Oh Keptn, my Keptn
A data/observability driven way to DevOps & SRE automation

Andreas Grabner

DevOps Activist at Dynatrace

DevRel for Keptn

@grabnerandi, https://www.linkedin.com/in/grabnerandi

Follow us @keptnProject

Star us @ https://github.com/keptn/keptn

Slack Us @ https://slack.keptn.sh

We are a Cloud Native Computing Foundation Sandbox project.

My talk in 3 Acts

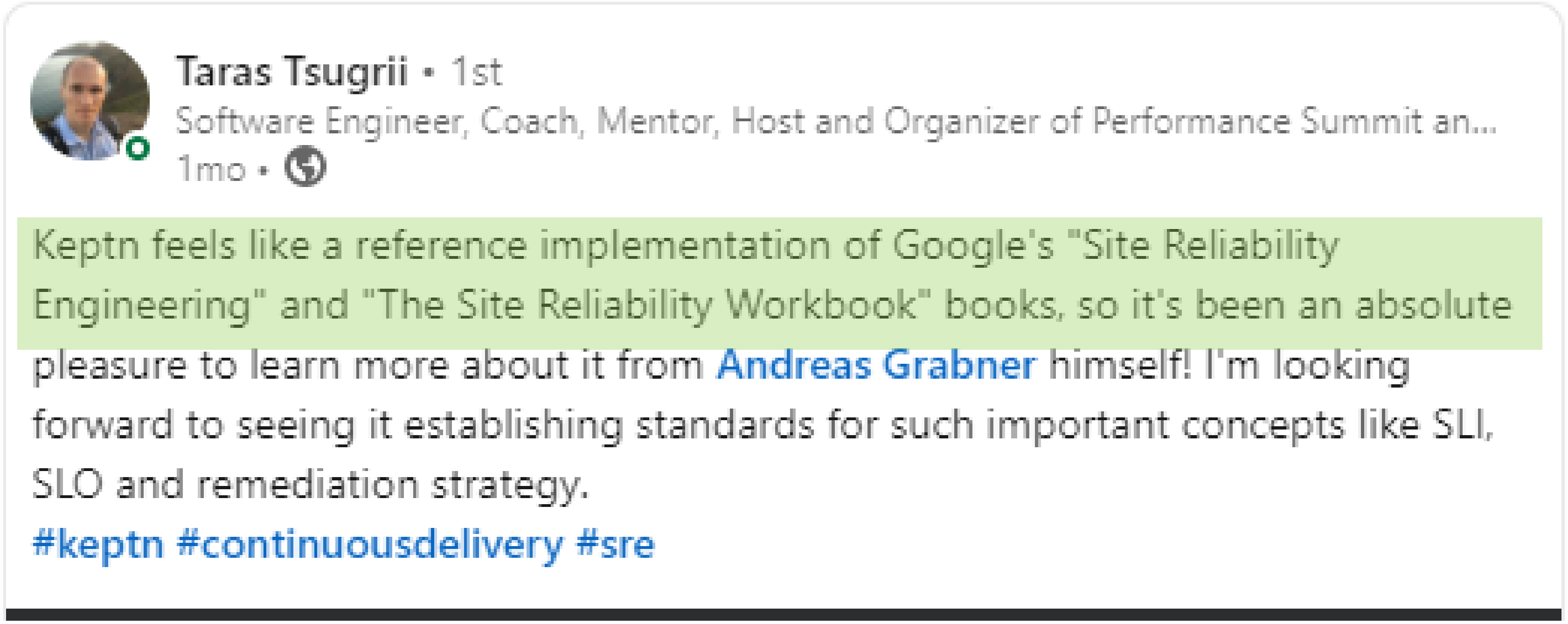
Act 1: What is Keptn
Key Use Cases and Adopters

Act 2: Why we built Keptn
What problems it solves!

Act 3: How Keptn works
Architecture, Extensibility …

Act 1 – What is Keptn

What is Keptn?



Taras Tsugrii • 1st

Software Engineer, Coach, Mentor, Host and Organizer of Performance Summit an...

1mo •

Keptn feels like a reference implementation of Google's "Site Reliability Engineering" and "The Site Reliability Workbook" books, so it's been an absolute pleasure to learn more about it from Andreas Grabner himself! I'm looking forward to seeing it establishing standards for such important concepts like SLI, SLO and remediation strategy.

#keptn #continuousdelivery #sre

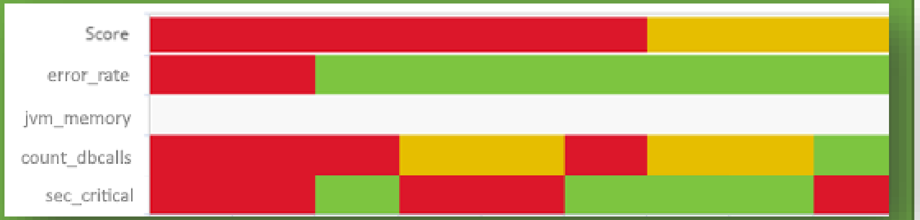
Keptn uses Service Level Objectives (SLO) to evaluate *App & Infra Desired State*

keptn

14:07 evaluation

evaluation

Lighthouse Service: Retrieves SLIs and compares them against SLOs



14:07 get-sli

get-sli

SLI Providers: Query SLIs based on sli.yaml and return individual values

DATADOG ... sonarqube Falco snyk

count_dbcalls: 5

jvm_memory: 360MB

error_rate: 4.3%

sec_critial: 1

slo.yaml (SLI Provider independent)

```
objectives:
  - sli: error_rate
    pass:
    - criteria:
      - "<=1“  # We expect a max error rate of 1%
  - sli: jvm_memory
  - sli: count_dbcalls
    pass:
    - criteria:
      - "=+2%"  # We allow a 2% increase in DB Calls between builds
    warning:
    - criteria:
      - "<=10"  # We expect no more than 10 DB Calls per TX
  - sli: sec_critical
    pass:
    - criteria:
      - "<=0"  # We do not allow any critical security issues
total_score:
  pass: "90%"
  warning: "75%"
```

sli.yaml (Prometheus)

```
indicators:
  error_rate:   "http_requests_total{status=“error"}"
  jvm_memory:   "jvm_memory_used_bytes{area="heap"}[1m]"
  sec_critical: "rate(falco_events[5m])"
```

sli.yaml (Dynatrace)

```
indicators:
  error_rate:    "builtin:service.errors.total.count"
  count_dbcalls: "calc:service.toptestdbcalls"
  jvm_memory:    "builtin:tech.jvm.memory.pool.committed"
  sec_critical:  "calc:secproblems:filter(risk,CRITICAL)"
```

SLO-based Evaluation in Action: Performance, Architecture, Security, …

```
$ keptn trigger evaluation myproject myservice buildId=4 timeframe=10m
```

SLIs (Service Level Indicators)	SLO pass	SLO warn	🟩 Build 1	🟨 Build 2	🟥 Build 3	🟩 Build 4
Response Time 95th Perc **Query**: builtin:service.responsetime(p95)	<=100ms	<= 250ms	🟩 80ms	🟨 120ms	🟩 90ms	🟩 95ms
Overall Failure Rate **Query**: builtin:service.errors.total	<= 2%	<= 5%	🟩 0%	🟨 4%	🟩 1%	🟩 0%
Test Step LOGIN Response Time **Query**: calc:service.teststeprt:filter(Test, LOGIN)	<=150ms & <=+10%	<= 400ms	🟩 100ms	🟩 90ms	🟨 120ms	🟩 95ms
Test Step LOGIN # Service Calls **Query**: calc:service.testsvc:filter(tx, LOGIN)	<= +0%		🟩 1	🟥 2	🟩 1	🟩 1
Critical Security Vulnerabilities **Query**: calc:secproblems:filter(risk,CRITICAL)	<=0		🟩 0	🟩 0	🟥 1	🟩 0
SLO: Overall Score Goal	90%	75%	🟩 100%	🟥 50%	🟥 70.0%	🟩 100%

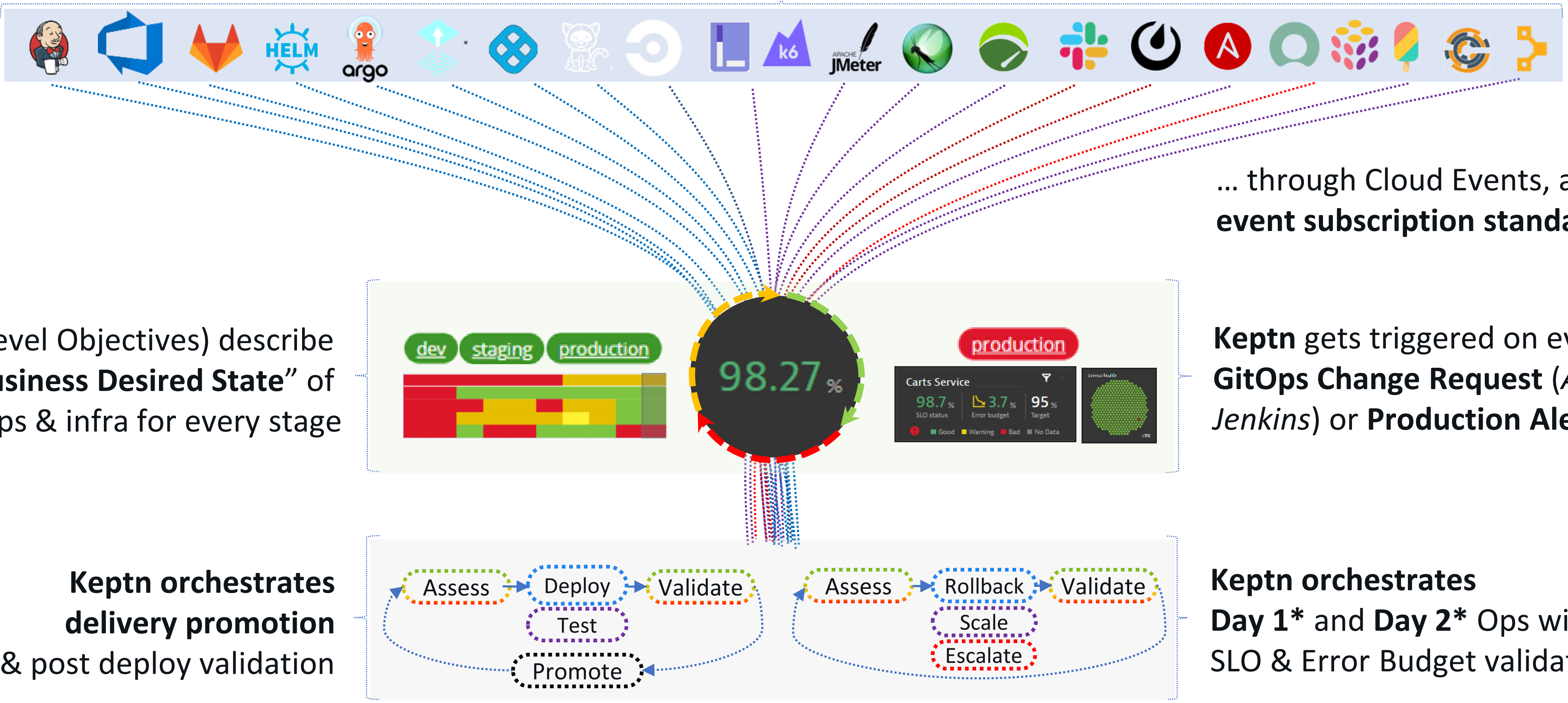
The big picture: SLO-Driven Lifecycle Orchestration with Keptn

Keptn connects to **any existing DevOps tool for** delivery, test, notification, ticketing, config mgmt …

… through Cloud Events, an **open event subscription standard!**

SLOs (Service Level Objectives) describe the "**Business Desired State**" of cloud native apps & infra for every stage

Keptn gets triggered on events such **GitOps Change Request** (*ArgoCD, Jenkins*) or **Production Alerts**



Keptn orchestrates delivery promotion with pre & post deploy validation

Keptn orchestrates Day 1* and Day 2* Ops with closed loop SLO & Error Budget validation

*Day 1: Progressive Delivery into Production

*Day 2: Automated incident response and remediation

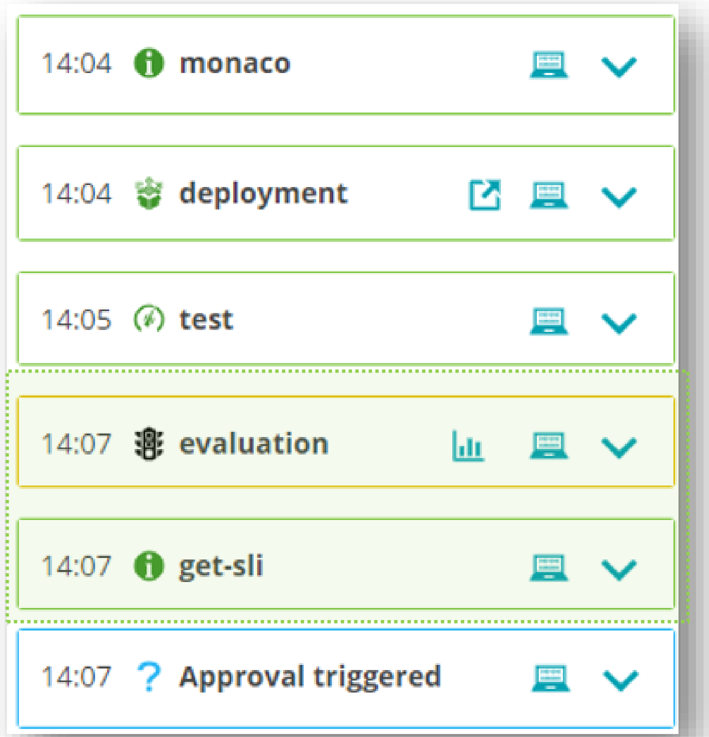
Trigger Keptn from your tools to orchestrate sequences around SLOs

keptn

triggers an automation sequence

keptn

orchestrates monitoring config, deployment, test execution, SLO evaluation & remediation



