Introduction to Cloudify

Cloud Orchestration in Today's Landscape

Table of Contents

The Founding Concepts Behind Cloudify

Taking the Cloud Native approach vs. traditional management and workflow systems

In the pre-cloud world, our entire data center was built under the assumption that each organization has its own special needs, and therefore requires a tailored approach and solution for almost every challenge. We used to be very proud of how special our data center was, and even kept the way it's run very secretive, and rarely talked about it publicly.

In a cloud native world, data centers are built as agile infrastructure that is optimized for the speed at which we can release new products and features, as well as the cost it takes us to achieve this. When we optimize for these goals, being special becomes a barrier, as it results in significantly higher costs and slower processes.

IT in a Pre-Cloud World	IT in Cloud Native World
Optimized for being special	Optimized for speed and scale

This is analogous to the shift that happened in the car industry, when they moved from building custom cars to a production line like Ford or Toyota. That was a major shift not just in the way cars were designed, but also in the way the car manufacturers' organizations were structured. When we optimize for speed and cost we cannot afford silos, nor can we afford high-end devices that are optimized for the extreme scenario. Instead, we have to break silos and we have to use commodity resources.

This difference in approach leads to a completely different architecture and feature set as outlined in the table below:

Traditional Management and Workflow	Cloud Native Orchestration
Monolithic	Tool Chain
Closed Source	Open Source
Limited Scale (x100s) - rely on a centralized database	Web Scale - everything needs to scale out
Manage Hosts/Devices	Managing Infrastructure Systems and Clusters
Infrastructure Centric	Application Centric
Limited plug-ins	Future Proof

Monolithic vs. Tool Chain

In a pre-cloud world, if you wanted to provide a management solution you had to develop your own logging, monitoring, billing, alerting and any other proprietary systems, simply because there was no other way to do it. This resulted in a fairly monolithic management solution.
In a post-cloud world, we're looking for a best of breed approach where we select a tool chain that keeps on changing and growing fairly rapidly. Every DevOps group tends to select their own set of tools in this chain, which are for the most part from the open source community.

Closed vs. Open Source

In the post cloud world, open source has become a key criterion, where many of the traditional management solutions were built as closed source solutions. Contrary to what most people think, the popularity of open source isn't because its entry level is by definition free. Open source determines how well one can use or customize a given framework to their needs in areas where they see gaps. What's more, it creates a community of users who develop skill sets around these tools, allows for more natural integration between tools, and many other aspects that at the end of the day have a direct impact on the ability to achieve higher productivity and speed of innovation.

Limited Scale vs. Web Scale

Most traditional management solutions were designed to handle tens or hundreds of services and applications at best. Quite often, they are built around a centralized management solution like SQL Database, whereas in the web-scale world we need to scale to 1000s, or even 100,000 nodes in a typical environment. To reach this level of scale, the architecture of the management framework needs to be designed to scale-out across the entire stack, from the ground up. This can be achieved by separating services such as provisioning, logging, load balancing, and real-time monitoring into independent services that can scale independently. It also needs to use other scalability best practices such as message brokering and asynchronous scale-out.

Manage Hosts/Devices vs. Managing Infrastructure Systems and Clusters

Likewise, this traditional tooling was designed to manage hosts and devices, whereas modern tooling should manage more sophisticated systems such as software containers and application-level monitoring. In this world, applications have mostly been built as a layer on top of these hosts. This basic assumption starts to break when we need to manage infrastructure systems and clusters.

Infrastructure vs. Application Centric

Once upon a time, in a pre-cloud world, most management tools were designed to manage compute, storage and network services. Application management was a layer on top, and quite often, was built in as an afterthought i.e. under the assumption that the application is not aware of the fact that it’s even being managed. Therefore, the focus has been on adding management and monitoring capabilities through complex discovery, or even code introspection.

The post-cloud world tends to be more application-centric, and the management tasks begin as an integrated part of the development process. In more advanced scenarios, it is also common to use modeling languages such as TOSCA, and other similar languages to orchestrate not just the configuration and installation of applications, but to manage "Day 2" operations (i.e. post-deployment). We do so simply because that’s the only way in which we can achieve real automation, and handle complex tasks such as self-healing and auto-scaling.

Limited Plug-Ins vs. Futureproofing

In a world in which the “only constant is change” we need to be able to continually introduce new frameworks and services, those that we know as well as those that we don’t yet know exist, but are probably in development as we speak. Traditional management solutions have been known to come with a concept of plugins, but quite often these plugins are fairly limited and complex to implement, and therefore need specific support by the management solution owner.

To really be future proof, we need to be more open than that, and allow integration throughout all of the layers of the stack i.e. compute, network, infrastructure, monitoring, logging. On top of this, all that integration needs to happen without the need to modify and change the management solution. In addition, we also need to have runtime integration in which we can easily deploy applications that use a different set of cloud resources and infrastructure, or even different versions of the same underlying cloud.

