Pattern-based Deployment Models Revisited: Automated Pattern-driven Deployment Configuration

Lukas Harzenet1, Uwe Breitenbücher, Michael Falkenthal, Jasmin Guth, and Frank Leymann

1 lukas.harzenetter@iaas.uni-stuttgart.de

Institute of Architecture of Application Systems
About me: Lukas Harzenetter

- Research associate at the University of Stuttgart, Institute of Architecture of Application Systems (IAAS)
- Master of Science in Software Engineering, 2018
- Funded by the German Research Foundation (DFG) project SustainLife
  - Enhancing the sustainability and longevity of research applications in the domain of digital humanities
- Research interests:
  - Cloud Computing
  - Deployment Automation
  - Deployment Models
Deployment and Configuration Automation

- Ansible
- Amazon Web Services
- Docker
- Juju
- OpenStack
- Bash Shell
- Azure
- CFEngine
- Puppet Labs
- Cloud Foundry
- Rackspace
- Google Cloud
- HashiCorp Terraform
- VMware
- Heat
Example Deployment Model

Order App (Java 8 Web App)
- Port: 80
- Order App (Java 8 Web App)

Order PaaS (AWS Webserver Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min
- PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

Queue (SQS Queue)
- Type: FIFO
- Server-Side Encryption: Yes
- Master-Key: Default
- MoM Provider (Simple Queue Service)
- Account: ust-iaas

Order Processor (Java 8 App)
- MaxHeap: 6GB
- Processor PaaS (AWS Worker Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

Order Processor (Java 8 App)
- MaxHeap: 6GB
- Processor PaaS (AWS Worker Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

Database (MySQL Database 5.7)
- DB-Name: webshop
- DBMS (MySQL DBMS 5.7)
- Server-Side Encryption: Yes
- Master-Key: Default

PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas
- PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

DBMS Provider (Relational Database Service)
- Account: ust-iaas

MoM Provider (Simple Queue Service)
- Account: ust-iaas
- Public Cloud (AWS)
- Region: EU
- Provider lock-in
- = JmsQueueCon_1.1
- = SQL-Con
- = hostedOn

Requires immense expertise to create...

Technology lock-ins
Patterns

- Pattern: Proven concept & solution to a recurring problem
  - Context & problem description
  - Abstract solution
  - Icon

- Domain specific:
  - Cloud Computing Patterns by Fehling et al. [1]
  - Enterprise Integration Patterns by Hohpe and Woolf [2]
  - Security Patterns by Schumacher et al. [3]

(2) G. Hohpe and B. Woolf: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004

Public Cloud [1]

Relational Database [1]
Example Deployment Model

- **Order App** (Java 8 Web App)
  - Port: 80
  - [...]

- **Order PaaS** (AWS Webserver Env)
  - AutoScaling: true
  - ScalingMetric: NetOut
  - TimeSpan: 5min
  - [...]

- **PaaS Provider** (Elastic Beanstalk)
  - Account: ust-iaas
  - [...]

- **Queue** (SQS Queue)
  - Type: FIFO
  - Server-Side Encryption: Yes
  - Master-Key: Default

- **Order Processor** (Java 8 App)
  - MaxHeap: 6GB
  - [...]

- **Processor PaaS** (AWS Worker Env)
  - AutoScaling: true
  - ScalingMetric: NetOut
  - TimeSpan: 5min
  - [...]

- **PaaS Provider** (Elastic Beanstalk)
  - Account: ust-iaas
  - [...]

- **Database** (MySQL Database 5.7)
  - DB-Name: webshop
  - [...]

- **DBMS** (MySQL DBMS 5.7)
  - Server-Side Encryption: Yes
  - Master-Key: Default
  - [...]

- **DBMS Provider** (Relational Database Service)
  - Account: ust-iaas
  - [...]

- **MoM Provider** (Simple Queue Service)
  - Account: ust-iaas
  - [...]

- **Public Cloud** (AWS)
  - Region: EU
  - [...]

