Cloud Computing Patterns

Fundamentals to Design, Build, and Manage Cloud Applications

Tutorial at SummerSoC 2013 (1 July – 6 July, 2013, Hersonissos, Crete, Greece)

Christoph Fehling, Frank Leymann

Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany

Phone +49-711-685-88 486
Fax +49-711-685-88 472
e-mail Fehling | Leymann
@iaas.uni-stuttgart.de

Patterns are structured text describing abstract problem-solution pairs.

The *Cloud Computing Patterns* describe ***abstract solutions to recurring problems*** in the domain of cloud computing to capture ***timeless knowledge*** that is ***independent of concrete providers***, products, programming languages etc.

Some Patterns…

A Pattern Language

Towns · Buildings · Construction

Christopher Alexander
Sara Ishikawa · Murray Silverstein
with
Max Jacobson · Ingrid Fiksdahl-King
Shlomo Angel

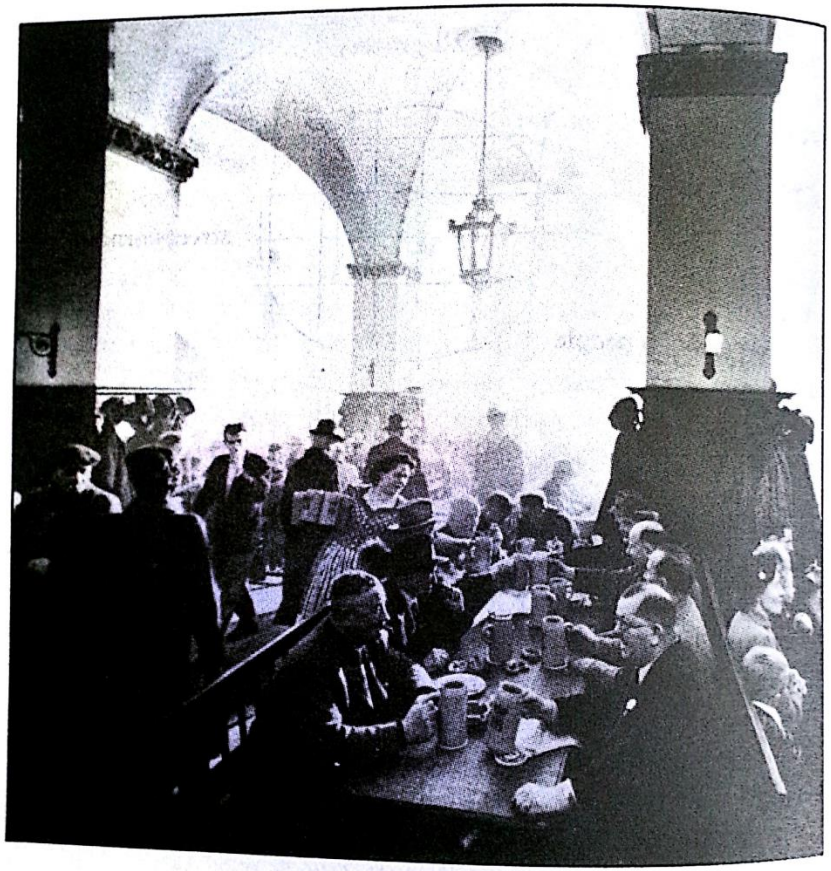
A Pattern Language

Alexander Ishikawa Silverstein Jacobson Fiksdahl-King Angel

2

Oxford

90 BEER HALL



. . . in an occasional neighborhood, which functions as the focus of a group of neighborhoods, or in a boundary between neighborhoods—NEIGHBORHOOD BOUNDARY (15)—or on the promenade which forms the focus of a large community—PROMENADE (31), NIGHT LIFE (33)—there is a special need for something larger and more raucous than a street cafe.

❖ ❖ ❖

Where can people sing, and drink, and shout and drink, and let go of their sorrows?

A public drinking house, where strangers and friends are drinking companions, is a natural part of any large community. But all too often, bars degenerate and become nothing more than anchors for the lonely. Robert Sommer has described this in "Design for Drinking," Chapter 8 of his book *Personal Space*, Englewood Cliffs, N.J.: Prentice-Hall, 1969.

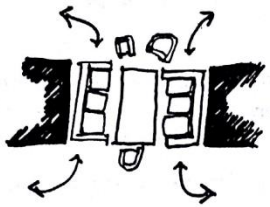
> . . . it is not difficult in any American city to find examples of the bar where meaningful contact is at a minimum. V. S. Pritchett describes the lonely men in New York City sitting speechlessly on a row of barstools, with their arms triangled on the bar before a bottle of beer, their drinking money before them. If anyone speaks to his neighbor under these circumstances, he is likely to receive a suspicious stare for his efforts. The barman is interested in the patrons as customers—he is there to sell, they are there to buy. . . .
>
> Another visiting Englishman makes the same point when he describes the American bar as a "hoked up saloon; the atmosphere is as chilly as the beer . . . when I asked a stranger to have a drink, he looked at me as if I were mad. In England if a guy's a stranger, . . . each guy buys the other a drink. You enjoy each other's company, and everyone is happy. . . ." (Tony Kirby, "Who's Crazy?" *The Village Voice*, January 26, 1967, p. 39.)

Let us consider drinking more in the style of these English pubs. Drink helps people to relax and become open with one another, to sing and dance. But it only brings out these qualities when the setting is right. We think that there are two critical qualities for the setting:

1. The place holds a crowd that is continuously mixing be-

tween functions—the bar, the dance floor, a fire, darts, the bathrooms, the entrance, the seats; and these activities are concentrated and located round the edge so that they generate continual criss-crossing.

2. The seats should be largely in the form of tables for four to eight set in open alcoves—that is, tables that are defined for small groups, with walls, columns, and curtains—but open at both ends.

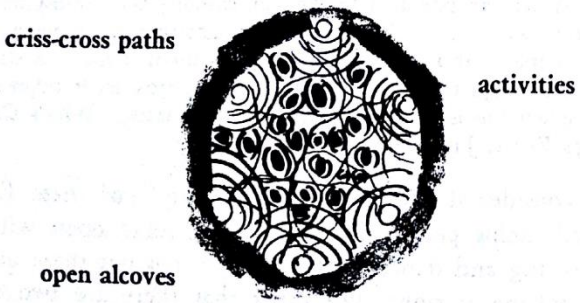


The open alcove—supports the fluidity of the scène.

This form helps sustain the life of the group and lets people come in and out freely. Also, when the tables are large, they invite people to sit down with a stranger or another group.

Therefore:

Somewhere in the community at least one big place where a few hundred people can gather, with beer and wine, music, and perhaps a half-dozen activities, so that people are continuously criss-crossing from one to another.

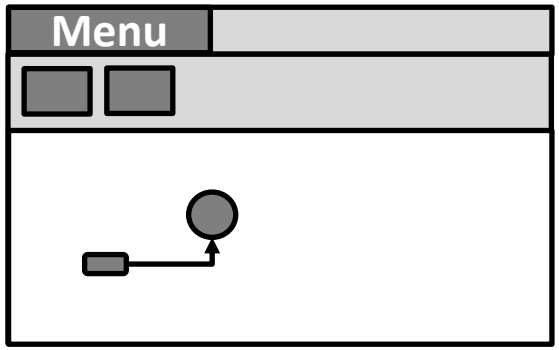


❖ ❖ ❖

Put the tables in two-ended alcoves, roomy enough for people to pass through on their way between activities—ALCOVES (179); provide a fire, as the hub of one activity—THE FIRE (181); and a variety of ceiling heights to correspond to different social groupings—CEILING HEIGHT VARIETY (190). For the shape of the building, gardens, parking, and surroundings, begin with BUILDING COMPLEX (95). . . .