All this needs to happen with complete isolation, and without the need to bring the management layer down every time that we want to introduce a new plugin. For example, a development team may have a local OpenStack cloud running in its own data center with an application that can scale-out using OpenStack resources, but when the local OpenStack deployment has insufficient capacity it can scale out into the AWS cloud.

Cloudify Built for Integration

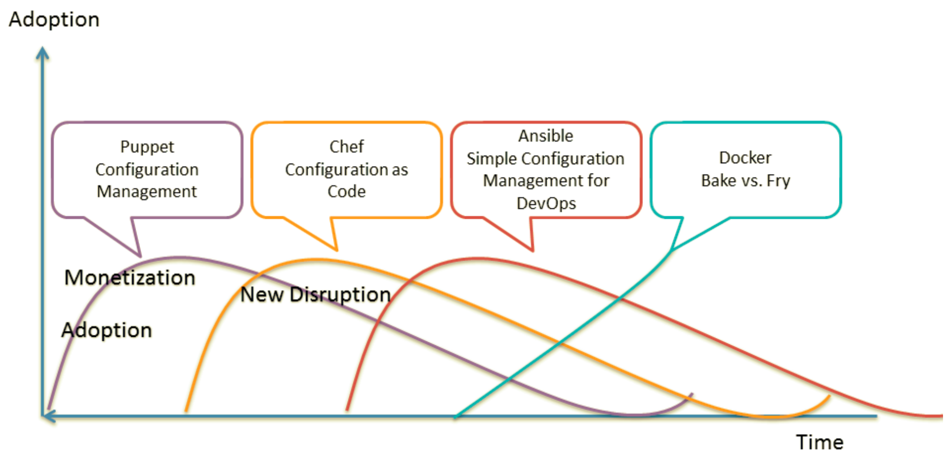
Because the Only Constant is Change

Cloud and the open source movement in general, reduces the barrier of entry for new startup companies to compete with large vendors. This results in a faster rate of innovation and thus disruption.

Historically this was directly linked to a specific choice of tooling and technology such as the VMwares, OpenStacks at the infrastructure level or Cloud Foundry, OpenShift at the PaaS layer. Containers are now disrupting all this by introducing a new class of application management tools such as Kubernetes, Docker Swarm and Mesos.

The previous attempt to cope with change, by betting on a specific technology or tool, has failed simply because many tools quickly become obsolete with the next wave of disruption.

Disruption →Adoption → Monetization
Vicious Cycle

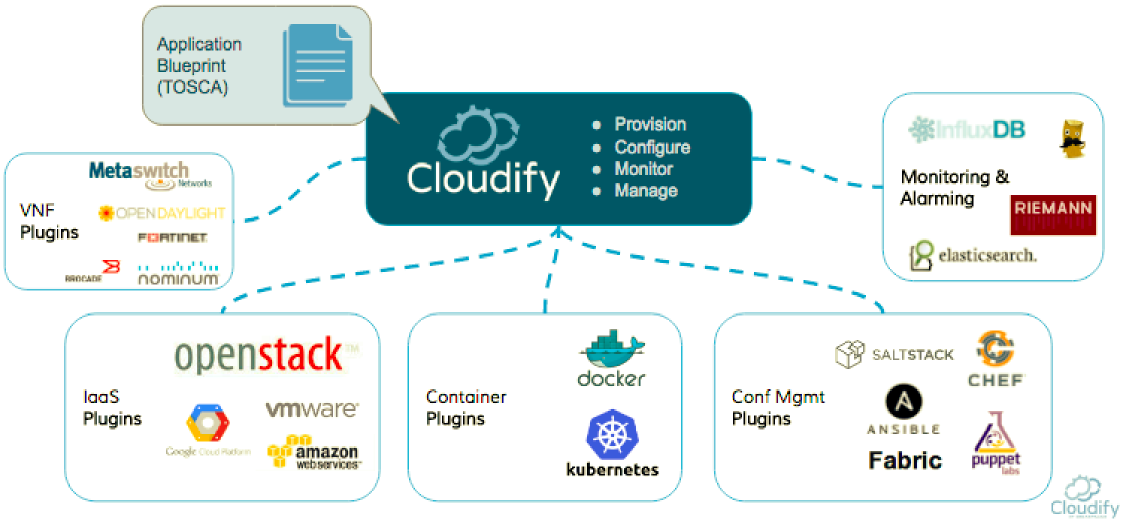

Adoption
Puppet Configuration Management
Chef Configuration as Code
Ansible Simple Configuration Management for DevOps
Docker Bake vs. Fry
Monetization
New Disruption
Adoption
Time

Amazon Web Services took a far less opinionated approach and rather than serving a dedicated a choice of certain tools they learned to offer a managed version of any tool fast (that gained enough demand behind it) by applying a generic tool to manage it.

