Training: Infrastructure as Code to deploy scientific applications in EOSC

EGI Conference 2022

Miguel Caballer (micafer@upv.es)
Amanda Calatrava (amcaar@i3m.upv.es)

Agenda

- Introduction
- Infrastructure Manager (IM)
- Elastic Cloud Computing Cluster (EC3)
- Questions

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Introduction (I)

"A long time ago, in a data center far, far away, an ancient group of powerful beings known as sysadmins used to deploy infrastructure manually. Every server, every route table entry, every database configuration, and every load balancer was created and managed by hand. It was a dark and fearful age: fear of downtime, fear of accidental misconfiguration, fear of slow and fragile deployments, and fear of what would happen if the sysadmins fell to the dark side (i.e. took a vacation). The good news is that thanks to the DevOps Rebel Alliance, we now have a better way to do things: ***Infrastructure-as-Code*** *(IAC)."*

Source https://blog.gruntwork.io/

Introduction (II)

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools. Benefits:

- You can automate your entire provisioning and deployment process, which makes it much faster and more reliable than any manual process.
- You can store those source files in version control, which means the entire history of your infrastructure is now captured in the commit log, which you can use to debug problems, and if necessary, roll back to older versions.
- You can validate each infrastructure change through code reviews and automated tests.
- You can create a library of reusable, documented, battle-tested infrastructure code that makes it easier to scale and evolve your infrastructure.

There are several tools to manage infrastructure-as-code, but the most well-known ones are **Ansible**, Puppet, Chef, Saltstack, Terraform and CloudFormation.

Introduction (III)

Both **IM** and **EC3** tools follow this principle, two services that allow users to automate the deployment and configuration process of virtual infrastructures on top of cloud resources.

In this training session, we will show both the IM Dashboard and the EC3 CLI tools in action.

With this tools:

- You can automate your entire provisioning and deployment process, which makes it much faster and more reliable than any manual process.
- You can use the same definition templates to provision the very same virtual infrastructure in different Cloud providers.
- You can use the OASIS TOSCA Simple Profile in YAML standard to describe your cloud topologies.
- You can store those source files in version control, which means the entire history of your infrastructure is now captured in the commit log, which you can use to debug problems, and if necessary, roll back to older versions.

Grid y Computación de Altas Prestaciones
GRyCAP

Infrastructure Manager (IM)

Infrastructure Manager

UPV

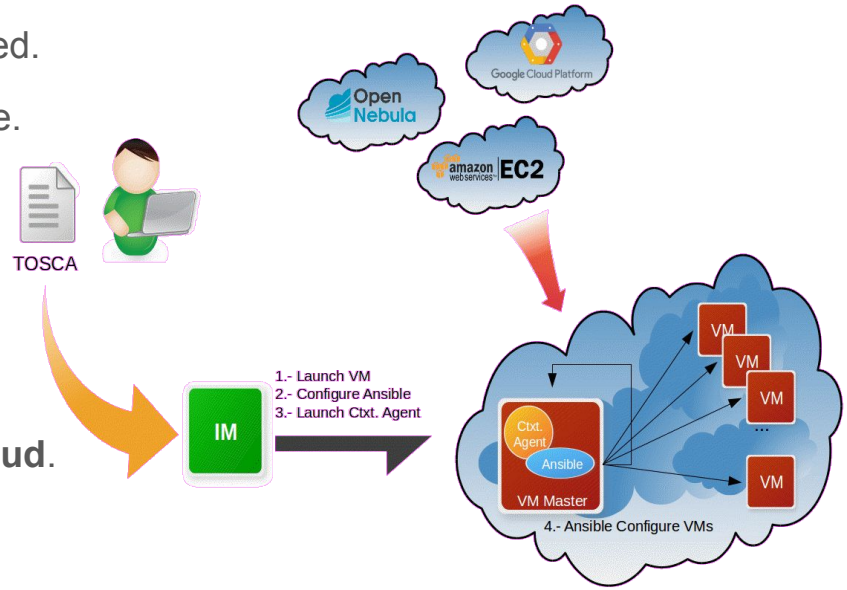
Introduction to IM

- IM is a service that **deploys virtual infrastructures** on top of Cloud resources.
- It uses **RADL** or **TOSCA** files to describe the infrastructure.
  - Infrastructure as code (**IaC**)
- The IM **automates** the deployment, configuration, software installation, monitoring and update of virtual infrastructures.
- It supports a wide variety of back-ends, thus making user applications **Cloud agnostic**.

Infrastructure
Manager

General View

- General platform to deploy on demand **customizable** virtual computing infrastructures.
  - With the **precise software configuration** required.
  - Allow to deploy any kind of complex infrastructure.
  - Share Infrastructure descriptions.
  - **No need of pre-packaged VMIs.**
    - Enable re-using of VMIs.
  - **The same complex infrastructure can be deployed both on-premise and in a public Cloud**.