Some existing IT and Cloud Patterns…

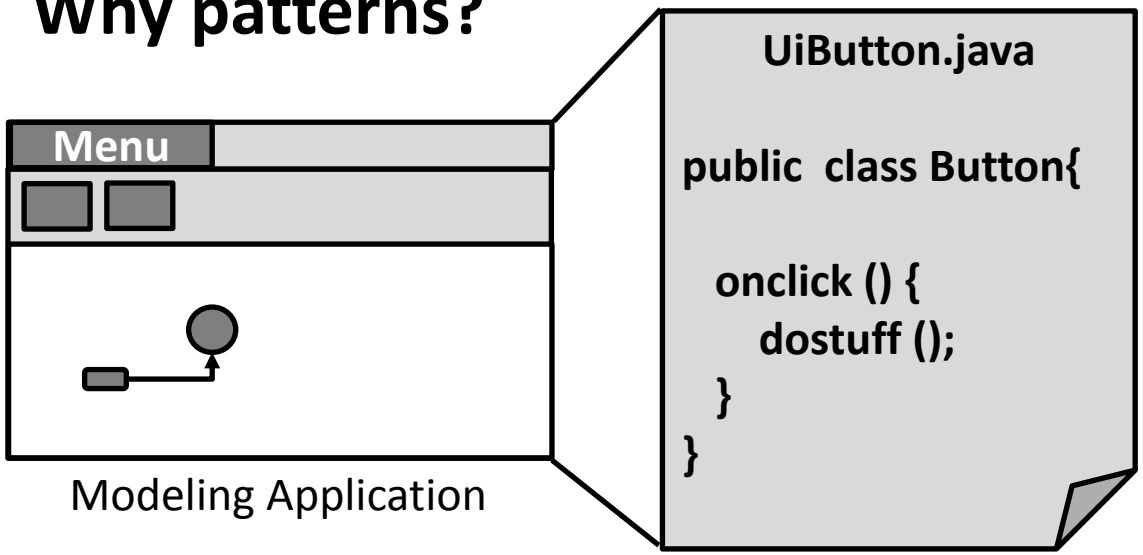
Why patterns?



Modeling Application

- **One of my first projects:** a modeling application
  - Buttons add elements to canvas
  - Elements are interconnected
- **Architecture:** UI – Processing – Data handling layers
  - Abstract interface between UI elements and data layer
  - Graphical elements in canvas have…
    - … logical representation
    - … data representation
- **Problem:** UI button pressed: **how to generate logical + data elements?**

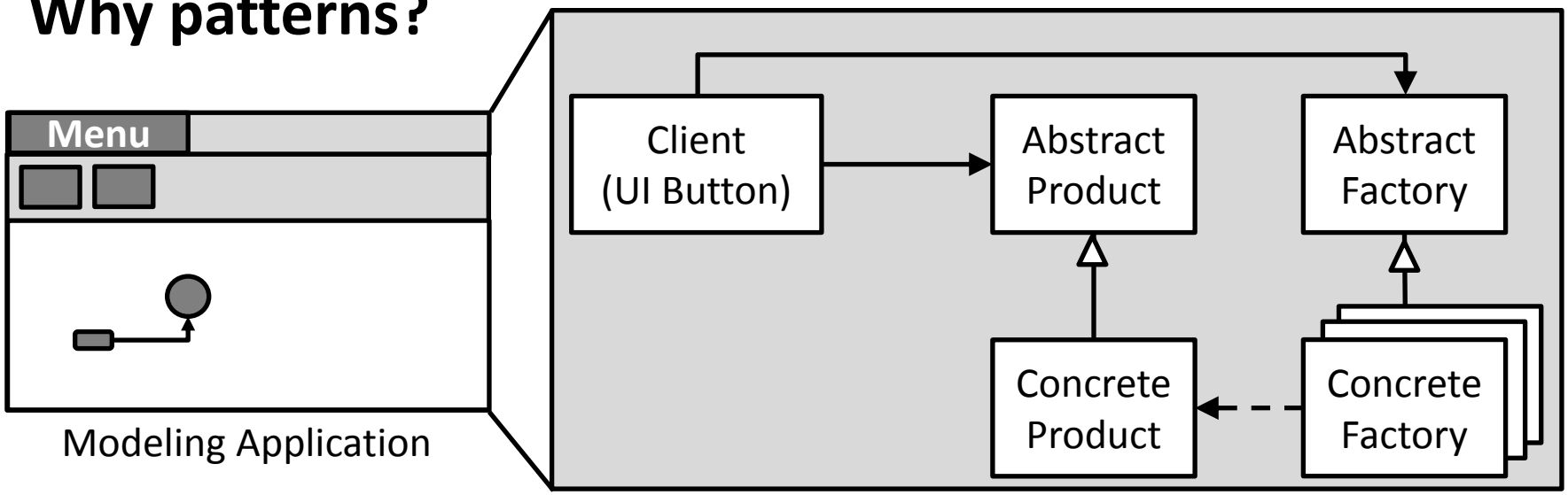
Why patterns?



- **What I wanted:**
  - I need to implement a method on click for each button
  - What code should I write in there?
  - Side note: until then, my curriculum included
    - Writing code
    - Optimizing code
    - Documenting code
    - Making code run in parallel

    → I was a good **builder** of code

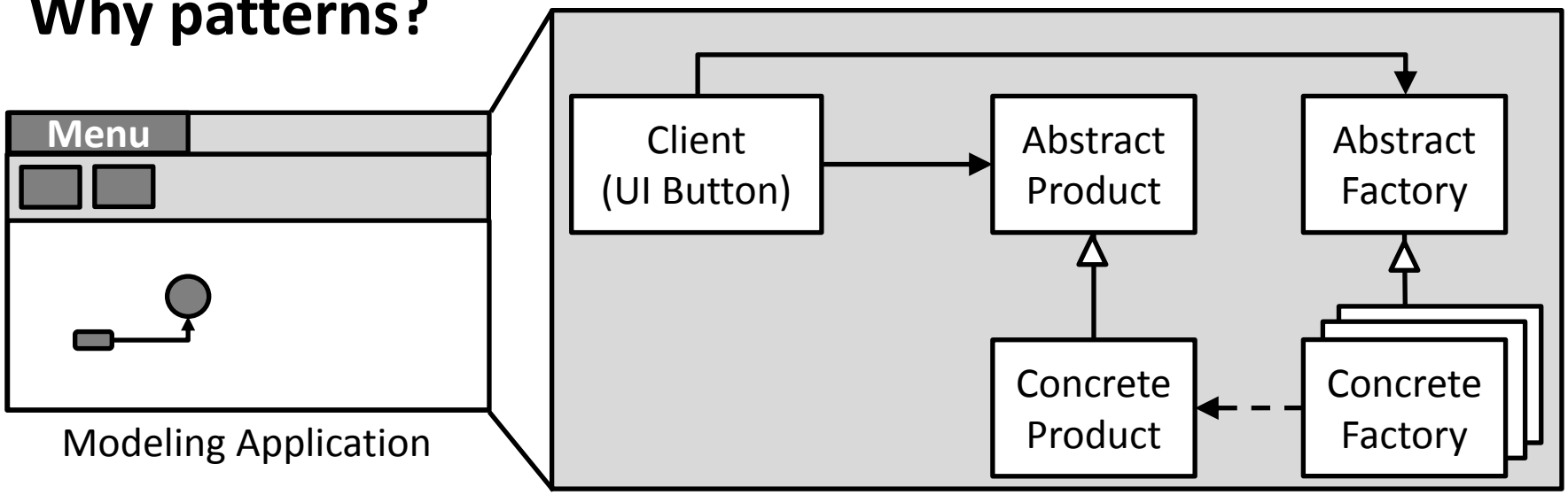
Why patterns?