Cloudify takes a similar approach to that of AWS, and it provides a generic orchestration engine that was designed with a unique integration approach in which the Cloudify orchestrator becomes an integration hub for applying common management and automation to the large class of stacks. Yet, in contrast with Amazon, Cloudify does this not just for a particular cloud infrastructure but for any hybrid cloud and stack environment.

Cloudify's Pluggable Architecture

Cloudify is built of a core engine that takes care of the lifecycle management of an application and a set of plugins that provides the integration points to the various components that need to serve this application. This includes the infrastructure elements (Compute, Storage, Network) up to the logging and monitoring components.



Cloudify's plugin approach serves as the core resource library in the Cloudify architecture.

Technically speaking, the plugins are responsible for providing declarative access to any API endpoint. There is no real limit as to the type of resource that can be included in that list. The typical ones include the various cloud endpoints (OpenStack, VMware, AWS, Azure, GCP) as well as network plugins, monitoring plugins. Other orchestration tools can also be plugged in a similar way by exposing their northbound API through a Cloudify plugin, allowing modeling of applications that integrates with other orchestration tools as part of their lifecycle. A good example in this regard is the Kubernetes or Docker Swarm plugins.

Cloudify is leading this open source disruption by being the first open source orchestrator to enter the conservative NFV market and by being a founding member of Open-O, as well as a pioneer in a real world application of the TOSCA specification. Cloudify is also a member in standard bodies such as ETSI, MEF and OASIS and the leading company behind Apache ARIA (an open governance reference implementation for TOSCA).

Cloudify is working on building a community and catalogue of technology partners to deliver on the promise of integration and interoperability being deeply integrated with both VMware and OpenStack alongside some of the leading DevOps tooling. This enables enterprises and telcos to build best of breed stacks - and not pre-dictated and vendor-specific stacks, through the diversity of blueprints and infrastructure integrations.

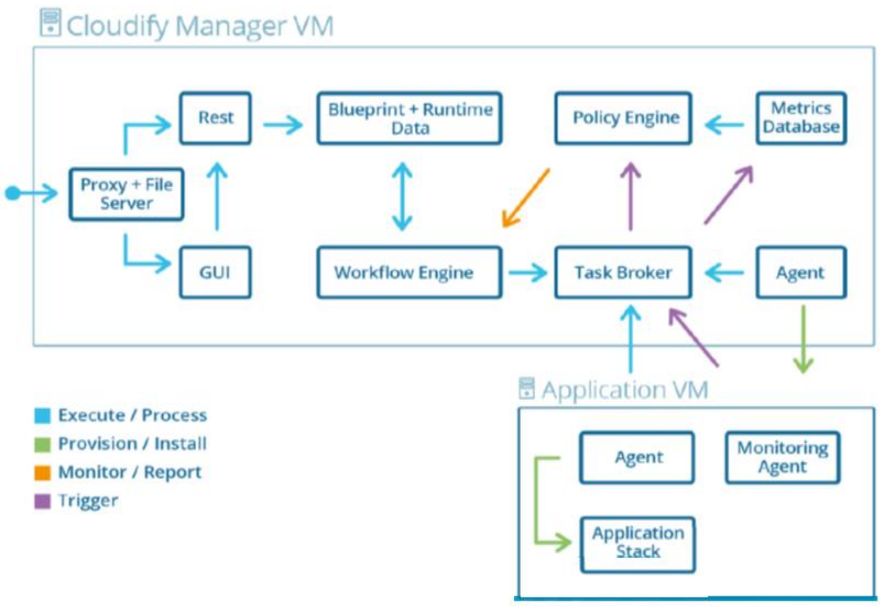
Cloudify - Built for Scale

Once upon a time, deploying a large scale application required a large IT team to provision and organize resources, for example to buy machines, set up hosting, guarantee network bandwidth, and so forth. All this before the actual application code could even be deployed to the machines, and the whole thing could be started up.

The cloud world, and the DevOps tools that have sprung up around it, have simplified this process significantly, although some of the fundamental issues surrounding large scale deployments have still not changed much to date.

Obviously, building an application that can handle 1,000 concurrent users is very different than building an application that can handle 100 million concurrent users, and with the growth of users and data on an exponential scale, applications these days, just can't afford to not be able to meet the load. It's not just the raw computing and storage powers needed to handle such volumes, it's the system's inherent design and algorithms that must "scale" to meet the requirements, as well.

Cloudify brings network virtualization and cloud native approaches together to leverage many of the best practices and tooling developed to meet web-scale deployments and apply them into enterprise architecture as well as network services.