IM features

- It features **DevOps** capabilities.
  - Based on **Ansible**.
  - Provides recipes for common deployments.
  - Also supporting cloud-init scripts.

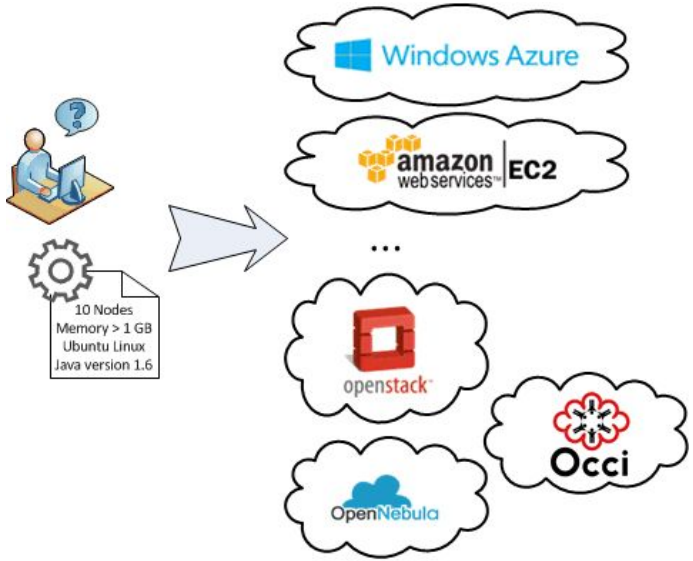
- IM works as a service that offers several interfaces:
  - XML-RPC and **REST** APIs.
  - Command-line application.
  - Web-based GUI.

- It is distributed under a GNU **GPL v3.0** open source license and its source code is available on **GitHub**.

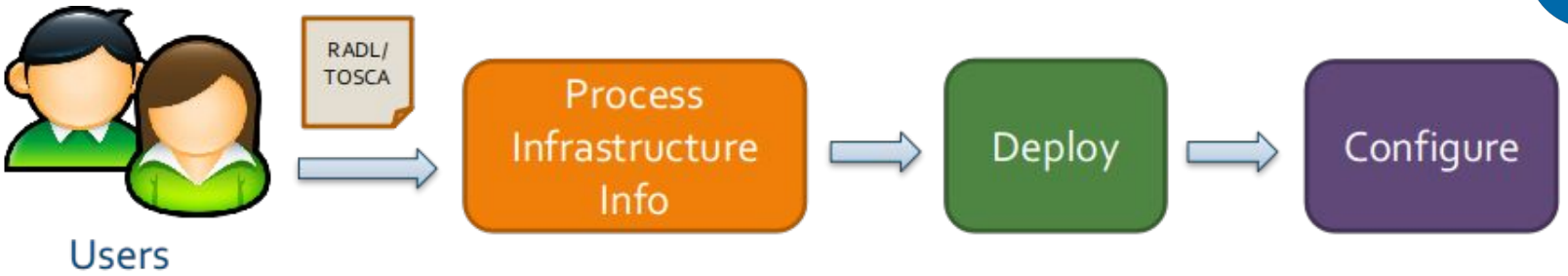
https://github.com/grycap/im

Cloud Providers

- It supports a **wide range of cloud providers** and other computing back-ends:
  - **Public**: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, T-Systems OTC, Exoscale, Cloud&Heat.
  - **On-premises**: OpenNebula, OpenStack, CloudStack, VMWare, libvirt.
  - **Federated**: EGI FedCloud (OCCI and OpenStack), FogBow.
  - **Containers**: Docker, Kubernetes
  - The list above can be **easily extended** by plugins.



IM - Working Scheme



- The user can provide an **RADL** or **TOSCA** documents as input to the IM, describing the infrastructure:
  - **RADL**: Resource and Application Description Language.
    - High level Language to define virtual infrastructures and Specify VM requirements.
  - **TOSCA: OASIS Standard**
    - Open standard language to model application architectures to be deployed on a Cloud.

RADL

An RADL document has the following general structure:

- The keywords ansible, network, system and configure assign some features or recipes to an identity <id>. The features are a list of constraints separated by and, and a constraint is formed by <feature name> <operator> <value>.

ansible <ansible_host_id> (<features>)

network <network_id> (<features>)

system <system_id> (<features>)

configure <configure_id> (<Ansible recipes>)

contextualize [max_time] (system <system_id>

configure <configure_id> [step <num>] ...)

deploy <system_id> <num> [<cloud_id>]

RADL Example

- In this example
  - A node type named “node” with 1 CPU and 512MB of RAM is defined.
  - Connected to a public network
  - In the configuration a user named “user1” is created.
  - 1 node of type “node” is deployed

```
network net (outbound = 'yes')
system node (
  cpu.count = 1 and
  memory.size >= 512M and
  net_interface.0.connection = 'net'
)
configure node (
@begin
---
 - tasks:
   - user: name=user1 password=1234
 @end
)
deploy node 1
```

