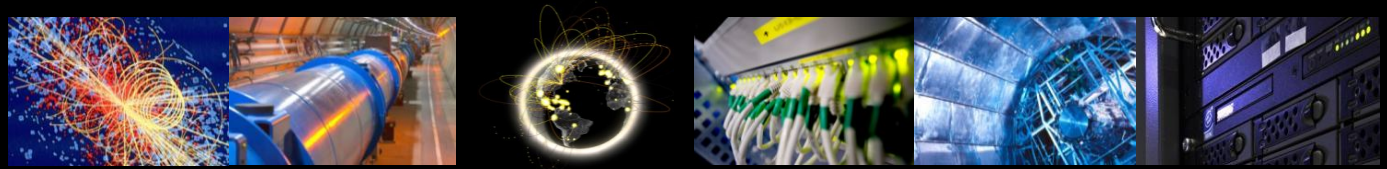
WLCG Experiments Test Framework Update

Marian Babik, CERN

WLCG operations coordination, Dec. 2019



WLCG
Worldwide LHC Computing Grid

Open Science Grid

Motivation

Current Service Availability Monitoring (SAM) structure

- **WLCG Experiments Test Framework (ETF)**
  - WLCG testing middleware (running so called SAM tests)
  - Active testing of the sites services and reporting back to SAM3/MONIT
  - Common to all experiments
  - Main source for WLCG Availability/Reliability Reports (different from EGI/ARGO)
- SAM3/MONIT
  - Aggregation (via custom algos), visualisation and reporting
  - Support for multiple sources of metrics (e.g. ALICE storage tests, ATLAS ASAP)

- A generic test framework remains fundamental for WLCG monitoring
  - Keeping track of sites availability/reliability
  - Running deployment campaigns (IPv6, HTTP, etc.)
  - Provides means of isolation when debugging site/experiments issues
    - Middleware bugs, site setup/configuration, latency sensitive issues/timeouts, etc.
  - Contributing to the operational toolchain of the experiments

Overview

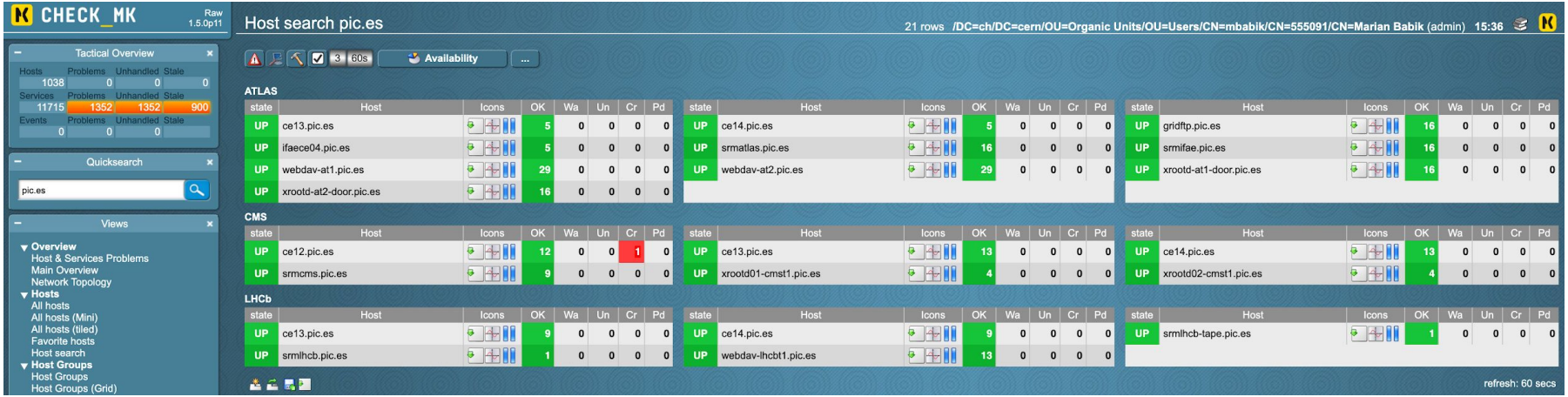
Generic test middleware based on open source

- Checkmk, Nagios core and Messaging (ActiveMQ)

Focuses on functional testing (atomic)

- Direct job submissions, worker node env. testing
- Core storage operations
- Remote API testing and/or network testing (ping/icmp)

~ 150 sites, 1200 hosts monitored
~ 10 metrics/host
~ 1M metrics/day
High-level functional testing
Plugins conforming to Nagios standard
Configurable schedule for test execution
Checkmk dashboard to show results



Architecture

ETF Core Framework

- Frontend API, configuration, scheduling, alerts

Plugins (probes/tests)

- Range of available plugins to test broad range of services
- Contributed by experiments, PTs, TFs and open source projects (Checkmk), etc.
- Python library to help write plugins (**python-nap**)