To allow massively scalable deployments, the Cloudify architecture is constructed of simple yet very scalable components starting from the message bus all the way through the web framework behind the REST API, datastore, task broker, and workflow engine. Each of these can separately fully scale-out to meet the requirements of a large deployment, and can be deployed in a manner that matches the performance requirements at such a scale. This also includes a built-in metrics database and a logging service to enable an immediate visualization of deployments.

With this architecture a single Cloudify manager can manage thousands of node instances (i.e.: applications/VNFs installed on VMs). In addition, Cloudify managers can be clustered into a group of managers which allows infinite scale.

A Single Manager Handling Many Clouds or Data Centers

Cloudify uses a fine-grained isolation level between each node and deployment instance which allows a composition of nodes from the same type but with different versions or different types to be part of the same deployment. A good example is instances that map to different versions of OpenStack or to different instances of OpenStack. Another example is a deployment that includes a database running on VMware and a web frontend running on Amazon or Azure.

In addition, a manager is kept completely isolated from the instances that it manages. This allows users to upload new types of deployments to an existing manager without having to change the manager itself.

Enterprise and Carrier-Grade

Built from the bottom up with enterprise and carrier-centric considerations in mind – Cloudify has focused its resources on building a solution that delivers stability, fault tolerance, and security all based on a leading standard (TOSCA). On top of this, Cloudify works closely with additional industry standards to ensure interoperability (ETSI, MEF, TMForum, Linux Foundation, Apache Foundation, Open-O, Oasis Foundation, and many more), as well as technologies to build a fully integrated and interoperable ecosystem (see our Technology and VNF Partners).

In order to deliver on the promise of enterprise-grade robustness, Cloudify has built-in detection of failure, enabling corrective action in real-time. While, being infrastructure agnostic makes it possible to build highly redundant and available architectures based on hybrid models and stacks, including non-virtualized and physical infrastructure, supporting remote synchronization in a reliable and flexible manner.

When it comes to security, Cloudify is able to comply with all existing internal security practices and measures in place across networks, ports, databases, servers, and add its own layer of security through a number parameters, namely securing access to the manager's REST service, which is the only access point of clients to the management server (i.e. clients communicate with the manager by sending http(s) requests to the REST service, that processes these requests and communicates with its internal management components [e.g. RabbitMQ, Elasticsearch]). Cloudify inherits all of these components security features, and add its own security profile layer on top that relates to the REST service and UI.

In addition, Cloudify offers out of the box SSL for enhanced client-server communication, and manager access control through authorization and authentication, alongside auditing capabilities. On top of this, Cloudify provides an added layer of security of the server that the manager itself is running on, which can also be controlled via security groups and access control lists (ACLs) - providing a fully secured and a managed solution that is intended to complement existing IT security best practices.

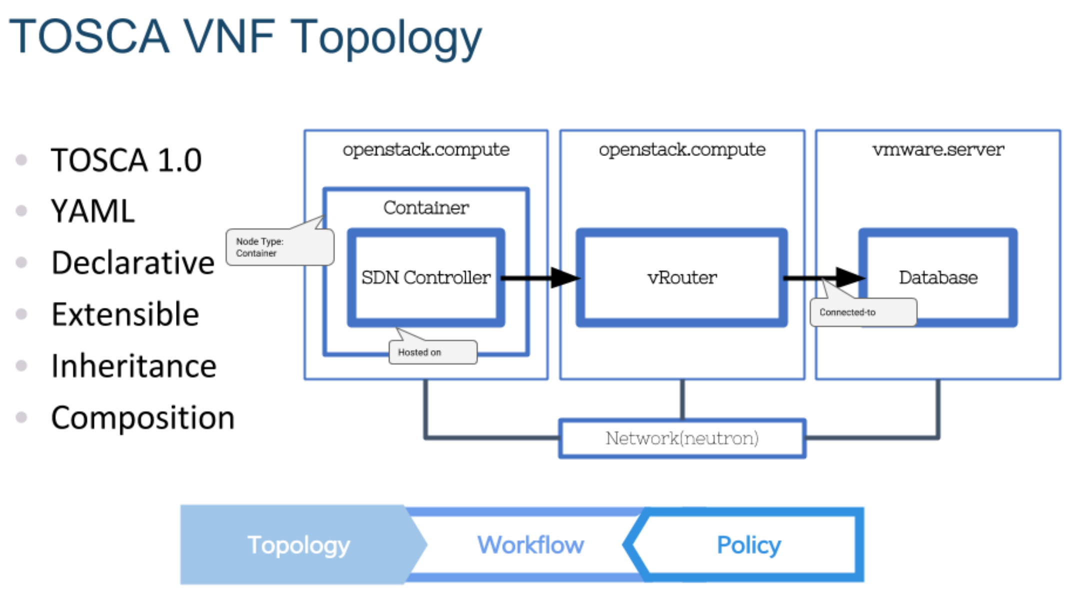
Orchestration Engine

The Cloudify orchestration engine is built out of three main components Topology, Workflow and Policy engines.