14:04 monaco

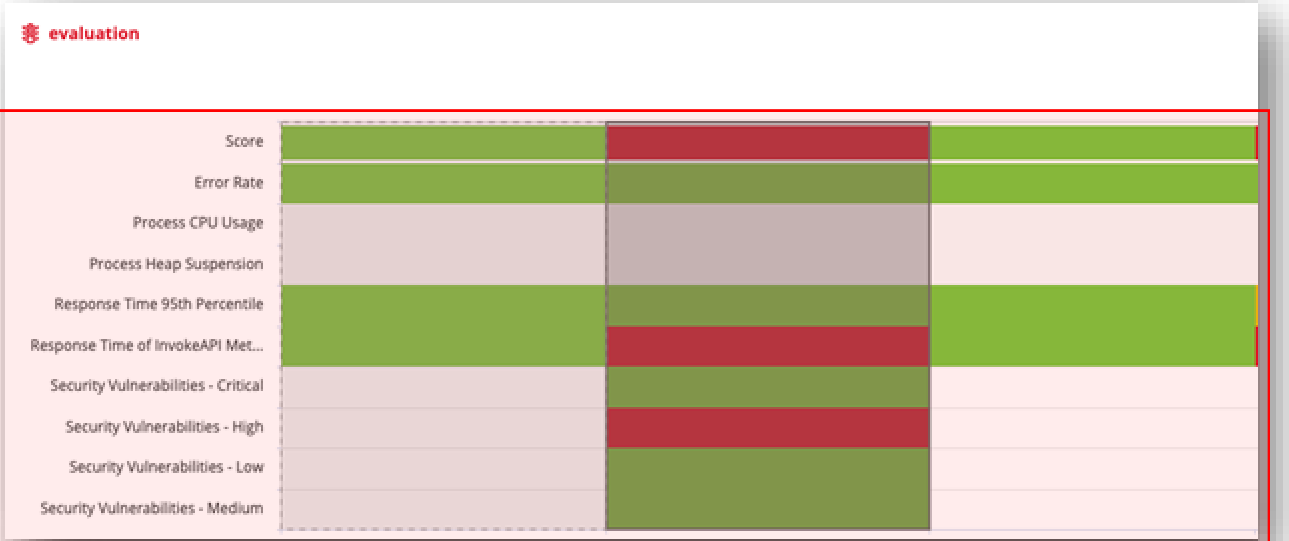
14:04 deployment

14:05 test

14:07 evaluation

14:07 get-sli

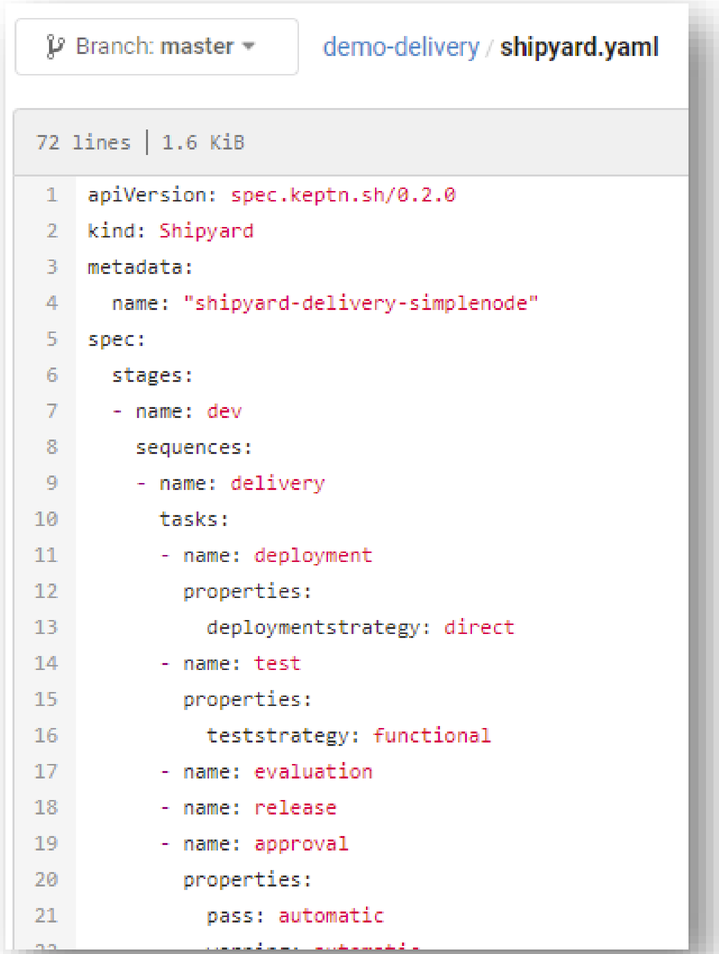
14:07 Approval triggered


evaluation
Score
Error Rate
Process CPU Usage
Process Heap Suspension
Response Time 95th Percentile
Response Time of InvokeAPI Met...
Security Vulnerabilities - Critical
Security Vulnerabilities - High
Security Vulnerabilities - Low
Security Vulnerabilities - Medium

dynatrace DATADOG snyk sonarqube Falco

JMeter HELM argo ...

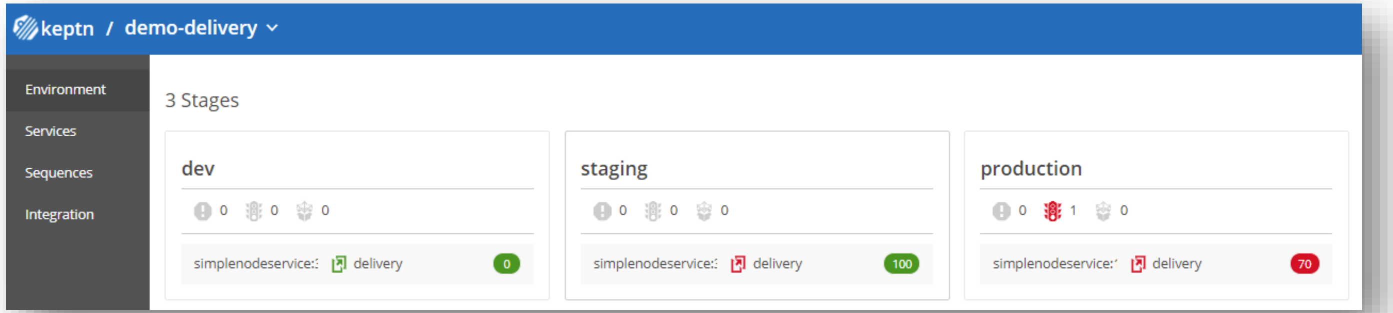
Lets see and explore it in action!



Branch: master demo-delivery / shipyard.yaml

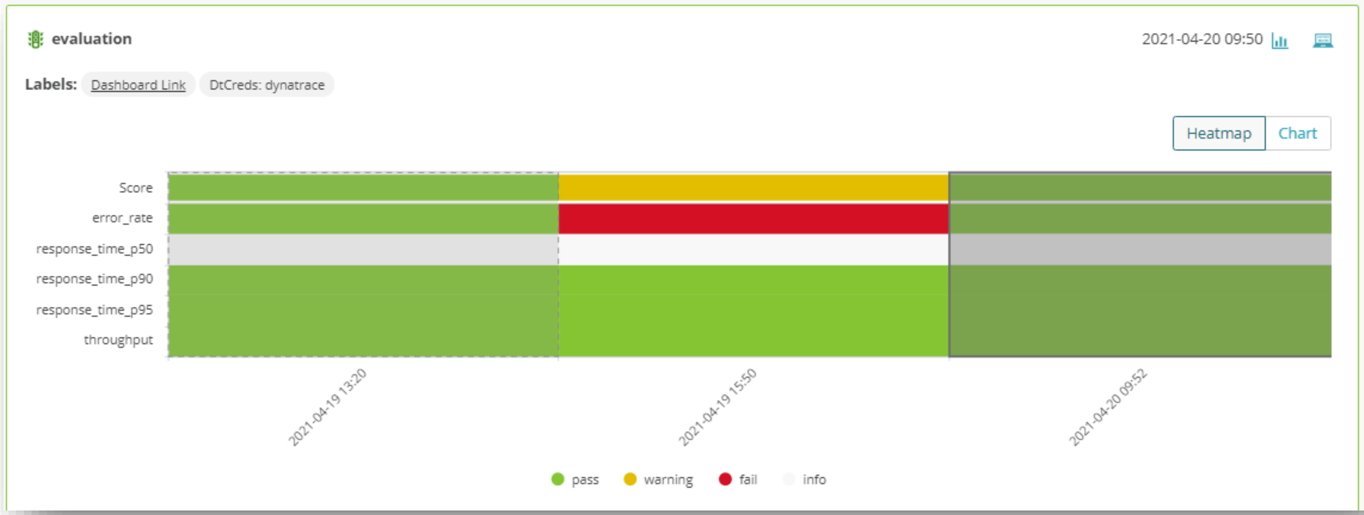
72 lines | 1.6 KiB

```yaml
apiVersion: spec.keptn.sh/0.2.0
kind: Shipyard
metadata:
  name: "shipyard-delivery-simplenode"
spec:
  stages:
  - name: dev
    sequences:
    - name: delivery
      tasks:
      - name: deployment
        properties:
          deploymentstrategy: direct
      - name: test
        properties:
          teststrategy: functional
      - name: evaluation
      - name: release
      - name: approval
        properties:
          pass: automatic
```



3 Stages

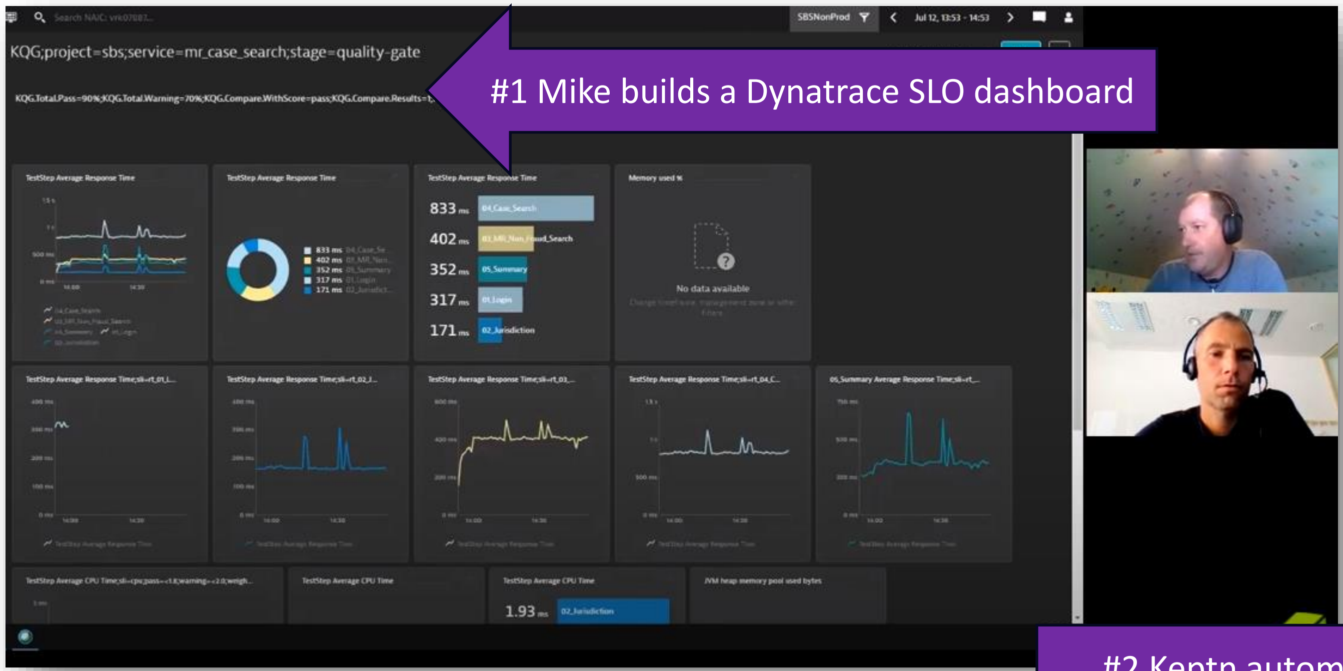
dev	staging	production
0 0 0	0 0 0	0 1 0
simplenodeservice: delivery 0	simplenodeservice: delivery 100	simplenodeservice: delivery 70



Who is adopting Keptn?

