DEEP: a user success story

Focus is on how a developer can start working on an application and integrate it with DEEP components
Designing and Enabling E-Infrastructures for intensive data Processing in a Hybrid DataCloud

- Started as a spin-off project (together with XDC) from INDIGO-DataCloud technologies
- H2020 project, EINFRA-21 call
- Runs November 1st 2017 – April 2020
- March 27, 2019: mid-term Review (Luxembourg)
- 9 academic partners + 1 industrial partner:
  CSIC, LIP, INFN, PSNC, KIT, UPV, CESNET, IISAS, HMGU; Atos

Goal: prepare a new generation of e-Infrastructures that harness latest generation technologies, supporting deep learning and other intensive computing techniques to exploit very large data sources

Global objective: promote the use of intensive computing services by different research communities and areas, an the support by the corresponding e-Infrastructure providers and open source projects

Ease and lower the entry barrier for non-skilled scientists

- Transparent execution on e-Infrastructures
- Build ready to use modules and offer them through a catalog or marketplace
- Implement common software development techniques also for scientist’s applications (DevOps)
DEEP from a user point of view
Basic user: prediction

- The **basic user** can **browse the different modules in the Open Catalogue**

- These modules can **be deployed in different ways**:
  - **Locally** (docker, udocker, ... )
  - **HPC environment** (udocker, singularity, ...)
  - **DEEP Testbed** using **orchent** and **oidc-agent** from the command line
  - **DEEP Testbed** from the **web dashboard**
  - **DEEP Testbed** using **Alien4Cloud DEEP**

- Why different ways of deploying:
  - **Local vs. HPC vs. DEEP Testbed**: depending on where you have access and what resources are best suited
  - **Orchent from command line**: **Scales more easily.**
  - **Dashboard**: **Easier to control** what is going on (smartphone, tablet...)
  - **Alien4Cloud**: **GUI builder** for TOSCA templates (useful for advanced users)
**Basic user: prediction**

- **Module from DEEP Open Catalogue**
  - Download and run it locally with udocker

- **Run the DEEPaaS API**
  - Access the API from the web browser

- **Dashboard**
  - Select the TOSCA template for the module

- **Send deployment to testbed**
  - Open the DEEPaaS API endpoint in the browser

- **Command Line**
  - Download the TOSCA template and modify if needed

- **Send deployment to testbed (orchent)**
  - Open the DEEPaaS API endpoint in the browser

- **Predict**

---

DEEP user demo | EOSChub week, 10-12 April
Intermediate User: Training

**Command Line**
- Download the TOSCA template and modify if needed
-Authenticate with a DEEP-IAM token using oidc-agent

**Send deployment to testbed (orchent)**
- Open the DEEPaaS API endpoint in the browser

Data for training
- mount Nextcloud folders inside the deployment

**Nextcloud** ➔ **rclone** ➔ **Train**
- Using the weights obtained in the training

**Predict**
Advanced user workflow

- Create project structure with **DEEP DataScience template**
- Work locally using your preferred Integrated Development Environment
- Test your integration with DEEP components by either building your own Docker image or by using **Development Docker Image** (DDI)
- Test your code directly on **DEEP Testbed** and debug it with DDI
- Profit from DEEP **CI/CD pipeline** based on Jenkins
  - ‘test’ github branch -> test environment with test Docker images
  - ‘master’ github branch -> production
Data Science Template


1. $>\text{pip install cookiecutter}$
2. $>\text{cookiecutter https://github.com/indigo-dc/cookiecutter-data-science}$
3. answer questions
4. Two directories created: <user_project> and <DEEP-OC-user_project>
   This are also git repositories. Both have ‘master’ and ‘test’ branches.
5. Create corresponding github repositories, push your local ones there
6. Start working on your application
DS template: answering questions

valentin@valentin-T470s:~$ cookiecutter https://github.com/indigo-dc/cookiecutter-data-science
You've downloaded /home/valentin/cookiecutters/cookiecutter-data-science before. Is it okay to delete and re-download it? [yes]: yes
github_user [User account at github.com, e.g. 'indigo-dc' in https://github.com/indigo-dc]: vykozlov
project_name [project_name]: A demo project
repo_name [a_demo_project]: demo_project
author_name [Your name (or your organization/company/team)]: V. Kozlov (KIT)
author_email [Your email]: valentin.kozlov@kit.edu
description [A short description of the project.]: A demo project using DEEP Data Science template
app_version [Application version (expects X.Y.Z (Major.Minor.Patch))]: 0.0.1
Select open_source_license:
1 - MIT
2 - BSD-3-Clause
3 - No license file
Choose from 1, 2, 3 [1]: 1
Select python_interpreter:
1 - python3
2 - python
Choose from 1, 2 [1]: 1
dockerhub_user [User account at hub.docker.com, e.g. 'deephdcc' in https://hub.docker.com/u/deephdcc]: vykozlov
docker_baseimage [Base Docker image for Dockerfile, e.g. tensorflow/tensorflow]: tensorflow/tensorflow
baseimage_cpu_tag [CPU tag for the Base Docker image, e.g. 1.12.0-py3. Has to match python version!]: 1.12.0-py3
baseimage_gpu_tag [GPU tag for the Base Docker image, e.g. 1.12.0-gpu-py3. Has to match python version!]: 1.12.0-gpu-py3
DEEP use-cases follow DS template

