Hands-On Lab:
Introduction to our Lab Environment

dynatrace

High-Level Overview of our Lab

Your laptop

Cloud Automation Attendee Host

Login with dtu_training / @dtulabs2021
Used to execute keptn CLI commands

Access through web ui, Keptn CLI or API

dynatrace

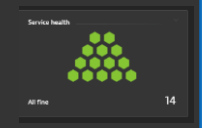
https://fvk03152.cloudautomation.live.dynatrace.com

Cloud Automation Control Plane
+ Dynatrace Services for SLO Validation

Connected

https://hci34192.live.dynatrace.com/

Dynatrace Software Intelligence Platform
- Monitors our k3s clusters
- Synthetic tests against our services



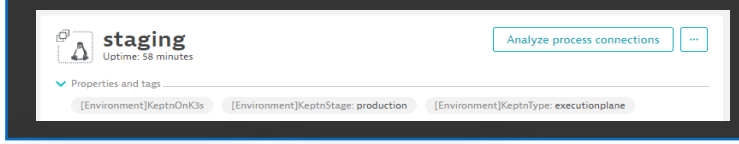
Registered to execute deploy, test, configure, ...

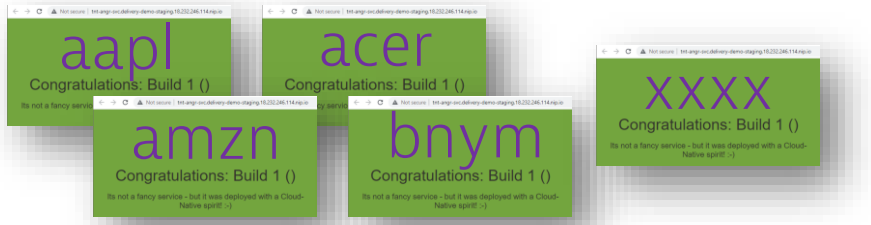
FullStack Monitored with OneAgent

kubernetes

https://*.delivery-demo-**staging**.claus-ws-**staging**.keptn.sh

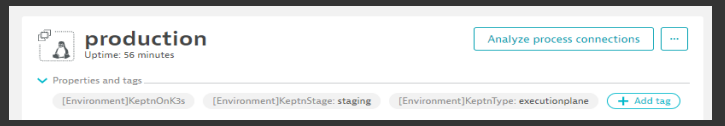
Execution Plane for STAGING
+ Helm, Monaco, Generic Executor

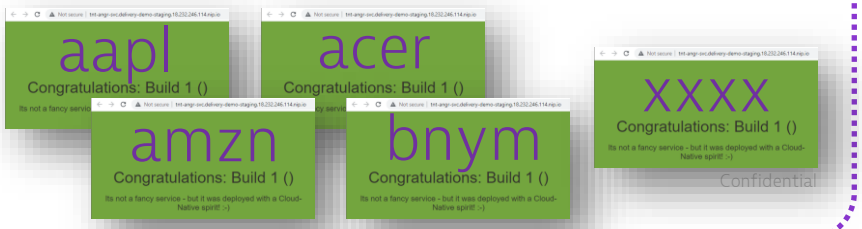




https://*.delivery-demo-**production**.claus-ws-**production**.keptn.sh

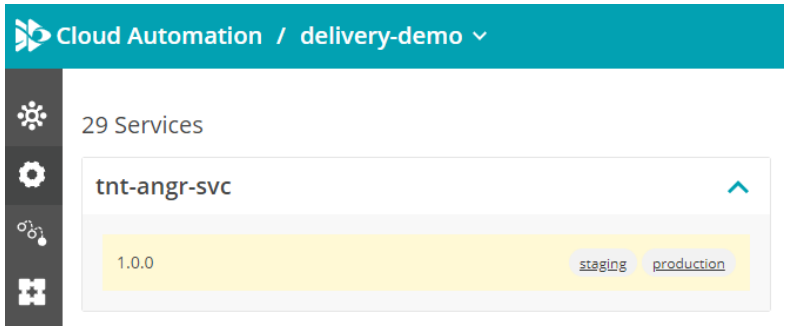
Execution Plane for PRODUCTION
+ Helm, Monaco, Generic Executor





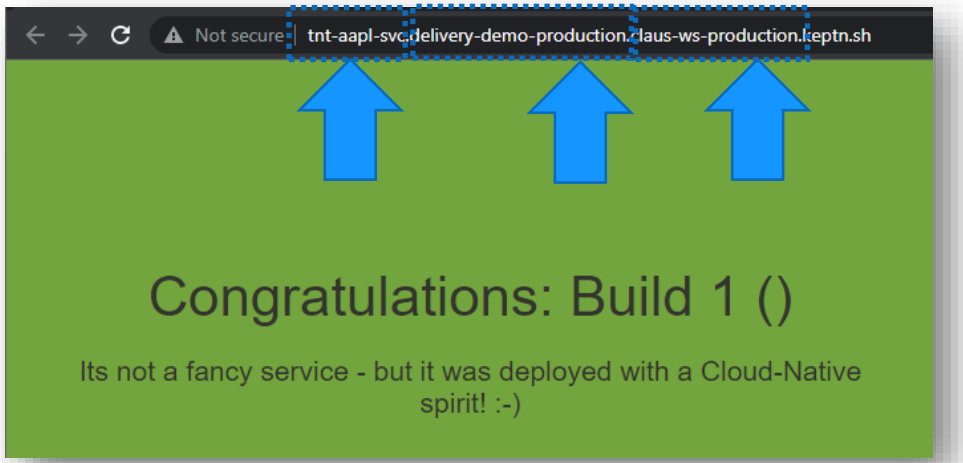
Our sample app: every attendee has its own application instance in staging & production

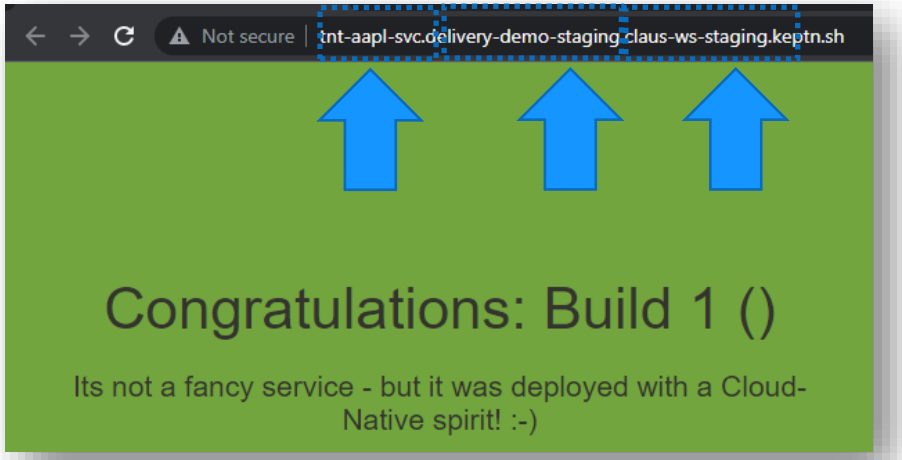
- Deployed through Cloud Automation on our two k8s clusters



Try to access YOUR app!

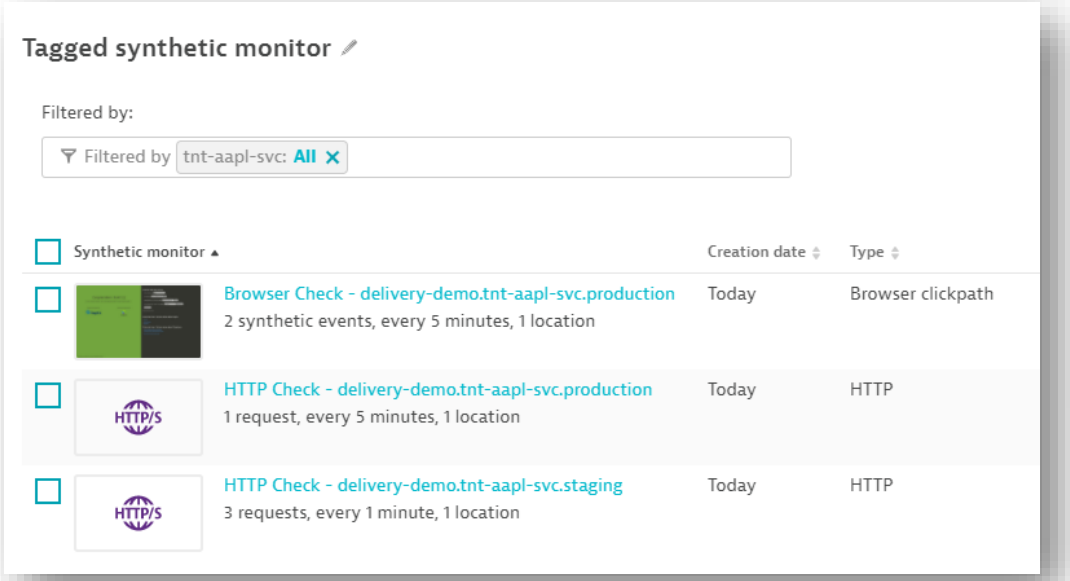
- With unique URLs including your own „TenantID“ and environment
  - http://tnt-aapl-svc.delivery-demo-staging.claus-ws-staging.keptn.sh/
  - http://tnt-aapl-svc.delivery-demo-production.claus-ws-production.keptn.sh/

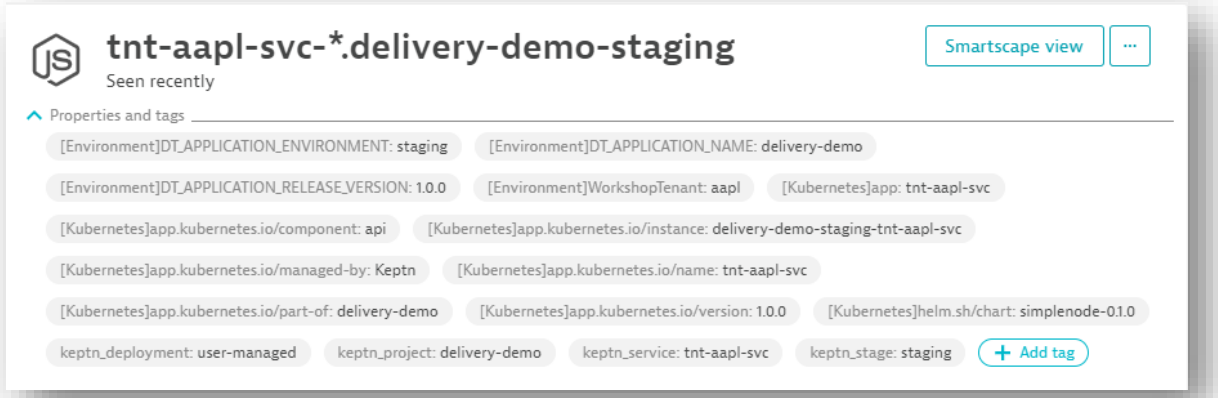




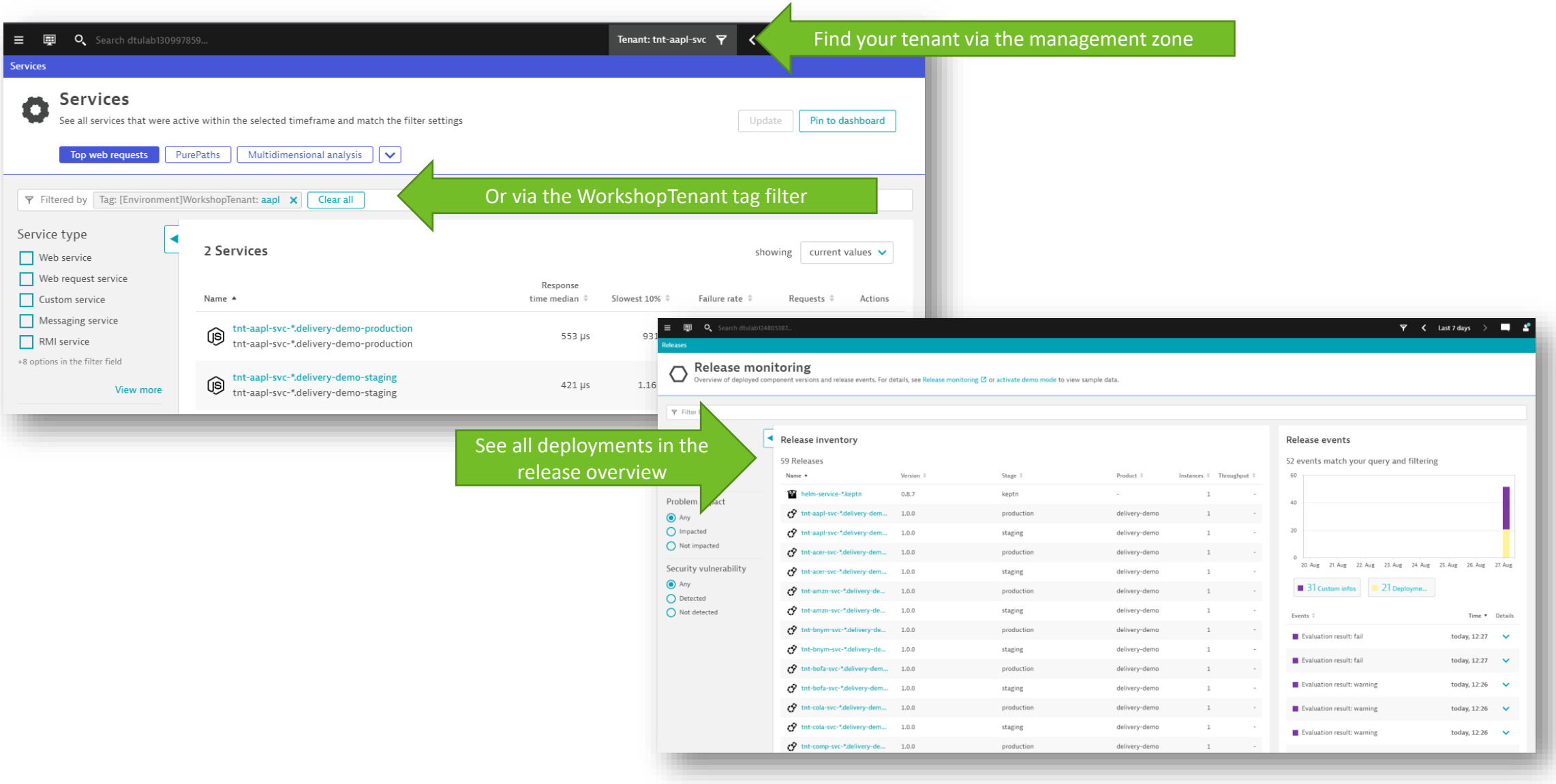
Our sample app is automatically monitored with Dynatrace

- Including automatic created Synthetic Checks in Staging and Production
- Proper automated taging and naming



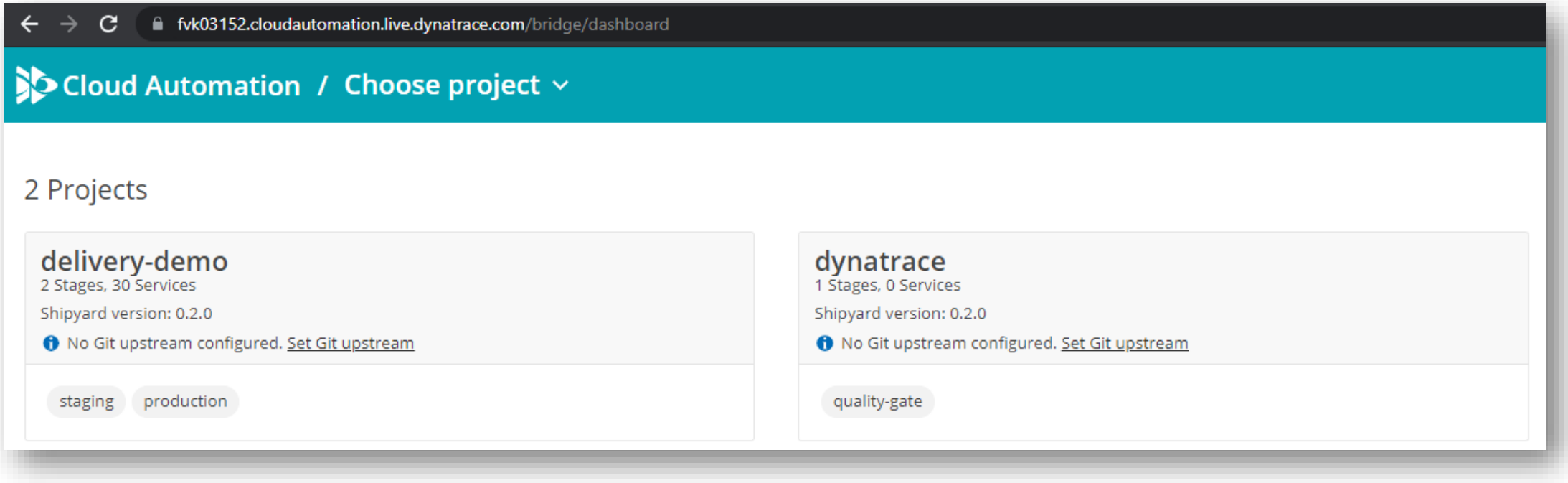


Find your app in the Dynatrace Environment



Dynatrace Cloud Automation Environment

- Login with the same Dynatrace user provided

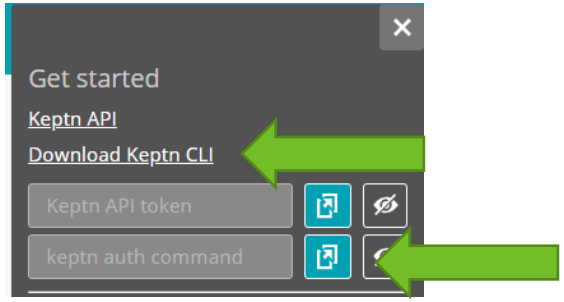


(OPTIONAL) Access Keptn CLI – via Bastion Host or locally on your machine

- Via Bastion Host
  - Connect via Putty (or equivalent SSH client): IP Address is in Excel File
  - Lets run a „*keptn status*“ command: should say its authenticated against fvk03152.xxx

```
[dtu_training@ip-10-0-1-101 ~]$ keptn status
Starting to authenticate
Successfully authenticated against the Keptn cluster https://fvk03152.cloudautomation.live.dynatrace.com/api
Using a file-based storage for the key because the password-store seems to be not set up.
[dtu_training@ip-10-0-1-101 ~]$
```

- Locally on your laptop
  - First install the Keptn CLI
  - Then authenticate it by copying kepth auth from the UI



Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file

					Validate successful access of environments		
Attendee	How familiar are you with Dynatrace? use it daily; occassionally; never	Workshop Tenant ID	OK to record session?	Claim your ID	Dynatrace Environment? Login with username / pwd given	Cloud Automation SaaS? (login with username / pwd given)	Access YOUR sample tenant app in production and staging?
Andreas Grabner		aapl	Done				

Lab 1: Production Reliability

Creating SLOs and putting them on a dashboard

dynatrace

REFRESHER!!

DevOps & SRE Language: SLIs drive SLOs which inform SLAs!!!

Service Level Indicators (SLIs)

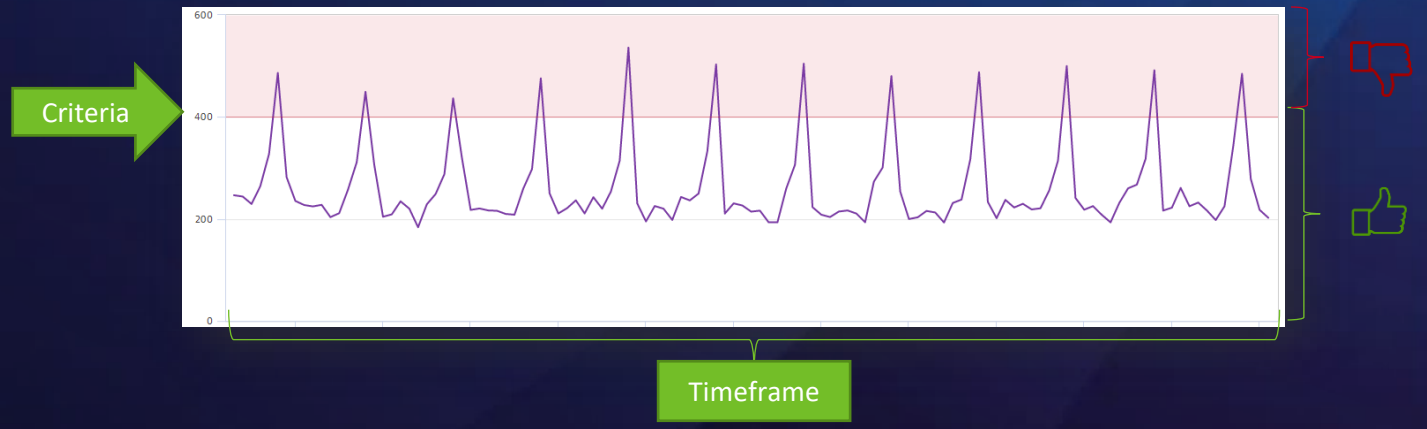
Percentage of an imporant metric against a criteria

Example: Service Response Time p95 < 400ms

Service Level Objectives (SLOs)

Success-% SLI over a timeframe

Example: p95 < 400ms in 90% of the time over 30 days



Error Budget

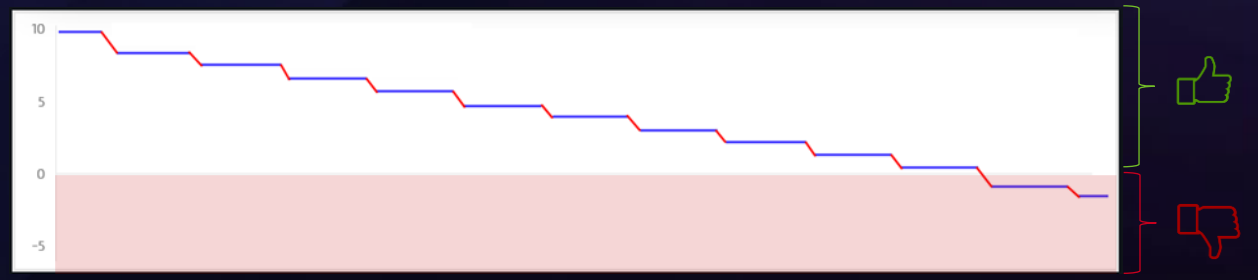
How much more impact can we afford before violating SLO?

Service Level Agreements (SLAs)

What happens IF SLO is breached

Example: Paying penalities, loosing customers ...

Error Budget: how much budget is left?



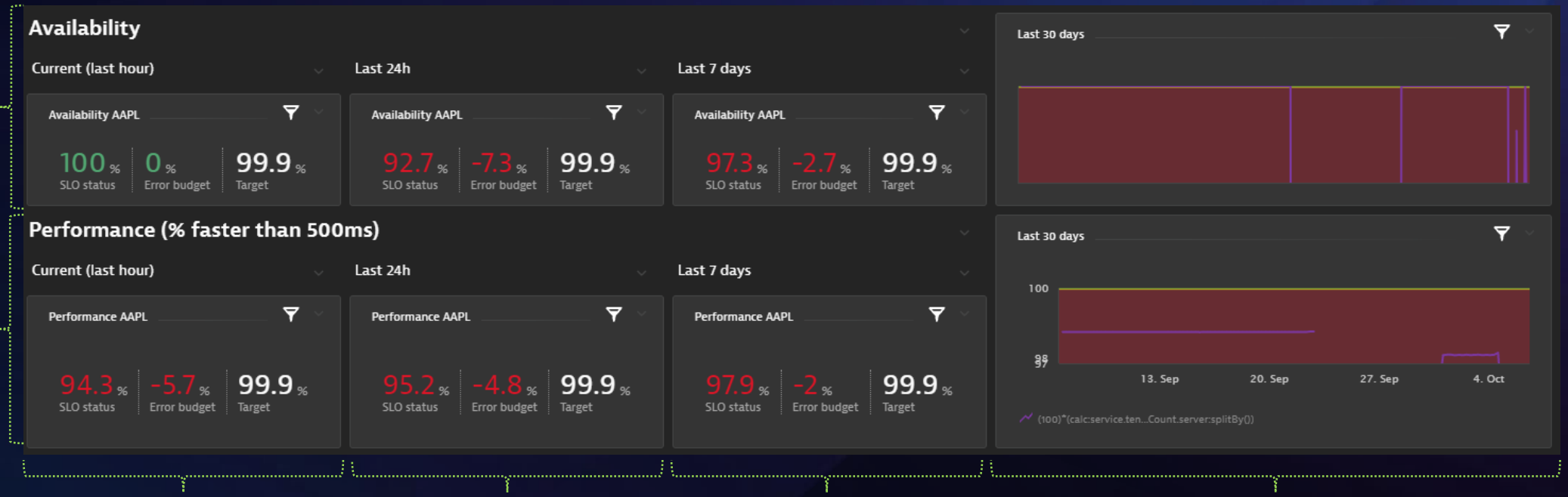
dynatrace

REFRESHER!!

A best practice SLO dashboard to start with

% of Time System is available

% of Requests meeting Performance Goal



Last 1 hour
Any current issues?

Last 24 hour
Any long running issues?

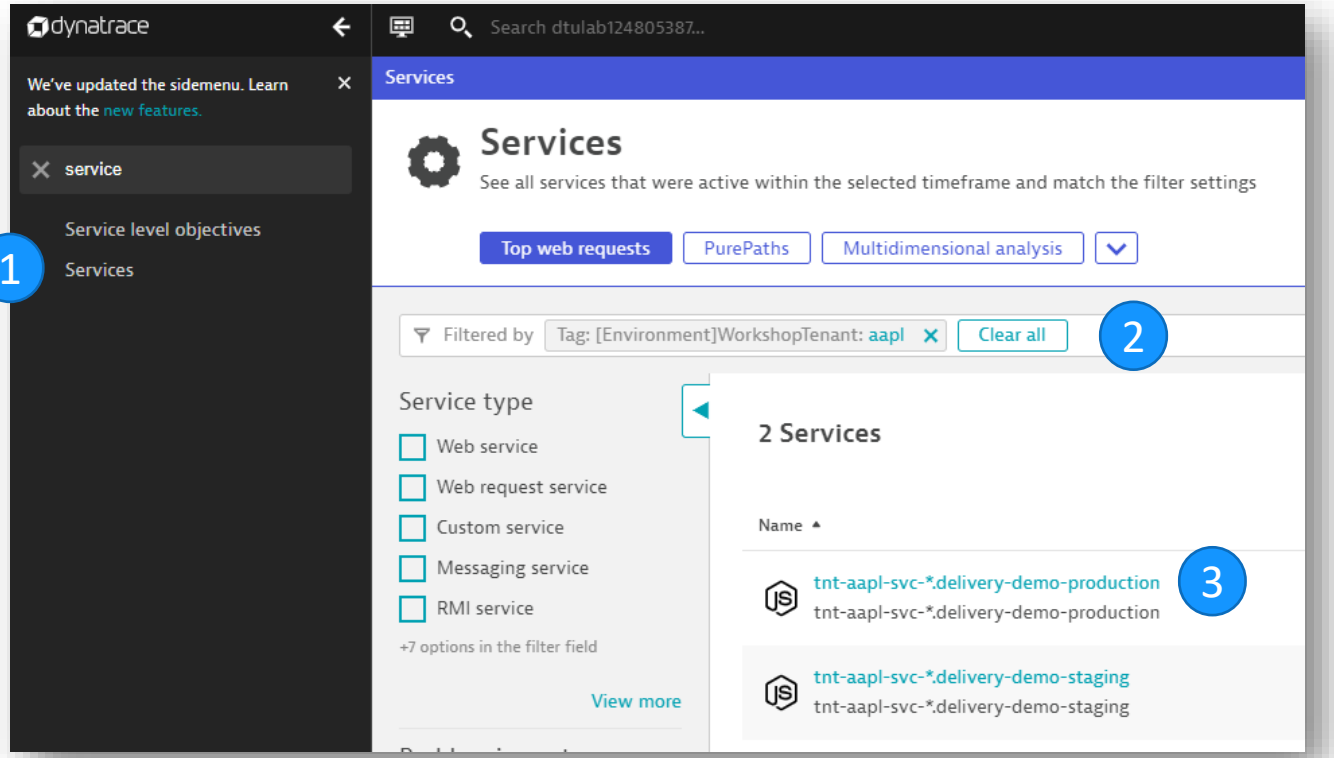
Last 7 days
Any permantent issues?