- **= JmsQueueCon_1.1**
- **= SQL-Con**
- **= hostedOn**
Example Deployment Model

1. **Order App** (Java 8 Web App)
   - Port: 80

2. **Order PaaS** (AWS Webserver Env)
   - AutoScaling: true
   - ScalingMetric: NetOut
   - TimeSpan: 5min

3. **Order Processor** (Java 8 App)
   - MaxHeap: 6GB

4. **Queue** (SQS Queue)
   - Type: FIFO
   - Server-Side Encryption: Yes
   - Master-Key: Default

5. **Processor PaaS** (AWS Worker Env)
   - AutoScaling: true
   - ScalingMetric: NetOut
   - TimeSpan: 5min

6. **DB**
   - Name: webshop
   - DB-Name: webshop

7. **Order PaaS Provider** (Elastic Beanstalk)
   - Account: ust-iaas

8. **MoM Provider** (Simple Queue Service)
   - Account: ust-iaas

9. **PaaS Provider** (Elastic Beanstalk)
   - Account: ust-iaas

10. **DBMS Provider** (Relational Database Service)
    - Account: ust-iaas

11. **DBMS** (MySQL Database 5.7)
    - Server-Side Encryption: Yes
    - Master-Key: Default

12. **DBMS Provider** (Relational Database Service)
    - Account: ust-iaas

Public Cloud

- **= JmsQueueCon_1.1**
- **= SQL-Con**
- **= hostedOn**
Order App (Java 8 Web App)
Port: 80
...

Queue (SQS Queue)
Type: FIFO
Server-Side Encryption: Yes
Master-Key: Default

Order Processor (Java 8 App)
MaxHeap: 6GB
...

Processor PaaS (AWS Worker Env)
AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min
...

DBMS (MySQL DBMS 5.7)
AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min
...

PaaS Provider (Elastic Beanstalk)
Account: ust-iaas
...

DBMS Provider (Relational Database Service)
Account: ust-iaas
...

MoM Provider (Simple Queue Service)
Account: ust-iaas
...

PaaS Provider (Elastic Beanstalk)
Account: ust-iaas
...

DBMS Provider (Relational Database Service)
Account: ust-iaas
...

Database (MySQL Database 5.7)
DB-Name: webshop
...

Order Processor
Type: FIFO
Server-Side Encryption: Yes
Master-Key: Default

Queue
Type: FIFO
Server-Side Encryption: Yes
Master-Key: Default

Order App
Port: 80
...

MoM Provider
Account: ust-iaas
...

Public Cloud

Platform as a Service

Public Cloud

= JmsQueueCon_1.1
= SQL-Con
= hostedOn
Example Deployment Model

Order App (Java 8 Web App)
- Port: 80

Queue (SQS Queue)
- Type: FIFO
- Server-Side Encryption: Yes
- Master-Key: Default

Order Processor (Java 8 App)
- MaxHeap: 6GB

Database (MySQL Database 5.7)
- DB-Name: webshop

DBMS (MySQL DBMS 5.7)
- DBMS Provider (Relational Database Service)
- Account: ust-iaas

MoM Provider (Simple Queue Service)
- Account: ust-iaas

MoM Provider (Simple Queue Service)
- Account: ust-iaas

Platform as a Service

Public Cloud

= JmsQueueCon_1.1
= SQL-Con
= hostedOn
Example Deployment Model

Order App (Java 8 Web App)
- Port: 80
- [...]

Order Processor (Java 8 App)
- MaxHeap: 6GB
- [...]

Database (MySQL Database 5.7)
- DB-Name: webshop
- [...]

DBMS (MySQL DBMS 5.7)
- Server-Side Encryption: Yes
- Master-Key: Default
- [...]

MoM Provider (Simple Queue Service)
- Account: ust-iaas
- [...]

MoM Provider (Simple Queue Service)
- [...]

DBMS Provider (Relational Database Service)
- Account: ust-iaas
- [...]

Public Cloud

Platform as a Service

Point-to-Point Channel

= JmsQueueCon_1.1

= SQL-Con

= hostedOn
Example Deployment Model

- **Order App** (Java 8 Web App)
  - Port: 80
  - [...]

- **Order Processor** (Java 8 App)
  - MaxHeap: 6GB
  - [...]

- **Database** (MySQL Database 5.7)
  - DB-Name: webshop
  - [...]

- **DBMS** (MySQL DBMS 5.7)
  - Server-Side Encryption: Yes
  - Master-Key: Default
  - [...]

- **DBMS Provider** (Relational Database Service)
  - Account: ust-iaas
  - [...]

