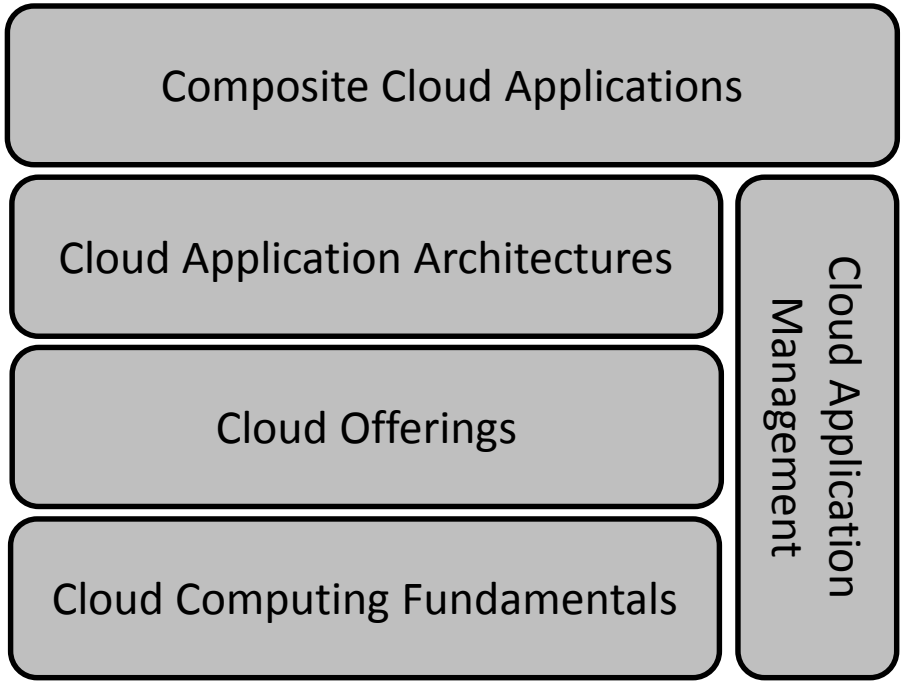
Sketch: Big Picture how Pattern works

Result: Detailed description of solution and its results

Related Patterns: Links to other patterns

Known Uses: Publicly Accessible Services, Solutions, Descriptions… that implement the pattern

Cloud Computing Patterns - Summary



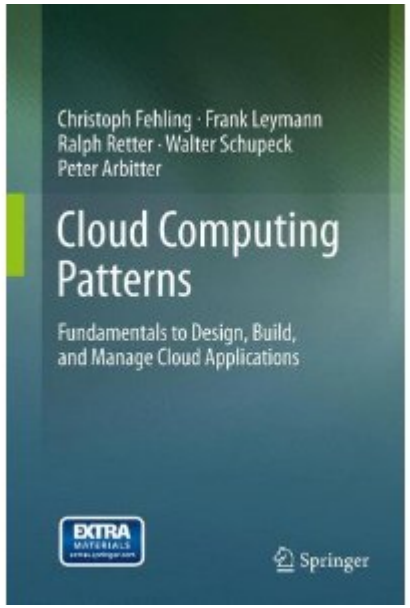
Not all Cloud Computing Patterns are new!

Many existing patterns can be transferred or simply used in the area of cloud computing.

- **Composite Cloud Applications**
  - Common use cases
  - Example Applications
- **Cloud Application Architectures**
  - Building cloud applications
  - Integrating different clouds
- **Cloud Offerings**
  - Processing, storage, and communication functionality
  - Behavior of cloud offerings
  - → Provide runtime functionality
- **Cloud Computing Fundamendals**
  - Cloud Service Models
  - Cloud Types
  - Application Workloads
  - → Characterize the environment
- **Cloud Application Management**
  - Elasticity, resiliency, updates etc.
  - Automation of management

Questions?

http://cloudcomputingpatterns.org



ralph.retter@daimler.com