JOINT INSTITUTE FOR NUCLEAR RESEARCH
Laboratory of Information Technologies

FINAL REPORT ON THE SUMMER STUDENT PROGRAM

Subsystems development for automated system of job execution management in distributed heterogeneous computing environment of COMPASS experiment

Supervisor:
Artem Petrosyan

Student:
Daniil Malevanniy, Saint Petersburg State University

Participation period:
July 13 – August 27

Dubna, 2017

Abstract

Modern High Energy Physics experiments generates large amount of raw data that must be analyzed. For that analysis big computational power is needed. This process requires great computing power, which is available as distributed heterogeneous resources. Different workload management systems (WMS) are used to assign jobs to distributed computing nodes and manage their execution and I/O data transfer. WMS that used in COMPASS experiment is called "Production and distributed analysis system" (PanDA). It was developed in ATLAS collaboration for workload management in ATLAS experiment and was used in several other experiments since then.

PanDA uses pilot jobs to manage heterogeneous resources and insulate end-users from complexity of computing infrastructure. It also allows to manage queues, users, sites, user access to job queues and map sites to a queues via database tables. It often may be inconvenient for administrators to use database clients to control PanDA. Thus, web interface for PanDA configuration tables can be useful.

At this moment COMPASS experiment production system is used to manage datasets handling in PanDA system. It controls job submission, monitoring, success or failure for each file in dataset. Because of large amount of datasets and file in them, it is dangerous to process all of them at the same time because of system overload risks. To prevent this COMPASS Production System splits datasets to data chunks and uses cron jobs for their automated submission and management. But this also may be inconvenient for administrators to edit crontab manually. So, web interface for Production System periodic tasks can also be useful.

Currently, COMPASS Production System has administration web-interface, implemented through Django framework. During this project, interfaces for PanDA configuration tables and Production System periodic tasks management was developed and integrated in current administration interface. Furthermore, new authentication and authorization system based on User SSL Certificates was implemented and integrated into administration interface.

This features extends Production System administration interface capabilities and allows users to use conventional authorization process.

Introduction

Modern High Energy Physics experiments use high sensitive event detectors that generates large amount of raw data for each event[1,2]. This data must be analysed as quick as possible to provide useful results. Thus, HEP experiment require great calculation power. Supercomputers was used for this purpose earlier, but now, because of the surge in communication capacity of global network, it is more effective to use distributed computing clusters or GRID[3].

Assigning analysis jobs to cluster computing nodes is complicated process that depends on many parameters like input data location, network status, authorization policy and others. This process is controlled by Workload Management Systems (WMS). COMPASS experiment uses "Production ANd Distributed Analysis system (PanDA)" WMS[2].

PanDA

PanDA was developed in 2005-2007 in ATLAS collaboration in order to obtain stable and scalable workload management. PanDA reached both of those goals by showing an excellent performance when working with data generated by the ATLAS detector during the LHC initial run from 2007 to 2013 and coping with the load increase from 10 thousand to over a million jobs per day. The system received a major upgrade during the collider hiatus of 2013–2014. PanDA was adapted to use MySQL as a backend, monitoring and brocker systems were improved and developers put ATLAS-specific system components into removable modules, enabling the use of PanDA in other physics experiments[4].

PanDA uses Pilot jobs to encapsulate all the different infrastructure features of computing clusters and provide single end-user interface and authorization method. Main component of PanDA is PandaServer. It handles request from clients and connection with pilots on the computing nodes, makes all decisions about jobs assigning and trace jobs' state[3]. There is other components like monitoring web-interface and different clients for job submission. However, all of those are optional and only PandaServer is required for PanDA WMS to work.