- **Order Processor** (Java 8 App) connected to **Database** (MySQL Database 5.7) via **Message-oriented Middleware**

- **Order App** (Java 8 Web App) connected to **Message-oriented Middleware** via **Point-to-Point Channel**

- **Public Cloud** connected to **Message-oriented Middleware** via **JmsQueueCon_1.1**

- **Message-oriented Middleware** connected to **DBMS Provider** via **hostedOn**

Legend:
- ➡️ = JmsQueueCon_1.1
- ➡️ = SQL-Con
- ➡️ = hostedOn
Example Deployment Model

Order App
(Java 8 Web App)
Port: 80
[...]

Order Processor
(Java 8 App)
MaxHeap: 6GB
[...]

Relational Database

Platform as a Service

Point-to-Point Channel

Message-oriented Middleware

Public Cloud

= JmsQueueCon_1.1
= SQL-Con
= hostedOn
Pattern-based Deployment Models [4]

Describing semantics rather than technologies

Order App (Java 8 Web App) ➜ Order Processor (Java 8 App)

- Platform as a Service
- Public Cloud
- Message-oriented Middleware
- Point-to-Point Channel
- Relational Database

However, behavior cannot be described using patterns...

= JMS-2.0-Queue-Connection  = SQL-Connection  = hostedOn

Pattern-based Deployment and Configuration Models
Pattern-based Deployment and Configuration Models (PbDCMs)

Stateless Component
Unpredictable Workload
UI Component
Exactly-once Delivery
Information Obscurity

Order App (Java 8 Web App)
Platform as a Service

Order Processor (Java 8 App)
Public Cloud
Relational Database

Processing Component
Secure Channel

=> JMS-2.0-Queue-Connection
=> SQL-Connection
= hostedOn
How to deploy an application that is modelled in the form of a PbDCM?
Pattern-based Deployment Modelling Method

Vendor & Technology Agnostic Modeling

(Semi-) Automatic Refinement

Vendor & Technology Specific Modeling

Software Artifacts, Architecture Models, Applied Patterns

Derive

Pattern-based Deployment Model

Choose PRM Candidate

Replace Fragment

Deployment Model Candidate

Refine (optional)

Executable Deployment Model

Harzenetter, Lukas; Breitenbücher, Uwe; Michael, Falkenthal; Guth, Jasmin; Krieger, Christoph; Leymann, Frank: Pattern-based Deployment Models and Their Automatic Execution. In: 11th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2018), IEEE Computer Society, 2018
Component and Behavior Pattern Refinement Model (CBPRM)

CBPRM Extensions

Detector

Mappings

RelationType: hostedOn
Direction: ingoing
SourceType: *

RelationMapping

Stay

Refinement Structure

PaaS Provider
(Elastic Beanstalk)

PaaS
(AWS Webserver Env)

AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min

[...]

Public Cloud
(AWS)

[...]

RelationType
(hostedOn)

Direction
(ingoing)

SourceType
(*)

Detected

Mappings

Stay

Refinement Structure

PaaS Provider
(Elastic Beanstalk)

PaaS
(AWS Webserver Env)

AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min

[...]

Public Cloud
(AWS)

[...]

Java 8 Web App

[...]

Component and Behavior Pattern Refinement Model (CBPRM)
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App) ➔ Order Processor (Java 8 App)

- JMS-2.0-Queue-Connection ➔ SQL-Connection ➔ hostedOn

Research
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App)

Detector

Relation Mapping
Relation Type: hostedOn
Direction: ingoing
SourceType: *

Mappings

Refinement Structure

PaaS Provider (Elastic Beanstalk)

Public Cloud (AWS)

Relation Type: hostedOn
Direction: ingoing
SourceType: *

Relation Mapping

Stay

PaaS (AWS Webserver Env)

AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min

[...]
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App) → Stay

- (Java 8 Web App)
- (Java 8 Web App)

RelationType: hostedOn
Direction: ingoing
SourceType: *

RelationMapping

Detector

Mappings

Refinement Structure

PaaS Provider (Elastic Beanstalk)

PaaS (AWS Webserver Env)

AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min

Public Cloud (AWS)

- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

Stay

= JMS-2.0-Queue-Connection

= SQL-Connection

= hostedOn
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App)

PaaS (AWS Webserver Env)
  - AutoScaling: true
  - ScalingMetric: NetOut
  - TimeSpan: 5min

PaaS Provider (Elastic Beanstalk)

Public Cloud (AWS)

- RelationType: hostedOn
- Direction: ingoing
- SourceType: *
- RelationMapping
- Stay

Detector

Mappings

Refinement Structure

PaaS (AWS Webserver Env)
  - AutoScaling: true
  - ScalingMetric: NetOut
  - TimeSpan: 5min

PaaS Provider (Elastic Beanstalk)

Public Cloud (AWS)

- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

- JMS-2.0-Queue-Connection
- SQL-Connection
- hostedOn
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App) → PaaS (AWS Webserver Env)
  | AutoScaling: true
  | ScalingMetric: NetOut
  | TimeSpan: 5min
  | ...

