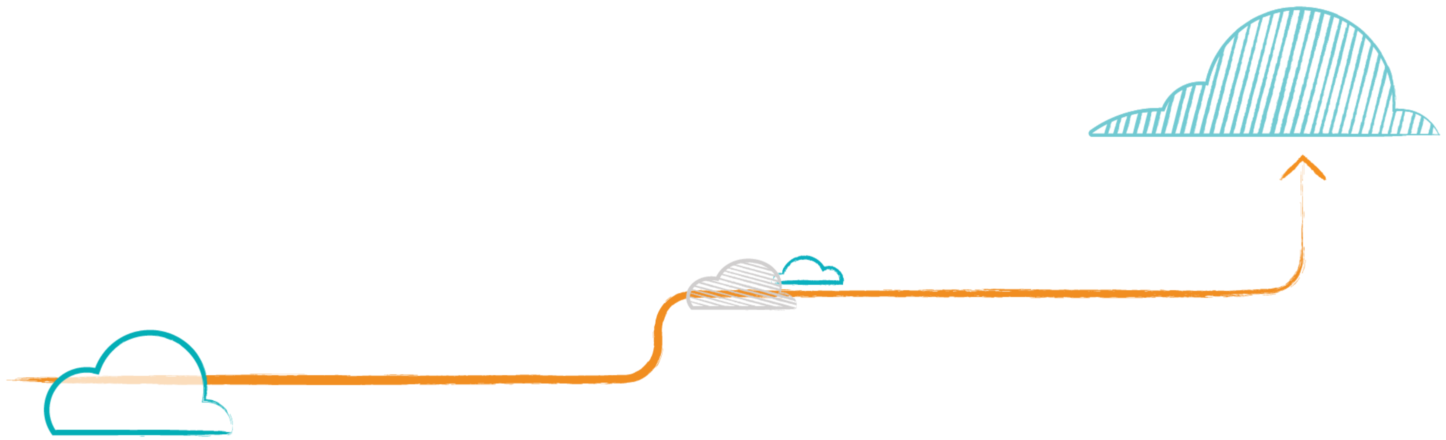
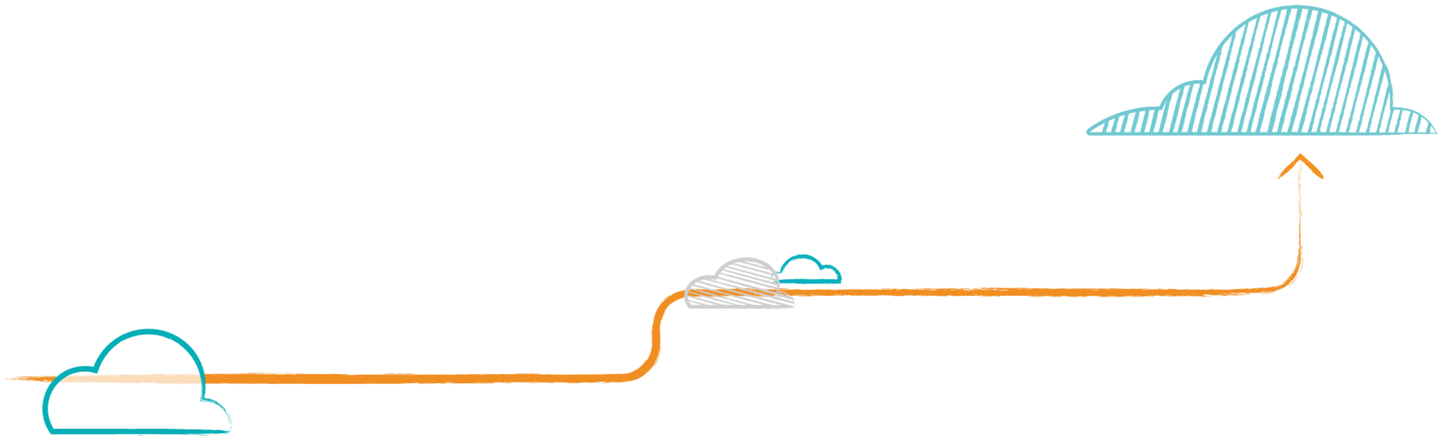
New Relic Practitioner

Optimizing Monitoring





Introduction

This lab is intended to give you experience of improving New Relic Infrastructure solution and of building an on-host integration to extend the information that the New Relic Infrastructure agent collects. Use your own free account License to perform this Lab.

Please note to begin the following lab, you must have gone through the following:

- Please ensure you have received your webserver's IP address and/or Public DNS name!

Exercise 1: Instrumenting the Infrastructure and APM Agent

Objective	*Add useful information to the Infra agent, making it easier to find specific hosts and services.*
Time	10 minutes

New Relic Infra agent allows for additional instrumentation.

Try making the following configuration file changes:

1. Open the newrelic-infra.yml config file
   ```
   sudo nano /etc/newrelic-infra.yml
   ```
2. Override the auto-generated hostname for reporting by setting a value for 'display_name' and also change the log file location to `/var/log/nr-infra.log` adding the following lines to the end of the configuration file:
   ```
         log_file: /var/log/nr-infra.log
         display_name: <YOUR_NAME>
   ```
3. Press CTRL+W to save and CTRL+X to exit the nano editor.
4. Restart the infrastructure agent
   ```
   sudo service newrelic-infra restart
   ```
5. Verify this change is visible in the New Relic Infrastructure UI.