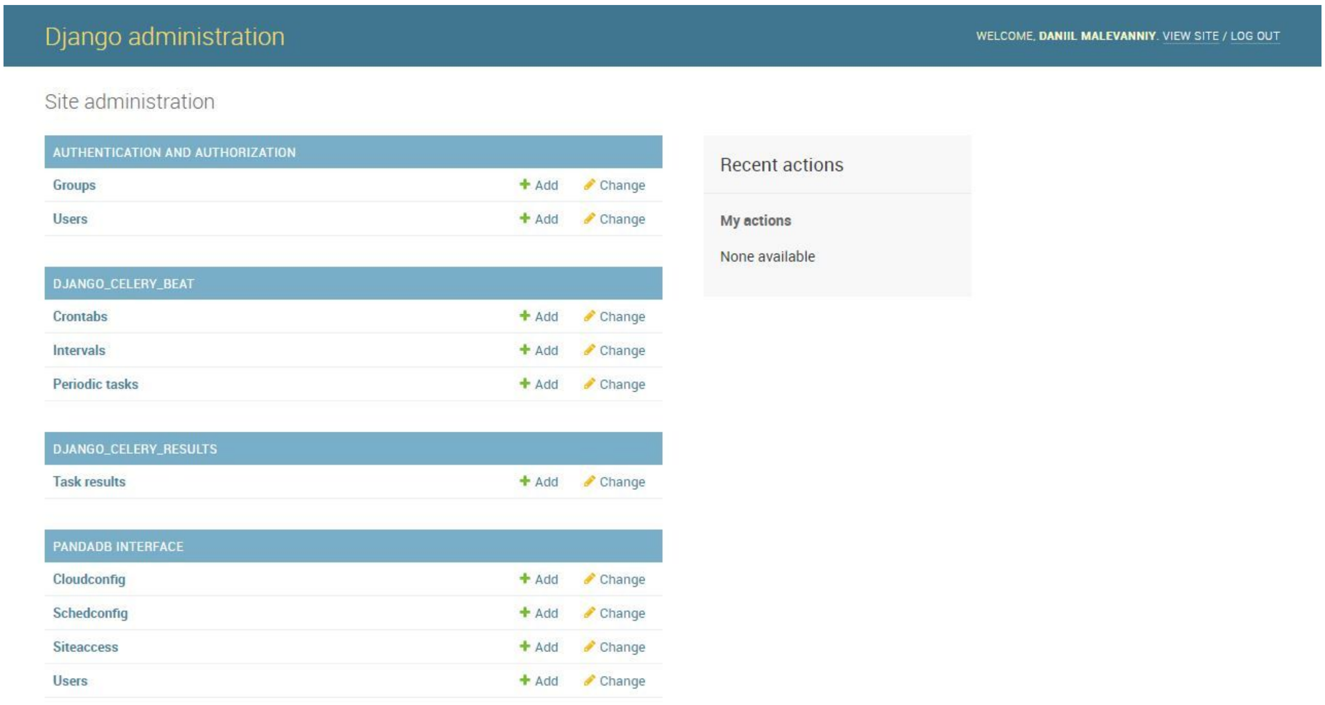
PandaServer job brokerage and authorization policy in runtime is configured with several database tables. Four main configuration tables are `schedconfig`, `cloudconfig`, `siteaccess` and `users`. However, there is not any interface for this tables, so only way to edit job brokerage and authorization policy configurations is database clients.

COMPASS Production System

PanDA WMS is used in COMPASS experiment Production System. Data obtained from event detectors are grouped in datasets. Production task for dataset consists of actual processing jobs submitted to PanDA for each file in dataset and result merging job, if needed[2]. Thus, big amount of jobs must be submitted to PanDA server and some of them can reach timeout before execution. To prevent this, Production System submits jobs in groups, dynamically calculating number of jobs for each group. Currently cron tasks is used to submit job groups to PandaServer in equal periods of time, keep track for their state, handle exceptions and merge jobs results for jobs in one task into single file. However, only way to configure Production System cron tasks is to edit crontab. It can be inconvenient.

This project was dedicated to development of new administration web interfaces for COMPASS experiment Production System that will simplify system control. Production System already already had web interface for task monitoring, implemented with Django framework. New developments must be integrated into existing infrastructure, and because of that all of them have to be implemented with Django framework, using SSL certificate-based user authentication and Django-compatible user authorization.

