
Modello Dichiarativo per l'Automazione del Cloud GARR

GIUSEPPE ATTARDI

Dipartimento CSD, Consortium GARR

Roma 30/5/2018

Workshop GARR 2018



Intent-Based Deployment

- Describe **what**, not how
 - Workflow **Engine generates execution plan** from the desired model
 - Asynchronous process that converges by computing the **differences** between the **current** and the **desired state**
- 

Benefits of Declarative Modeling

- Portability

- Models can be deployed across platforms

- Consistency

- Both physical and virtual infrastructure can be modeled

- Relationships between components

- Changes are propagated

- Automation

- Mapping model to infrastructure delegated to orchestrator

- Evolution

- Scaling up/down

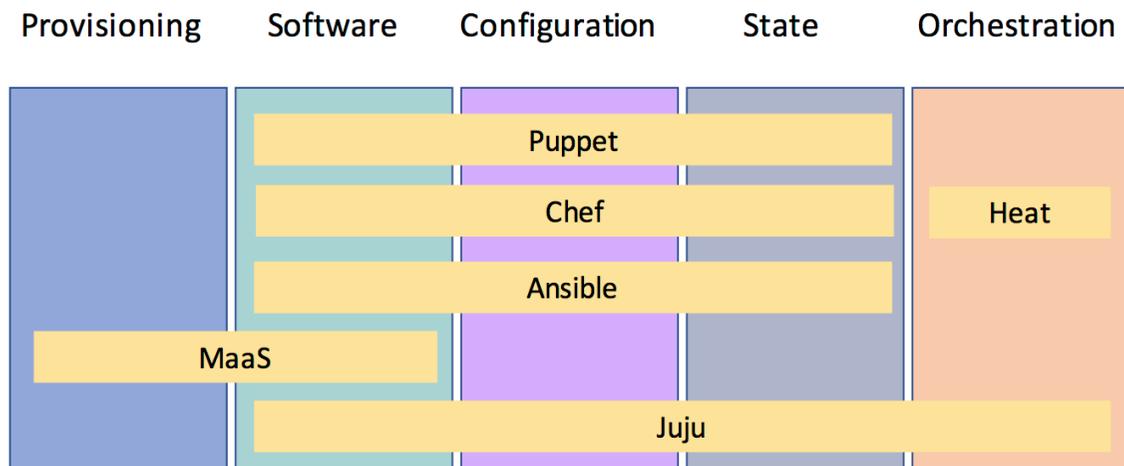
- Upgrades

- maintenance

Automazione



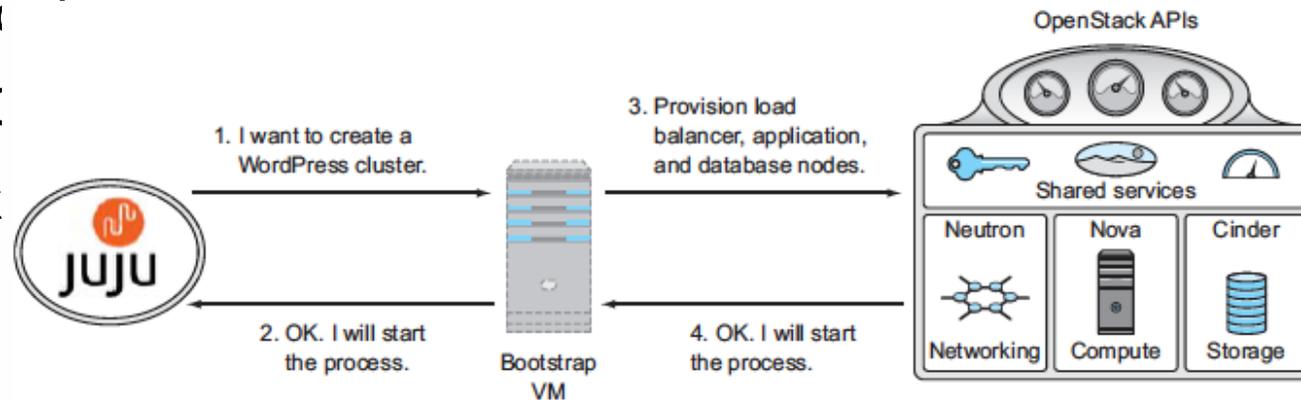
Automation Tools



Juju

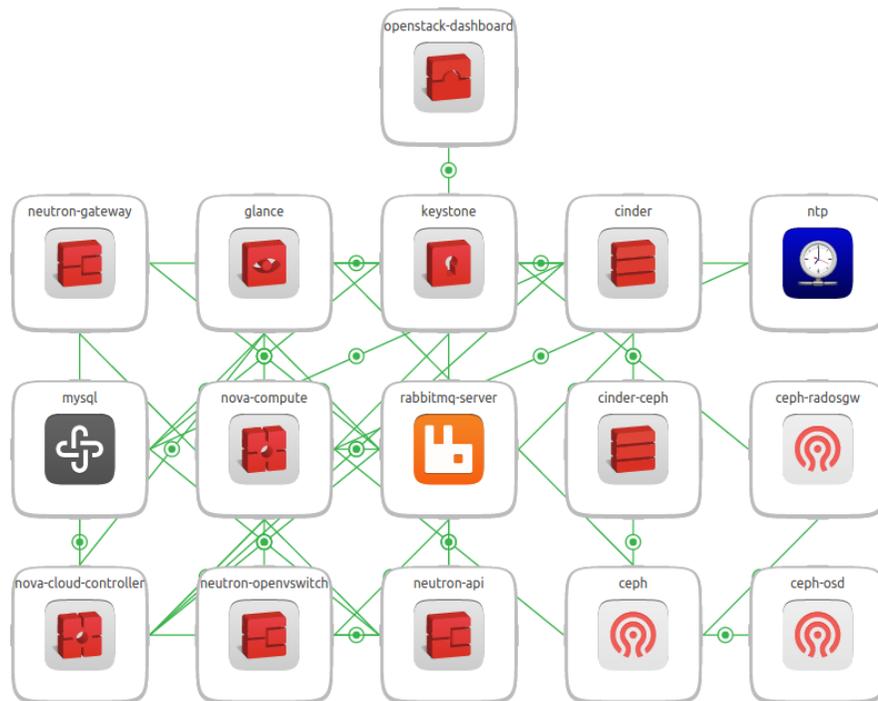
- Declarative modelling tool for composing software applications
- Charms express the steps through the lifecycle of a software component
- Performs installation on any cloud: OpenStack, AWS, Azure, Google
- Automates deployment of both OpenStack and cloud applications
- Similar to

