

# Performance as a Self-Service

*Based on SLIs/SLOs with Keptn*



# LTB 2021

## Load Testing & Benchmarking



**Andreas Grabner**

DevOps Activist at Dynatrace  
DevRel for Keptn

@grabnerandi

<https://www.linkedin.com/in/grabnerandi>



Visit us @ <https://keptn.sh>

Follow us @keptnProject

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

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



# My Demo Today: Performance as a Self-Service with Keptn, JMeter & Dynatrace



„Hey Keptn, my latest app runs on <http://myapp.domain> and is monitored by Dynatrace/Prometheus/APM XYZ!  
Please execute **my performance workload** using **JMeter/Neotys/Gatling/Tool XYZ** for me against that app.  
Once done, analyze my **SLO** score based on my SLIs retrieved from my **Test and Monitoring tool!**“

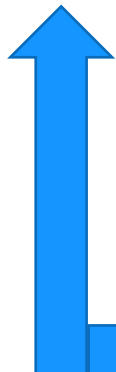
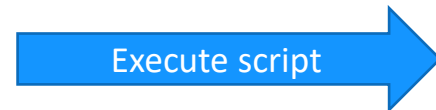
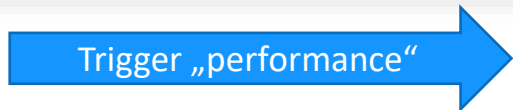


📄 Evaluation of performance test on performance

---

75 <= **86** < 90 Result: **warning**

Evaluation timeframe: 2020-12-08 22:07 - 2020-12-08 22:14 (6 minutes 38 seconds)



### Performance Workload

Script: myscript.jmx  
 Data: mydatafile.txt  
 VUCount: 10  
 ServerUrl: <http://myapp.domain>  
 Dynatrace Token & URL

SLO Validation based on SLIs from Testing and/or Monitoring tools



# Why Keptn?

And not DIY (Do It Yourself)



# DevOps & SREs have to automate the following tasks through pipelines

Many automation tasks related to performance engineering

Deployment

Blue/Green

Canary

Feature Flagging

Performance Engineering

API Test

Functional Test

Chaos Testing

Configure Monitoring

Quality Gates

Security & Vulnerability Scans

Integrate Approval Process

Chat Notifications

Setup Alerting

SLO Monitoring

SLA Validation

Rollbacks

Auto Remediation



# But, building this makes pipelines complex and hard to to maintain!

*„I am constantly reacting to  
„Pipeline Broken – please fix!““*



**Christian Heckelmann**  
Senior DevOps Engineer

**2800**  
projects

**966**  
CI/CDs

```
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/test
1009 tags:
1010     - docker
1011     - linux
1012 except:
1013     - tags
1014
1015
```





