# Automated SLO-Based Testing

*"Testing as a Self-Service with Keptn"*

TestCon

15th October 2020 @ 11:10 (GMT+3)

TestCon Europe 2020
ONLINE EDITION
The biggest software testing conference in Europe
October 13-15 Online

**Andreas Grabner**

DevOps Activist at Dynatrace

DevRel for Keptn

@grabnerandi, [https://www.linkedin.com/in/grabnerandi](https://www.linkedin.com/in/grabnerandi)

keptn

**Follow us** @keptnProject

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

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

CLOUD NATIVE
COMPUTING FOUNDATION

We are a Cloud Native Computing Foundation Sandbox project.

# What does Keptn do? End-2-End Capabilities overview!

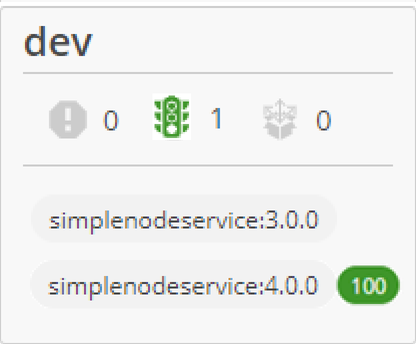
Increased **Speed** and **Quality** of Progressive Delivery through **Automation**

