Anomaly Detection in CI Jobs

https://etherpad.openstack.org/p/wadci

tdecacqu@redhat.com

2018-03-08

2018-05-22

Anomaly Detection in CI Jobs

Anomaly Detection in CI Jobs
https://etherpad.openstack.org/p/wadci
tdecacqu@redhat.com
2018-03-08

Outline

1. Introduction
2. Learning Machine
3. Introducing log-classify
4. Integration in CI Workflow
5. Conclusion

Outline
1. Introduction
2. Learning Machine
3. Introducing log-classify
4. Integration in CI Workflow
5. Conclusion

Notes:

- Slides and materials are available at `https://github.com/TristanCacqueray/opendevconf` Please clone and execute the start.sh script now to cache the dataset.
- The goal of this workshop is to present a new anomaly extraction workflow for CI job results.
- We will see how machine learning methods can be used to compare job results and detect anomalies.
- Then, we will learn how to use the log-classify tool.
- Lastly, we will see how this workflow can be integrated with CI systems.

Topic

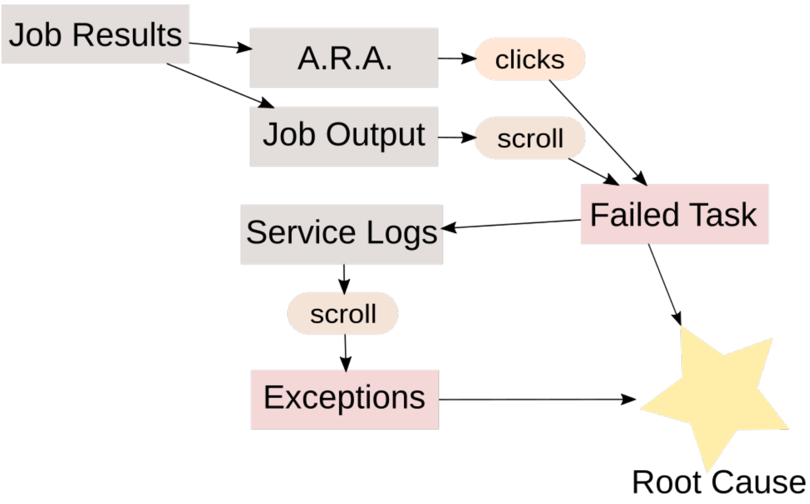
Topic

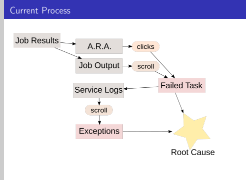
1 Introduction
2 Learning Machine
3 Introducing log-classify
4 Integration in CI Workflow
5 Conclusion

Notes:

- This section introduces the goal of anomaly detection in CI logs.

Current Process



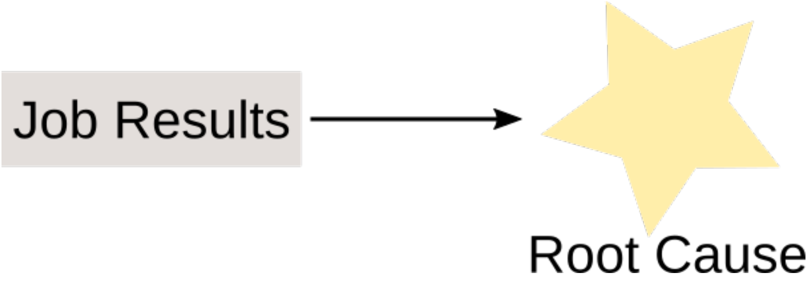


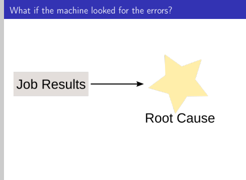
Notes:

- This diagram shows the current actions a developper usually does to understand why a job failed.
- This process is tedious and time consuming and usually involves lots of clicking and scrolling...

Demo:

- Find a random change with a failed job.
- Demonstrate A.R.A.
- Try to figure out why it failed

