

# CONTENTS

---

## PART I FOUNDATIONS

<b>1</b>	<b>Introduction: Developer and operation teams converge and both use software engineering practices</b>	<b>3</b>
<b>2</b>	<b>Developers use the Continuous Delivery Pipeline</b>	<b>5</b>
2.1	The Continuous Delivery Pipeline consists of commitment, continuous integration and deployment	5
2.2	Software Deployment approaches evolved from manual to automated	5
2.2.1	Blue-Green Deployment allows Zero Downtime releases	5
2.2.2	Automation leads to resource saving Phoenix Deployment and Rolling Deployments	5
2.2.3	Canaries test releases with a small amount of traffic	5
2.2.4	continuous deployment is not continuous delivery	5
<b>3</b>	<b>Operators evolved to Site Reliability Engineers</b>	<b>7</b>
3.1	Site Reliability Engineers maintain applications like software engineers	7
3.2	Monitoring to identify Problems	7
3.2.1	Health checks measure availability	7
3.2.2	Measuring Latency, Traffic, Errors and Saturation identifies failures and performance problems	7
		<b>i</b>

3.2.3	Incident Management (/Notifications) for appropriate and fast actions in case of Problems	7
-------	---	---

## PART II NEW PRACTICES

<b>4</b>	<b>Post Release Testing extends the Continuous Delivery Pipeline to support maintaining a system</b>	<b>11</b>
4.1	Post Release Testing leads to lower time to market	11
4.2	It makes Releases consistent, measurable, fast and scalable	11
4.3	It is a new opportunity for risk management	11
4.4	Companies are already post release testing their software systems	12
4.4.1	Netflix uses Simian Army to live test their systems	12
4.4.2	Synthetic Monitoring tests a complex distributed system	12
4.5	Non Change Post Release Testing with Canaries	12
4.5.1	black box monitoring is only one part and monitoring change is difficult	12
4.5.2	Continuous Delivery is a requirement	12
4.5.3	Notifications in case a canary behaves different	12
4.5.4	Automated Rollbacks for a automatic self healing system	12
<b>5</b>	<b>Implementing Canary Post Release Testing</b>	<b>13</b>
5.1	New technologies drive new techniques	13
5.1.1	Kubernetes is a Cluster OS	13
5.1.1.1	overview: cluster os - resource management	13
5.1.1.2	immutability	13
5.1.2	datadog	13
5.1.3	ruby - sinatra	13
5.2	the deployer	13
5.2.1	architecture and how it integrates in the pipeline	13
5.2.2	importance for maintenance and feature deploys	13
5.2.3	cycle time vs. notifications	13
<b>6</b>	<b>Velocity and cycletime as efficiency metrics of an agile team</b>	<b>15</b>
<b>7</b>	<b>MTTR and ErrorRate measures the quality of a software</b>	<b>17</b>

## PART III EVALUATION

<b>8</b>	<b>GapFish a startup company</b>	<b>21</b>
<b>9</b>	<b>GapFish's traditional toolchain and teams</b>	<b>23</b>
<b>10</b>	<b>team agility metrics</b>	<b>25</b>

10.1	cycle time measures quality of delivery engine	25
10.2	locs/deploy measures risk	25
10.3	deploys per day measures agility	25
<b>11</b>	<b>software quality metrics</b>	<b>27</b>
11.1	ErrorRate as monitoring measure for automation	27
11.2	problems in error rate measure defect and failure	27
11.3	solution a secific heuristic	27
<b>12</b>	<b>Results</b>	<b>29</b>
12.1	traditional vs. new	29
<b>13</b>	<b>Lessons learned and future</b>	<b>31</b>
<b>14</b>	<b>theoretical/practical conclusion</b>	<b>33</b>
<b>15</b>	<b>for Gapfish</b>	<b>35</b>