Developed tools overview


Django administration

WELCOME, DANIIL MALEVANNIY. VIEW SITE / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION
- Groups — Add — Change
- Users — Add — Change

DJANGO_CELERY_BEAT
- Crontabs — Add — Change
- Intervals — Add — Change
- Periodic tasks — Add — Change

DJANGO_CELERY_RESULTS
- Task results — Add — Change

PANDADB INTERFACE
- Cloudconfig — Add — Change
- Schedconfig — Add — Change
- Siteaccess — Add — Change
- Users — Add — Change

Recent actions

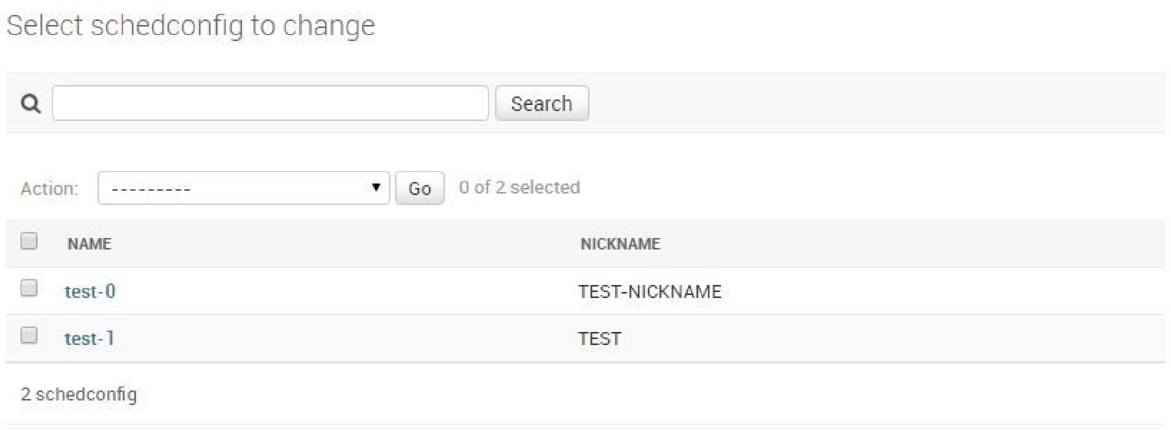
My actions

None available

Administration interface home page.

Configuration tables interface

This interface is implemented as part of Django built-in admin site. It allows to edit configuration tables, controls required fields and data validity, so PanDA authorization and broker policies is fully configurable through the web interface. It also provides search tool for tables. Since new interface implemented only with Django tools, it can be easily expanded to meet new demands using Django features, like bulk operations definition.