- **What I was given:**
  - You need to implement an **Abstract Factory Pattern**
  - Client (UI button): accesses Abstract Factory class
  - Concrete factories:
    - extend Abstract Factory
    - create Concrete Products (logic / data elements)
  - Abstract Product: provides interface to client

  → **Clients create concrete products without direct access to them!**

Why patterns?



- **And I was lost (yet amazed)…**
- **Patterns…**
  - … provide abstract solutions to reoccurring problems.
  - … are applicable to various domains.
  - … can be implemented in different programming languages.
  - … make you an **architect** instead of a **builder**
    (imagine building a house with only builders, however well educated…)

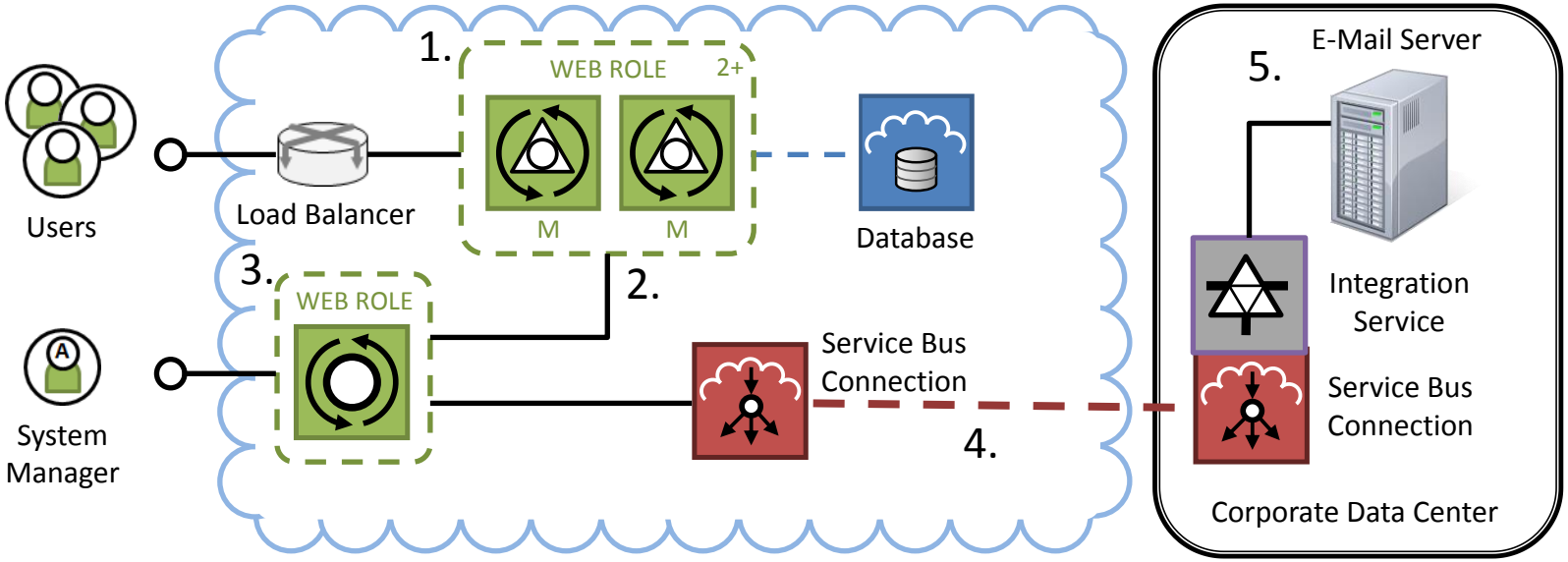
Existing Cloud Computing Patterns?

- Windows Azure Design Patterns & Amazon Architecture Icons
  - http://azuredesignpatterns.com/ (discontinued)
  - http://aws.amazon.com/architecture/icons/
  - … describe available functions
  - … describe conditions when to use these functions
  - … have an icon to create architecture diagrams (new patterns)

→ Let‘s use these patterns to describe a best practice:

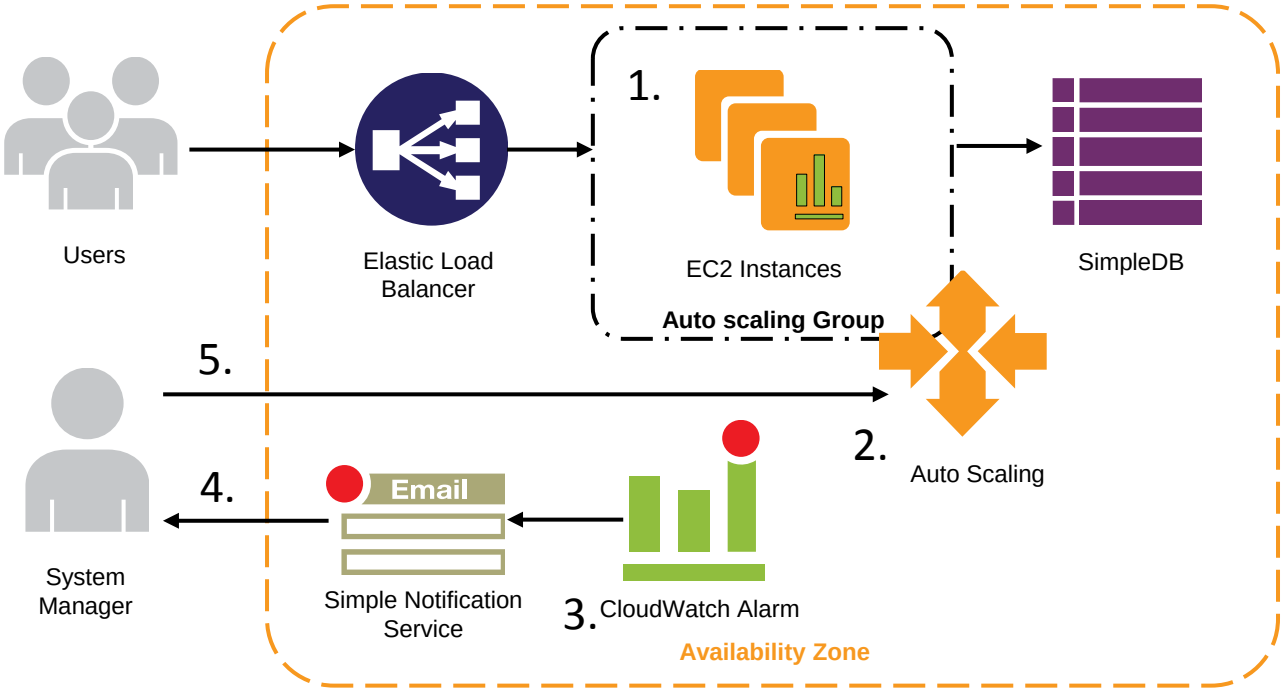
- Small web application: website and backend database
- Load balancing among user interface instances
- Automatic scaling
- Approve scaling beyond a certain threshold via mail