**Automate Operations**

Developer

Pull Request

simplenodeservice:4.0.0

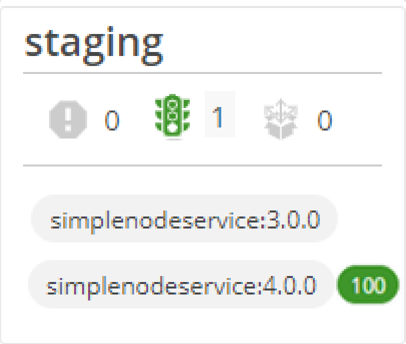


dev

0 1 0

simplenodeservice:3.0.0

simplenodeservice:4.0.0 100

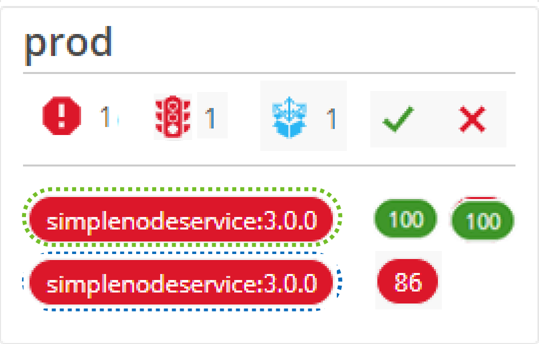


staging

0 1 0

simplenodeservice:3.0.0

simplenodeservice:4.0.0 100



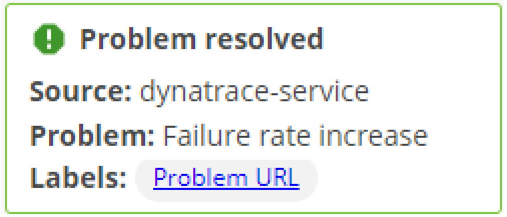
prod

1 1 1 ✓ ✗

simplenodeservice:3.0.0 100 100

simplenodeservice:3.0.0 86

Closed-Loop Remediation



**Problem resolved**

**Source:** dynatrace-service

**Problem:** Failure rate increase

**Labels:** Problem URL

(1) Deploy & **Test**

(2) Evaluate SLOs 100

(3) Auto-Promote ✓

(4) Deploy & **Test**

(5) Evaluate SLOs 100

(6) Promote? ✓ ✗

(7) Deploy Blue / Green

(8) Evaluate SLOs 86

(9) Toggle Blue / Green

(10) Re-Evaluate SLOs 100

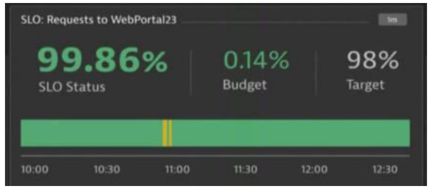
(1) Action 1: Scale Up

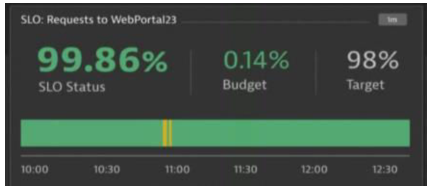
(2) Evaluate SLOs 86

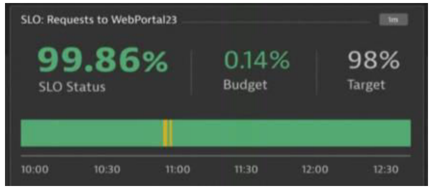
(3) Action 2: Roll Back

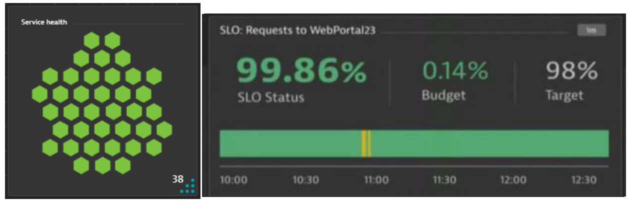
(4) Evaluate SLOs 100

Observabilty

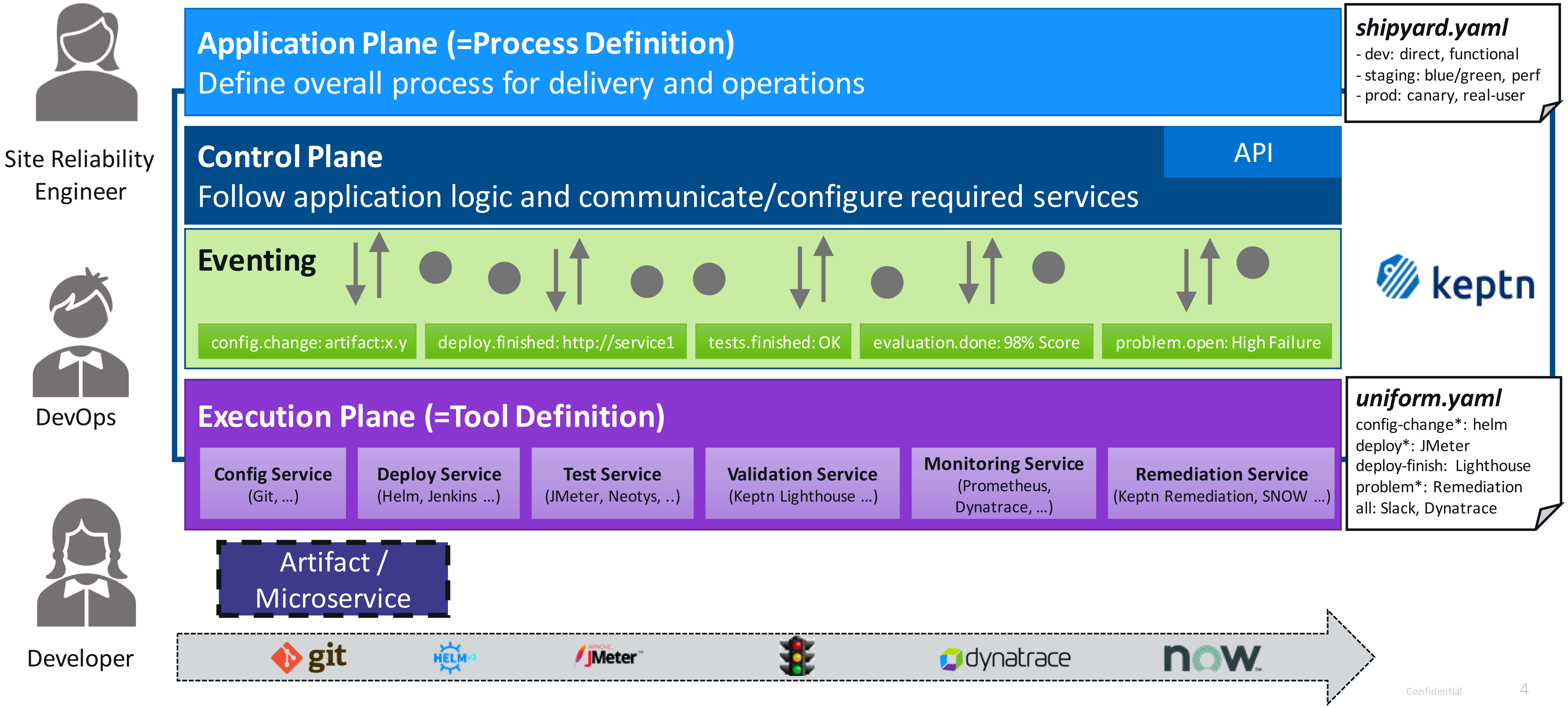








# Keptn's architecture and event-driven approach for delivery & operation



Site Reliability Engineer

DevOps

Developer

**Application Plane (=Process Definition)**
Define overall process for delivery and operations

***shipyard.yaml***
- dev: direct, functional
- staging: blue/green, perf
- prod: canary, real-user

**Control Plane**
Follow application logic and communicate/configure required services

API

**Eventing**

config.change: artifact:x.y

deploy.finished: http://service1

tests.finished: OK

evaluation.done: 98% Score

problem.open: High Failure

keptn

**Execution Plane (=Tool Definition)**

**Config Service**
(Git, ...)

**Deploy Service**
(Helm, Jenkins ...)

**Test Service**
(JMeter, Neotys, ..)

**Validation Service**
(Keptn Lighthouse ...)

**Monitoring Service**
(Prometheus, Dynatrace, ...)

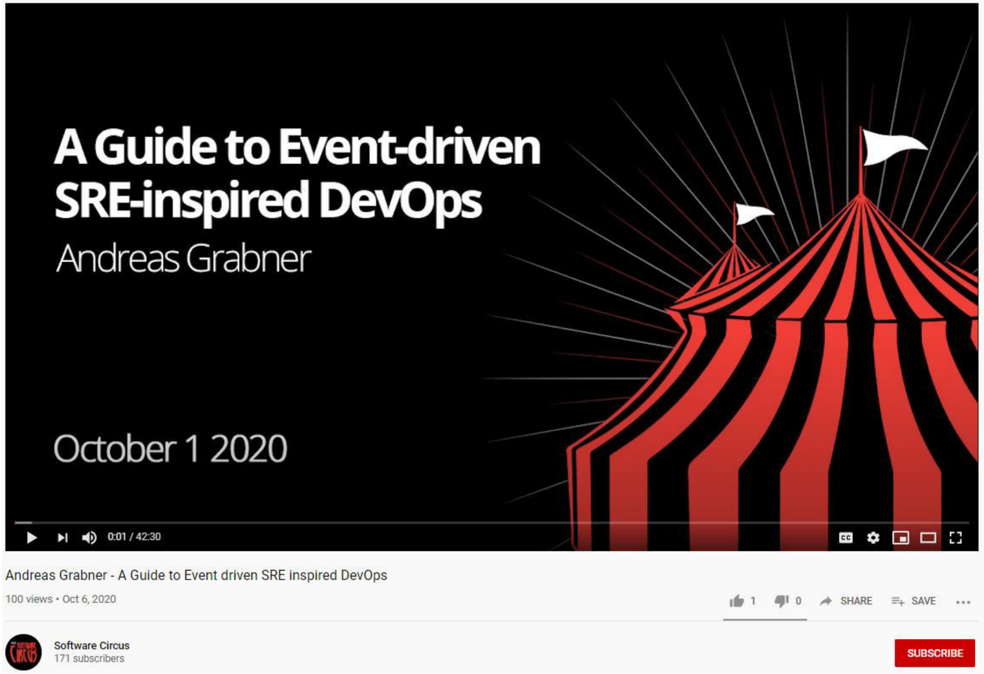
**Remediation Service**
(Keptn Remediation, SNOW ...)

***uniform.yaml***
config-change*: helm
deploy*: JMeter
deploy-finish: Lighthouse
problem*: Remediation
all: Slack, Dynatrace

Artifact / Microservice

# If you want to learn more about Keptn

- GitHub: https://github.com/keptn/keptn
- Website: www.keptn.sh , Tutorials: https://tutorials.keptn.sh
- 2 Minute install of Keptn on k3s: https://github.com/keptn-sandbox/keptn-on-k3s
- Meetup Recording: https://www.youtube.com/watch?v=qWq1rPKMFQI
- Slack: https://slack.keptn.sh



A Guide to Event-driven SRE-inspired DevOps

Andreas Grabner

October 1 2020

0:01 / 42:30

Andreas Grabner - A Guide to Event driven SRE inspired DevOps

100 views • Oct 6, 2020

Software Circus

171 subscribers

- **But now lets focus on the following Use Cases**
  - Automating Test Analysis using SLIs/SLOs
  - Providing Testing as a Self-Service

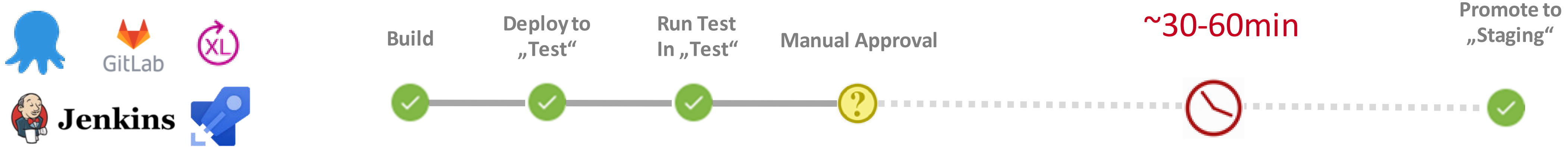
# Use Case #1

Automating Test Analysis using SLIs/SLOs

**80%** of lead time

spent in manual test result / deployment validation

# Root Cause: Lengthy manual approval

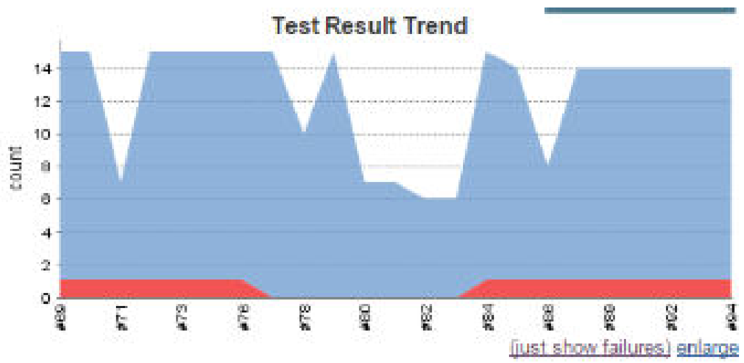


*Is this regression impacting key business use cases* ?

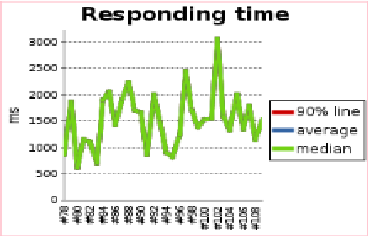
*Which metrics are important and which build is therefore better* ?

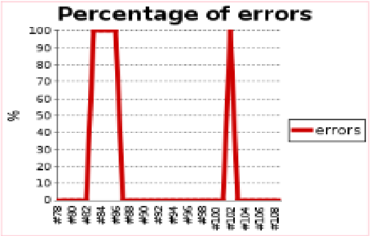
*Which data comes from my test and is relevant for business transactions* ?

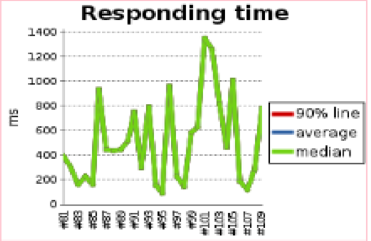
**Functional**: Test Result Trend Not Enough