PaaS Provider (Elastic Beanstalk) → Public Cloud (AWS)
  | ...

(Java 8 Web App) → Detector
  | RelationMapping
  | RelationType: hostedOn
  | Direction: incoming
  | SourceType: *

AutoScaling: true
ScalingMetric: NetOut
TimeSpan: 5min

PaaS Provider (Elastic Beanstalk) → Public Cloud (AWS)
  | ...

RelationType: hostedOn
Direction: ingoing
SourceType: *

Network Flow:
- = JMS-2.0-Queue-Connection
- = SQL-Connection
- = hostedOn
But how to refine relations that are annotated with patterns?
Component and Behavior Pattern Refinement Model (CBPRM)

- Detector
- Mappings
- Refinement Structure

= SQL-Connection
= Secure-SQL-Connection
Pattern-based Deployment and Configuration Models – Refinement

**Order App (Java 8 Web App)**

**PaaS (AWS Webserver Env)**
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

**PaaS Provider (Elastic Beanstalk)**

**Public Cloud (AWS)**

**Order Processor (Java 8 App)**

**AutoScaling**: true
**ScalingMetric**: NetOut
**TimeSpan**: 5min

- **JMS-2.0-Queue-Connection**
- **SQL-Connection**
- **hostedOn**
Pattern-based Deployment and Configuration Models – Refinement

Order Processor (Java 8 App)

Detector

Mappings

Refinement Structure

= JMS-2.0-Queue-Connection

= SQL-Connection

= hostedOn
Pattern-based Deployment and Configuration Models – Refinement

Order Processor (Java 8 App)

Detector

Mappings

Refinement Structure

(Application) ➔ Stay ➔ (Application)

= JMS-2.0-Queue-Connection

= SQL-Connection

= hostedOn
Pattern-based Deployment and Configuration Models – Refinement

Order App (Java 8 Web App)

PaaS (AWS Webserver Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

PaaS Provider (Elastic Beanstalk)

Public Cloud (AWS)

Order Processor (Java 8 App)

= JMS-2.0-Queue-Connection = SQL-Connection = hostedOn

[...]
One of Many Refinement Result

Order App (Java 8 Web App)
- Port: 80

Order PaaS (AWS Webserver Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

Queue (SQS Queue)
- Type: FIFO
- Encryption: Yes
- Master-Key: Default

MoM Provider (Simple Queue Service)
- Account: ust-iaas

Public Cloud (AWS)
- Region: EU

Order Processor (Java 8 App)
- MaxHeap: 6GB

Processor PaaS (AWS Worker Env)
- AutoScaling: true
- ScalingMetric: NetOut
- TimeSpan: 5min

PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

Database (MySQL Database 5.7)
- DB-Name: webshop

DBMS (MySQL DBMS 5.7)
- Server-Side Encryption: Yes
- Master-Key: Default

DBMS Provider (Relational Database Service)
- Account: ust-iaas

MoM Provider (Simple Queue Service)
- Account: ust-iaas

PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

PaaS Provider (Elastic Beanstalk)
- Account: ust-iaas

DBMS Provider (Relational Database Service)
- Account: ust-iaas

- = Secure-JMS-Queue-Connection
- = Secure-SQL-Connection
- = hostedOn
Conclusion & Future Work

- Open-source implementation based on Eclipse Winery
  - Demo video available on YouTube
- Pattern-based Deployment and Configuration Models
  - Less technology-specific knowledge needed
  - Avoid vendor and technology lock-ins
- Component and Behavior Pattern Refinement Models
  - Rules how refinement can be done (semi-) automatically
- Limitation: assumption that Deployment Models are correct
  - Close gap between architecture and deployment models
- Extend approach to automatically generate CBPRM permutations

Thank you!