NOTE: More configuration changes can be found here: https://docs.newrelic.com/docs/infrastructure/new-relic-infrastructure/configuration/configure-infrastructure-agent

6. Add custom attributes to the agent by editing the configuration file:
   a. Set 'environment' to 'development'
   b. Set 'status' to 'active´
   c. Set 'team' to 'NRP-Students'

Upon completion, your configuration file should look something like this:

```
      license_key: <LICENSE KEY>
      log_file: /var/log/nr-infra.log
      display_name: <YOUR NAME>
      custom_attributes:
        environment: development
        status: active
        team: NRP-Students
```

Don't forget to restart the New Relic infrastructure agent

```
sudo service newrelic-infra restart
```

7. Click on the **Query your data** icon on the top menu, and enter the following query:

```
FROM SystemSample select * since 5 minutes ago
```

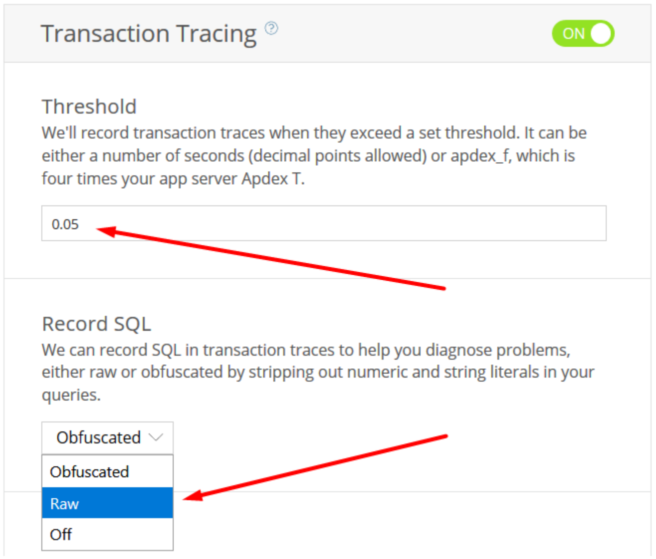
Using the NRQL syntax we used on Mod 02 and on Lab 01, try to answer these questions:
   a. Can you see the columns of the SystemSample events with the data you just configured?
   b. Can you select the average of CPU usage of all hosts from the NRP-Students team?
   c. Can you see the DisplayName of all the hosts on the NRP-Students team?

   You can ask your instructor for help if needed.

Now, let´s change the APM agent for Petclinic to be able, for example, to read the obfuscated SQL queries on traces and also reduce the APDEX threshold.

1. On New Relic UI, go to the **APM** at the top menu
2. Click into your Petclinic application

3. In the left Navigation menu, under **Settings**, click on **Application**
4. In the **Server-side Agent Configuration** tab, click the **Enable** switch
5. Under Record SQL, switch **Obfuscated** to **Raw**
6. Under Threshold, replace the text **apdex_f** by **0.05**



Transaction Tracing

Threshold

We'll record transaction traces when they exceed a set threshold. It can be either a number of seconds (decimal points allowed) or apdex_f, which is four times your app server Apdex T.

0.05

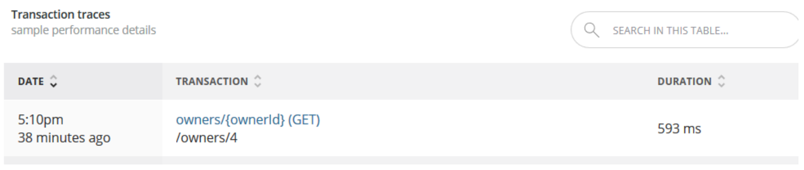
Record SQL

We can record SQL in transaction traces to help you diagnose problems, either raw or obfuscated by stripping out numeric and string literals in your queries.

7. Click then on **Save server-side configuration.**

You may need to wait a few minutes until a trace is collected. To see a SQL query collected:

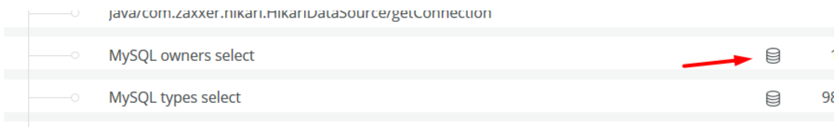
1. Go to the **APM** at the top menu
2. Click into your Petclinic application
3. In the left Navigation menu, click **Transactions**.
4. Select the **owners/{ownerId} (GET)** transaction
5. Scroll down on the right screen, until you find the **Transaction Traces** and select the most recent trace (it should be one that happened AFTER you changed the agent configuration – if you can´t find one, wait until one is collected, is a good time to get a cup of coffee.



Transaction traces
sample performance details

DATE	TRANSACTION	DURATION
5:10pm 38 minutes ago	owners/{ownerId} (GET) /owners/4	593 ms

6. Click on the trace, and then on **Trace details** on the top tabs.
7. Look for the SQL entries, and click on the DB icon:



8. You should see the SQL query that was executed at this part of the code. Note that you can view the entire query, including IDs from the WHERE SQL clauses, and not a “?” replacing it.
9. Compare this with older traces, where you can´t see the IDs and values.

Exercise 2: Creating a distributed Petclinic app and enabling Prometheus monitoring

Objective	*Applications are usually distributed, and some have metrics collected by Prometheus. On this lab, we will start a distributed Petclinic app and will check how to redirect Prometheus metrics from it to New Relic.*
Time	20 minutes

For this lab, we will use another Spring Boot project with a microservice-oriented Petclinic application, where each application part (UI, Vets, Customer and Visits) is a separate container. For simplicity, all containers have already been injected the New Relic agent (same exercise done on Lab 2). We will be enabling New Relic instrumentation and configuring all services to run with Distributed tracing enabled.