- AWS CloudFormation
- Brook

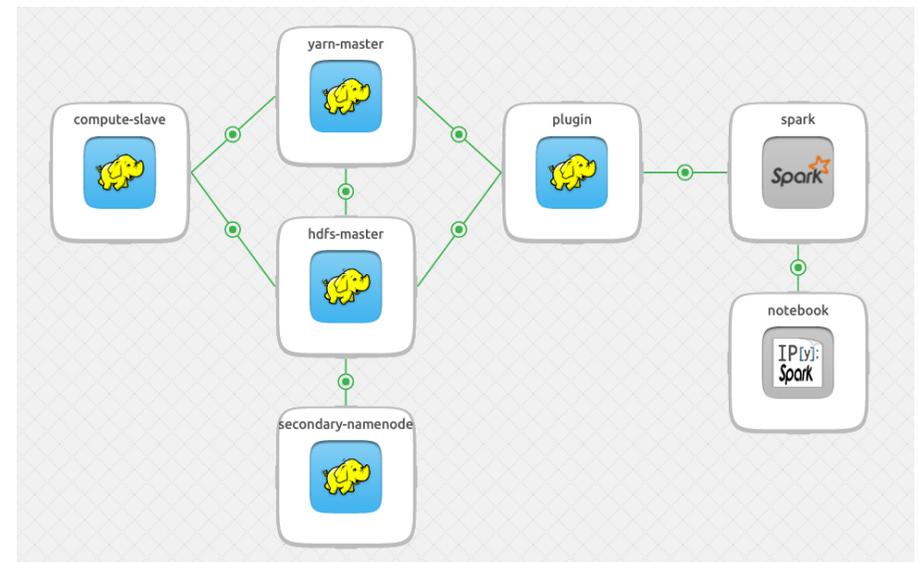


A Single Automation Tool for all Tasks

Platform Deployment: OpenStack



Application Deployment: Big Data Analytics





Deployment Duration

- Deploying an OpenStack region from bare metal
 - Half a day
 - Automated upgrade to OpenStack releases
 - Half an hour
 - Deploying a Container Platform
 - 2 days
- 

Deployment as a Service

Self-service app deployment





App Deployed on AWS

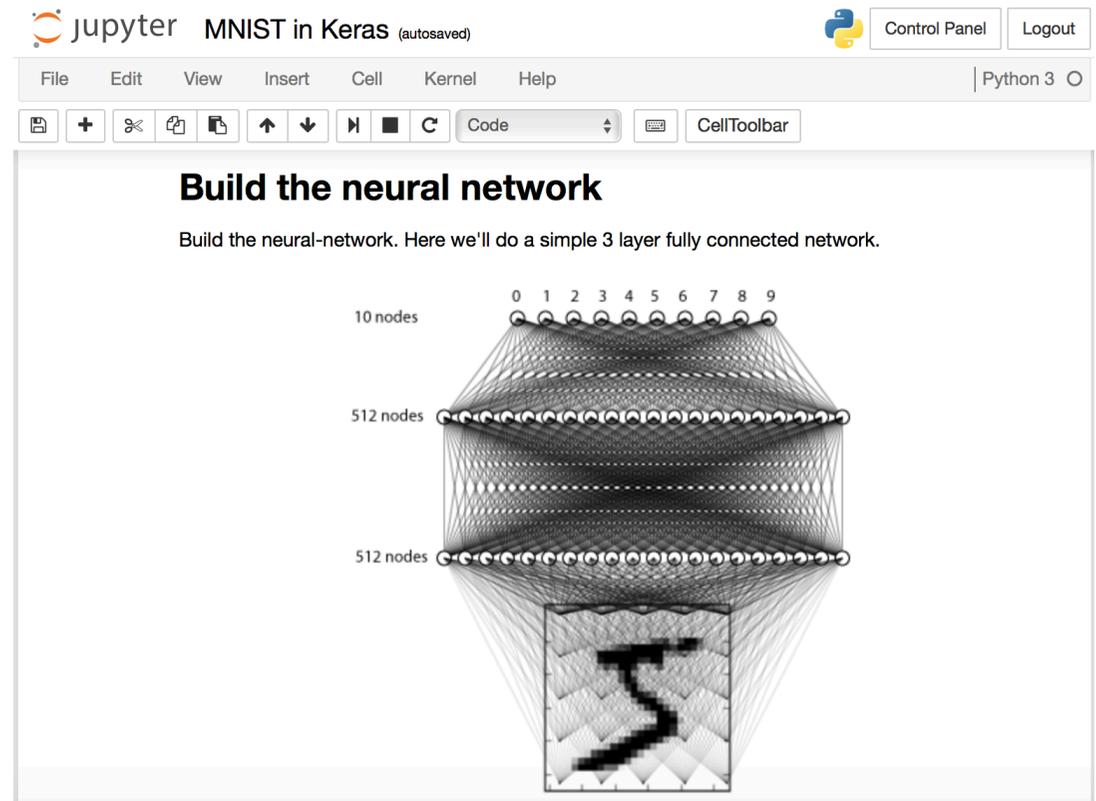
The screenshot shows the Juju web interface. At the top, it displays the user 'admin@local' and the environment 'default'. A search bar for the store and a 'Logout' link are also visible. Below the header, there are statistics for '2 applications' and '2 machines'. A list of applications shows '1 mysql' and '1 mediawiki' with a notification badge. A 'New units' button is present, along with a confirmation message: 'You have placed all of your units'. The interface also shows details for a 'default (2)' environment containing '1 unit, trusty, 1x3.5GHz, 3.75GB, 8.00GB' and a 'Root container' with 'mysql/0'.

