ML4DevOps.

Anomaly detection based on cluster resource utilization metrics.

Table of Contents

1. ML for DevOps case studies.

Some of the key examples of applying Machine Learning to DevOps include:

- **Tracking Application Delivery**

The activity data from 'DevOps tools' such as Git, SonarQube, Jira, Ansible and many others provide delivery process visibility. Application of ML on these tools uncovers the anomalies such as long build times, late code check-ins, slow release

rates which result in non-efficient resourcing and slowdowns of the software development process .

- **Application Quality**

ML applications analyze output from testing tools and review QA results and identify test patterns. This understanding of a 'known good release' helps to ensure comprehensive testing on every release and increasing the quality .

- **Securing Application .**

ML is applied to user behavior for learning the user behavior patterns and to help in identifying anomalous patterns, which represents dangerous activity. For example , the access to the code repos, deployment activity, automation, test execution, system provision, and more can highlight user were exercising 'familiar bad patterns' in a rapid pace both intentionally or accidentally.

- **Resource utilization patterns.**

ML for analyzing general patterns including resource utilization, user volumes, etc. and detecting abnormal patterns caused by memory leaks, programmatic errors, malicious activities and race conditions. By analyzing general patterns system can veg trained to detect anomalous patterns and issue alert warnings .

Identified resource utilization patterns can further be used for capacity planning. ML based system can map utilizing patterns to predict the required configuration for a desired level of performance, the percentage of clients can use a brand new feature and infrastructure necessities.

System can further suggest Optimized capacity limits mapped the specific customer requirements profile.

- **Managing Alert Storms**

Application of ML in managing the gigantic flood of alert and enabling filtering to reduce alert storms and "fatique".

2. Demo project. Applying ML for Anomaly detection based on cluster resource utilization metrics.

2.1. Problem statement and definitions.

- **An outlier** is a value of a metric that is different from other values of that same metric when you would expect them all to be similar.
- **An anomaly** is where a measurement doesn't follow historical trends.
- **Anomaly detection** problem for time series is usually formulated as finding outlier data points relative to some standard or usual signal. Examples of anomaly types: unexpected spikes, drops, trend changes and level shifts.

A simplistic approach for finding anomalies is to use thresholding techniques.
For example, If CPU is greater than 80% it's an anomaly. However, some thresholds might not be right for all server nodes and might different during the day for the same server node. Accounting for all possible scenarios with manually created ruleset becomes too complex and not manageable.

Some very effective (and non-machine learning techniques) algorithms do exist to capture these complex criteria, such as dynamic thresholding and "seasonality & trend decomposition". However even those techniques are not able to capture the full array of scenarios.

2.2. Project description.

Current demo project focuses on building ML model for the anomaly detection in the resource's utilization metrics.
The project uses metrics collected over Open Shift cluster deployed on the RedHat OpenShift container platform 3.11 on RHEL 7.

The diagram 1 depicts how the Ml4DevOps application can be integrated with the open Shift cluster monitoring stack.
The Ml4DevOps as multimodule application would provide the fowling services :

- (1) Alert subscriber
- (2) Metrics query collector
- (3) ML to identify anomaly
- (4) Warning and Notification

Demo project focuses on **the module (3)** of that application:
" Applying ML to identify anomaly in the resource utilization metrics.