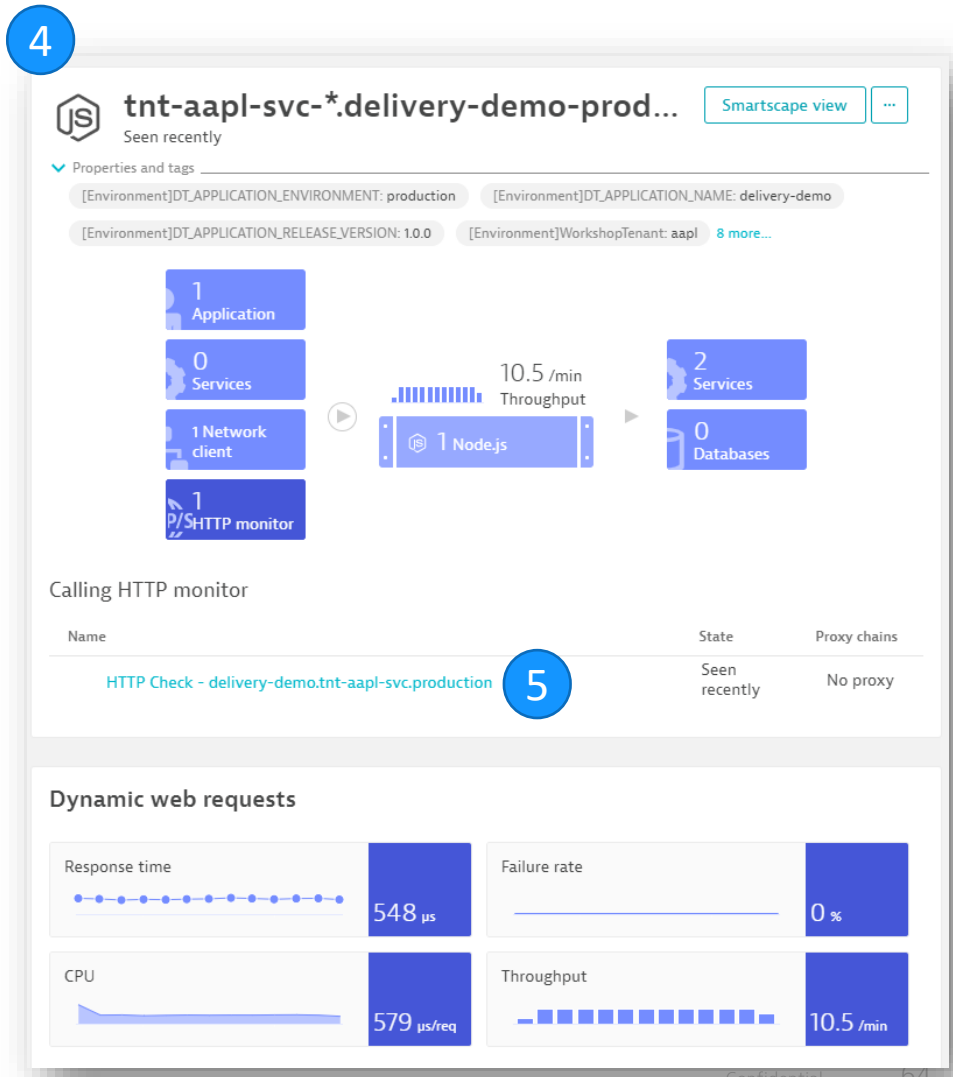
Last 30 days
Any repeting issues? Will we meet our SLO?

dynatrace

HANDS-ON SECTION

Lets quickly explore our responsible service in production

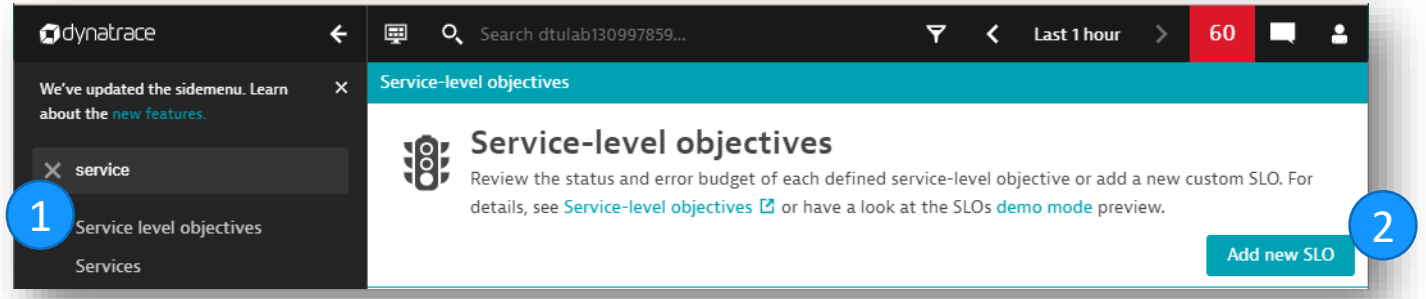


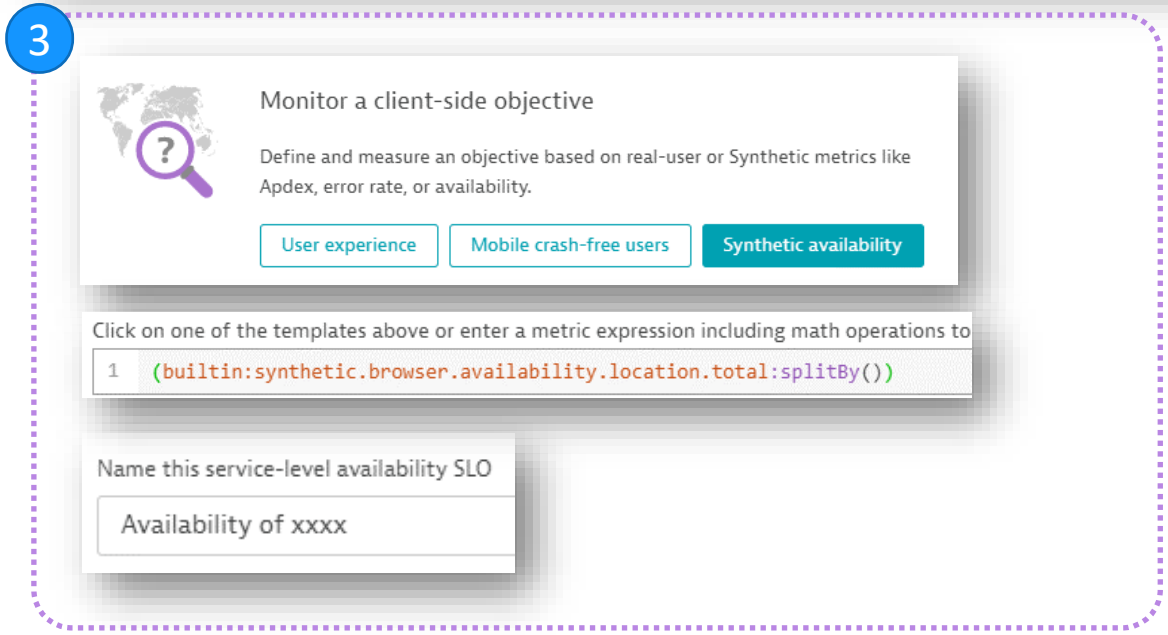


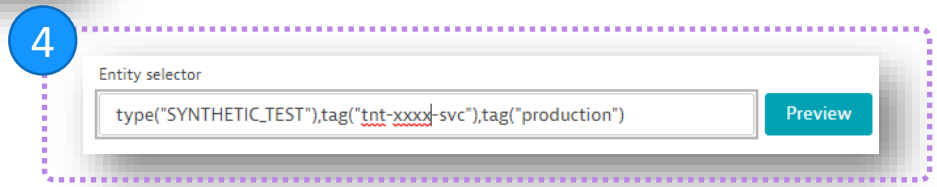
HANDS-ON SECTION

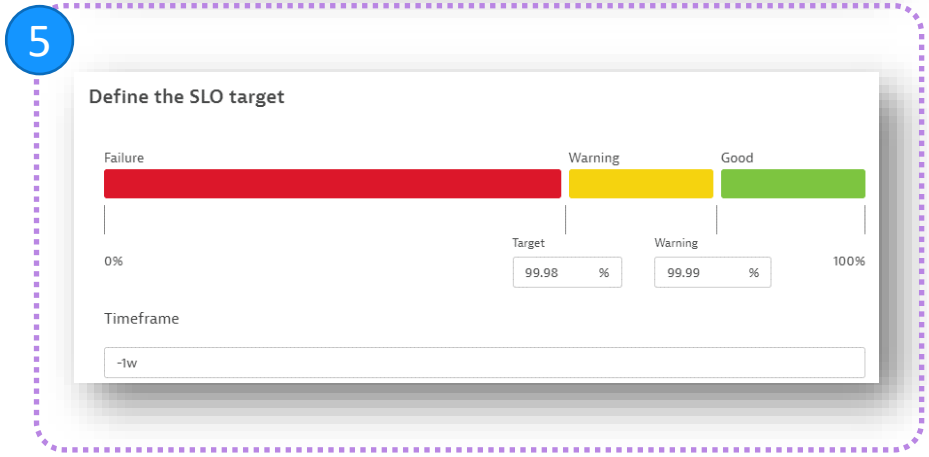
Step #1: Lets create an SLO for production availablity

- Lets create our own SLO based on your Synthetic Test Availablity of our tenant in Production







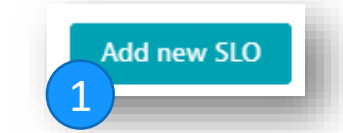


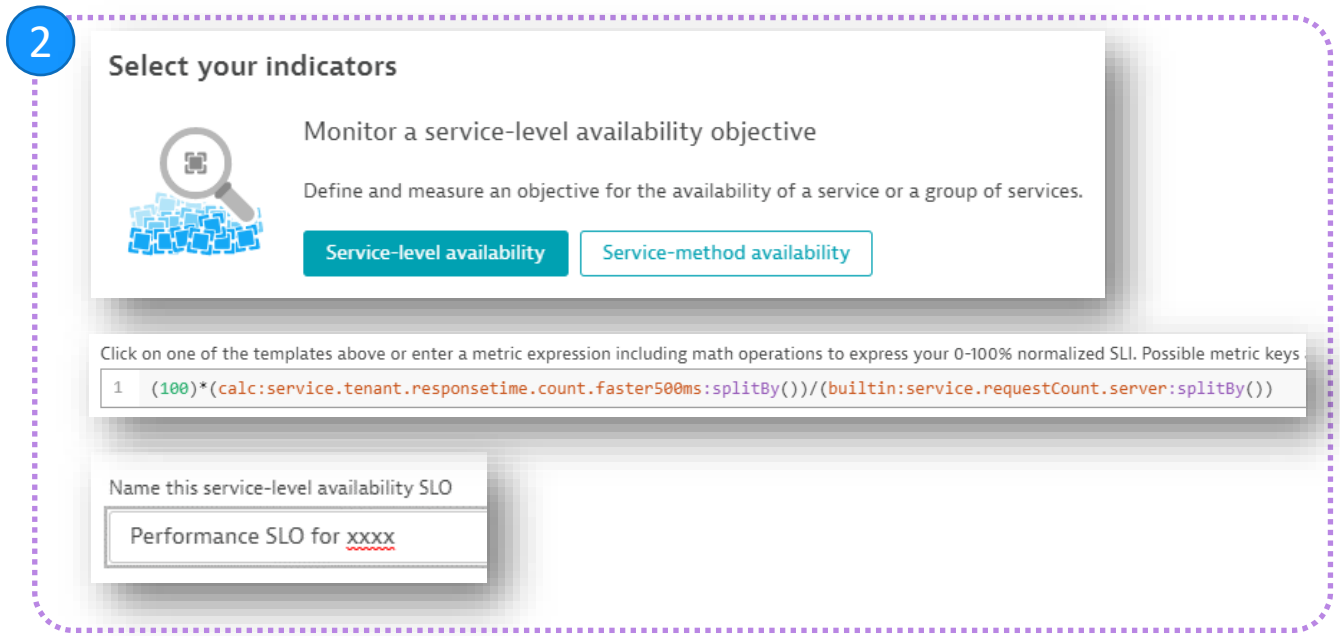
Hint: Copy queries and tags from slide notes!

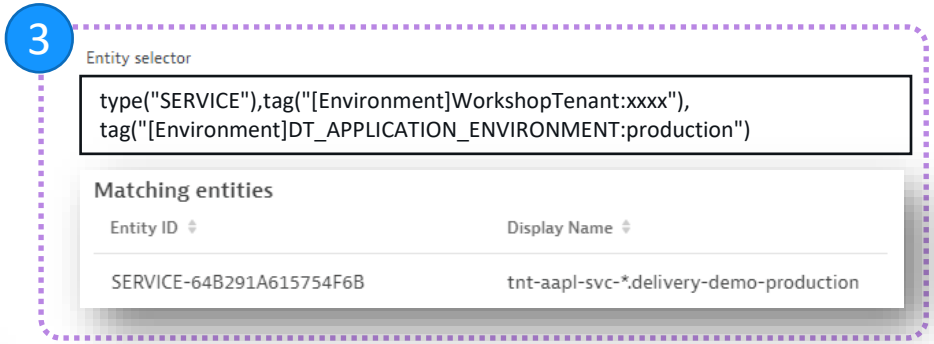
HANDS-ON SECTION

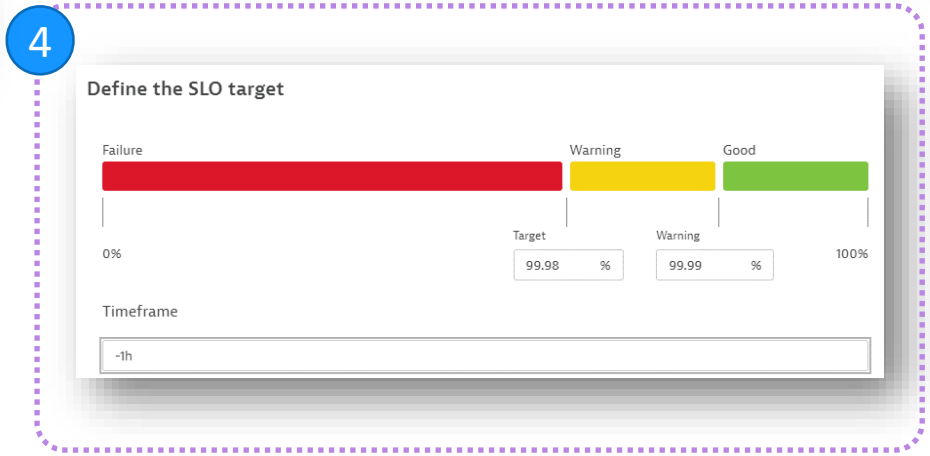
Step #2: Lets create an SLO for production performance

- Let‘s create an SLO that measures the % of requests faster than 500ms on our tenant
  - Select „Service-level Availability“ and replace nominator metric with *calc:service.tenant.responsetime.count.faster500ms*







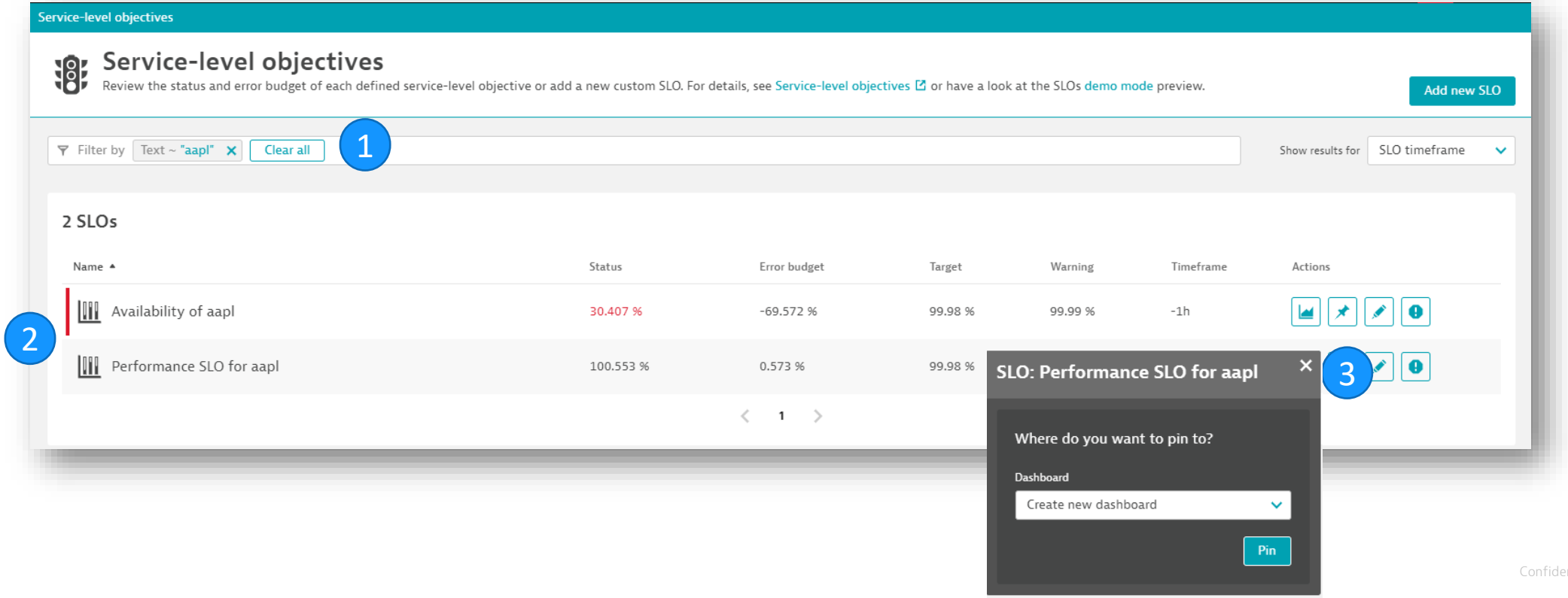


Hint: Copy queries and tags from slide notes!

HANDS-ON SECTION

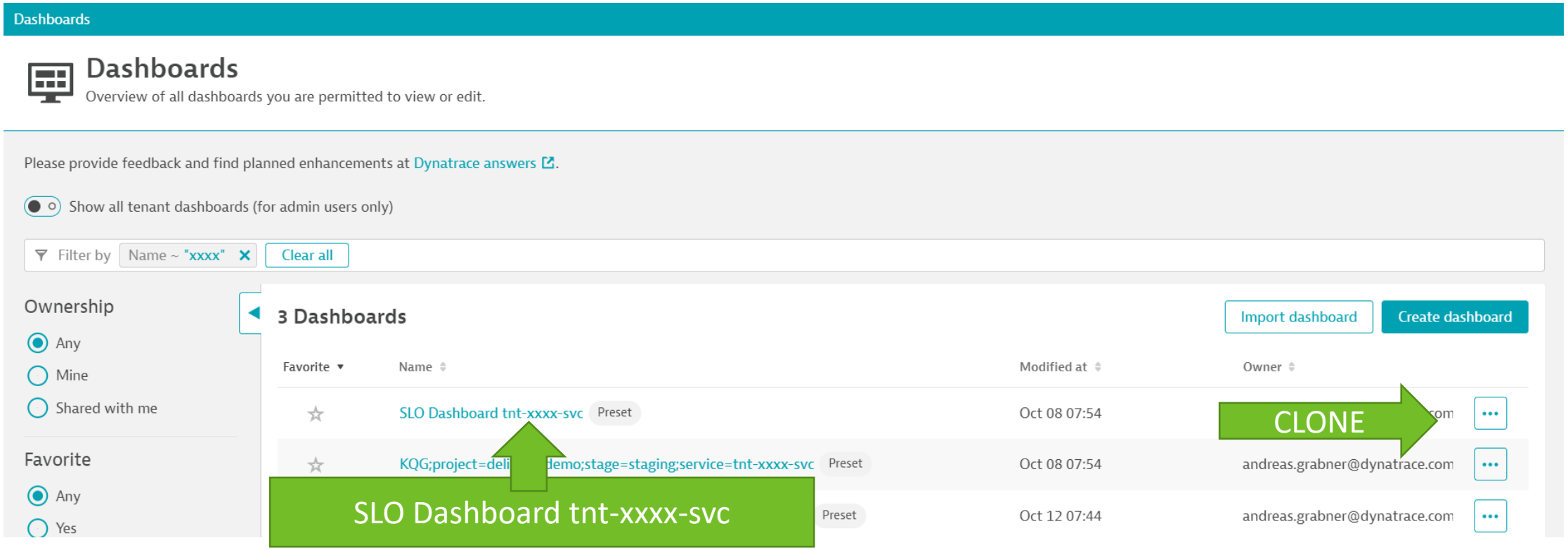
Step #3: Validate our SLOs are created and delivery data

1. Filter by name, e.g: xxxx to find your SLOs
2. Validate you see your SLOs
3. Start creating a new dashboard



HANDS-ON SECTION

Step #4 – Clone the existing dashboard template



HANDS-ON SECTION

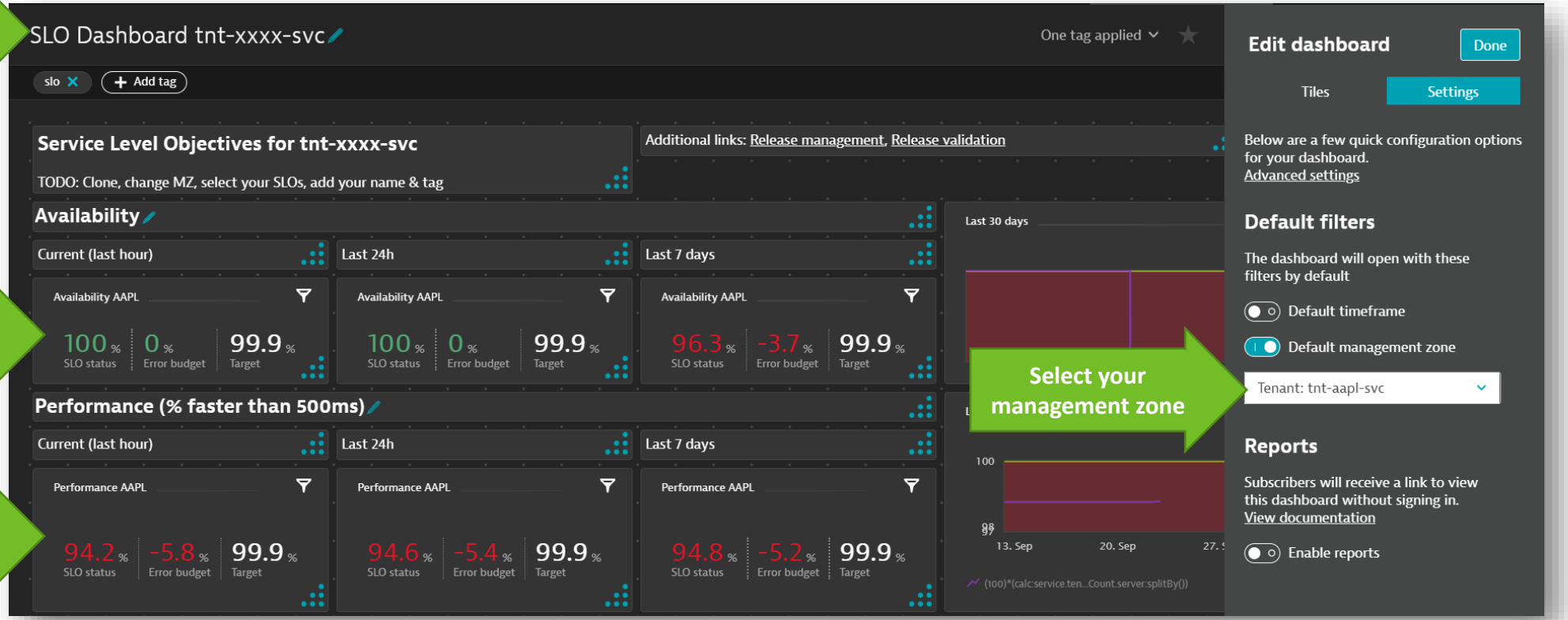
Step #5: Customize it for your tenant

- Edit the settings necessary to pick up the data from your tenant

Give it a good name

Select your Availablity SLOs and select corret timeframe

Select your Performance SLOs and select corret timeframe



Select your management zone

WATCH TRAINER SECTION

From Manual to Automated Creation of SLOs & Dashboard with Monaco

- Monaco (Monitoring as Code): https://dynatrace-oss.github.io/dynatrace-monitoring-as-code/

```
export OWNER=youremail@domain.com
export TENANT=aapl
monaco -e environment.yaml -p lab1 projects
```

slo.yaml | slo.json

keptn-on-k3s > cloudautomation > monaco > projects > lab1 > slo > slo.json > ...

```
{
    "enabled": true,
    "name": "{{ .name }}",
    "description": "{{ .description }}",
    "useRateMetric": true,
    "metricExpression": "{{ .expression }}",
    "metricRate": "",
    "metricDenominator": "",
    "metricNumerator": "",
    "evaluationType": "AGGREGATE",
    "filter": "{{ .filter }}",
    "target": "{{ .target }}",
    "warning": "{{ .warning }}",
    "timeframe": "{{ .timeframe }}"
}
```

slo.yaml | slo.json

keptn-on-k3s > cloudautomation > monaco > projects > lab1 > slo > slo.yaml > [ ] performanceSlo > timeframe

```
config:
  - availabilitySlo: "slo.json"
  - performanceSlo: "slo.json"

availabilitySlo:
  - name: "Availability of {{ .Env.TENANT_ID }}"
  - description: "% of time {{ .Env.TENANT_ID }} service is available based on synthetic test"
  - expression: "(builtin:synthetic.browser.availability.location.total:splitBy())"
  - filter: "mzName(\\\"Tenant: tnt-{{ .Env.TENANT_ID }}-svc\\\"),type(\\\"SYNTHETIC_TEST\\\")"
  - target: "95.00"
  - warning: "99.00"
  - timeframe: "-1d"

performanceSlo:
  - name: "Performance SLO of {{ .Env.TENANT_ID }}"
  - description: "% of requests handled by {{ .Env.TENANT_ID }} service faster than 500ms"
  - expression: "(100)*(calc:service.tenant.responsetime.count.faster500ms:splitBy())/(builtin:servi
  - filter: "mzName(\\\"Tenant: tnt-{{ .Env.TENANT_ID }}-svc\\\"),type(\\\"SERVICE\\\"),tag(\\\"[Env
  - target: "85.00"
  - warning: "90.00"
  - timeframe: "-1d"
```

Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file

Production Reliability		
Created the SLO based on Synthetic Availablity	Created SLO based on Response Time	Created an SLO dashboard

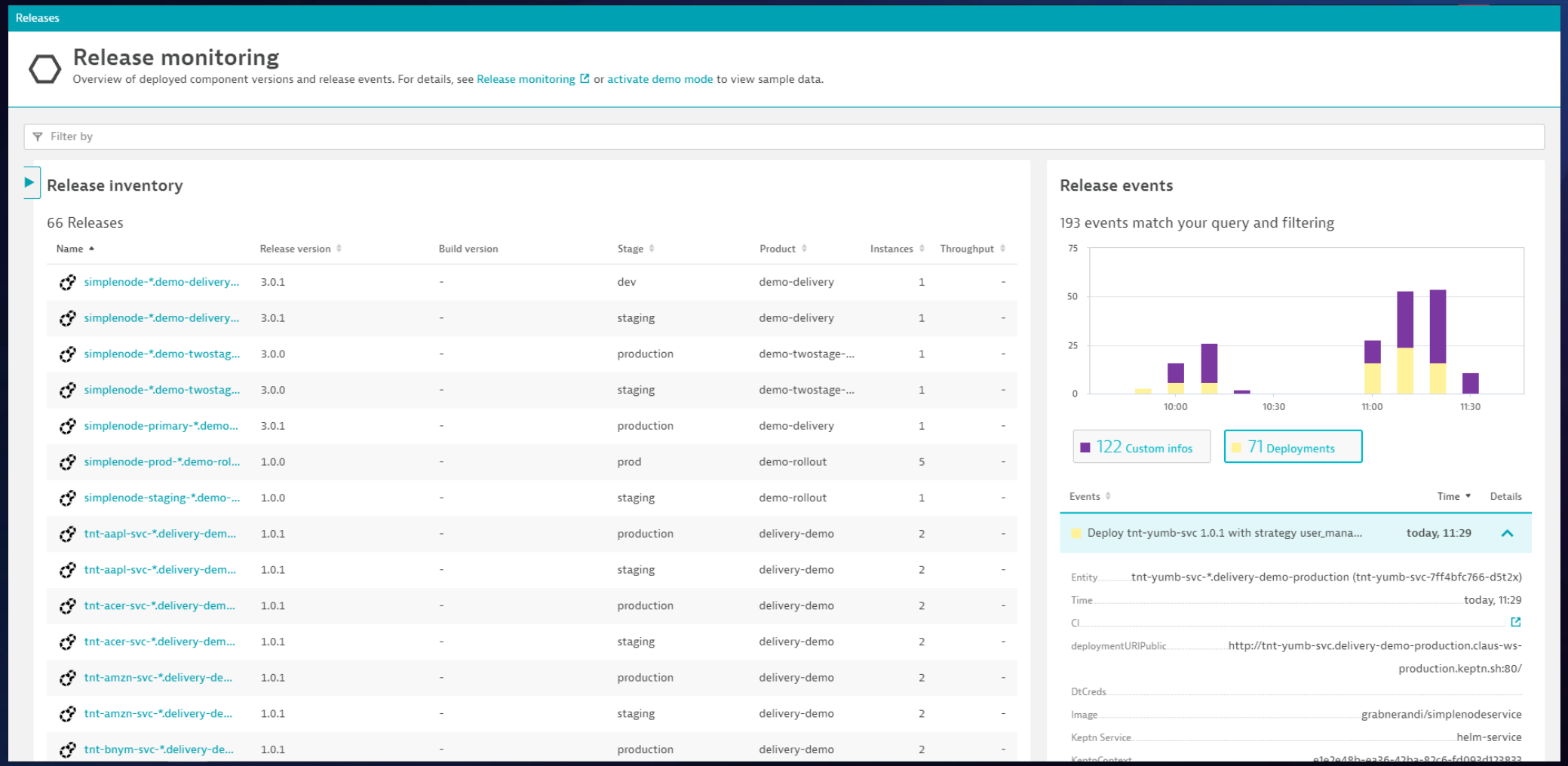
Lab 2: Release Validation

Automating release validation through an SLO Dashboard

dynatrace

REFRESHER!!

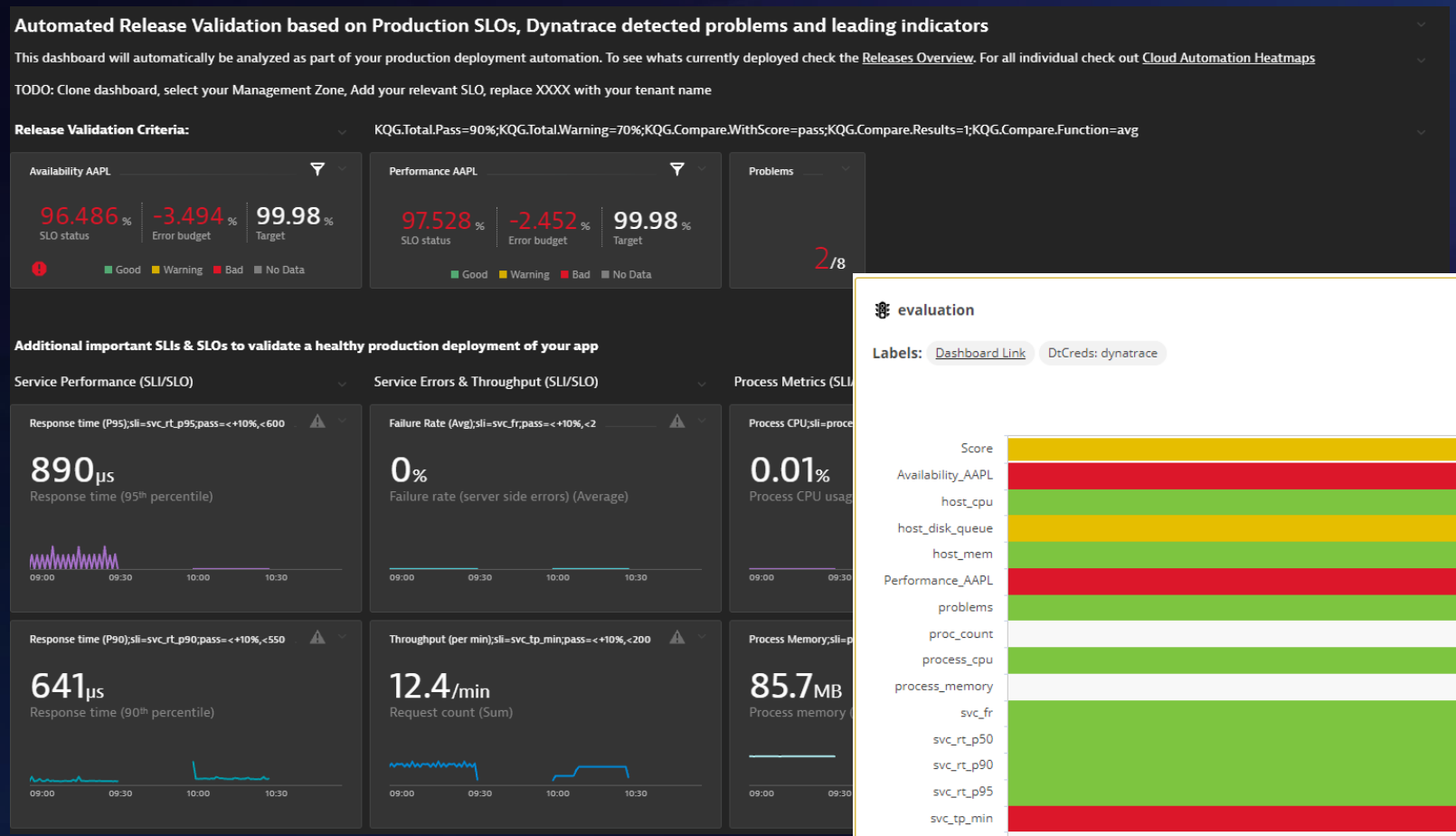
Reminder: Release monitoring overview based on meta data

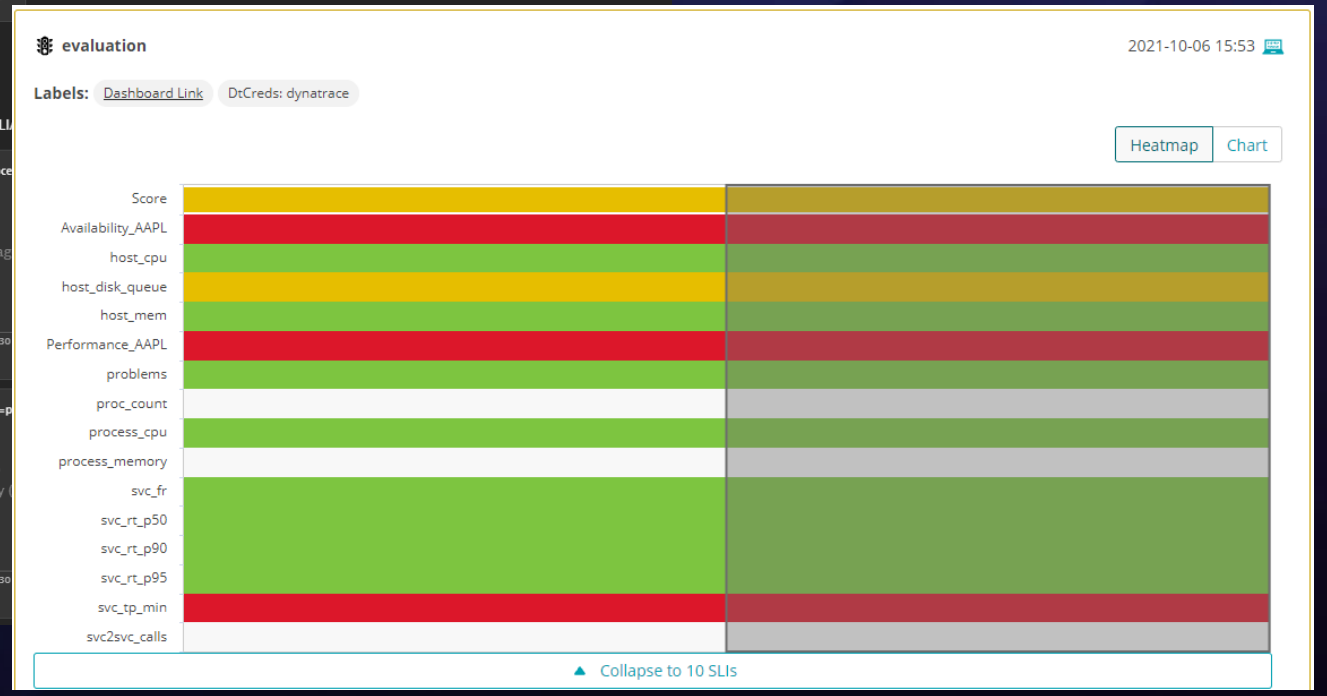


dynatrace

REFRESHER!!