TOSCA

- Topology and Orchestration Specification for Cloud Applications
  - OASIS Standard
  - TOSCA Simple Profile in YAML Version 1.0
  - Standard to specify Cloud Topologies
  - Defines the interoperable description of services and applications hosted on the cloud
    - Including their components, relationships, dependencies, requirements, and capabilities
  - Enabling portability and automated management across cloud providers

TOSCA

```
tosca_definitions_version:
tosca_simple_yaml_1_0

imports:

  - types: https://../custom_types.yaml

description: Deploy instance for Kepler

topology_template:

  inputs:

    memory_size:
      type: string
      description: RAM memory
      default: 1 GB

  node_templates:

    kepler:
      type: tosca.nodes.indigo.Kepler
      requirements:
        - host: kepler_server
```

```
kepler_server:
  type: tosca.nodes.indigo.Compute
  capabilities:
    endpoint:
      properties:
        network_name: PUBLIC
        ports:
          vnc_port:
            protocol: tcp
            source: 5900
    host:
      properties:
        num_cpus: 1
        mem_size:
          get_input: memory_size
    os:
      properties:
        type: linux
        distribution: ubuntu
        version: 18.04
```

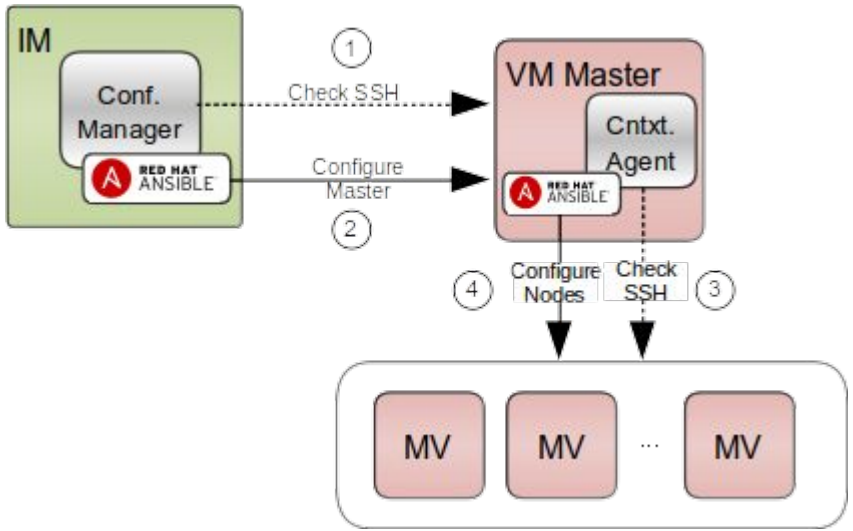
```
outputs:
  instance_ip:
    value:
      get_attribute:
       - kepler_server,
       - public_address
       - 0
  instance_creds:
    value:
      get_attribute:
       - kepler_server,
       - endpoint
       - credential
       - 0
```

IM image URIs

- The user specifies the image (or list of images) to use.
  - **URI naming convention** to abstract from cloud provider:
    - one://server:port/image-id
    - ost://server:port/ami-id
    - aws://region/ami-id
    - appdb://site_name/image_name?vo_name
    - <site end-point>/<image-id>
  - In INDIGO-DataCloud, the image information is obtained from the CMDB.
- Then, the IM obtains the list of IaaS providers available to the user.
  - From the credentials provided by the user.
- Finally, it contacts the IaaS provider selected and deploys the infrastructure.

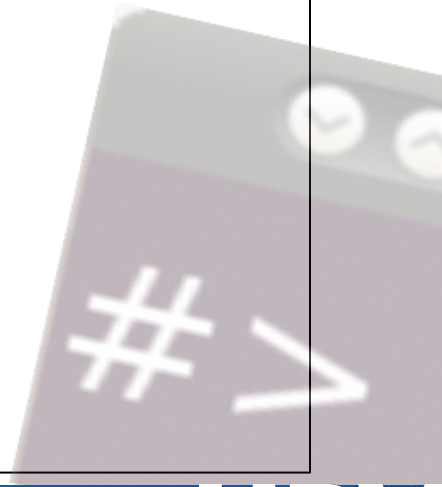
Contextualization process

1. SSH connection to the Master VM
   - A Linux-based VM with a public IP
2. Configure Master VM
   - Install and configure Ansible
     i. Also with Ansible
3. Launch Contextualization Agent
   - Check SSH from VMs
   - Call Ansible