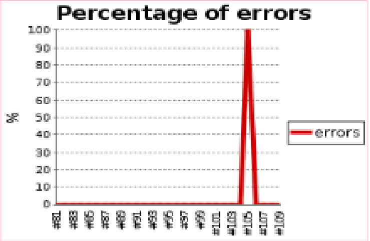


**Performance**: Manual Comparison Is Slow

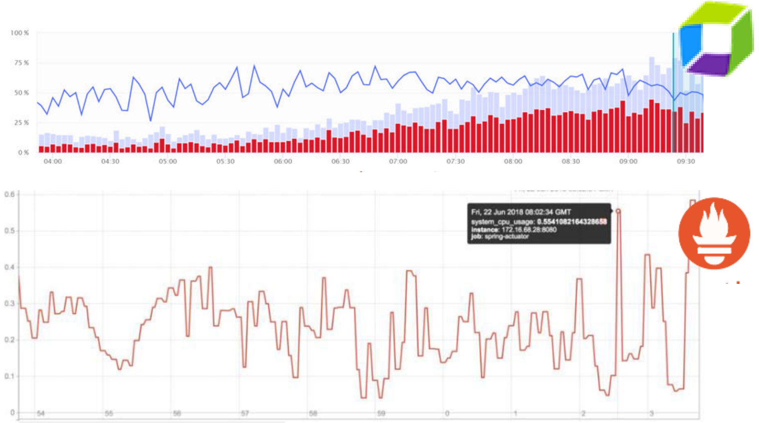








**Monitoring**: Too much unstructed data

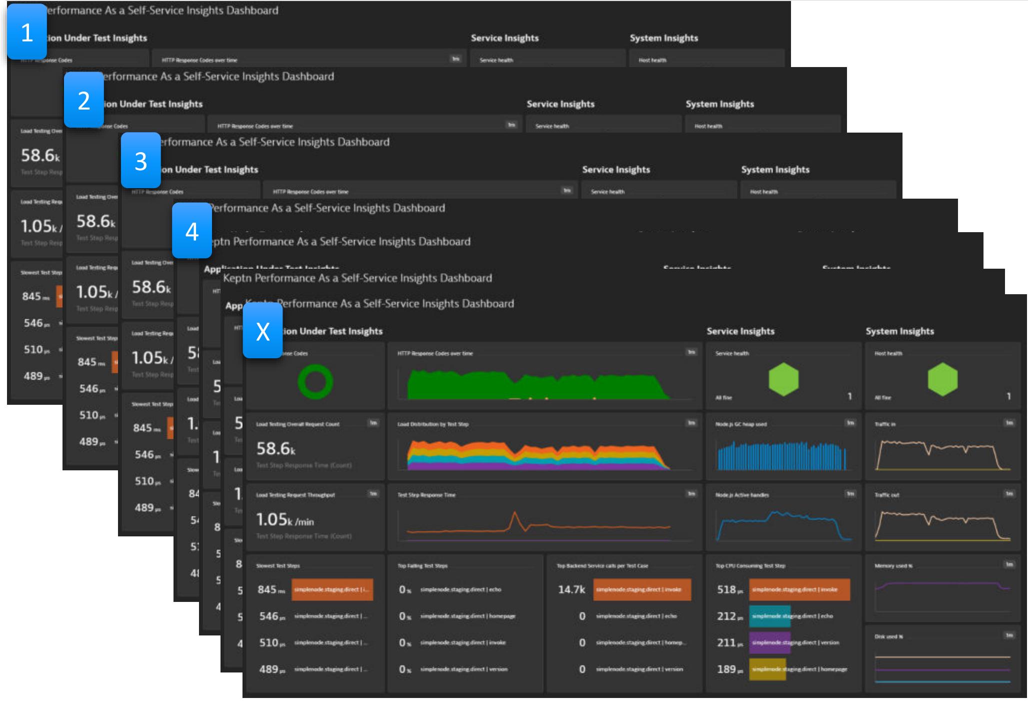


Speed up delivery lead time by **80%**

through Automated SLI/SLO-based Quality Gates

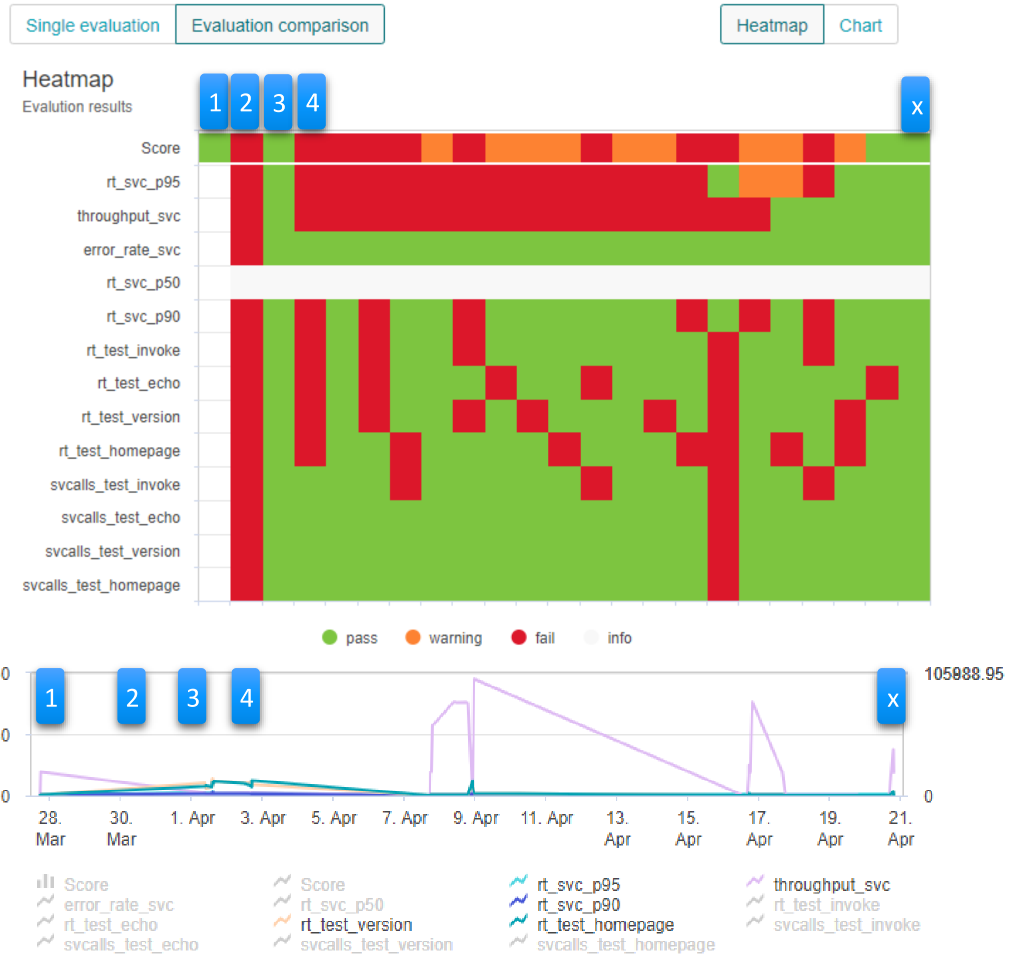
# The problem Keptn solves: Automates Data Analysis

Instead of manually analyzing and comparing test results



1
2
3
4
X
Keptn Performance As a Self-Service Insights Dashboard
Application Under Test Insights
Service Insights
System Insights
58.6k
1.05k /min
845
546
510
489
14.7k
518
212
211
189

Keptn automates that process based on **SLIs** & **SLOs**



# SLI/SLOs – Key Concept from Google's SRE Practices

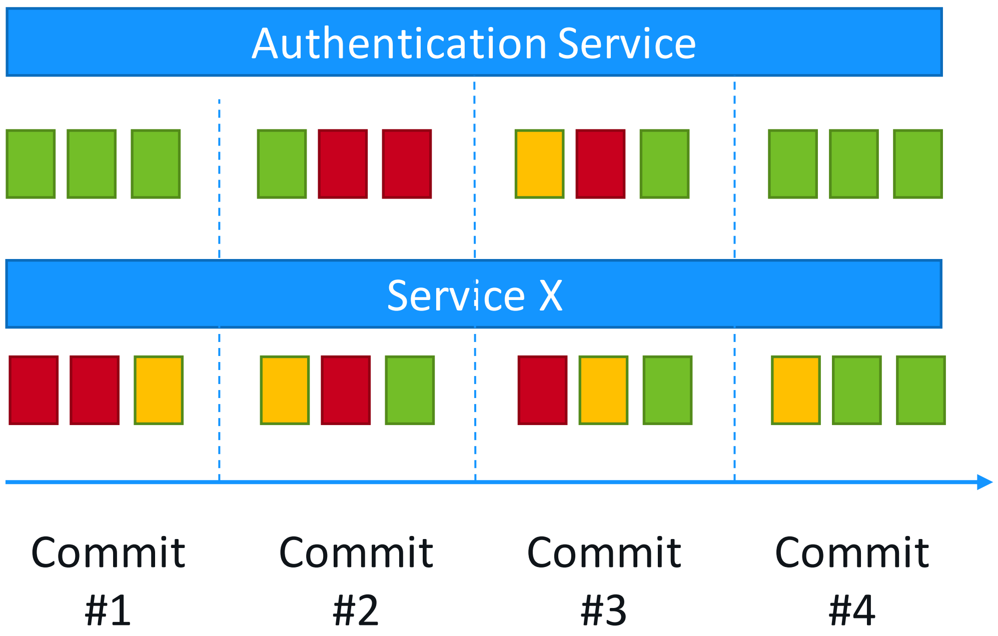
## SLIs drive SLOs which inform SLAs

- Service Level Indicators (SLIs)
  - Definition: Measurable Metrics as the base for evaluation
  - Example: Error Rate of Login Requests
- Service Level Objectives (SLOs)
  - Definition: Binding targets for Service Level Indicators
  - Example: Login Error Rate must be less than 2% over a 30 day period
- Service Level Agreements (SLAs)
  - Definition: Business Agreement between consumer and provider typically based on SLO
  - Example: Logins must be reliable & fast (Error Rate, Response Time, Throughput) 99% within a 30 day window
- Google Cloud YouTube Video
  - SLIs, SLOs, SLAs, oh my! (class SRE implements DevOps): https://www.youtube.com/watch?v=tEylFyxbDLE

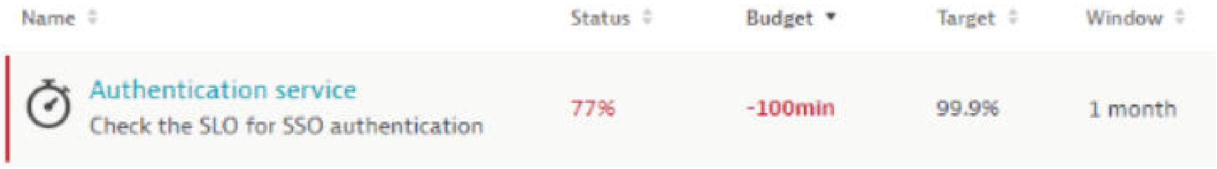
# Keptn applies SRE Best Practices across the lifecycle

**Continuous Delivery** ← Shift-Left **Production**

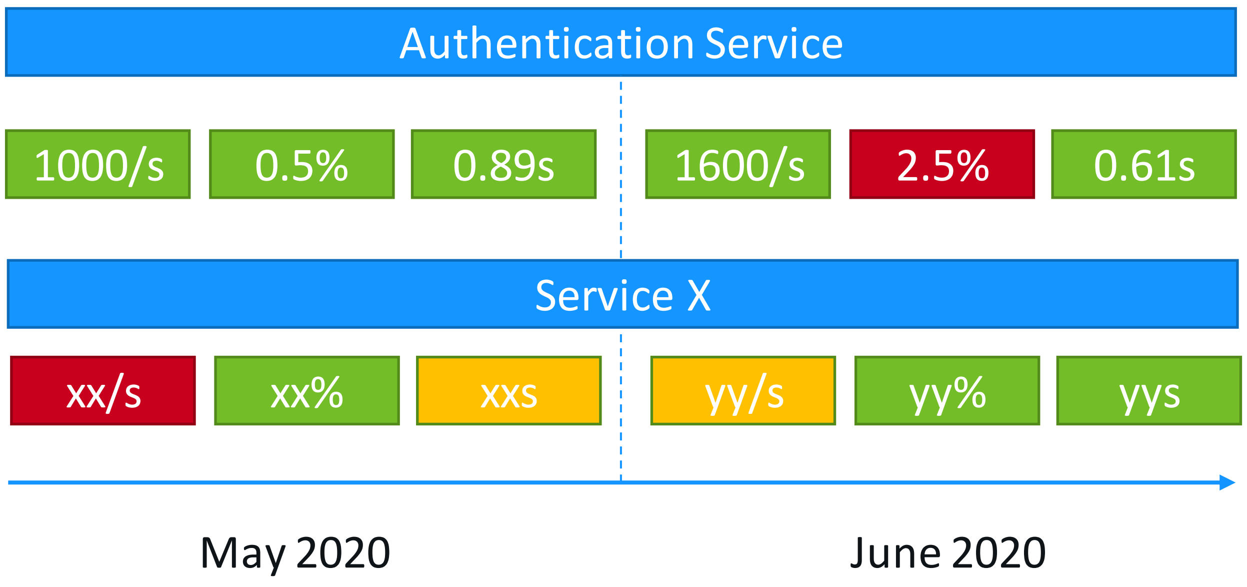
keptn Quality Gates



| | Commit #1 | Commit #2 | Commit #3 | Commit #4 |
|---|---|---|---|---|
| Authentication Service | 🟩 🟩 🟩 | 🟩 🟥 🟥 | 🟨 🟥 🟩 | 🟩 🟩 🟩 |
| Service X | 🟥 🟥 🟨 | 🟨 🟥 🟩 | 🟥 🟨 🟩 | 🟨 🟩 🟩 |



| Name | Status | Budget | Target | Window |
|---|---|---|---|---|
| Authentication service<br>Check the SLO for SSO authentication | 77% | -100min | 99.9% | 1 month |



| | May 2020 | | | June 2020 | | |
|---|---|---|---|---|---|---|
| Authentication Service | 🟩 1000/s | 🟩 0.5% | 🟩 0.89s | 🟩 1600/s | 🟥 2.5% | 🟩 0.61s |
| Service X | 🟥 xx/s | 🟩 xx% | 🟨 xxs | 🟨 yy/s | 🟩 yy% | 🟩 yys |