1. Enter the distributed application directory and edit the `docker-compose.yml` file:

```
cd ~/nrp-lab/petclinic-distributed-docker
nano docker-compose.yml
```

2. This file describes how each container starts. Besides the 4 microservices, there are other containers (discovery services, Zipkin, Prometheus) that instrument the petclininc app. You can find more information if you need here: https://github.com/spring-petclinic/spring-petclinic-microservices. We will focus on the following services:
   a. customers-service
   b. visits-service
   c. vets-service
   d. api-gateway (UI)
3. Note that for each one of those services, there are already environment variables configured for NEWRELIC_APP_NAME and NEW_RELIC_DISTRIBUTED_TRACING_ENABLED. Put your license key (API Ingestion key) on the NEW_RELIC_LICENSE_KEY for all 4 services. Here is an example of the vets-service, make sure you add your key on all 4 services:

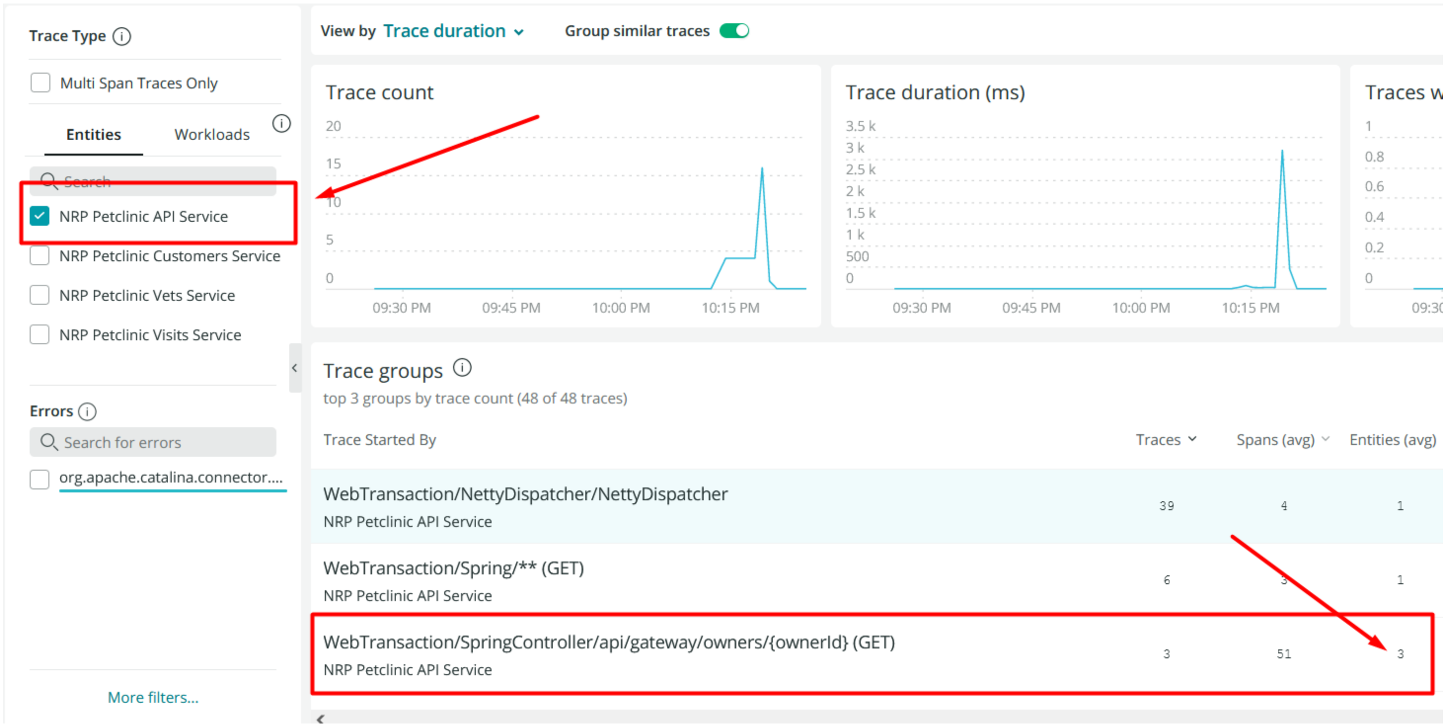
```
  vets-service:
    image: registry.pegasus-consultancy.co.uk:5000/newrelic/nrp-lab/...
    container_name: vets-service
    mem_limit: 512M
    depends_on:
     - config-server
     - discovery-server
    entrypoint: ["./dockerize","-wait=tcp://discovery-server:8761"...
    ports:
     - 8083:8083
    environment:
    - NEW_RELIC_LICENSE_KEY=<YOUR_LICENSE_KEY_HERE>
    - NEW_RELIC_APP_NAME=NRP Petclinic Vets Service
    - NEW_RELIC_DISTRIBUTED_TRACING_ENABLED=true
```

4. Save the file and start the application:

```
docker-compose -f docker-compose.yml up -d
```

5. After all containers start, check if the application is up at http://<YOUR_LAB_IP>:8080/ - it may take 1 or 2 minutes for the application to start. Navigate over the application, check Vets, Owners, etc, to generate some load.
6. Check on the New Relic UI if the new application appears.
7. Click on **Query your data > Data explorer > Traces**. Select the NRP Petclinic API Service on the left menu and look for any trace that have more than or 2 entities (if you don´t find

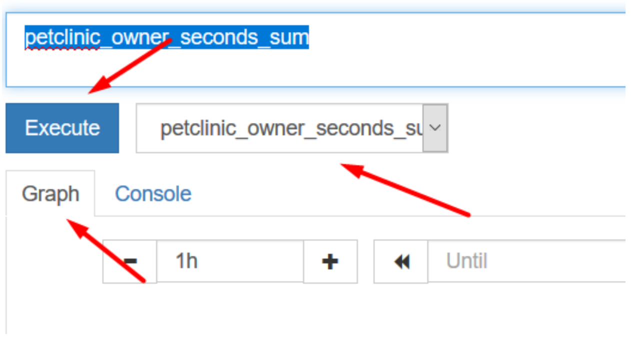
it, generate more load on the Petclinic application and go back to traces). Explore the trace information and observe what are the service used on the Services map.