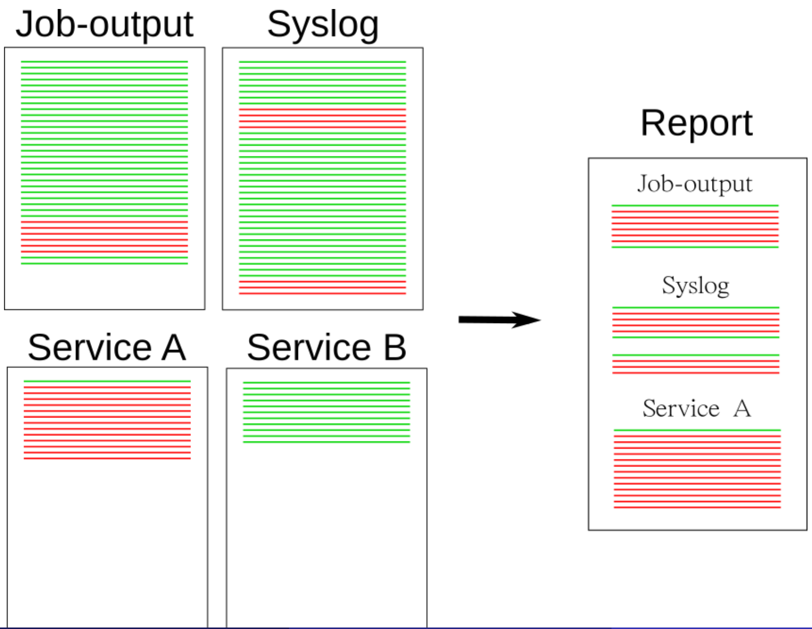


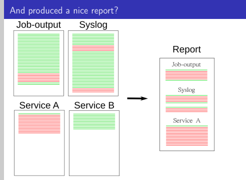


Notes:

- Most of this process can be automated.
- Automatic anomaly detection may greatly reduces investigation time.

And produced a nice report?





Notes:

- Moreover, the machine can produce a nice report.
- Anomalies can be spread accross multiple log files.
- Only a small fraction of the log files are useful to understand a failure.

Base Principle

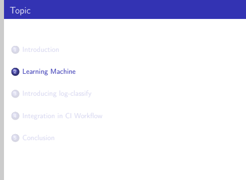
- Baseline: previous job logs
- Target: failed job logs
- Anomaly: new lines missing from the baseline

Base Principle
- Baseline: previous job logs
- Target: failed job logs
- Anomaly: new lines missing from the baseline

Notes:

- Anomalies are defined as novelties from previous runs.
- Thus, comparing a failed job with a successful job usually yields anomalies.
- Next, we will see how machine learning methods can be applied to this challenge.

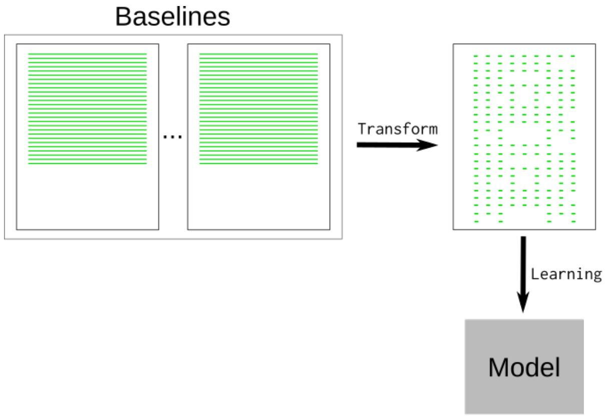
Topic

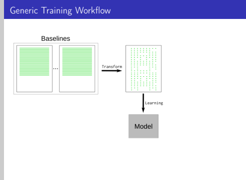


Notes:

- This section introduces two objects that can be used with CI logs:
  - the HashingVectorizer processor; and
  - the NearestNeighbor model.
- Note that other models may easily be used while keeping the same generic workflow.

Generic Training Workflow

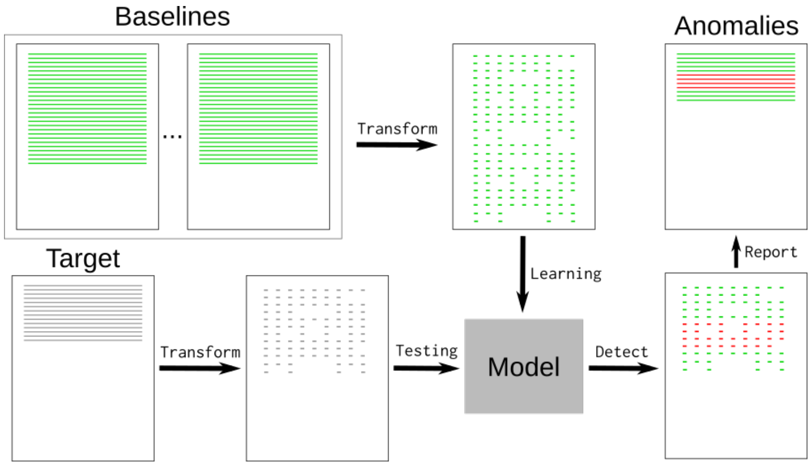


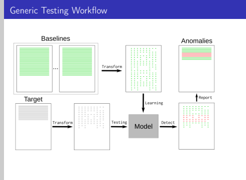


Notes:

- This diagram shows how baselines are processed to train a model.
- The raw text lines need to be converted before being used by a machine learning model.