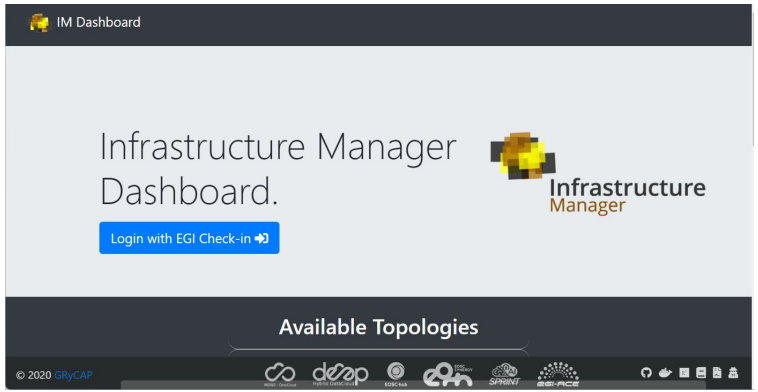
Client-side Tools: CLI

```
Usage: im_client.py [-u|--xmlrpc-url <url>] [-r|--restapi-url <url>] [-v|--verify-ssl] [-a|--auth_file
<filename>] operation op_parameters

Operation:

  list

  create          <radl_file> [async_flag]

  destroy         <inf_id>

  getinfo         <inf_id> [radl_attribute]

  getradl         <inf_id>

  getcontmsg      <inf_id>

  getstate        <inf_id>

  getvminfo       <inf_id> <vm_id> [radl_attribute]

  getvmcontmsg    <inf_id> <vm_id>

  addresource     <inf_id> <radl_file> [ctxt flag]

  ...
```



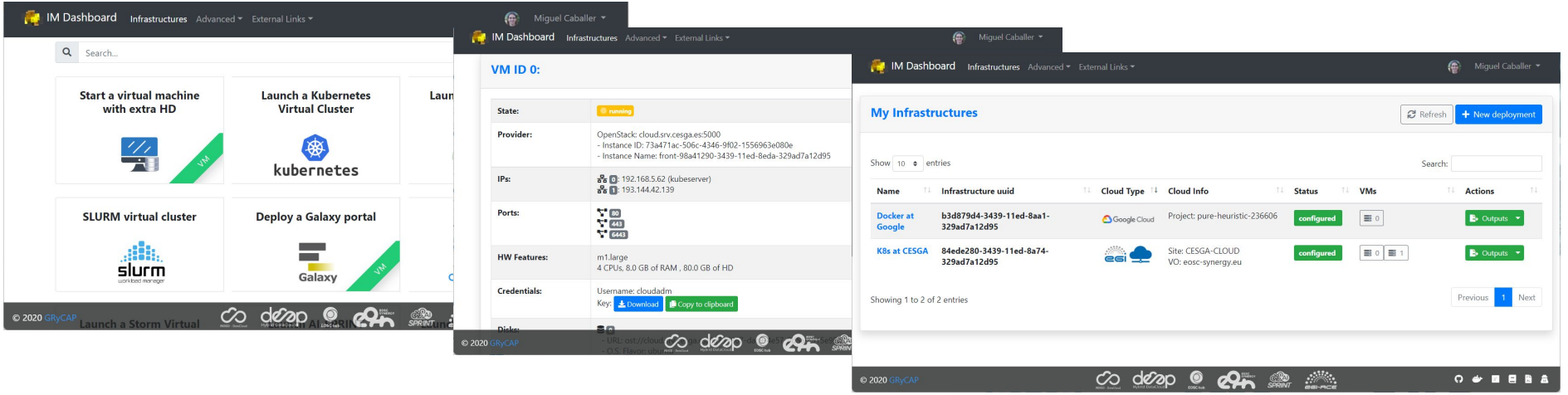
Client-SIDE Tools: Web

- Publicly-available web interface (also open-sourced).
  - https://im.egi.eu
    - Login with EGI Checkin.
    - Integrated with AppDB.
  - Easily deploy infrastructures from a web browser
  - Also on GitHub:
    - https://github.com/grycap/im-dashboard



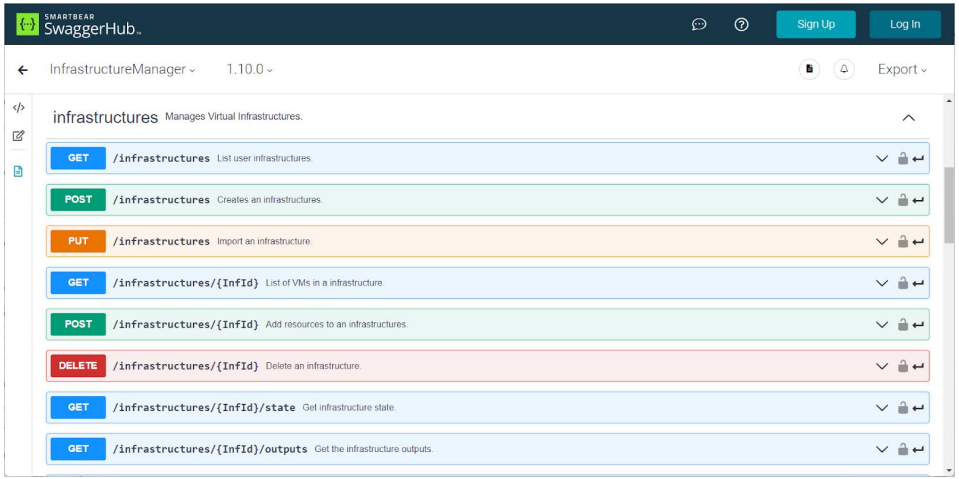
Client-SIDE Tools: Web

- Easy interface
  - For non advanced users
  - Easily deploy infrastructures from a web browser
    - Select it from a list of configurable list of templates.