Pattern in Windows Azure



1. User interface is realized by web roles
2. Utilization information is provided by a Web service interface
3. Another Web roles scales the user interface automatically
4. If the threshold is reached, a system manager is contacted via e-mail
5. E-mail is sent trough a corporate e-mail server connected through a service bus

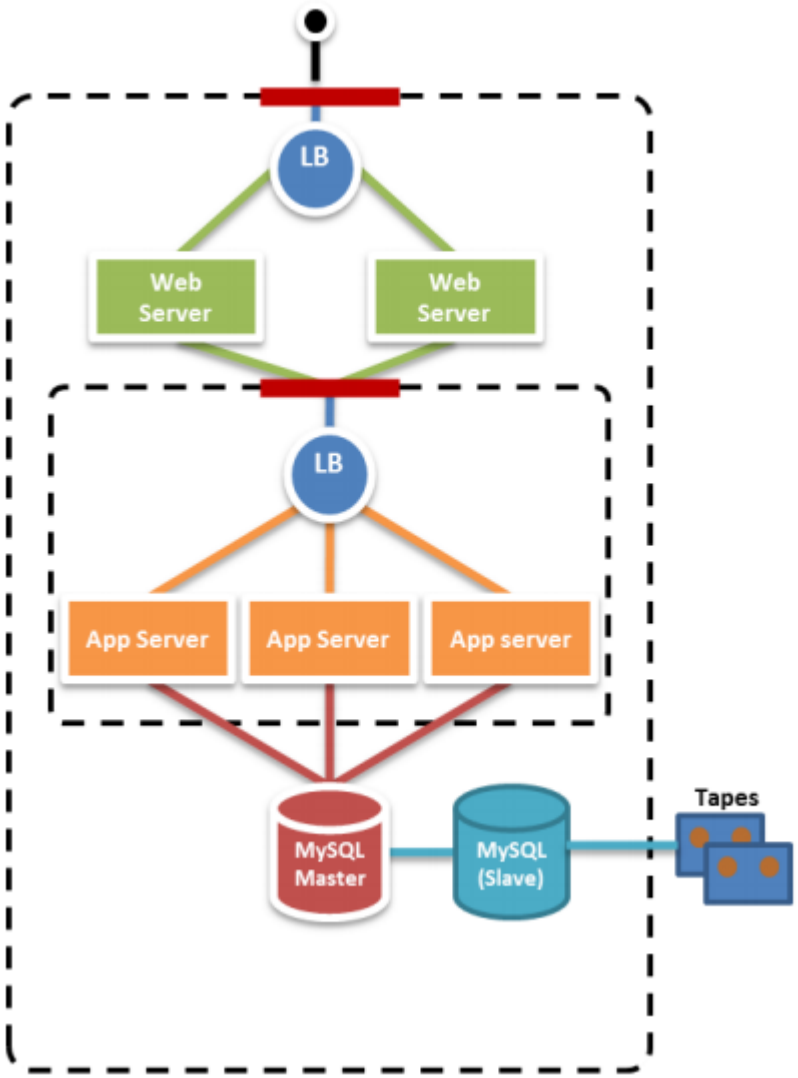
Pattern in Amazon Web Services



1. User interface is realized as EC2 instances
2. The Auto Scaling service scales EC2 instances
3. A Cloud Watch Alarm is triggered if too many EC2 instances are started
4. The system manager is informed via the Simple Notification Service
5. The system manager reconfigures the Auto Scaling Service

Amazon Web Applications (A Pattern?)

Web Application: Before Migration



Application

- Marketing portal
- Product catalog
- Customer Relationship Management

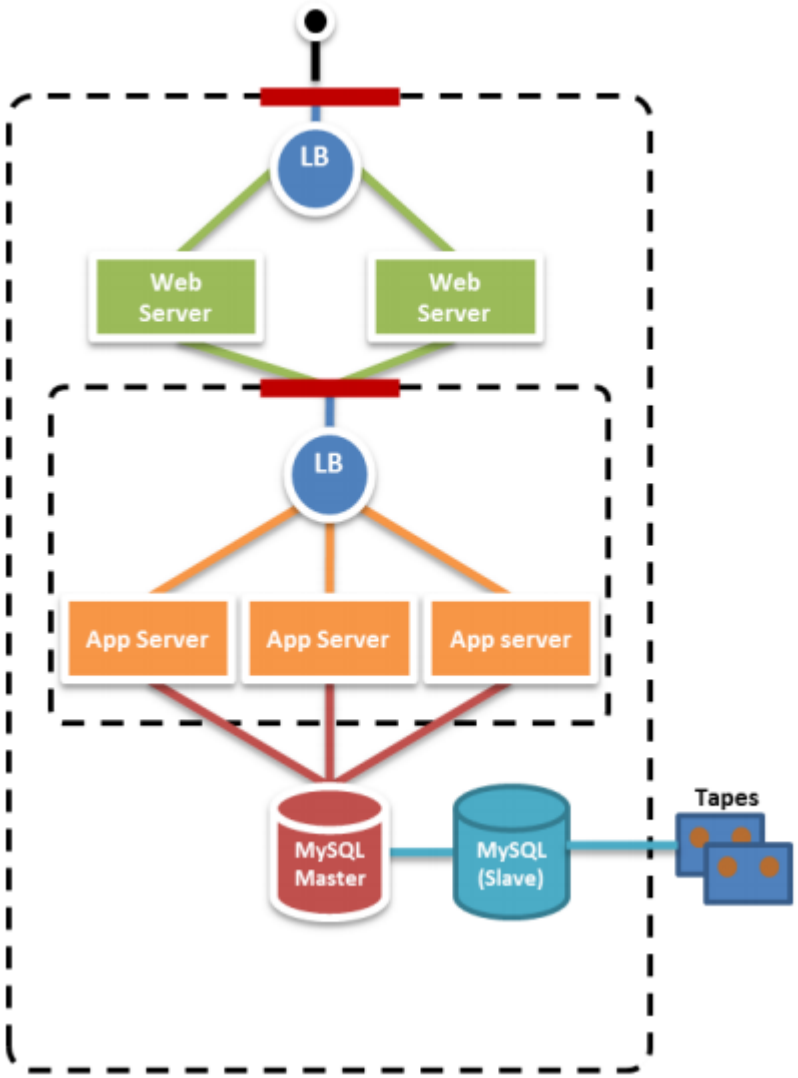
Workload

- High workload on week days
- Low workload on weekends
- Strong peaks during at new product announcements