#1: Automated Performance Test Analysis using SLOs at NAIC

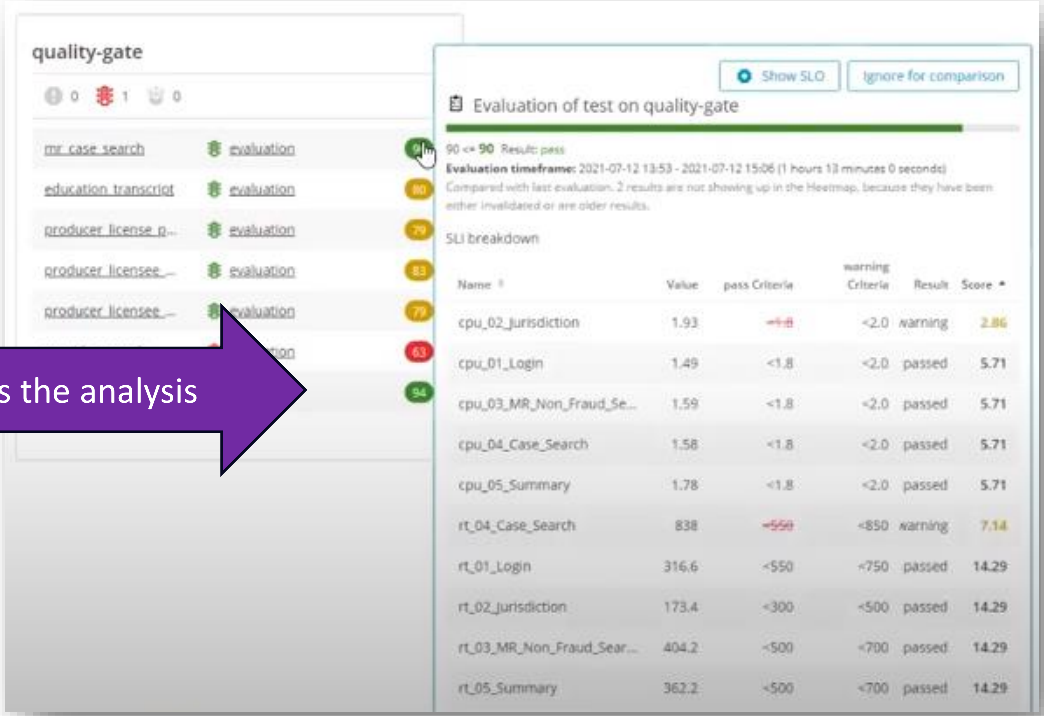
keptn



KQG;project=sbs;service=mr_case_search;stage=quality-gate

KQG.Total.Pass=90%;KQG.Total.Warning=70%;KQG.Compare.WithScore=pass;KQG.Compare.Results=1;

#1 Mike builds a Dynatrace SLO dashboard



quality-gate

mr_case_search	evaluation	9
education_transcript	evaluation	80
producer_license_p...	evaluation	79
producer_licensee_...	evaluation	83
producer_licensee_...	evaluation	79
		63
		94

Show SLO | Ignore for comparison

Evaluation of test on quality-gate

90 <= 90 Result: pass

Evaluation timeframe: 2021-07-12 13:53 - 2021-07-12 15:06 (1 hours 13 minutes 0 seconds)

Compared with last evaluation. 2 results are not showing up in the Heatmap, because they have been either invalidated or are older results.

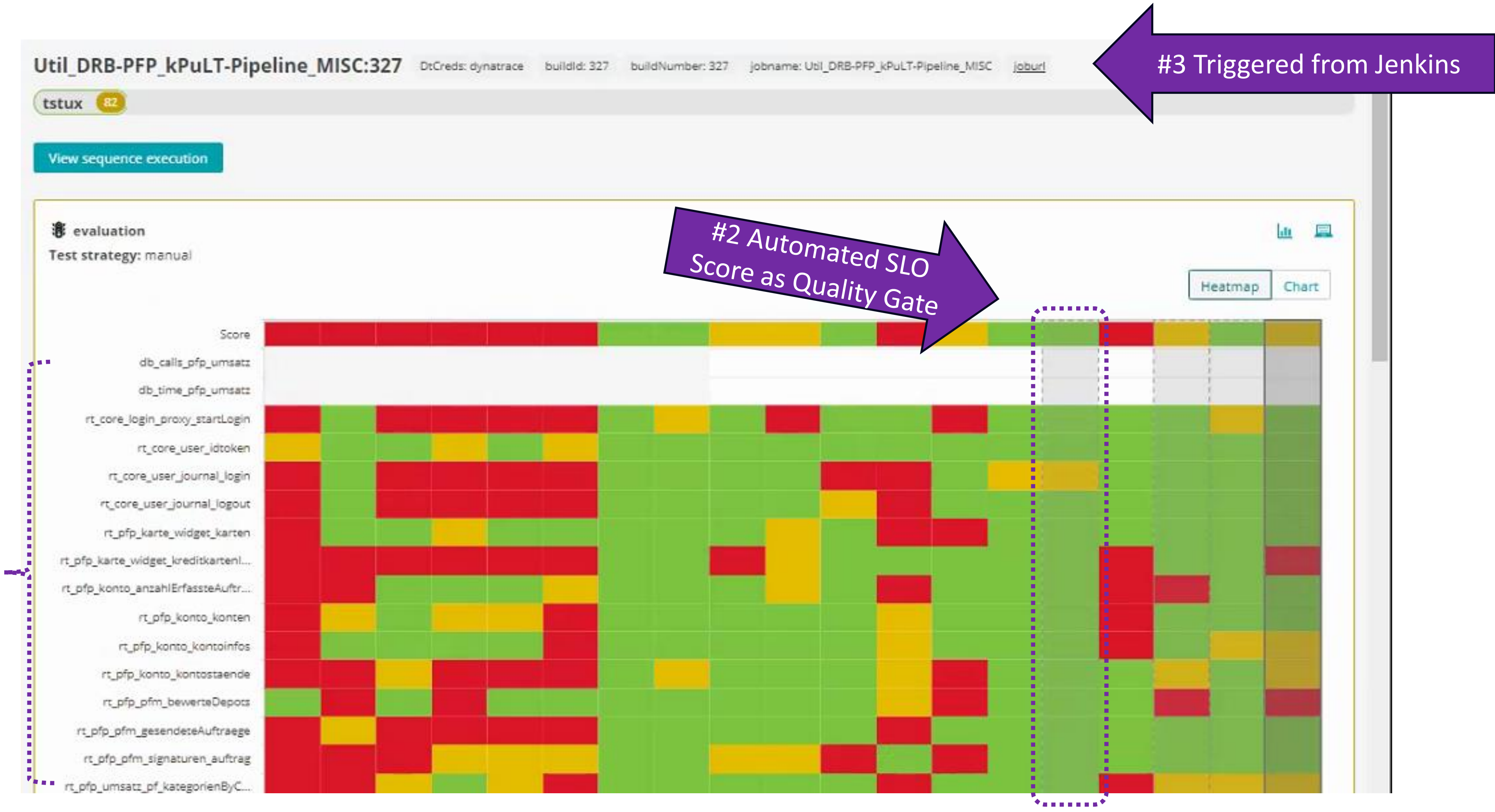
SLI breakdown

Name	Value	pass Criteria	warning Criteria	Result	Score
cpu_02_Jurisdiction	1.93	~~<1.8~~	<2.0	warning	2.86
cpu_01_Login	1.49	<1.8	<2.0	passed	5.71
cpu_03_MR_Non_Fraud_Se...	1.59	<1.8	<2.0	passed	5.71
cpu_04_Case_Search	1.58	<1.8	<2.0	passed	5.71
cpu_05_Summary	1.78	<1.8	<2.0	passed	5.71
rt_04_Case_Search	838	~~<550~~	<850	warning	7.14
rt_01_Login	316.6	<550	<750	passed	14.29
rt_02_Jurisdiction	173.4	<300	<500	passed	14.29
rt_03_MR_Non_Fraud_Sear...	404.2	<500	<700	passed	14.29
rt_05_Summary	362.2	<500	<700	passed	14.29

#2 Keptn automates the analysis

https://www.youtube.com/watch?v=6vd8rtcoV9k&list=PLqt2rd0eew1YFx9m8dBFSiGYSBcDuWG38&index=5&t=2s

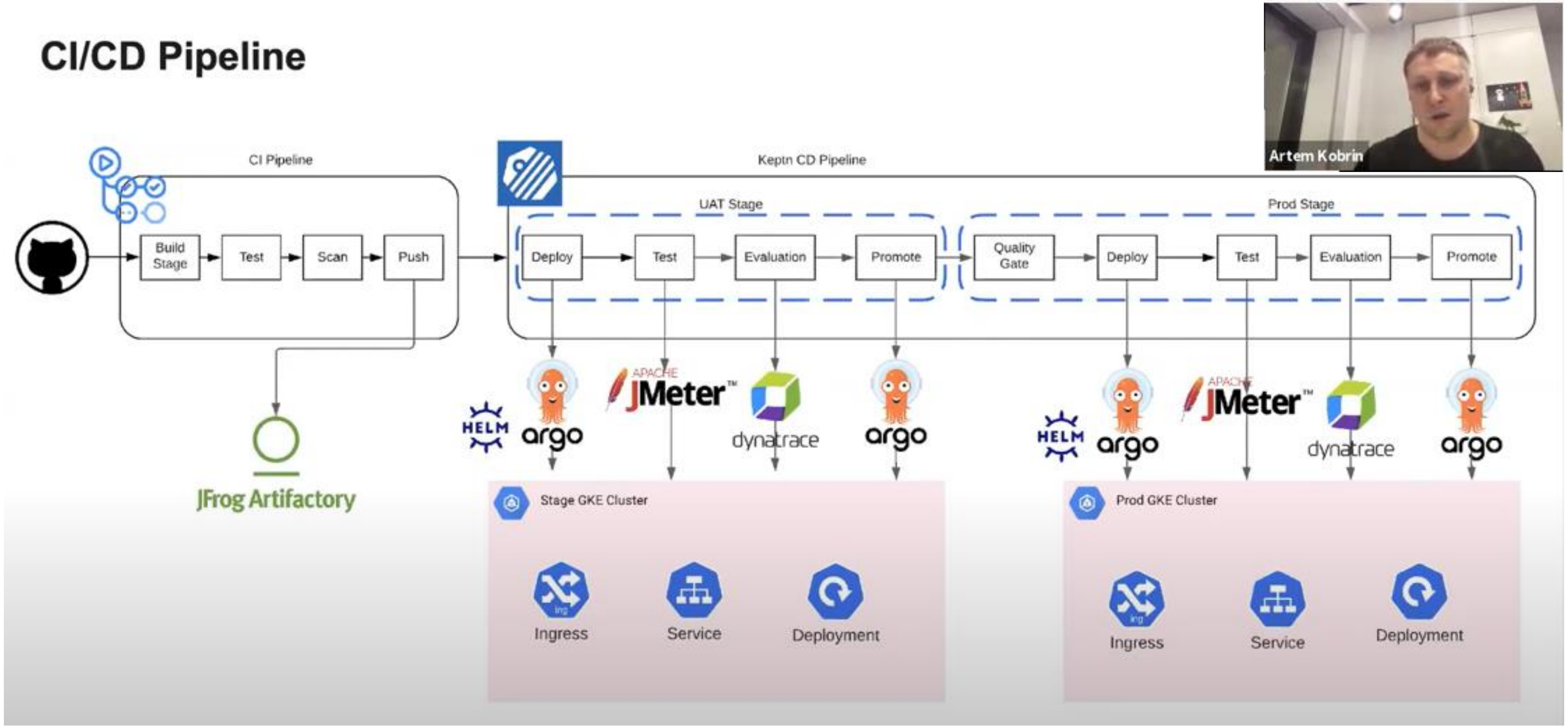
#2: Automated Release Validation of Austrian Online Banking


Util_DRB-PFP_kPuLT-Pipeline_MISC:327
DtCreds: dynatrace
buildId: 327
buildNumber: 327
jobname: Util_DRB-PFP_kPuLT-Pipeline_MISC
joburl
tstux 82
View sequence execution
evaluation
Test strategy: manual
Heatmap
Chart
Score
db_calls_pfp_umsatz
db_time_pfp_umsatz
rt_core_login_proxy_startLogin
rt_core_user_idtoken
rt_core_user_journal_login
rt_core_user_journal_logout
rt_pfp_karte_widget_karten
rt_pfp_karte_widget_kreditkartenl...
rt_pfp_konto_anzahlErfassteAuftr...
rt_pfp_konto_konten
rt_pfp_konto_kontoinfos
rt_pfp_konto_kontostaende
rt_pfp_pfm_bewerteDepots
rt_pfp_pfm_gesendeteAuftraege
rt_pfp_pfm_signaturen_auftrag
rt_pfp_umsatz_pf_kategorienByC...
#3 Triggered from Jenkins
#2 Automated SLO Score as Quality Gate

#1 Release-relevant SLOs defined using YAML

Read the blog: https://medium.com/keptn/keptn-automates-release-readiness-validation-for-austrian-online-banking-software-eaaab7ad7856