# Often leads to duplicated pipeline code -> hard to maintain, extend & optimize

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



**Dieter Ladenhauf**  
Senior ACE Engineer

**25**  
services

**96**  
workloads

**1 Service =  
1 Pipeline**

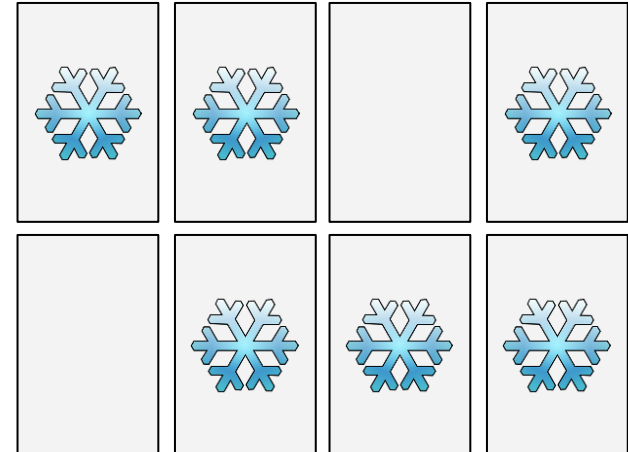
```
pipeline {
  stages {
    stage('Deploy to dev namespace') {
      steps {
        container('helm') {
        }
      }
    }
    stage('Run tests') {
      steps {
        container('jmeter') {
        }
      }
    }
    stage('Evaluate performance') {
      steps {
        container('curl') {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

**n Teams = n\*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 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



# Keptn brings opinionated cloud native automation to all of your projects

Reduce your pipeline'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	sigmap-service
