



# NXCALS - big data logging system- deployment & monitoring

Wiktor Jurasz



# Agenda

- **Deployment**
  - Git + Gitlab
  - Gradle
  - Ansible
  - Jenkins
  - SonarQube
  - OpenStack
  - Nxcals deployment flow
- **Monitoring**
  - Monit
  - Prometheus
  - Alertmanager
  - Grok
  - ElasticSearch
  - Filebeat + Logstash
  - Grafana



# Git + Gitlab

- Used as:
  - DVCS for project code management




# Git + Gitlab

nxcalls ▾ This project Search 🔍

**Project** Repository Registry Issues 0 Merge Requests 12 Pipelines Wiki Members

[Home](#) [Activity](#) [Cycle Analytics](#)

  
**nxcals** 🔒

BE-CO-DS new Logging system (NXCALLS)


☆ Star 4    🍴 Fork 1    KRBS ▾    <https://gitlab.cern.ch:8443>    📄    👤    + ▾    🔔 Global ▾

---

Files (9.2 MB)    Commits (1,190)    Branches (38)    Tags (17)

develop ▾    nxcalls / +

History 🔍 Find file 📄 ▾

 **JENKINS: Changing version after PRO promotion from 0.1.22 to 0.1.23-SNAPSHOT** 35e66893 📄  
Acc Logging Handler committed a week ago

Name	Last commit	Last Update
📁 accsoft-nxcals-ansible	NXCALS-1089 Changed Kafka retention time f...	a week ago
📁 accsoft-nxcals-client	NXCALS-971 Regexp expressions to search for...	a week ago
📁 accsoft-nxcals-client-demo	Removed leftovers from cmmnbuild	6 months ago
📁 accsoft-nxcals-client-inttest	Removed leftovers from cmmnbuild	6 months ago
📁 accsoft-nxcals-common	NXCALS-971 Regexp expressions to search for...	a week ago







# Git + Gitlab

- Used as:
  - DVCS for project code management
  - Code review tool








# Git + Gitlab

```
20 + @Query("SELECT u from User u JOIN u.realm r WHERE u.userName = :userName AND r.realmName = :realmName")
21 + User findUserWithRealm(@Param("userName") String userName, @Param("realmName") String realmName);
```

 **gavgitid** @gavgitid commented a week ago Developer   


Could this be translated to pure spring data query with keywords? ex. findByUsernameAndRealm

Also, why do we need both args for that search? It there a use case where a user has more than one realm or vice-versa?

 **Wiktor Jan Jurasz** @wjurasz commented a week ago Developer    

Unfortunately no <https://jira.spring.io/browse/DATAJPA-198> This issue is still open, so as far as I know you cannot perform join in method name. But please correct me if I'm wrong.

About the second question, it is possible that user with the same username (not necessarily the same user) exists in different realms. Joe@CERN.CH and Joe@OTHER.COM are two different users with different set of roles in system.



```
22 +
23 + }
```