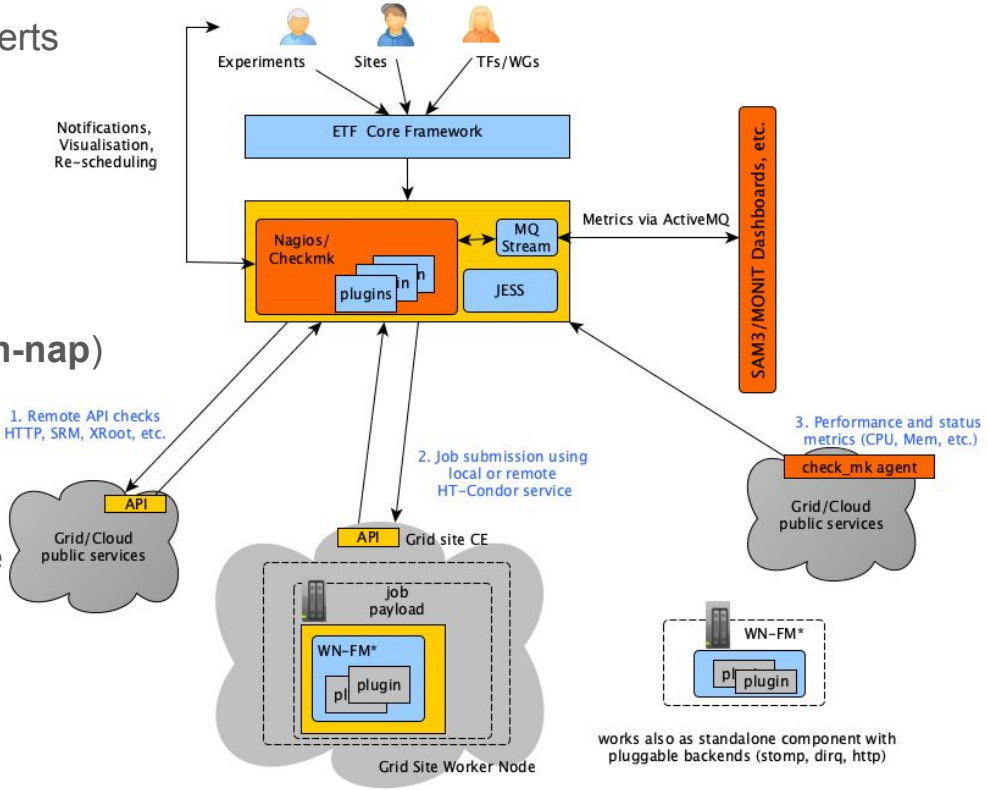
MQ Stream for publishing results

Job Submission Framework (JESS)

- Framework to write job submission plugins (submit/manage jobs, retrieve worker node results, etc.)

Worker Node Framework (WN-FM)

- Micro-scheduler to run tests on the WNs (configure and execute WN tests, collect results)

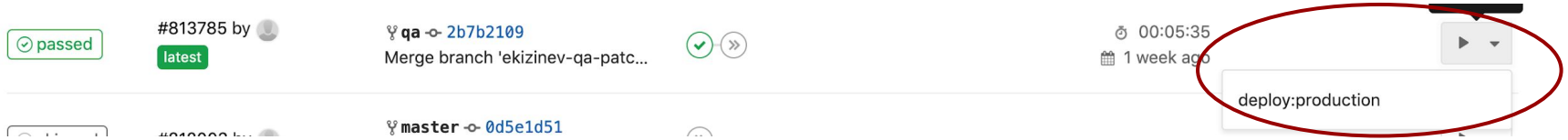


Deployment and Operations

Experiment instances @CERN (IPv4-only/IPv6-only in QA, IPv4-only in PROD)
perfSONAR infrastructure monitoring @OSG

ETF now runs in containers and is integrated with gitlab CI

- Each experiment has its own container/image and gitlab repository
  - Full control over packages and versions to be deployed
- ETF can be deployed in the experiment-specific environment if needed
- Faster development cycle - changes propagated to QA upon each commit
  - Each commit triggers container rebuild and deployment to QA, one-click deploy to prod
- Simplified deployment - auto-deployed directly from gitlab
  - Easy to rollback



Core

ETF core currently running Checkmk 1.4, migration to 1.5 planned for Q1 2020

- Latest version is 1.6 - currently evaluating impact on ETF

ETF frontend - rule-based configuration system (ncgx)

- **Pluggable** - python-based frontend/API for processing experiments topologies (what), tests (how) and schedule (when) (example)
  - Template language to define how/when tests are executed
  - Tests can be configured to have different params per service/host/site, etc.
- Configuration now part of the experiments codebase (gitlab repository)
  - Experiments have full control over each aspect of the configuration (including topology)
  - Gitlab repository - single place for code (images, tests) and configuration

Topology - hosts/service types currently taken from VO feeds

- XML-feed produced by the experiments (docs, intro)
  - Currently main source for hosts, queues, storage paths, etc.
  - While there are no limitations in ETF, for reporting/aggregation it's important to align topology with the aggregation layer (SAM3/MONIT)