The screenshot shows the AWS Management Console 'EC2 Dashboard'. The navigation menu on the left includes 'Events', 'Tags', 'Reports', 'Limits', and 'INSTANCES'. Under 'INSTANCES', the 'Instances' link is selected. The main content area shows a 'Launch Instance' button, a 'Connect' button, and an 'Actions' dropdown. Below these is a search bar and a table of instances.

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone
<input type="checkbox"/>		i-00f1ec62d532cea8d	m3.medium	us-east-1b
<input type="checkbox"/>		i-02d632e9b1d7b85...	m3.medium	us-east-1a
<input type="checkbox"/>	juju-controller...	i-033284f4eacc5497b	t2.medium	us-east-1a
<input type="checkbox"/>		i-09d3b4dd6ed8799...	t2.micro	us-east-1b

Jupyter Notebook Server

- [Experiment live](#) with Machine Learning and GPUs



The screenshot displays a Jupyter Notebook titled "MNIST in Keras (autosaved)". The interface includes a top navigation bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help" menus, and a "Python 3" environment selector. Below the navigation bar is a toolbar with icons for file operations and execution. The main content area features a heading "Build the neural network" and a sub-heading "Build the neural-network. Here we'll do a simple 3 layer fully connected network." Below this text is a diagram of a neural network with three layers: an input layer with 10 nodes labeled 0-9, a hidden layer with 512 nodes, and another hidden layer with 512 nodes. The nodes are connected in a fully connected manner. At the bottom of the diagram, a handwritten digit '5' is shown on a grid, representing the input to the network.

Juju Details

Typical Fabric, Ansible, Chef code

- Connect to each server
- Install packages for a web application
- Configure web application, styles and database credentials
- Connect database server XXX: NEEDS REPLICA
- Create table and populate data

Charm Interfaces

- Interfaces define how different charms can be related to share data
- One charm is the provider, like a socket
- Any char can consume the interface, like a p;ug
- Juju operates as the information exchange broker between the two charms

Repeatability

- A model can be described through a YAML file
- The model can be deployed with a single command
- The file can also contain the deployment details
 - The number of instances of an application
 - Where the applications should be located
 - Whether the applications are deployed into containers or machines
- Same model can be used for pre-production testing on a small scale and then scaled up

Managing Evolution

- From development to production
- Security updates
- Monitoring
- Log aggregation
- Certificates

Day Two Operations: Scaling and Adapting

- Scaling applications, while keeping related applications notified
- Perform configuration updates
- Relation configuration updates

Upgrades

- Upgrading complex software requires coordination between components
 - Juju provides this coordination point
- OpenStack upgrades:
 - Mitaka -> Newton -> Ocata
- Kubernetes upgrades:
 - 1.8 -> 1.9
- Upgrading the charms themselves as their functionality is improved

Lifecycle Events

- install
 - Invoked just once when the charm is deployed
- config-changed
 - Invoked whenever a configuration parameter is changed (either from the GUI, or from the CLI)
- relation-joined, relation-changed
 - When a relation is added to a charm relation-joined fires first, so that the two units can communicate with each other, and then relation-changed fires
- leader-elected
 - Occurs when many nodes require a “leader” node to coordinate among them
- pool-storage-attached, pool-storage-detached
 - Actions to take when a storage pool is attached/detached

Hooks

- Represent the handlers to be run when an event occurs
- Hooks must be *idempotent*
 - To avoid inconsistencies or divergence if run more than once

Bundles

- Bundles describe a service consisting of several charms
- They express constraints, configuration parameters and relationships between charms that provide/implement an interface
- Can be configured before/after deployment
- They provide scalability options