#3: Multi-Stage Canary Deployments with Argo Rollouts, JMeter, ... At Volusion



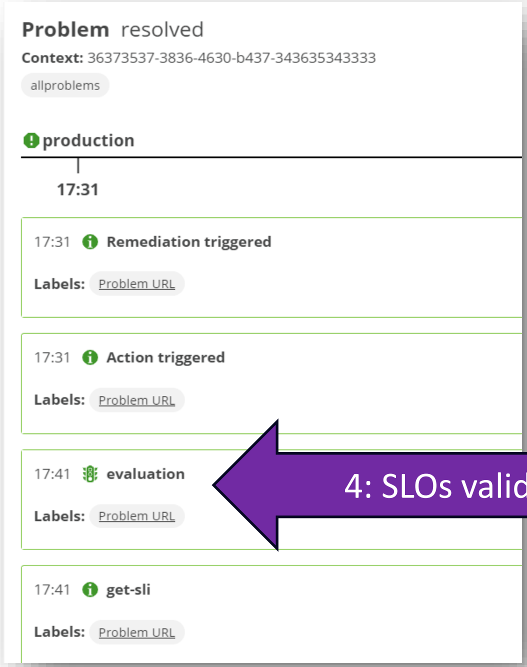
Watch the full demo on Keptn YouTube: https://www.youtube.com/watch?v=21gwtOpgklA

#4: Auto-Remediation of SLA violations at P&G

1: Alert triggers

2: Keptn orchestrates remediation sequence

3: Actions to fix the problem



Problem resolved
Context: 36373537-3836-4630-b437-343635343333
allproblems

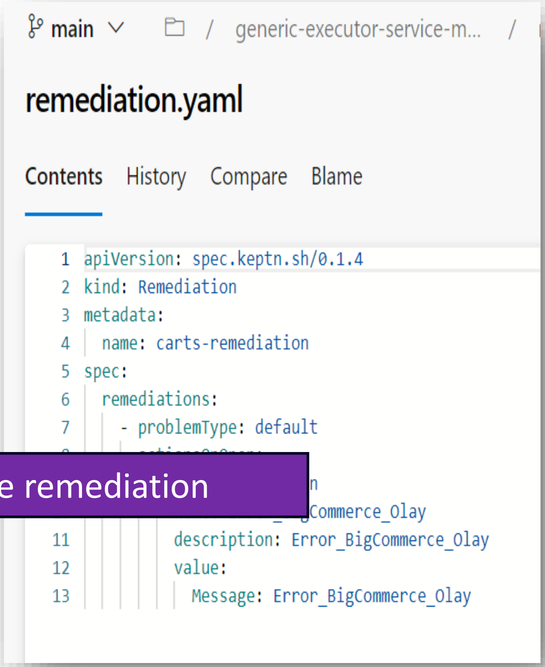
production
17:31

17:31 Remediation triggered
Labels: Problem URL

17:31 Action triggered
Labels: Problem URL

17:41 evaluation
Labels: Problem URL

17:41 get-sli
Labels: Problem URL

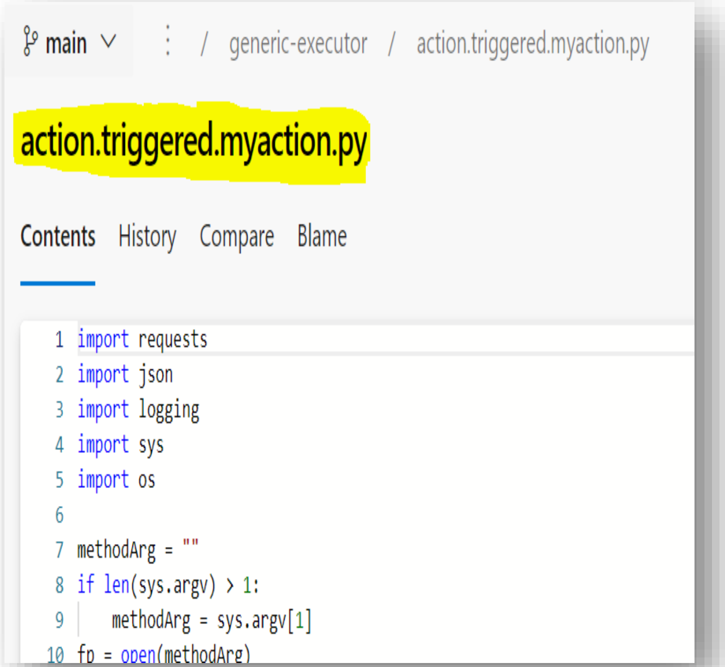


main / generic-executor-service-m... /

remediation.yaml

Contents History Compare Blame

```yaml
apiVersion: spec.keptn.sh/0.1.4
kind: Remediation
metadata:
  name: carts-remediation
spec:
  remediations:
    - problemType: default
      actionsOnOpen:
                                 n
                        _ gCommerce_Olay
          description: Error_BigCommerce_Olay
          value:
            Message: Error_BigCommerce_Olay
```



main / generic-executor / action.triggered.myaction.py

action.triggered.myaction.py

Contents History Compare Blame

```python
import requests
import json
import logging
import sys
import os

methodArg = ""
if len(sys.argv) > 1:
    methodArg = sys.argv[1]
fp = open(methodArg)
```

4: SLOs validate remediation

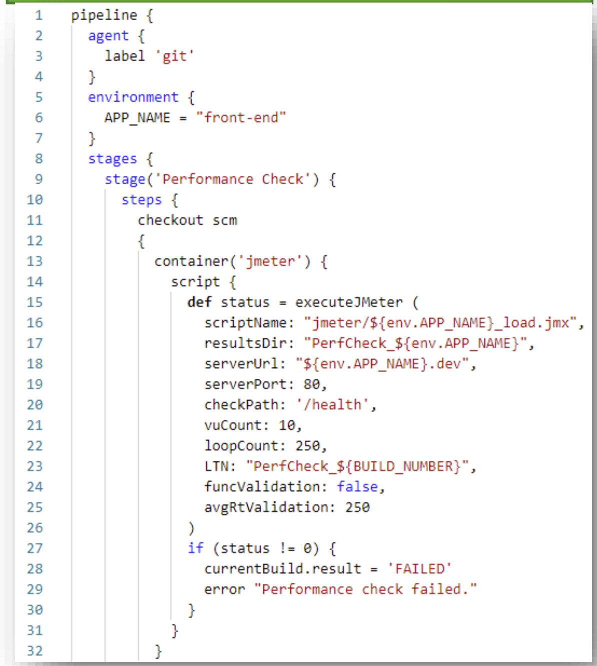
Watch the full demo on Keptn YouTube: https://www.youtube.com/watch?v=UiFTFinalsg

Act 2 – Why we built Keptn

Implementing your own automation in your tool of choice is possible

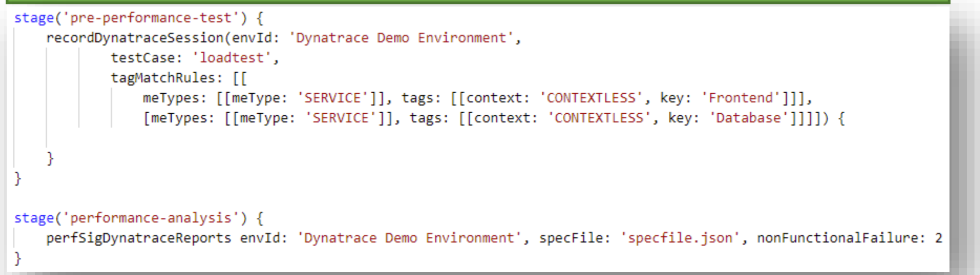
There clearly is no shortage of *„DIY swiss-army knife“* tools to build awesome automation

#1 Execute Test



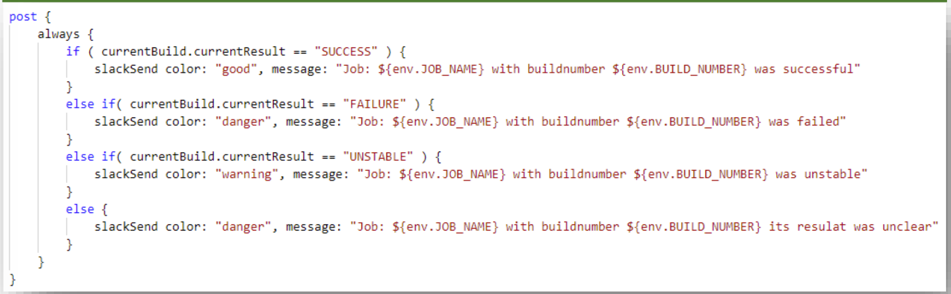
```
pipeline {
  agent {
    label 'git'
  }
  environment {
    APP_NAME = "front-end"
  }
  stages {
    stage('Performance Check') {
      steps {
        checkout scm
        {
          container('jmeter') {
            script {
              def status = executeJMeter (
                scriptName: "jmeter/${env.APP_NAME}_load.jmx",
                resultsDir: "PerfCheck_${env.APP_NAME}",
                serverUrl: "${env.APP_NAME}.dev",
                serverPort: 80,
                checkPath: '/health',
                vuCount: 10,
                loopCount: 250,
                LTN: "PerfCheck_${BUILD_NUMBER}",
                funcValidation: false,
                avgRtValidation: 250
              )
              if (status != 0) {
                currentBuild.result = 'FAILED'
                error "Performance check failed."
              }
            }
          }
```

#2 Add result analysis



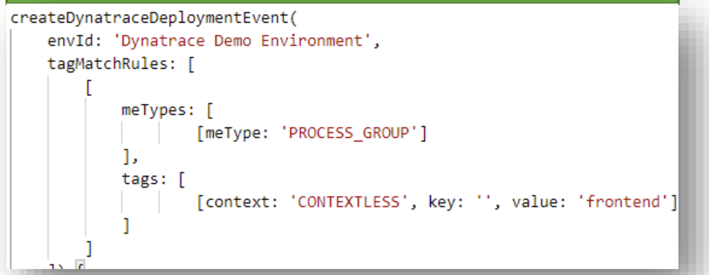
```
stage('pre-performance-test') {
    recordDynatraceSession(envId: 'Dynatrace Demo Environment',
            testCase: 'loadtest',
            tagMatchRules: [[
                meTypes: [[meType: 'SERVICE']], tags: [[context: 'CONTEXTLESS', key: 'Frontend']]],
                [meTypes: [[meType: 'SERVICE']], tags: [[context: 'CONTEXTLESS', key: 'Database']]]]]) {

    }
}

stage('performance-analysis') {
    perfSigDynatraceReports envId: 'Dynatrace Demo Environment', specFile: 'specfile.json', nonFunctionalFailure: 2
}
```

#3 Add results notifications to Slack



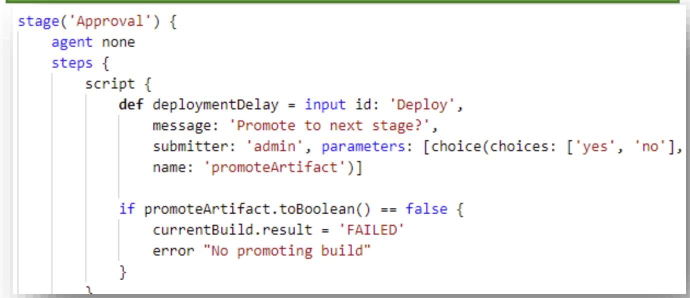
```
post {
    always {
        if ( currentBuild.currentResult == "SUCCESS" ) {
            slackSend color: "good", message: "Job: ${env.JOB_NAME} with buildnumber ${env.BUILD_NUMBER} was successful"
        }
        else if( currentBuild.currentResult == "FAILURE" ) {
            slackSend color: "danger", message: "Job: ${env.JOB_NAME} with buildnumber ${env.BUILD_NUMBER} was failed"
        }
        else if( currentBuild.currentResult == "UNSTABLE" ) {
            slackSend color: "warning", message: "Job: ${env.JOB_NAME} with buildnumber ${env.BUILD_NUMBER} was unstable"
        }
        else {
            slackSend color: "danger", message: "Job: ${env.JOB_NAME} with buildnumber ${env.BUILD_NUMBER} its resulat was unclear"
        }
    }
}
```

#4 Integrate with your APM



```
createDynatraceDeploymentEvent(
    envId: 'Dynatrace Demo Environment',
    tagMatchRules: [
        [
            meTypes: [
                    [meType: 'PROCESS_GROUP']
            ],
            tags: [
                    [context: 'CONTEXTLESS', key: '', value: 'frontend']
            ]
        ]
```

#5 Add approval process



```
stage('Approval') {
    agent none
    steps {
        script {
            def deploymentDelay = input id: 'Deploy',
                message: 'Promote to next stage?',
                submitter: 'admin', parameters: [choice(choices: ['yes', 'no'],
                name: 'promoteArtifact')]

            if promoteArtifact.toBoolean() == false {
                currentBuild.result = 'FAILED'
                error "No promoting build"
            }
```