Generic Testing Workflow





Notes:

- After the model is trained, we can repeat the same process to test the target and extract the novelties.

Hashing Vectorizer

Mar 11 02:43:28 localhost sudo[5195]: pam_unix(sudo:session): session opened for user root by (uid=5)

↓ *tokenization*

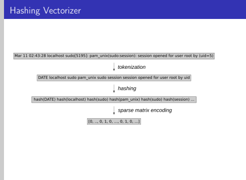
DATE localhost sudo pam_unix sudo session session opened for user root by uid

↓ *hashing*

hash(DATE) hash(localhost) hash(sudo) hash(pam_unix) hash(sudo) hash(session) ...

↓ *sparse matrix encoding*

[0, .., 0, 1, 0, ..., 0, 1, 0, ...]



Notes:

- The first step of the workflow is to transform raw log lines into something more convenient for machines.
- The raw data can't be used because it's noisy: it contains random parts that would yield false positives.
- Let's use simple tokenization and a hashing vectorizer to transform the data.
- The sparse matrix is a numeric array of all possible hashes (2**20 by default).
- Each vector is very sparse as it only contains the token hashes.

Noise Reduction

- Random words may be replaced with known tokens:

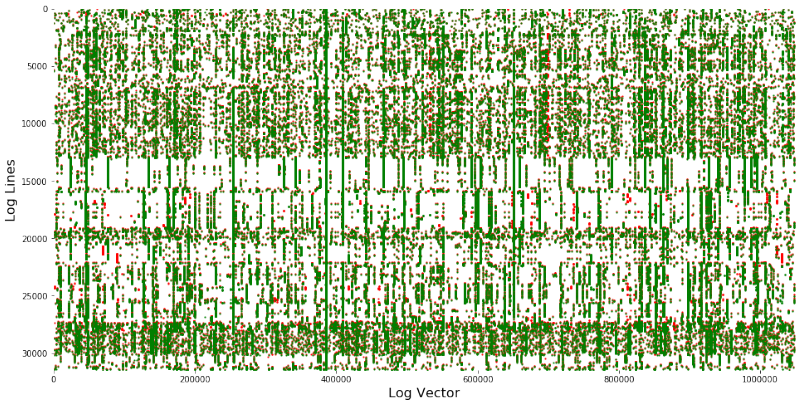
Token	Raw text
DATE	months/days/date
RNGU	uuid
RNGI	ipv4/ipv6/mac
RNGN	words that are 32, 64 or 128 char long
""	all numbers and non letters

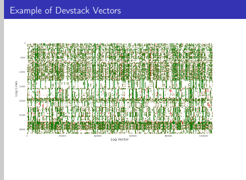
Noise Reduction

- Random words may be replaced with known tokens:

Token	Raw text
DATE	months/days/date
RNGU	uuid
RNGI	ipv4/ipv6/mac
RNGN	words that are 32, 64 or 128 char long
""	all numbers and non letters

Example of Devstack Vectors

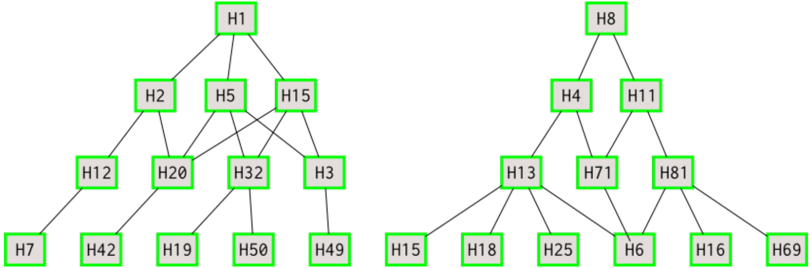


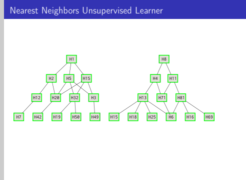


Notes:

- This example shows the vectors of a devstack job-output of 34k lines.
- The green dots show baseline vectors.
- The red dots show target vectors.
- This representation shows all the vectors in order, though we will look for the distances of each target vector to any baseline vectors.
- We can use a learning model to detect the red dots.