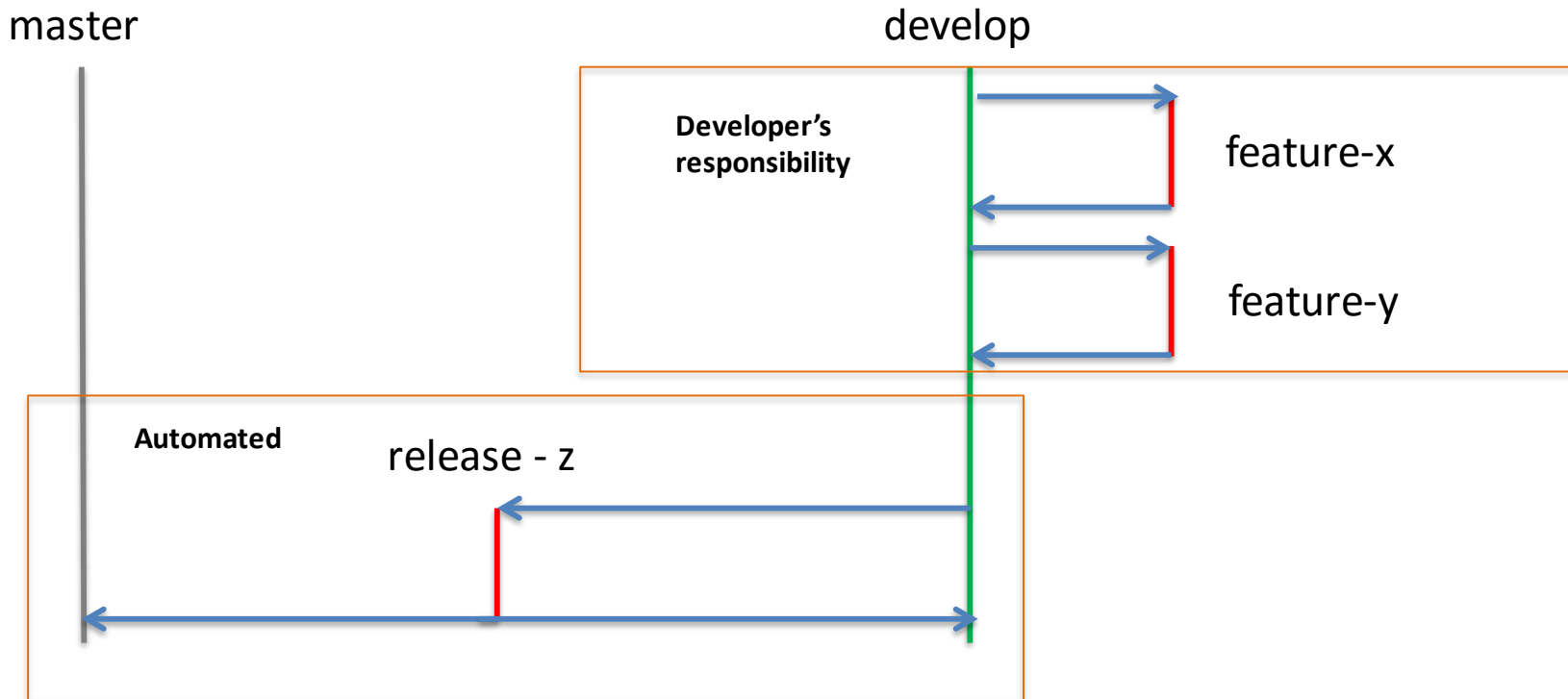


# Git + Gitlab

- Used as:
  - DVCS for project code management
  - Code review tool
- Generale positive impression
  - With big merge requests web interface works slowly



# Branching model







# Gradle

- Builds Java, Javascript (npm) & Python (PyGradle)
- Using *./gradlew* wrapper
  - to configure the gradle & build
  - don't need gradle installed upfront
- Integrates with Idea IntelliJ



# Ansible

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.



# Ansible

- Where to put project configuration so it stays agnostics from environment?
- How to provision and deploy with minimal developers effort?
  - How to do it in secure way?
  - How to do it on multiple machines on multiple environments?
  - How to make it idempotent?



# Ansible

## hosts.yml

```
[prometheus]
cs-ccr-nxcalsstr3.cern.ch

[producers]
cs-ccr-nxcalsstr[1:3].cern.ch

[kafkas]
cs-ccr-nxcalsstr[1:3].cern.ch

[zookeepers]
cs-ccr-nxcalsstr[1:3].cern.ch

[services]
cs-ccr-nxcalsstr[1:3].cern.ch

[compactors]
cs-ccr-nxcalsstr[1:3].cern.ch

[monitoring-readers]
cs-ccr-nxcalsstr[1:3].cern.ch

[etls]
cs-ccr-nxcalsstr1.cern.ch

[test]
cs-ccr-nxcalsstr1.cern.ch
|

[logstashers]
cs-ccr-nxcalsstr3.cern.ch
```

## vars.yml

```
nxcals_namespace: nxcals_dev_wjurasz
oom_notification_recipients: wikt0r.jan.jurasz@cern.ch

#Artifactory
#####
artifactory_address: file:///afs/cern.ch/user/w/wjurasz/.m2
repository: repository

#Hadoop
#####
hadoop_cluster_name: dev
hdfs_namenode: hdfs://hdpdev01.cern.ch/
hbase_zookeeper_quorum: hdpdev01.cern.ch

#Wikis
#####
wiki_page_id: 94550578

#Monit
#####

#Zookeeper
#####
zookeeper_memory: -Xmx512m -Xms128m

#Kafka
#####
kafka_log_retention_hours: 4
kafka_memory: -Xms1g -Xmx2g

#NXCALS - Etl
#####
etl_systems: ['MOCK-SYSTEM']
etl_instances: [
  {
    name: hdfs,
    stop_timeout: 120,
```