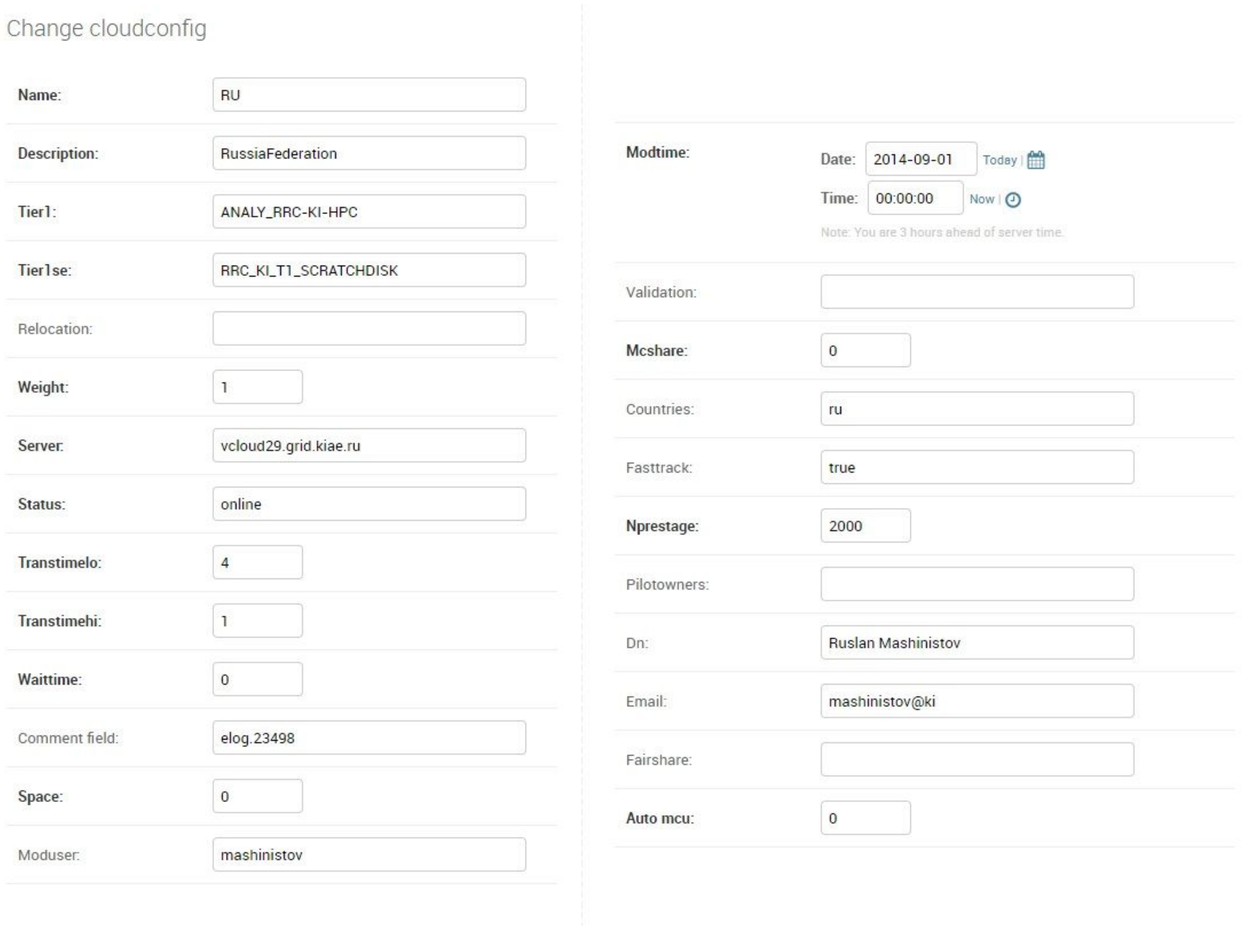

Select schedconfig to change

Action: --------- Go 0 of 2 selected

NAME	NICKNAME
test-0	TEST-NICKNAME
test-1	TEST

2 schedconfig

Schedconfig table overview.


Change cloudconfig

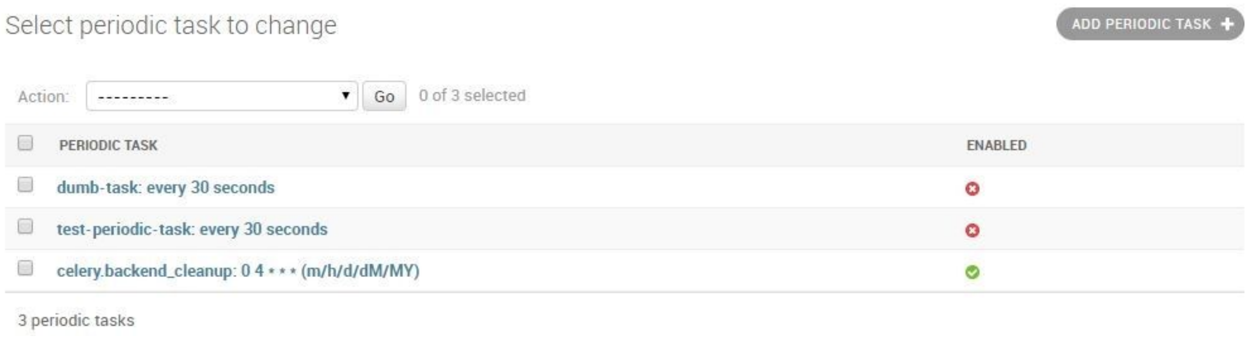
Name:	RU
Description:	RussiaFederation
Tier1:	ANALY_RRC-KI-HPC
Tier1se:	RRC_KI_T1_SCRATCHDISK
Relocation:	
Weight:	1
Server:	vcloud29.grid.kiae.ru
Status:	online
Transtimelo:	4
Transtimehi:	1
Waittime:	0
Comment field:	elog.23498
Space:	0
Moduser:	mashinistov
Modtime:	Date: 2014-09-01 Today; Time: 00:00:00 Now; Note: You are 3 hours ahead of server time.
Validation:	
Mcshare:	0
Countries:	ru
Fasttrack:	true
Npprestage:	2000
Pilotowners:	
Dn:	Ruslan Mashinistov
Email:	mashinistov@ki
Fairshare:	
Auto mcu:	0

Cloudconfig editing page. Required fields are highlighted. Django provides convenient input methods for any type of data.

Periodic tasks

In order to implement convenient manage for periodic tasks, the mechanism of their work has been changed. Celery task scheduling was used instead of cron tasks.