## PART I

---

# FOUNDATIONS

---



## CHAPTER 1

---

# INTRODUCTION: DEVELOPER AND OPERATION TEAMS CONVERGE AND BOTH USE SOFTWARE ENGINEERING PRACTICES

---





## **CHAPTER 2**

---

# **DEVELOPERS USE THE CONTINUOUS DELIVERY PIPELINE**

---

- 2.1 The Continuous Delivery Pipeline consists of commitment, continuous integration and deployment**
- 2.2 Software Deployment approaches evolved from manual to automated**
  - 2.2.1 Blue-Green Deployment allows Zero Downtime releases**
  - 2.2.2 Automation leads to resource saving Phoenix Deployment and Rolling Deployments**
  - 2.2.3 Canaries test releases with a small amount of traffic**
  - 2.2.4 continuous deployment is not continuous delivery**



## CHAPTER 3

---

# OPERATORS EVOLVED TO SITE RELIABILITY ENGINEERS

---

### **3.1 Site Reliability Engineers maintain applications like software engineers**

sw installation, hw installation, logging, scaling, monitoring (detecting problems), security, incident management, support

### **3.2 Monitoring to identify Problems**

#### **3.2.1 Health checks measure availability**

#### **3.2.2 Measuring Latency, Traffic, Errors and Saturation identifies failures and performance problems**

#### **3.2.3 Incident Management (/Notifications) for appropriate and fast actions in case of Problems**



## PART II

---

## NEW PRACTICES

---



## CHAPTER 4

---

# POST RELEASE TESTING EXTENDS THE CONTINUOUS DELIVERY PIPELINE TO SUPPORT MAINTAINING A SYSTEM

---

**4.1 Post Release Testing leads to lower time to market**

**4.2 It makes Releases consistent, measurable, fast and scalable**

mttr

**4.3 It is a new opportunity for risk management**

identify test before release is a mttr of zero, after release still fast. easier to test in production (complexity of system)

#### **4.4 Companies are already post release testing their software systems**

##### **4.4.1 Netflix uses Simian Army to live test their systems**

##### **4.4.2 Synthetic Monitoring tests a complex distributed system**

#### **4.5 Non Change Post Release Testing with Canaries**

##### **4.5.1 black box monitoring is only one part and monitoring change is difficult**

##### **4.5.2 Continuous Delivery is a requirement**

##### **4.5.3 Notifications in case a canary behaves different**

##### **4.5.4 Automated Rollbacks for a automatic self healing system**



## CHAPTER 5

---

# IMPLEMENTING CANARY POST RELEASE TESTING

---

### 5.1 New technologies drive new techniques

#### 5.1.1 Kubernetes is a Cluster OS

##### *5.1.1.1 overview: cluster os - resource management*

##### *5.1.1.2 immutability*

#### 5.1.2 datadog

#### 5.1.3 ruby - sinatra

### 5.2 the deployer

#### 5.2.1 architecture and how it integrates in the pipeline

#### 5.2.2 importance for maintenance and feature deploys

#### 5.2.3 cycle time vs. notifications



## CHAPTER 6

---

### VELOCITY AND CYCLETIME AS EFFICIENCY METRICS OF AN AGILE TEAM

---



## CHAPTER 7

---

MTTR AND ERRORRATE MEASURES THE  
QUALITY OF A SOFTWARE

---



## PART III

---

## EVALUATION

---





## CHAPTER 8

---

### GAPFISH A STARTUP COMPANY

---



## CHAPTER 9

---

### GAPFISH'S TRADITIONAL TOOLCHAIN AND TEAMS

---



## **CHAPTER 10**

---

### **TEAM AGILITY METRICS**

---

**10.1 cycle time measures quality of delivery engine**

**10.2 locs/deploy measures risk**

**10.3 deploys per day measures agility**



## CHAPTER 11

---

### SOFTWARE QUALITY METRICS

---

11.1 ErrorRate as monitoring measure for automation

11.2 problems in error rate measure defect and failure

11.3 solution a secific heuristic





## CHAPTER 12

---

## RESULTS

---

### 12.1 traditional vs. new



## CHAPTER 13

---

### LESSONS LEARNED AND FUTURE

---



## CHAPTER 14

---

## THEORETICAL/PRACTICAL CONCLUSION

---



## CHAPTER 15

---

FOR GAPFISH

---