## Explainer on SLI/SLO Validation as part of Continuous Delivery with Dynatrace & Keptn!

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

| | SLIs (Service Level Indicators) | SLO pass | SLO warn | Build 1 | Build 2 | Build 3 | Build 4 |
|---|---|---|---|---|---|---|---|
| DevOps | **Response Time 95th Perc**<br>**Query**: builtin:service.responsetime(p95) | <=100ms | <= 250ms | 80ms | 120ms | 90ms | 95ms |
| | **Overall Failure Rate**<br>**Query**: builtin:service.errors.total | <= 2% | <= 5% | 0% | 4% | 1% | 0% |
| | **Test Step LOGIN Response Time**<br>**Query**: calc:service.teststeprt:filter(Test, LOGIN) | <=150ms & <=+10% | <= 400ms | 100ms | 90ms | 120ms | 95ms |
| | **Test Step LOGIN # Service Calls**<br>**Query**: calc:service.testsvc:filter(tx, LOGIN) | <= +0% | | 1 | 1 | 2 | 1 |
| | **SLO: Overall Score Goal** | 90% | 75% | 100% | 75% | 62.5% | 100% |

# SLI/SLO-based evaluation implementation in Keptn

## 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"
```

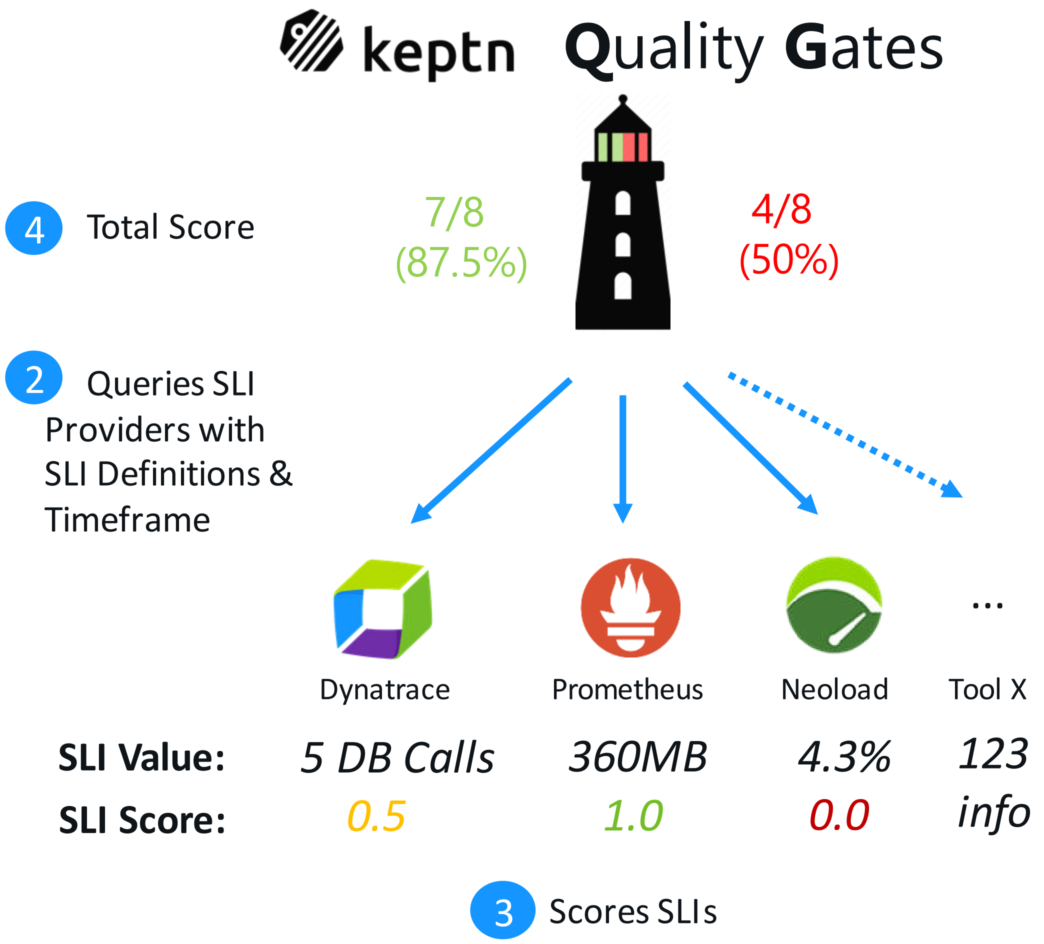
## SLOs defined on Keptn Service Level as YAML

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%"
```

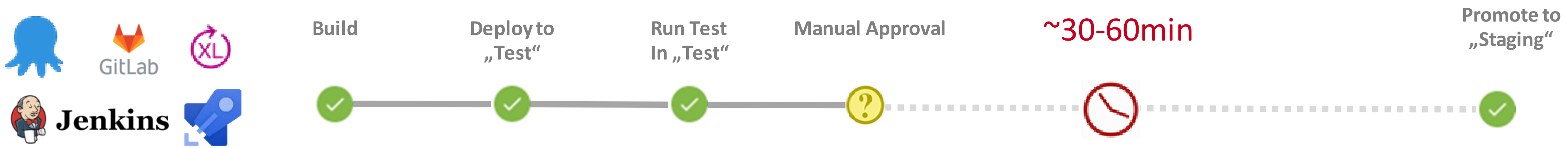
1

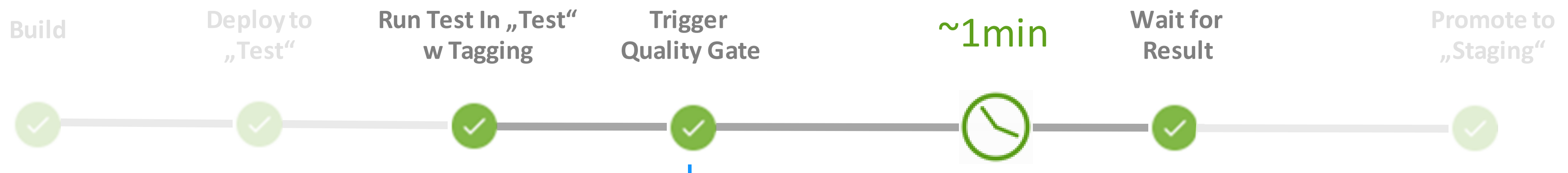
```
$ keptn start-evaluation 30m myservice sli.yaml slo.yaml
```



| | Dynatrace | Prometheus | Neoload | Tool X |
|---|---|---|---|---|
| **SLI Value:** | *5 DB Calls* | *360MB* | *4.3%* | *123* |
| **SLI Score:** | *0.5* | *1.0* | *0.0* | *info* |

# Integrate Automated SLI/SLO-based Analysis into your existing workflows





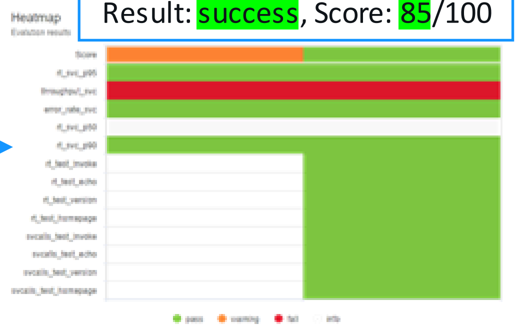
Tagging

**SLI & SLO**

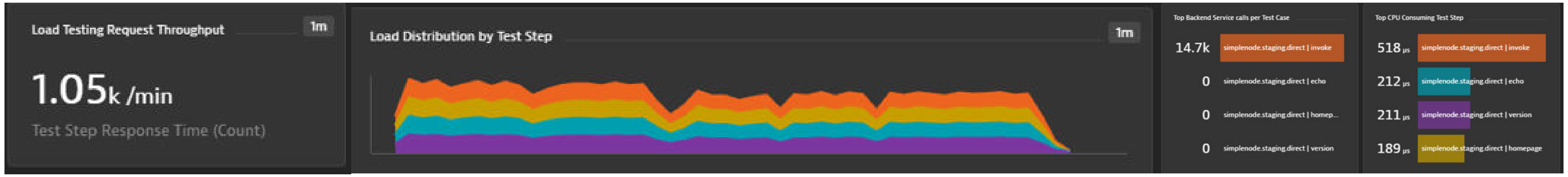
```
Rt(p95)  <  500ms
#ofSQLs  <=   5
cpu(max)<   80%
Java  GC  <    2%
...
```

Validate SLOs

Result: success, Score: 85/100



Pull SLIs for Testing time frame



Load Testing Request Throughput 1m

1.05k /min

Test Step Response Time (Count)

Load Distribution by Test Step 1m

Top Backend Service calls per Test Case

| | |
|---|---|
| 14.7k | simplenode.staging.direct \| invoke |
| 0 | simplenode.staging.direct \| echo |
| 0 | simplenode.staging.direct \| homep... |
| 0 | simplenode.staging.direct \| version |