Topology

The topology engine is responsible for modeling the service components (nodes), their relationships, and lifecycle operations.

Cloudify uses TOSCA as a standard topology templating language in YAML.

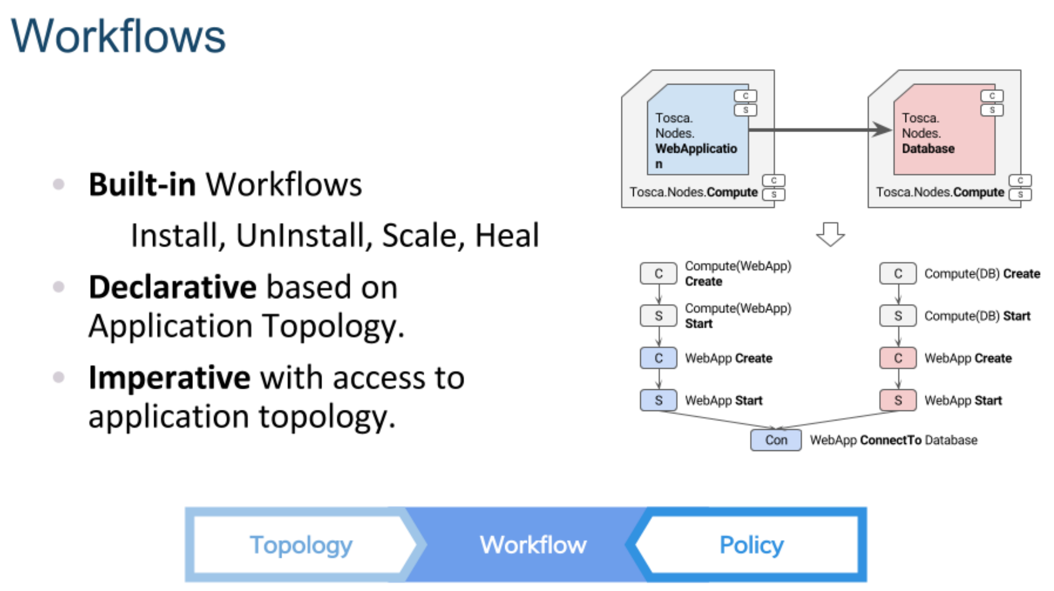


The diagram above provides a visual representation of a typical NFV service running on OpenStack that itself is comprised of three services that runs on OpenStack.

Workflow

The workflow is responsible for the execution of the operations that are described in the TOSCA graph. Cloudify comes with a set of generic workflows such as the *Install* workflow that is responsible for executing the lifecycle operations (configure->start,..) according to their dependency graph. The *Uninstall* workflow does the reverse operation, the *Scale* workflow duplicates a specific node or a group of nodes from the graph, and is also responsible for wiring them up after the execution is completed. The *Auto-heal* workflow is responsible for recovering a failed node by executing the lifecycle operation associated with that specific node.

Custom workflows are a common pattern to allow continuous interaction with the deployment graph. A typical case would be a continuous delivery workflow such as blue-green where an update workflow would first need to take a snapshot of the current state before it applies changes, and in a case of a failure will roll back the deployment to the previous state.



Workflows

- **Built-in** Workflows

  Install, UnInstall, Scale, Heal
- **Declarative** based on Application Topology.
- **Imperative** with access to application topology.

The diagram below shows how the topology and workflow interact with each other.

The TOSCA based topology is provided as a YAML file input known as blueprint. The blueprint describes the desired deployment. The Cloudify parser breaks that blueprint into a deployment graph which is basically an object graph that is also stored in a persistent store.

The Cloudify workflow is basically Python code that references the object graph. Workflows are defined as part of the deployment graph.

The execution of the workflow can take place after the blueprint has been uploaded and an instance of a deployment has been created.