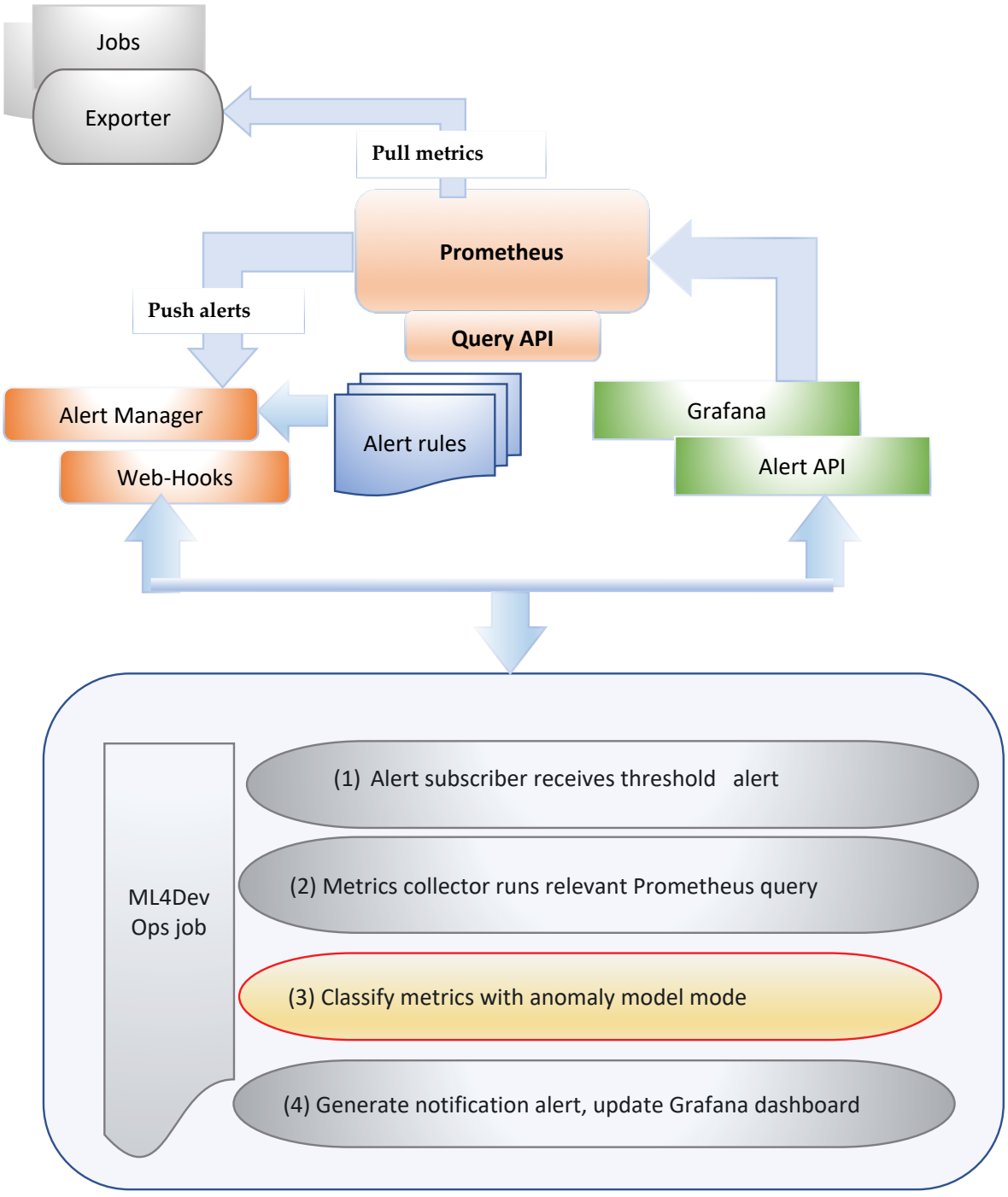

Diagram 1. Integration Ml4Devops application with cluster monitoring stack.

2.3. Environment and tools.

- RedHat OpenShift container platform 3.11 on RHEL 7
- Cluster is created and configured with one master node and one worker node.
- OCP 3.11. Prometheus.
- OCP 3.11. Grafana dashboards
- IBM Watson studio @IBM cloud

2.4. Scope.

- For the purposes of this demo we use simplified dataset.
- We collect resource utilization metrics using two-dimension dataset for the test project namespace.
- We then apply Machine Learning K-mean clustering to train the model.
- We apply this model at real time (or test data) for detecting anomalies.

2.5. Data collection.

For simplicity, we are creating following metrics dataset:

- Metrics Tracking CPU Utilization: "container_cpu_usage_seconds_total"
- Metrics Tracking total HTTP requests: "http_requests_total"

Data are collected for a certain period of time while sample application was deployed and running in the cluster.

Note. For the demo I used Grafana export function for exporting metrics data in the csv format.

2.6. Implementation details.

The problem we are addressing here is simple. It has only two dimensions. Building comprehensive anomaly detection system involves analyzing multidimensional set of the resources utilization metrics such as RAM memory consumption, GPU usage Disk usage and Network usage. Setting a fixed threshold for n-dimension problem can be a bit tricky

Evaluated various anomaly detection algorithms using collected metrics datasets. (ref 6.)

- K-Means can detect patterns in n-dimensional space and to generalize to more complex situation.

- Isolation forests
- OneClass SVM

Following notebooks were created:

Submission_1 notebook project.

- Illustrates anomaly detection by building models using one dimensional dataset with CPU utilization metrics and corresponding timestamp.
- Project categorizes the time data into 4 categories: “weekday Day”, “weekdayNight”, “weekendDay” and “weekendNight”.Project then clusters CPU metrics around identifies categories .
- This model helps to investigate the seasonality factor as impact on CPU utilization over test container when test application usage depends on day/time/weekend /weekday as “seasonal” or temporal factors.

Submission_2 notebook.

- Illustrates anomaly detection by building models using two dimensional dataset with CPU utilization and HTTP requests metrics . These two metrics are synchronized over corresponding timestamps.
- Project categorizes the HTTP requests load into 3 categories: “low” , “moderate” and “high”. Project clusters CPU and HTTP metrics .
- This model helps to investigate pattern of CPU usage correlation with HTTP requests load.

Project Github: https://github.ibm.com/nkochura/https-github.ibm.com-ML4DevOps

2.7. Challenges and next steps.

To complete application implementation according to the Diagram1 require the following steps:

- Create and deploy application docker container in the Cluster, expose the analytics results through the service endpoints.
- Store analytics data patterns results in the cluster persistent storage.
- Integrate with Cluster Prometheus Alert manager.

Challenges:

- Differentiating between normal and abnormal effectively.
- Distinguishing between noise and anomaly.

That to overcome the challenges would require extensive evaluation of various collected metrics datasets and identifying most prominent and typical use cases. Building domain knowledge should progress naturally starting with simple use cases simile to the ones we described in our submission.

Resources and links.

1. https://prometheus.io/
2. http://stanford.edu/~cpiech/cs221/handouts/kmeans.html
3. https://arxiv.org/pdf/1811.06901.pdf
4. https://www.ibm.com/cloud/watson-studio
5. AlertManagerAPI http://petstore.swagger.io/?url=https://raw.githubusercontent.com/prometheus/alertmanager/master/api/v2/openapi.yaml
6. Novelty and outlier detection: https://scikit-learn.org/stable/modules/outlier_detection.html