→ *Periodic workload* and *Once-in-a-lifetime workload*

Source: Amazon White Paper: Migration Scenarios: Web Application Architecture

Web Application: Before Migration

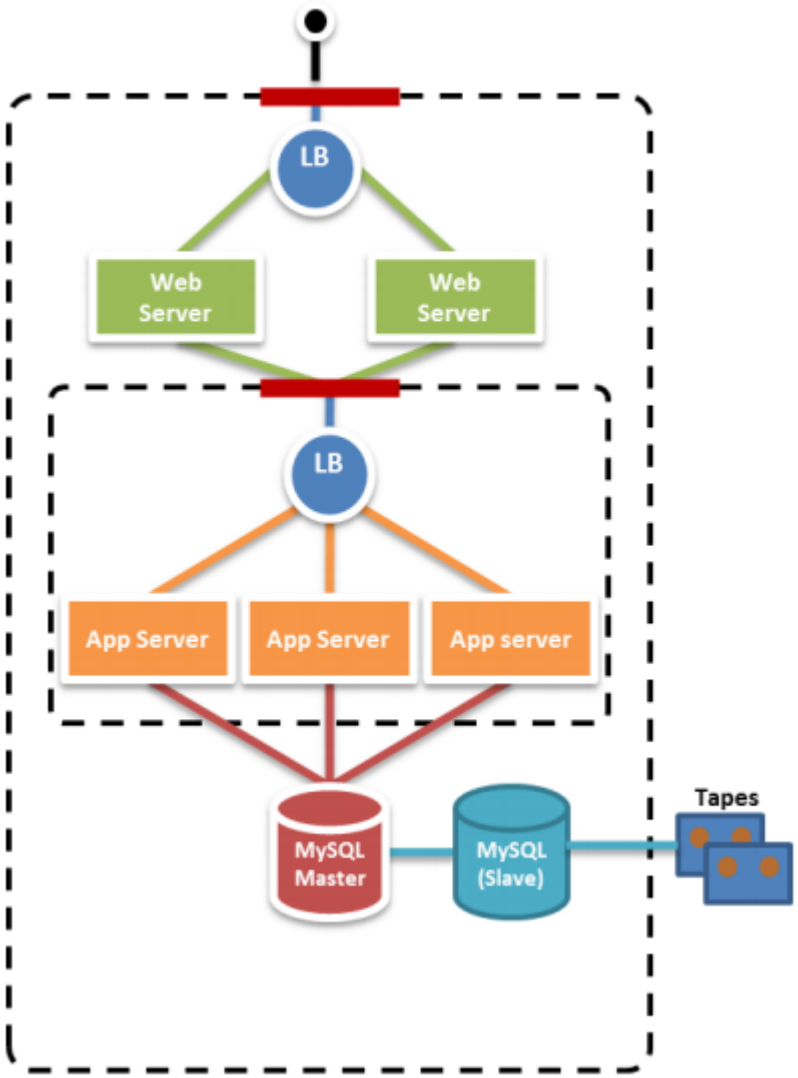


Three-tier Architecture

- **Red Bars:** routing-based access control mechanism
- **LB**: hardware-based load balancer
- **Web Server:** physical server running Apache
- **App Server:** Java application running on Apache Tomcat
- **MySQL Master:** main data storage
- **MySQL Slave:** data replica for performance increase
- **Tapes:** backups stored at the location of a 3rd party

Source: Amazon White Paper: Migration Scenarios: Web Application Architecture

Web Application: Before Migration

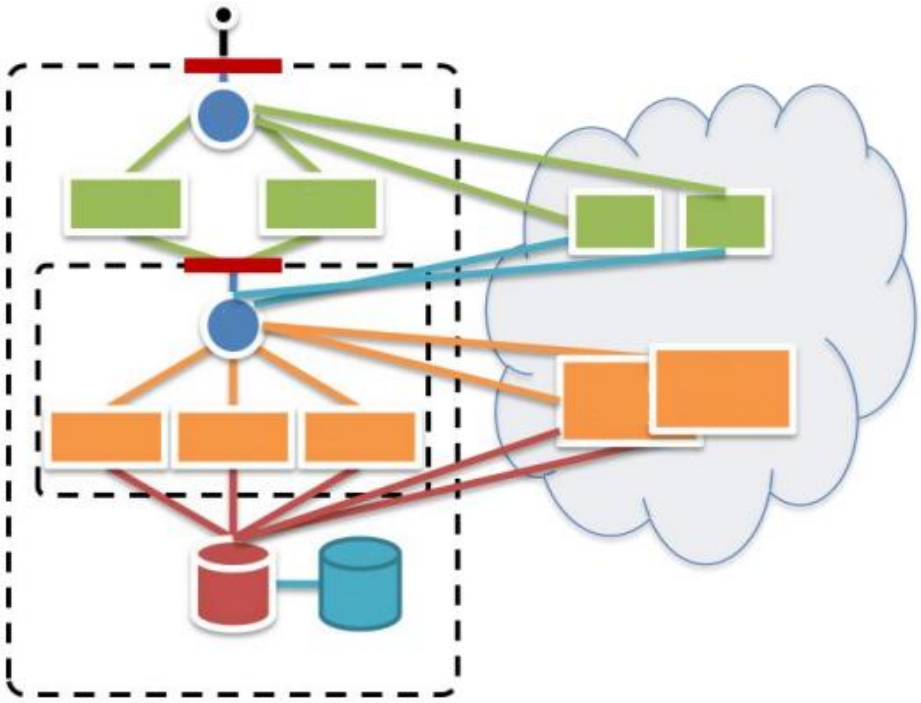


Motivation to migrate

- **Scale out:** handle growing demand without investments in additional hardware
- **Cost reduction:** reduce administrative costs through automation
- **Elasticity:** handle workload peaks at product announcements flexibly