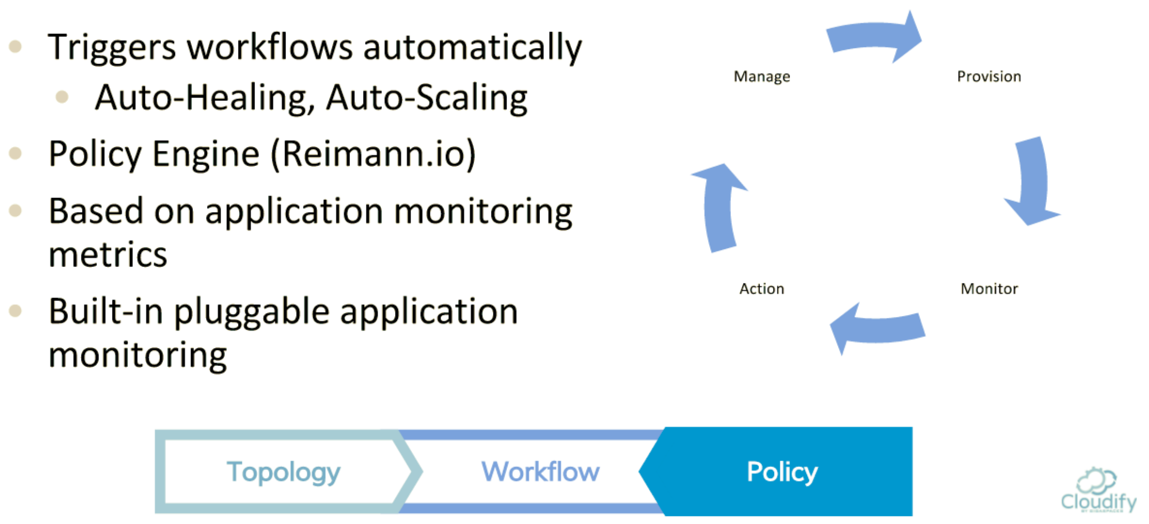
Cloudify Manager

Cloudify Manager is responsible for managing concurrent executions of multiple deployments.

Each blueprint can have many instances with a different set of inputs. A typical use case would be the deployment of multiple instances of the same blueprint for dev, QA, and production.

Policy Engine

The Cloudify policy engine is basically a complex event processing engine that listens to all the Cloudify events and correlates those metrics over a window of time to trigger a workflow.
A typical policy would be auto-scale and auto-heal.

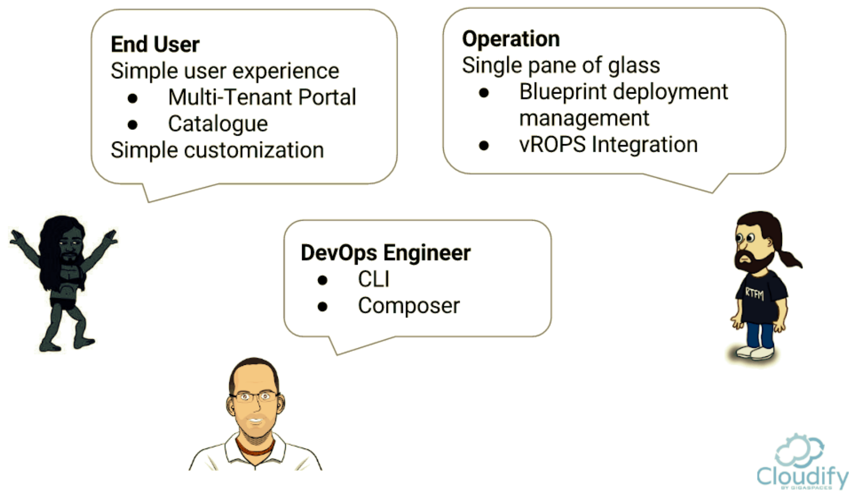


- Triggers workflows automatically
  - Auto-Healing, Auto-Scaling
- Policy Engine (Reimann.io)
- Based on application monitoring metrics
- Built-in pluggable application monitoring

Management and Monitoring UI

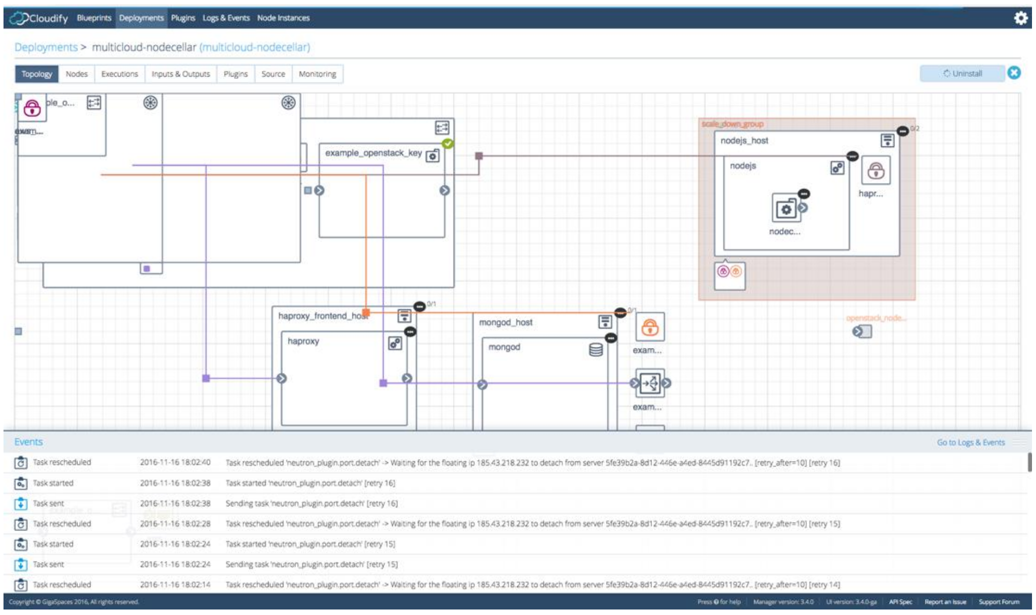
Many of the traditional management tools cater mostly to operations users, and are focused on showing flashy GUIs and reports. These tools fail to address developers who are becoming key stakeholders in the system. This simply because management and monitoring now need to be integrated more intimately into the application itself, and therefore be part of the development cycle to address things like auto-scaling, continuous deployment or self-healing.