# Ansible

## install-java.yml

```
- name: copy java {{java_version}}
  copy:
    src: '{{download_dir}}/jdk-{{java_version}}-linux-x64.rpm'
    dest: '/tmp/jdk-{{java_version}}-linux-x64.rpm'

- name: install jdk-{{java_version}} package
  yum:
    name: '/tmp/jdk-{{java_version}}-linux-x64.rpm'
    state: present
    update_cache: true

- name: create symlink to current java version
  file:
    src: '/usr/java/jdk{{java_full_version}}'
    dest: '/usr/java/jdk'
    state: link

- name: cleanup
  file:
    path: '/tmp/jdk-{{java_version}}-linux-x64.rpm'
    state: absent
```



# Ansible

## playbook.yml

```
- hosts: services
  # run rolling update one by one on each hosts
  serial: 1
  vars:
    http_proxy: '{{proxy}}'
    https_proxy: '{{proxy}}'
    module: 'accsoft-nxcals-service'
    db_module: 'accsoft-nxcals-service-db'
    migration_db_module: 'accsoft-nxcals-migration-db'
    module_group: service
    version: '{{nxcals_version}}'
    prometheus_port: '{{service_prometheus_port}}'
    process_port: '{{service_port}}'
    start_timeout: 240
    stop_timeout: 120
    monit_checks: ""
    java_opts: '{{service_jvm_opts}} -javaagent:lib/jmx_prometheus_javaagent-{{prometheus_jmx_exporter_version}}.jar={{promet
  vars_files:
    - '{{inventory_dir}}/vault-encrypted'
    - '{{inventory_dir}}/vars.yml'
  roles:
    - { role: common-stop, wait_timeout: '{{stop_timeout}}' }
    - { role: common-db, module: '{{db_module}}', url: '{{service_db_url}}', user: '{{service_db_user}}',
      password: '{{service_db_password}}', enable_install: '{{service_db_enable_install}}',
      enable_update: '{{service_db_enable_update}}', enable_script: "false",
      script_changelog_file: "" }
    - { role: common-db, module: '{{migration_db_module}}', url: '{{service_db_url}}', user: '{{migration_repo_db_username}}',
      password: '{{migration_repo_db_password}}', enable_install: '{{service_db_enable_install}}',
      enable_update: '{{service_db_enable_update}}', enable_script: "false",
      script_changelog_file: "" }
    - { role: common-deploy, remove_old: true }
    - { role: service }
    - { role: common-startup-scripts, instance: '{{module}}-{{version}}', script_args: }
    - { role: common-monit }
    - { role: common-prometheus-jmx }
    - { role: common-prometheus-jar }
    - { role: monit-reload }
    - { role: common-start, wait_timeout: '{{start_timeout}}' }
```



# Ansible

*ansible-playbook -i /path/to/inventory playbook.yml*



# Jenkins

- Full automation of builds & deployments





# SonarQube

- Continuous Inspection tool.
- SonarQube provides the capability to not only show health of an application but also to highlight issues newly introduced.