APIs to be consumed by Clients

- XML-RPC API
  - API that follows the XML-RPC specification.
- REST API
  - IM Service can be accessed through a REST(ful) API
  - Follows **OpenAPI** Specification

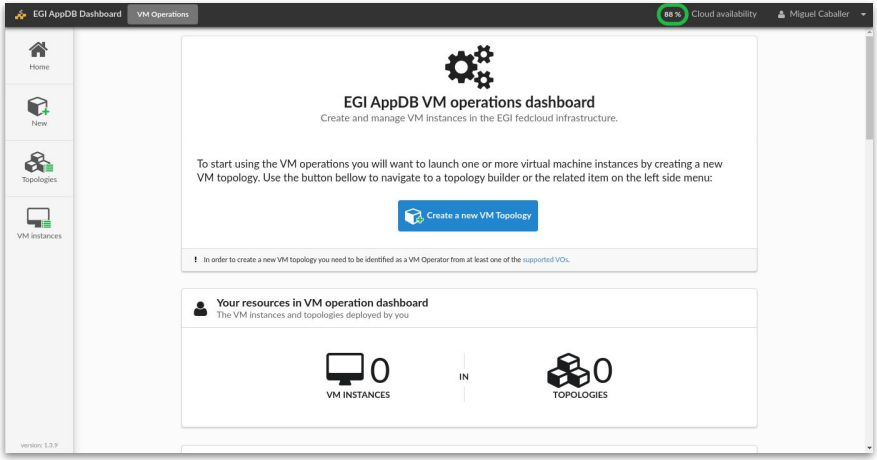


More info:

- https://app.swaggerhub.com/apis/grycap/InfrastructureManager/
- http://www.grycap.upv.es/im/documentation.php

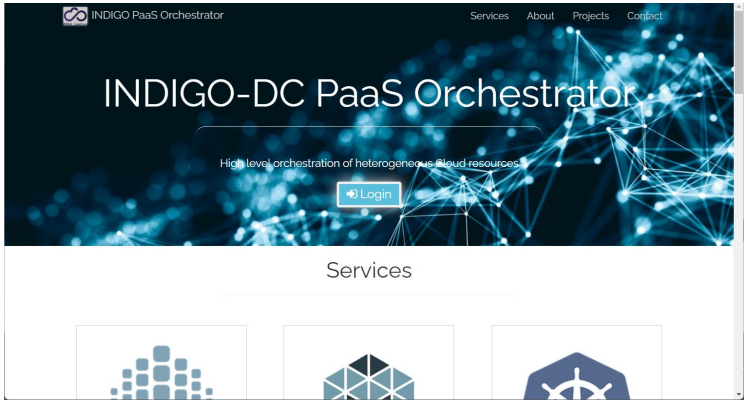
Where is the IM used?

- The IM is used in the VMOps Dashboard of EGI.
  - As the EGI Cloud Compute communication layer to create VM topologies.
  - https://dashboard.appdb.egi.eu
  - https://docs.egi.eu/users/compute/cloud-compute/monitor/



Where is the IM used?

- By the INDIGO PaaS Orchestrator:
  - IM is a key component of the architecture:
  - Used at IaaS level to provide TOSCA-based deployment of infrastructures.
  - https://indigo-paas.cloud.ba.infn.it



Demo

Let's access the IM Dashboard!!

Infrastructure Manager

https://im.egi.eu

https://marketplace.eosc-portal.eu/services/infrastructure-manager-im

See full demo video at:

- https://youtu.be/vmtzGOZxiUg

More Information

Video demos in YouTube:

https://youtube.com/playlist?list=PLgPH186Qwh_37AMhEruhVKZSfoYpHkrUp

IM images in Docker Hub:

https://hub.docker.com/r/grycap/im/

https://hub.docker.com/r/grycap/im-dashboard/

Source Code in GitHub:

https://github.com/grycap/im

https://github.com/grycap/im-dashboard

IM Info Web:

http://www.grycap.upv.es/im

Grid y Computación de Altas Prestaciones
GRyCAP

Elastic Compute Cluster in the Cloud (EC3)

UPV

What is EC3?

- EC3 was created with the idea of providing virtual elastic computer clusters on Cloud platforms.

Facilitate access to computing platforms for non-experienced users

Maintain the *traditional* work environment, with clusters configured with a well-known middleware.