Nearest Neighbors Unsupervised Learner





Notes:

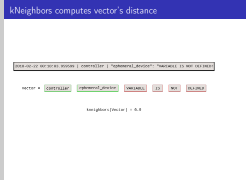
- Nearest Neighbors learns from baseline vectors.
- This builds a tree of connected tokens.
- This doesn't hold the whole dataset.

kNeighbors computes vector's distance

```
2018-02-22 00:18:03.959599 | controller | "ephemeral_device": "VARIABLE IS NOT DEFINED!
```

```
Vector = controller  ephemeral_device  VARIABLE  IS  NOT  DEFINED
```

```
kneighbors(Vector) = 0.9
```

Notes:

- This example illustrates an anomaly from the previous devstack example.
- The Nearest Neighbors model quickly computes the distance of a new vector to the baseline.

Caveats

- Need DEBUG in baseline logs.

Caveats
- Need DEBUG in baseline logs.

Notes:

- This method relies on the fact that the baseline contains all non-anomalous data. Anything that can't be found in the baseline will be reported as anomalous.
- For example, *testr* logs only contains 'SUCCESS' when they succeed, and all the logs are only emited when the job fails.
- The example shows that both lines have the same distance, though we are only interested in the "pcre disabled" one.
- The next section introduces the log-classify tool, an implementation of this method.

Caveats

- Need DEBUG in baseline logs.
- Noise may hide important information:

```
pcre enabled              | pcre disabled
setup mirror hostnameA    | setup mirror hostnameB
```


Caveats
- Need DEBUG in baseline logs.
- Noise may hide important information:
pcre enabled | pcre disabled
setup mirror hostnameA | setup mirror hostnameB

Notes:

- This method relies on the fact that the baseline contains all non-anomalous data. Anything that can't be found in the baseline will be reported as anomalous.
- For example, *testr* logs only contains 'SUCCESS' when they succeed, and all the logs are only emited when the job fails.
- The example shows that both lines have the same distance, though we are only interested in the "pcre disabled" one.
- The next section introduces the log-classify tool, an implementation of this method.

Caveats

- Need DEBUG in baseline logs.
- Noise may hide important information:

```
pcre enabled            | pcre disabled
setup mirror hostnameA  | setup mirror hostnameB
```

- Tokenization may need adjustment for small dataset.

Caveats
- Need DEBUG in baseline logs.
- Noise may hide important information:
pcre enabled | pcre disabled
setup mirror hostnameA | setup mirror hostnameB
- Tokenization may need adjustment for small dataset.

Notes:

- This method relies on the fact that the baseline contains all non-anomalous data. Anything that can't be found in the baseline will be reported as anomalous.
- For example, *testr* logs only contains 'SUCCESS' when they succeed, and all the logs are only emited when the job fails.
- The example shows that both lines have the same distance, though we are only interested in the "pcre disabled" one.
- The next section introduces the log-classify tool, an implementation of this method.

Topic

Notes:

- Note: the project is actually incubated as *logreduce*.
- A *log-classify* projects has been proposed to make this part of the regular openstack-infra or zuul tooling, but the integration details remain to be defined.

Installation

- Use the container image or install using:

```
sudo dnf install -y python3-scikit-learn python3-aiohttp
sudo dnf install -y python3-pip
pip3 install --user logreduce
```


Installation
- Use the container image or install using:
sudo dnf install -y python3-scikit-learn python3-aiohttp
sudo dnf install -y python3-pip
pip3 install --user logreduce

- Output *distance* | *filename:line-number*: anomaly

```
$ pushd 01-files/
$ logreduce diff dib-success.log dib-failure.log
0.250 | dib-failure.log:2258: Package python-setuptools-0.9.8-
                              is obsoleted by python2-setuptoo
```


Compare 2 files
Output distance | filename:line-number: anomaly
$ pushd 01-files/
$ logreduce diff dib-success.log dib-failure.log
0.250 | dib-failure.log:2258: Package python-setuptools-0.9.8-
is obsoleted by python2-setuptoo

Notes:

- The first number tells the distance.
- Logreduce includes some contextual lines, by default 3 lines before and 1 line after the anomaly. This can be changed using *–context-length* command line argument.

Workshop:

- Go to the 01-files dataset.
- Use the logreduce diff command to extract the anomalies from the failure logs.