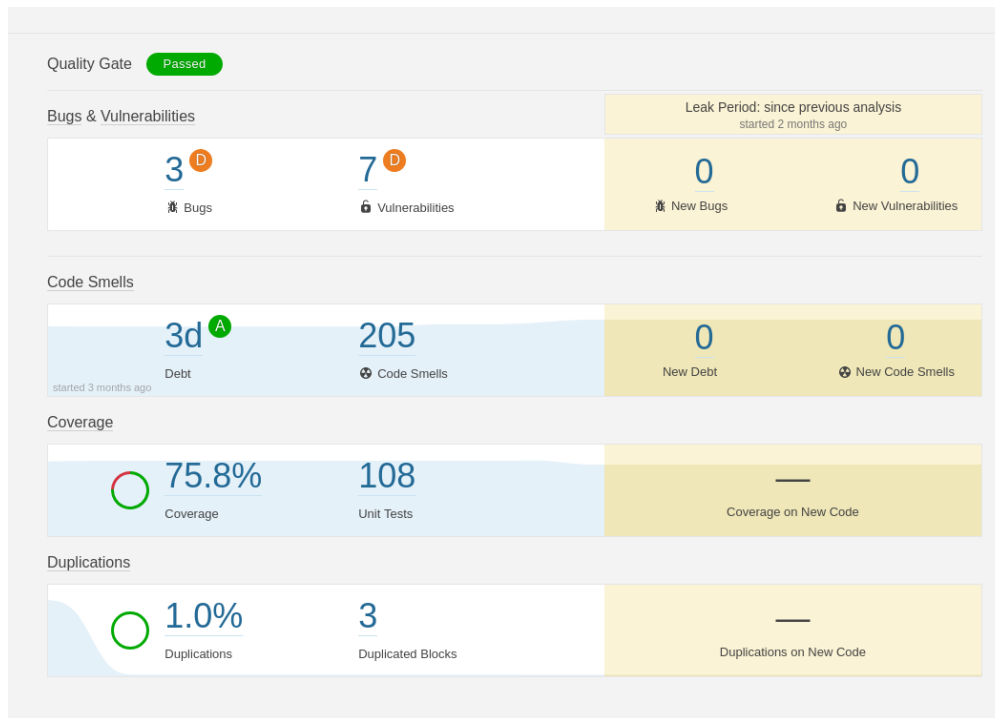


# SonarQube

1. Create Quality Profile out of available rules
  - "equals" methods should be symmetric and work for subclasses
  - Neither "Math.abs" nor negation should be used on numbers that could be "MIN\_VALUE"
  - Public method without javadocs
2. Assign profile to your project (or use default)
3. Create Quality Gate
  - Minimum test coverage
  - Maximum code smells
4. Assign Quality Gate to your project (or use default)
5. Decide when and how to run your SonarQube analysis
  - IntelliJ plugin
  - Gralde plugin
  - Jenkins plugin



# SonarQube





# OpenStack(.cern.ch)

- IaaS
- Very easy to use
- Satisfy the need for multiple environments
  - Each developers own environment
  - Test
  - Staging



# Software Engineering Process

- Create feature branch -> code & test -> create merge request -> merge
- Jenkins + Gradle + Ansible will do the rest



# CI/CD Pipeline Developer's Part

- Code & test
- Deploy to your environment (running ansible script)
- **Integration test & monitoring**
- If every thing is fine create merge request
- Apply sugested changed
- Merge



# CI/CD Pipeline Automation

- Jenkins **TEST** plan picks it up and follows steps:
  - Checks out Git repo (uses **develop** branch)
  - **Builds** the entire system (using Gradle)
  - Runs it via **quality gates** (like tests, coverage, etc)
  - **Puts** the **artifacts** in **ds-development-local** repo (Artifactory plugin)
  - **Deploys** to **TEST Openstack** environment (using Ansible)
  - Does that **continuously** for **every commit** to **warn early** about failures
  - This does **not get released**, just produces **SNAPSHOTs**



# CI/CD Pipeline Automation

- Jenkins **STAGING** plan executes **once per day in the evening**
  - Git checkout from ***develop*** branch
  - Creates a ***release*** branch, removes the **SNAPSHOT** from version
  - Builds, runs the junit & coverage QA gates
  - Deploys to **STAGING** environment using Ansible
  - Waits for compaction 12h (very specific to NXCALS)
  - Executes **Full Integration Tests Suite**
  - If **all fine** asks for **user input** – do we want to **promote** this build to **production**?
  - If yes it **moves** (***promotes***) the build artifacts (jars, zips) between **staging** repository to **production** repository in Artifactory
  - **Merges** the ***release branch*** into ***master branch***
  - **Updates** the SNAPSHOT version in ***develop branch*** to the next one
  - **Deploys** the **pro version** to **PRODUCTION** environment using Ansible





# CI/CD Pipeline Automation

- Remarks:
  - Artifacts are build once and and versions traceable to exact git revisions
  - Developer does not do any releases from local dev machine Jenkins does it better for him/her
  - Requiers clustering & rolling updates



# Monitoring



# Monit

- *Monit is a utility for managing and monitoring processes, programs, files, directories and filesy stems on a Unix system. Monit conducts automatic maintenance and repair and can execute meaningful causal actions in error situations.*
- In NXCALS we use monit to:
  - Start/stop/monitor our processes
  - Send alerts on actions, states, conditions etc.
- Not very reactive when launching stuff in parallel, sometimes causes deployment problems and makes it just longer
  - Waiting for version 6 to fix this issue