Top CPU Consuming Test Step

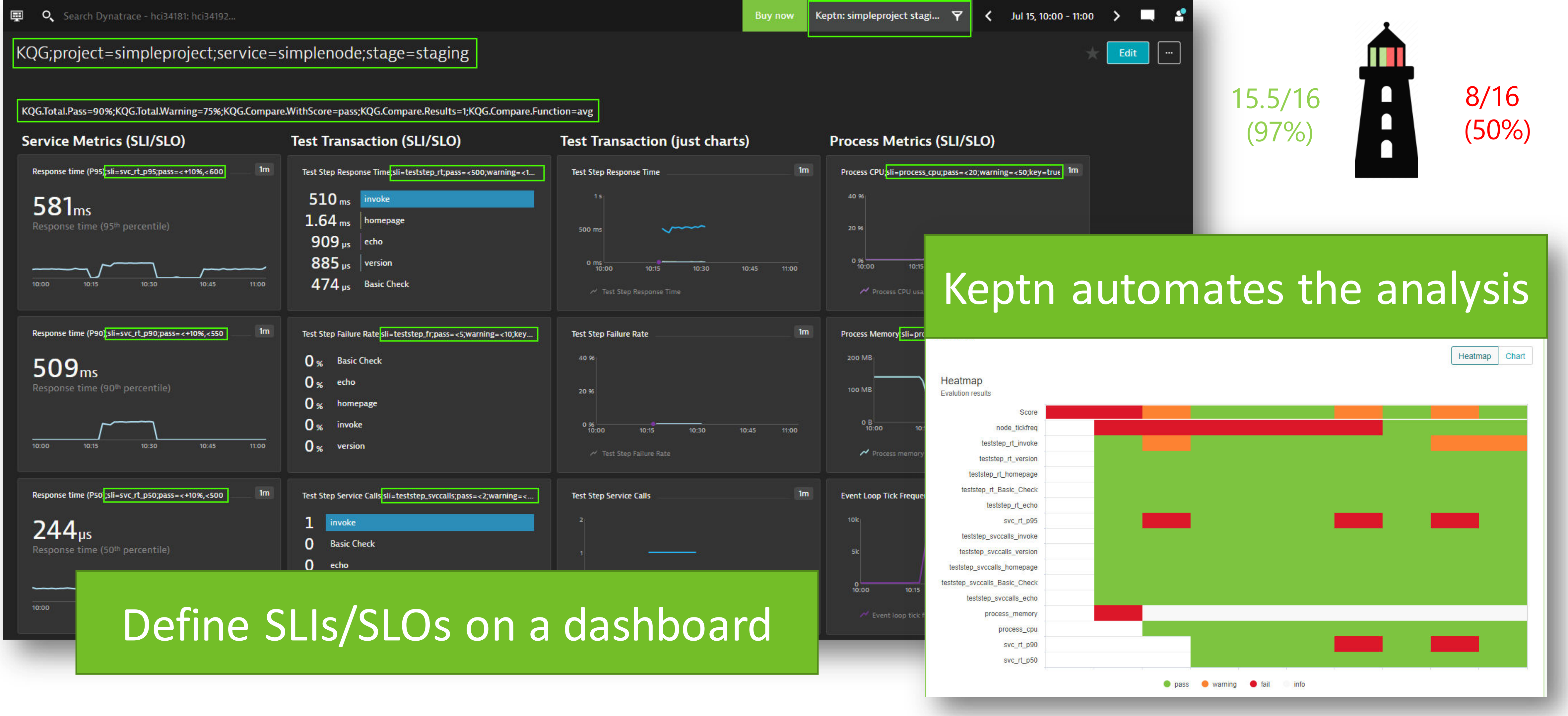
| | |
|---|---|
| 518 µs | simplenode.staging.direct \| invoke |
| 212 µs | simplenode.staging.direct \| echo |
| 211 µs | simplenode.staging.direct \| version |
| 189 µs | simplenode.staging.direct \| homepage |

# Demo: Automated SLI/SLO Validation based on Dynatrace Dashboards

keptn

15.5/16
(97%)

8/16
(50%)



Search Dynatrace - hci34181: hci34192...

Keptn: simpleproject stagi...

Jul 15, 10:00 - 11:00

KQG;project=simpleproject;service=simplenode;stage=staging

KQG.Total.Pass=90%;KQG.Total.Warning=75%;KQG.Compare.WithScore=pass;KQG.Compare.Results=1;KQG.Compare.Function=avg

## Service Metrics (SLI/SLO)

Response time (P95);sli=svc_rt_p95;pass=<+10%,<600

581ms
Response time (95th percentile)

Response time (P90);sli=svc_rt_p90;pass=<+10%,<550

509ms
Response time (90th percentile)

Response time (P50);sli=svc_rt_p50;pass=<+10%,<500

244µs
Response time (50th percentile)

## Test Transaction (SLI/SLO)

Test Step Response Time;sli=teststep_rt;pass=<500;warning=<1...

| | |
|---|---|
| 510 ms | invoke |
| 1.64 ms | homepage |
| 909 µs | echo |
| 885 µs | version |
| 474 µs | Basic Check |

Test Step Failure Rate;sli=teststep_fr;pass=<5;warning=<10;key...

| | |
|---|---|
| 0 % | Basic Check |
| 0 % | echo |
| 0 % | homepage |
| 0 % | invoke |
| 0 % | version |

Test Step Service Calls;sli=teststep_svccalls;pass=<2;warning=<...

| | |
|---|---|
| 1 | invoke |
| 0 | Basic Check |
| 0 | echo |

## Test Transaction (just charts)

Test Step Response Time

Test Step Failure Rate

Test Step Service Calls

## Process Metrics (SLI/SLO)

Process CPU;sli=process_cpu;pass=<20;warning=<50;key=true

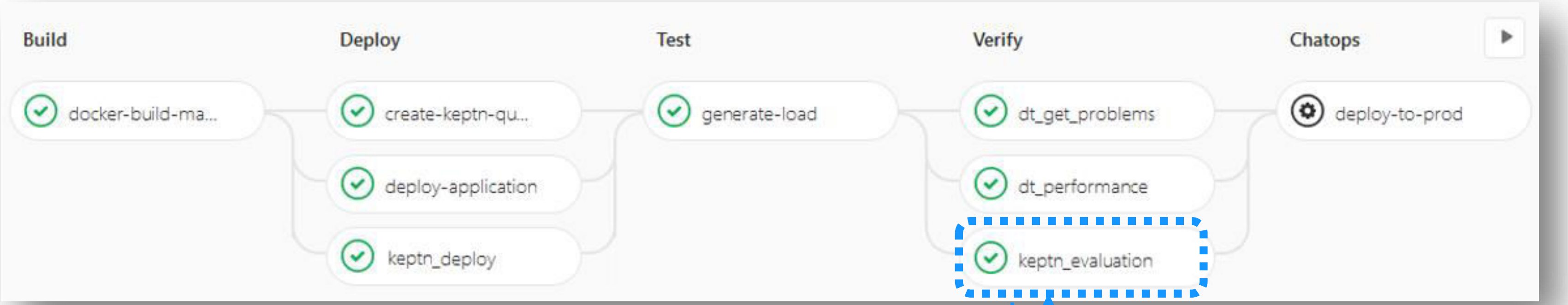
Process Memory;sli=pro

Event Loop Tick Freque

Keptn automates the analysis

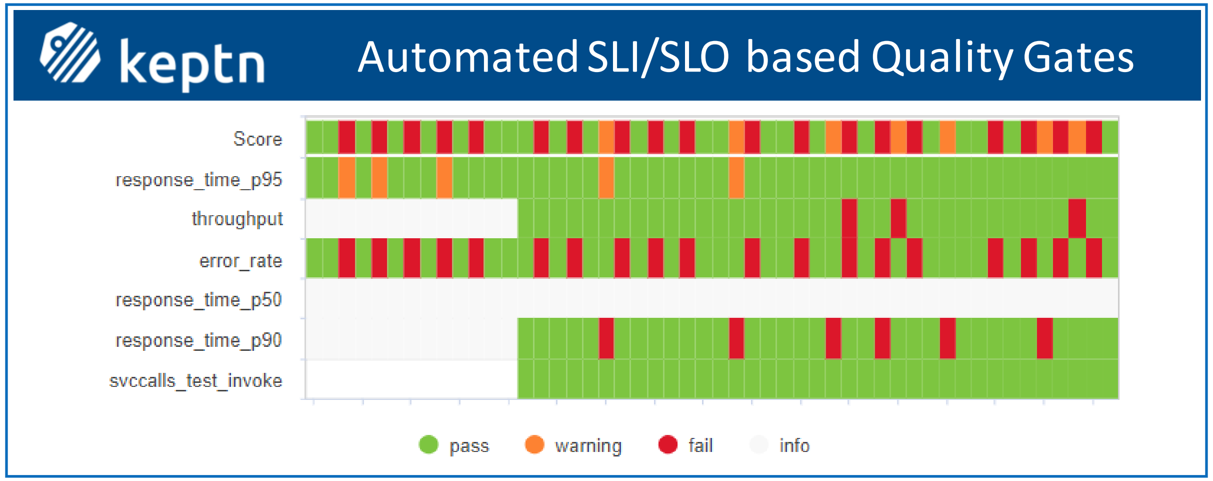
Define SLIs/SLOs on a dashboard

# User Example: Automating Build Approvals using Keptn's SLIs/SLOs in GitLab



Trigger Evaluation

87.5%: passed



**Christian Heckelmann**

Senior Systems Engineer

ERT®

Confidence at every turn™

# Build your own SLI Provider?

- Some ideas for custom SLI Providers
  - Pull in Testing Tool Result, e.g: Selenium, Load Runner, Gatling, …
  - Pull in Code Quality Metrics, e.g: SonarQube, Coverity, xTest …
  - Pull in Monitoring/Observability Data, e.g: Datadog, NewRelic, AppDynamics, …
- I built a custom SLI Provider for a recent conference parsing a JSON Result File
  - https://github.com/grabnerandi/pac-sliprovider
- You can start building our own SLI Provider
  - https://github.com/keptn-sandbox/keptn-service-template-go/

# Use Case #2

Testing as a Self-Service

# Challenge: Automating Performance with Jenkins requires managing tools & environments

Jenkins

- ✓ Build
- ✓ Deploy to „Test“
- ✓ Standup Test Infrastructure
- ✓ Execute Tests
- ✓ Gather Performance Data
- ✓ Evaluate Results