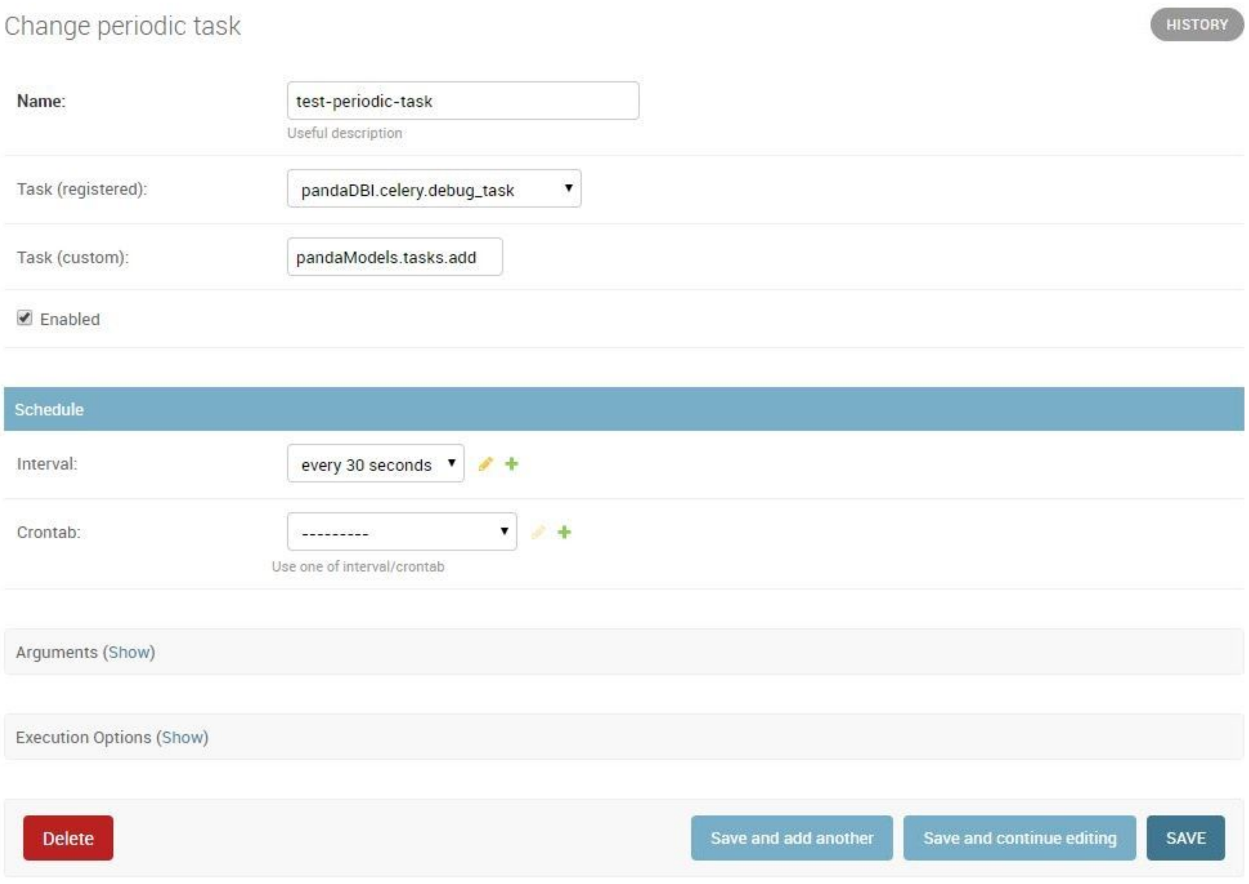
Celery is asynchronous task queue, it aims mostly on real-time operations, but also provides task scheduling via celery beat daemon. Celery periodic task management and tasks monitoring was integrated into Django admin site. This allows to configure tasks execution periods and to trace theirs state and results in real time in web-interface. However, all tasks must be redefined in Celery style.


Select periodic task to change
ADD PERIODIC TASK +
Action: --------- Go 0 of 3 selected

PERIODIC TASK	ENABLED
dumb-task: every 30 seconds	✗
test-periodic-task: every 30 seconds	✗
celery.backend_cleanup: 0 4 * * * (m/h/d/dM/MY)	✓

3 periodic tasks

Periodic tasks manage overview.