# Monit

```
Process 'accsoft-nxcals-monitoring-reader'  
  status          OK  
  monitoring status Monitored  
  monitoring mode  active  
  on reboot        start  
  pid              19516  
  parent pid       19512  
  uid              21020  
  effective uid    21020  
  gid              1077  
  uptime           1h 1m  
  threads          577  
  children         0  
  cpu              4.9%  
  cpu total        4.9%  
  memory           4.6% [2.9 GB]  
  memory total     4.6% [2.9 GB]  
  disk read        0 B/s [126.9 MB total]  
  disk write       0 B/s [83.2 MB total]  
  data collected   Thu, 19 Oct 2017 10:27:12
```



# Prometheus

- Ingests ALL the metrics by Pulling the targets via web end point
- Non-intrusive with provided java JMX exporter
- Metrics in easy, text format
- Metrics have dimensions (“labels”)
- Rule engine with many useful functions
  - *rate(total\_rec\_count{process="ds-lhc"}[1s])*
  - Derived metrics easily defined
- Alert engine based on signals -> Alertmanager



# Prometheus

Prometheus Alerts Graph Status Help

kafka\_server\_BrokerTopicMetrics\_Count

Load time: 54ms  
Resolution: 14s

Execute - insert metric at cursor -

Graph Console

Element	Value
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="TotalProduceRequestsPerSec",topic="CMW"}	2266845
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="MessagesInPerSec",topic="MOCK-SYSTEM"}	59109
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="FailedFetchRequestsPerSec",topic="PM"}	0
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr3.cern.ch:19503",job="kafka",name="TotalFetchRequestsPerSec"}	138632
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="TotalProduceRequestsPerSec",topic="PM"}	0
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr3.cern.ch:19503",job="kafka",name="BytesOutPerSec",topic="MOCK-SYSTEM"}	9856545
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr3.cern.ch:19503",job="kafka",name="BytesOutPerSec",topic="PM"}	0
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="BytesInPerSec",topic="PM"}	0
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="MessagesInPerSec",topic="PM"}	0
kafka_server_BrokerTopicMetrics_Count{instance="cs-ccr-nxcalstr1.cern.ch:19503",job="kafka",name="TotalProduceRequestsPerSec",topic="MOCK-SYSTEM"}	31232



# Prometheus

Prometheus Alerts Graph Status Help

## Alerts

- ErrorIn\_LogFile (1 active)
- InstanceDown (1 active)
- MismatchBetweenHBASEGobblinAndKafka (1 active)
- MismatchBetweenHDFS\_GobblinAndKafka (1 active)
- CompactorErrors (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV18\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV18\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV18\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV1\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV1\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV1\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV2\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV2\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV2\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV3\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV3\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV3\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV4\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV4\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV4\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV5\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV5\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV5\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV6\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV6\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV6\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV7\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV7\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV7\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV8\_byDay (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV8\_byHour (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV8\_byMinute (0 active)
- ErrorInEndToEndMonitoringFor\_NXCALS\_MONITORING\_DEV9\_byDay (0 active)



# Alertmanager

- Receives alerts from Prometheus (or other source)
- Processes them by:
  - Grouping by configurable types
  - Deduplication
  - Throttling (avoid spamming)
  - Sending to predefined “sinks” (email, sms, etc)





# Grok

- Problem: count errors or expose fields from logs (as Prometheus metrics)
  - Publicly available ones can trace only one file at the time
  - Had to write our own using Apache Tailer



# Grok

- A little process that greps the log files and searches for patterns
- Exposes them via JMX
- Than it is easy with JMX Prometheus exporter



# Filebeat + Logstash

- Filebeat scraps logs from all services on the machine and sends them to Logstash
- Logstash uses Elasticsearch as a sink
  - Sending directly from Filebeat to ES was causing communication problems



# Elastic Search

- NXCALS 16 micro-services x 5 machines x 13 envs
- Sending all the logs there using filebeats and logstash
- Browsing the logs from Grafana