ada	-								
config-service	192	-							
hub-api	86	145	-						
hubfront	78	124	93	-					
hub-manager	93	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	-	
sigmap-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
- Connects with all your tools





# Keptn helps you LEVEL-UP your automation with new use cases such as ...

CD



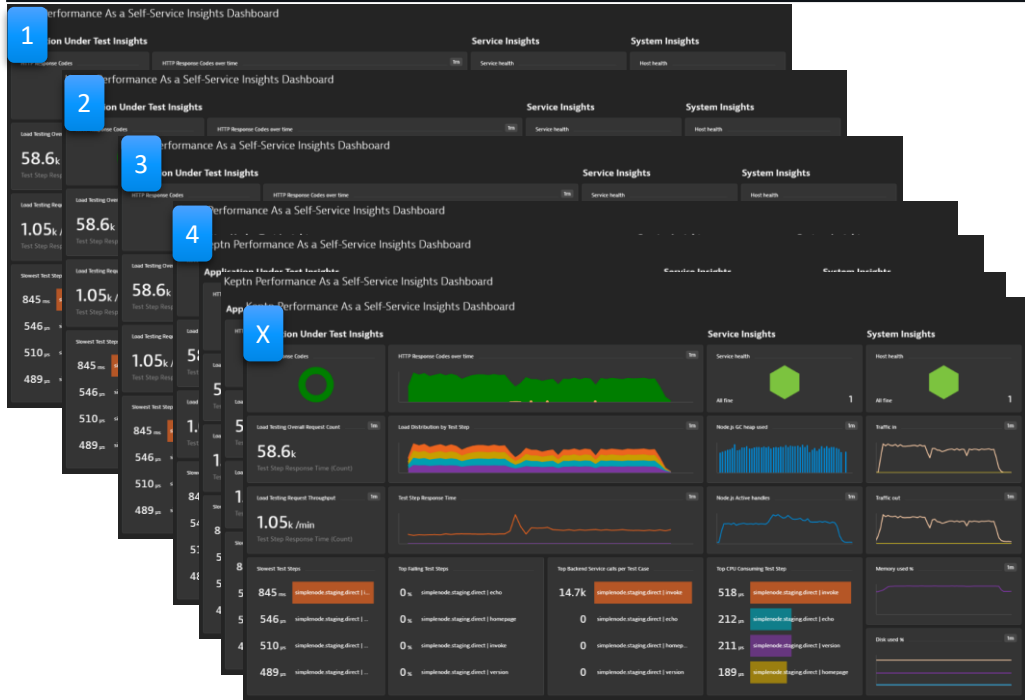
~30-60min

~1min

P  
E  
R  
F  
O  
R  
M  
A  
N  
C  
E

as  
Self-  
Svc

Instead of **manually** test execution and report based analysis



Keptn **automates** test execution and SLO-based evaluation





# Keptn built on Core SRE Practices

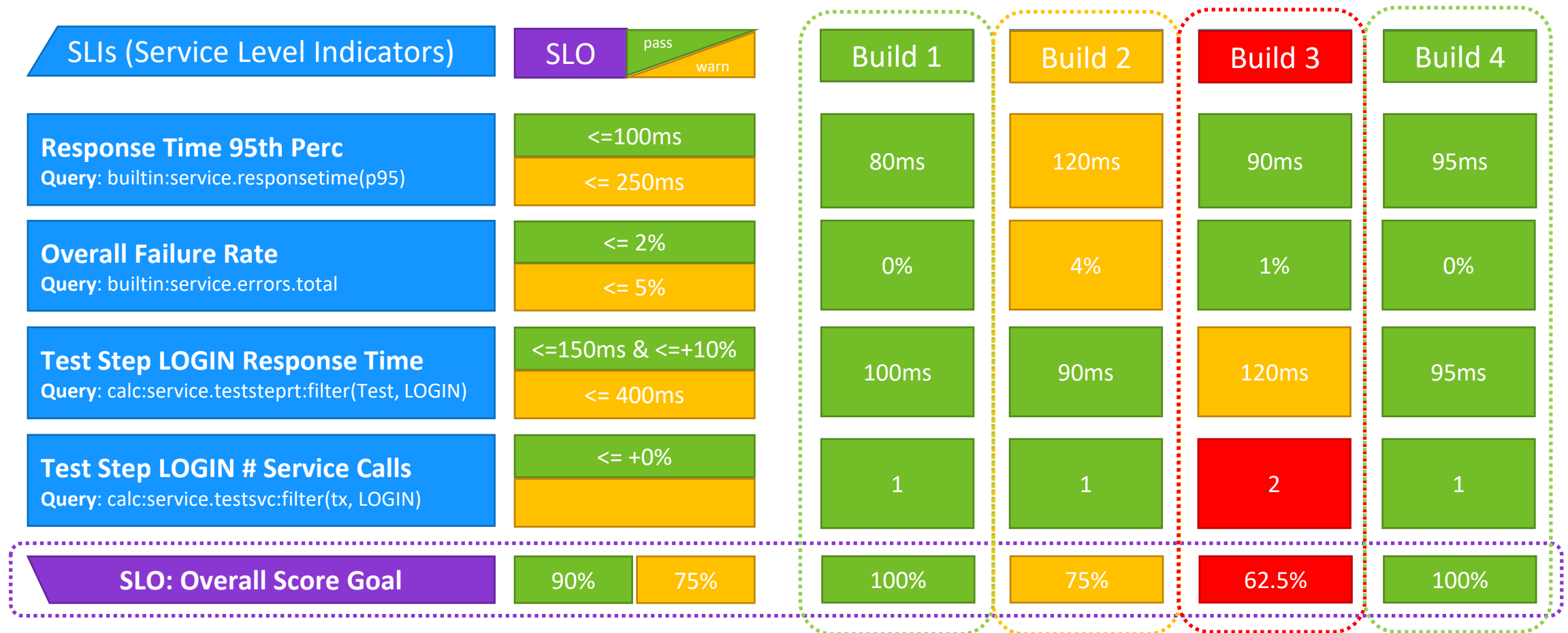
SLIs (Service Level Indicators) & SLOs (Service Level Objectives)

For automated Validation



# Explained: Keptn uses SLIs/SLOs at the core of the result evaluation