#6 Add chaos engineering

#7 Add security scan

#8 Add more stages

#9 Add Deployment

But DIY (Do It Yourself) can become very complex and hard to maintain

keptn

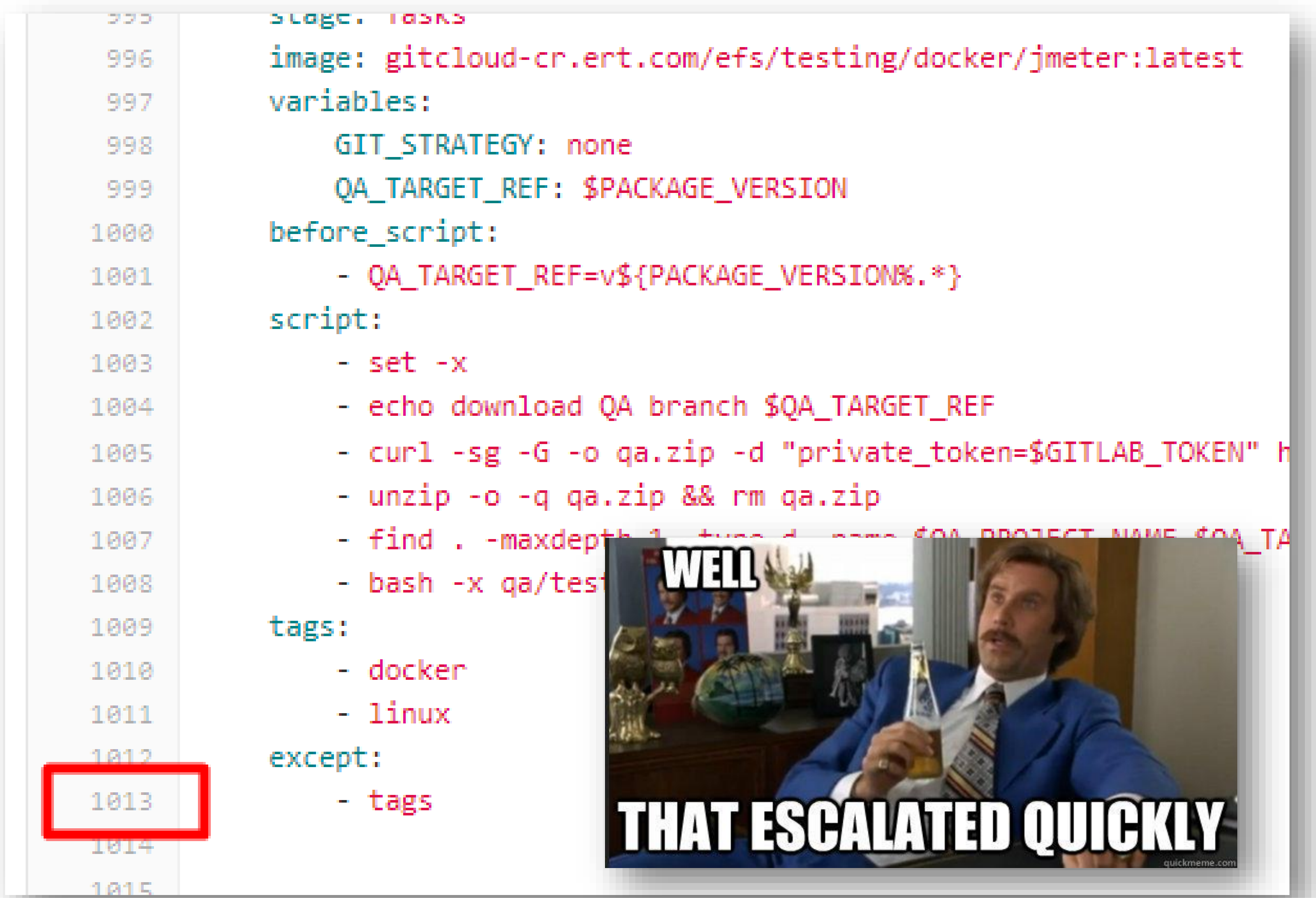
„I am constantly reacting to ‚Pipeline Broken – please fix!'"

ERT
Confidence at every turn™

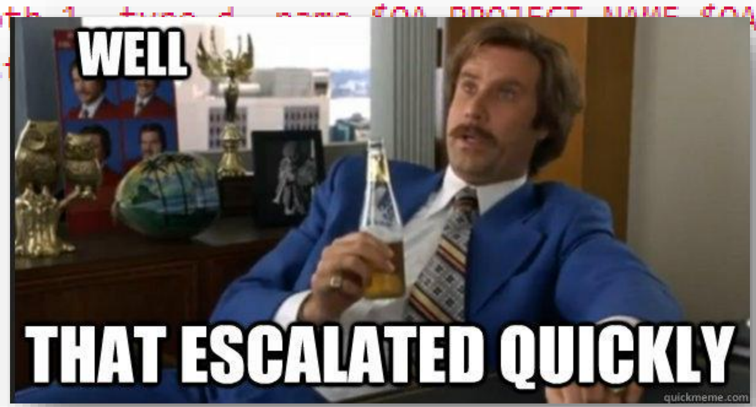
Christian Heckelmann
Senior DevOps Engineer

2800
projects

966
CI/CDs



```
  stage: Tasks
  image: gitcloud-cr.ert.com/efs/testing/docker/jmeter:latest
  variables:
      GIT_STRATEGY: none
      QA_TARGET_REF: $PACKAGE_VERSION
  before_script:
      - QA_TARGET_REF=v${PACKAGE_VERSION%.*}
  script:
      - set -x
      - echo download QA branch $QA_TARGET_REF
      - curl -sg -G -o qa.zip -d "private_token=$GITLAB_TOKEN" h
      - unzip -o -q qa.zip && rm qa.zip
      - find . -maxdepth 1 -type d -name $QA_PROJECT_NAME $QA_TA
      - bash -x qa/test
  tags:
      - docker
      - linux
  except:
      - tags
```



Scaling DIY to many projects often highlights technical debt in your automation

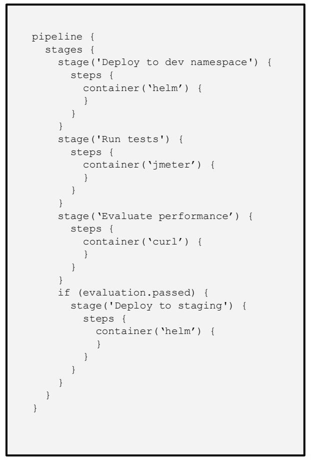
„Onboarding or updating pipelines is manual & error prone!‘“

Dieter Ladenhauf
Senior ACE Engineer

25 services

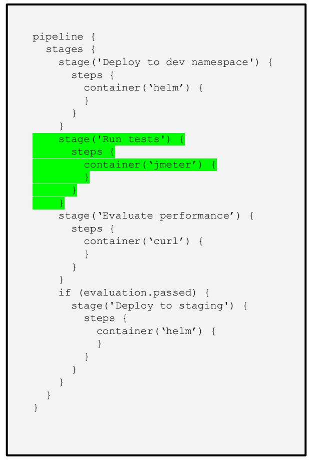
96 workloads

1 Service = 1 Pipeline

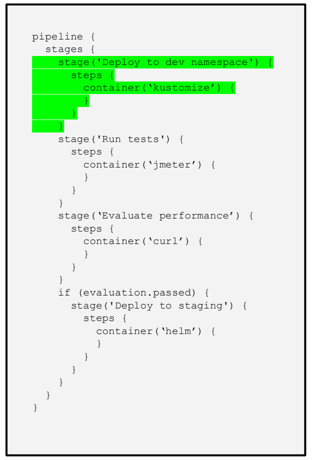


```
pipeline {
  stages {
    stage('Deploy to dev namespace') {
      steps {
        container('helm') {
        }
      }
    }
    stage('Run tests') {
      steps {
        container('jmeter') {
        }
      }
    }
    stage('Evaluate performance') {
      steps {
        container('curl') {
        }
      }
    }
    if (evaluation.passed) {
      stage('Deploy to staging') {
        steps {
          container('helm') {
          }
        }
      }
    }
  }
}
```

1 Project = x Pipelines

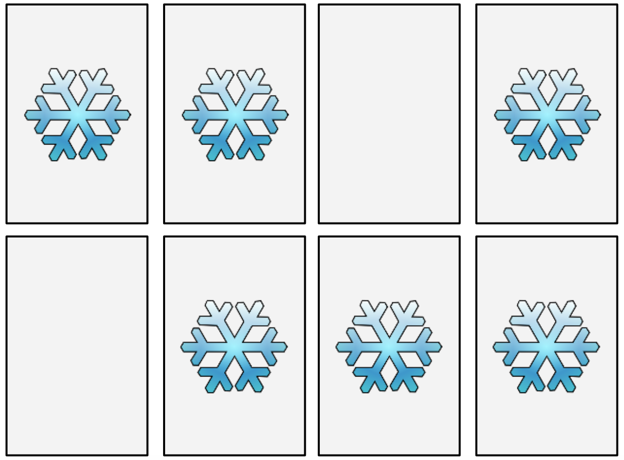


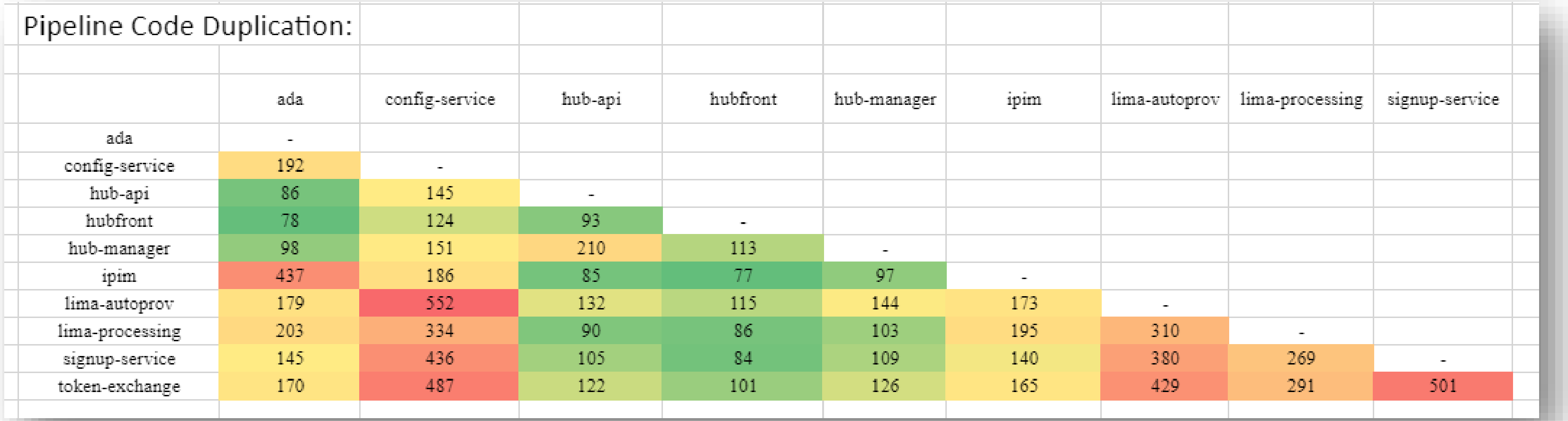
```
pipeline {
  stages {
    stage('Deploy to dev namespace') {
      steps {
        container('helm') {
        }
      }
    }
    stage('Run tests') {
      steps {
        container('jmeter') {
        }
      }
    }
    stage('Evaluate performance') {
      steps {
        container('curl') {
        }
      }
    }
    if (evaluation.passed) {
      stage('Deploy to staging') {
        steps {
          container('helm') {
          }
        }
      }
    }
  }
}
```



```
pipeline {
  stages {
    stage('Deploy to dev namespace') {
      steps {
        container('kustomize') {
        }
      }
    }
    stage('Run tests') {
      steps {
        container('jmeter') {
        }
      }
    }
    stage('Evaluate performance') {
      steps {
        container('curl') {
        }
      }
    }
    if (evaluation.passed) {
      stage('Deploy to staging') {
        steps {
          container('helm') {
          }
        }
      }
    }
  }
}
```

n Teams = n*x Pipelines



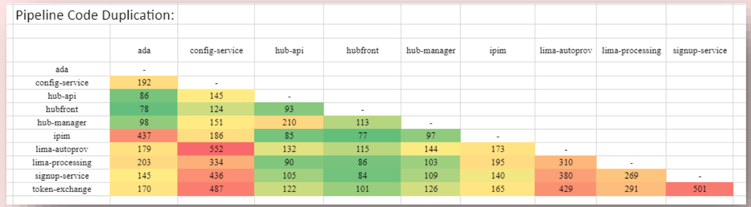


Pipeline Code Duplication:

	ada	config-service	hub-api	hubfront	hub-manager	ipim	lima-autoprov	lima-processing	signup-service
ada	-								
config-service	192	-							
hub-api	86	145	-						
hubfront	78	124	93	-					
hub-manager	98	151	210	113	-				
ipim	437	186	85	77	97	-			
lima-autoprov	179	552	132	115	144	173	-		
lima-processing	203	334	90	86	103	195	310	-	
signup-service	145	436	105	84	109	140	380	269	-
token-exchange	170	487	122	101	126	165	429	291	501

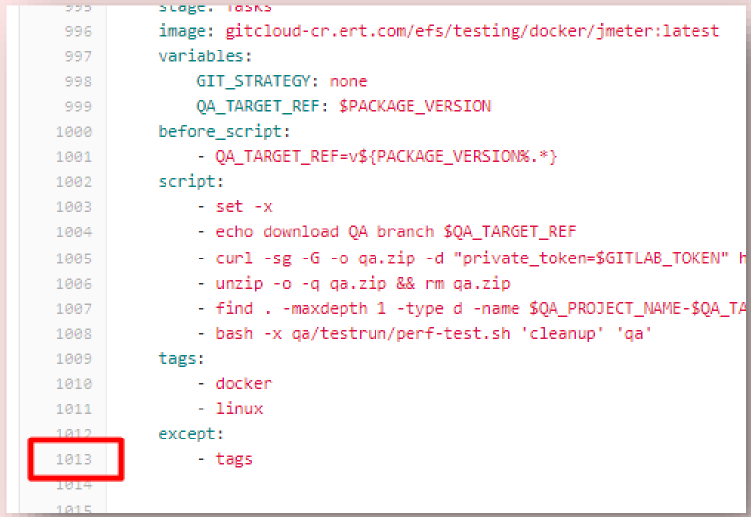
Conclusion ….