Trace groups
top 3 groups by trace count (48 of 48 traces)

Trace Started By	Traces	Spans (avg)	Entities (avg)
WebTransaction/NettyDispatcher/NettyDispatcher NRP Petclinic API Service	39	4	1
WebTransaction/Spring/** (GET) NRP Petclinic API Service	6	3	1
WebTransaction/SpringController/api/gateway/owners/{ownerId} (GET) NRP Petclinic API Service	3	51	3

8. Explore the trace information and observe what are the services used on the Services map for the trace.
9. You can also check the Prometheus interface and check metrics. Navigate to http://<YOUR_LAB_IP>:9091 and click on the **Graph** tab, then select the metric **petclinic_owner_seconds_sum** and click on **Execute.** This will display how many seconds the Owner transaction is taking.



10. Remember, this was instrumented by the developers and have nothing to do with New Relic. Our Prometheus server is storing this metrics and we are able to see it as long as the server have resources to store it – but we can also store those metrics on New Relic.
11. Go back to the New Relic UI, go to **Add more data,** under **Open Source monitoring systems** and select **Prometheus**
12. Provide “nrp-prometheus” as a name for your monitored system and click on **Generate URL.**
13. Go back to the VM console and edit the prometheus.yml configuration file:

```
nano ~/nrp-lab/petclinic-distributed-docker/docker/prometheus/prometheus.yml
```

14. Copy the generated URL in the New Relic UI clicking on the **Copy to clipboard** icon and paste the copied lines to the END of the configuration file. You should have the following file contents (just showing the last lines here):

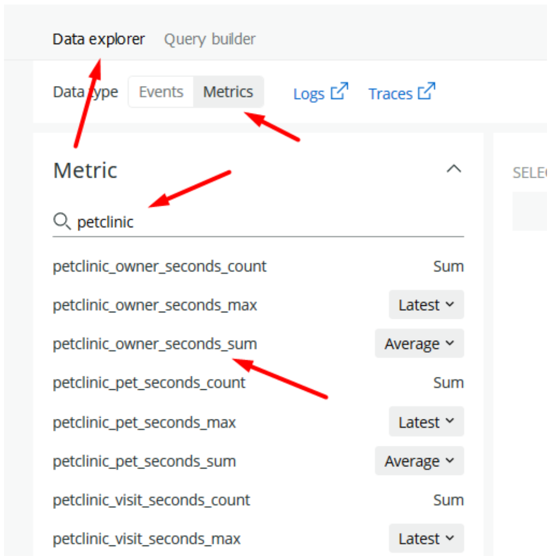
```
<TRUNCATED OUTPUT>
  static_configs:
  - targets: ['visits-service:8082']

- job_name: vets-service
  metrics_path: /actuator/prometheus
  static_configs:
  - targets: ['vets-service:8083']

remote_write:
- url: https://metric-api.newrelic.com/prometheus/v1/write?prometheus_server=nrp...
  bearer_token: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

15. Save with CTRL+O and exit with CTRL+X. Now let´s rebuild the Prometheus application including the new file and restart the application:

```
docker-compose -f docker-compose.yml down
docker rmi petclinic-distributed-docker_prometheus-server
docker-compose -f docker-compose.yml up -d
```

16. Generate some traffic again on the http://<YOUR_LAB_IP>:8080 (it may take 1 or 2 minutes to start) by clicking on different owners (try 3 or 4).

17. Go back to your New Relic account UI. Click the **Query your data** icon on the top menu. Select the Data explorer, and check for petclinic related metrics, selecting the same **petclinic_owner_seconds_sum** metric:


Data explorer
Query builder
Data type
Events
Metrics
Logs
Traces
Metric
petclinic
petclinic_owner_seconds_count
Sum
petclinic_owner_seconds_max
Latest
petclinic_owner_seconds_sum
Average
petclinic_pet_seconds_count
Sum
petclinic_pet_seconds_max
Latest
petclinic_pet_seconds_sum
Average
petclinic_visit_seconds_count
Sum
petclinic_visit_seconds_max
Latest

18. You should start to see the metric being plotted. Explore under **Dimensions** what are the filters available to better filter only the metrics you want to see, in case you have several Prometheus instances sending data, or have several URI on the metrics.
19. Let´s stop this lab containers as it uses a lot of memory, and we will need resources for the next labs:

```
docker-compose -f docker-compose.yml down
```

Exercise 3: Enabling Kubernetes monitoring

Objective	*To be able to monitor Kubernetes, we must apply the New Relic configuration to a running cluster. Pods, Nodes, Deployments and Services from on-premises and Cloud clusters can be easily monitored with this feature.*
Time	15 minutes

For this lab, we will need to install a small Kubernetes Cluster on our VM as an example. To install it, execute the following commands:

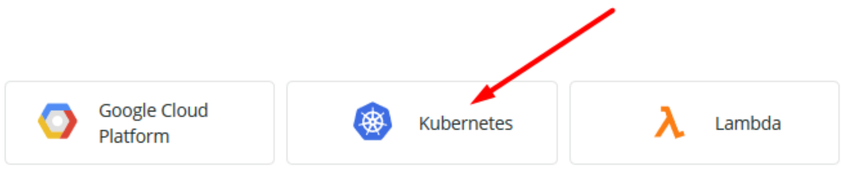
1. Install the Cluster using the following command:

```
curl -sfL https://get.k3s.io | sh -
```

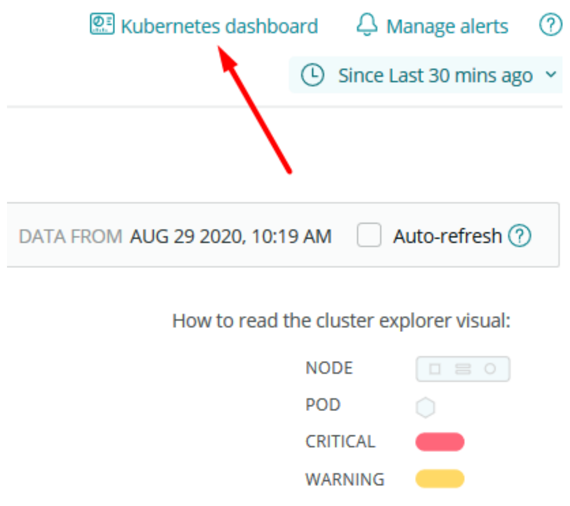
2. For the benefit of the lab exercise (this is NOT the safest way to do that), let´s allow access to our Kubernetes config so we can run commands on the cluster and also install HELM to apply New Relic configurations:

```
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
sudo chmod 644 /etc/rancher/k3s/k3s.yaml
curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 | bash
```

3. On the New Relic UI, go to **Add more data** and, under **Cloud and platform technologies**, select **Kubernetes**.



4. Fill in the following data on the form:
    a. On **Cluster name**, type “nrp-kubernetes”.
    b. Accept the Pixie terms and services.
5. Click on **Continue** and copy the HELM 3 script, pasting on your Lab VM console.
6. Click on **Continue** and wait for data to be retrieved. It may take 1 or 2 minutes. After that click on **Kubernetes explorer**.
7. Check the information being displayed for your cluster.
8. Click on **K8s dashboard** on the top right:



9. Now let´s move our Petclinic application to our Kubernetes cluster. First, let´s shutdown the container:

```
docker-compose -f ~/nrp-lab/petclinic-docker/docker-compose.yml down
```

10. Enter the petclininc Kubernetes configuration directory and edit the deployment config:

```
cd ~/nrp-lab/petclinic-kubernetes
nano petclinic-deployment.yml
```

11. On this configuration, we need to specify the same environment variables as we did on the docker-compose.yml file, but in a distinct syntax. Configure the "value:" for the **NEW_RELIC_LICENSE_KEY** and **NEW_RELIC_APP_NAME** with the same values configured on the docker-compose.yml file. Your configuration should look like this (notice the fileds to be changed in bold):

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: petclinic-kubernetes
  labels:
    app: petclinic
spec:
  replicas: 1
  selector:
    matchLabels:
      app: petclinic
  template:
    metadata:
      labels:
        app: petclinic
    spec:
      hostAliases:
      - ip: "172.17.0.1"
        hostnames:
        - "mysql"
      containers:
      - name: petclinic
        env:
        - name: JAVA_OPTS
          value: "-javaagent:/newrelic-agent/newrelic.jar"
        - name: NEW_RELIC_LICENSE_KEY
          value: XXXXXXXXXXXXXXXXXXXXXXXX
        - name: NEW_RELIC_APP_NAME
          value: NRP Petclinic
        image: registry.pegasus-consultancy.co.uk:5000/newrelic/nrp-lab/pegasus...
        ports:
        - containerPort: 8080
```

12. Save and exit the file. Notice we configure the environment variables in another way – you could do that with any other application as well if it accepts environment variables configuration. Also, notice we are using an image available over the internet – this is an image our developers already pushed and have the New Relic agent embedded (as we did on lab 2).
13. (OPTIONAL) Read the petclinic-service.yml and petclinic-ingress.yml files to check how the configuration is done -any questions please ask your instructor.
14. Now let´s apply all the config by doing:

```
kubectl apply -f petclinic-deployment.yml
kubectl apply -f petclinic-service.yml
kubectl apply -f petclinic-ingress.yml
```

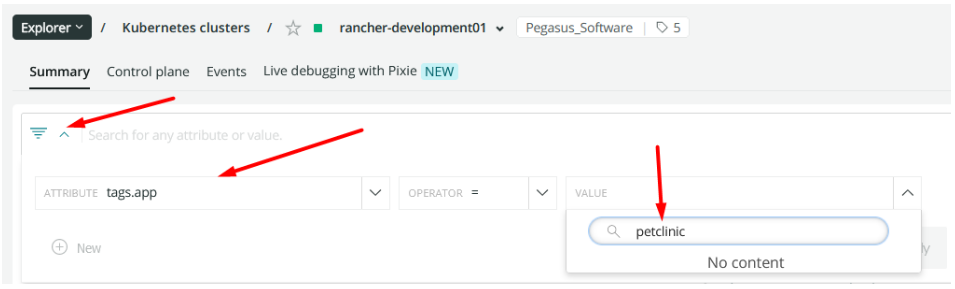
15. Wait around 1 or 2 minutes and run the following command to check if the application started:

```
kubectl get pods
```

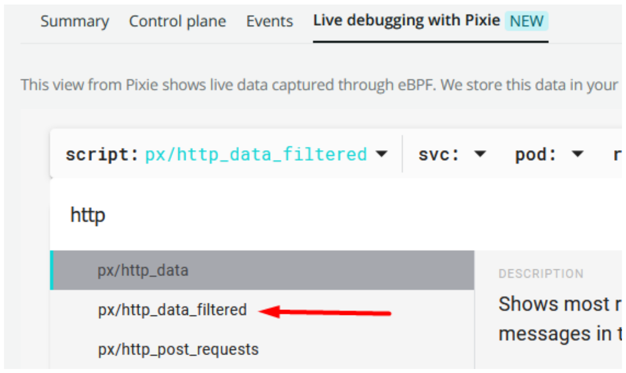
16. Check the pod with a name starting with "petclinic-kuberetes" – the status should be "running" – if is not, check again in a few seconds.

```
NAME                                                      READY STATUS  RESTARTS  AGE
newrelic-bundle-newrelic-logging-z8sq2                     1/1  Running 0         26m
newrelic-bundle-nri-metadata-injection-6c48474664-dzbpl    1/1  Running 0         26m
newrelic-bundle-nri-prometheus-cd5bc79d9-db8js             1/1  Running 0         26m
newrelic-bundle-kube-state-metrics-857949d85-7qkdg         1/1  Running 0         26m
newrelic-bundle-nri-kube-events-fffb5bf5-22j87             2/2  Running 0         26m
newrelic-bundle-newrelic-infrastructure-f9nxp              1/1  Running 0         26m
petclinic-kubernetes-57db5cfc7d-q7s65                      1/1  Running 0         2m33s
```

17. Go to your browser and check if the application is responding correctly on http://<YOUR_LAB_IP>/petclinic
18. Back in the New Relic UI, click on Explorer > Kubernetes and select your Kubernetes cluster.
19. Click on the small Filter icon and select all objects where the label **app.name** being equals to **petclinic**.