Automatic management of elasticity, reducing costs (public cloud) and energy expenditure (private cloud).

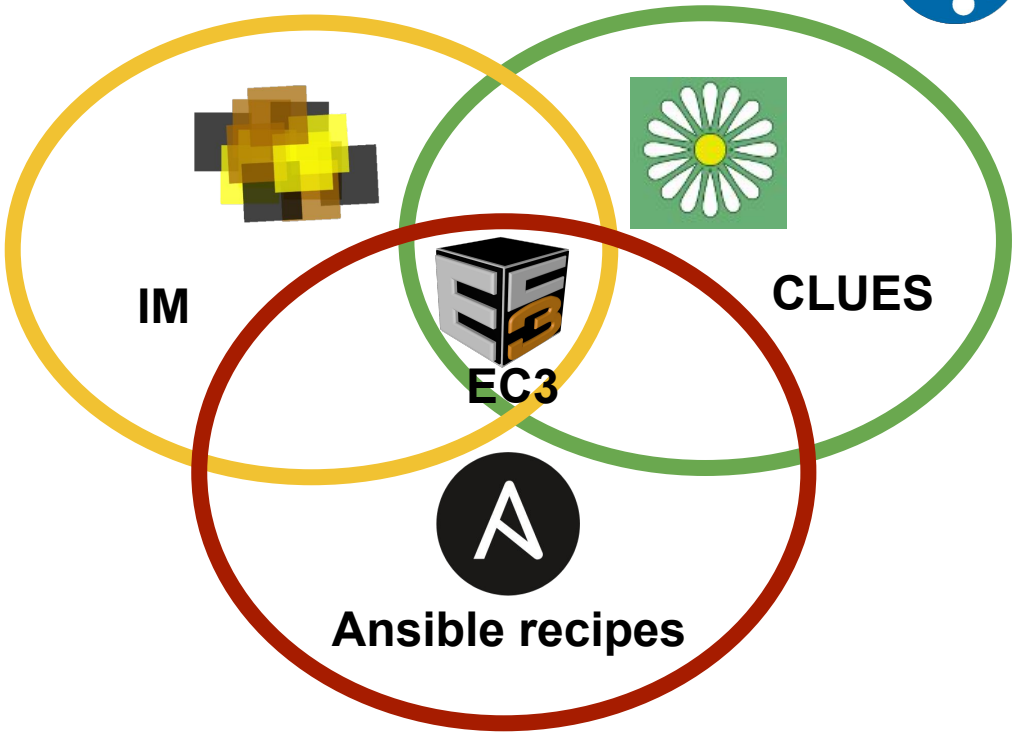
Automatic configuration of the application execution environment.

Compatible with a wide range of cloud providers (public, federated and on-premises).

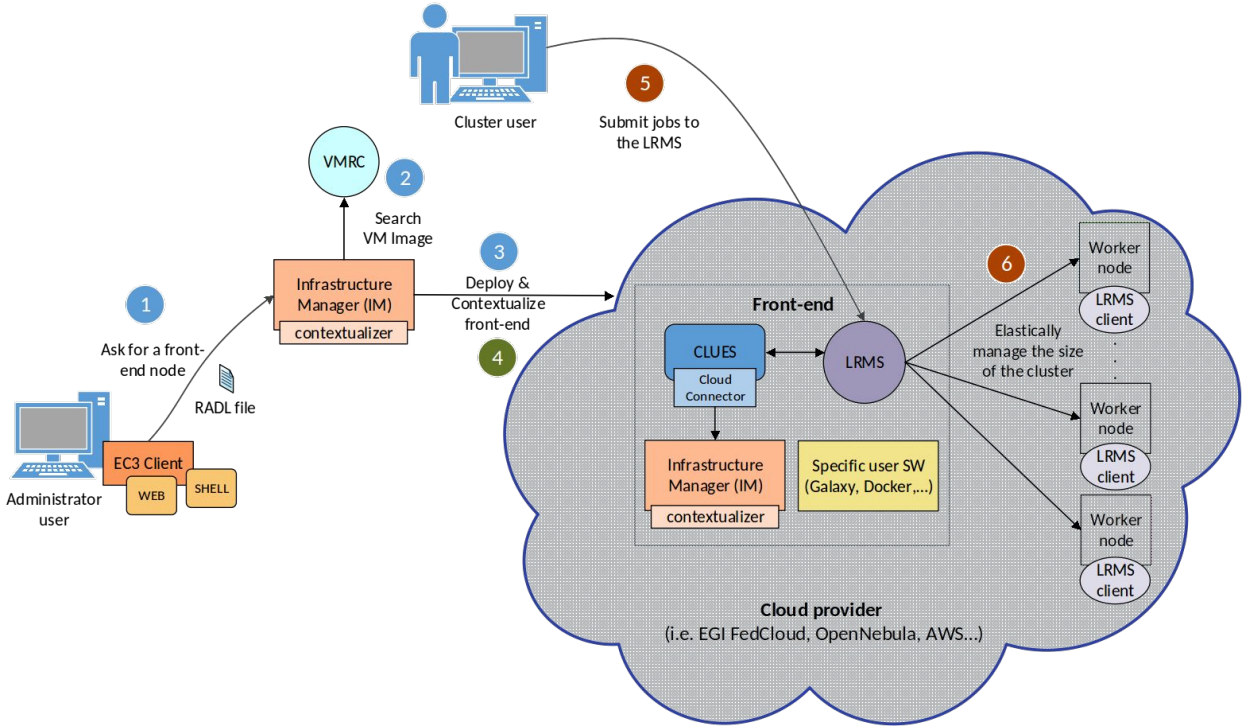
Support for hybrid clusters.