Source: Amazon White Paper: Migration Scenarios: Web Application Architecture

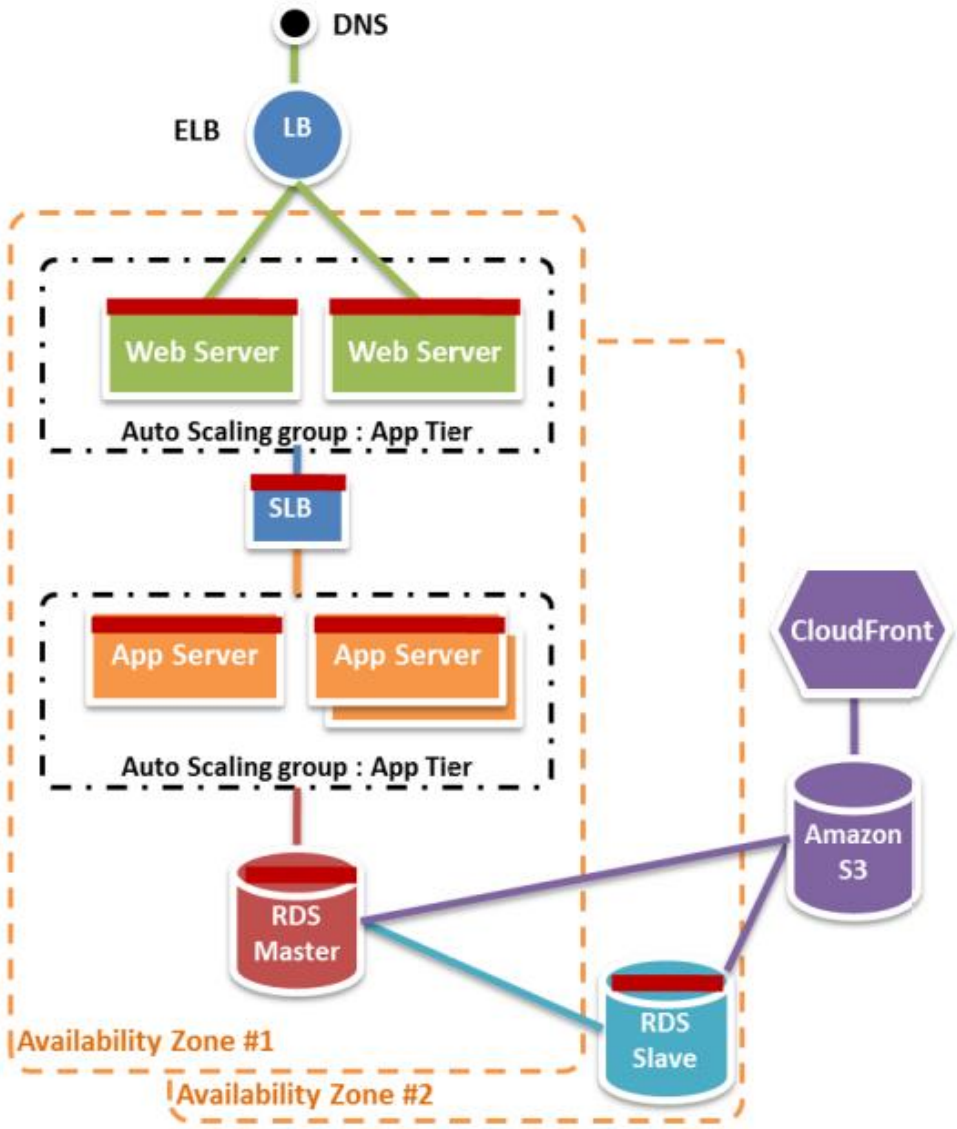
Web Application: Co-existence Phase



1. Additional web servers and application servers were deployed in the cloud
2. Load balancers were reconfigured to send requests to all instances
3. Physical servers where switched off
4. Amazon load balancers were configured
5. DNS entry was switched to IP address of Amazon load balancers
6. Tape backups were used to migrate data to Amazon S3
7. Relational data is moved from S3 to Amazon Relational Data Storage (RDS)

Source: Amazon White Paper: Migration Scenarios: Web Application Architecture

Web Application: After Migration



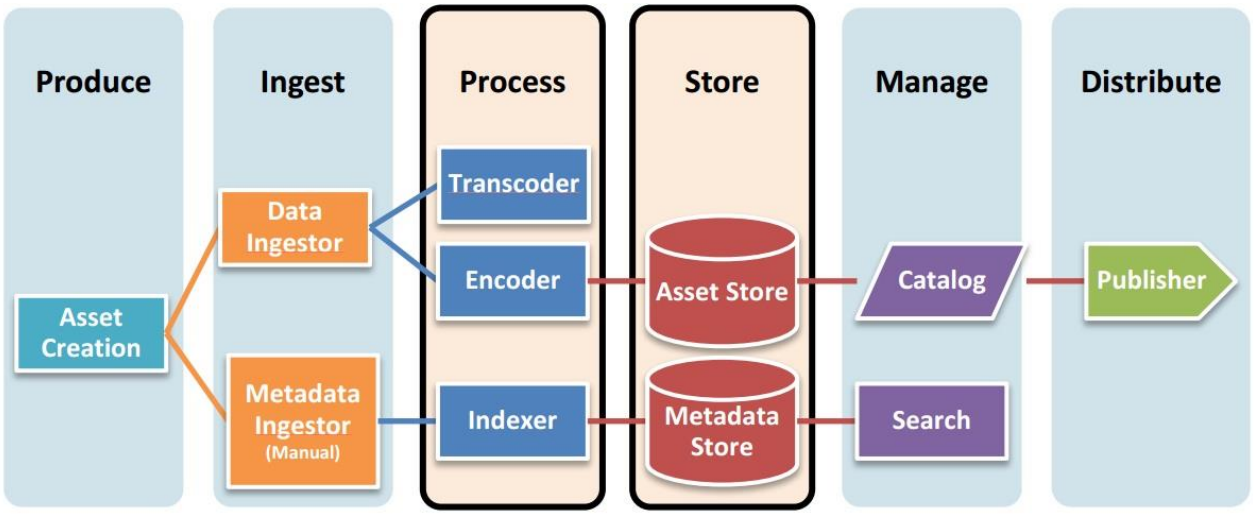
- **LB:** Elastic Load Balancer (ELB) assigning requests
- **SLB:** secure load balancer (with access control)
- **Web Server / App Server:** scaled automatically (within certain boundaries)
- **RDS Master:** relational data storage replacing MySQL
- **RDS Slave:** replica in a different availability zone
- **Amazon S3:** stores large static data and backups
- **CloudFront:** replicates large data globally for access locality (*content distribution network*)

Source: Amazon White Paper: Migration Scenarios: Web Application Architecture

Amazon Migration Scenario
Migrating Batch Processes to the AWS Cloud (Another Pattern?)

Migrating Batch Processes to the AWS Cloud

Use Case: Digital Assets Management (DAM)

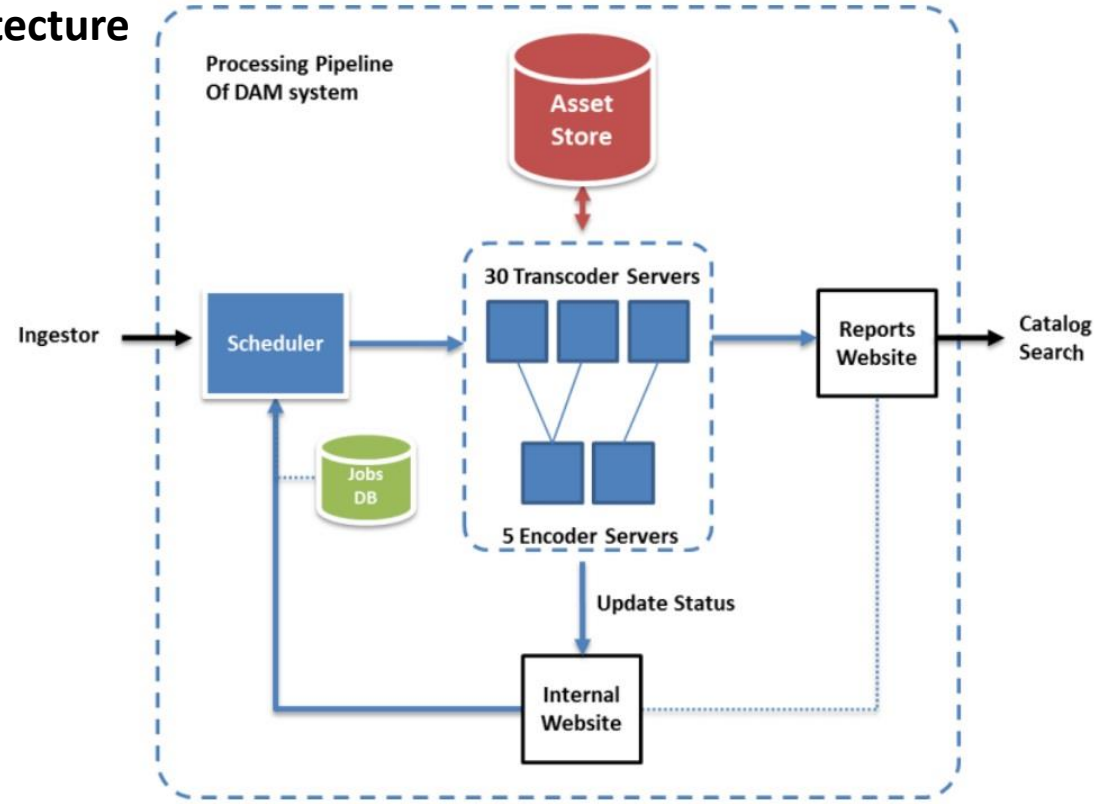


- Manage, process, encode etc. digital assets (videos, pictures, documents…)
  - **Produce**: digital content is created
  - **Ingest**: digital content is inserted in the DAM system and tagged with meta information (for search)
  - **Process**: batch processing of data manipulation and indexing
  - **Store**: contains the assets and meta information in usable format
  - **Manage**: components to offer and search the content
  - **Distribute**: content is bought by publishers.
- System has to handle 300 jobs / night, 1 hour processing time per job on one server