Because this doesn't scale!!



Pipeline Code Duplication:

	ada	config-service	hub-api	hubfront	hub-manager	ipim	lima-autoprov	lima-processing	signup-service
ada	-								
config-service	192	-							
hub-api	86	145	-						
hubfront	78	124	93	-					
hub-manager	98	151	210	113	-				
ipim	437	186	85	77	97	-			
lima-autoprov	179	552	132	115	144	173	-		
lima-processing	203	334	90	86	103	195	310	-	
signup-service	145	436	105	84	109	140	380	269	-
token-exchange	170	487	122	101	126	165	429	291	501



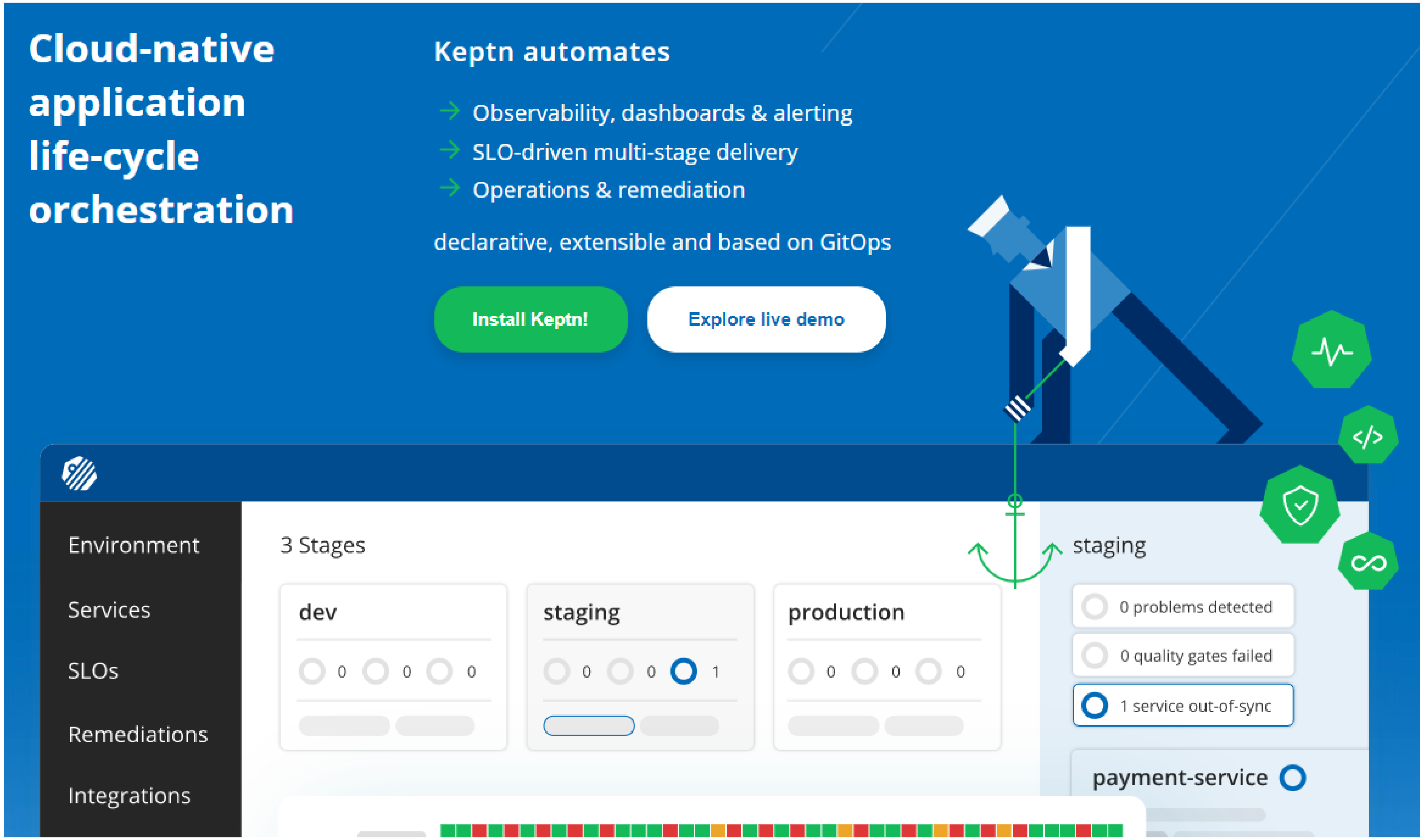
```
 995      stage: Tasks
 996      image: gitcloud-cr.ert.com/efs/testing/docker/jmeter:latest
 997      variables:
 998          GIT_STRATEGY: none
 999          QA_TARGET_REF: $PACKAGE_VERSION
1000      before_script:
1001          - QA_TARGET_REF=v${PACKAGE_VERSION%.*}
1002      script:
1003          - set -x
1004          - echo download QA branch $QA_TARGET_REF
1005          - curl -sg -G -o qa.zip -d "private_token=$GITLAB_TOKEN" h
1006          - unzip -o -q qa.zip && rm qa.zip
1007          - find . -maxdepth 1 -type d -name $QA_PROJECT_NAME-$QA_TA
1008          - bash -x qa/testrun/perf-test.sh 'cleanup' 'qa'
1009      tags:
1010          - docker
1011          - linux
1012      except:
1013          - tags
1014
1015
```

We need a new approach to automation

keptn

Act 3 – How Keptn works!

Oh Keptn, my Keptn!



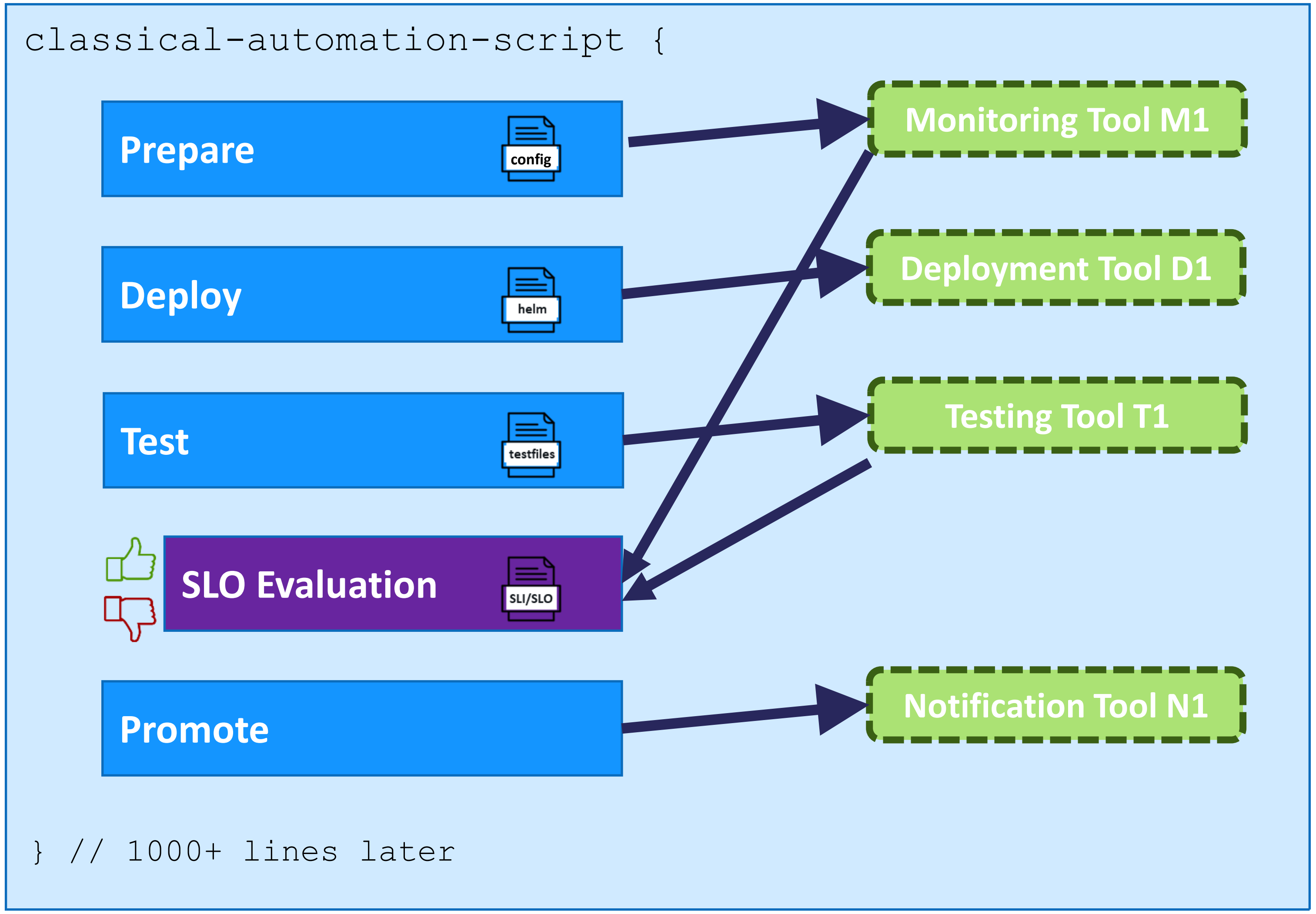
Cloud-native application life-cycle orchestration

Keptn automates

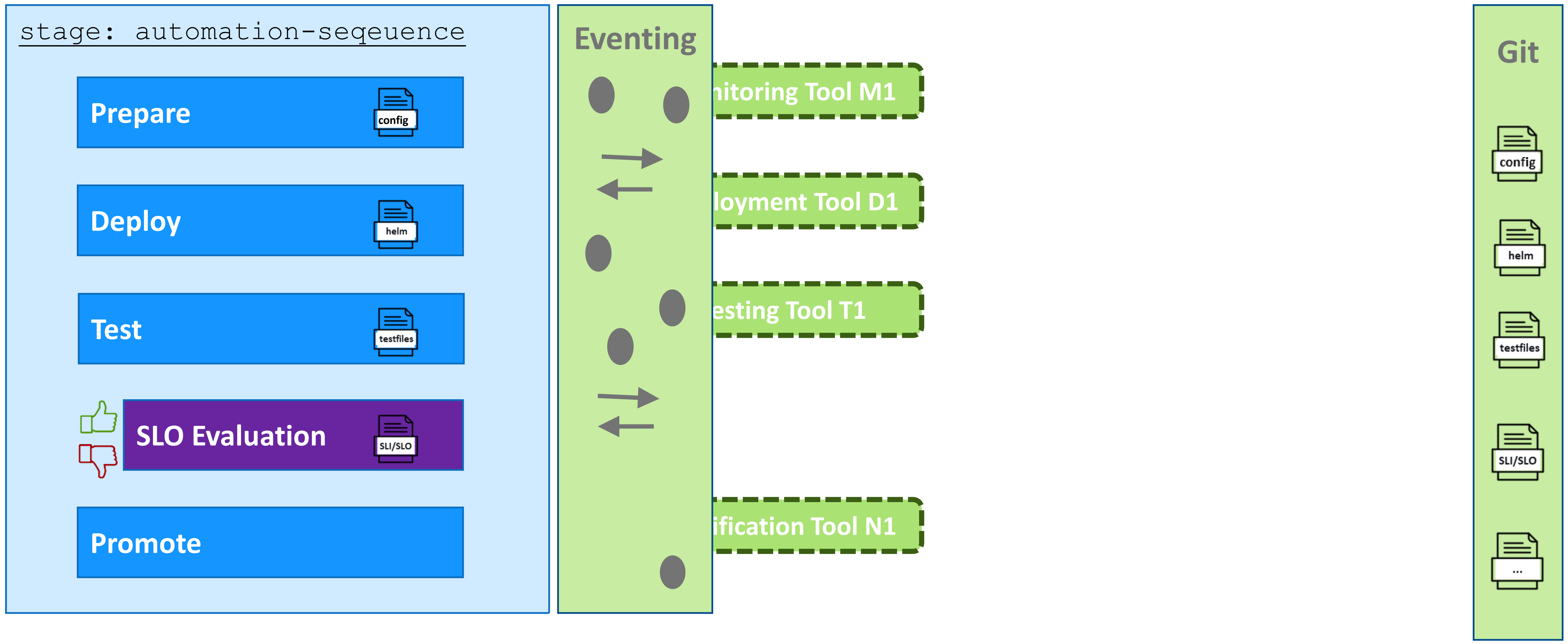
- Observability, dashboards & alerting
- SLO-driven multi-stage delivery
- Operations & remediation