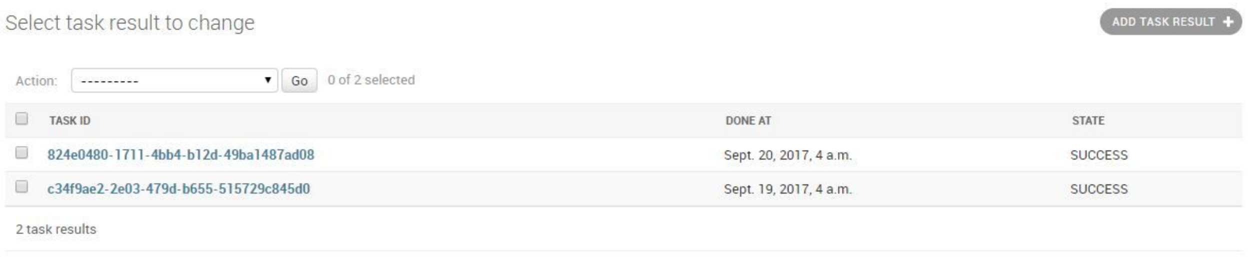


Name:	test-periodic-task (Useful description)
Task (registered):	pandaDBI.celery.debug_task
Task (custom):	pandaModels.tasks.add
☑ Enabled	

Schedule

Interval:	every 30 seconds
Crontab:	--------- (Use one of interval/crontab)

Periodic task creation page. Execution intervals can be created from this page via shortcut.



Select task result to change

ADD TASK RESULT

Action: --------- Go 0 of 2 selected

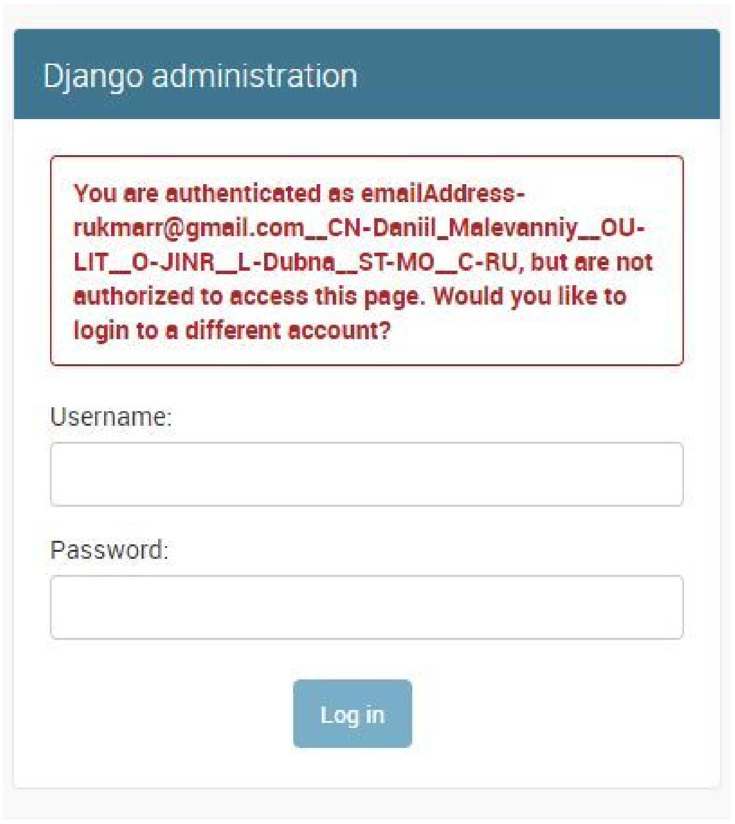
TASK ID	DONE AT	STATE
824e0480-1711-4bb4-b12d-49ba1487ad08	Sept. 20, 2017, 4 a.m.	SUCCESS
c34f9ae2-2e03-479d-b655-515729c845d0	Sept. 19, 2017, 4 a.m.	SUCCESS

2 task results

Celery tasks results overview. More details are available in individual result pages.

Certificate-based authentication

By default Django uses password-based authentication. However, just like anything in Django, it can be customized. New authorization backend was developed. It uses apache mod_ssl functionality to create or find existing Django user by data obtained from provided user certificate (mostly Common Name). For each new client with valid certificate, new user will be created. However, access to any admin site functionality must be granted manually. All authorization decisions are made by Django authorization backend, that also can be configured on Django admin site.