„To Keep or Not?“ - Automate Release Validation

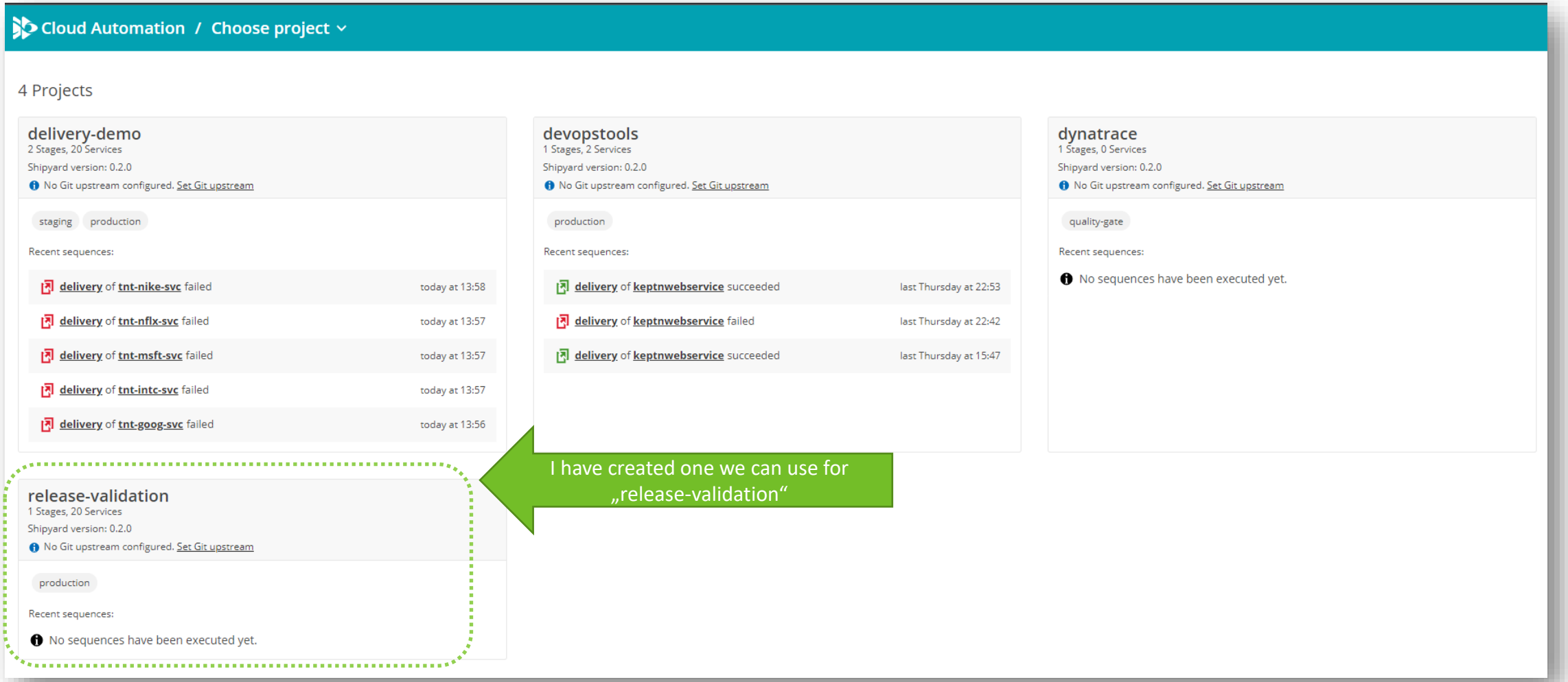




dynatrace

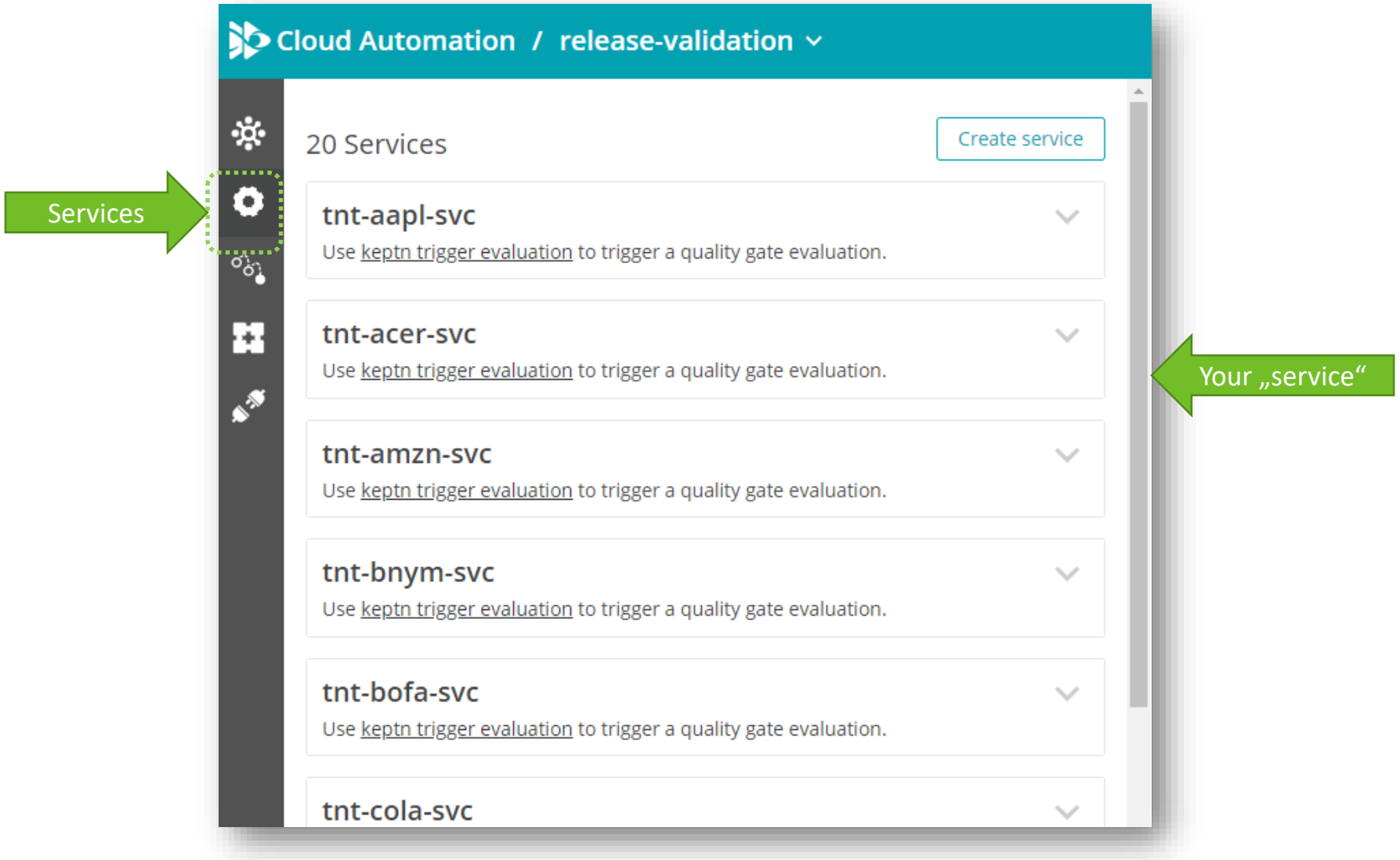
WATCH TRAINER SECTION

After you login to Cloud Automation you see several projects



WATCH TRAINER SECTION

I have created „services“ already for all our tnt-XXXX-svc



WATCH TRAINER SECTION

How does dashboards get analyzed when used in automation?

Dashboard name links to Cloud Automation
KQG;project=xxx,stage=xxx,service=xxx

Pulled Value via API and compared against	Points Normalized to 100
Availability: 96.296 <=99.98	0
Response Time: 1.14 <600 and <=+10% increase	25
Failure Rate: 0.25 < 2 and <=+10% increase	25
Open Problems: 2 Always compared against 0	0
Total Score Compared based on pass/warn	50

WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Now lets create a dashboard that we can use for release validation automation

- The automation will look for a Dynatrace dashboard with the following naming schema:
  - KQG;project=<PROJECT>;stage=<STAGE>;service=<SERVICE>
- In our case *project=release-validation*, *stage=production* and *service=tnt-xxxx-svc*
- We could start from scratch, or …..
  - start from a template ☺
  - Use Monaco ☺

WATCH TRAINER SECTION

Using Monaco to create release validation dashboard

- Monaco (Monitoring as Code): https://dynatrace-oss.github.io/dynatrace-monitoring-as-code/

```
export OWNER=youremail@domain.com
export TENANT=aapl
monaco -e environment.yaml -p lab2 projects
```

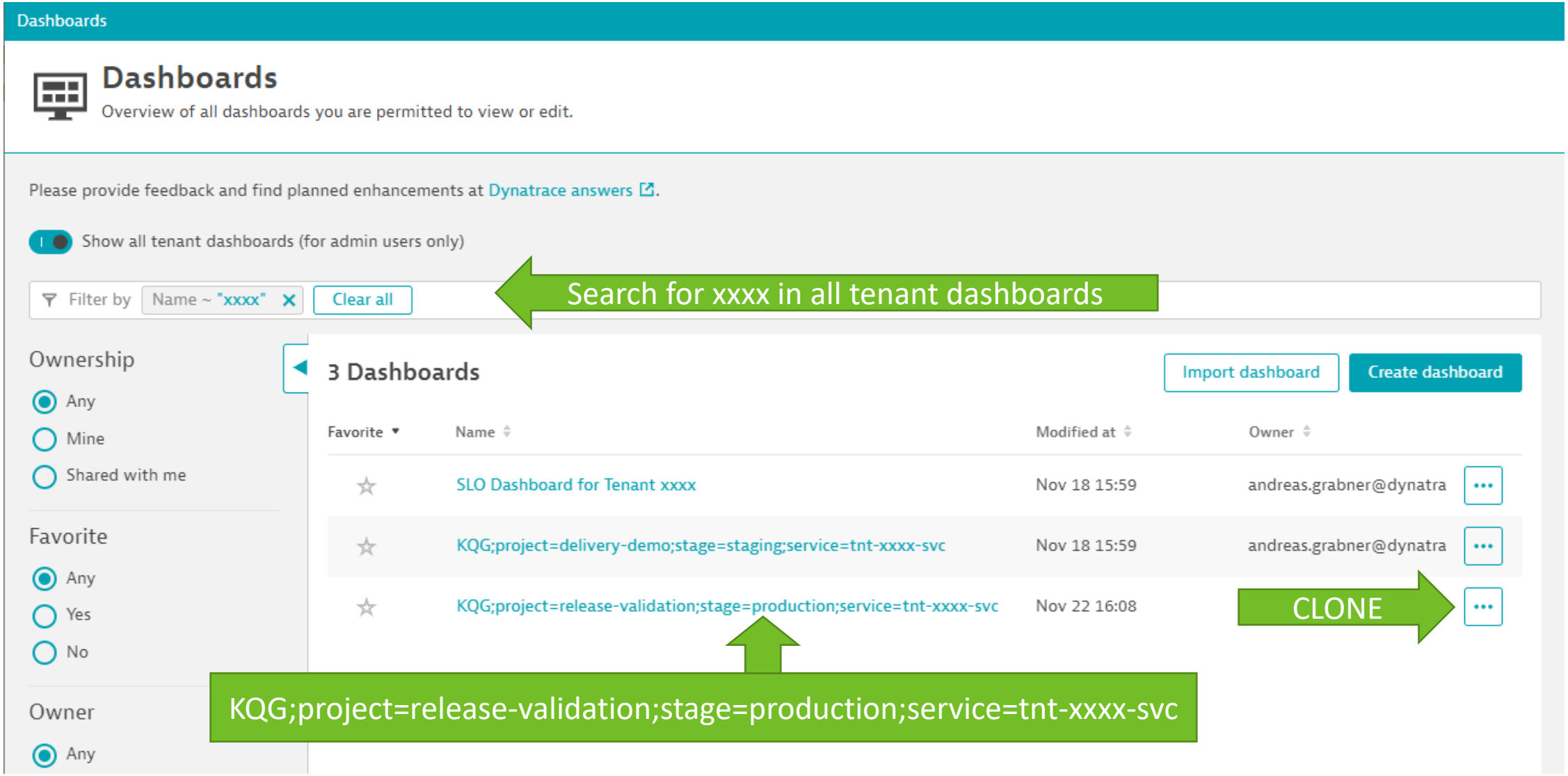
slo.yaml | dashboard.yaml

keptn-on-k3s > cloudautomation > monaco > projects > lab2 > dashboard > dashboard.yaml > [ ] release_validation_dashboard > bridgeLink

```
1   config:
2     - release_validation_dashboard: "default_release_validation_dashboard.json"
3
4   release_validation_dashboard:
5     - name: "KQG;project=release-validation;stage=production;service=tnt-{{ .Env.TENANT_ID }}-svc"
6     - owner: "{{ .Env.OWNER_EMAIL }}"
7     - tenant: "{{ .Env.TENANT_ID }}"
8     - availabilitySloId: "/lab1/slo/availabilitySlo.id"
9     - performanceSloId: "/lab1/slo/performanceSlo.id"
10    - mzId: "/general/management-zone/zone.id"
11    - mzName: "/general/management-zone/zone.name"
12    - bridgeLink: "https://{{ .Env.KEPTN_CONTROL_PLANE_DOMAIN }}/bridge"
```

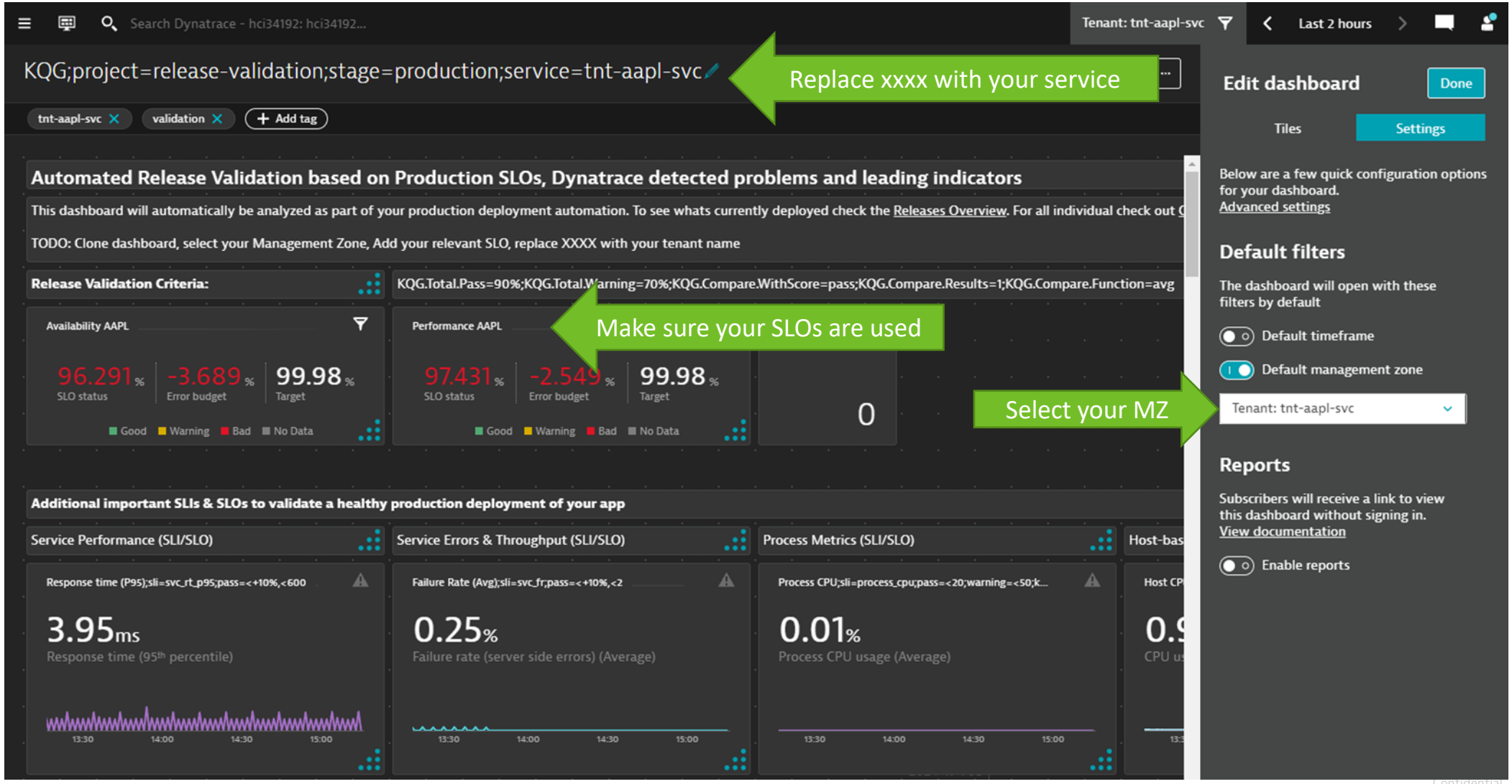
HANDS-ON SECTION: If you DIDNT already create the dashboards with Monaco

Step 1 – Clone the existing dashboard template



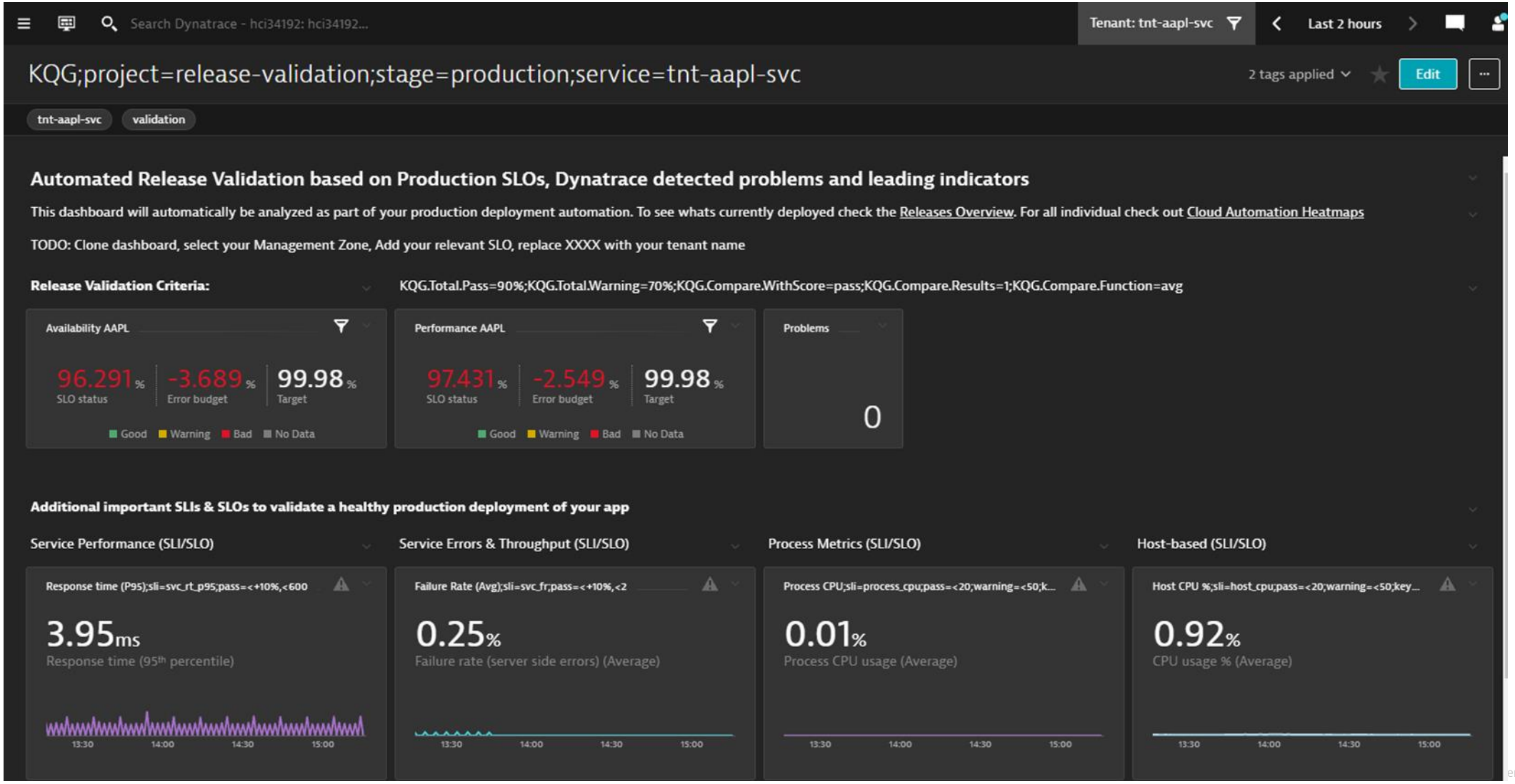
HANDS-ON SECTION: If you DIDNT already create the dashboards with Monaco

Step 2 – Give it proper name, change MZs and add your SLOs



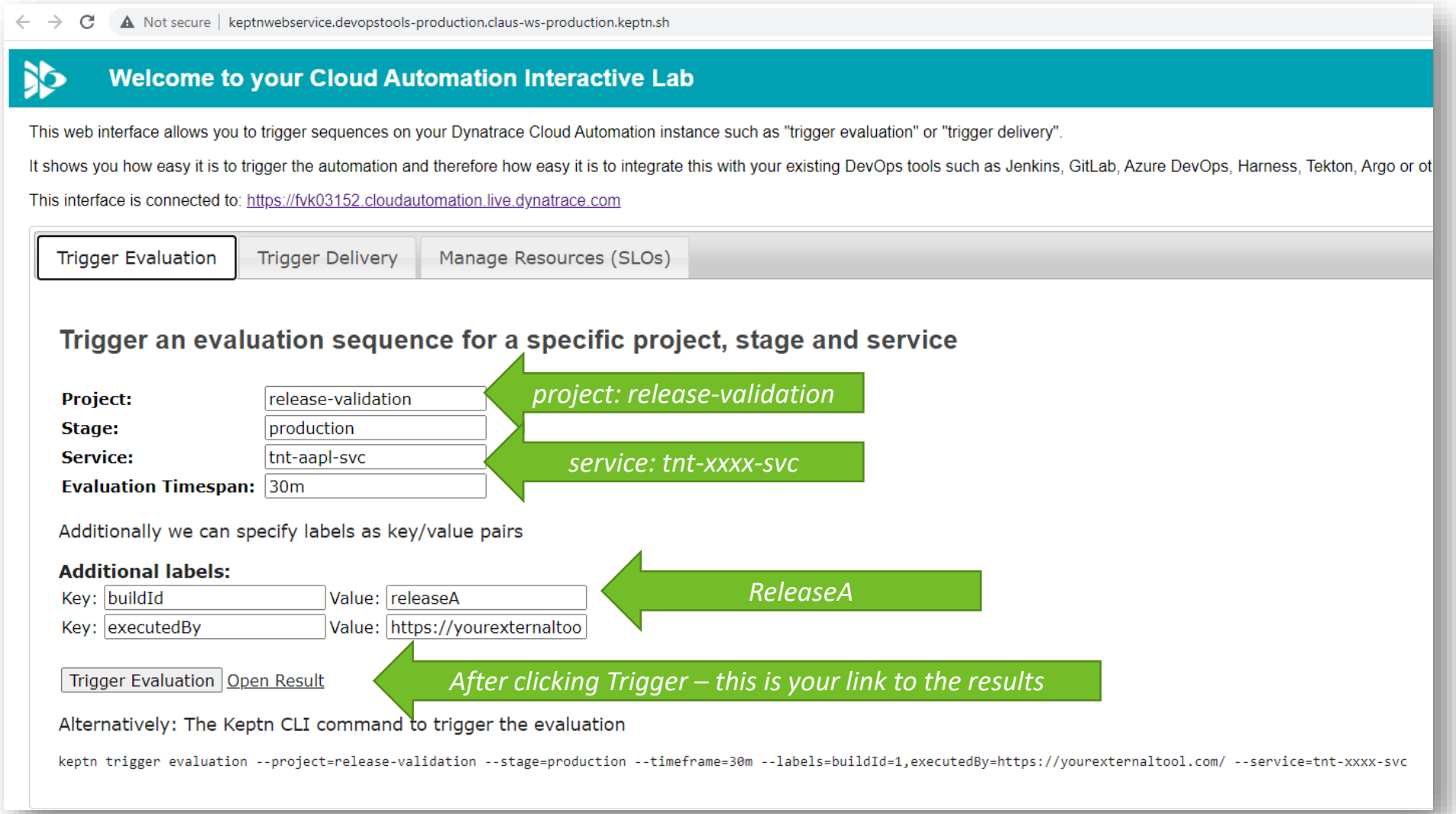
HANDS-ON SECTION: If you DIDNT already create the dashboards with Monaco

Step 3 – Save the dashboard!