```
$ keptn send event start-evaluation myproject myservice starttime=build4_teststart endtime=build4_testsend
```





# Keptn's SLI/SLO-based evaluation can be extended to any data source!

## SLIs defined per SLI Provider as YAML

SLI Provider specific queries, e.g: Dynatrace Metrics Query

```

indicators:
  error_rate:      "builtin:service.errors.total.count:merge(0):avg"
  count_dbcalls:  "calc:service.toptestdbcalls:merge(0):sum"
  jvm_memory:     "builtin:tech.jvm.memory.pool.committed:merge(0):sum"

```

## SLOs defined on Keptn Service Level as YAML

List of objectives with fixed or relative pass & warn criteria

```

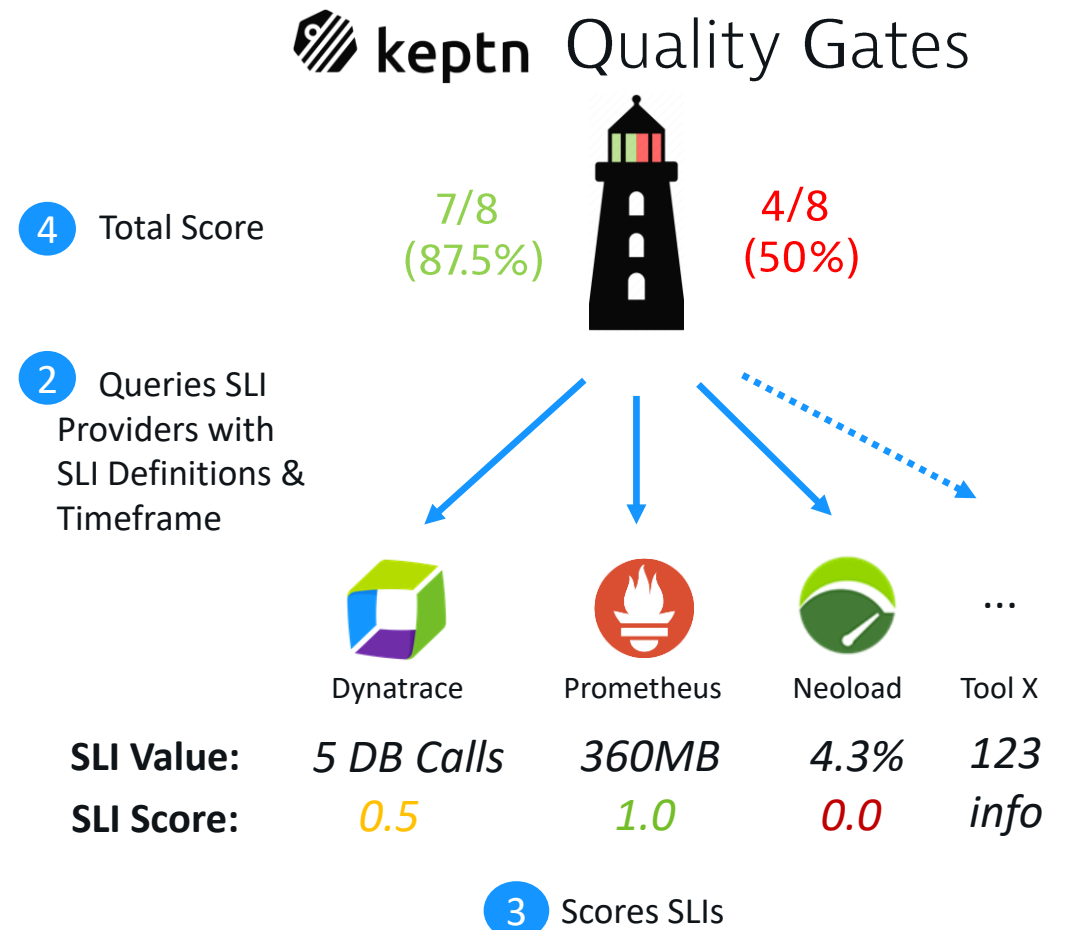
objectives:
- sli: error_rate
  pass:
  - criteria:
    - "<=1" # We expect a max error rate of 1%
- sli: jvm_memory
- sli: count_dbcalls
  pass:
  - criteria:
    - "+=2%" # We allow a 2% increase in DB Calls to previous runs
  warning:
  - criteria:
    - "<=10" # We expect no more than 10 DB Calls per TX
total_score:
  pass: "90%"
  warning: "75%"

```