Juju Engine

- The Juju engine follows a reactive pattern, triggered by events that cause corresponding hook handlers to run
- Multiple handlers may match for a given hook and will be run in a non-determined order
- Running the handlers or issuing Juju commands may cause additional events
- The state engine is evaluated every time an event occurs
- The engine runs until convergence to a stable state

Actions

- Actions are executable scripts defined in the charms
- High level functionality related to the application:
 - Pause and resume replication for postgresql
 - Creating, renaming or deleting pools in ceph
- Can be executed on one or more instances of the application

Developing Bundles

- Expand a shared Catalogue of services
- Examples:
 - Moodle as a Service
 - Jupyter Notebooks as a Service

Status

● Resources

- ~9000 vCPU
- 10 PB Storage

● Usage

- Over 700 users
- Over 1000 VM

● Guarantees

- Service Continuity
- Data Protection

Usage

[Download CSV Summary \(?start=2016-11-01&end=2017-03-22&format=csv\)](#)

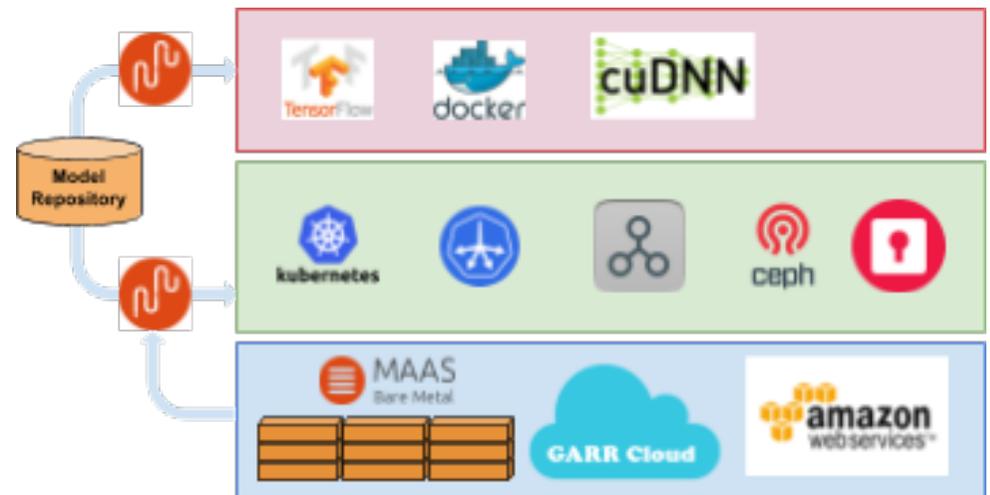
Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours	Memory MB Hours
isti	368	2.8TB	656GB	164333.47	1322184.30	279763673.66
unipa	336	7.5TB	1.2TB	693206.14	16207815.52	2692199747.8
ws2017ipv6	128	2.5TB	256GB	14955.82	299116.37	30629516.41
lns-prj1	54	543GB	71.5GB	118279.27	1629777.45	189620873.70
garrdemo318	32	20GB	32GB	16914.14	10571.34	17320083.68
INFN-FI	32	40GB	16GB	601.92	771.89	309779.80
wsosadmin	27	540GB	54GB	4510.02	90200.37	9236517.59
GEANT	25	481GB	48.5GB	29681.28	591502.48	60615623.66
infn-vlabs	24	480GB	48GB	8052.53	161050.61	16491582.24
demo	17	340GB	34GB	9877.99	197559.85	20230128.54
SSSUP	16	80GB	16GB	12651.09	63258.04	12954892.40
garrdemo125	8	100GB	128GB	13429.14	167864.31	220023110.77

Container Platform



Container Platform for AI

- Automated deployment on bare metal, AWS or other clouds by Juju
- Workloads deployed by Juju
- Distributed storage system using Ceph
- NFS cluster for sharing big data
- Docker containers managed Kubernetes