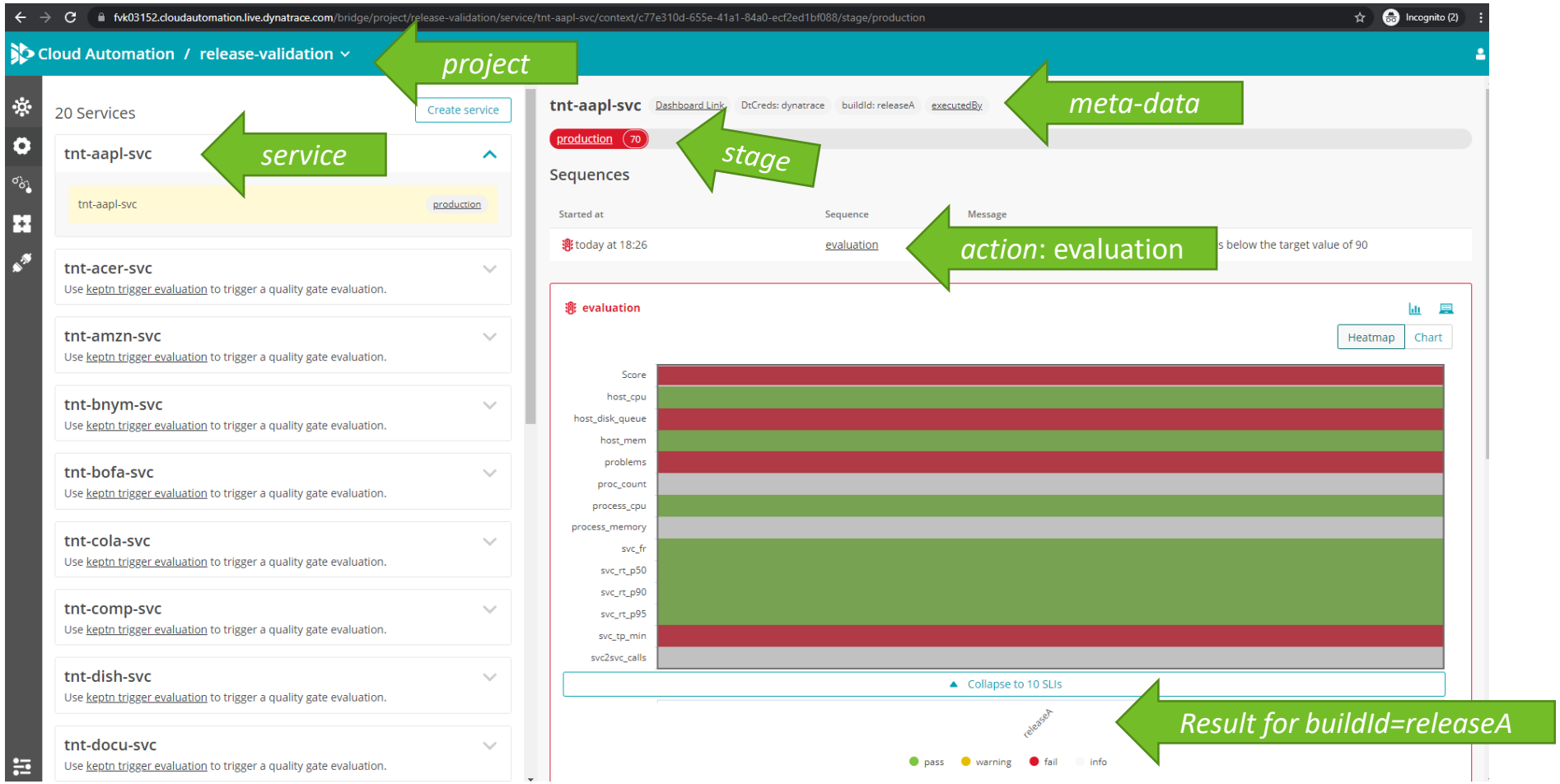
HANDS-ON SECTION

Step 4a: Trigger through our „Simple CI/CD Tool“



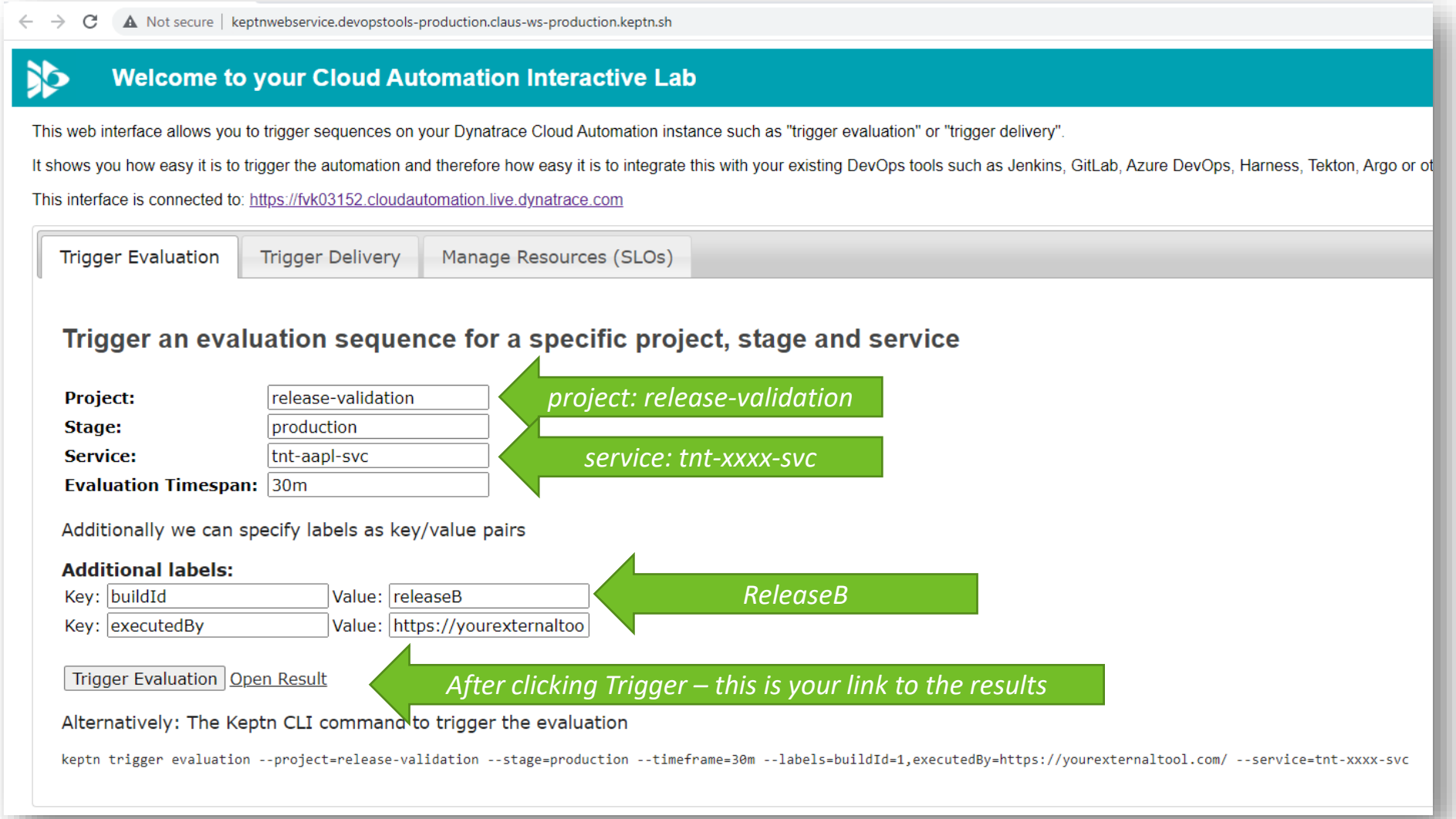
HANDS-ON SECTION

Step 4a: Analyze result for „Release A“



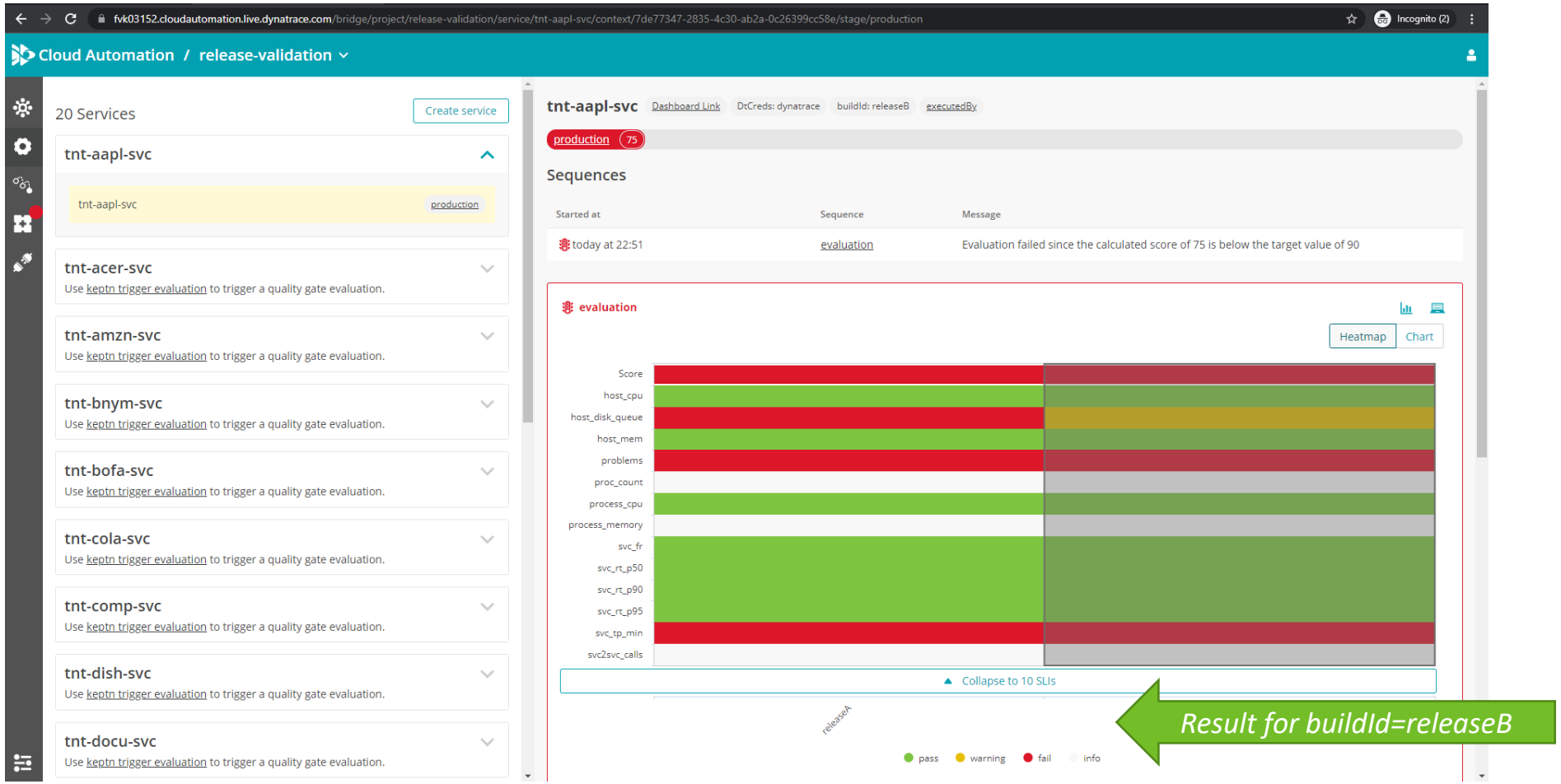
HANDS-ON SECTION

Step 4a: Trigger through our „Simple CI/CD Tool“



HANDS-ON SECTION

Step 4a: Analyze result for „Release B“



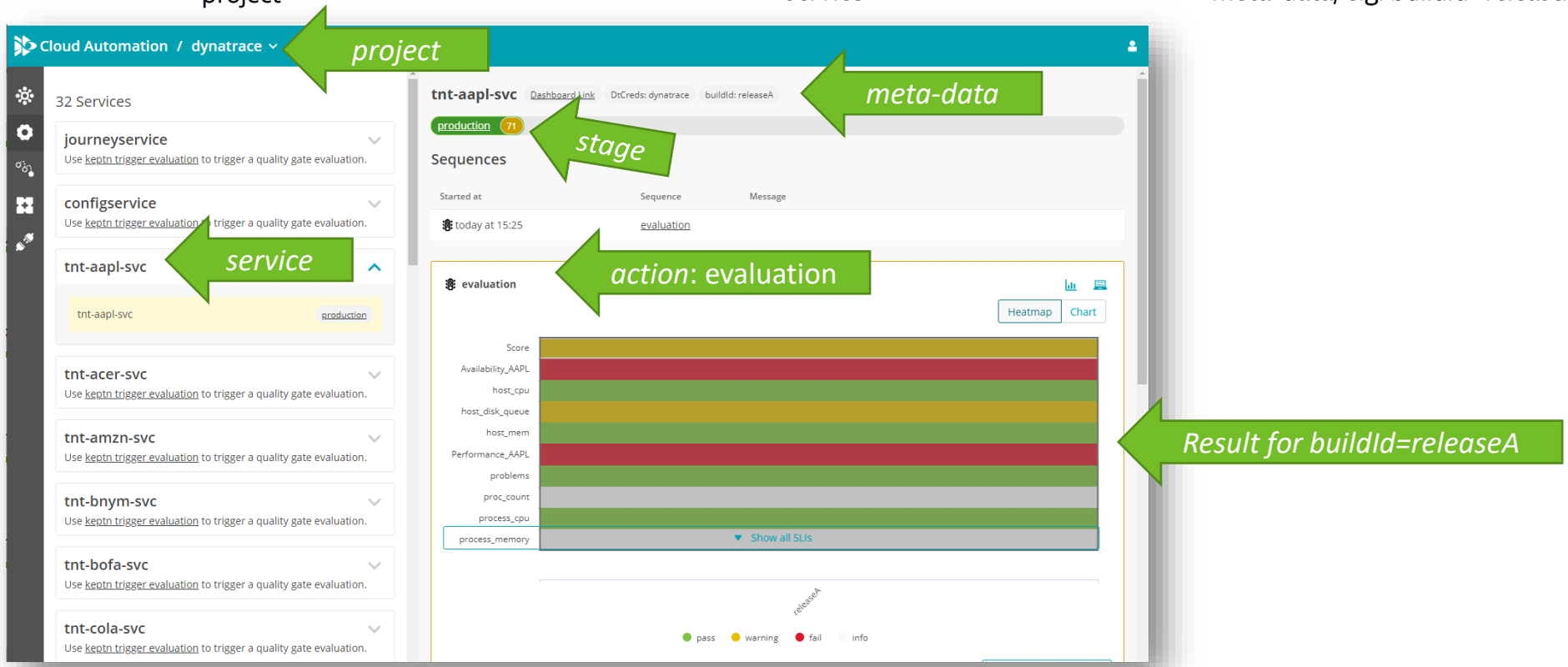
HANDS-ON SECTION

Step 4b: Using the CLI Trigger an evaluation for your service

action | stage | evaluation timeframe

```
$ keptn trigger evaluation --project=release-validation --stage=production --service=tnt-xxxx-svc --timeframe=30m --labels=buildId=releaseA
```

project | service | meta-data, e.g: buildId=releaseA



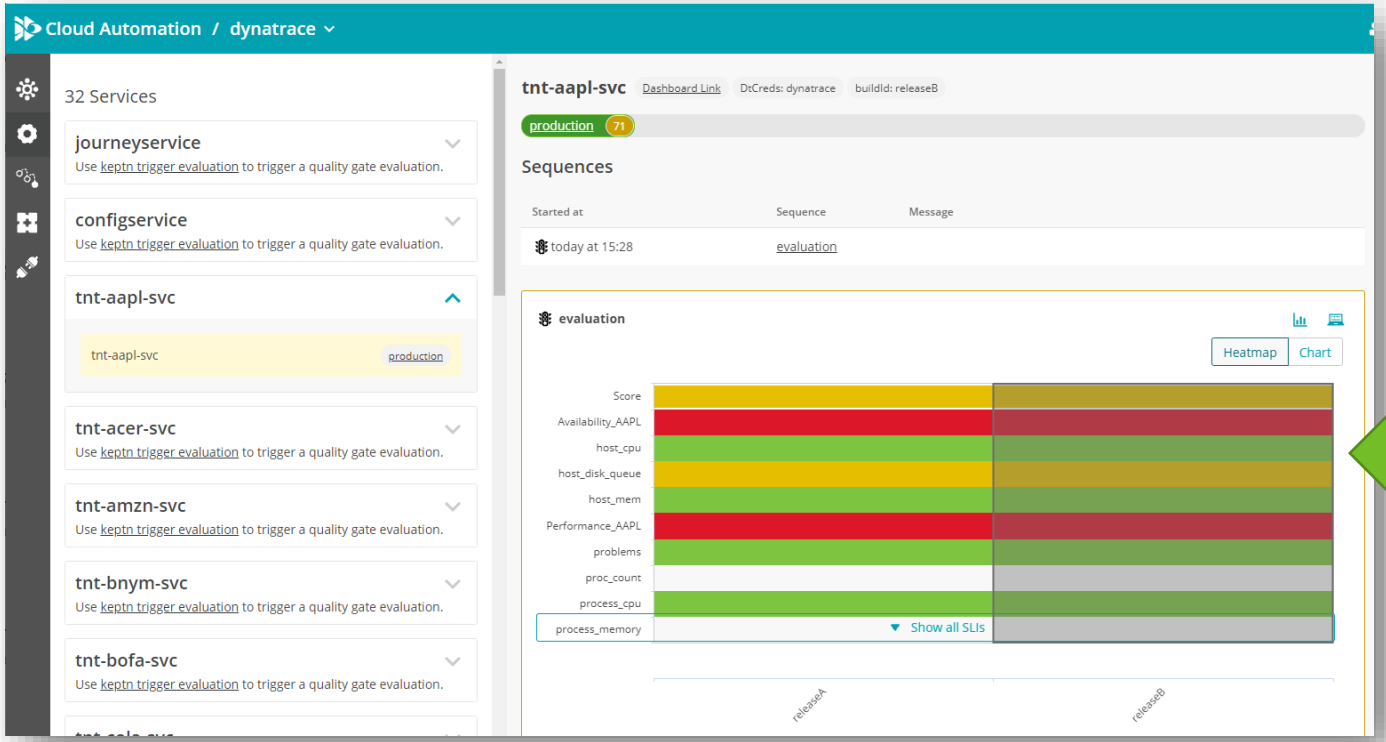
Hint: copy command from slide notes!

HANDS-ON SECTION

Step 4b: Trigger another evaluation through the CLI

```
$ keptn trigger evaluation --project=release-validation --stage=production --service=tnt-xxxx-svc --timeframe=30m --labels=buildId=releaseB
```

meta-data, e.g: buildId=releaseB

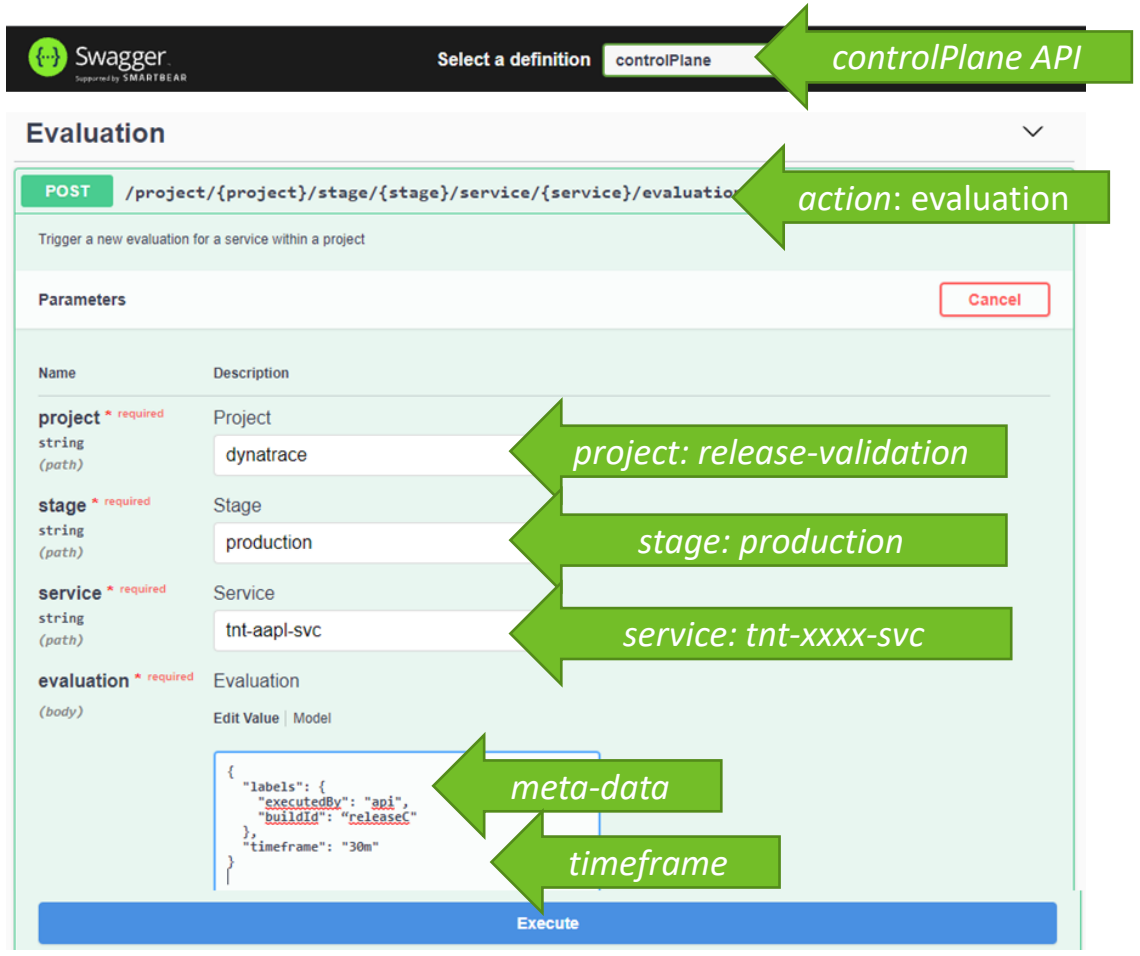


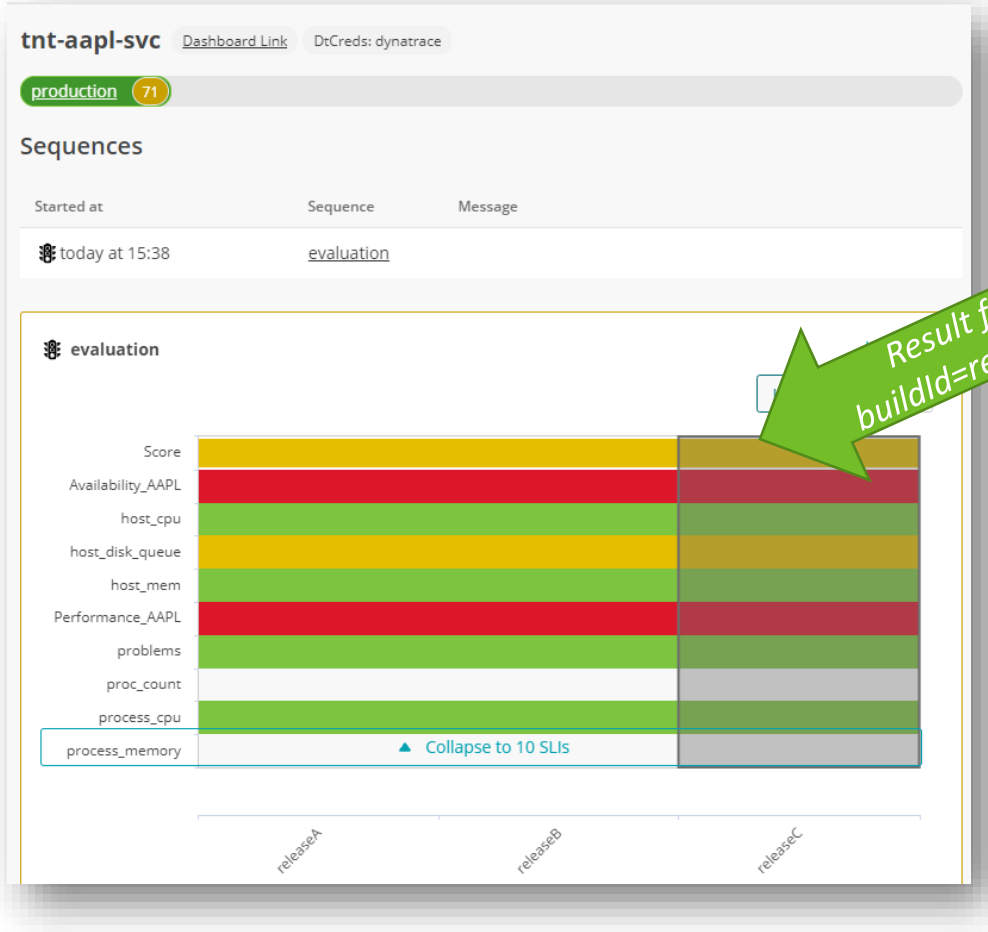
Result for buildId=releaseB

Hint: copy command from slide notes!

Step 4c: Trigger an evaluation through the API

- Open the Swagger UI; select the controlPlane API, Authenticate with the API Token, execute the evaluation endpoint



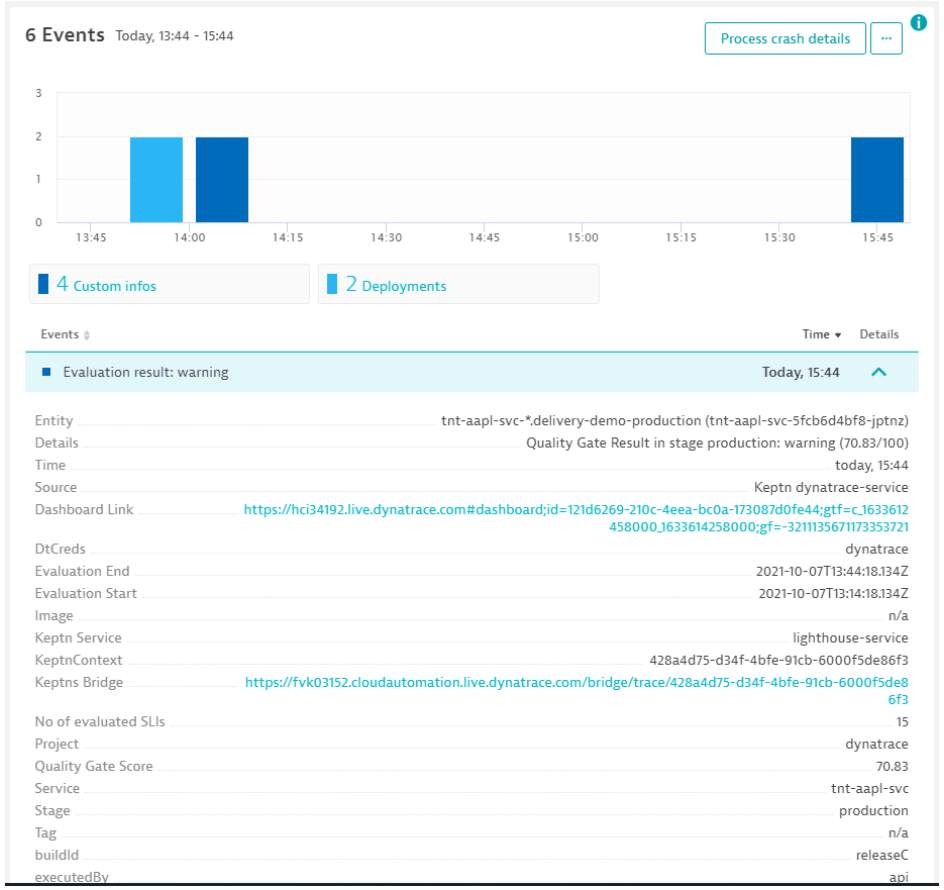


Hint: copy HTTP body from slide notes!

HANDS-ON SECTION

Phase #2 – Step 4: Automation Events also available in Dynatrace

- Evaluation Events also sent to Dynatrace monitored entities



Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file

Release Validation		
Create/Clone your Release Validation Dashboard	Trigger an evaluation for Release A	Trigger evaluation for Release B, C

Lab 3: Delivery Pipelines

Automating Quality Gates as part of Delivery Pipelines

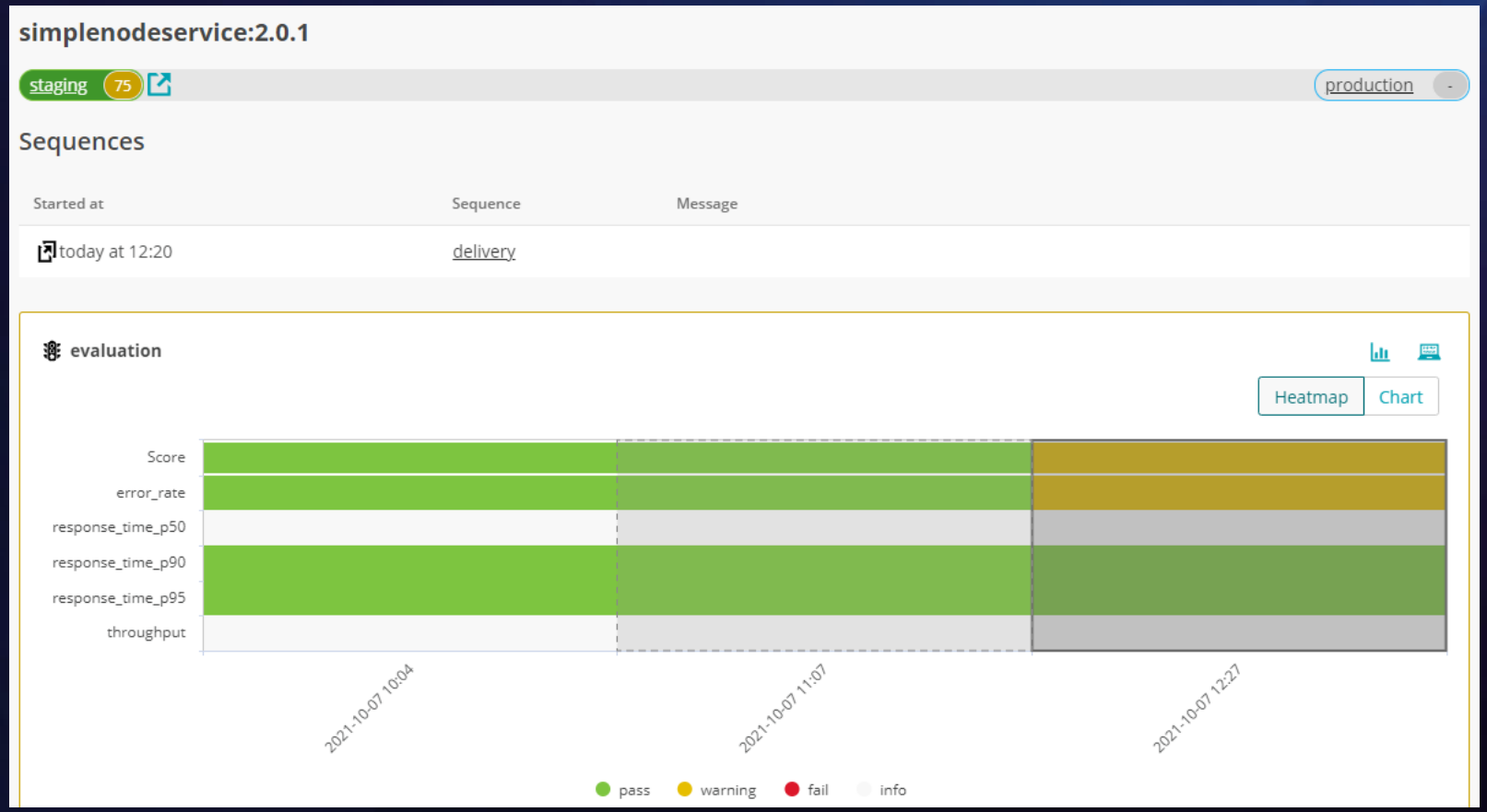
dynatrace

REFRESHER!!