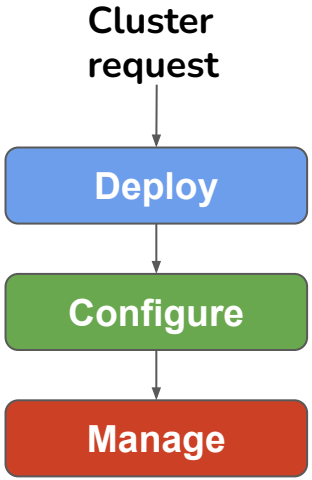
EC3 components

- **EC3** deploys and configures **virtual elastic** clusters. It relies on **IM** to deploy the machines and on **CLUES** to automatically manage the elasticity.
- Offers a set of predefined templates to configure the resources through **Ansible**:
  - Kubernetes, Mesos, SLURM, Torque, SGE, HTCondor, Nomad.



EC3 Architecture





Automatic Elasticity

- **Elasticity Management**: ability to adapt the size of the cluster to the workload dynamically and automatically:
  - **Horizontal Elasticity**: increase / decrease the number of VMs.
- **Self-management**: elasticity rules are evaluated from the main node of the cluster without requiring any external entity in charge of monitoring the cluster to decide when to increase / reduce the size of the cluster.
- **Transparency**: elasticity should not affect the execution of tasks, going unnoticed both for tasks and for the user.

Automatic Elasticity (II)

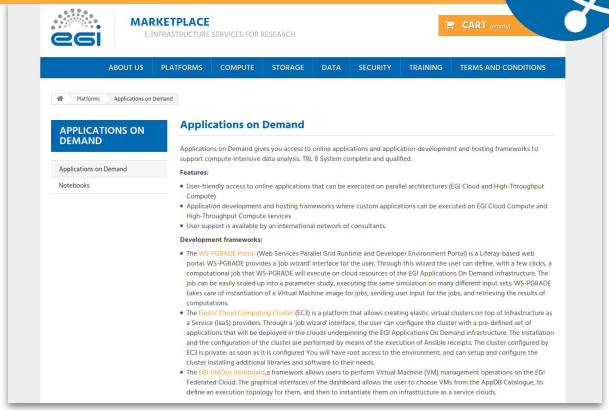
- The elasticity module is responsible for dynamically adding and removing nodes from the cluster by monitoring the LRMS.
- Deployment policies (scale out):
  - On demand: a node is deployed for each job that comes to the queue.
  - Bursts: deploys a group of VMs for each job in the queue, assuming that if a job arrives at the LRMS, there is an increased chance that new jobs will arrive soon. (i.e HTC applications).
- Undeployment policies (scale in):
  - On demand: ends idle nodes when there are no pending jobs in the LRMS queue.
  - Delayed power off: inactive nodes turn off after a certain configurable period of time. (i.e public clouds)
- CLUES supports the monitoring and management of several LRMS, such as SLURM, Kubernetes and Mesos, among others. As it is implemented based on a plug-in structure, a new plug-in can be easily developed to support a new batch system. All the current available plug-ins can be seen here.

EC3 in the EGI Applications on Demand

- The EGI AoD allows small laboratories and individual researchers the access to a wide range of computational resources and on-line services to manage and analyse large amount of data.
- Inside this service we find the EC3 portal:
  - The EC3 AoD portal enables to launch virtual elastic clusters on top of EGI FedCloud resources using the EC3 tool.
  - It only requires the EGI checking account (and vo.access.egi.eu VO) to access to the service.
  - The user is guided step by step in the deployment process.
  - Documentation and tutorials are available, i.e. configuring a Galaxy cluster for data intensive research

EC3 Portal

- EC3aaS facilitates the usage of EC3 to non-experienced users:
  - It presents an user-friendly web interface that allows to easily deploy and configure a virtual elastic cluster on several cloud providers, including EGI FedCloud.
  - Limited actions: create, list and destroy.
- Documentation: https://ec3.readthedocs.io/en/latest/ec3aas.html
- Endpoint: https://servproject.i3m.upv.es/ec3-ltos
- Marketplace: https://marketplace.eosc-portal.eu/services/elastic-cloud-compute-cluster-ec3/details