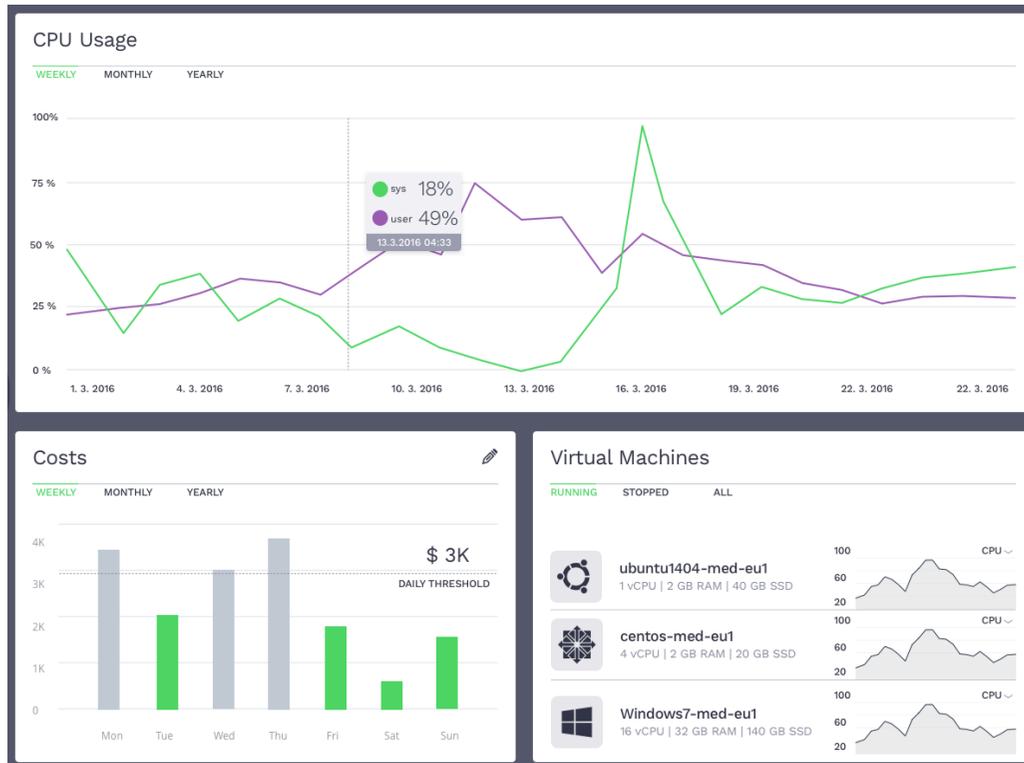
Kubernetes Deployment by Juju

- Kubernetes Nine node Kubernetes cluster with one master and a configurable number (> 3) of worker nodes.
- EasyRSA performs the role of a certificate authority serving self-signed certificates to the requesting units of the cluster.
- Etcd provides a distributed key value store: three node cluster for reliability.
- Ceph provides distributed resilient storage
- CephFS provides shared storage
- Keystone enables authenticating registered OpenStack users to the Container Platform
- Flannel provides a CNI (Container Network Interface) among the nodes

Charging and Billing



Domain Administration Dashboard



Conclusioni

Pro

- Esplicitare l'architettura desiderata
 - Livello di astrazione più alto
 - Assicura consistenza tra le parti
 - Opera su strutture anziché su file di configurazione
 - Delega al tool dei passi elementari
 - Delega al tool scelte non cruciali
 - Riduce il rischio di sviste
- 90% delle interruzioni del servizio dovuti a interventi manuali

Cons

- Conoscere strumento
- Sistemisti abituati ad operare direttamente sui file di configurazione
- Dipendenza dallo strumento e dalle sue evoluzioni
- Seguire l'evoluzione dello strumento/i
- Standards (Tosca?)

GET

INTO THE

CLOUD

WITH US

#OPENSTACKDAYITA18



IL PROSSIMO OPENSTACK DAY È A
ROMA IL 21.09.2018

SCOPRI DI PIÙ AL #MEETUP APERITECH DI OPENSTACK

14 GIUGNO H.19



14 GIUGNO H.19
#MEETUP APERITECH
@LUISS ENLABS
ROMA, VIA MARSALA 29H



Iscrizioni su Eventbrite:
<https://bit.ly/2xodHCl>