```

$ keptn start-evaluation 30m myservice sli.yaml slo.yaml

```





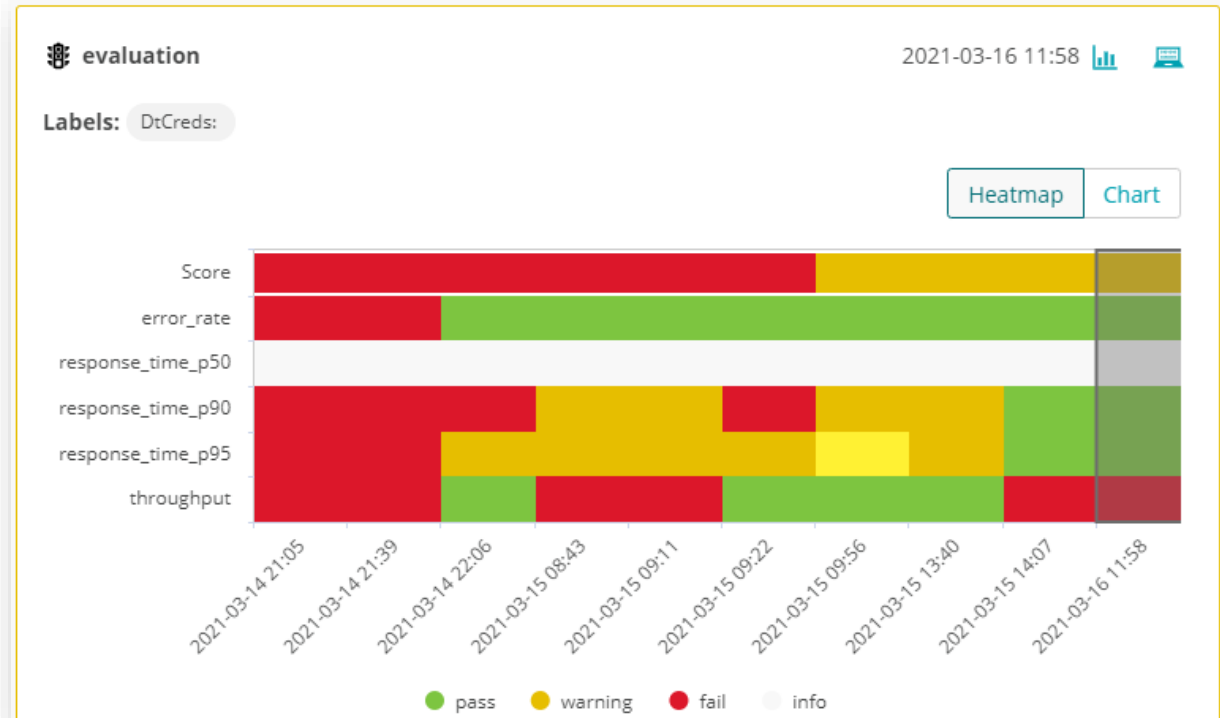
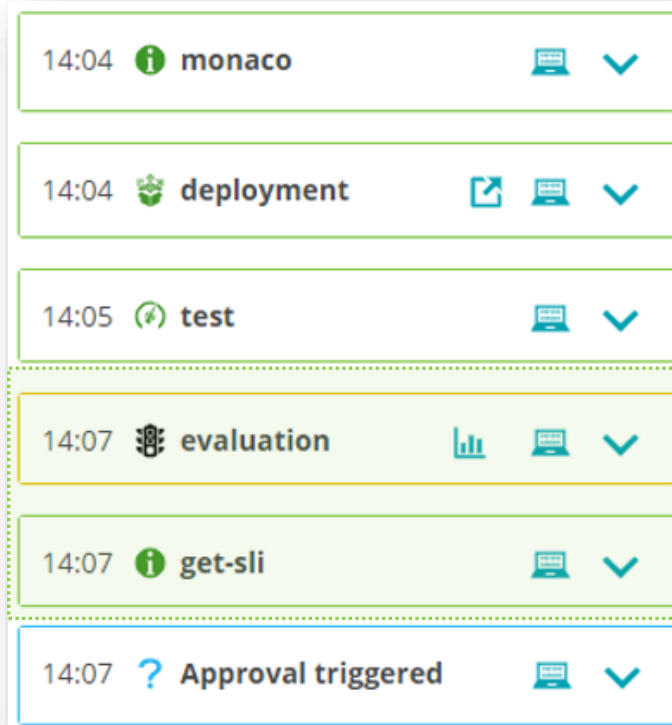
# Integrate Keptn through a simply API Call from your existing automation tool



triggers an automation sequence



orchestrates monitoring config, test execution and SLO evaluation



# Who is using it?

What else can I do? How can I get started?



## Great testimonials from Taras

---



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](#)



# 3 recent Keptn User Group Stories showing the value Keptn brings



**Sumit Nagal**  
Principal Engineer



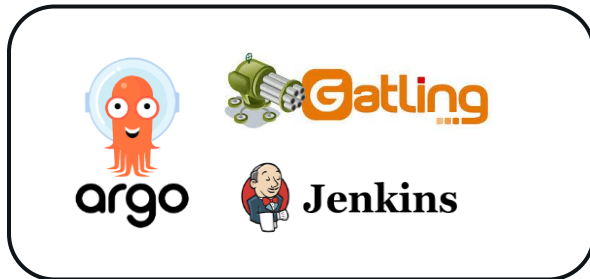
**Roman Ferstl**  
Managing Director



**Christian Heckelmann**  
Senior DevOps Engineer



Tool stack



SLO-based Quality Gate Automation

Performance & Resilience Test Automation

Scaling SRE through Delivery Automation

**50+**

Continuous SLO Validations

**15x**

Perf tests

**10x**

tested apps

**~60**

Services