Compare 2 files

- Output *distance* | *filename:line-number*: anomaly

```
$ pushd 01-files/
$ logreduce diff dib-success.log dib-failure.log
0.250 | dib-failure.log:2258: Package python-setuptools-0.9.8-
                              is obsoleted by python2-setuptoo
```

- Multiple baselines can be used

```
$ logreduce diff audit.log.1 audit.log.2 audit.log
0.614 | audit.log:24516: msg='cwd="/home/centos/logreduce" \
                         cmd="su" terminal=pts/7 res=failed'
```


Compare 2 files
Output distance | filename:line-number: anomaly
$ pushd 01-files/
$ logreduce diff dib-success.log dib-failure.log
0.250 | dib-failure.log:2258: Package python-setuptools-0.9.8-
is obsoleted by python2-setuptoo
Multiple baselines can be used
$ logreduce diff audit.log.1 audit.log.2 audit.log
0.614 | audit.log:24516: msg='cwd="/home/centos/logreduce" \
cmd="su" terminal=pts/7 res=failed'

Notes:

- The first number tells the distance.
- Logreduce includes some contextual lines, by default 3 lines before and 1 line after the anomaly. This can be changed using –*context-length* command line argument.

Workshop:

- Go to the 01-files dataset.
- Use the logreduce diff command to extract the anomalies from the failure logs.

Compare 2 directories

```
$ pushd 02-dirs/
$ logreduce --debug diff success-*/ failure-*/ \
            --html report.html
INFO  Classifier - Training took 84.141 seconds to ingest 33.4
INFO  Classifier - Testing took 173.464 seconds to test 22.952
99.67% reduction (from 128882 lines to 424)
```


Compare 2 directories
$ pushd 02-dirs/
$ logreduce --debug diff success-*/ failure-*/ \
--html report.html
INFO Classifier - Training took 84.141 seconds to ingest 33.4
INFO Classifier - Testing took 173.464 seconds to test 22.952
99.67% reduction (from 128882 lines to 424)

Notes:

- In this second example, we use an html report to better see multiple files.
- A model is built per file. The model name is a minified version of the filename to include variations, e.g. audit.1 and audit.2 use the same model.
- "Loading" and "Testing" debug shows the *model-name*: used for each file.
- Before printing the anomalies, the baseline sources are also displayed, see the *compared with* debug.

Workshop:

- Go to the 02-dirs dataset
- Compare the job-output and notice it's not enough
- Run the diff command on the two directories with an html report logs
- Open the report.html

Model Training

- Model can be trained offline first:

```
$ logreduce dir-train sosreport.clf sosreport-good/ other/
INFO Training took 1.696 seconds to ingest 0.513 MB
$ du --si sosreport.clf
66k     sosreport.clf
```

- To be used later:

```
$ logreduce dir-run sosreport.clf sosreport-customer/
0.000 | ansible.log:0012: TASK [Command with long output]
0.626 | ansible.log:0014: fatal: [localhost]: FAILED!
0.364 | syslog:1576: localhost: System clock wrong by 1.417479
99.62% reduction (from 1595 lines to 2)
```


Model Training
- Model can be trained offline first:
$ logreduce dir-train sosreport.clf sosreport-good/ other/
INFO Training took 1.696 seconds to ingest 0.513 MB
$ du --si sosreport.clf
66k   sosreport.clf
- To be used later:
$ logreduce dir-run sosreport.clf sosreport-customer/
0.000 | ansible.log:0012: TASK [Command with long output]
0.626 | ansible.log:0014: fatal: [localhost]: FAILED!
0.364 | syslog:1576: localhost: System clock wrong by 1.417479
99.62% reduction (from 1595 lines to 2)

Notes:

- The Nearest Neighbor Tree of the sparse matrix is very small compared to the raw data.

Journald

- Extract novelty from the last day:

```
$ logreduce journal --range day
```

- Build a model using last month's logs and look for novelties in the last week:

```
$ logreduce journal-train --range month journald.clf
$ logreduce journal-run   --range week  journald.clf
```


Journald
- Extract novelty from the last day:
$ logreduce journal --range day
- Build a model using last month's logs and look for novelties in the last week:
$ logreduce journal-train --range month journald.clf
$ logreduce journal-run   --range week journald.clf

Notes:

- The journald range sets baseline as the previous day/week/month and the target as the current day/week/month.

Zuul Jobs

- Build a model

```
$ logreduce job-train model.clf
    --job devstack
    --include-path logs/
    --pipeline gate
    --project openstack-dev/devstack
    --zuul-web http://zuul.openstack.org/api
```

- Re-use the model

```
$ logreduce job-run model.clf http://logs.openstack.org/...
```


Zuul Jobs
- Build a model
$ logreduce job-train model.clf --job devstack --include-path logs/ --pipeline gate --project openstack-dev/devstack --zuul-web http://zuul.openstack.org/api
- Re-use the model
$ logreduce job-run model.clf http://logs.openstack.org/...

