

CONTENTS

PART I FOUNDATIONS

1	Introduction: Developer and operation teams converge and both use software engineering practices	3
2	Developers use the Continuous Delivery Pipeline	5
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
3	Operators evolved to Site Reliability Engineers	7
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
		i

3.2.3	Incident Management (/Notifications) for appropriate and fast actions in case of Problems	7
4	Metrics can indirectly measure team efficiency and software quality	9
4.1	Velocity and cycletime are efficiency metrics for an agile team	9
4.1.1	Deploys/Week indirectly measures velocity	9
4.1.2	Deploy Duration is import for cycle time	9
4.2	MTTR and Failure rate measures the quality of a software	9
4.2.1	LOCS/Deploy indirectly measures the risk per Deploy	9
PART II NEW PRACTICES		
5	Post Release Testing extends the Continuous Delivery Pipeline to support maintaining a system	13
5.1	Post Release Testing leads to lower time to market	13
5.2	It makes Releases consistent, measurable, fast and scalable	13
5.3	It is a new opportunity for risk management	13
5.4	Companies are already post release testing their software systems	14
5.4.1	Netflix uses Simian Army to live test their systems	14
5.4.2	Synthetic Monitoring tests a complex distributed system	14
5.5	Post Release Testing with Canaries is appropriate for testing non change	14
5.5.1	Black-Box monitoring is only one part and monitoring change is difficult	14
5.5.2	Canary testing is important for maintenance but not feature deploys	14
5.5.3	Continuous Delivery is a requirement	14
5.5.4	Notifications in case a canary behaves different	14
5.5.5	Automated Rollbacks for a automatic self healing system	14
6	Implementing Canary Post Release Testing	15
6.1	New technologies drive new techniques	15
6.1.1	Kubernetes is a Cluster OS	15
6.1.1.1	Resource Management in Kubernetes is made for high available services	15
6.1.1.2	Deployments implement Rolling Updates	15
6.1.2	DataDog is a Cluster Monitoring Systems as SaaS	15
6.1.2.1	The main components are: Datacollection, Timeseriesdatabase, Graphing and Alerting	15
6.1.2.2	DataDog integrates well in Kubernetes	15
6.2	Deployer is a Service for Continuous Deployment and enables Canary Post Release testing	16
6.2.1	Deployer integrates into the Continuous Delivery Pipeline	16
6.2.2	Deployer integrates into Kubernetes and deploys itself	16
6.2.3	Deployer integrates into Monitoring and enables Canary testing	16

6.2.4	The main Deployer API features are Deployments and Canary deployments	16
6.2.4.1	Deployments deploy a whole repository	16
6.2.4.2	One Canary per Replicated Pods can be deployed and monitored	16
6.2.4.3	Immediate Notifications in case of failure	16
6.2.4.4	Automatic Rollbacks in case defect Canary	16

PART III EVALUATION

7	GapFish is the company to evaluate Deployer	19
7.1	GapFish's services are complex and highly available	19
7.1.1	GapFish's Operation Service is used by internal Staff and Customers	19
7.2	GapFish uses tools for continuous deployment	19
7.2.1	GapFish differentiates between development and operation	19
7.2.2	Kubernetes enables GapFish to have a development and operation in one team	19
8	Implementation of Metrics: traditional vs new	21
8.1	Deploys/Week	21
8.2	Deploy Duration	21
8.3	LOCS/Deploy	21
9	Results	23
9.1	Metrics: traditional vs. new	23
9.2	theoretical/practical conclusion for deployer and cd	23
9.3	for Gapfish	23
9.4	Lessons learned and future	23
10	resume	25

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

CHAPTER 4

METRICS CAN INDIRECTLY MEASURE TEAM EFFICIENCY AND SOFTWARE QUALITY

4.1 Velocity and cycletime are efficiency metrics for an agile team

cycle time measures quality of delivery engine

4.1.1 Deploys/Week indirectly measures velocity

and it measures the effect of a quality delivery engine we want many deploys per week

4.1.2 Deploy Duration is import for cycle time

4.2 MTTR and Failure rate measures the quality of a software

we want low risk per deploy to achieve MTTR and low Failure Rate

4.2.1 LOCS/Deploy indirectly measures the risk per Deploy

PART II

NEW PRACTICES

CHAPTER 5

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

5.1 Post Release Testing leads to lower time to market

cycle time

5.2 It makes Releases consistent, measurable, fast and scalable

mttr, automation/automatic discussion

5.3 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)

5.4 Companies are already post release testing their software systems

5.4.1 Netflix uses Simian Army to live test their systems

5.4.2 Synthetic Monitoring tests a complex distributed system

5.5 Post Release Testing with Canaries is appropriate for testing non change

ErrorRate as monitoring measure for automation problems in error rate measure defect and failure solution a specific heuristic

5.5.1 Black-Box monitoring is only one part and monitoring change is difficult

5.5.2 Canary testing is important for maintenance but not feature deploys

5.5.3 Continuous Delivery is a requirement

5.5.4 Notifications in case a canary behaves different

5.5.5 Automated Rollbacks for a automatic self healing system

CHAPTER 6

IMPLEMENTING CANARY POST RELEASE TESTING

6.1 New technologies drive new techniques

6.1.1 Kubernetes is a Cluster OS

6.1.1.1 Resource Management in Kubernetes is made for high available services

6.1.1.2 Deployments implement Rolling Updates

6.1.2 DataDog is a Cluster Monitoring Systems as SaaS

alternativen Graphana, Prometheus

6.1.2.1 The main components are: Datacollection, Timeseriesdatabase, Graphing and Alerting

6.1.2.2 DataDog integrates well in Kubernetes

6.2 Deployer is a Service for Continuous Deployment and enables Canary Post Release testing

6.2.1 Deployer integrates into the Continuous Delivery Pipeline

6.2.2 Deployer integrates into Kubernetes and deploys itself

6.2.3 Deployer integrates into Monitoring and enables Canary testing

6.2.4 The main Deployer API features are Deployments and Canary deployments

depctl, curl

6.2.4.1 Deployments deploy a whole repository

6.2.4.2 One Canary per Replicated Pods can be deployed and monitored

6.2.4.3 Immediate Notifications in case of failure difference depctl and curl

6.2.4.4 Automatic Rollbacks in case defect Canary via datadog and triggers deployer future: staging deploys

PART III

EVALUATION

CHAPTER 7

GAPFISH IS THE COMPANY TO EVALUATE DEPLOYER

7.1 GapFish's services are complex and highly available

overview of Gapfish's services

7.1.1 GapFish's Operation Service is used by internal Staff and Customers

operation service == operation.gapfish.com

7.2 GapFish uses tools for continuous deployment

7.2.1 GapFish differentiates between development and operation

how is the deployment traditionally done

7.2.2 Kubernetes enables GapFish to have a development and operation in one team

which services are migrated to kubernetes

CHAPTER 8

IMPLEMENTATION OF METRICS: TRADITIONAL VS NEW

8.1 Deploys/Week

8.2 Deploy Duration

8.3 LOCS/Deploy

CHAPTER 9

RESULTS

9.1 Metrics: traditional vs. new

9.2 theoretical/practical conclusion for deployer and cd

9.3 for Gapfish

9.4 Lessons learned and future

CHAPTER 10

RESUME