*Where do we run these tests?*

*How much hardware is needed to run these tests?*

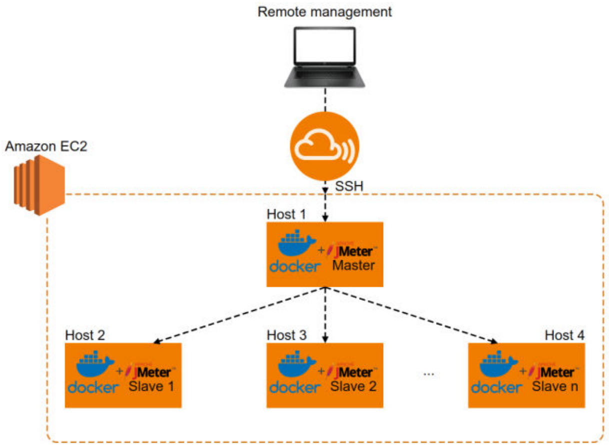
*Who manages this infrastructure?*

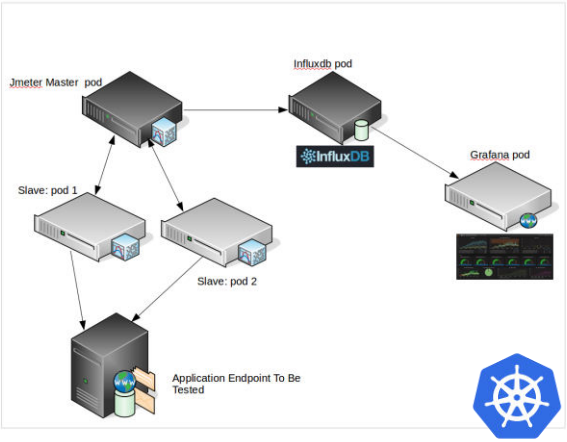
*How to enable different workloads?*

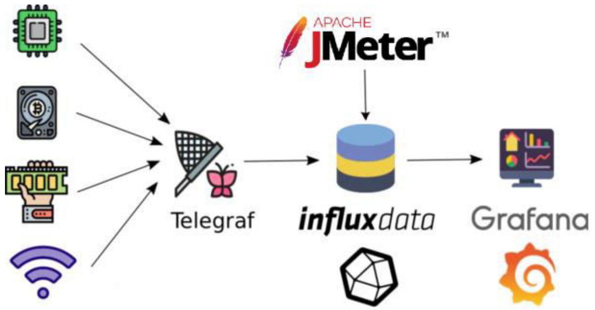
*Where do we stream test metrics to?*

*How to analyze the results?*

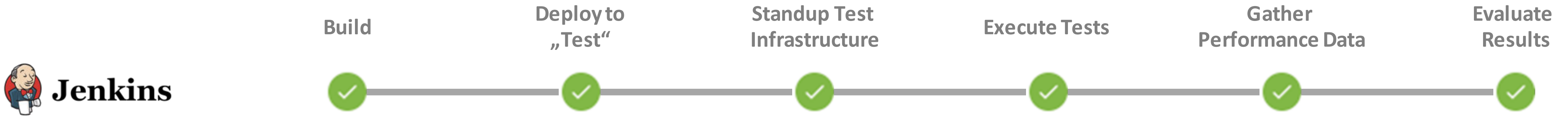
## DIY (Do It Yourself) approach: lots of online guides available!

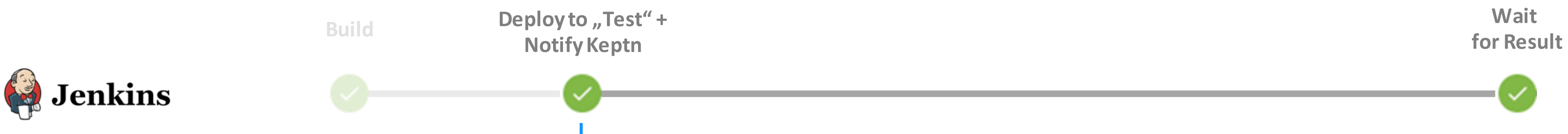


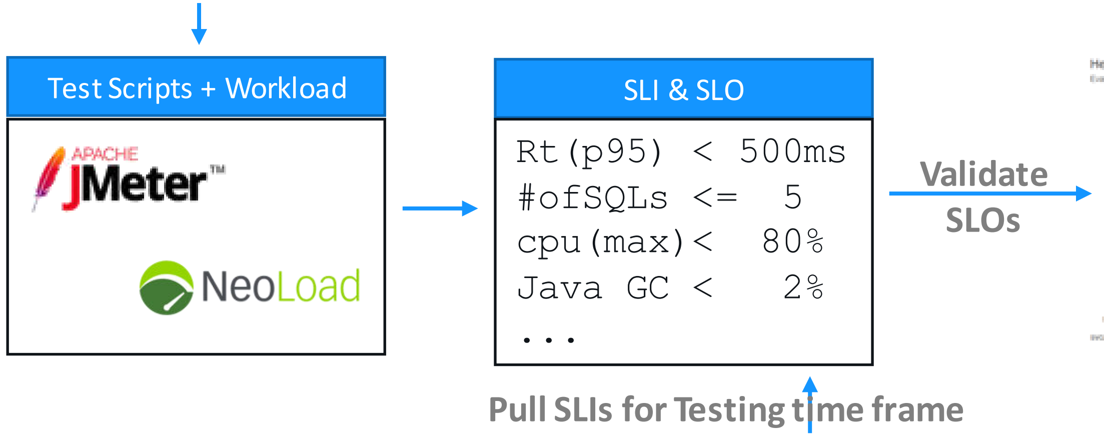




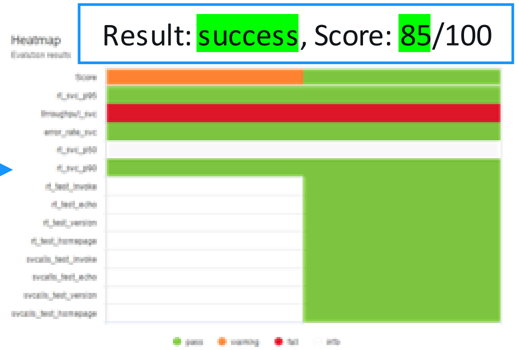
# Solution: Keptn automates Test Orchestration

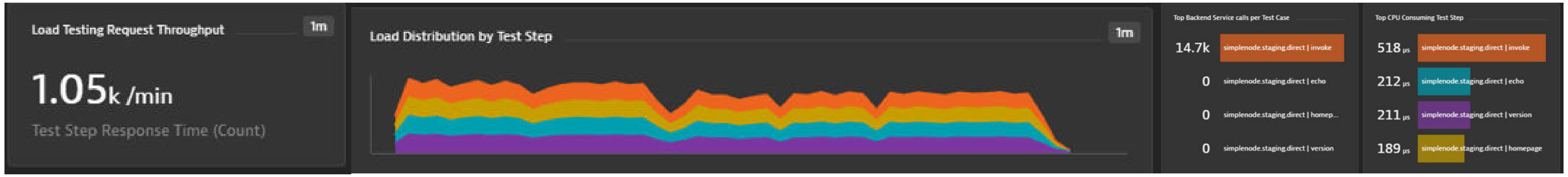






```
Rt(p95)  <  500ms
#ofSQLs  <=   5
cpu(max)<   80%
Java GC <    2%
...
```


Result: success, Score: 85/100


Load Testing Request Throughput 1m

1.05k /min

Test Step Response Time (Count)

Load Distribution by Test Step 1m

Top Backend Service calls per Test Case

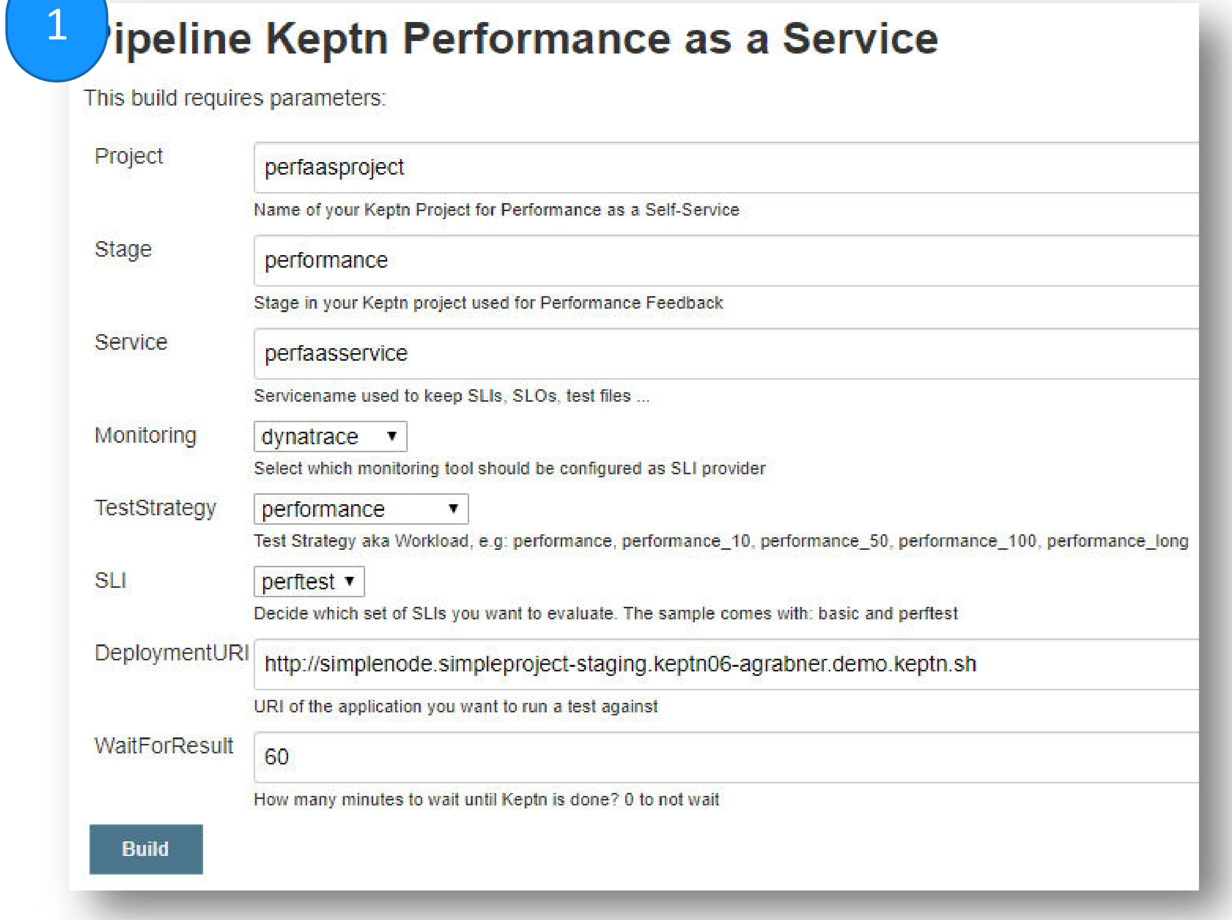
| | |
|---|---|
| 14.7k | simplenode.staging.direct \| invoke |
| 0 | simplenode.staging.direct \| echo |
| 0 | simplenode.staging.direct \| homep... |
| 0 | simplenode.staging.direct \| version |