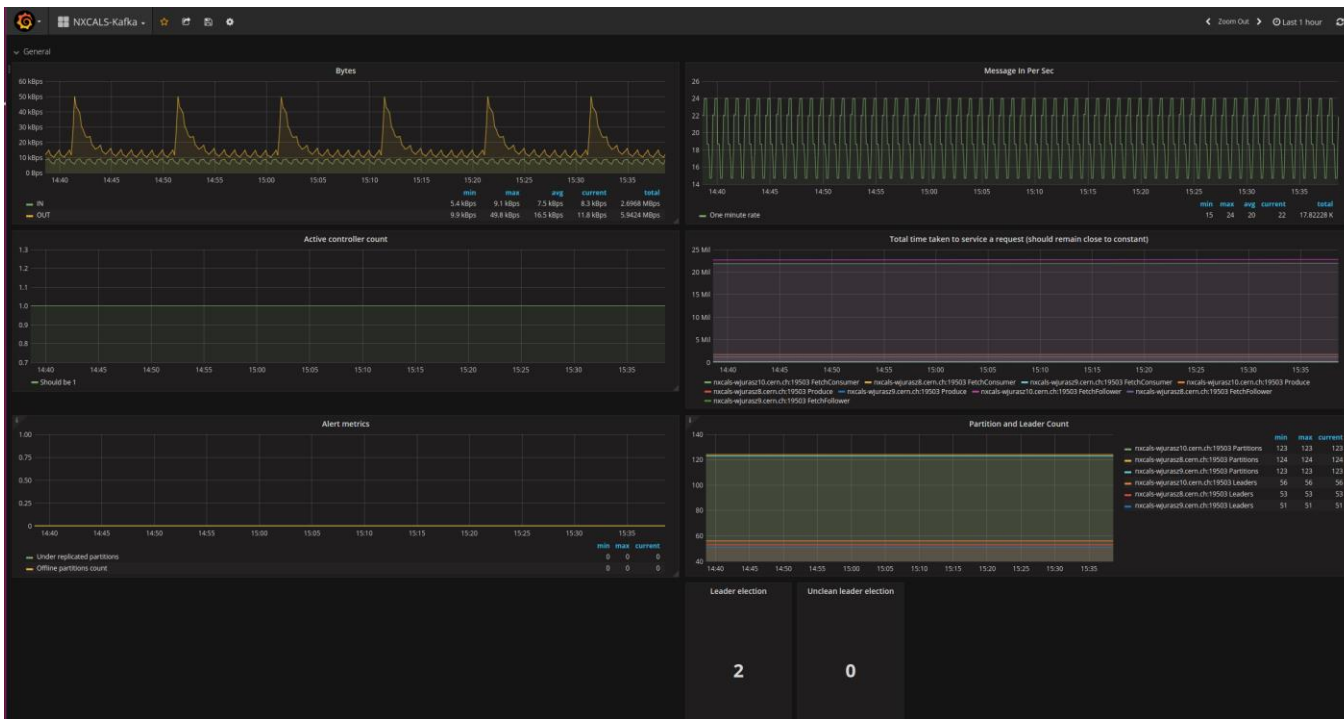


# Grafana

- Dashboard visualization application
- Gets data from:
  - Prometheus
  - ElasticSearch
- Possibility to generate dashboards using Ansible



# Grafana





# Grafana

The screenshot shows the Grafana interface with a table of error logs. The table has columns for @timestamp, message, and host. The logs contain error messages from a monitoring system.

@timestamp	message	host
2017-10-23 14:46:11	2017-10-23 14:46:10.563 [ERROR] [pool-24-thread-1] ProcessorImpl - ERROR: Entity=mock-system_nxcals_monitoring_dev4 check=byHour condition= var data = dataSet.takeAsList(10); dataSet.count() == 3600 startTime=2017-10-23T02:00:00Z endTime=2017-10-23T02:59:59.999999999Z failed with message=Expected 3600 by hour but got: 3568	nxcals-wjurasz8.cern.ch
2017-10-23 14:46:21	2017-10-23 14:46:20.994 [ERROR] [pool-24-thread-1] ProcessorImpl - Errors found for mock-system_nxcals_monitoring_dev4 byhour errorsCount=1	nxcals-wjurasz8.cern.ch
2017-10-23 14:59:04	2017-10-23 14:59:03.653 [ERROR] [pool-24-thread-1] ProcessorImpl - ERROR: Entity=mock-system_nxcals_monitoring_dev10 check=byHour condition= var data = dataSet.takeAsList(10); dataSet.count() == 3600 startTime=2017-10-23T02:00:00Z endTime=2017-10-23T02:59:59.999999999Z failed with message=Expected 3600 by hour but got: 3569	nxcals-wjurasz8.cern.ch
2017-10-23 14:59:14	2017-10-23 14:59:14.515 [ERROR] [pool-24-thread-1] ProcessorImpl - Errors found for mock-system_nxcals_monitoring_dev10 byhour errorsCount=1	nxcals-wjurasz8.cern.ch



**Thank you!**

Questions?