20. Observe that New Relic finds the petclinic pod and shows where it is running. Click on the pod (green hexagon) and check the information about it.
21. Now, on the top of the explorer, click on **Live debugging with Pixie**.
22. On the Script button, select px/http_data_filtered.



This view from Pixie shows live data captured through eBPF. We store this data in your

Shows most r
messages in t

23. On the **svc** submenu, select the petclinic service we just deployed. Observe the transactions being executed – it should be the same ones as we see on the transactions, with telemetry data (duration, execution, etc).
24. Check on the **Kubernetes Dashboard** if you can see information about the "petclinic" pod. You can also run this NRQL query and change the "select" part to check additional data that ca be retrieved:

```
FROM K8sContainerSample select uniques(status) where podName='petclinic-
kubernetes<AUTO-COMPLETE HERE>
```

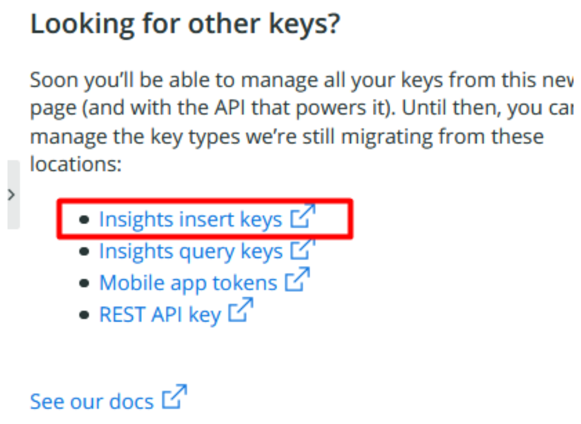
25. Explore the Kubernetes dashboard and Metrics retrieved with this integration. Also, check if you can view the logs from the Kubernetes cluster on the Logs UI.

Exercise 4: Send custom events

Objective	*New Relic accepts the ingestion of any type of information using Custom Events – we just need to create the logic to send those events to the NRDB. We will be checking if a specific file exists in our VM and report as an event.*
Time	20 minutes

For this lab, we need to create a ingestion API key in order to get information from a API call done by a script:

1. On the top right menu with your name, click on **API Keys**.
2. On the right menu, click on **Insights insert Keys.**



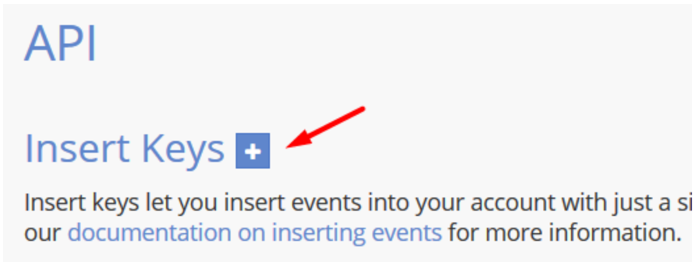
Looking for other keys?

Soon you'll be able to manage all your keys from this nev page (and with the API that powers it). Until then, you cai manage the key types we're still migrating from these locations:

- Insights insert keys
- Insights query keys
- Mobile app tokens
- REST API key

See our docs

3. Click on the + sign on the **Insert Keys**

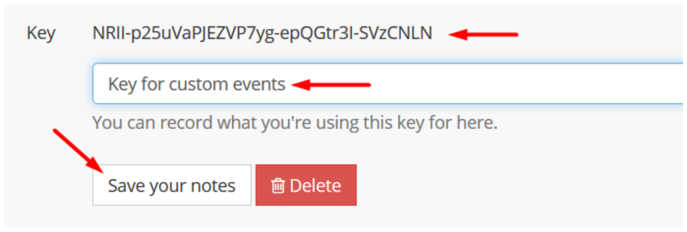


API

Insert Keys

Insert keys let you insert events into your account with just a si our documentation on inserting events for more information.

4. Copy your key to a notepad (we will need it later as well) and add a comment so you know what this key is being used for. Click on **Save your notes**



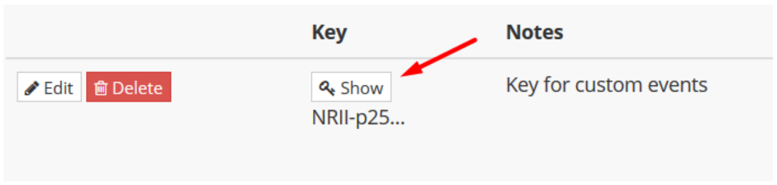
Key NRII-p25uVaPJEZVP7yg-epQGtr3I-SVzCNLN

Key for custom events

You can record what you're using this key for here.