Source: Amazon Web Services White Paper: Migration Scenarios: Batch Processing

Migrating Batch Processes to the AWS Cloud

Original Architecture

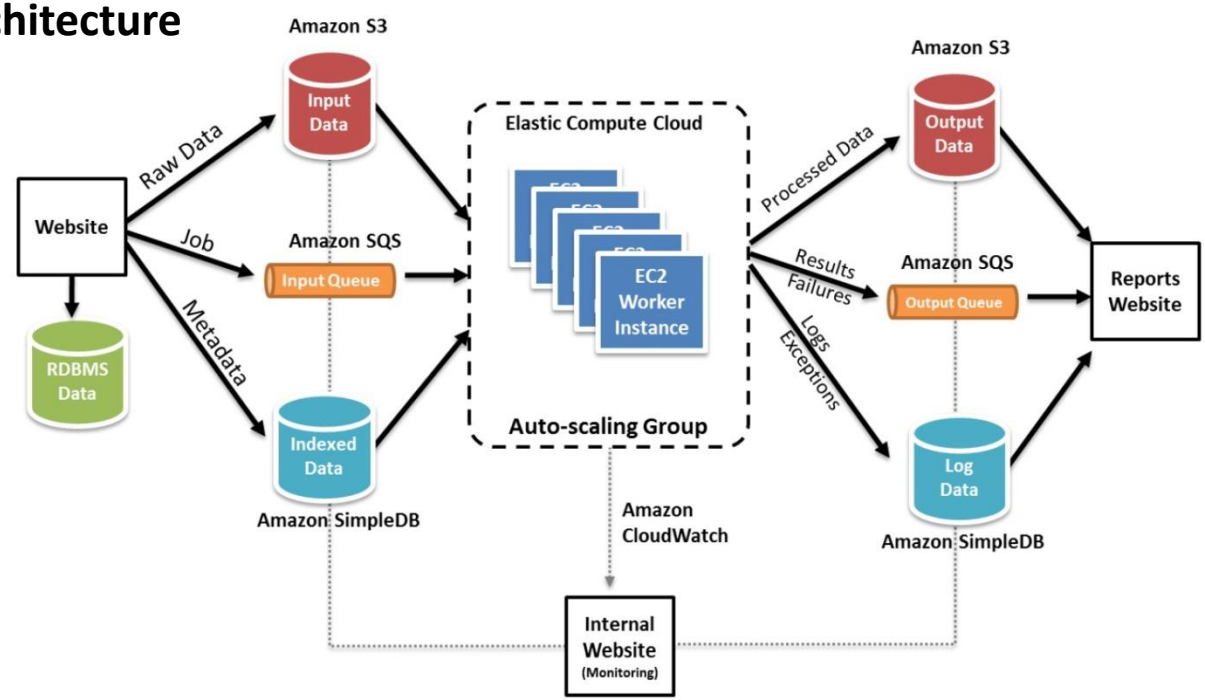


- 30 servers: transcoding and watermarking of media files
- 5 servers: encoding, encryption and DRM
- Scheduler: manages jobs, status and monitors the system
- Reports Website: used to offer search functionality
- Internal Website: accessed by employees to search content etc.
- **Motivation for Migration:** growing demand and limited capital, man power etc. to scale up!

Source: Amazon Web Services White Paper: Migration Scenarios: Batch Processing

Migrating Batch Processes to the AWS Cloud

Original Architecture

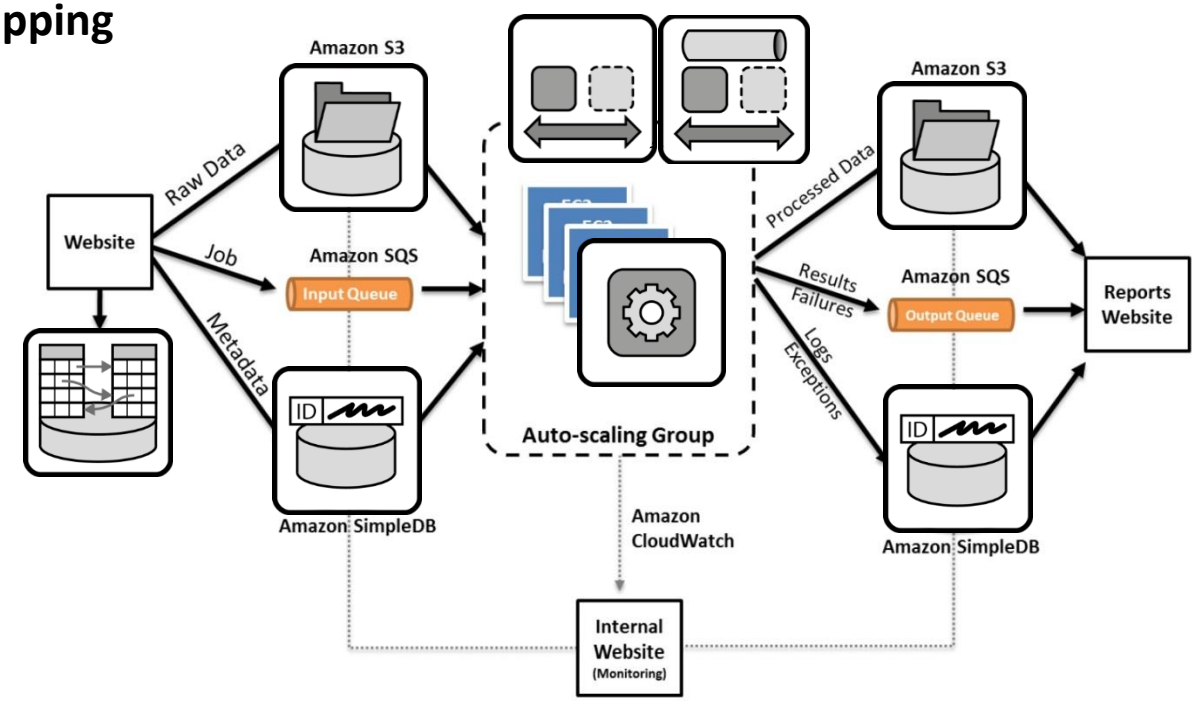


- RDBMS Data: contains small information for the input website
- Amazon S3: handles raw media data to be processed and the result data
- SimpleDB: handles annotated metadata and log information
- EC2 Worker Instance: virtual server handling the actual processing
- Auto-scaling Group: controls number of EC2 Worker Instances
- Amazon SQS: provides queues containing the process requests
- **Can this be mapped to patterns?** ☺

Source: Amazon Web Services White Paper: Migration Scenarios: Batch Processing

Migrating Batch Processes to the AWS Cloud

Pattern Mapping



- RDBMS Data: can be handled by a *Relational Database*
- Amazon SimpleDB: *Key-value Storage*
- Amazon S3: *Blob Storage*
- Auto Scaling Group: *Elasticity Manager* or *Elastic Queue*
  (see Cloud Management Patterns)
- EC2 Worker Instance: *Processing Component*
  (*Batch Processing Component* if media is only processed when the queue is full)