Top CPU Consuming Test Step

| | |
|---|---|
| 518 µs | simplenode.staging.direct \| invoke |
| 212 µs | simplenode.staging.direct \| echo |
| 211 µs | simplenode.staging.direct \| version |
| 189 µs | simplenode.staging.direct \| homepage |

# Demo: based on https://github.com/keptn-sandbox/jenkins-tutorial

1



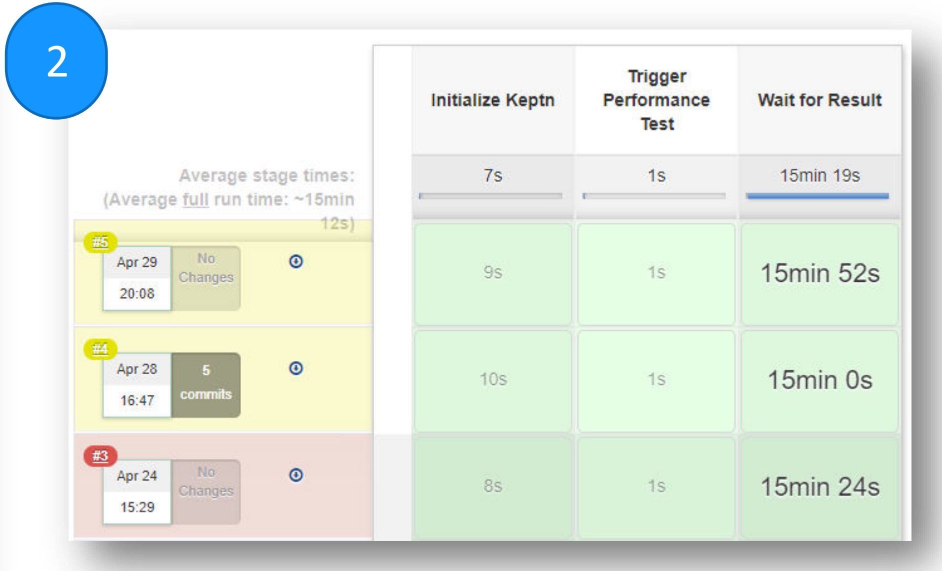
## Pipeline Keptn Performance as a Service

This build requires parameters:

| | |
|---|---|
| Project | perfaasproject<br>Name of your Keptn Project for Performance as a Self-Service |
| Stage | performance<br>Stage in your Keptn project used for Performance Feedback |
| Service | perfaasservice<br>Servicename used to keep SLIs, SLOs, test files ... |
| Monitoring | dynatrace ▼<br>Select which monitoring tool should be configured as SLI provider |
| TestStrategy | performance ▼<br>Test Strategy aka Workload, e.g: performance, performance_10, performance_50, performance_100, performance_long |
| SLI | perftest ▼<br>Decide which set of SLIs you want to evaluate. The sample comes with: basic and perftest |
| DeploymentURI | http://simplenode.simpleproject-staging.keptn06-agrabner.demo.keptn.sh<br>URI of the application you want to run a test against |
| WaitForResult | 60<br>How many minutes to wait until Keptn is done? 0 to not wait |

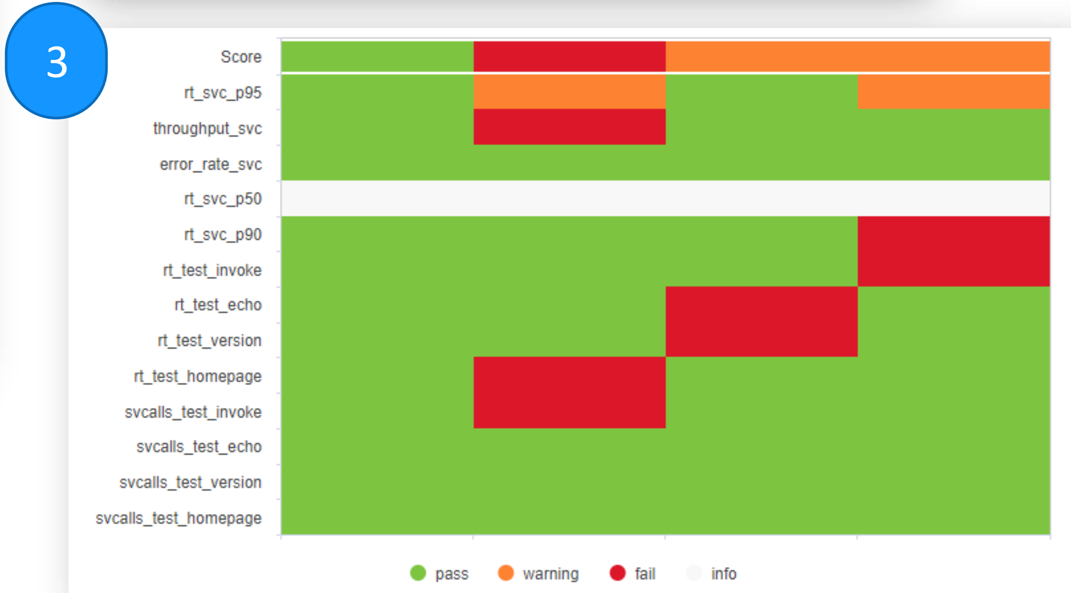
Build

2

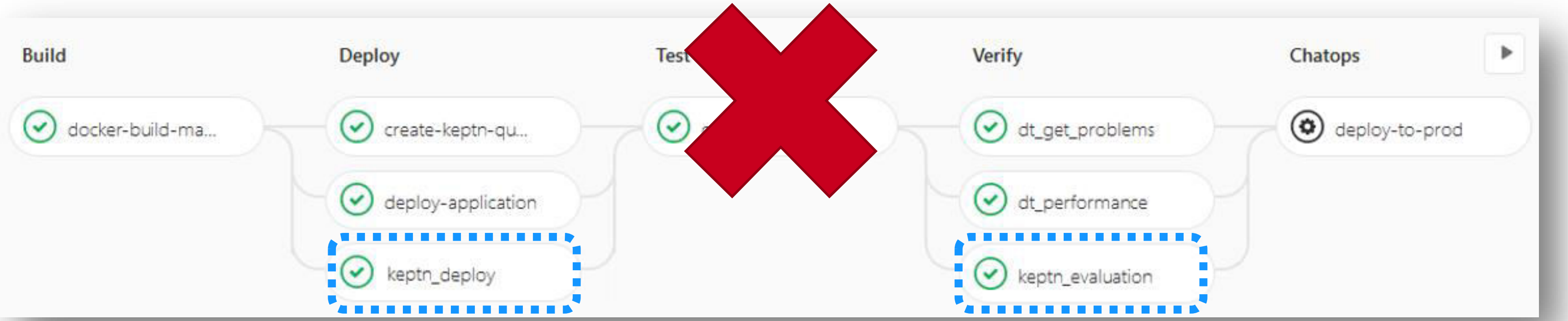


| | Initialize Keptn | Trigger Performance Test | Wait for Result |
|---|---|---|---|
| Average stage times:<br>(Average full run time: ~15min 12s) | 7s | 1s | 15min 19s |
| #5 Apr 29 20:08 No Changes | 9s | 1s | 15min 52s |
| #4 Apr 28 16:47 5 commits | 10s | 1s | 15min 0s |
| #3 Apr 24 15:29 No Changes | 8s | 1s | 15min 24s |

3



# User Example: Reduce pipeline complexity by using Keptn‘s Testing as a Self-Service



Notify about deployment

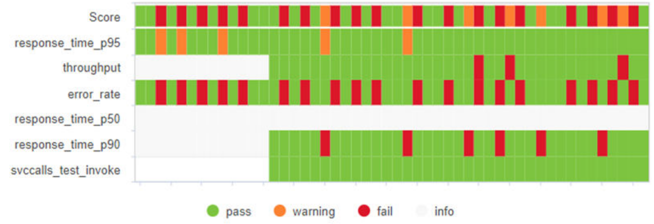
87.5%: passed

keptn Autonomous Cloud Control Plane

Performance as a Service

APACHE JMeter™

keptn Automated SLI/SLO based Quality Gates



**Christian Heckelmann**

Senior Systems Engineer

ERT® Confidence at every turn™

# Build your own Keptn Test Execution Service?

- Some ideas for additional Test Execution Services
  - Execute Tests, e.g: Selenium, Load Runner, Gatling, …
  - Pull in Monitoring/Observability  Data, e.g: Datadog, NewRelic, AppDynamics, …
- You can start building our own Keptn Test Execution Service
  - https://github.com/keptn-sandbox/keptn-service-template-go/

# Integrate Keptn with your existing CI/CD tools

# Keptn can be integrated with any existing tools

- Through the Keptn CLI or API
- Or through the existing integrations
  - Jenkins: https://github.com/keptn-sandbox/keptn-jenkins-library
  - GitLab: https://github.com/keptn-sandbox/gitlab-tutorial
  - Azure DevOps: https://github.com/keptn-sandbox/keptn-azure-devops-extension
- Want to build your own integration?
  - Check out existing integrations: https://github.com/keptn-sandbox
  - Let us know and contribute to keptn: https://slack.keptn.sh