EC3 Client

- More powerful client interface than the Web interface:
  - More control over the cluster (reconfigure, clone, migrate, stop, restart).
  - Support for hybrid clusters
  - Support for golden images
- The user needs to define an authorization file
- Documentation: https://ec3.readthedocs.io/en/latest/ec3.html
- EC3 Client Source Code in GitHub: https://github.com/grycap/ec3
- EC3 Client image in Docker Hub: https://hub.docker.com/r/grycap/ec3/

EC3 Client (II)

```
usage: ec3 [-h] [-v] [-l LOG_FILE] [-ll LOG_LEVEL] [-q]
           {launch,list,show,templates,ssh,reconfigure,destroy,clone,migrate,stop,restart}
           ...

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  -l LOG_FILE, --log-file LOG_FILE
                        log output file
  -ll LOG_LEVEL, --log-level LOG_LEVEL
                        log level. 1: debug; 2: info; 3: warning; 4: error
  -q, --quiet           only print messages from front-end

subcommands:
  valid subcommands

  {launch,list,show,templates,ssh,reconfigure,destroy,clone,migrate,stop,restart}
                        additional help
    launch              launch a new cluster
    list                list launched clusters
    show                print RADL
    templates           list available templates
    ssh                 connect to cluster via SSH
    reconfigure         reconfigure the cluster
    destroy             destroy a launched cluster
    clone               clone a launched cluster in another Cloud provider
    migrate             migrate a launched cluster together with its workload
                        to another Cloud provider
    stop                stop a launched cluster
    restart             restart a previously stopped cluster
```

EC3 Client (II)

```
usage: ec3 [-h] [-v] [-l LOG_FILE] [-ll LOG_LEVEL] [-q]
           {launch,list,show,templates,ssh,reconfigure,destroy,clone,migrate,stop,restart}
Operation:
     launch                launch a new cluster
     list                  list launched clusters
     show                  print RADL
     templates             list available templates
     ssh                   connect to cluster via SSH
     reconfigure           reconfigure the cluster
     destroy               destroy a launched cluster
     clone                 clone a launched cluster in another Cloud provider
     migrate               migrate a launched cluster to another Cloud provider
     stop                  stop a launched cluster
     restart               restart a previously stopped cluster
  ...
```

Where is EC3 used?

- **EGI-ACE** Use cases:
  - ENES Data Space:
    - https://enesdataspace.vm.fedcloud.eu/
    - Kubernetes elastic cluster + JupyterHub
  - Protein pK a and isoelectric point calculations
    - https://pypka.org/
    - SLURM elastic cluster + NFS

- **EOSC-Synergy** Use cases:
  - SAPS
    - https://www.eosc-synergy.eu/supporting-science/saps/
    - Kubernetes elastic cluster + NFS
  - MSWSS
    - https://www.eosc-synergy.eu/supporting-science/mswss
    - SLURM elastic cluster + Galaxy

Demo

Let's test the EC3 CLI !!!

```
$ sudo apt update
$ sudo apt install -y python3-pip
$ sudo pip3 install ec3-cli
```

https://github.com/grycap/ec3

More Information

Video tutorials and demos in YouTube:

https://youtube.com/playlist?list=PLgPH186Qwh_1IOesmaTLjd35Q-QqdWf9k

EC3 AoD portal:

https://servproject.i3m.upv.es/ec3-ltos

EC3 in EOSC Marketplace:

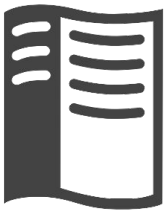
https://marketplace.eosc-portal.eu/services/elastic-cloud-compute-cluster-ec3

EC3 official documentation:

https://ec3.readthedocs.io/en/latest/

EC3 source code:

https://github.com/grycap/ec3



Which tool should I choose…?

Situation	IM		EC3	
	IM Client	IM Web	EC3 CLI	EC3aaS
… if I need a cluster which size is fixed or can be changed manually?	✔	✔		
… if I need a cluster able to adapt its size to the workload automatically?			✔	✔
… if I need predefined recipes for well-known tools?	✔	✔	✔	✔ (*)
… if I need to define my own recipes?	✔		✔	
… if I need to access a wide variety of cloud providers (public, private, federated)?	✔	✔	✔	
… if I need to use standard TOSCA templates?	✔	✔	✔	
… if I want to run the tool in a docker container?	✔		✔	
… if I have to deploy a cluster from code instructions using an API?	✔ (IM REST API or XML-RPC API)			

(*) Yes, but more variety is available at the CLI version

Questions?

Contact

Amanda Calatrava
amcaar@i3m.upv.es

Miguel Caballer
micafer@upv.es

Instituto de Instrumentación para la Imagen Molecular (I3M)
Universitat Politècnica de València

@amcaar

@micafer77