- Plant classification + Image Classification
  
  https://github.com/deephdc/DEEP-OC-plants-classification-tf
  
  https://github.com/deephdc/DEEP-OC-image-classification-tf
  
  https://github.com/deephdc/image-classification-tf

- Satellite imagery
  
  https://github.com/deephdc/DEEP-OC-sen2sr
  
  https://github.com/deephdc/sen2sr

- MODS
  
  https://github.com/deephdc/DEEP-OC-mods
  
  https://github.com/deephdc/mods

- Retinopathy
  
  https://github.com/deephdc/DEEP-OC-retinopathy_test
  
  https://github.com/deephdc/retinopathy_test

- DLI: dogs breed detector
  
  https://github.com/deephdc/DEEP-OC-dogs_breed_det
  
  https://github.com/deephdc/dogs_breed_det
In the live demo

- Deploy Development Docker Image on the Testbed by means of Alien4Cloud
- Demonstrate that DEEPaaS API can be started but there is no user application installed
- Clone an application from a github repository
- Switch to ‘test’ branch and modify a file
- Reload DEEPaaS API to show that changes are applied
- (optionally) Execute training protected by FLAAT, therefore provide IAM token to trigger it
- Commit changes back to the github repository
- Show that Jenkins CI/CD starts automatically and finishes with success
Alien4Cloud

• **Alien4Cloud**: An open-source platform that provides visual composition of TOSCA templates (following the YAML specification) and provides a pluggable mechanism to connect to Orchestrators [1]

• **Alien4Cloud in DEEP**:
  • Forked source code [2]
  • Docker image: [https://hub.docker.com/r/indigodatacloud/alien4cloud-deep/](https://hub.docker.com/r/indigodatacloud/alien4cloud-deep/)
  • Modified parsers to support normative TOSCA YAML specification (aiming at v1.2)
  • Support to the custom node types and the plugin to deploy through the PaaS Orchestrator (integrated with the INDIGO-DataCloud PaaS Orchestrator) [2]

• A prototype version is shown. Further enhancements will be included in Alien4Cloud (potentially merged upstream) [2]

Development Docker Image


1. Exposes Jupyter notebook and Jupyter Lab together with the DEEP as a Service API component. There is no application code inside!

2. Has pre-installed:
   - tensorflow, git, curl, wget, openssh-client, python, pip, rclone, flaat, jupyter, jupyterlab, deepaas

3. JupyterLab terminal allows you debugging your application directly on the Testbed!
FLAAT: FLAsk with Access Token

1. FLAAT provides a simple way to verify OIDC Access Tokens
2. Allows protecting REST interfaces with simple decorators like:
   @flaat.login_required()
3. Supports complex group membership checking as well
4. Tested OIDC Providers: IAM, EGI, Unity / B2Access as used in the Helmholtz-Data-Federation, KIT's Shibboleth installation, Google, EduTEAMS
Once a user code is committed to either ‘master’ or ‘test’ branch, it goes through CI/CD pipeline and ends with Docker images tagged for CPU/GPU version and marked in addition with ‘test’ in case of the ‘test’ branch.

- **Code version control**
  - github.com/deephdc
  - test branch
  - master branch

- **Code testing & Quality Control**
  - Jenkins
  - flake8 / PEP8
  - Bandit security scanner

- **Docker build & push to registry**
  - Jenkins

DEEP Open Catalog
1. Create a `deep-oc-<user_application>.md` file with the description of your application and necessary links

2. Create Pull Request (PR) in github for:
   
   https://github.com/deephdc/deephdc.github.io/tree/pelican/content/modules

   More automatization is coming (meta info) ...
Thank you

More docs: https://docs.deep-hybrid-datacloud.eu
DEEP Project: https://deep-hybrid-datacloud.eu
DEEP Open Catalog: https://marketplace.deep-hybrid-datacloud.eu