Demo: SLO evaluation part of DevOps delivery

SLO

```
spec_version: '0.1.0'
comparison:
  compare_with: "single_result"
  include_result_with_score: "pass"
  aggregate_function: avg
objectives:
  - sli: response_time_p95
    pass:
      - criteria:
          - "<=1000"
    warning:
      - criteria:
          - "<=1500"
  - sli: throughput
  - sli: error_rate
    weight: 2
    pass:
      - criteria:
          - "<=1"
    warning:
      - criteria:
          - "<=2"
  - sli: response_time_p50
  - sli: response_time_p90
    pass:
      - criteria:
          - "<=500"
    warning:
```

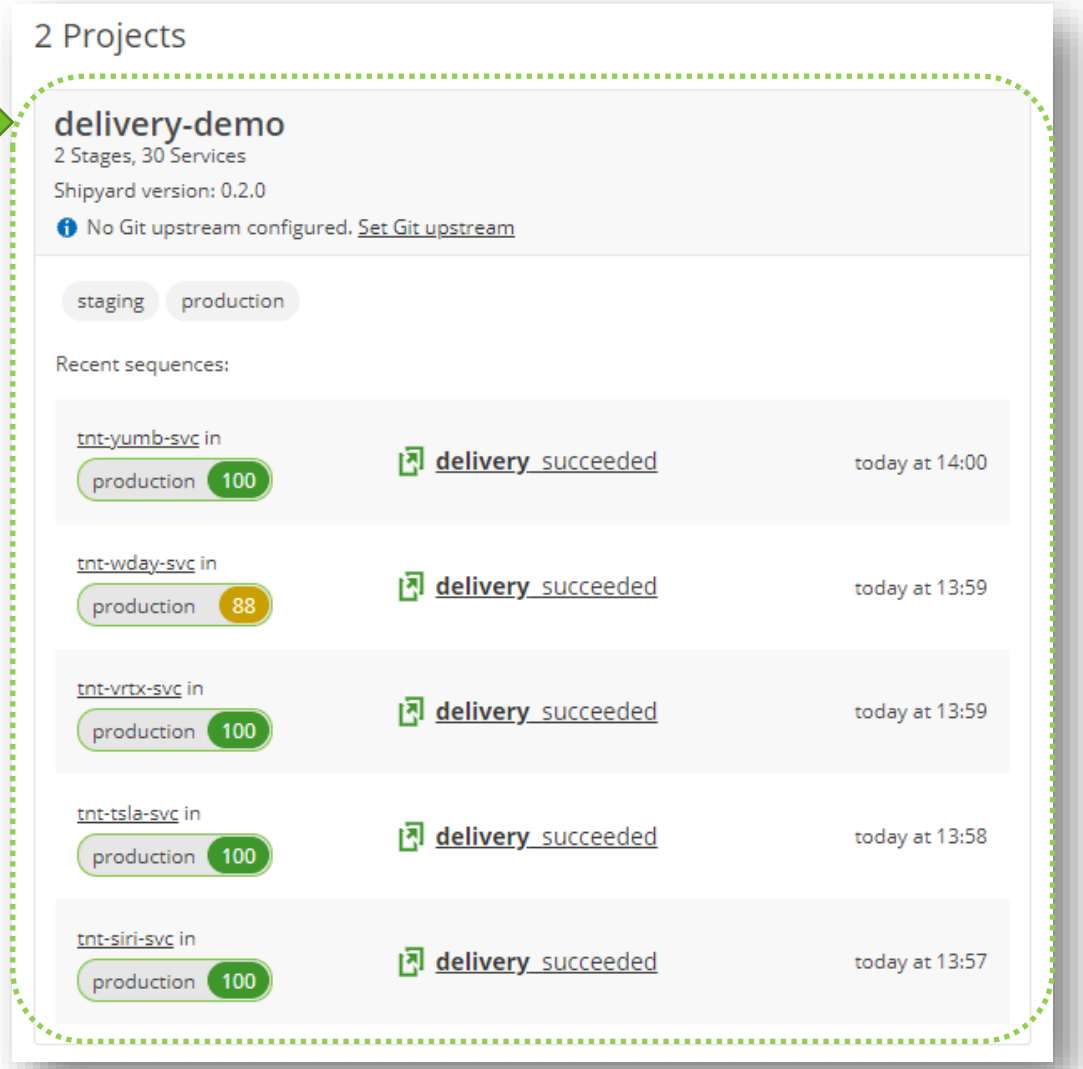


dynatrace

HANDS-ON SECTION

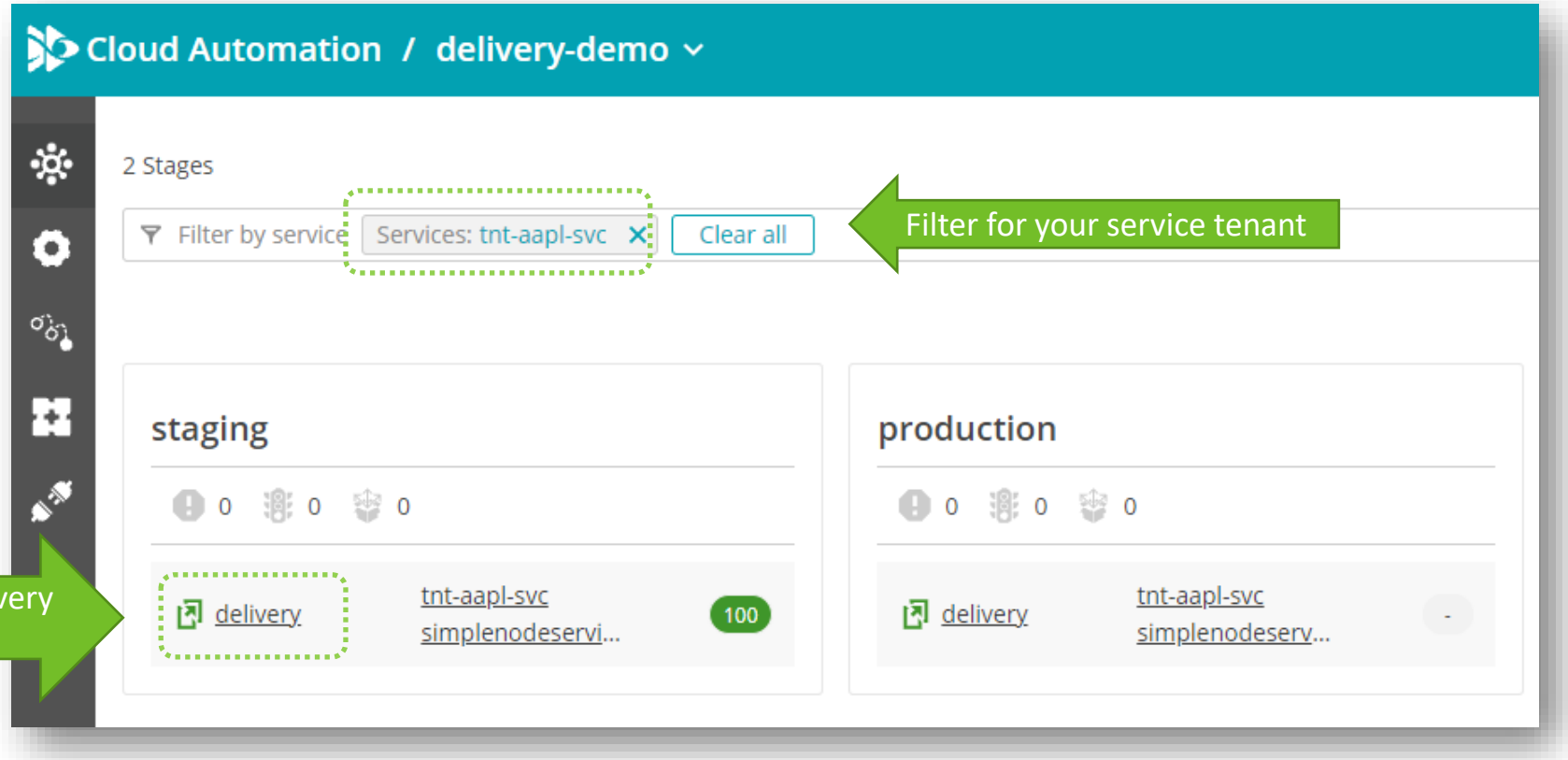
After you login to Cloud Automation you see the project „delivery-demo“

Delivery-demo project



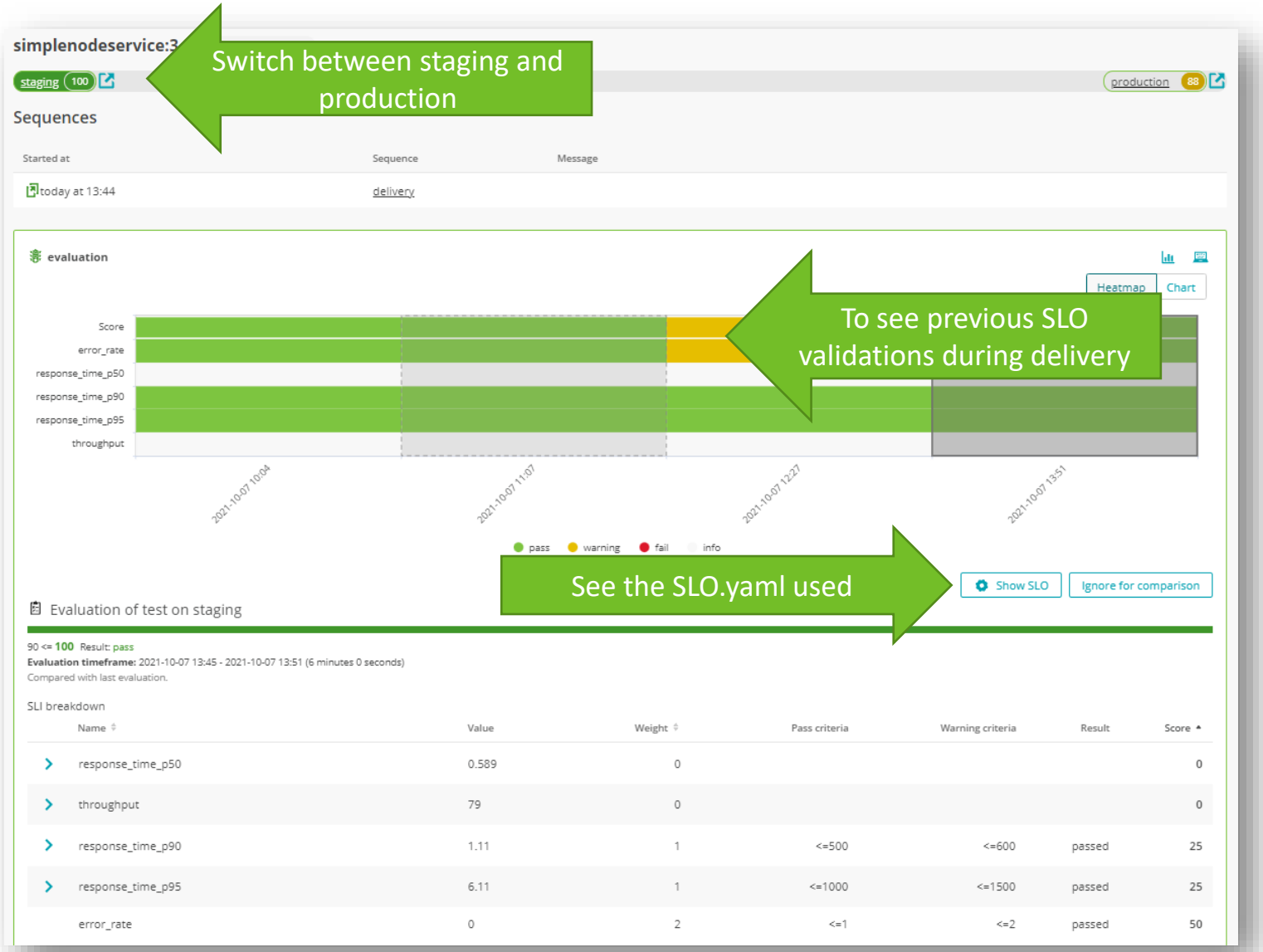
HANDS-ON SECTION

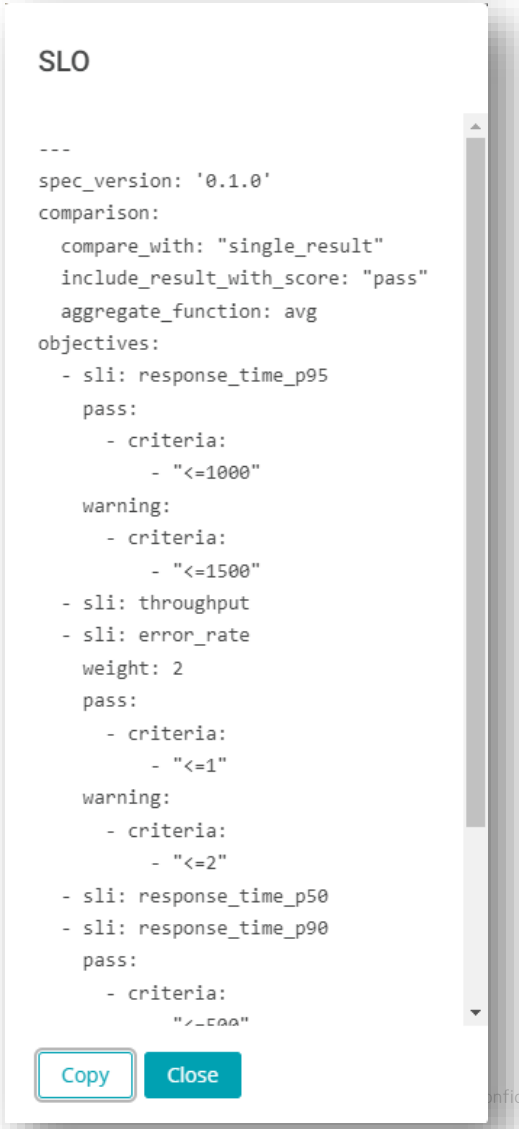
Explore your delivery automation for your tenant



HANDS-ON SECTION

Explore previous delivery automation runs

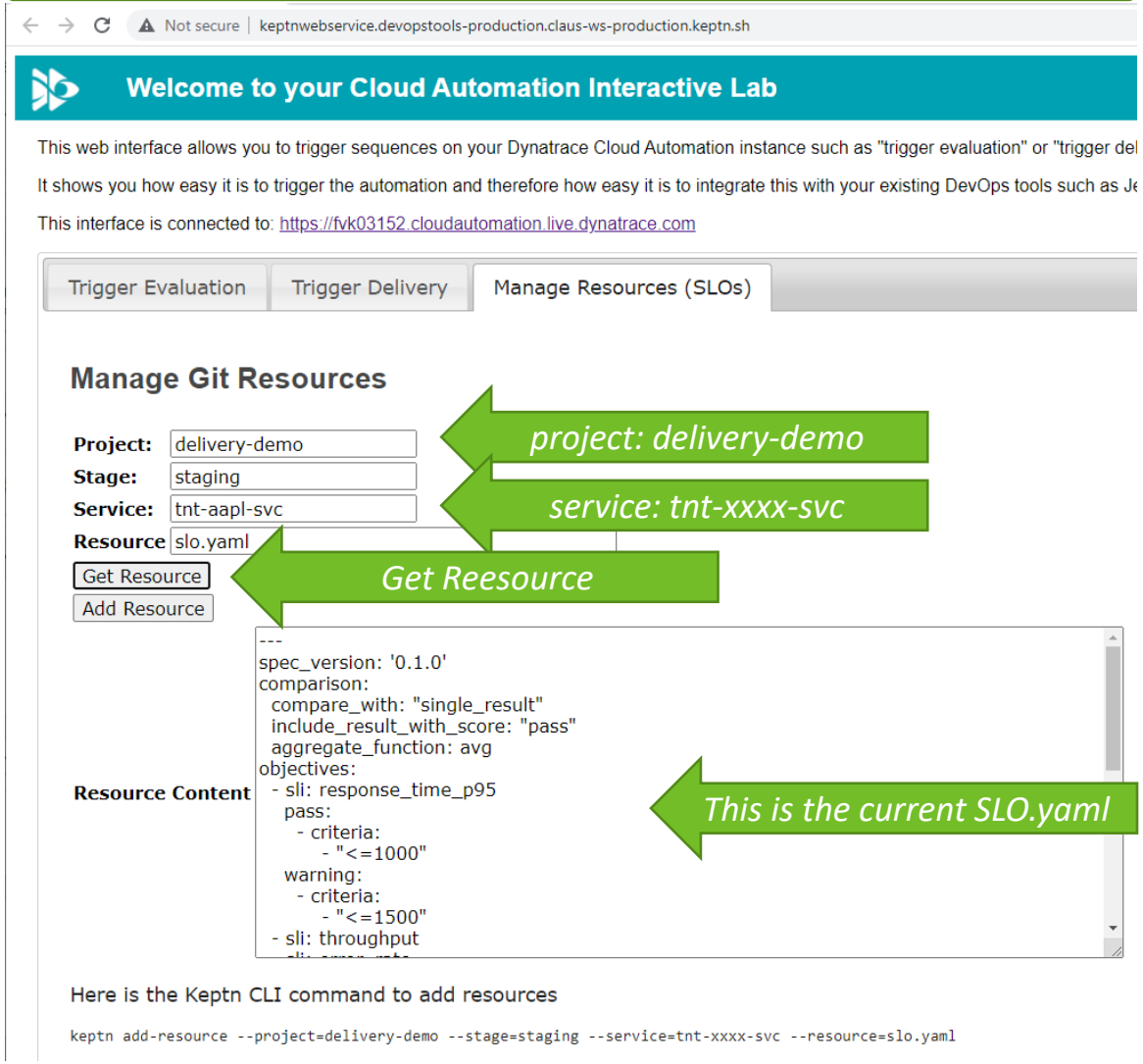




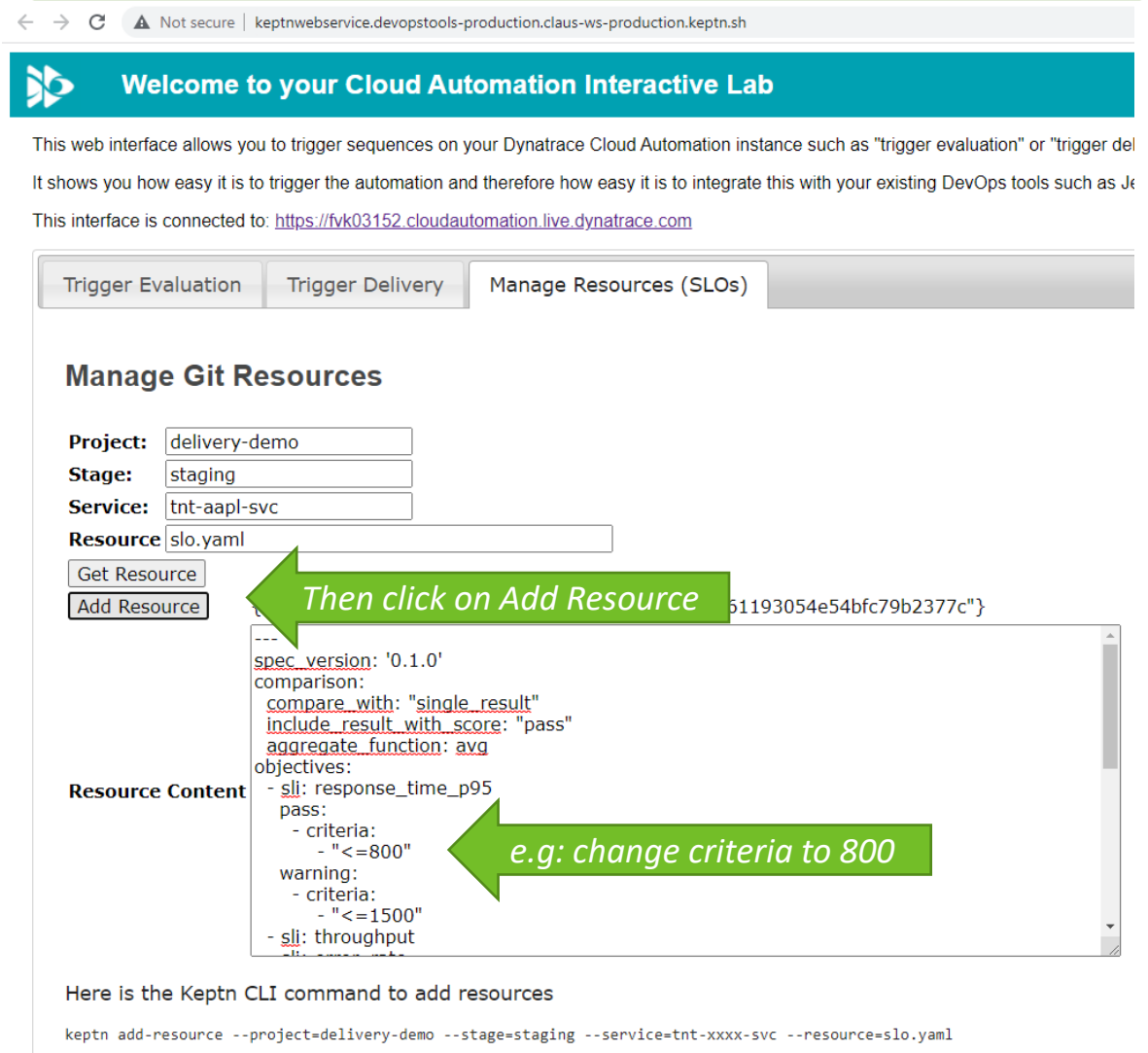
HANDS-ON SECTION

Lets change the SLO.yaml

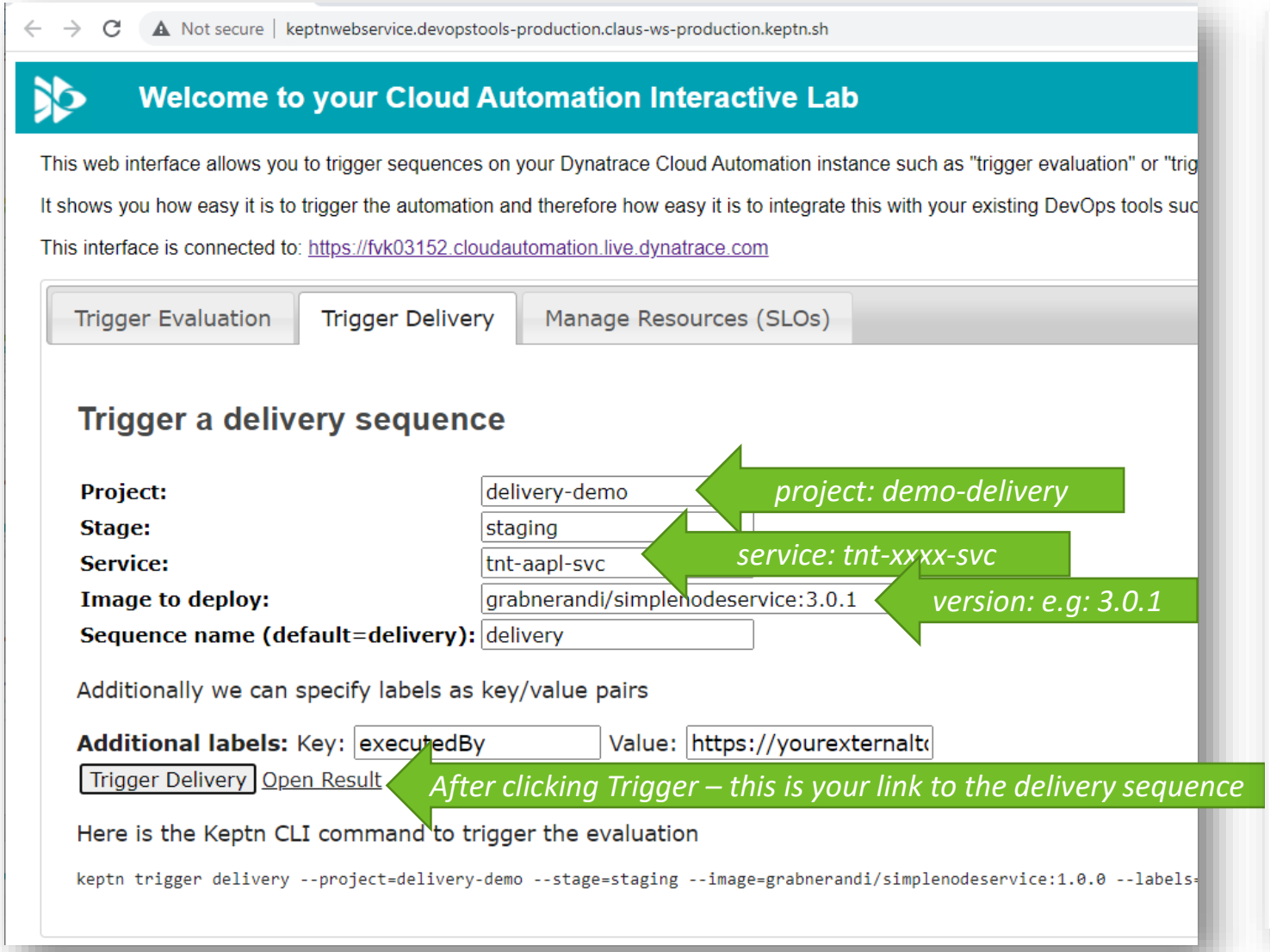
Step 1: Retrieve current SLO.yaml

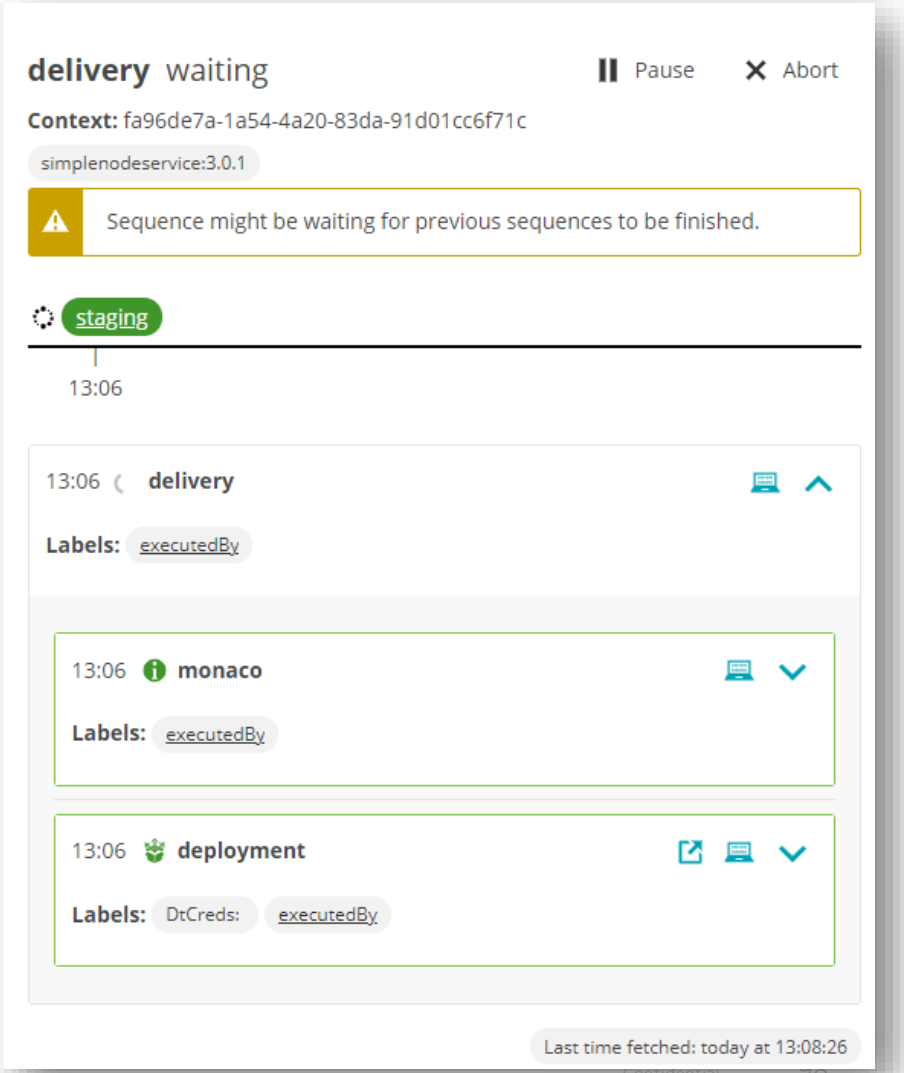


Step 2: Update SLO.yaml



Lets trigger a new deployment





WATCH TRAINER SECTION

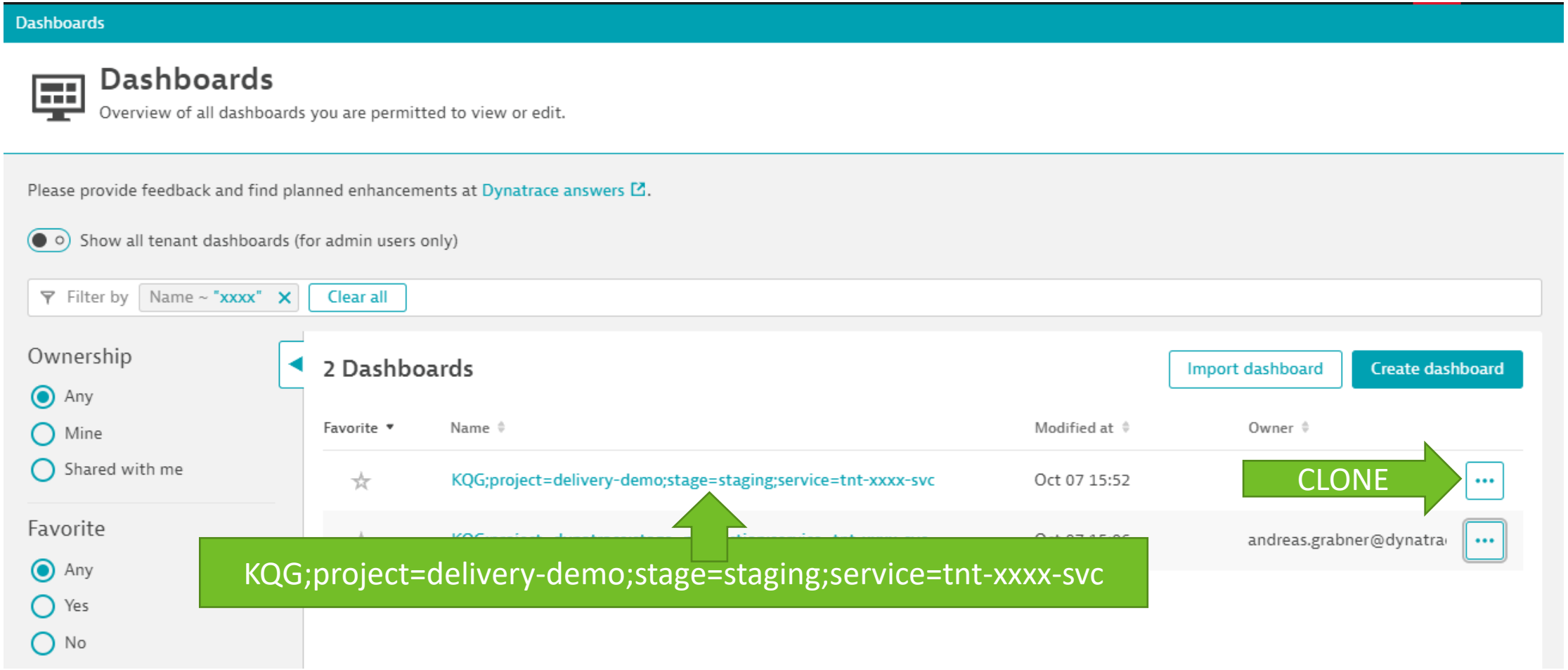
dynatrace
Performance Clinic
webinar series

Just as we did for release validation automation – we need a dashboard

- The automation will look for a Dynatrace dashboard with the following naming schema:
  - KQG;project=<PROJECT>;stage=<STAGE>;service=<SERVICE>
- In our case project=delivery-demo, stage=staging and service=tnt-xxxx-svc
- We could start from scratch, or ….. – start from a template ☺

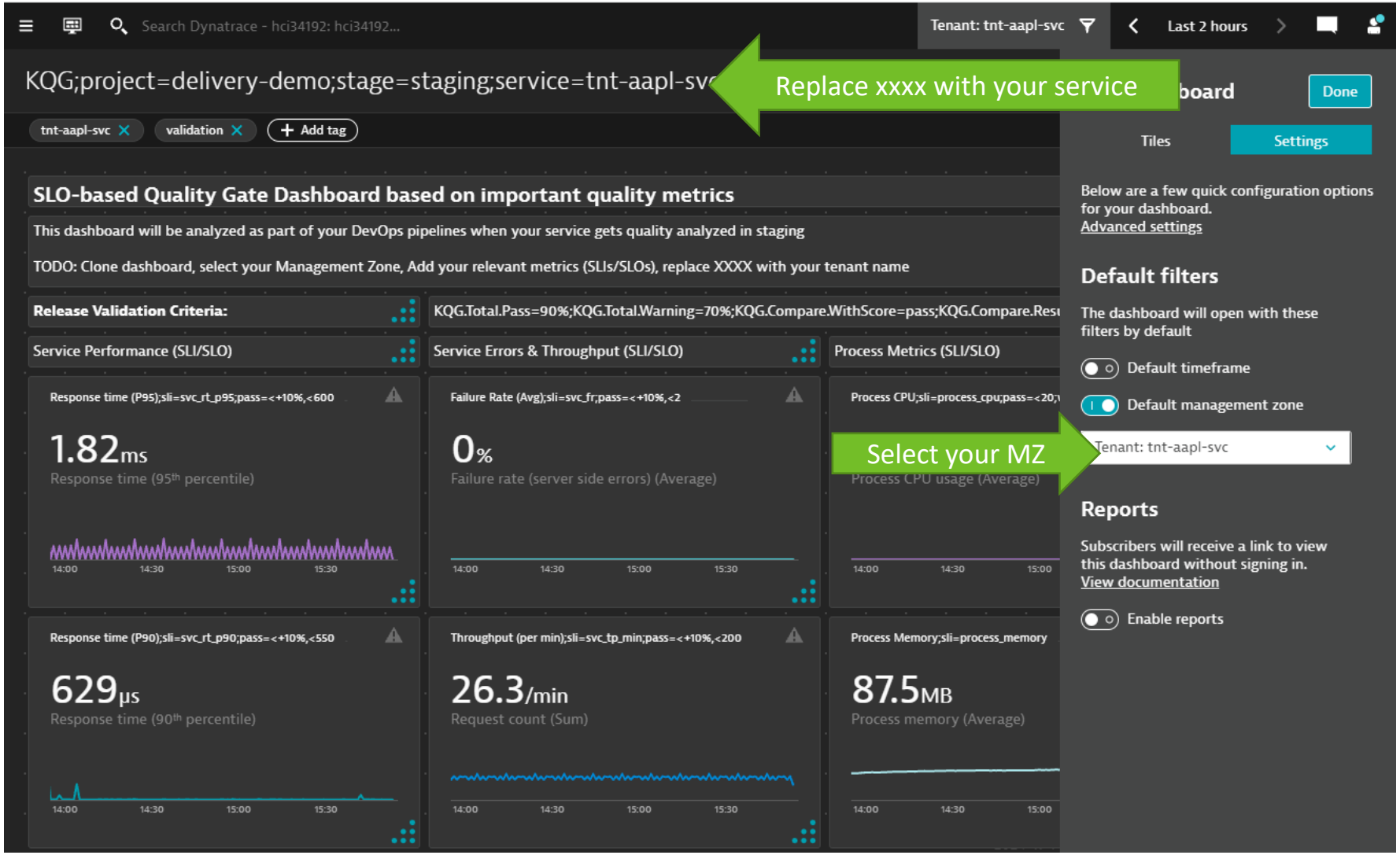
HANDS-ON SECTION

Step 1 – Clone the existing dashboard template



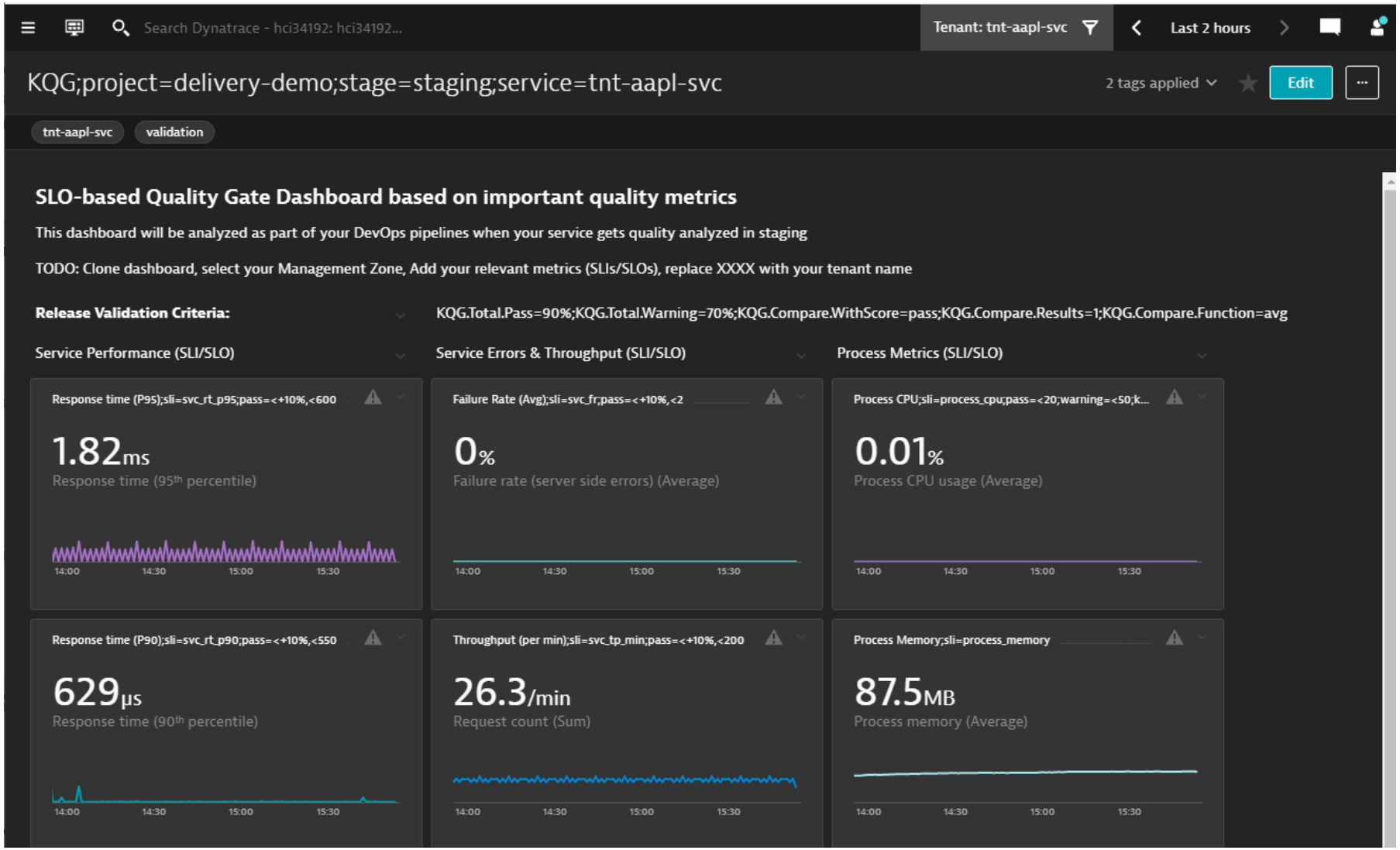
HANDS-ON SECTION

Step 2 – Give it proper name, change MZs and add your SLOs



HANDS-ON SECTION

Step 3 – Save the dashboard!



