CONTENTS

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

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 Failurerate 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 secific 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 tradditionally 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

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