Save your notes   Delete

5. You can also check your key later by clicking **Show**

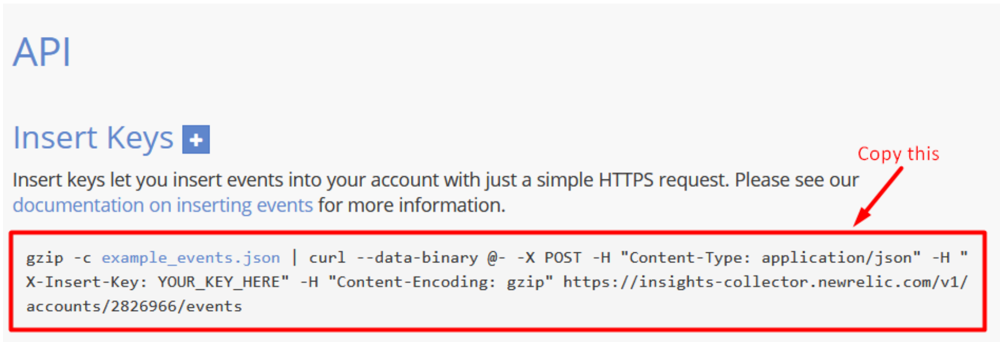


	Key	Notes
Edit Delete	Show NRII-p25...	Key for custom events

6. Now go to your Linux machine. Run the following command to create a file that we will suppose is a super important one (for example, it could be a backup file or some batch TXT file we need for our systems). And also edit a simple script that checks if the file exists and perform a API call to tell New Relic the status of it.

```
touch ~/super-important-file.txt
nano ~/nrp-lab/lab3-exercise5.sh
```

7. We will be creating a Custom Event type called **Custom file check**, sending the API payload from a curl command. On the New Relic API Inserts key UI copy the cURL command:

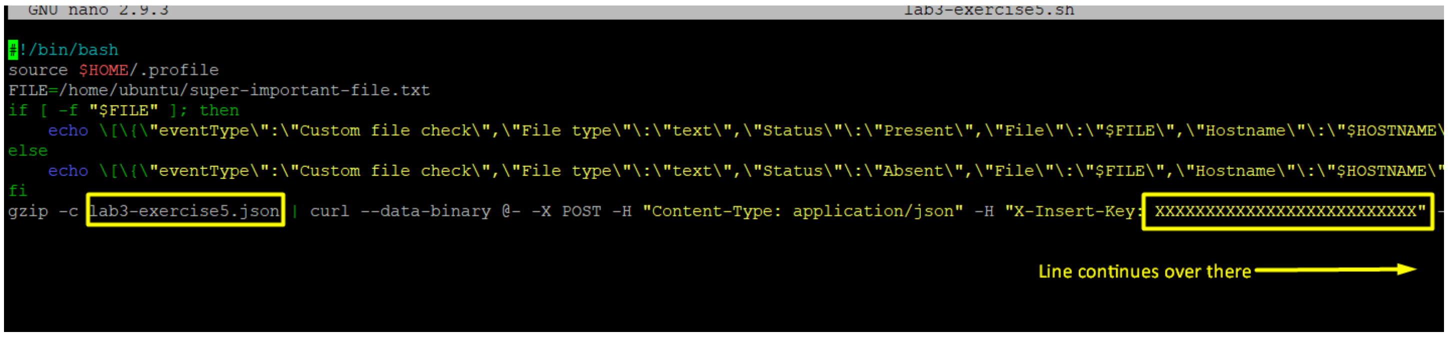

API

Insert Keys

Insert keys let you insert events into your account with just a simple HTTPS request. Please see our documentation on inserting events for more information.

```
gzip -c example_events.json | curl --data-binary @- -X POST -H "Content-Type: application/json" -H "
X-Insert-Key: YOUR_KEY_HERE" -H "Content-Encoding: gzip" https://insights-collector.newrelic.com/v1/
accounts/2826966/events
```

8. Paste it at the end of the lab3-exercise5.sh file, replacing **example_events.json** by **lab3-exercise5.json** and **YOUR_API_KEY** by your Insights Insert key. Your file should look like this:


```
  GNU nano 2.9.3                                                    lab3-exercise5.sh

#!/bin/bash
source $HOME/.profile
FILE=/home/ubuntu/super-important-file.txt
if [ -f "$FILE" ]; then
    echo \[\{\"eventType\":\"Custom file check\",\"File type\"\:\"text\",\"Status\"\:\"Present\",\"File\"\:\"$FILE\",\"Hostname\"\:\"$HOSTNAME\
else
    echo \[\{\"eventType\":\"Custom file check\",\"File type\"\:\"text\",\"Status\"\:\"Absent\",\"File\"\:\"$FILE\",\"Hostname\"\:\"$HOSTNAME\"
fi
gzip -c lab3-exercise5.json | curl --data-binary @- -X POST -H "Content-Type: application/json" -H "X-Insert-Key: XXXXXXXXXXXXXXXXXXXXXXXXXX" -
```

The payload, that is decoded on the command, is the following (just so you know what you are sending there is no need to use the below text for anything):

```
[
	{
	"eventType":"Custom file check",
	"File type":"text",
	"Status":"Present",
	"File":"$FILE",
	"Hostname”:"$HOSTNAME"
	}
]
```

You can try to use this payload replacing the hostname and file variables by anything on a REST API client. The Status is changed to “Absent” if the file don´t exists.

9. Execute the script to test.

```
~/nrp-lab/lab3-exercise5.sh
```

10. You should see an output with a ID and the word **"success":true** on it. Now let´s add the script to cron so it runs every minute. Run the command:

```
crontab -e
```

11. Place the following line at the end of the crontab file and saving with CTRL+O and CTRL+X:

```
* * * * * ~/nrp-lab/lab3-exercise5.sh
```

NOTE: the five * are needed and there must be a space between them. Ignore any other information the file may have.

12. Go back to the New Relic interface, and click on the top menu option **Query your data.** On the **Query builder** type the following query:

```
FROM `Custom file check` select *
```

13. You will notice one event per minute with the file information. Now try deleting and recreating the ~/super-important.file.txt to check how this affect the events being sent.