For this reason, Cloudify provides a management framework that addresses both operations and developers, as it has become clear that only through one unified tool is truly possible to gain full control.



Deployment Management UI

The Cloudify management and monitoring UI provides a web interface to control and manage service deployments. This includes a topology view which provides a unique view for monitoring the application dependencies, as well as deployment status.


Cloudify Blueprints Deployments Plugins Logs & Events Node Instances
Deployments > multicloud-nodecellar (multicloud-nodecellar)
Topology Nodes Executions Inputs & Outputs Plugins Source Monitoring
Uninstall
example_openstack_key
scale_down_group
nodejs_host
nodejs
nodec...
hapr...
haproxy_frontend_host
haproxy
mongod_host
mongod
exam...
exam...
Events
Go to Logs & Events

```
Task rescheduled   2016-11-16 18:02:40   Task rescheduled 'neutron_plugin.port.detach' -> Waiting for the floating ip 185.43.218.232 to detach from server 5fe39b2a-8d12-446e-a4ed-8445d91192c7.. [retry_after=10] [retry 16]
Task started       2016-11-16 18:02:38   Task started 'neutron_plugin.port.detach' [retry 16]
Task sent          2016-11-16 18:02:38   Sending task 'neutron_plugin.port.detach' [retry 16]
Task rescheduled   2016-11-16 18:02:28   Task rescheduled 'neutron_plugin.port.detach' -> Waiting for the floating ip 185.43.218.232 to detach from server 5fe39b2a-8d12-446e-a4ed-8445d91192c7.. [retry_after=10] [retry 15]
Task started       2016-11-16 18:02:24   Task started 'neutron_plugin.port.detach' [retry 15]
Task sent          2016-11-16 18:02:24   Sending task 'neutron_plugin.port.detach' [retry 15]
Task rescheduled   2016-11-16 18:02:14   Task rescheduled 'neutron_plugin.port.detach' -> Waiting for the floating ip 185.43.218.232 to detach from server 5fe39b2a-8d12-446e-a4ed-8445d91192c7.. [retry_after=10] [retry 14]
```

Copyright © GigaSpaces 2016, All rights reserved. Press ? for help | Manager version: 3.4.0 | UI version: 3.4.0-ga | API Spec | Report an Issue | Support Forum

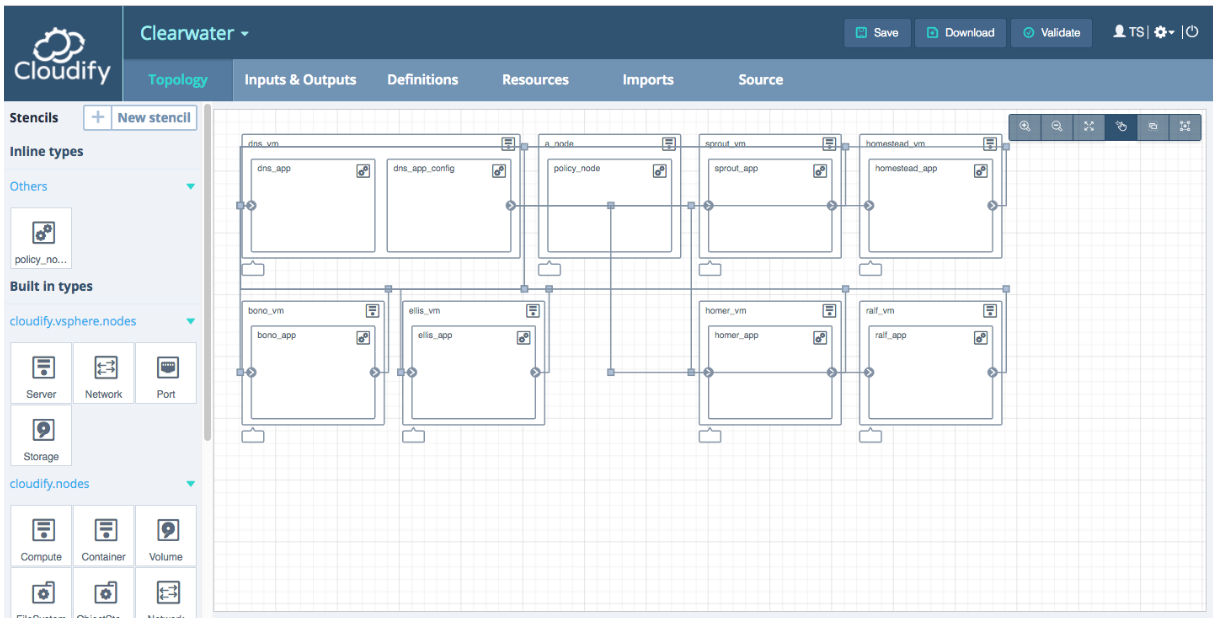
The logging view provides a way to correlate logging events for a specific application tier, alongside the metrics view which provides real time application monitoring.