Source: Amazon Web Services White Paper: Migration Scenarios: Batch Processing

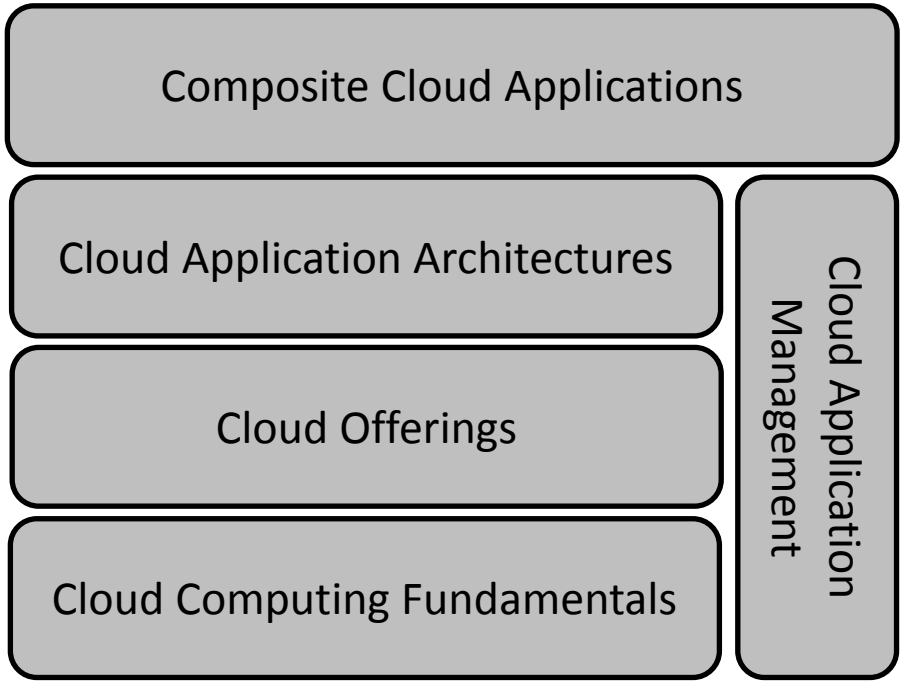
Problems of Provider-specific Patterns

- Provider services are considered by patterns
  - Each provider has different services
  - Naming of services differ even if functionality is similar
  - Set of services differ
    (more or less customer-implemented functions)
- Levels of abstraction differ
  - Infrastructure as a Service: servers, instances
  - Platform as a Service: roles etc.

We need provider-independent abstract patterns!

Cloud Computing Patterns

Cloud Computing Patterns



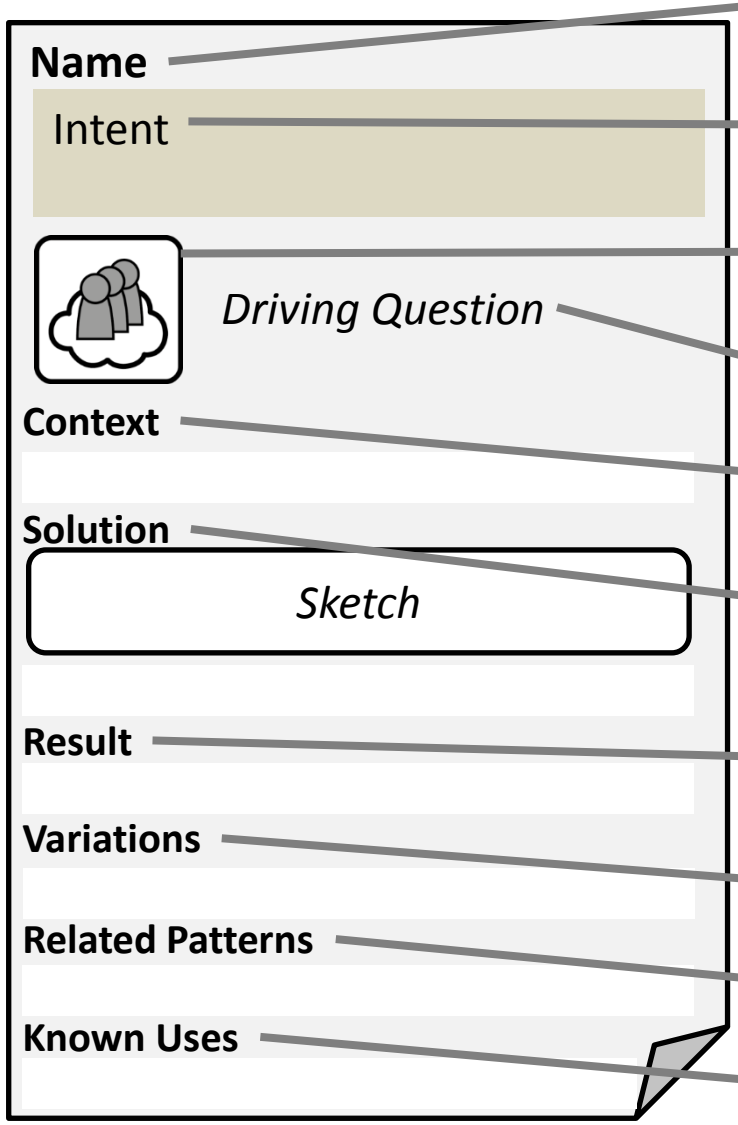
Not all Cloud Computing Patterns are new!

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

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

Pattern Format

Pattern Format



Unique identifier.
Followed by „(page number)“ in book.

Summarizes the purpose of the pattern.

Identifies pattern and is used in diagrams / sketches of other patterns.

Captures the problem answered by the pattern.

Setting in which the pattern can be applied.

Brief statement how the problem is solved supported by a sketch.

Detailed discussion of the solution and new challenges.

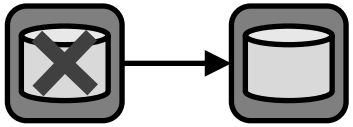
Alternative ways to apply the pattern.

Patterns that are used together, used instead etc.

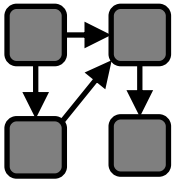
Discussion where the pattern has been applied.

Cloud Application Properties

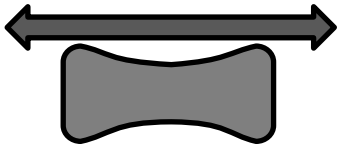
IDEAL Cloud Application Properties



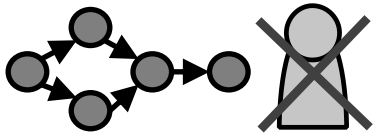
Isolated State: most of the application is *stateless* with respect to:
Session State: state of the communication with the application
Application State: data handled by the application



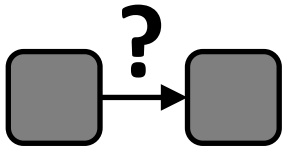
Distribution: applications are decomposed to…
… use multiple cloud resources
… support the fact that clouds are large globally distributed systems



Elasticity: applications can be scaled out dynamically
Scale out: performance increase through addition of resources
Scale up: performance increase by increasing resource capabilities



Automated Management: runtime tasks have to be handled quickly
Example: exploitation of pay-per-use by changing resource numbers
Example: resiliency by reacting to resource failures



Loose Coupling: influence of application components is limited
Example: failures should not impact other components
Example: addition / removal of components is simplified