**14**

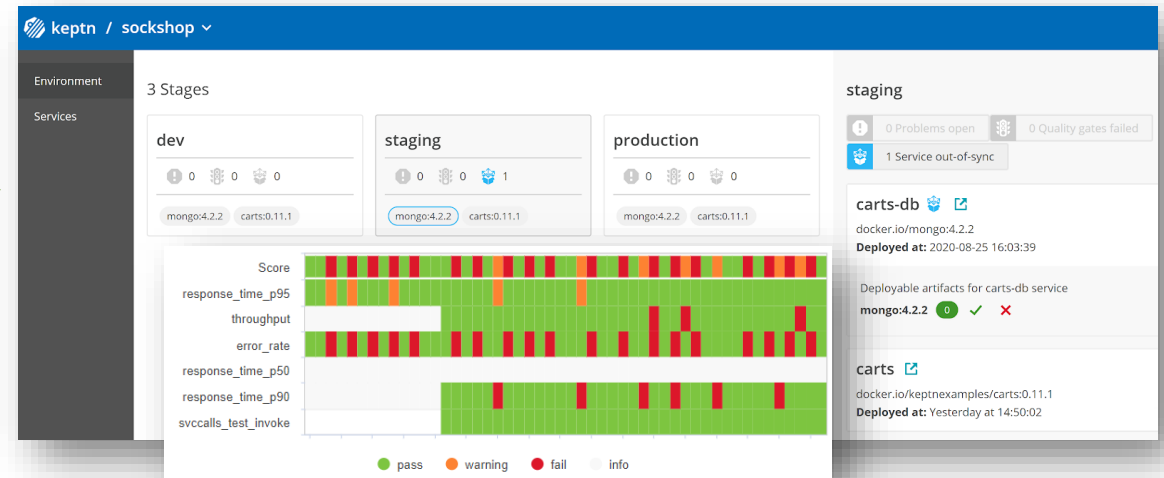
Stages Confidential



# For the newbies - Keptn: Data-Driven Delivery & Operations Automation



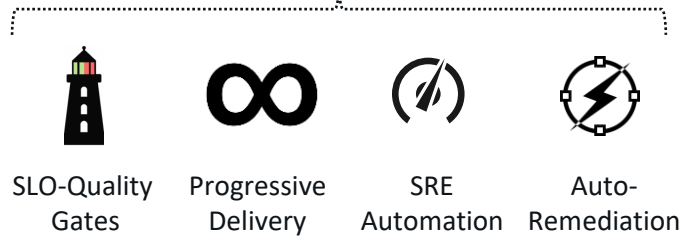
automates configuration and provides self-service for



through event-driven process orchestration based on



pick your use case

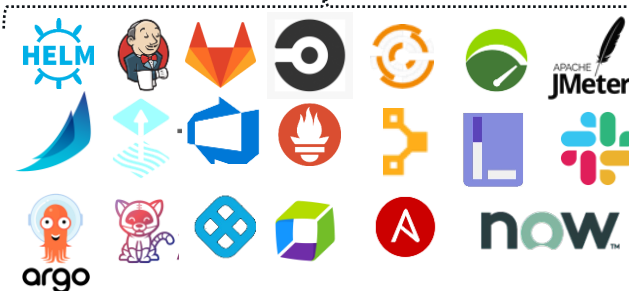


bring your configuration

You (Dev/Ops/SRE)



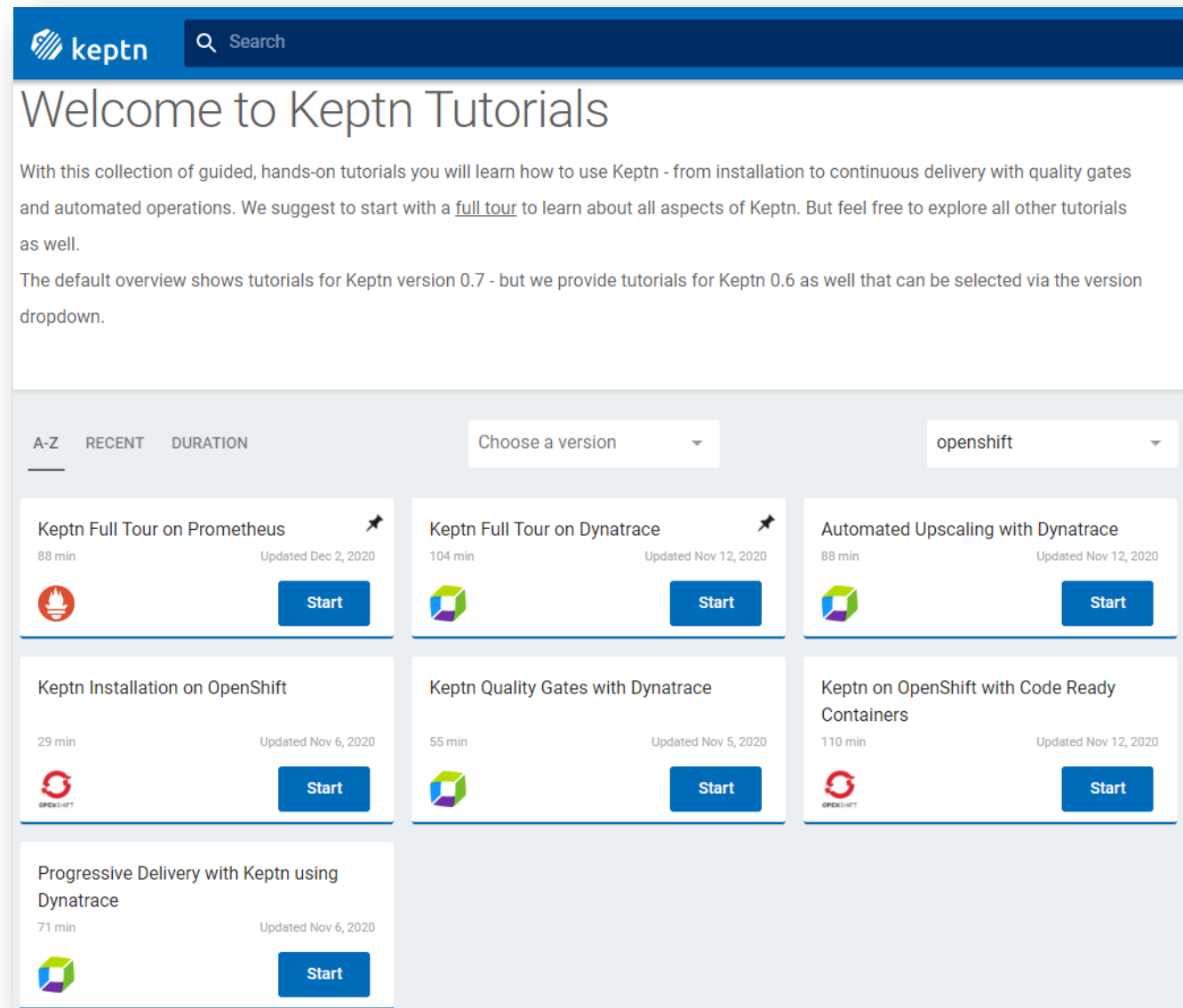
connect your tools







# Get started with our tutorials: tutorials.keptn.sh



The screenshot shows the Keptn Tutorials website. At the top, there is a blue header with the Keptn logo and a search bar. Below the header, the main heading reads "Welcome to Keptn Tutorials". A paragraph of text explains that the collection includes guided, hands-on tutorials for using Keptn, from installation to continuous delivery with quality gates and automated operations. It suggests starting with a "full tour" and mentions that tutorials are available for Keptn version 0.7 by default, but also for version 0.6 via a dropdown menu.