Designed for Customization and White Labeling

The Cloudify UI was designed with customization in mind in which each screen can be embedded within another UI by applying a single URL. In addition, the Cloudify UI framework will include widget framework that will allow even higher degree of integration and componentization.

Blueprint Composer

The Cloudify Composer is targeted for the blueprint developer. It provides an “IDE” like development interface that makes it easy to design a blueprint through a drag & drop stencil.



Management CLI & API

DevOps process interact with the management interface through API and CLI. The Cloudify CLI and API provide full access to all the functionality that is exposed in the UI including workflow execution.

Managed & Unmanaged Execution

A typical development cycle is an iterative process.
To make the experience simpler Cloudify provides a special model through the CLI that allows direct execution of blueprint without a manager. This mode is useful for handling both the development cycles but is also used to allow execution of blueprints by other managers.

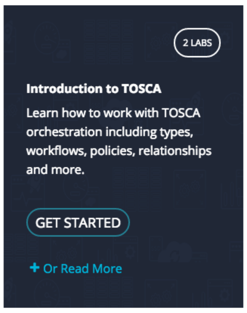
Final Notes

No matter what your use case – hybrid clouds and stacks that all need to communicate through a single management nucleus, frustrated operator syndrome of accumulating too much technology in siloes that are a management nightmare, Enterprise or Telco NFV, simplified migrations paths – Cloudify was built to power this revolution in IT delivery.

By betting on orchestration first, and not a specific tool or technology, you can future proof your organization against the disruptions that are already here, as well as those that are yet to come.

Where Should I Go From Here?

Cloudify Academy provides diverse self-learning tools, from links to real working demo environments, documentation, through video tutorials, or even training services, that enable you to learn how to use Cloudify as you go. You can also just download Cloudify or launch a pre-packaged image, and get started.

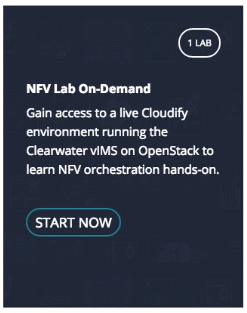

2 LABS

Introduction to TOSCA

Learn how to work with TOSCA orchestration including types, workflows, policies, relationships and more.

GET STARTED

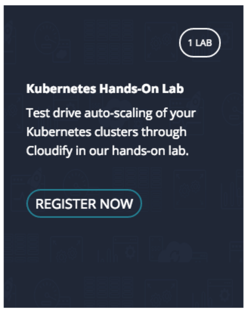
+ Or Read More


1 LAB

NFV Lab On-Demand

Gain access to a live Cloudify environment running the Clearwater vIMS on OpenStack to learn NFV orchestration hands-on.

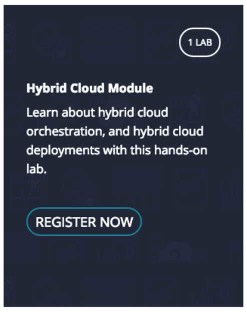
START NOW


1 LAB

Kubernetes Hands-On Lab

Test drive auto-scaling of your Kubernetes clusters through Cloudify in our hands-on lab.

REGISTER NOW


1 LAB

Hybrid Cloud Module

Learn about hybrid cloud orchestration, and hybrid cloud deployments with this hands-on lab.

REGISTER NOW

Additional Resources:

- Cloud Management in the Enterprise: An Overview of Orchestration vs. PaaS vs. CMP
- From Virtual Appliance to Cloud Native VNF
- What A No Compromises Hybrid Cloud Looks Like
- The Road to Cloud Native VNFs

GIGASPACES

www.getcloudify.org