declarative, extensible and based on GitOps

Keptn removes hard-coded dependencies of classical automation approaches

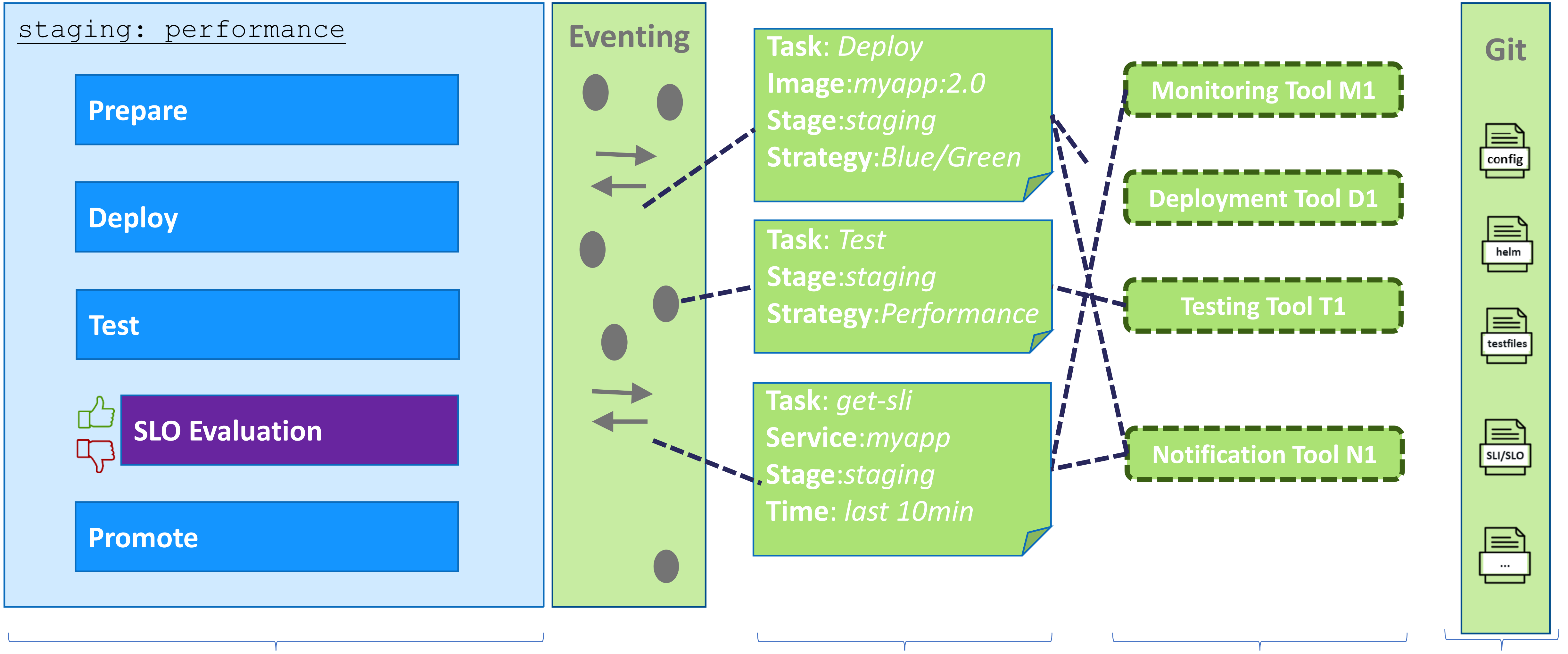


Keptn separates process, tooling and configuration and connects via events



Example #1: automate performance sequence in staging

```
$ keptn trigger performance --stage=staging --image=myapp:2.0
```



Shipyard
Sequence of automation tasks grouped in stages

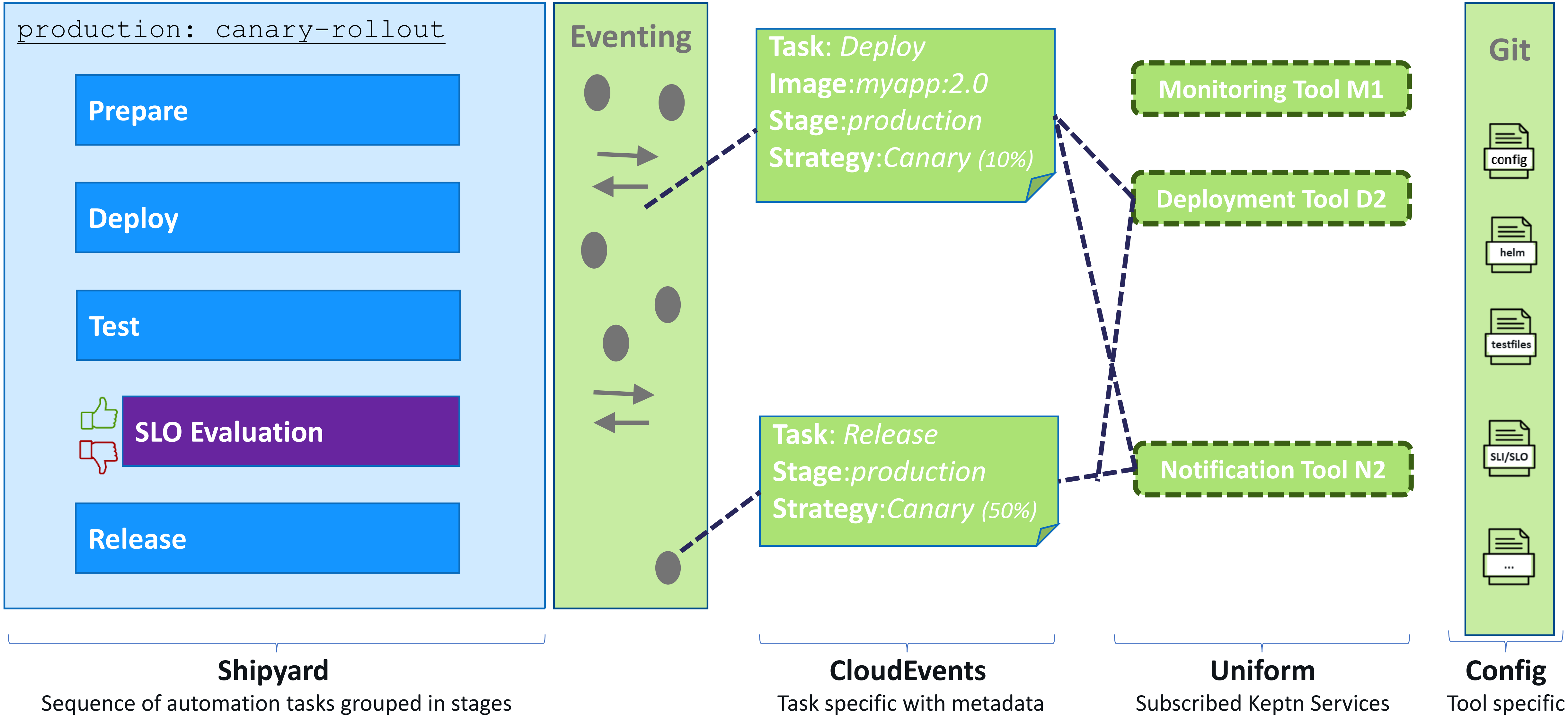
CloudEvents
Task specific with metadata

Uniform
Subscribed Keptn Services

Config
Tool specific

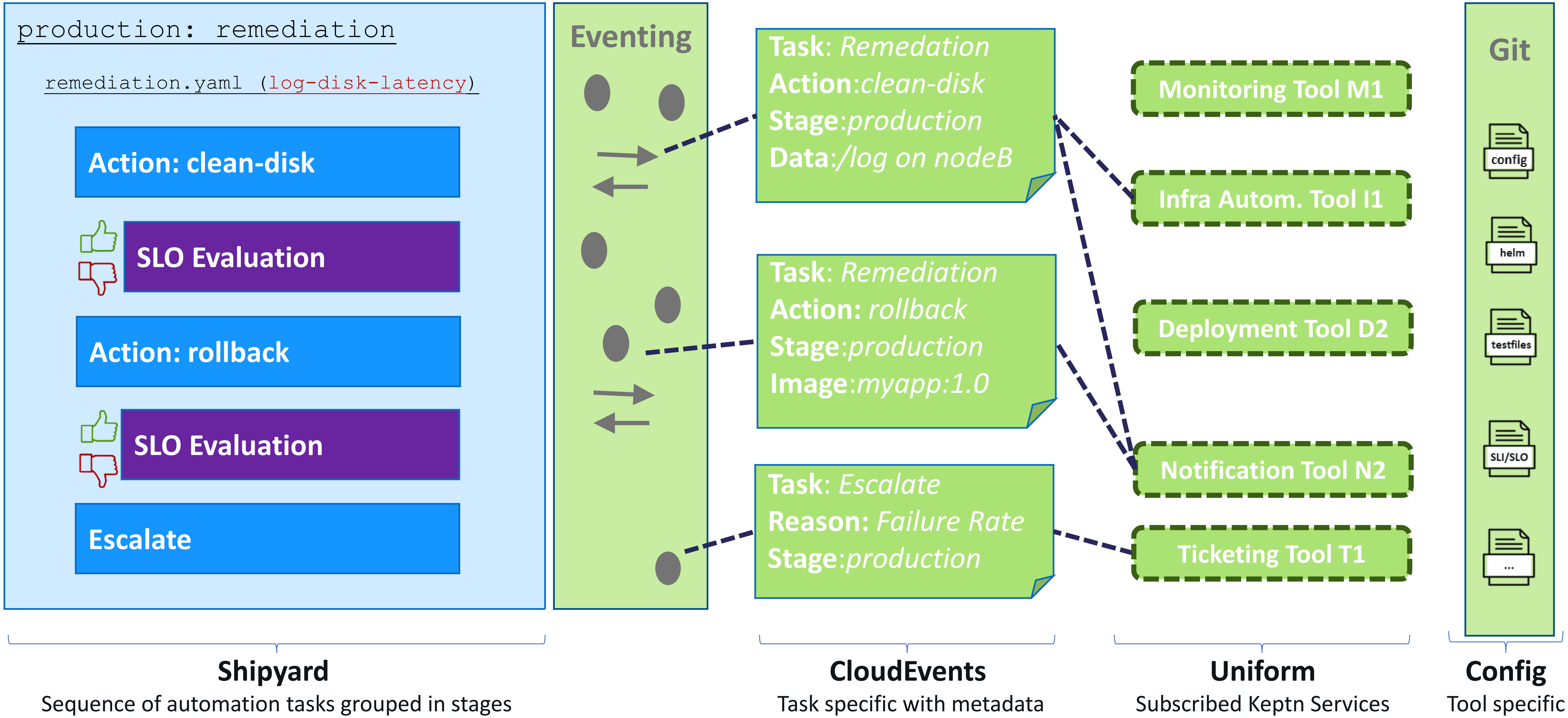
Example #2: automate canary rollout sequence in production

*$ keptn trigger **canary-rollout --stage=production --image=myapp:2.0***



Example #3: remediate production issue

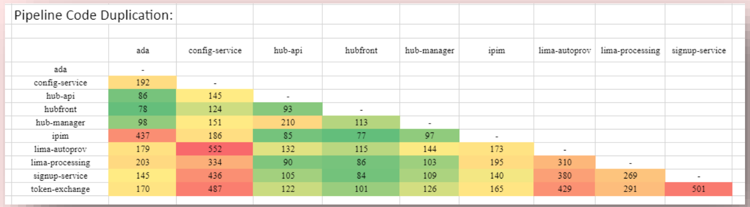
*$ keptn trigger **remediation --stage=production --problem=high-failurerate --rootcause=log-disk-latency***



Keptn brings opinionated cloud native automation to all your projects

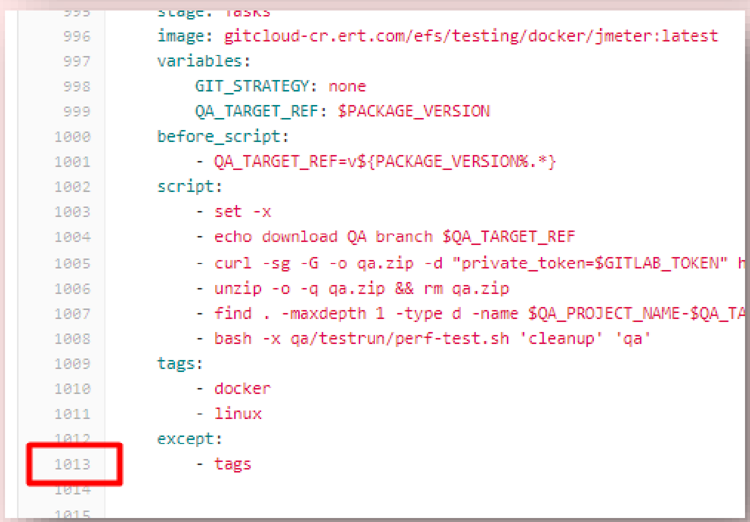
keptn

Reduce your automation's **complexity** by letting keptn orchestrate **declarative, data-driven delivery and ops automation**

