



A PRACTICAL GUIDE TO

---

# CONTINUOUS DELIVERY

EBERHARD WOLFF

# **A Practical Guide to Continuous Delivery**

# Practical Guide to Continuous Delivery, A

## Table of Contents

Cover

Title Page

Copyright Page

Contents

Preface

Acknowledgments

About the Author

Part I: Foundations

Chapter 1: Continuous Delivery: What and How?

1.1 Introduction: What Is Continuous Delivery?

1.2 Why Software Releases are So Complicated

1.2.1 Continuous Integration Creates Hope

1.2.2 Slow and Risky Processes

1.2.3 Its Possible to be Fast

1.3 Values of Continuous Delivery

1.3.1 Regularity

1.3.2 Traceability/Confirmability

1.3.3 Regression

1.4 Benefits of Continuous Delivery

1.4.1 Continuous Delivery for Time to Market

1.4.2 One Example

1.4.3 Implementing a Feature and Bringing It into Production

1.4.4 On to the Next Feature

1.4.5 Continuous Delivery Generates Competitive Advantages

1.4.6 Without Continuous Delivery

# **Table of Contents**

1.4.7 Continuous Delivery and Lean Startup

1.4.8 Effects on the Development Process

1.4.9 Continuous Delivery to Minimize Risk

1.4.10 Faster Feedback and Lean

## **1.5 Generations and Structure of a Continuous Delivery Pipeline**

1.5.1 The Example

## **1.6 Conclusion**

Endnotes

## **Chapter 2: Providing Infrastructure**

### **2.1 Introduction**

2.1.1 Infrastructure Automation: An Example

### **2.2 Installation Scripts**

2.2.1 Problems of Classical Installation Scripts

### **2.3 Chef**

2.3.1 Chef versus Puppet

2.3.2 Other Alternatives

2.3.3 Technical Foundations

2.3.4 Chef Solo

2.3.5 Chef Solo: Conclusion

2.3.6 Knife and Chef Server

2.3.7 Chef Server: Conclusion

### **2.4 Vagrant**

2.4.1 An Example with Chef and Vagrant

2.4.2 Vagrant: Conclusion

### **2.5 Docker**

2.5.1 Dockers Solution

2.5.2 Creating Docker Containers

2.5.3 Running the Example Application with Docker

2.5.4 Docker and Vagrant

2.5.5 Docker Machine

2.5.6 Complex Configurations with Docker

2.5.7 Docker Compose

# Table of Contents

## 2.6 Immutable Server

### 2.6.1 Disadvantages of Idempotency

### 2.6.2 Immutable Server and Docker

## 2.7 Infrastructure as Code

### 2.7.1 Testing Infrastructure as Code

## 2.8 Platform as a Service (PaaS)

## 2.9 Handling Data and Databases

### 2.9.1 Handling Schemas

### 2.9.2 Test and Master Data

## 2.10 Conclusion

## Endnotes

# Part II: The Continuous Delivery Pipeline

## Chapter 3: Build Automation and Continuous Integration

### 3.1 Introduction

#### 3.1.1 Build Automation: An Example

### 3.2 Build Automation and Build Tools

#### 3.2.1 Build Tools in the Java World

#### 3.2.2 Ant

#### 3.2.3 Maven

#### 3.2.4 Gradle

#### 3.2.5 Additional Build Tools

#### 3.2.6 Choosing the Right Tool

### 3.3 Unit Tests

#### 3.3.1 Writing Good Unit Tests

#### 3.3.2 TDDTest-Driven Development

#### 3.3.3 Clean Code and Software Craftsmanship

### 3.4 Continuous Integration

#### 3.4.1 Jenkins

#### 3.4.2 Continuous Integration Infrastructure

#### 3.4.3 Conclusion

### 3.5 Measuring Code Quality

# **Table of Contents**

3.5.1 SonarQube