HANDS-ON SECTION

Triggering end-2-end delivery of version 2.0.1 of our sample app

action

service

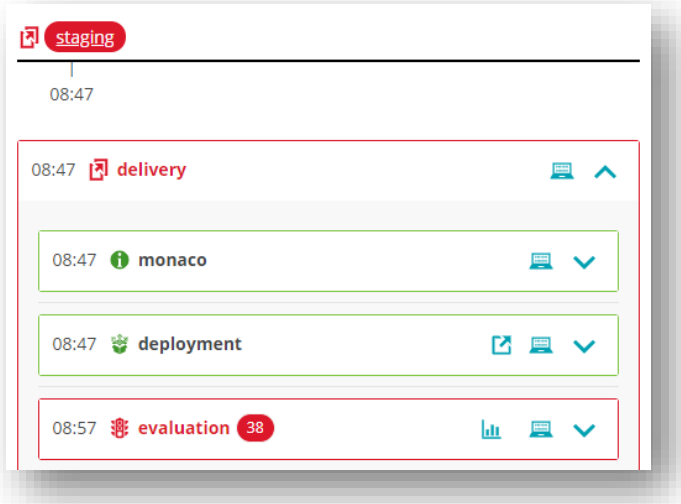
```
$ keptn trigger delivery --project=delivery-demo --service=tnt-xxxx-svc --image=grabnerandi/simplenodeservice:2.0.1
```

project

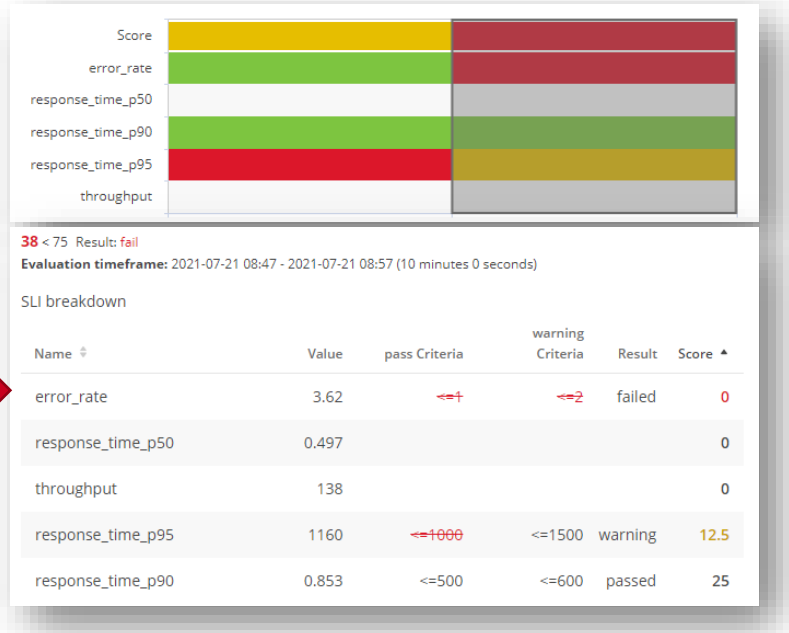
image

Other optional parameters:
--service defaults to staging
--labels is also possible
--values additional artifact metadata
--sequence if sequence other than delivery

Build #2 should fail due to high error rate and wont be promoted to production!



Failure Rate not meeting SLO of <2



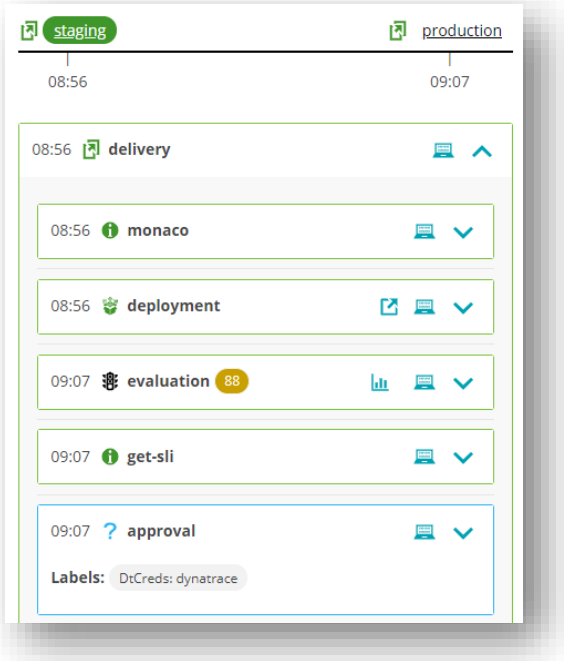
HANDS-ON SECTION

Triggering end-2-end delivery of version 3.0.1 of our sample app

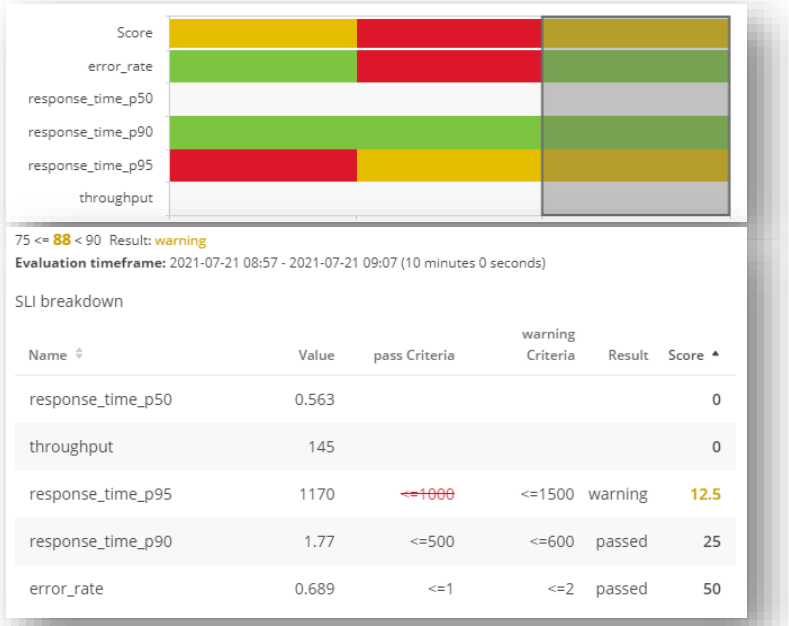
```
$ keptn trigger delivery --project=delivery-demo --service=tnt-xxxx-svc --image=grabnerandi/simplenodeservice:3.0.1
```

image

Build #3 should meet Quality Gate and gets promoted to production

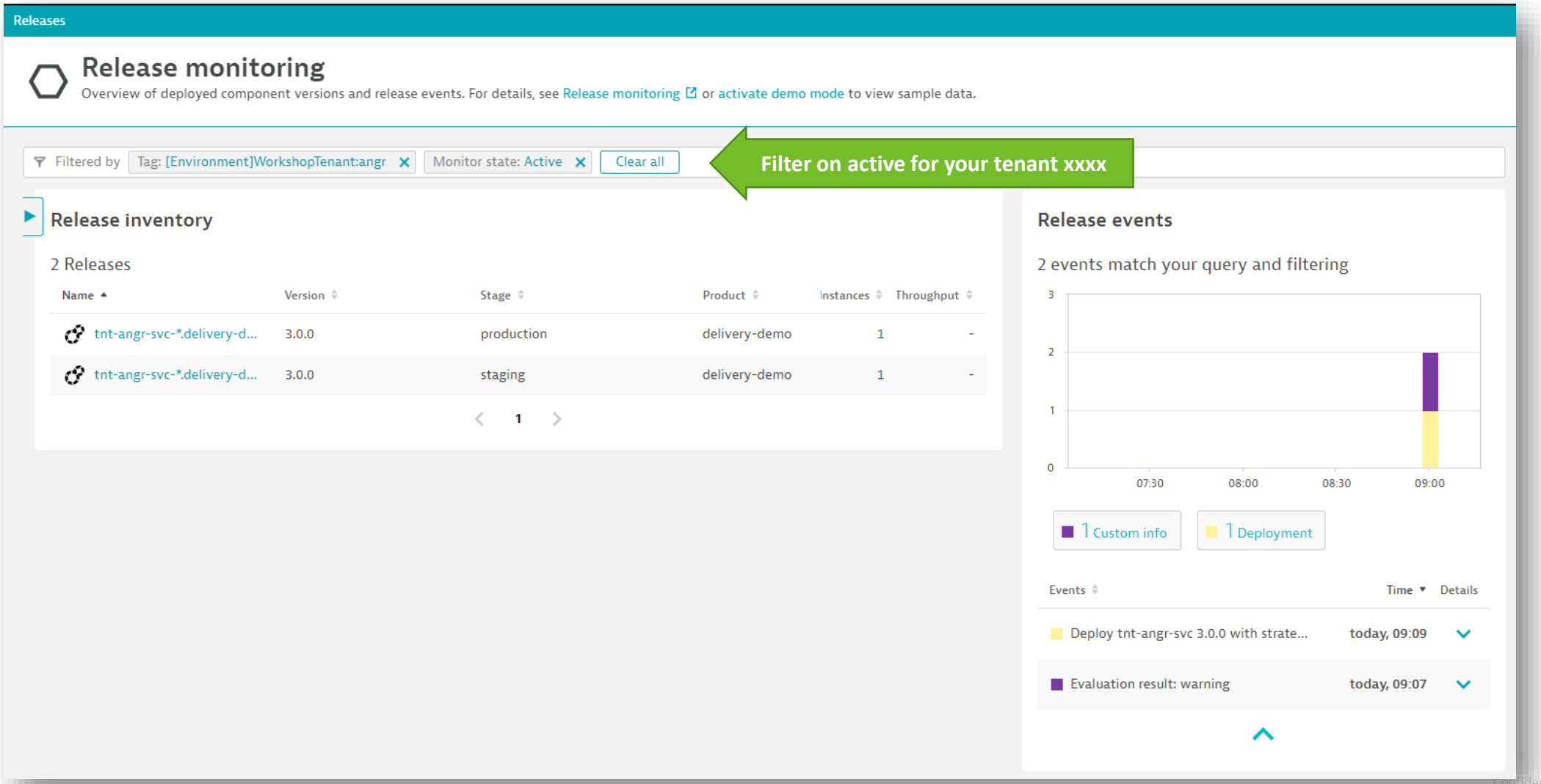


No major issues any more



HANDS-ON SECTION

Validate Deployment Events in Release Inventory



HANDS-ON SECTION

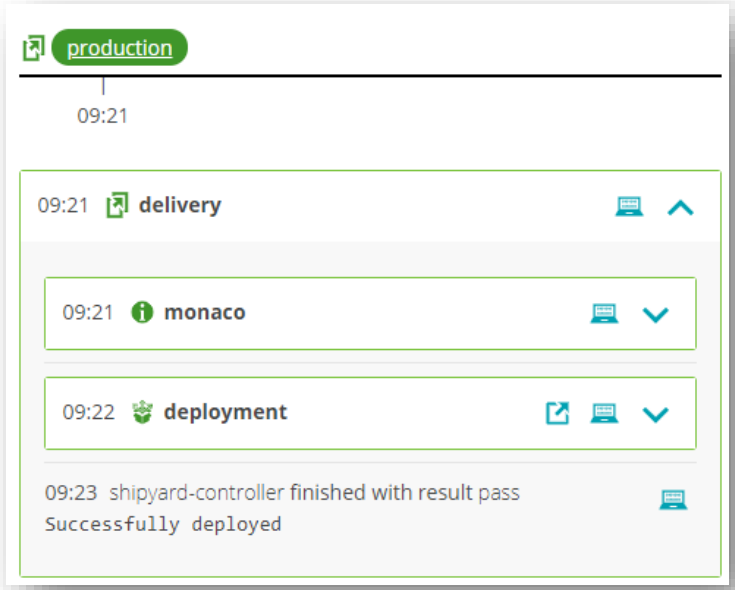
Triggering direct deployment of 4.0.1 into production

stage

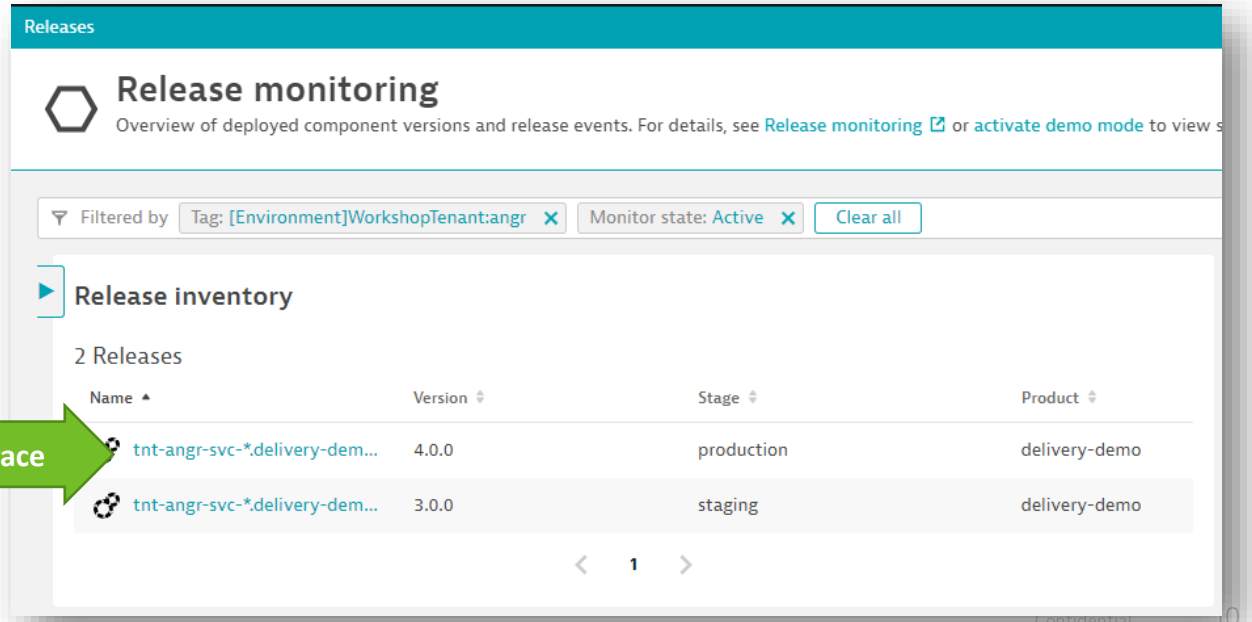
```
$ keptn trigger delivery --project=delivery-demo --stage=production --service=tnt-xxxx-svc --image=grabnerandi/simplenodeservice:4.0.1
```

image

Build #4 deploys straight into production



Also reflected in Dynatrace



Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file

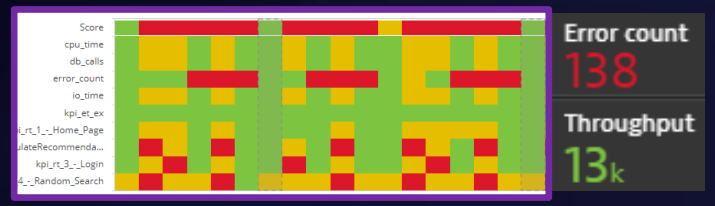
Delivery Pipelines		
Create / Clone your SLO-based Quality Gate Dashboard for staging	Trigger new deployments and validate dashboard is used	Validate results and events in Dynatrace release monitoring

Hands-On Lab: Wrap Up!
How Dynatrace helps DevOps & SRE

dynatrace

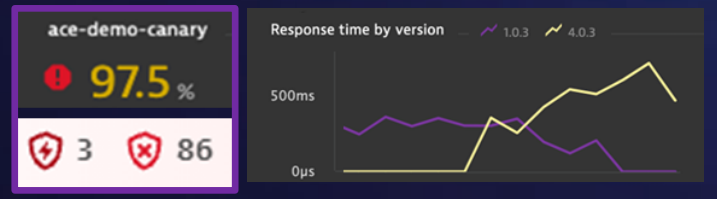
Dynatrace helps DevOps & SREs to Shift-Left SLOs

Delivery Pipelines



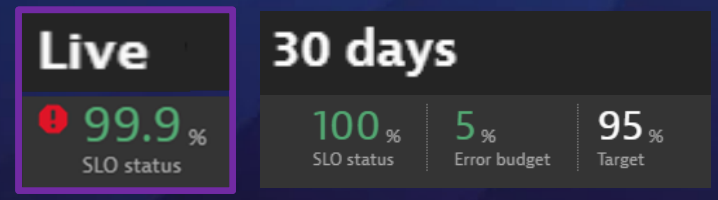
Speed up high-quality value creation

Release Validation



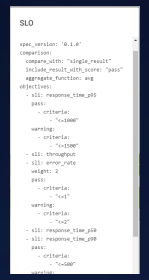
Eliminate Failed Releases

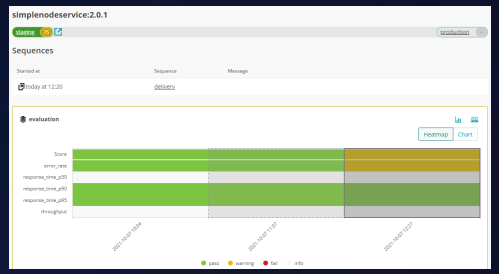
Production Reliability



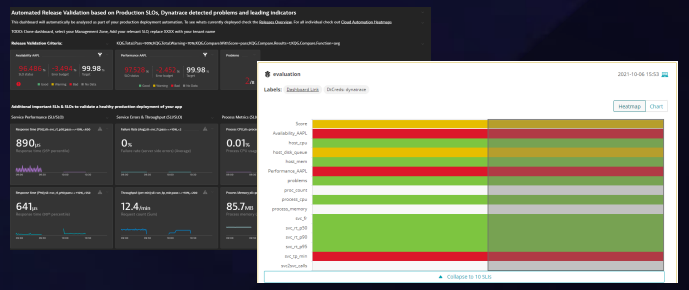
Ensure 100% Business Up-Time

Lab 3: SLO-based Quality Gates

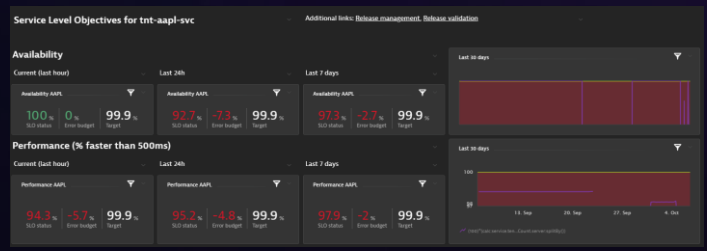




Lab 2: SLOs for Release Validation



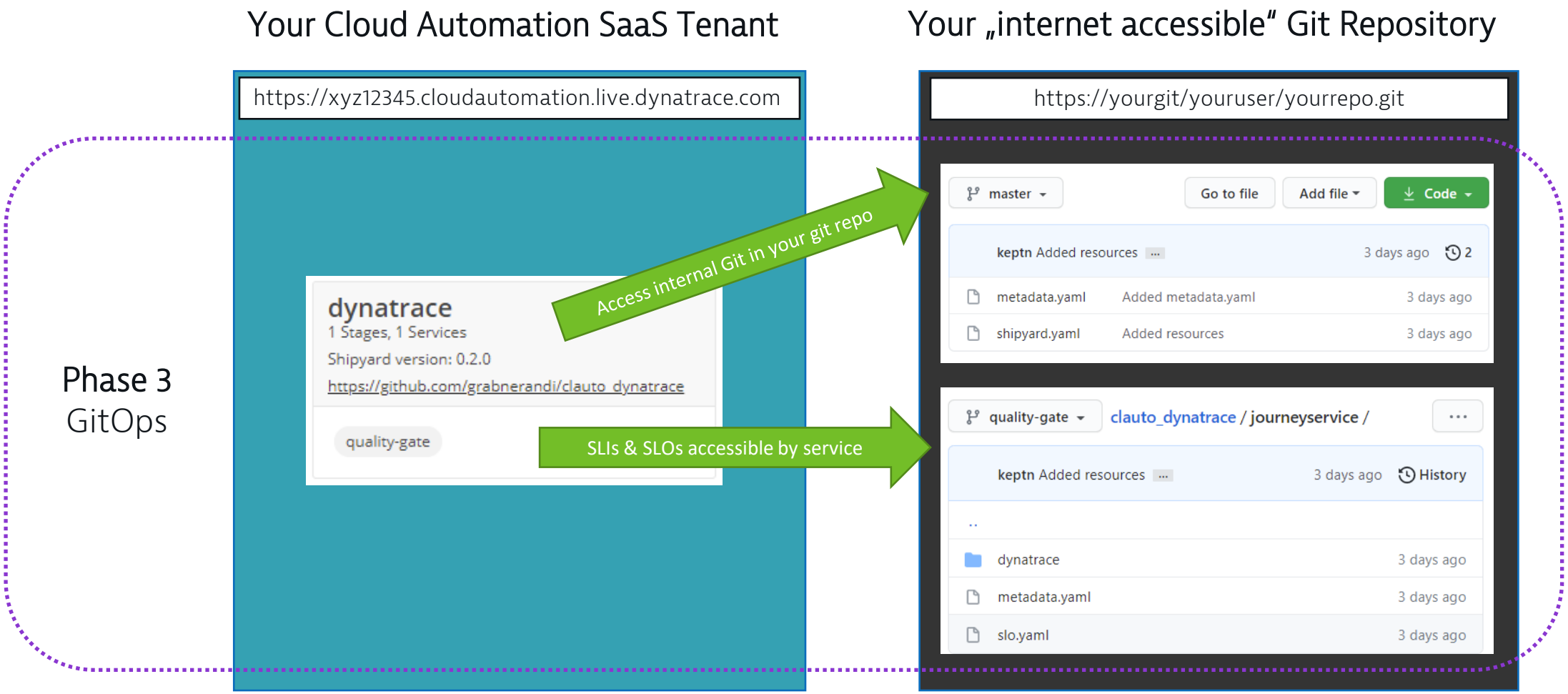
Lab 1: SLOs for Reporting



GitOps

Quality Gates as Code & Quality Gates as Dashboard

Here is what we are trying to achieve

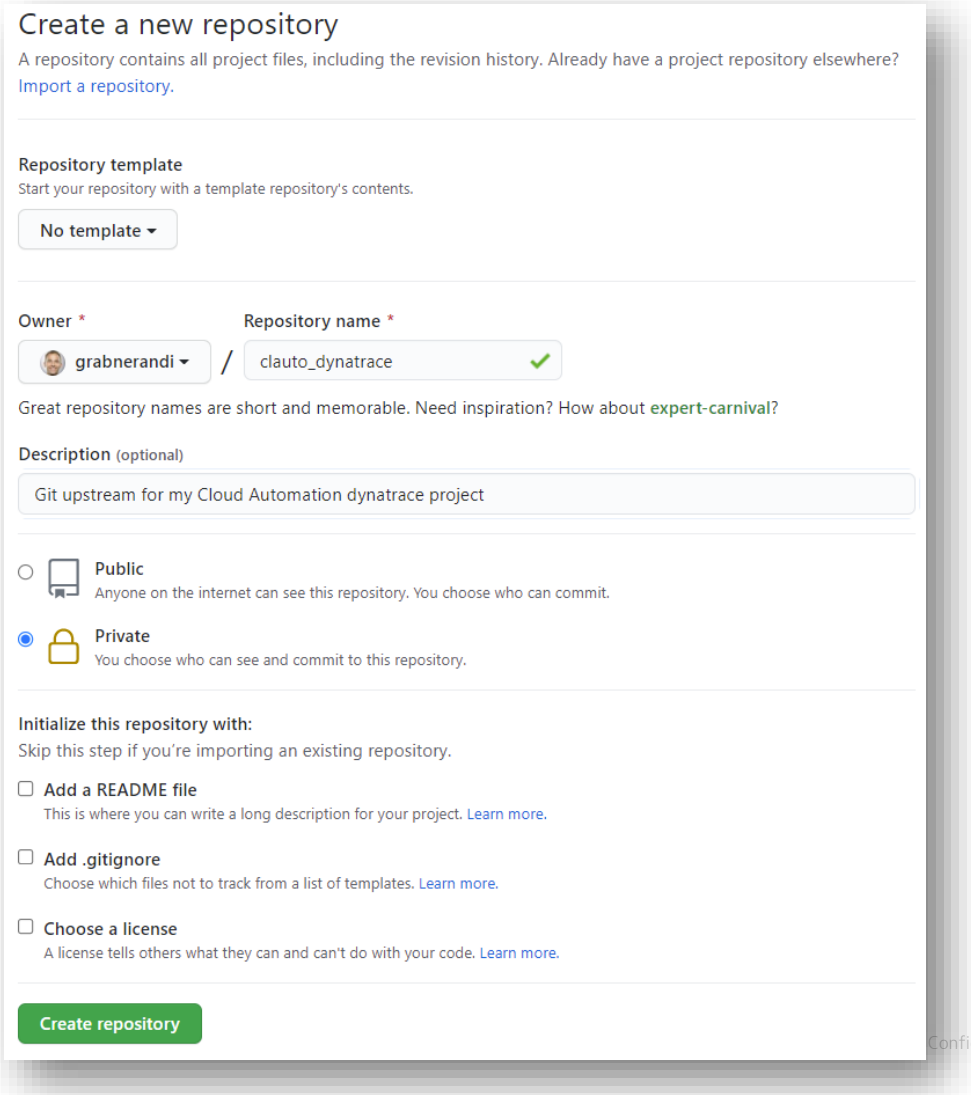


WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Phase #3 - Step 1: Create an empty git repository, e.g: GitHub, Bitbucket, Azure, GitLab ...