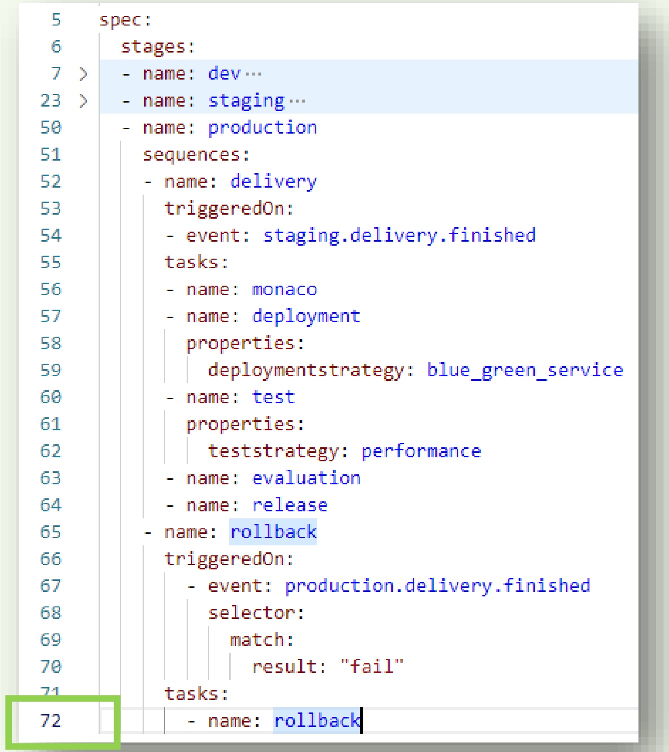


Pipeline Code Duplication:

	ada	config-service	hub-api	hubfront	hub-manager	ipim	lima-autoprov	lima-processing	signup-service
ada	-								
config-service	192	-							
hub-api	86	145	-						
hubfront	78	124	93	-					
hub-manager	98	151	210	113	-				
ipim	437	186	85	77	97	-			
lima-autoprov	179	552	132	115	144	173	-		
lima-processing	203	334	90	86	103	195	310	-	
signup-service	145	436	105	84	109	140	380	269	-
token-exchange	170	487	122	101	126	165	429	291	501



```
995       stage: Tasks
996       image: gitcloud-cr.ert.com/efs/testing/docker/jmeter:latest
997       variables:
998           GIT_STRATEGY: none
999           QA_TARGET_REF: $PACKAGE_VERSION
1000      before_script:
1001          - QA_TARGET_REF=v${PACKAGE_VERSION%.*}
1002      script:
1003          - set -x
1004          - echo download QA branch $QA_TARGET_REF
1005          - curl -sg -G -o qa.zip -d "private_token=$GITLAB_TOKEN" h
1006          - unzip -o -q qa.zip && rm qa.zip
1007          - find . -maxdepth 1 -type d -name $QA_PROJECT_NAME-$QA_TA
1008          - bash -x qa/testrun/perf-test.sh 'cleanup' 'qa'
1009      tags:
1010          - docker
1011          - linux
1012      except:
1013          - tags
1014
1015
```



```
5    spec:
6      stages:
7 >    - name: dev ···
23 >   - name: staging ···
50     - name: production
51       sequences:
52       - name: delivery
53         triggeredOn:
54         - event: staging.delivery.finished
55         tasks:
56         - name: monaco
57         - name: deployment
58           properties:
59             deploymentstrategy: blue_green_service
60         - name: test
61           properties:
62             teststrategy: performance
63         - name: evaluation
64         - name: release
65       - name: rollback
66         triggeredOn:
67           - event: production.delivery.finished
68             selector:
69               match:
70                 result: "fail"
71         tasks:
72           - name: rollback
```

90% less automation code

Separation of process & tool

GitOps: all config in git

SLOs built-in: drive decisions

Connects with all your tools

HELM argo APACHE JMeter

Keptn brings opinionated cloud native automation to all your projects

Reduce your automation's **complexity** by letting keptn orchestrate **declarative, data-driven delivery and ops automation**

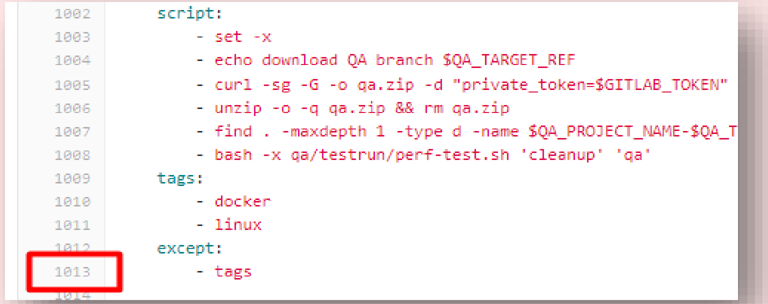
~ **90%** of pipeline code is technical debt

Process, tool and config are **hard coded**

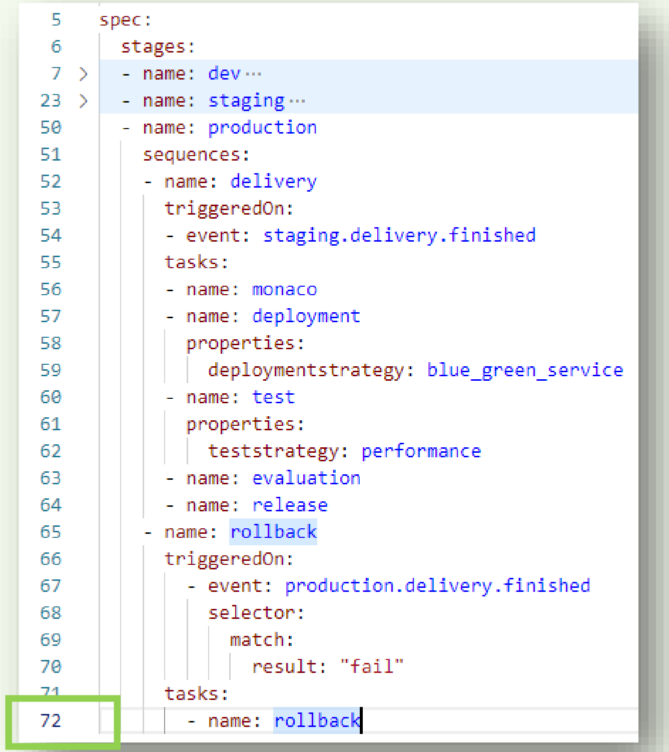
GitOps is often an afterthought

Validation typically happens **manually**

Every new tool is extra work



```
    script:
        - set -x
        - echo download QA branch $QA_TARGET_REF
        - curl -sg -G -o qa.zip -d "private_token=$GITLAB_TOKEN"
        - unzip -o -q qa.zip && rm qa.zip
        - find . -maxdepth 1 -type d -name $QA_PROJECT_NAME-$QA_T
        - bash -x qa/testrun/perf-test.sh 'cleanup' 'qa'
    tags:
        - docker
        - linux
    except:
        - tags
```



```
spec:
  stages:
  - name: dev ···
  - name: staging ···
  - name: production
    sequences:
    - name: delivery
      triggeredOn:
      - event: staging.delivery.finished
      tasks:
      - name: monaco
      - name: deployment
        properties:
          deploymentstrategy: blue_green_service
      - name: test
        properties:
          teststrategy: performance
      - name: evaluation
      - name: release
    - name: rollback
      triggeredOn:
        - event: production.delivery.finished
          selector:
            match:
              result: "fail"
      tasks:
        - name: rollback
```

90% less automation code

Separation of process & tool

GitOps: all config in git

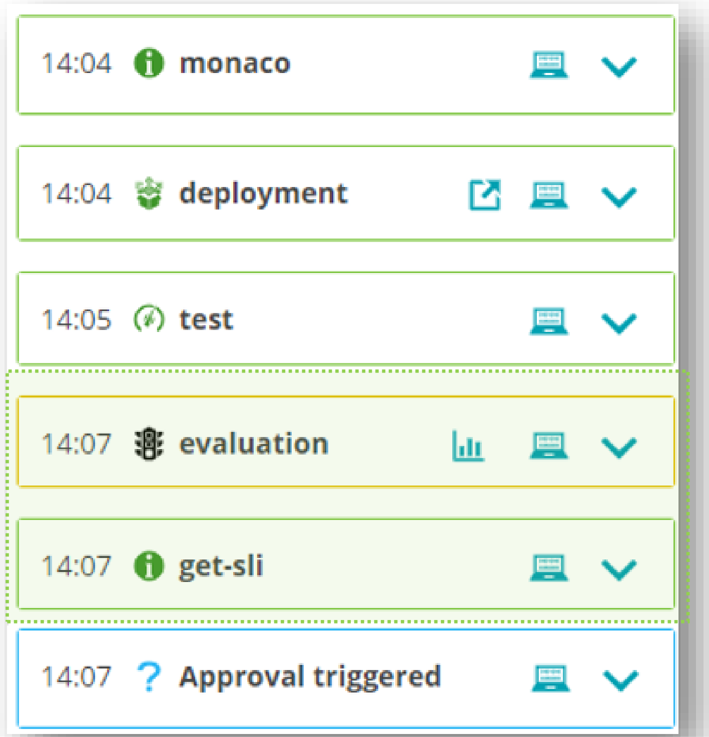
SLOs built-in: drive decisions

Connects with all your tools

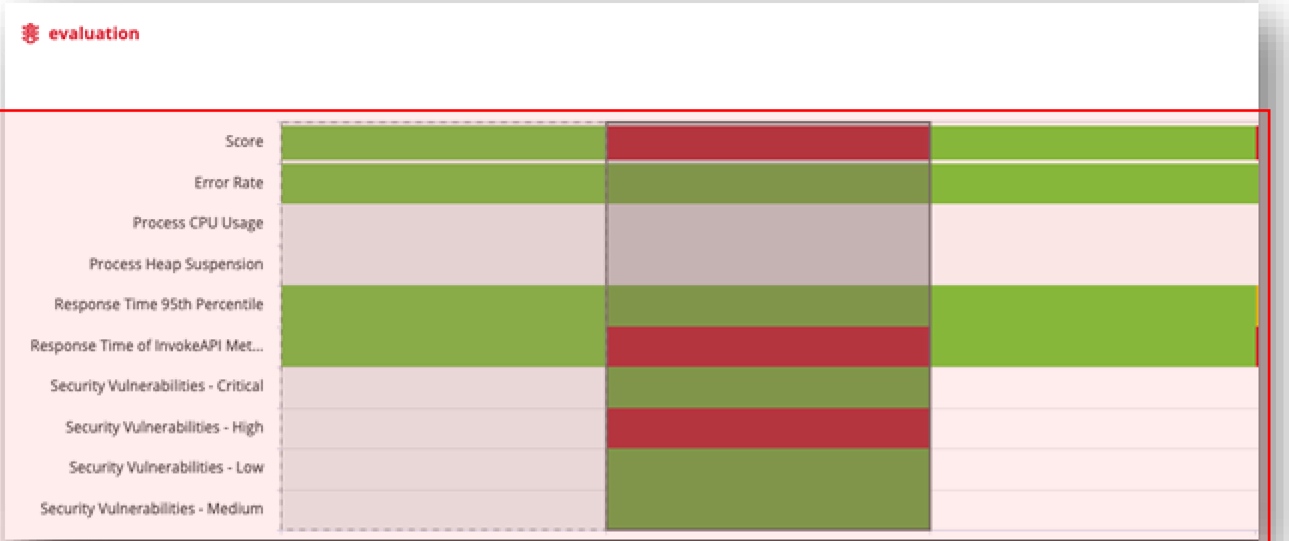
Most start with integrating SLO-Evaluation – then scale out to more use cases

triggers an automation sequence

orchestrates monitoring config, **deployment**, **test** execution, **SLO** evaluation & **remediation**



14:04 monaco
14:04 deployment
14:05 test
14:07 evaluation
14:07 get-sli
14:07 Approval triggered



...

Big Thanks to Dynatrace for driving innovation for the DevOps & SRE community

keptn

Speed

Increase Speed of App Delivery by 75%

Eliminate toil through **automation** (monitoring, test evaluation, remediation)

Quality

Improve Confidence with 50% Better Release Quality

Increase quality and resiliency with **Shift-left, Shift-right**

Collaboration

100% Better Collaboration between Dev & Ops

One platform, single source of truth, **common language** of SLOs

Scale

Scale DevOps Enterprise-wide

Self-service models, **Automation, AI-driven** decision making

Oh Keptn, my Keptn
A data/observability driven way to DevOps & SRE automation

Andreas Grabner

DevOps Activist at Dynatrace

DevRel for Keptn

@grabnerandi, https://www.linkedin.com/in/grabnerandi

Follow us @keptnProject

Star us @ https://github.com/keptn/keptn

Slack Us @ https://slack.keptn.sh

We are a Cloud Native Computing Foundation Sandbox project.