## **3.6 Managing Artifacts**

3.6.1 Integration into the Build

3.6.2 Advanced Features of Repositories

## **3.7 Conclusion**

Endnotes

## **Chapter 4: Acceptance Tests**

### **4.1 Introduction**

4.1.1 Acceptance Tests: An Example

### **4.2 The Test Pyramid**

### **4.3 What Are Acceptance Tests?**

4.3.1 Automated Acceptance Tests

4.3.2 More Than Just an Increase in Efficiency

4.3.3 Manual Tests

4.3.4 What about the Customer?

4.3.5 Acceptance versus Unit Tests

4.3.6 Test Environments

### **4.4 GUI-Based Acceptance Tests**

4.4.1 Problems of GUI Tests

4.4.2 Abstractions against Fragile GUI Tests

4.4.3 Automation with Selenium

4.4.4 Web Driver API

4.4.5 Tests without Web Browser: HtmlUnit

4.4.6 Selenium Web Driver API

4.4.7 Selenium IDE

4.4.8 Problems with Automated GUI Tests

4.4.9 Executing GUI Tests

4.4.10 Exporting the Tests as Code

4.4.11 Manual Modifications of the Test Cases

4.4.12 Test Data

4.4.13 Page Object

### **4.5 Alternative Tools for GUI Tests**

# Table of Contents

4.5.1 PhantomJS

4.5.2 Windmill

## 4.6 Textual Acceptance Tests

4.6.1 Behavior-Driven Development

4.6.2 Different Adaptors

## 4.7 Alternative Frameworks

## 4.8 Strategies for Acceptance Tests

4.8.1 The Right Tool

4.8.2 Rapid Feedback

4.8.3 Test Coverage

## 4.9 Conclusion

Endnotes

## Chapter 5: Capacity Tests

### 5.1 Introduction

5.1.1 Capacity Tests: An Example

### 5.2 Capacity TestsHow?

5.2.1 Objectives of Capacity Tests

5.2.2 Data Volumes and Environments

5.2.3 Performance Tests Only at the End of the Implementation?

5.2.4 Capacity Tests = Risk Management

5.2.5 Simulating Users

5.2.6 Documenting Performance Requirements

5.2.7 Hardware for Capacity Tests

5.2.8 Cloud and Virtualization

5.2.9 Minimizing Risk by Continuous Testing

5.2.10 Capacity TestsSensible or Not?

### 5.3 Implementing Capacity Tests

### 5.4 Capacity Tests with Gatling

5.4.1 Demo versus Real Life

### 5.5 Alternatives to Gatling

5.5.1 Grinder

5.5.2 Apache JMeter

# Table of Contents

5.5.3 Tsung

5.5.4 Commercial Solutions

5.6 Conclusion

Endnotes

## Chapter 6: Exploratory Testing

6.1 Introduction

6.1.1 Exploratory Tests: An Example

6.2 Why Exploratory Tests?

6.2.1 Sometimes Manual Testing Is Still Better

6.2.2 Test by the Customers

6.2.3 Manual Tests for Non-Functional Requirements

6.3 How to Go About It?

6.3.1 Missions Guide the Tests

6.3.2 Automated Environment

6.3.3 Showcases as a Basis

6.3.4 Example: An E-Commerce Application

6.3.5 Beta Tests

6.3.6 Session-Based Tests

6.4 Conclusion

Endnotes

## Chapter 7: DeployThe Rollout in Production

7.1 Introduction

7.1.1 Deployment: An Example

7.2 Rollout and Rollback

7.2.1 Benefits

7.2.2 Disadvantages

7.3 Roll Forward

7.3.1 Benefits

7.3.2 Disadvantages

7.4 Blue/Green Deployment

7.4.1 Benefits



# Table of Contents

7.4.2 Disadvantages

## 7.5 Canary Releasing

7.5.1 Benefits

7.5.2 Disadvantages

## 7.6 Continuous Deployment

7.6.1 Benefits

7.6.2 Disadvantages

## 7.7 Virtualization

7.7.1 Physical Hosts

## 7.8 Beyond Web Applications

## 7.9 Conclusion

Endnotes