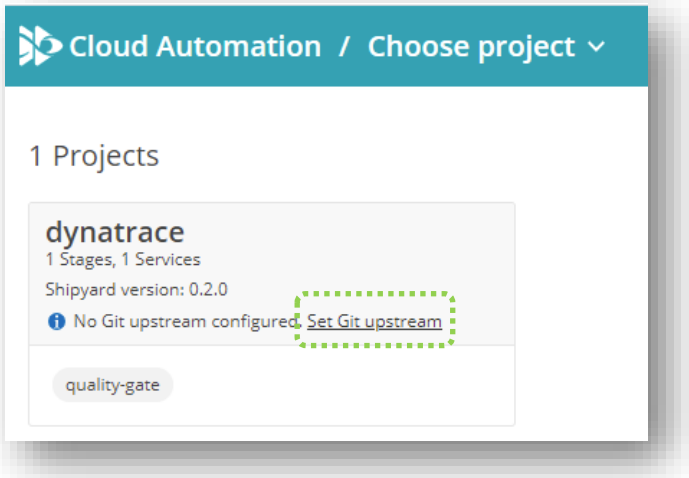
- Example: create an empty GitHub repository
  - More documentation for other git systems
- Besides a Git repo you also need
  - Git username
  - Git token with priviliges described here
- Pre-Req
  - Repository must be *accessible from the internet*
  - *Can be private*

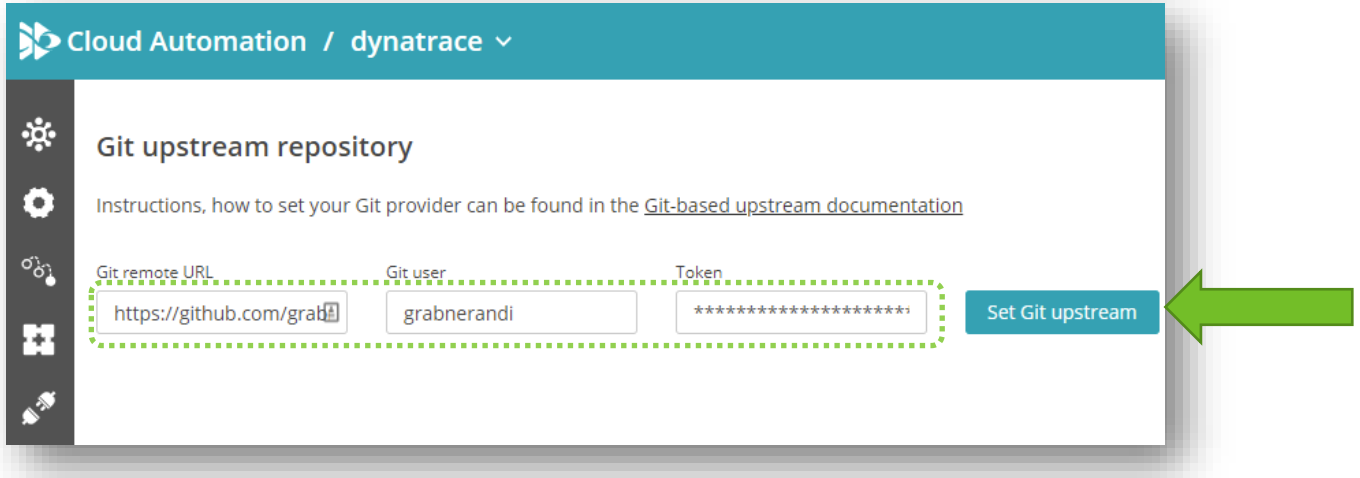


WATCH TRAINER SECTION

Phase #3 - Step 2: Set the git upstream credentials on your project

- Option A: Via the Cloud Automation UI

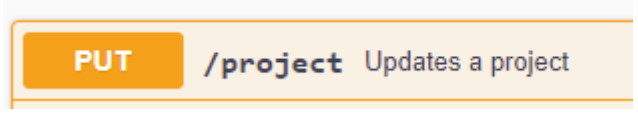




- Option B: Via the Keptn CLI

```
$ keptn update project dynatrace --git-user=YOUR_GIT_USER --git-token=YOUR_GIT_TOKEN --git-remote-url=YOUR_GIT_REMOTE_URL
Starting to update project
Project updated successfully
```

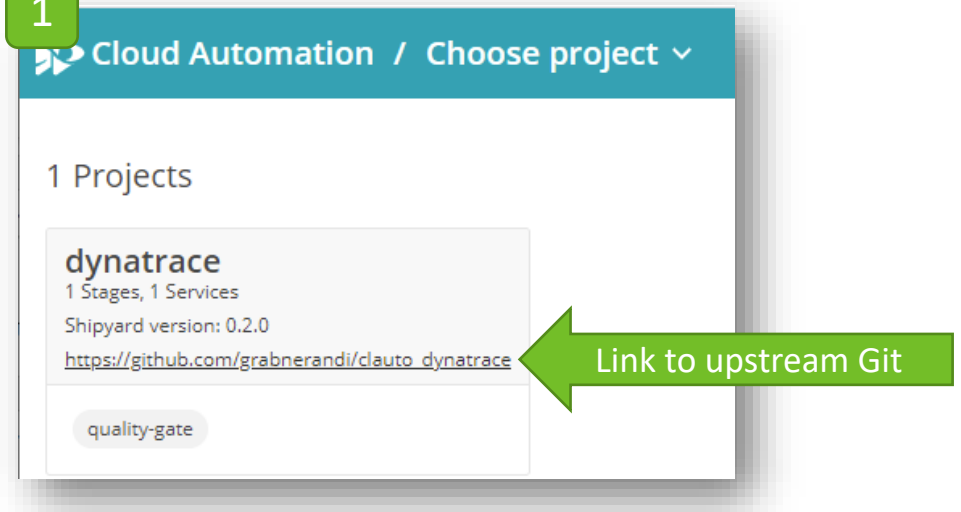
- Option C: Via the Keptn API



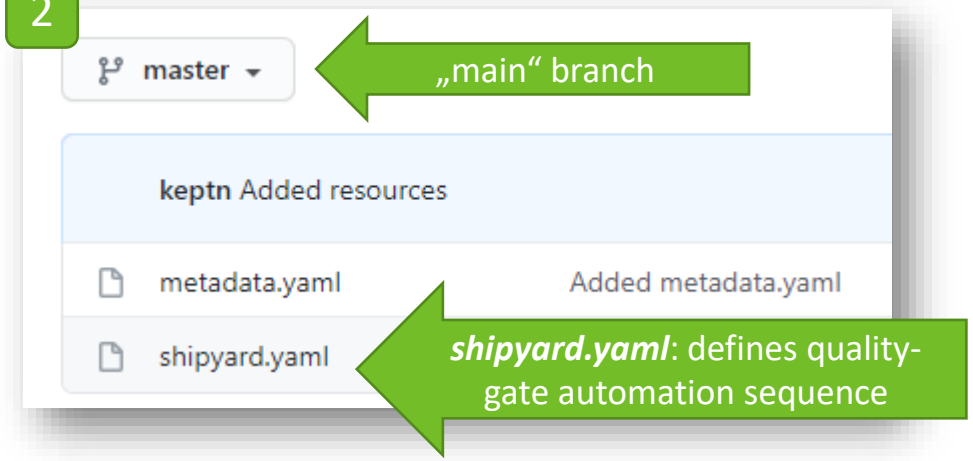
HANDS-ON SECTION

Phase #3 - Step 3: Explore dynatrace project upstream Git: shipyard, SLI, SLO …

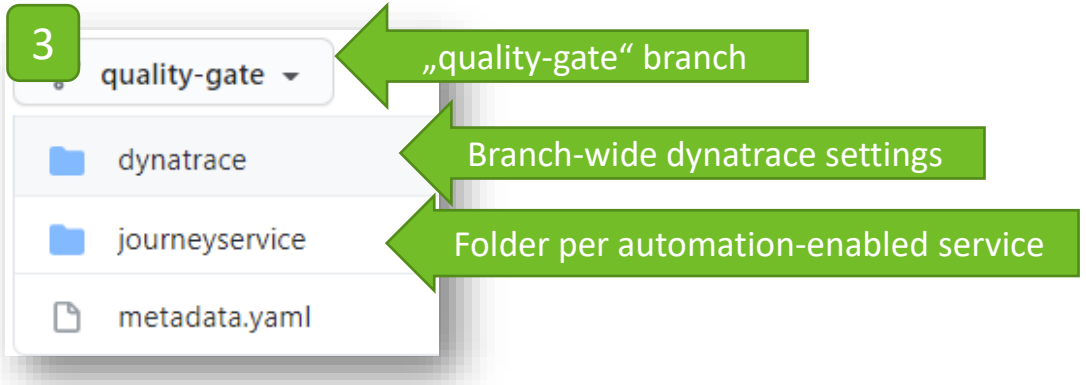
1



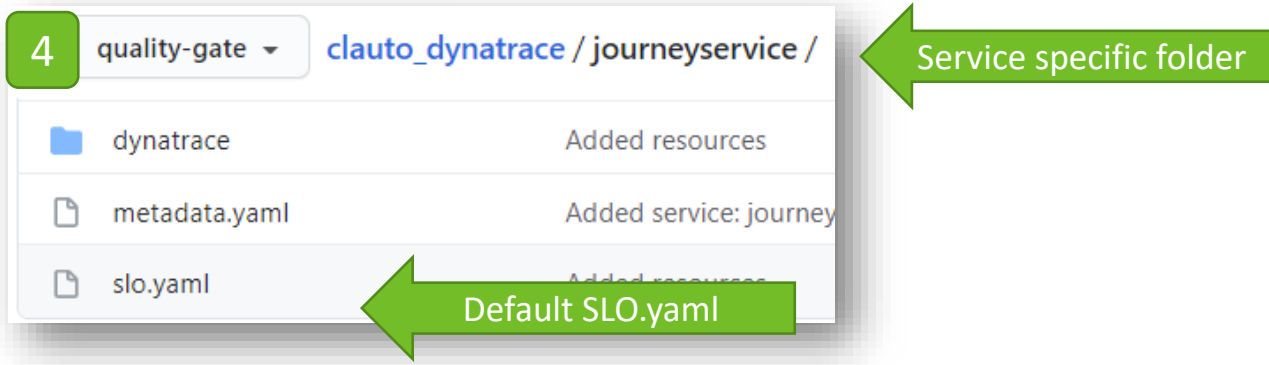
2



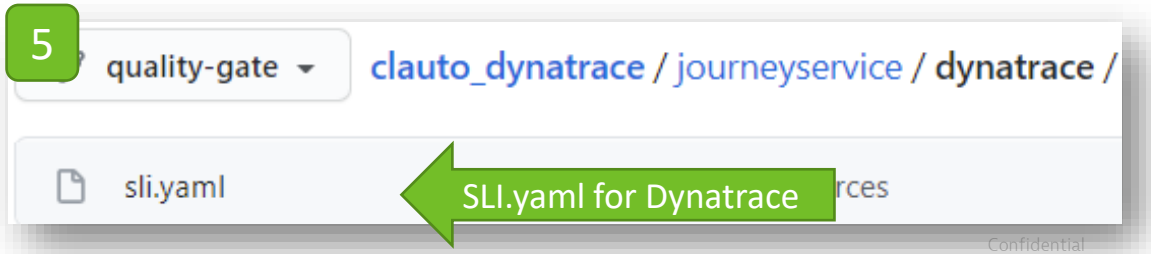
3



4



5



WATCH TRAINER SECTION

Phase #3 - Step #4: Customize your SLOs: add, remove or edit your SLOs

Branch: quality-gate — dynatrace / tnt-angr-svc / slo.yaml

27 lines | 520 B

```yaml
---
spec_version: "1.0"
comparison:
  aggregate_function: "avg"
  compare_with: "single_result"
  include_result_with_score: "pass"
  number_of_comparison_results: 1
filter:
objectives:
  - sli: "response_time_p95"
    key_sli: false
    pass:
      - criteria:
          - "<600"
    warning:
      - criteria:
          - "<=800"
    weight: 1
  - sli: "error_rate"
    key_sli: false
    pass:
      - criteria:
          - "<5"
  - sli: throughput
total_score:
  pass: "90%"
  warning: "75%"
```

Update SLO.yaml

dynatrace / tnt-angr-svc / slo.yaml — or Cancel

Edit File | Preview Changes

```yaml
---
spec_version: "1.0"
comparison:
  aggregate_function: "avg"
  compare_with: "single_result"
  include_result_with_score: "pass"
  number_of_comparison_results: 1
filter:
objectives:
  - sli: "response_time_p95"
    key_sli: false
    pass:
      - criteria:
          - "<500"
    warning:
      - criteria:
          - "<=1000"
    weight: 2
  - sli: "error_rate"
    key_sli: true
    pass:
      - criteria:
          - "<5"
  - sli: throughput
    pass:
      - criteria:
          - ">10"
  - sli: response_time_p50
  - sli: response_time_p90
total_score:
  pass: "90%"
  warning: "50%"
```

e.g: change thresholds for rt_p95

e.g: double the weight of this sli

e.g: Make „error_rate“ a „key_sli

e.g: Add criteria to throughput

e.g: Add additional SLOs

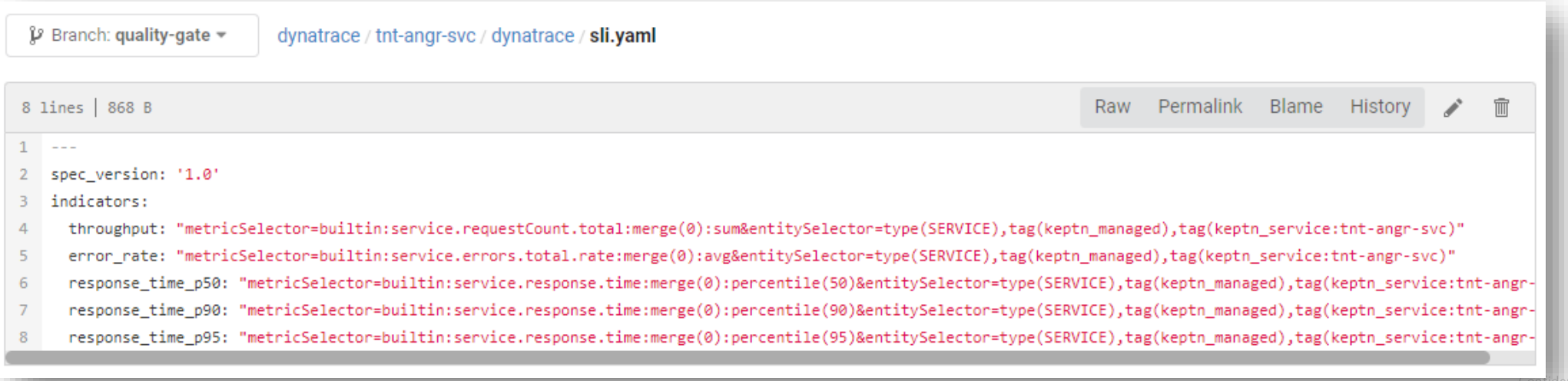
e.g: Change total score criteria

WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Phase #3 - Step #5: Customize the SLI.yaml: update or add new SLI definitions

- Add additional SLI definitions by providing an sli name and the Dynatrace Metrics API v2 Query
  - https://www.dynatrace.com/support/help/dynatrace-api/environment-api/metric-v2/get-data-points/
- You can also use the following PLACEHOLDERS as part of your query
  - $PROJECT: Cloud Automation projectname, e.g: dynatrace
  - $STAGE: Cloud Automation stage, e.g: quality-gate
  - $SERVICE: Cloud Automation service, e.g: journeyservice

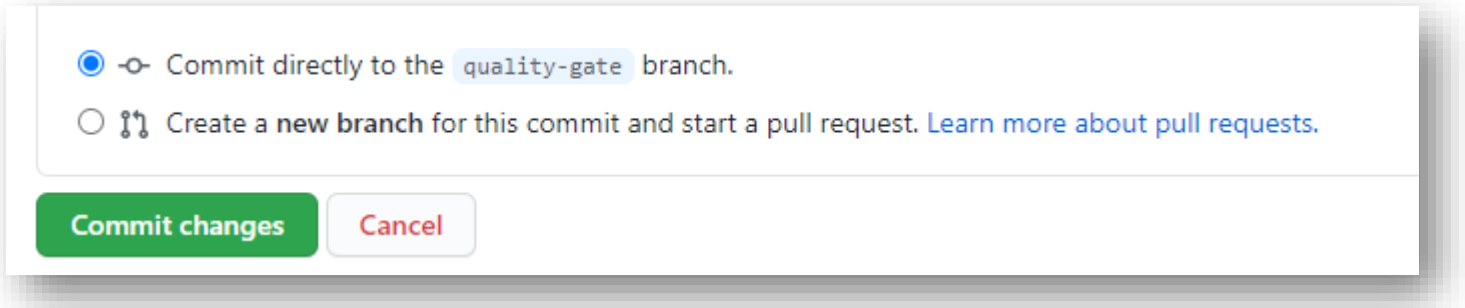


WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Phase #3 - Step #6: Update SLI & SLO yamls

- You can either edit sli.yaml and slo.yaml in the git repository



- Or upload / overwrite a new version of sli.yaml & slo.yaml via *keptn add-resource*
  - Use slo.yaml for resourceUri
  - Use dynatrace/sli.yaml for resourceURI

```
$ keptn add-resource --project=dynatrace --stage=quality-gate --service=journeyservice --resource=mylocalslo.yaml --resourceUri=slo.yaml
```

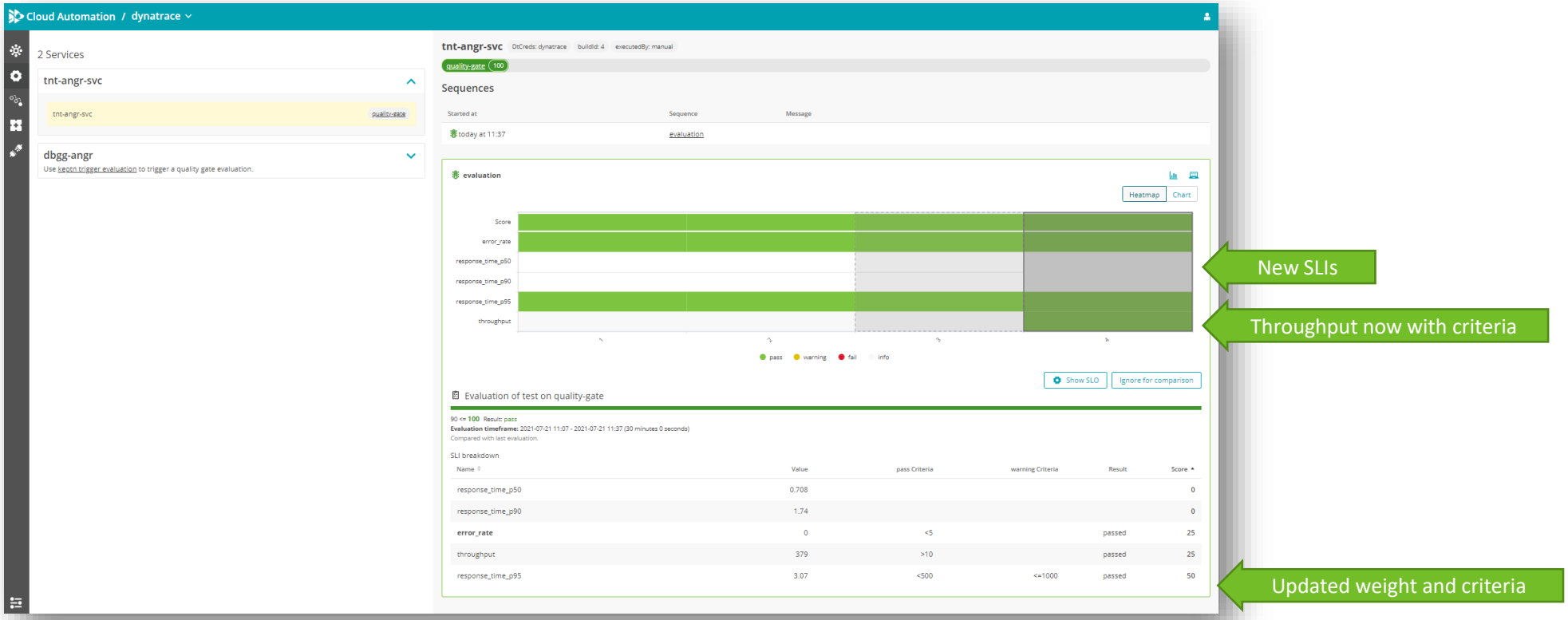
```
$ keptn add-resource --project=dynatrace --stage=quality-gate --service=journeyservice --resource=mylocalsli.yaml --resourceUri=dynatrace/sli.yaml
```

WATCH TRAINER SECTION

Phase #3 - Step #7: Run a new quality gate with the changed settings

```
$ keptn trigger evaluation --project=dynatrace --stage=quality-gate --service=tnt-xxxx-svc --timeframe=30m --labels=buildId=4,executedBy=manual
```

meta-data, e.g: buildId=4



New SLIs

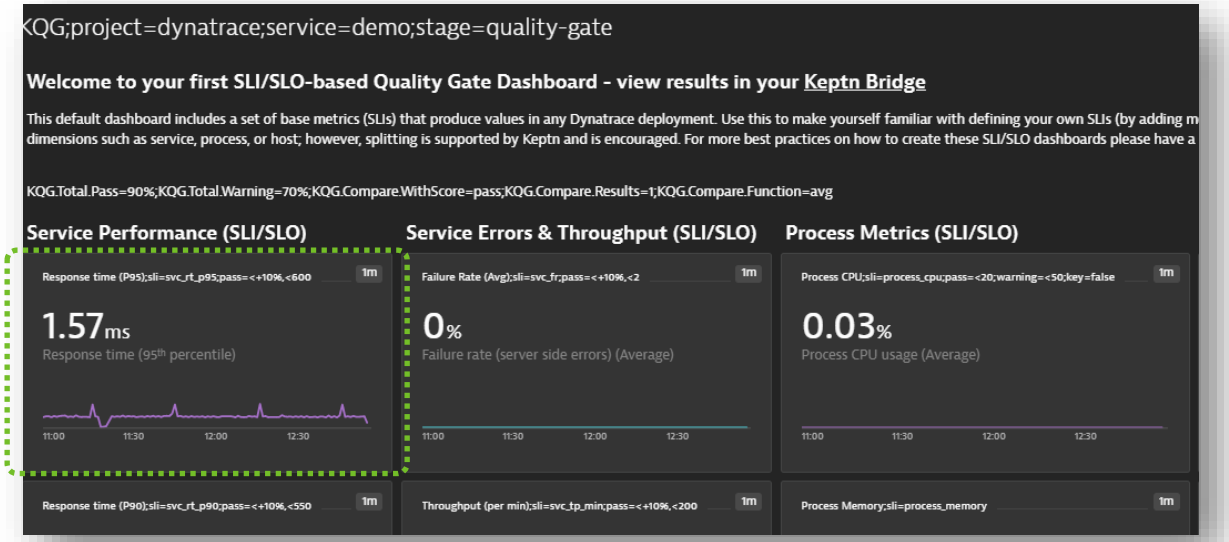
Throughput now with criteria

Updated weight and criteria

WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Behind the scenes on dashboards: What gets generated?



Branch: quality-gate — dynatrace / demo / dynatrace / sli.yaml

15 lines | 1.5 KiB — Raw Permalink Blame History

```yaml
spec_version: 0.1.4
indicators:
  host_cpu: MV2;Percent;metricSelector=builtin:host.cpu.usage:merge(0):avg:names&entitySelector=type(HOST)
  host_disk_queue: MV2;Count;metricSelector=builtin:host.disk.queueLength:merge(1):merge(0):max:names&entitySelector=type(HOST)
  host_mem: MV2;Percent;metricSelector=builtin:host.mem.usage:merge(0):avg:names&entitySelector=type(HOST)
  proc_count: MV2;Count;metricSelector=builtin:tech.generic.processCount:merge(0):avg:names&entitySelector=type(PROCESS_GROUP_INSTANCE)
  process_cpu: MV2;Percent;metricSelector=builtin:tech.generic.cpu.usage:merge(0):avg:names&entitySelector=type(PROCESS_GROUP_INSTANCE)
  process_memory: MV2;Byte;metricSelector=builtin:tech.generic.mem.workingSetSize:merge(0):avg:names&entitySelector=type(PROCESS_GROUP_INSTANCE)
  svc_fr: MV2;Percent;metricSelector=builtin:service.errors.server.rate:merge(0):avg:names&entitySelector=type(SERVICE)
  svc_rt_p50: MV2;MicroSecond;metricSelector=builtin:service.response.time:merge(0):percentile(50.000000):names&entitySelector=type(SERVICE)
  svc_rt_p90: MV2;MicroSecond;metricSelector=builtin:service.response.time:merge(0):percentile(90.000000):names&entitySelector=type(SERVICE)
  svc_rt_p95: MV2;MicroSecond;metricSelector=builtin:service.response.time:merge(0):percentile(95.000000):names&entitySelector=type(SERVICE)
  svc_tp_min: MV2;Count;metricSelector=builtin:service.requestCount.total:merge(0):value:names&entitySelector=type(SERVICE)
  svc2svc_calls: MV2;Count;metricSelector=builtin:service.nonDbChildCallCount:merge(0):value:names&entitySelector=type(SERVICE)
```

Dynatrace Metrics Query based on chart definition

SLO definition based on chart title

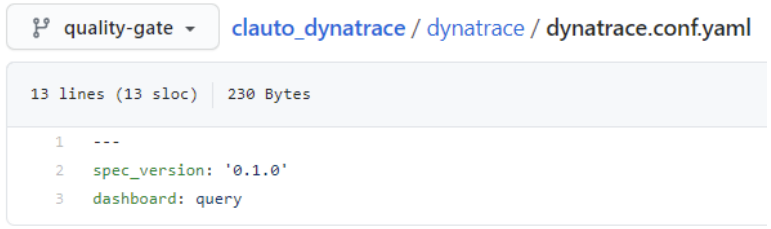
Branch: quality-gate — dynatrace / demo / slo.yaml

115 lines | 1.6 KiB

```yaml
spec_version: ""
filter: {}
comparison:
  compare_with: single_result
  include_result_with_score: pass
  number_of_comparison_results: 1
  aggregate_function: avg
objectives:
- sli: proc_count
  displayName: ""
  pass: []
  warning: []
  weight: 1
  key_sli: false
- sli: svc_rt_p95
  displayName: ""
  pass:
  - criteria:
    - <+10%
    - <600
  warning: []
  weight: 1
  key_sli: false
- sli: svc_rt_p90
  displayName: ""
  pass:
  - criteria:
    - <+10%
    - <550
  warning: []
  weight: 1
  key_sli: false
- sli: svc_rt_p50
```

Behind the scenes: dynatrace.conf.yaml enables dashboard parsing

- Its ***enabled by default*** for the *quality-gate* stage in the *dynatrace* project



- Read all details on github
  - https://github.com/keptn-contrib/dynatrace-sli-service#configurations-of-dashboard-slislo-queries-through-dynatraceconfyaml
- Essentially you need to upload a dynatrace.conf.yaml on project, stage or service level
  - Either add or edit that file in your git upstream repository

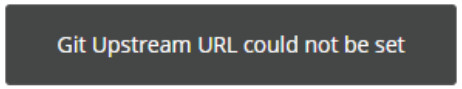
  - Our create that file and upload it through the keptn CLI like this

```
$ keptn add-resource --project=dynatrace --stage=quality-gate --service=yourservice --resource=dynatrace.conf.yaml --
resourceUri=dynatrace/dynatrace.conf.yaml
```

WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Phase #3 - Troubleshooting if you receive



- Validate your Git repository URL is internet accessible
  - Cloud Automation SaaS currently runs in the cloud and therefore needs access to that URL
  - You can validate by pinging the URL from a machine outside your company network
- Validate your Git username is valid
  - It should be the same username that shows up when logged in to your github, gitlab, azure …
- Validate your Git token has the right priviliges
  - See the required priviliges in the documentation
- Improvement request for easier troubleshooting in UI
  - https://github.com/keptn/keptn/issues/4626

Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file

Codify Quality Gates		
Understand how to set Git Upstream for a Cloud Automationproject	Customize SLI.yaml & SLO.yaml via Git Interface	Execute Quality Gate with modified SLI / SLO.yaml

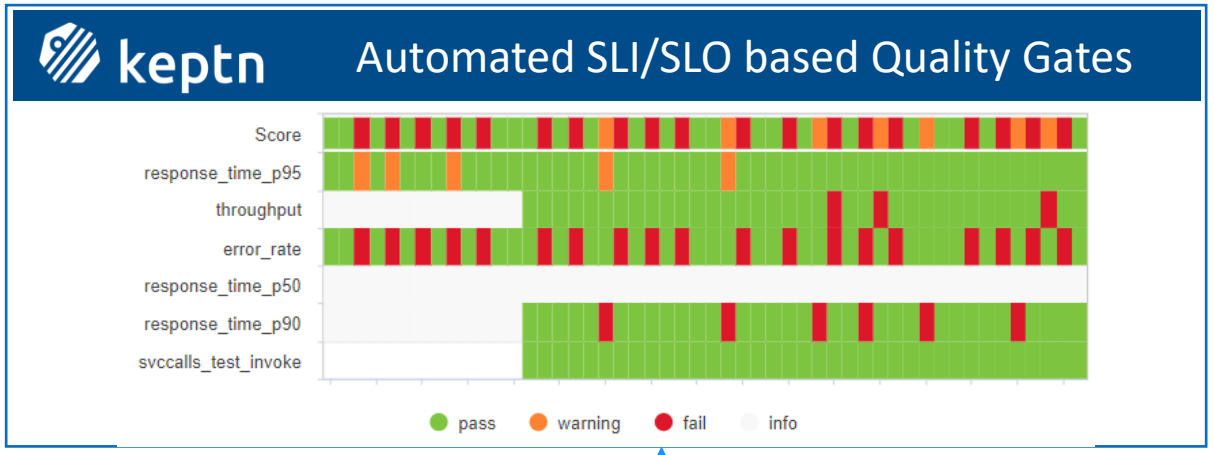
GitOps

Integrate into your existing CI/CD DevOps Processes & Tools

Example: Quality Gates accelerates GitLab Pipelines lead time by 80%

Trigger Evaluation

87.5%: passed



Pull SLI Metrics

dynatrace

Christian Heckelmann
Senior Systems Engineer

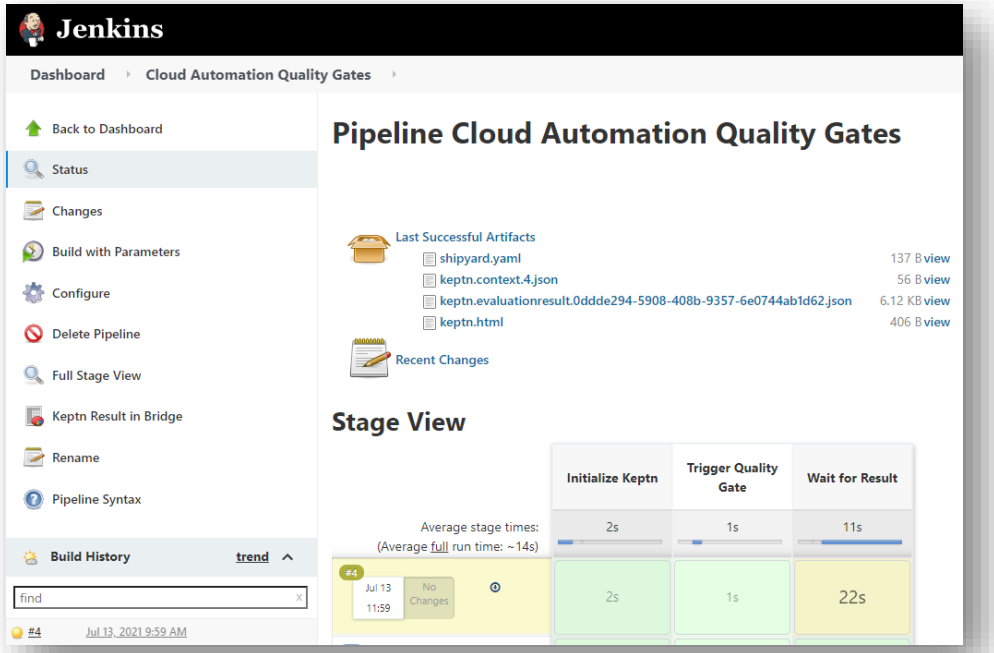
ERT
Confidence at every turn™

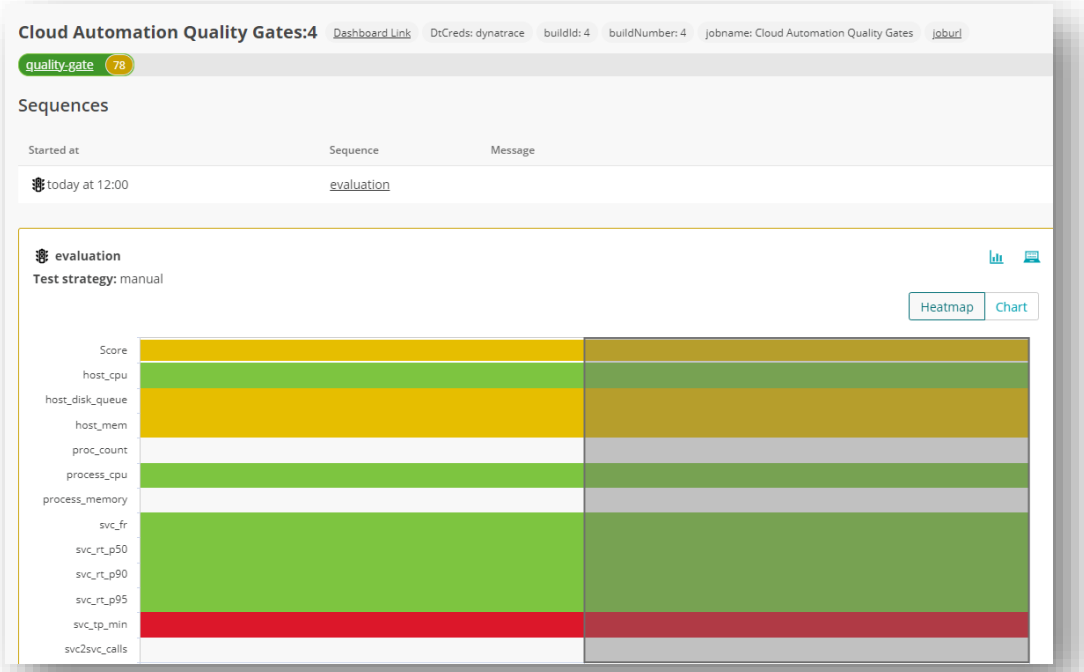
WATCH TRAINER SECTION

dynatrace Performance Clinic webinar series

Phase #5 - Step 1: Trigger Quality Gate from your CI/CD, e.g: Jenkins, GitLab, Azure DevOps

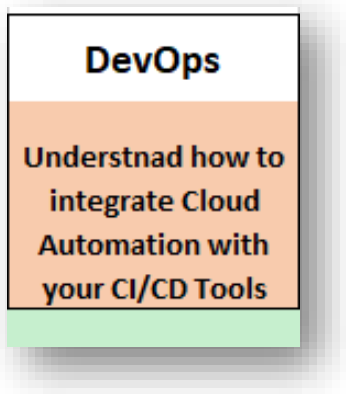
- Either leverage the existing libraries & extensions for
  - Jenkins: https://github.com/keptn-sandbox/keptn-jenkins-library/
  - Azure DevOps: https://github.com/keptn-sandbox/keptn-azure-devops-extension
  - GitLab: https://gitlab.com/checkelmann/keptn-templates
- Or simply call the Keptn API yourself from your CI/CD tool like we did it in prior steps





Quick Status Check: are we all good with accessing our environments?