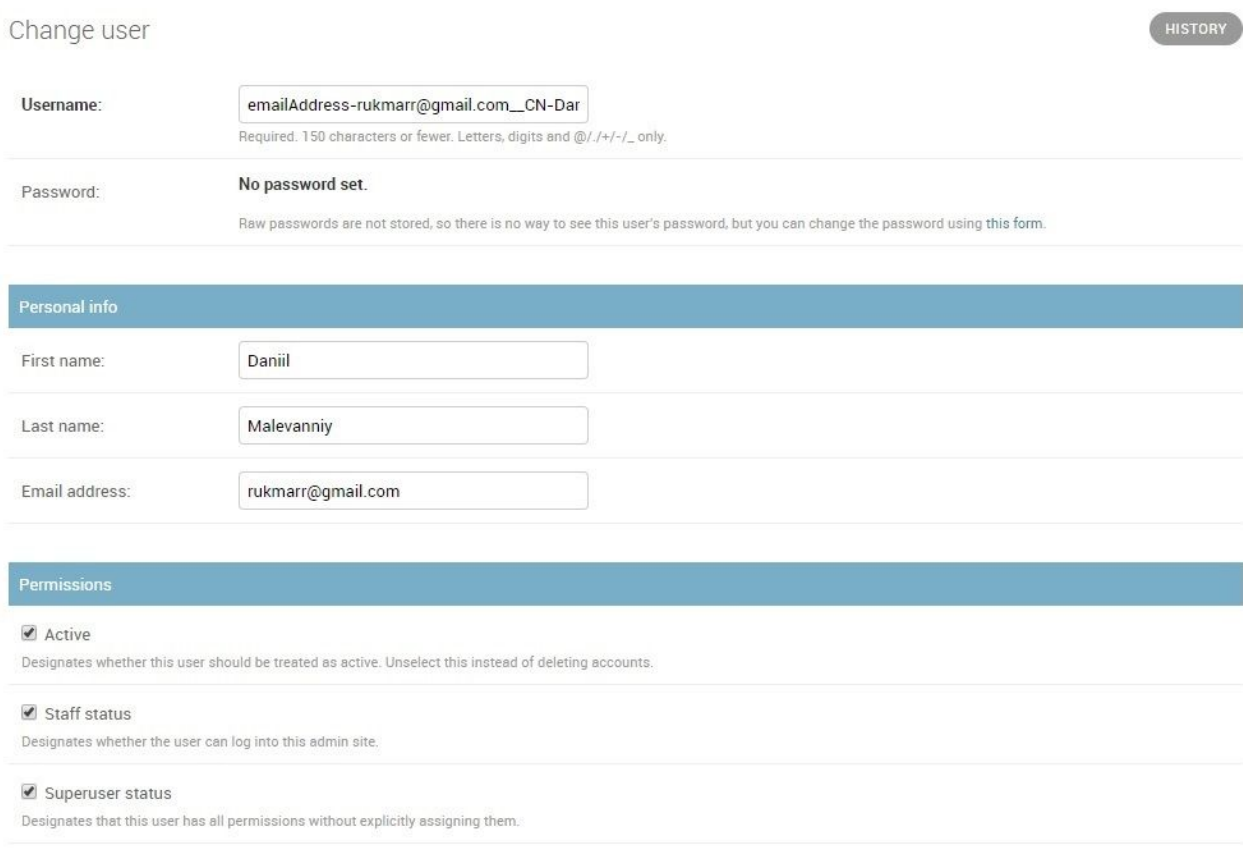
Django administration

You are authenticated as emailAddress-rukmarr@gmail.com__CN-Daniil_Malevanniy__OU-LIT__O-JINR__L-Dubna__ST-MO__C-RU, but are not authorized to access this page. Would you like to login to a different account?

Username:

Password:

Greeting page for all new users.


Change user

HISTORY

Username: emailAddress-rukmarr@gmail.com__CN-Dar

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **No password set.**

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name: Daniil

Last name: Malevanniy

Email address: rukmarr@gmail.com

Permissions

☑ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

☑ Staff status

Designates whether the user can log into this admin site.

☑ Superuser status

Designates that this user has all permissions without explicitly assigning them.