# Let‘s wrap it up!

# What is Keptn?

keptn

www.keptn.sh

**an event-based control plane for continuous delivery and automated operations for cloud-native applications**

Define application delivery and operations processes **declaratively**

Use **predefined CloudEvents** to separate the process from the tools

**Easy way to integrate and switch between** different tools

Blue/Green Deployments

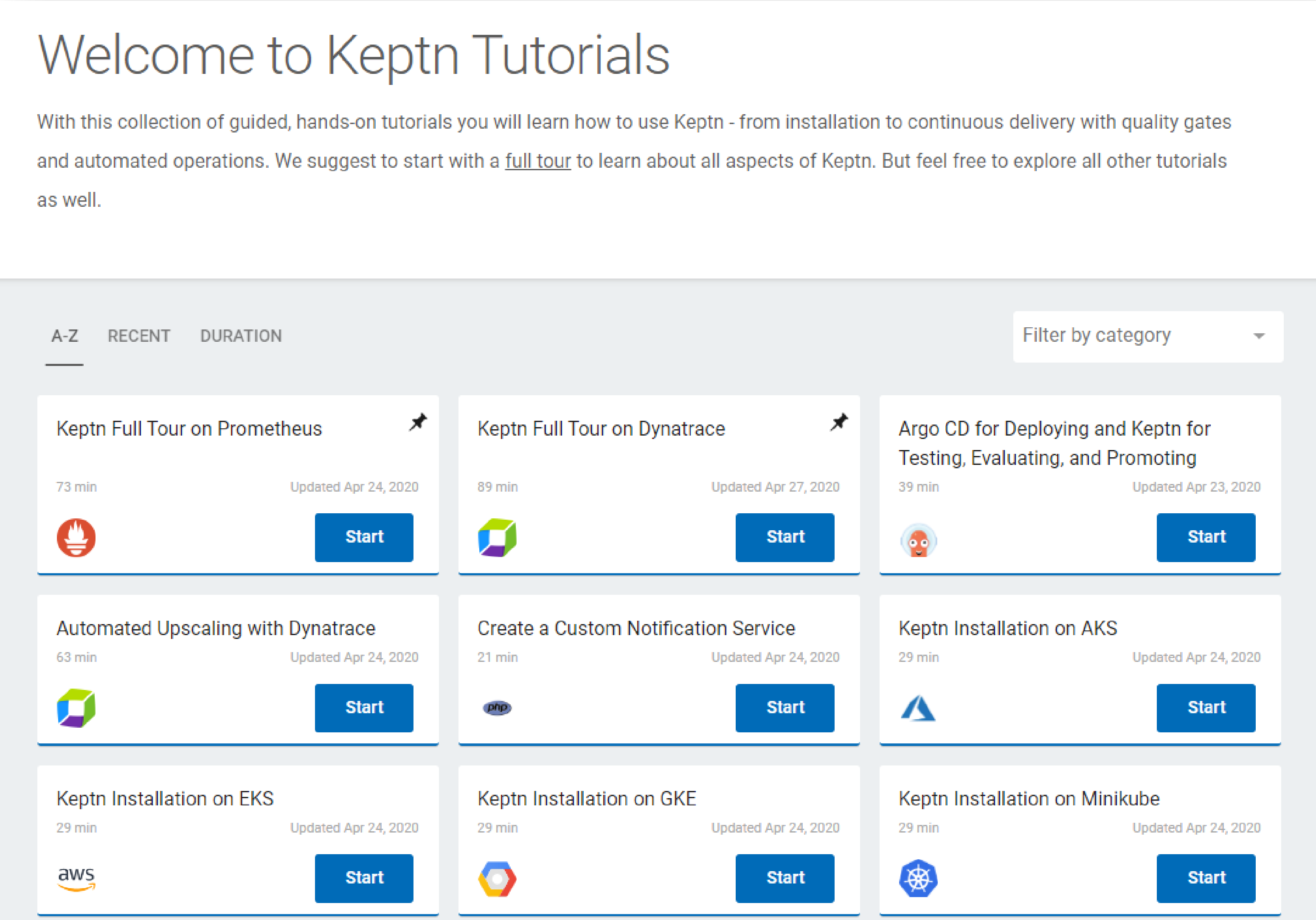
Automated Quality Gates

Automated Operations

Standardized communication protocol

Keptn's uniform

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



## Welcome to Keptn Tutorials

With this collection of guided, hands-on tutorials you will learn how to use Keptn - from installation to continuous delivery with quality gates and automated operations. We suggest to start with a full tour to learn about all aspects of Keptn. But feel free to explore all other tutorials as well.

A-Z RECENT DURATION

Filter by category

Keptn Full Tour on Prometheus
73 min Updated Apr 24, 2020
Start

Keptn Full Tour on Dynatrace
89 min Updated Apr 27, 2020
Start

Argo CD for Deploying and Keptn for Testing, Evaluating, and Promoting
39 min Updated Apr 23, 2020
Start

Automated Upscaling with Dynatrace
63 min Updated Apr 24, 2020
Start

Create a Custom Notification Service
21 min Updated Apr 24, 2020
Start

Keptn Installation on AKS
29 min Updated Apr 24, 2020
Start

Keptn Installation on EKS
29 min Updated Apr 24, 2020
Start

Keptn Installation on GKE
29 min Updated Apr 24, 2020
Start

Keptn Installation on Minikube
29 min Updated Apr 24, 2020
Start

# Questions & Answers

keptn

## Automated SLO-Based Testing

*“Testing as a Self-Service with Keptn”*

TestCon

TestCon Europe 2020

ONLINE EDITION

The biggest software testing conference in Europe

October 13-15 Online

**Andreas Grabner**

DevOps Activist at Dynatrace

DevRel for Keptn

@grabnerandi, [https://www.linkedin.com/in/grabnerandi](https://www.linkedin.com/in/grabnerandi)

keptn

**Follow us** @keptnProject

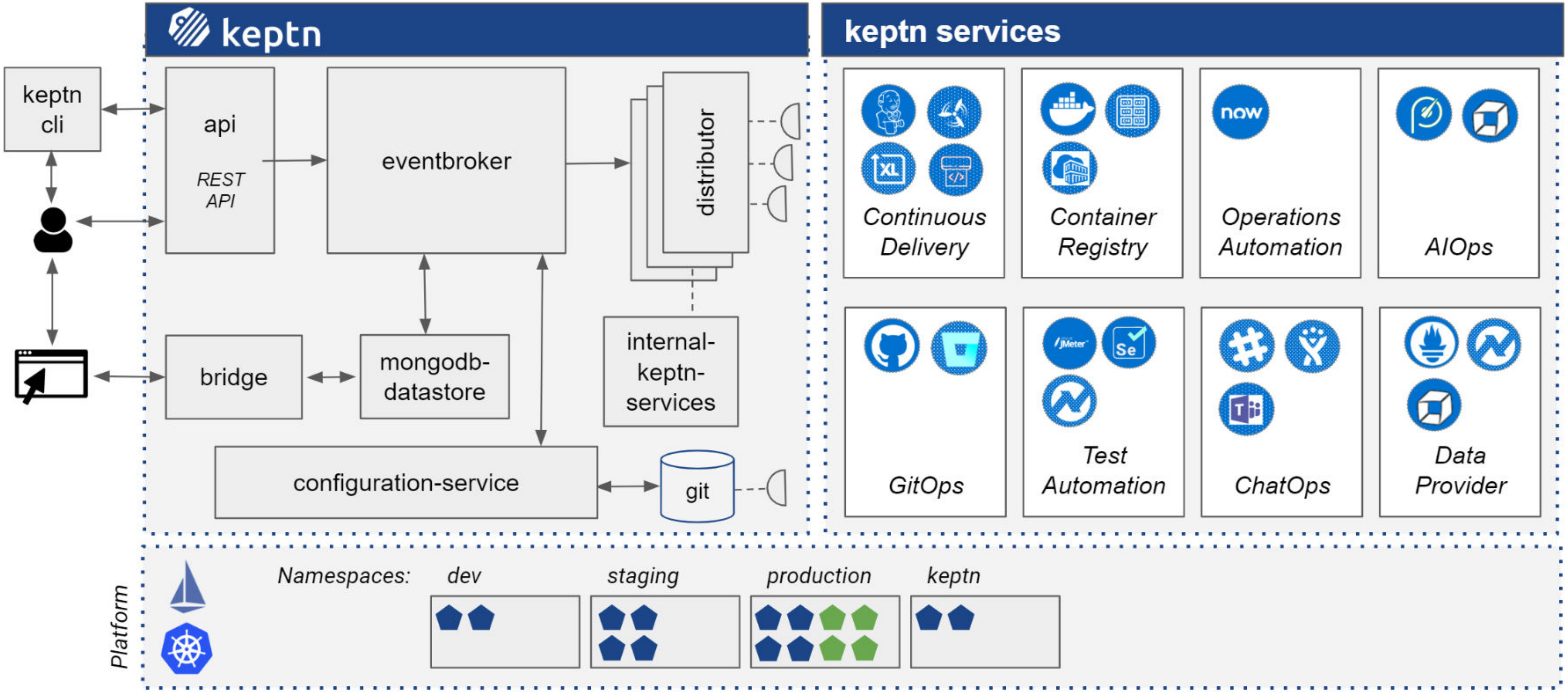
**Star us** @ https://github.com/keptn/keptn

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

CLOUD NATIVE COMPUTING FOUNDATION

We are a Cloud Native Computing Foundation Sandbox project.

# Keptn Architecture


keptn
keptn cli
api
REST API
eventbroker
distributor
bridge
mongodb-datastore
internal-keptn-services
configuration-service
git
keptn services
Continuous Delivery
Container Registry
Operations Automation
AIOps
GitOps
Test Automation
ChatOps
Data Provider
Platform
Namespaces:
dev
staging
production
keptn