Notes:

- *–pipeline* can be used to restrict baseline discovery to a specific pipeline
- *–project* can be used to restrict baseline discovery to a specific project. For example tox-py35 jobs likely need to be trained per project.
- *–count* specifies the number of previous jobs to use as training data.
- *–include-path* tells logreduce to fetch job artifacts in the logs/ directory.

DEMO:

- pick a job that failed and run "log log_{url}" command.

Zuul Jobs

- Build a model

```
$ logreduce job-train model.clf
    --job devstack
    --include-path logs/
    --pipeline gate
    --project openstack-dev/devstack
    --zuul-web http://zuul.openstack.org/api
```

- Re-use the model

```
$ logreduce job-run model.clf http://logs.openstack.org/...
```

- Extract anomalies from a job result:

```
$ logreduce job http://logs.openstack.org/...
```


Zuul Jobs
Build a model
$ logreduce job-train model.clf --job devstack --include-path logs/ --pipeline gate --project openstack-dev/devstack --zuul-web http://zuul.openstack.org/api
Re-use the model
$ logreduce job-run model.clf http://logs.openstack.org/...
Extract anomalies from a job result:
$ logreduce job http://logs.openstack.org/...

Notes:

- *–pipeline* can be used to restrict baseline discovery to a specific pipeline
- *–project* can be used to restrict baseline discovery to a specific project. For example tox-py35 jobs likely need to be trained per project.
- *–count* specifies the number of previous jobs to use as training data.
- *–include-path* tells logreduce to fetch job artifacts in the logs/ directory.

DEMO:

- pick a job that failed and run "log $\$log_{url}$" command.

Zuul Jobs Example: tempest-full

- Model trained with:

```
$ logreduce job-train tempest.clf
      --job tempest-full
      --include-path controller/
      --count 3
      --zuul-web http://zuul.openstack.org/api
```

- Usage:

```
$ pushd 03-jobs/
$ logreduce job-run _models/tempest.clf $log_url
      --include-path controller/
```


Zuul Jobs Example: tempest-full
- Model trained with:
$ logreduce job-train tempest.clf --job tempest-full --include-path controller/ --count 3 --zuul-web http://zuul.openstack.org/api
- Usage:
$ pushd 03-jobs/
$ logreduce job-run _models/tempest.clf $log_url --include-path controller/

Notes:

- We are going to use the tempest-full job as a case study.

Workshop:

- Go to the 03-jobs dataset.
- If there is enough time, attendees can build a model for another job. Otherwise, run the model with the pre-loaded logs.

Command line interface summary

- Supports directories, journald and Zuul jobs.
- Model can be trained *dir-train*, *jounal-train* and *job-train*.
- Model can be re-used: *dir-run*, *journal-run* and *job-run*.
- Or all in one command: *dir*, *journal* and *job*.

Command line interface summary
- Supports directories, journald and Zuul jobs.
- Model can be trained *dir-train*, *jounal-train* and *job-train*.
- Model can be re-used: *dir-run*, *journal-run* and *job-run*.
- Or all in one command: *dir*, *journal* and *job*.

Notes:

- Next section introduces integration in Zuul CI workflows.

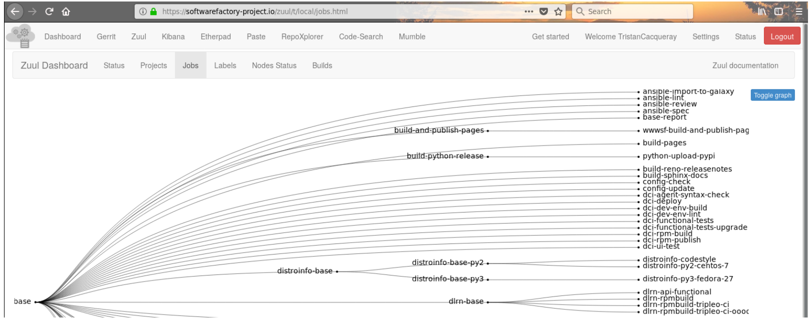
Topic

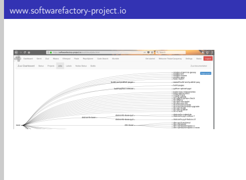
Topic
1. Introduction
2. Learning Machine
3. Introducing log-classify
4. Integration in CI Workflow
5. Conclusion

Notes:

- Using the tool manually may be cumbersome.
- We will now see different ways to integrate anomaly detection in a CI workflow.