User permissions can be changed in Django admin built-in interface.

Development

Django is a open-source web framework with many features, but the most important for this project was Django admin site. It provides powerful web interface for any registred data models manage. All interfaces developed during this project, was implemented as parts of Django admin site.

Configuration tables

Only data models corresponded to configuration tables is required to use Django admin site as an interface for them. Django has built-in utility that generates such models based on database schema. However, these models need to be checked manually, as utility may not be compatible with some database features.

After models has been generated for `schedconfig`, `cloudconfig`, `siteaccess` and `users` tables, they were registered in Django admin site and some metadata was added to them in order to get cleaner look for the interface. Besides that. search was enabled for all four tables.

After that interface was ready to use. It was available in Django admin site and already had inside all data that was stored in database before interface setup.

Periodic tasks

Earlier, cron was used for Production system tasks scheduling. So, to configure schedule in runtime, crontabs had to be changed. In order to change this behavior, task scheduling has been moved from Cron to Celery Beat. Celery Beat is scheduler used with Celery task queue. It kicks off tasks at regular intervals. Celery then spread these tasks over available worker nodes, where they will be executed.

Celery execution unit, tasks, should be defined as python functions in files that later will be passed to the Celery daemon. Celery beat stores tasks execution period in the database. “django-celery-beat” package was used to set django as a backend for Celery beat. This means that Celery beat will read Django configuration files and will be using same backend database as Django. This package also contains Django data models that allows to manage schedule from Django admin site.

Celery also can trace task metadata, execution state and results and store them in the database if needed. Another package, django-celery-results, uses Django backend database to store task metadata, state and results and defines Django data model for them. This allows to use Django admin site as an interface for tasks execution tracing.

All together, these three interfaces represents complete periodic task management interface in Django admin site.

Certificate-based authentication

By default Django uses password authentication. To alter this, new Django authentication backend and middleware must be implemented. To do this, django-ssl-client-auth package was used. Since it is developed to work on NGINX server with NGINX environment variables, it has to be changed to work on Apache server with mod_ssl.

Final authentication backend uses Common Name from user certificate to find Django user and log him in if he exist. If not, it will create new user, with Common Name in username and first name and surname extracted from Common Name.All new users have no restriction to use or even see Django admin site, it must be changed manually by site administrator in Django admin site authorization management, Django built-in feature.

Conclusion

Two developed interfaces simplifies administration of PanDA workload management system and COMPASS Production System periodic tasks. Moreover, certificate-based Django user authentication backend allows to unify new tools with the rest COMPASS infrastructure. Since it fully compatible with all Django features, it can can later be used in the Django-based COMPASS applications.

Because of wide use of ready-made solutions, minimal number of new tools must be created. This speed up the development process and simplify the integration of new tools into the existing infrastructure. Besides that, system maintenance should also be easier.

References

1. P. Calafiura1, K. De, W. Guan, T. Maeno, P. Nilsson, D. Oleynik, S. Panitkin, V. Tsulaia, P. Van Gemmeren, and T. Wenaus on behalf of the ATLAS Collaboration: “The ATLAS Event Service: A new approach to event processing” // J. Phys.: Conf. Ser. 664 062065 (2015). IOP Publishing
2. A. Sh. Petrosyan , E. V. Zemlyanichkina “PanDA for COMPASS: processing data via Grid” // Distributed Computing and Grid-technologies in Science and Education 2016
3. A. Klimentov, P. Buncic, K. De, S. Jha, T. Maeno, R. Mount, P. Nilsson, D. Oleynik, S. Panitkin, A. Petrosyan, R. J. Porter, K. F. Read, A. Vaniachine, J. C. Wells and T. Wenaus "Next Generation Workload Management System For Big Data on Heterogeneous Distributed Computing" // J. Phys.: Conf. Ser. 608 012040 (2015). IOP publushing
4. K. De, A. Klimentov, T. Maeno, P. Nilsson, D. Oleynik, S. Panitkin, A. Petrosyan, J. Schovancova, A. Vaniachine, T. Wenaus, on behalf of the ATLAS Collaboration "The future of PanDA in ATLAS distributed computing" // J. Phys.: Conf. Ser. 664 062035 (2015). IOP Publishing
5. Django Framework documentation: docs.djangoproject.com
6. Celery task queue documentation: docs.celeryproject.org