Below the text, there is a navigation bar with tabs for "A-Z", "RECENT", and "DURATION". To the right of these tabs are two dropdown menus: "Choose a version" and "openshift".

The main content area displays a grid of tutorial cards. Each card includes the title, duration, update date, a small icon, and a "Start" button. The cards are:

- Keptn Full Tour on Prometheus**: 88 min, Updated Dec 2, 2020. Icon: Prometheus.
- Keptn Full Tour on Dynatrace**: 104 min, Updated Nov 12, 2020. Icon: Dynatrace.
- Automated Upscaling with Dynatrace**: 88 min, Updated Nov 12, 2020. Icon: Dynatrace.
- Keptn Installation on OpenShift**: 29 min, Updated Nov 6, 2020. Icon: OpenShift.
- Keptn Quality Gates with Dynatrace**: 55 min, Updated Nov 5, 2020. Icon: Dynatrace.
- Keptn on OpenShift with Code Ready Containers**: 110 min, Updated Nov 12, 2020. Icon: OpenShift.
- Progressive Delivery with Keptn using Dynatrace**: 71 min, Updated Nov 6, 2020. Icon: Dynatrace.



## Commonly Asked Questions

---

- Where does Keptn run?
  - On any Kubernetes (k8s) / OpenShift flavor
- What if I don't have a k8s cluster?
  - Go with „Keptn in a Box“
  - Or go with Keptn on k3s
- How can we extend Keptn?
  - Use our service template: <https://github.com/keptn-sandbox/keptn-service-template-go>
- Can Keptn also execute tests in remote systems?
  - Yes! Keptn „Execution Plane“ can be installed ANYWHERE!

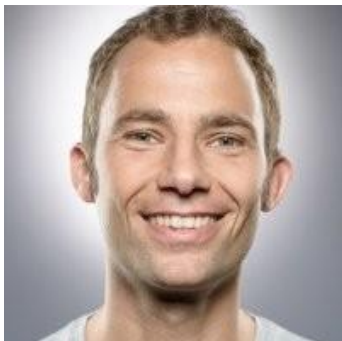
# Performance as a Self-Service

*Based on SLIs/SLOs with Keptn*



# LTB 2021

## Load Testing & Benchmarking



**Andreas Grabner**

DevOps Activist at Dynatrace  
DevRel for Keptn

@grabnerandi

<https://www.linkedin.com/in/grabnerandi>



Visit us @ <https://keptn.sh>

Follow us @keptnProject

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

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