www.softwarefactory-project.io

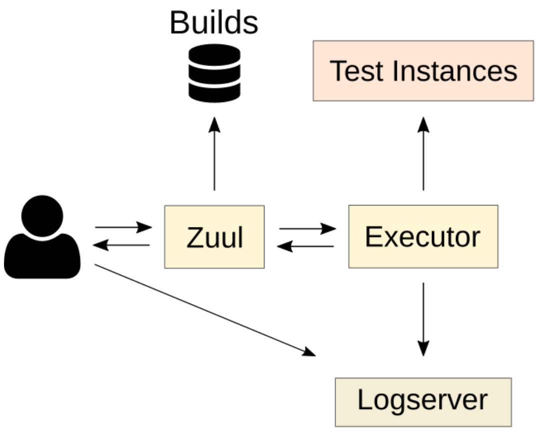


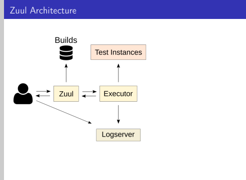


Notes:

- Logreduce has been created in the context of Software Factory.
- It is a development forge that integrates many component to be easily deployed on premise or as a service.
- The architecture is modular and the screenshot shows some of the ready-to-use components.
- Logreduce is being used to analyze sf-ci logs.

Zuul Architecture

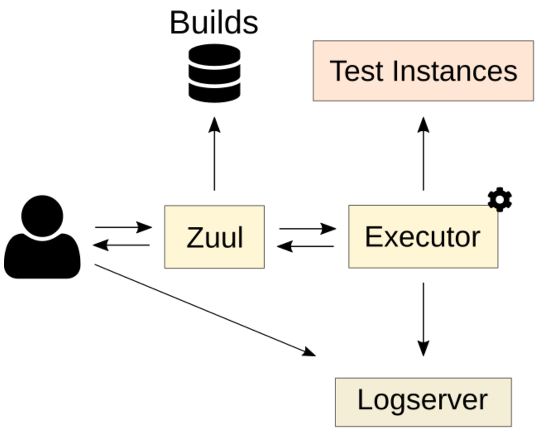


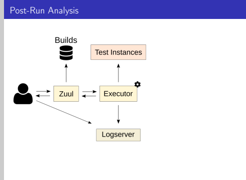


Notes:

- This diagram shows the basic zuul workflow.
- Jobs are executed on ephemeral test instances.
- The executor retrieves the logs and publishes them to a logserver.
- Zuul returns the logserver url to the user.
- Zuul stores build information in a database. This is the key component to make the log-classify process possible.

Post-Run Analysis





Notes:

- This diagram shows the log-classify process running on the executor node.
- Pros: users/job doesn't have to be adapted, the post-run can be added to the base job.
- Cons: memory/cpu overhead on shared resources.

Post-Run Playbook

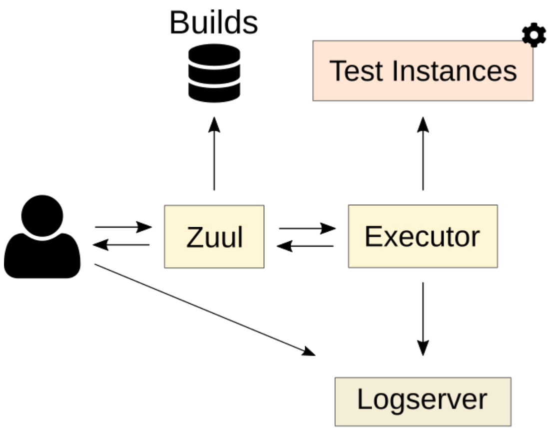
```
- job:
    name: base
    post-run:
      - upload-log
      - clasify-log

- tasks:
  - name: Fetch or build the model
    command: log-classify job-build ...
  - name: Generate report
    command: log-classify job-run ...
  - name: Return report url
    zuul_return: {zuul: url: log: ...}
  - name: Upload model
    synchronize: ...
```

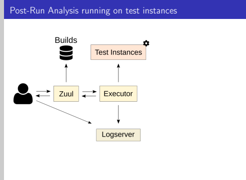

2018-05-22

Anomaly Detection in CI Jobs
└Integration in CI Workflow
└Post-Run Playbook

Post-Run Playbook
- job:
    name: base
    post-run:
      - upload-log
      - clasify-log
- tasks:
  - name: Fetch or build the model
    command: log-classify job-build ...
  - name: Generate report
    command: log-classify job-run ...
  - name: Return report url
    zuul_return: {zuul: url: log: ...}
  - name: Upload model
    synchronize: ...