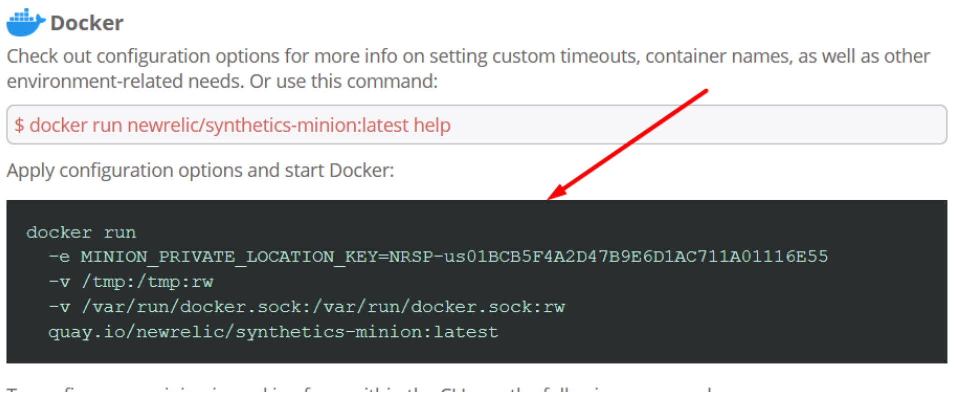
Exercise 5: Synthetics Private Locations

Objective	Synthetics can provide performance information from a private enabled location. Let´s use our lab VM to host a Private Location (minion).
Duration	20 Minutes

Create a private location

We will start creating a private location. Set up a private location as follows:

1. Go to the New Relic interface.
2. Click on **Synthetics** on the top menu, navigate to **Private locations**, and click **Create new private location**.
3. Enter a suitable name and description for your private location, and then click on **Generate key,** we will use it in the next command line.
4. Copy the instructions under "Docker" section (not Kubernetes) and paste it on your Lab VM. **IMPORTANT: Add a "-d" right after the "docker run" command so it runs in background – otherwise your console will be locked up.**

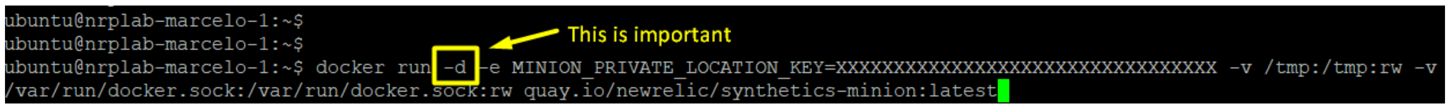


Check out configuration options for more info on setting custom timeouts, container names, as well as other environment-related needs. Or use this command:

```
$ docker run newrelic/synthetics-minion:latest help
```

Apply configuration options and start Docker:

```
docker run
  -e MINION_PRIVATE_LOCATION_KEY=NRSP-us01BCB5F4A2D47B9E6D1AC711A01116E55
  -v /tmp:/tmp:rw
  -v /var/run/docker.sock:/var/run/docker.sock:rw
  quay.io/newrelic/synthetics-minion:latest
```



```
ubuntu@nrplab-marcelo-1:~$
ubuntu@nrplab-marcelo-1:~$
ubuntu@nrplab-marcelo-1:~$ docker run -d -e MINION_PRIVATE_LOCATION_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX -v /tmp:/tmp:rw -v
/var/run/docker.sock:/var/run/docker.sock:rw quay.io/newrelic/synthetics-minion:latest
```

5. Check the logs for the container startup:

   ```
   docker logs -f <MINION_CONTAINER_NAME>
   ```

   NOTE: It may take around 15-30 minutes for the container to be ready, as the image being downloaded have around 1.5GB in size.
6. Go back to the Synthetics UI and check to see if your private location is showing up in the **Private locations** tab (click on the tab to return to the list of locations).
7. Create a Synthetics Simple Browser monitor using the private location, checking your Petclinic Owners section on http://<YOUR_VM_IP>/petclinic/owners?lastName=, Checking for the String "Owners" (title of the page) and wait to see if the check executed successfully. Write down the Monitor name as we will use it on Lab 04.

Exercise 6: (Optional) Create a Wordpress site with Cloudformation

Objective	AWS allows you to create several services using Cloudformation, and these services allow automatic configuration with AWS Cloudformation. We will be running a template that will create a usable Wordpress website
Duration	60 Minutes

NOTE: creating a Instance on AWS may imply costs. We will be checking how to create AND how to destroy the instance if needed at the end of the lab. **The cost of having this up for each single day is around 1.00 USD on your account, if you are not Free Tier eligible.**

Access Cloudformation

1. On the documents public folder your instructor provided, download a file called **NewRelic-WordPress.template**.
2. On AWS console, click on **Services** and look for Cloudformation
3. Click on **Create stack** and choose **With new resources.**
4. Under **Specify template**, upload the file you just downloaded. Click **Next.**
5. Add the following information on the form at Step 2:

   a. **Stack name:** simply name your stack for future reference (it will be needed if you want to delete the cluster)
   b. **Key name:** will be needed to access the instance via SSH if needed.
   c. **License key:** add your account license key here.
   d. **Display name**: a hostname for the instance to be created. Will be easier to find it on New Relic if you specify something meaningful here.
   e. **DBPassword, DBRootPassword and DBUsername:** type Password123 on all 3 fields (or whatever password/username you want to use, Must include alphanumerical chars only).
6. Leave all other settings pre-filled (change whatever you feel needed, or ask your instructor if you want some clarification) and click **Next** on the bottom.
7. Don´t change Anything on Step 3 and click **Next.**
8. Review all the information, scroll to the bottom and check the 2 Acknowledgements, and click **Create stack.**
9. This process will take around 10 minutes to finish, you can monitor it from the Cloudformation screen.
10. After all tasks are completed, click on the **Outputs** tab click on the Wordpress link. You will access the Wordpress site startup wizard
11. After a few minutes, check on New Relic if you can see the information from your Wordpress site and the instance on AWS dashboards and Infrastructure.
12. OPTIONAL: to delete your Wordpress deployment, go back to Cloudformation and select the stack you just created. Click on the **Delete** button. **<u>Don´t forget this instance will be charged on your account for the time it will be running.</u>**