- Please mark your tasks accordingly in the Excel file



Extra

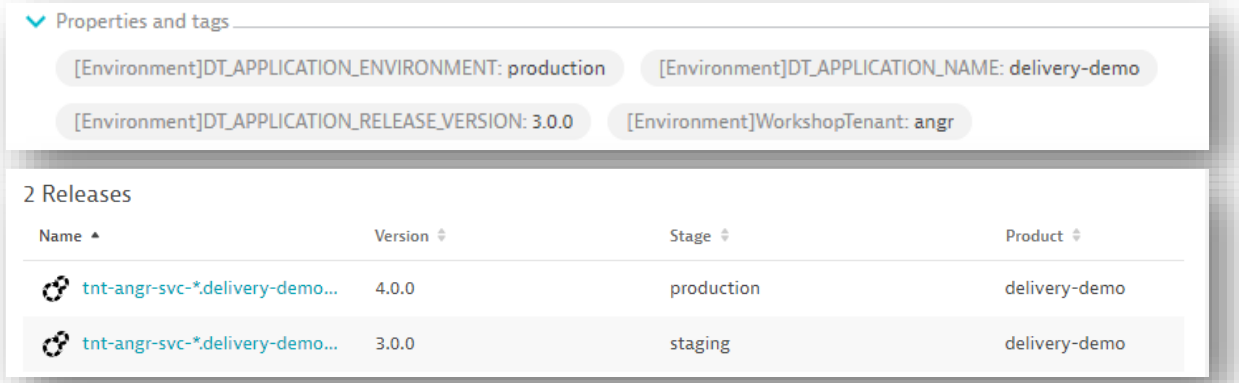
Monaco and Release Inventory

WATCH TRAINER SECTION

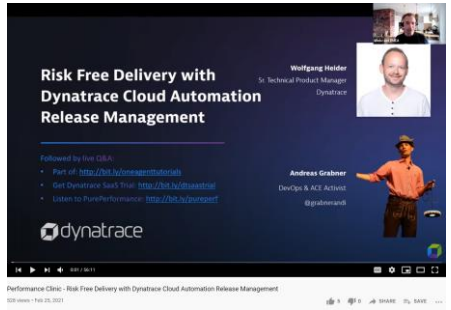
Behind the scenes of Release Inventory

- 3 Environment Variables define Application, Environment and Version
  - More on https://www.dynatrace.com/support/help/how-to-use-dynatrace/cloud-automation/release-monitoring/

```
- name: DT_RELEASE_VERSION
  valueFrom:
    fieldRef:
      fieldPath: "metadata.labels['app.kubernetes.io/version']"
- name: DT_RELEASE_PRODUCT
  value: "{{ .Values.keptn.project }}"
- name: DT_RELEASE_STAGE
  value: "{{ .Values.keptn.stage }}"
```



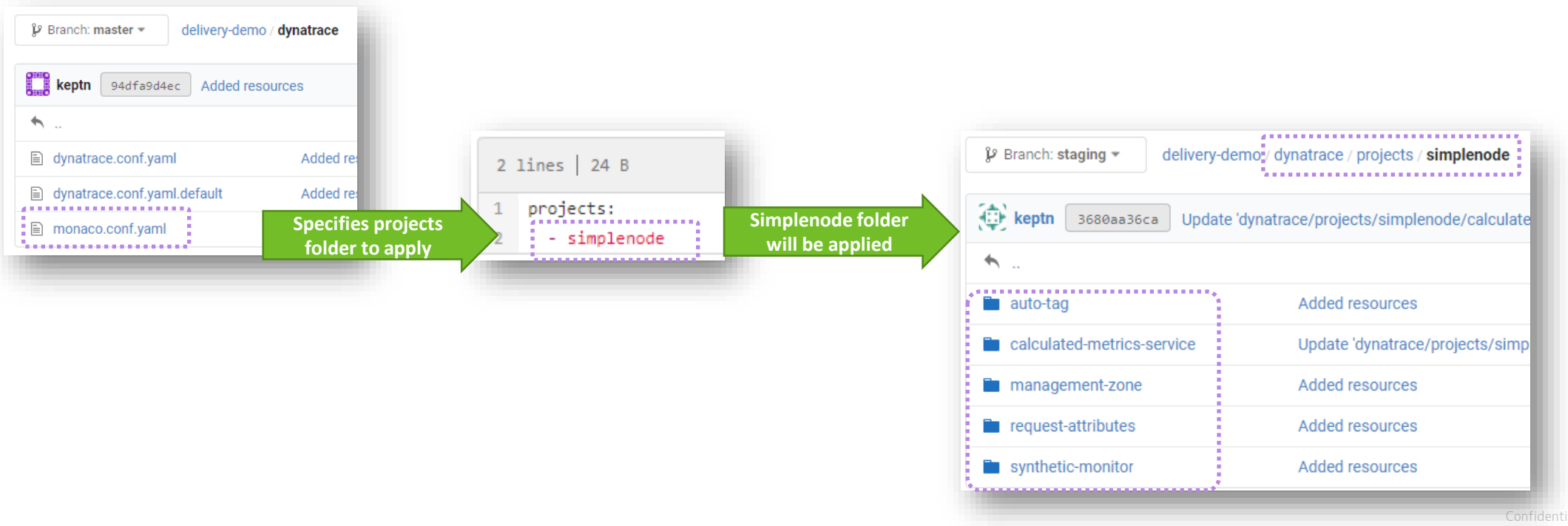
- More details about Release Management with Dynatrace in our Performance Clinic
  - YouTube: https://www.youtube.com/watch?v=TacwGZXX9pw&list=PLqt2rd0eew1YFx9m8dBFSiGYSBcDuWG38&index=17



WATCH TRAINER SECTION

Lets explore Monaco (Monitoring as Code)

- Monaco is a Dynatrace Open Source project driven by our own internal Autonomous Cloud Team
  - https://dynatrace-oss.github.io/dynatrace-monitoring-as-code/
  - You can simply use it from the command line to apply any dynatrace configuration!
- Cloud Automation can use Keptn's monaco-service to apply Monaco configuration to Dynatrace



Quick Status Check: are we all good with accessing our environments?

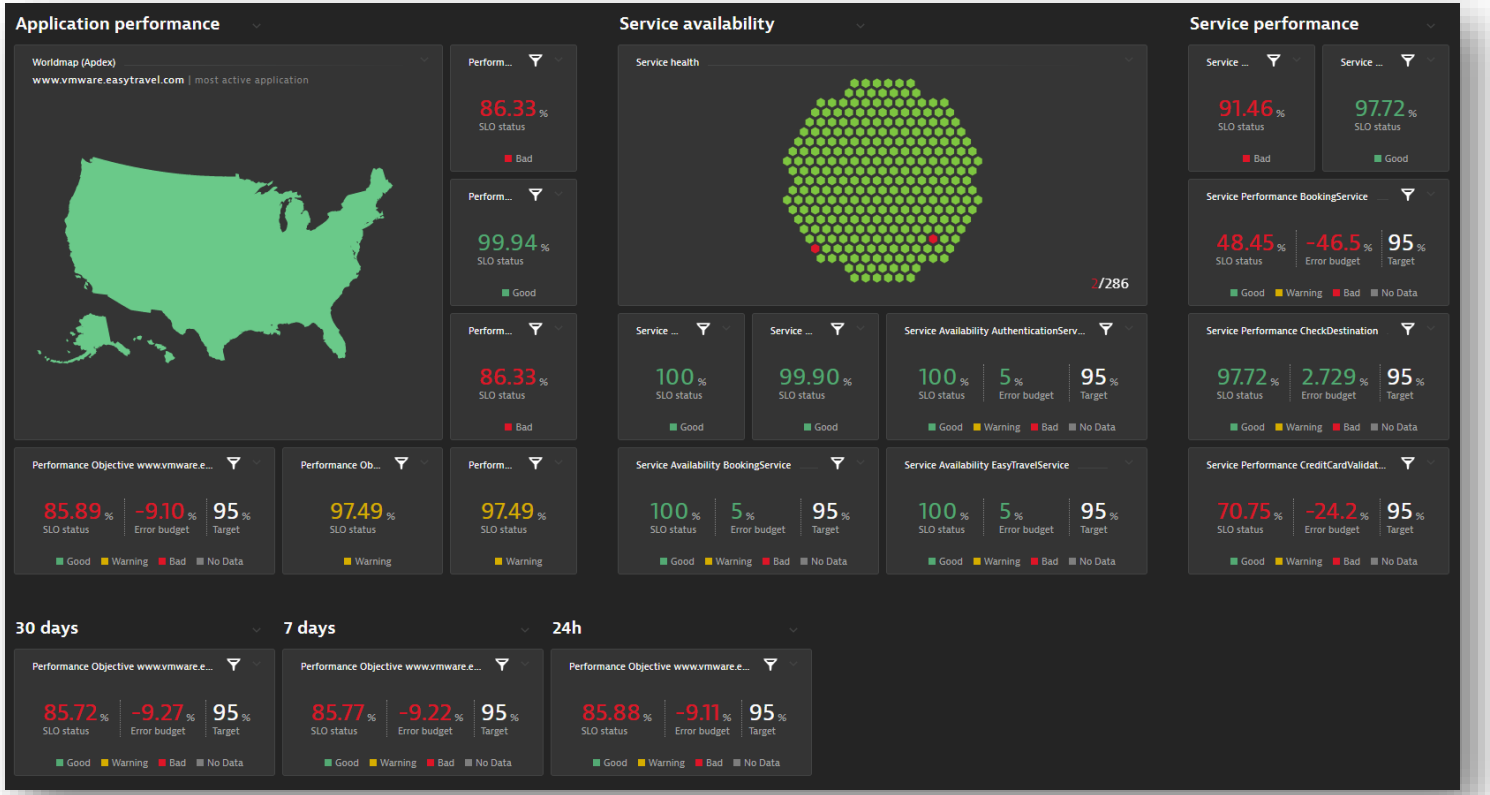
- Please mark your tasks accordingly in the Excel file

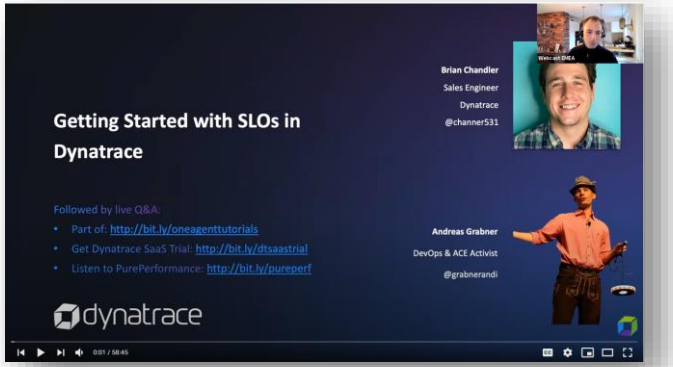
Monaco & Release Inventory	
Understand the basics behind Release Inventory and where to look for more	Understand the basics behind Monaco and where to look for more

More content for you to take home ...

dynatrace

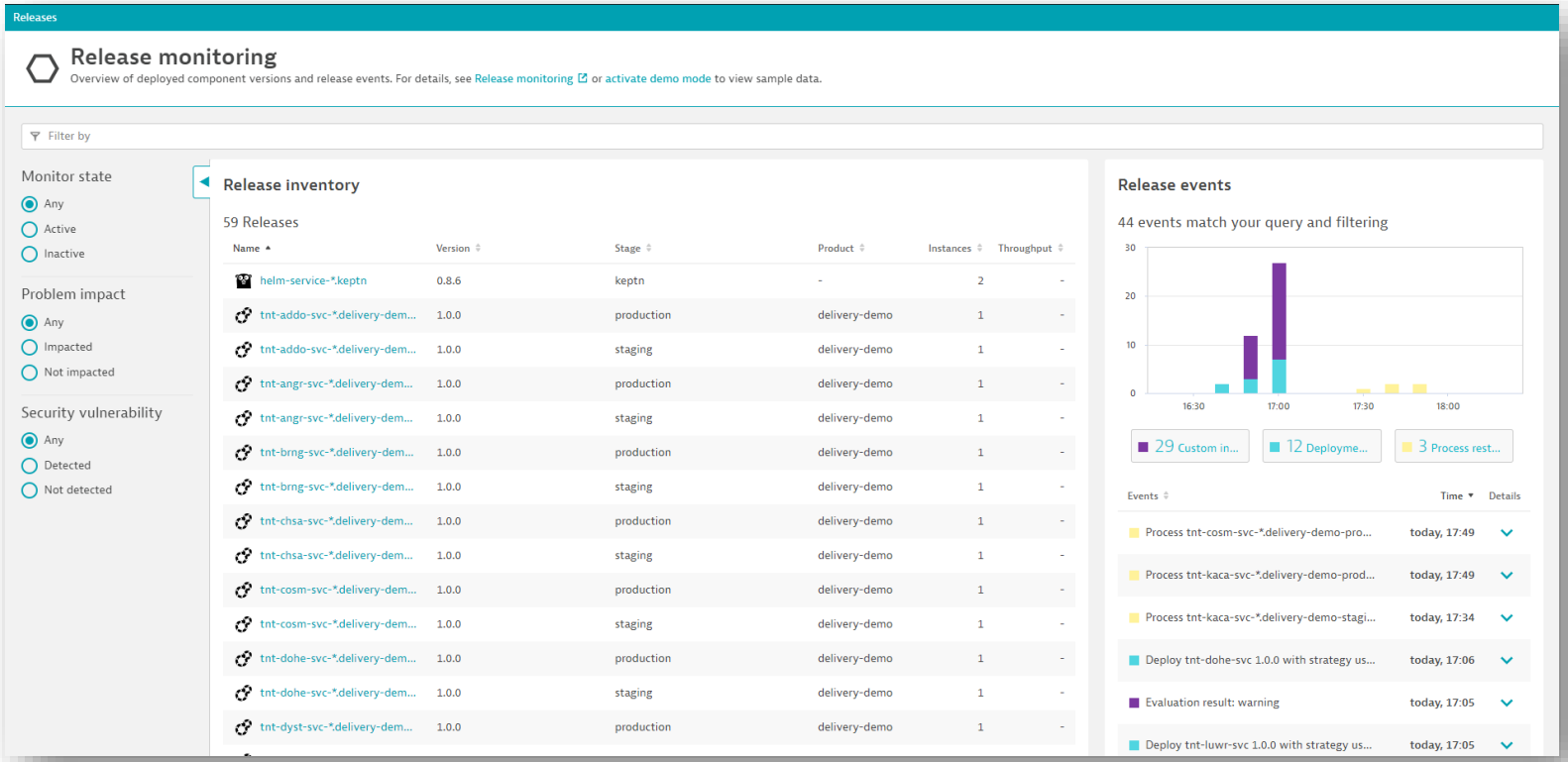
Service Level Objectives for Production Monitoring

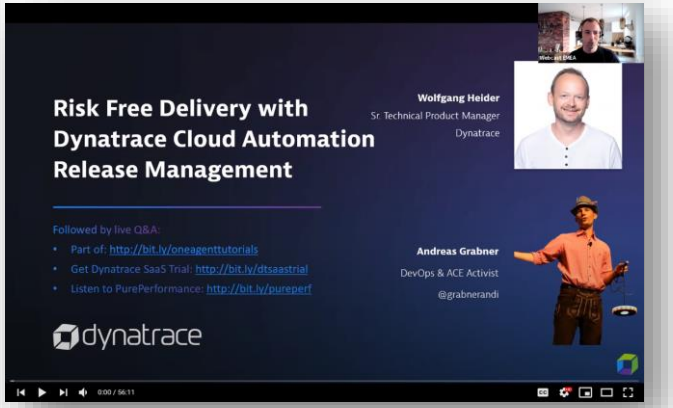




https://www.youtube.com/watch?v=4fhcxCa3m_c

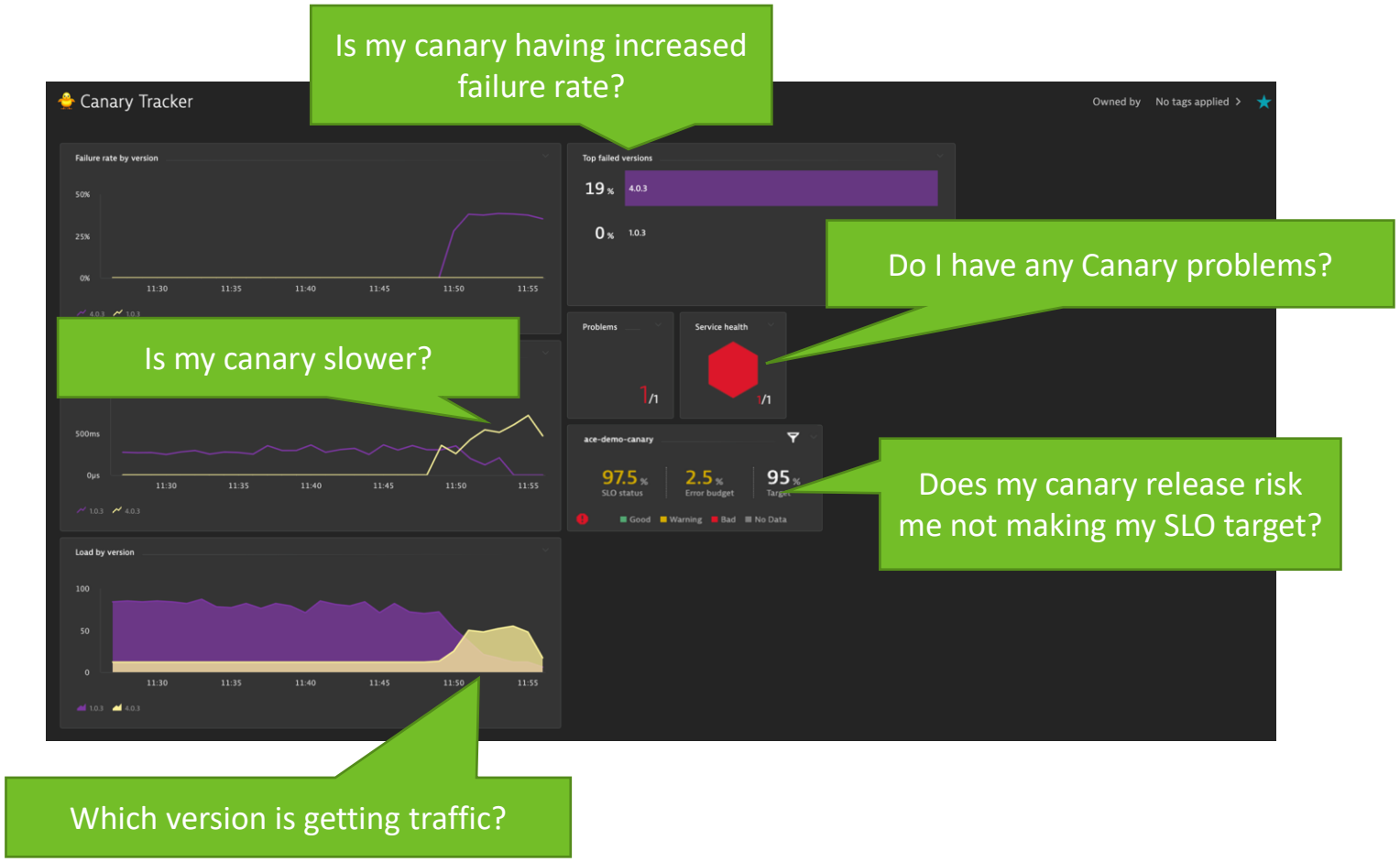
Release Awareness / Version Detection

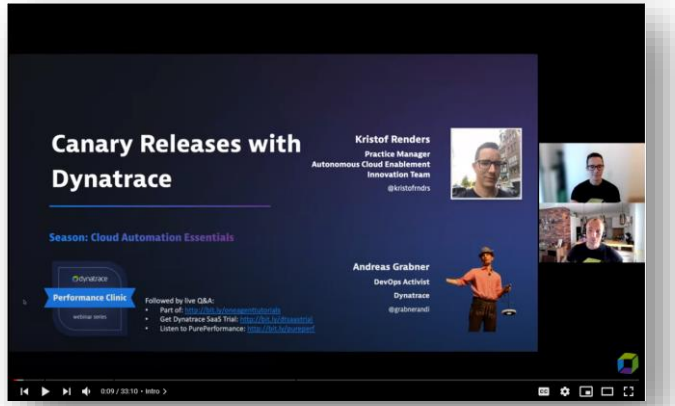




https://www.youtube.com/watch?v=TacwGZXX9pw

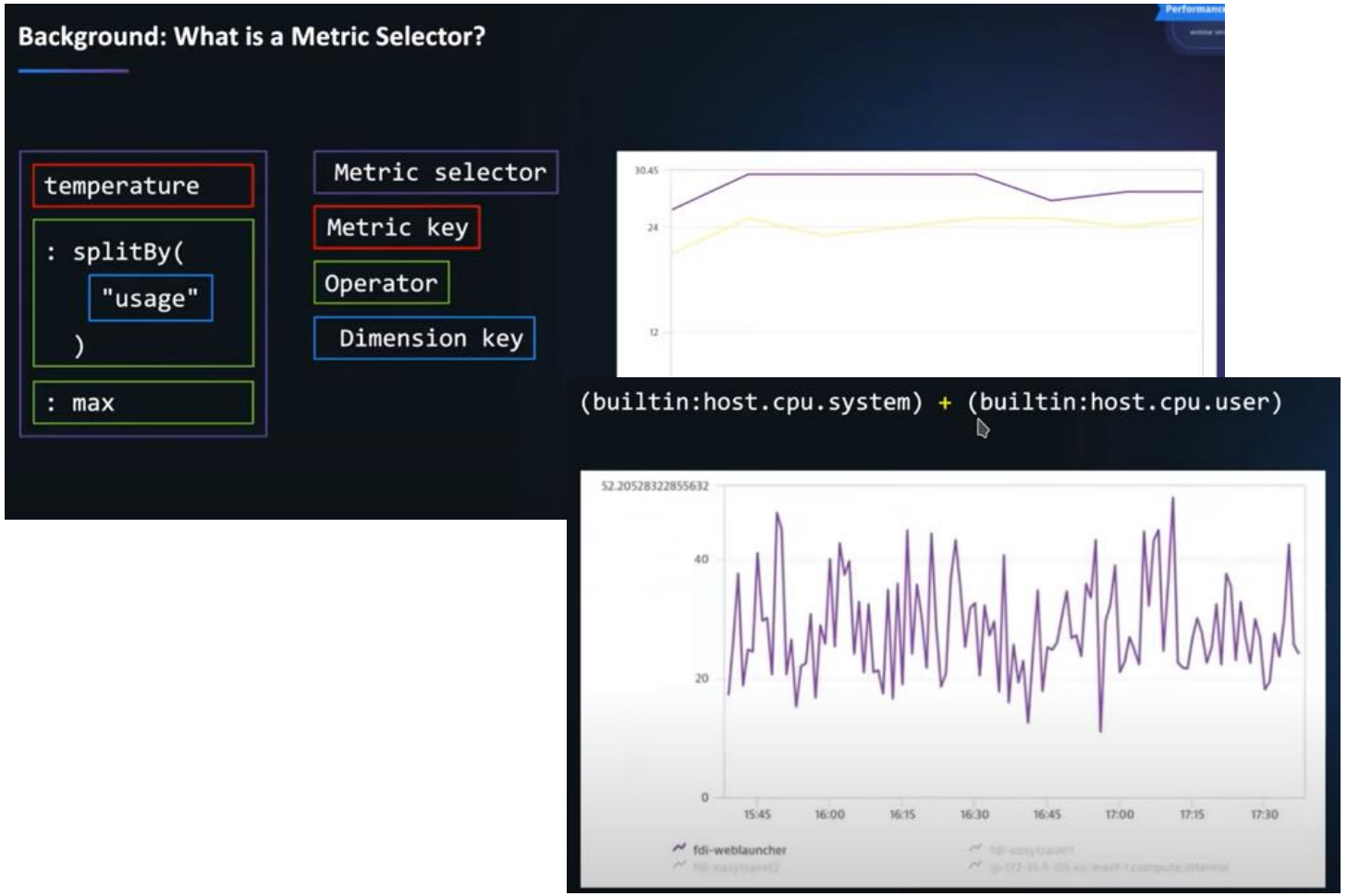
Better Zero Downtime Deployment Decisions for Canary, Blue/Green, Feature Flagging

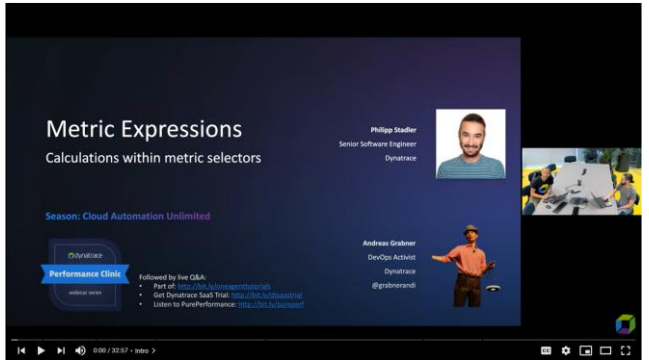




https://www.youtube.com/watch?v=VgBw3amiL-c

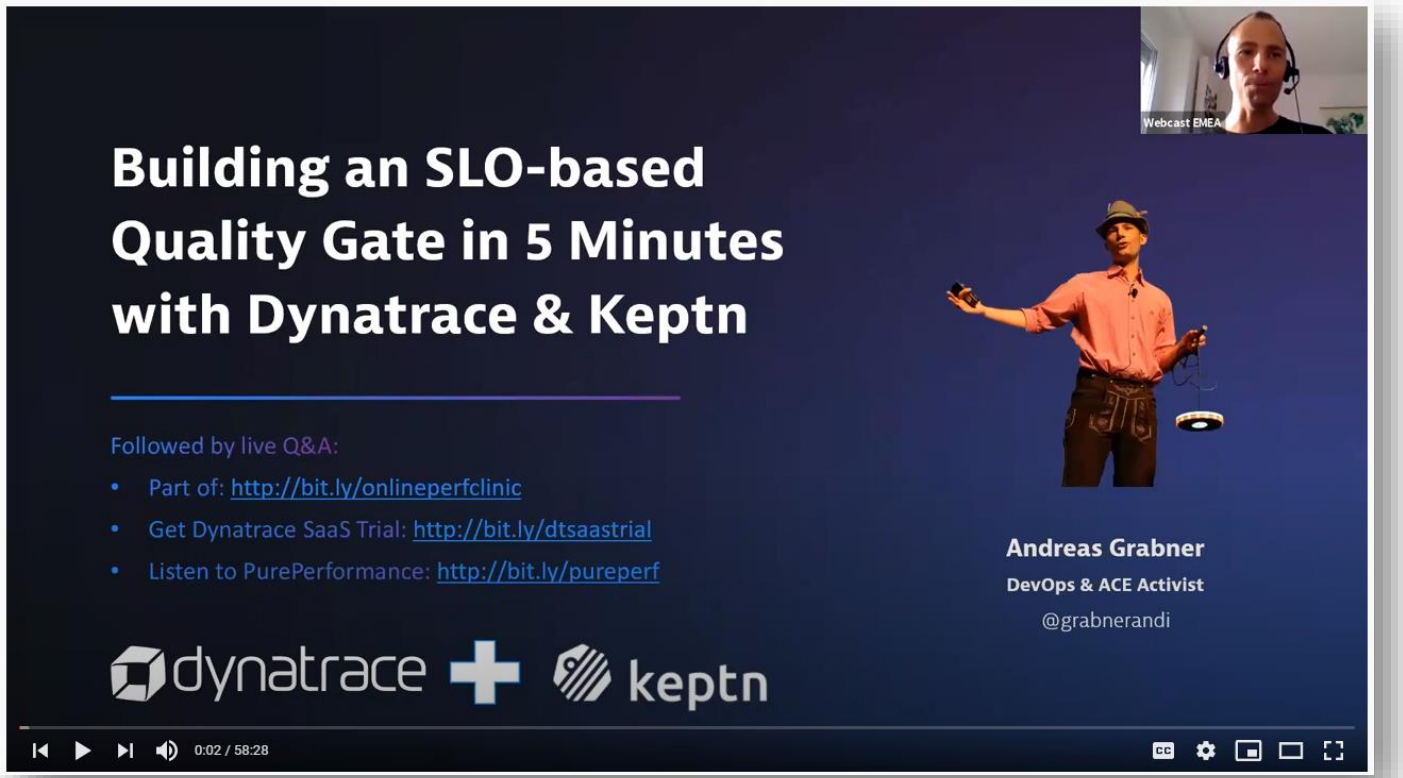
Metrics expressions give better answers through calculation support





https://www.youtube.com/watch?v=jkVyJn1HdkM

For more details on the dashboarding check out the following Performance Clinic



https://www.youtube.com/watch?v=650Gn--XEQE&list=PLqt2rd0eew1YFx9m8dBFSiGYSBcDuWG38&index=12&t=2s