Post-Run Analysis running on test instances

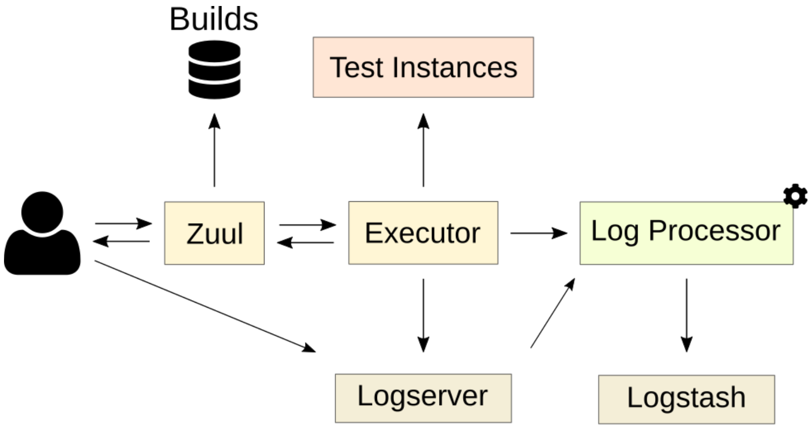


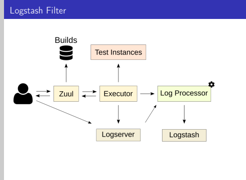


Notes:

- The same playbook could run on the test instance.
- Pros: doesn't cause memory/cpu overhead on shared resources.
- Cons: test instances need the tooling pre-installed.

Logstash Filter

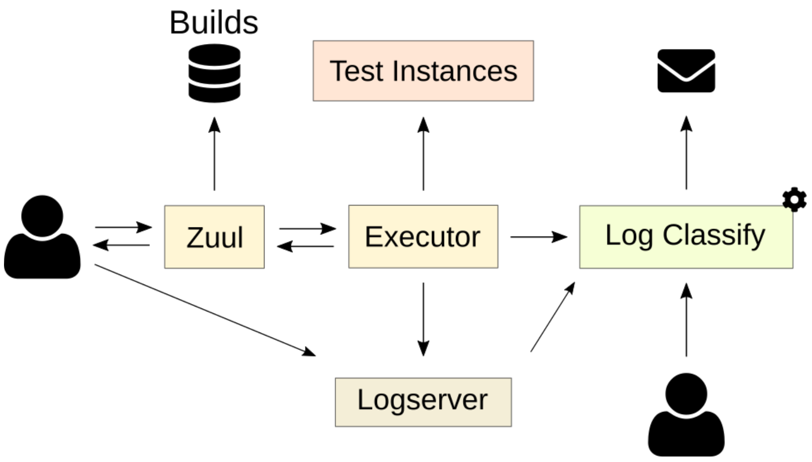


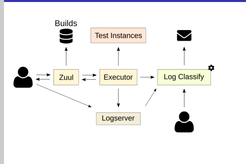


Notes:

- This diagram shows a more advanced Zuul workflow including a log-processor.
- The log-classify could be used as a library to add distance values to logstash events.
- Cons: the users need to wait and go to Kibana to get the report.

Standalone Service





Notes:

- The log-processor could be adapted as a standalone service (TBD).
- Could interface with elastic-recheck.
- This would enable user interaction, for example:
  - Trigger manual analysis
  - Feedback false-positive
  - ...

DEMO:

- Show a softwarefactory-project.io sf-ci job report.

Topic

Topic

1 Introduction
2 Learning Machine
3 Introducing log-classify
4 Integration in CI Workflow
5 Conclusion

Notes:

- And this concludes the workshop.

Credits

- Roadmap:
  - Bootstrap community project.

Credits
- Roadmap:
  - Bootstrap community project.

Notes:

- Thank you for your time!

Credits

- Roadmap:
  - Bootstrap community project.

  - Better supports more jobs.
  - Interface with elastic-recheck.
  - Integrate in openstack-infra.

Credits
- Roadmap:
  - Bootstrap community project.
  - Better supports more jobs.
  - Interface with elastic-recheck.
  - Integrate in openstack-infra.

Notes:

- Thank you for your time!

Credits

- Roadmap:
  - Bootstrap community project.
  - Better supports more jobs.
  - Interface with elastic-recheck.
  - Integrate in openstack-infra.

- Icons used in diagrams are licensed under Creative Commons Attribution 3.0: `https://fontawesome.com/license`

Credits
Roadmap:
Bootstrap community project.
Better supports more jobs.
Interface with elastic-recheck.
Integrate in openstack-infra.
Icons used in diagrams are licensed under Creative Commons Attribution 3.0: https://fontawesome.com/license

Notes:

- Thank you for your time!