Plugins/Tests

Plugins	Users/Experiments	Maintained by
Job Submission		
CREAM, ARC, HTCONDOR-CE *JESS** (next slide)*	LHCb, ALICE, ATLAS, CMS	ETF
Worker Nodes		
ATLAS (3), CMS (11), LHCb (7)	ATLAS, CMS, LHCb	ATLAS, CMS, LHCb
Storage		
GFAL2 (SRM, gsiftp, XRoot, HTTP)	ATLAS	ATLAS
GFAL2 (SRM)	CMS	CMS
XRoot**	CMS	CMS
HTTPs/WebDAV**	HTTP TF*	HTTP TF*
Network		
perfSONAR infrastructure**	WLCG Network Throughput WG	OSG, WLCG

Uses **new library for writing plugins (python-nap) *Probe is still supported by GFAL2 team

Jess

- **Simple job submission framework used to develop ETF JS plugins**
  - Independent (not tied to Nagios), ETF JS plugins combine python-nap and jess
- **Pluggable** - easy to extend to support different submission systems
  - Direct submission to ARC, CREAM and HT-Condor-CE
  - Submissions via local HT-Condor (to ARC/CREAM/HTCondorCE and potentially other backends supported by HT-Condor)
  - **Submissions via remote HT-Condor pool**
  - ETF can also host a **local HT-Condor pool** (in a separate container) to which remote startds can connect (implements pull-based submission model - tested for CMS DODAS)
- Generic **job tracking and monitoring**
  - Currently tracking a single job per CE; can also be extended to track multiple jobs/CE
  - Manual re-scheduling of job submissions via web interface improved
  - Full log(s) of the running job (details depend on the backend)
- **Support for configuration/env on the worker nodes**
  - This can be configured by the experiments in the ETF core plugin (per host/service/site)
- Drops length limit on the worker node results

WN-FM

- **Micro-framework to execute tests on the worker nodes**
  - Currently we're using a statically compiled nagios binary, which has a number of limitations
- Supports basic scheduling of tests
  - Can run tests in parallel (configurable), can timeout/kill runaway tests
  - Initial support for alternate schedule of tests (execute once every 3 runs)
- Runs nagios standard compliant tests
  - **Supports performance metrics readout** (status| perf1, perf2, perf3, perf4 \n test out)
- Configured directly from ETF frontend
- **WN environment setup**
  - Using env/config passed from JESS (via env file) - easy to pass variables directly from frontend to the WN (like sitename, paths, originating CE/queue, etc.)
- Support different backends for metric output
  - Directory queues, message queues, json, http upload, etc.
- **Can also run as a standalone component**

Challenges and Plans

- Py3 compatibility (https://pythonclock.org/)
  - Impacts test dependencies as well, recommended to migrate to python-nap
- Infrastructure keeps evolving, which will impact our current A/R model
  - Resource landscape becoming more complex (HPCs, opport., K8s, Clouds, etc.) -> Experiments workload management systems diverging in the way they use (submit jobs to) the resources
  - Requires follow up if a common testing tool is to be used in the future
  - Two possible options going forward:
    - Experiments help develop the respective backends (and expose APIs to external systems)
    - Experiments start providing their own A/Rs
- Plan is to start rolling out new features gradually during the next year
  - In this order: 1. ALICE, 2. LHCb, ATLAS and 3. CMS
  - We should take the opportunity to also converge on topology (CRIC), queues discovery, etc.
  - Py3 compatibility for WN metrics to be checked
- K8s integration to be tested in QA later on
  - Migrate to full Auto DevOps model (with Helm/Flux) and consolidate resources

Summary

- ETF is a container-based application combining open source software with a set of frameworks and APIs to provide flexible testing suite
- Easy to extend, re-locate and support new experiments and technologies
- Supported as part of the CERN IT Monitoring Stack
- Currently deployed at CERN for all four experiments
  - Supporting IPv4-only and IPv6-only monitoring
  - Experiments contacts have access in case they need to debug and/or follow up on issues
  - Central instance provides a site-level view (one place to see results from all experiments)
- Additional deployment at OSG for perfSONAR infrastructure monitoring
  - Strong interest from other communities to have this available as a generic tool
- Open for additional use cases and ideas
- Feedback welcome via standard support channels or directly
  - Experiments priorities for different tasks and new use cases best communicated via tickets
  - Technical issues/merge requests can be added directly via gitlab

Questions ?

Docs: https://etf.cern.ch/docs/latest/

Central instance: https://etf.cern.ch/etf/check_mk/

Instances (access requires IGTF/x509 cert loaded in the browser):

CMS production	CMS QA IPv6	CMS QA	Code: CMS gitlab
ATLAS production	ATLAS QA IPv6	ATLAS QA	Code: ATLAS gitlab
LHCb production	LHCb QA IPv6	LHCb QA	Code: LHCb gitlab
ALICE production		ALICE QA	Code: ALICE gitlab
pS production		pS QA	Code: pS gitlab

ETF framework

ETF core containers ETF Job Submission (Jess)

ETF nagios plugins lib. NAP ETF rule-based configuration (ncgx)

ETF support channels: GGUS: Grid Monitoring or etf-support@cern.ch (SNOW)