## Chapter 8: Operations

### 8.1 Introduction

8.1.1 OperateAn Example

### 8.2 Challenges in Operations

### 8.3 Log Files

8.3.1 What Should Be Logged?

8.3.2 Tools for Processing Log Files

8.3.3 Logging in the Example Application

### 8.4 Analyzing Logs of the Example Application

8.4.1 Analyses with Kibana

8.4.2 ELKScalability

### 8.5 Other Technologies for Logs

### 8.6 Advanced Log Techniques

8.6.1 Anonymization

8.6.2 Performance

8.6.3 Time

8.6.4 Ops Database

### 8.7 Monitoring

### 8.8 Metrics with Graphite

# **Table of Contents**

## 8.9 Metrics in the Example Application

### 8.9.1 Structure of the Example

## 8.10 Other Monitoring Solutions

## 8.11 Additional Challenges When Operating an Application

### 8.11.1 Scripts

### 8.11.2 Applications in a Clients Data Center

## 8.12 Conclusion

### Endnotes

## Part III: Management, Organization, and Architecture for Continuous Delivery

### Chapter 9: Introducing Continuous Delivery into Your Enterprise

#### 9.1 Introduction

#### 9.2 Continuous Delivery Right from the Start

#### 9.3 Value Stream Mapping

##### 9.3.1 Value Stream Mapping Describes the Sequence of Events

##### 9.3.2 Optimizations

#### 9.4 Additional Measures for Optimization

##### 9.4.1 Quality Investments

##### 9.4.2 Costs

##### 9.4.3 Benefits

##### 9.4.4 Do not Check in on a Red Build!

##### 9.4.5 Stop the Line

##### 9.4.6 5 Whys

##### 9.4.7 DevOps

#### 9.5 Conclusion

### Endnotes

### Chapter 10: Continuous Delivery and DevOps

#### 10.1 Introduction

#### 10.2 What Is DevOps?

##### 10.2.1 Problems

##### 10.2.2 The Client Perspective

# Table of Contents

10.2.3 Pioneer: Amazon

10.2.4 DevOps

## 10.3 Continuous Delivery and DevOps

10.3.1 DevOps: More Than Continuous Delivery

10.3.2 Individual Responsibility and Self-Organization

10.3.3 Technology Decisions

10.3.4 Less Central Control

10.3.5 Technology Pluralism

10.3.6 Exchange Between Teams

10.3.7 Architecture

## 10.4 Continuous Delivery without DevOps?

10.4.1 Terminating the Continuous Delivery Pipeline

## 10.5 Conclusion

Endnotes

## Chapter 11: Continuous Delivery, DevOps, and Software Architecture

### 11.1 Introduction

### 11.2 Software Architecture

11.2.1 Why Software Architecture?

### 11.3 Optimizing Architecture for Continuous Delivery

11.3.1 Smaller Deployment Units

### 11.4 Interfaces

11.4.1 Postels Law or the Robustness Principle

11.4.2 Design for Failure

11.4.3 State

### 11.5 Databases

11.5.1 Keeping Databases Stable

11.5.2 Database = Component

11.5.3 Views and Stored Procedures

11.5.4 A Database per Component

11.5.5 NoSQL Databases

### 11.6 Microservices

# **Table of Contents**

- 11.6.1 Microservices and Continuous Delivery
- 11.6.2 Introducing Continuous Delivery with Microservices
- 11.6.3 Microservices Entail Continuous Delivery
- 11.6.4 Organization

## **11.7 Handling New Features**

- 11.7.1 Feature Branches
- 11.7.2 Feature Toggles
- 11.7.3 Benefits
- 11.7.4 Use Cases for Feature Toggles
- 11.7.5 Disadvantages

## **11.8 Conclusion**

Endnotes

## **Chapter 12: Conclusion: What Are the Benefits?**

Endnotes

Index