# Dynamic On-Demand Analysis Service: DODAS

Marica Antonacci[1], Tommaso Boccali[2], Andrea Ceccanti[3], Giacinto Donvito[1], Cristina Duma[3], Davide Salomoni[3], Daniele Spiga[4]

[1] INFN-Bari, [2] INFN-Pisa, [3] INFN-CNAF, [4] INFN-Perugia

## Introduction

**The Compact Muon Solenoid** (CMS) is one of the two general purpose experiments at the Large Hadron Collider (LHC) at CERN in Geneva. CMS relies on the distributed computing capacities of the WLCG in order to process and analyze the collision data taken during LHC live time. Solutions having the potential to provide additional (e.g donated, hired, temporary, etc...) computing capacity to the LHC experiments and hence to CMS are of extreme interest.
As shown below, the expectation for the next decade predicts an increase in the computing needs which will be difficult to support with standard Grid facilities.


## Objectives

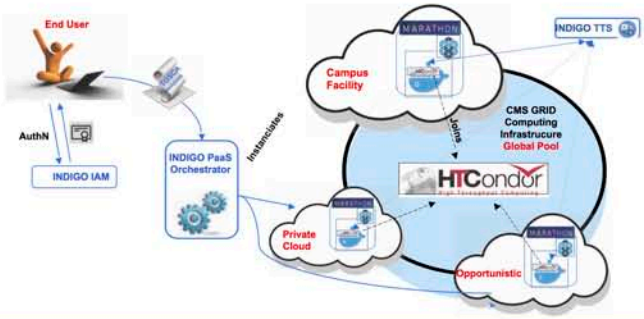
**To develop a solution for generating an on-demand, container based HTCondor cluster, possibly seamlessly integrating an existing HTCondor pool**

By simplifying and automating the process of creating, managing and accessing a pool of computing resources the project aims to improve

1. **Sites management:**
   a. A simple solution for elastic site extensions on "opportunistic"/stable resources
   b. Generation of a ephemeral WLCG-Type facility on demand, for data
2. **Users experience:**
   a. A friendly procedure to dynamically instantiate a spot 'analysis resource center'
3. **Experiment-Collaboration resources:**
   a. A comprehensive approach to opportunistic computing.
      a. Access and orchestrate multiple e.g. campus centers, harvesting all the free CPU cycles without major deployment effort

## Results

**DODAS**, an enabler of on demand dynamic clusters executing HTCondor batch systems, has been developed.
The service has been integrated with CMS computing infrastructure. This demonstrates how DODAS can dynamically extend an already existing HTCondor global queue.
The CMS integration emphasizes how such mechanism represents a possible solution to harvest geographically distributed cloud resources and to serve them as a single batch system: a possible approach to the opportunistic computing.




## Architecture

**The architecture of DODAS is composed by multiple INDIGO-DataCloud components as detailed below.**

**The four pillars are:**

**Cluster Management:**

**Mesos clusters** a solution for the execution of docker containers for all the services required by a regular CMS site (Worker Node, HTCondor Schedd and squids, X509 cache).

**Marathon** Application Framework that guarantees the dynamic scaling up and down of resources, a key point.

**AuthN/Z & Credential Management:**

**INDIGO Identity Access Management (IAM)** service responsible for AuthN/Z to the cluster generation.

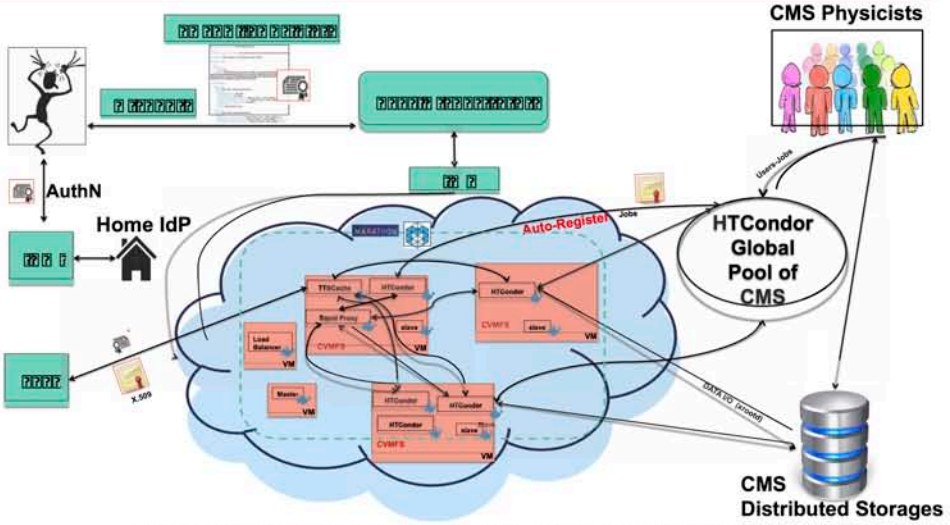
**Token Translation Service (TTS)** enables the conversion of IAM tokens in to a X.509 certificates. A key to implement a trusted condor_startd Auto Registration.

**Data Management:**

**Dynafed & FTS** is the approach currently followed by the project. We will investigate **Oneclient** (from **Onedata**) as a tool allowing to mount remote Posix file-system.

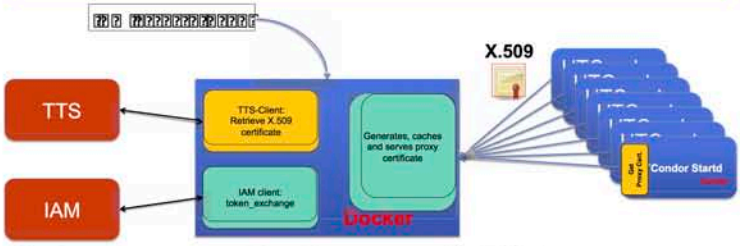
**Automation:**

- **TOSCA templates,** meant to be managed by INDIGO PaaS **Orchestrator**, allow the automation of the overall setup.
  - A single YAML file describes the complex setup of all required services and dependencies




The schema represents an high level view of the end user interactions with DODAS. Green colored boxes represent the major INDIGO-DataCloud adopted components.
The figure refers to the HTCondor based scenario, where a user, upon authentication with IAM, submits a properly configured TOSCA template to the PaaS Orchestrator. The latter takes care to interact with IaaS provider, possibly through Infrastructure Manager. At the end a Apache Mesos cluster is deployed and the worker nodes (Docker applications) auto register with Global Pool to execute CMS Analysis Jobs.
CVMFS is installed on host slave machines, while squid proxy and TTS Cache are containerized.

## AuthN/Z




The above schema represents the major components of the TTSCache.
The incoming token passed as TOSCA parameter, is propagated to downstream services till the Docker applications, through Marathon.
TTSCache performs a token exchange, which is a controlled way of obtaining the ability of acting on behalf of a user for a possibly long amount of time, and the exchanged token is used to retrieve a X.509 certificate.
The latter is used to perform a trusted auto-registration with HTCondor Global Pool.

## Automation


Ansible excerpts of CMS experiment specific configurations