Application Defined Governance

Automating application IT governance through orchestration to enable increased control without compromising on agility

hello@cloudify.co | cloudify.co

Cloud infrastructure provides great power and flexibility. Every developer can now create applications, spawn VMs, and create their own networks at the call of an API.

This easy access to resources has created a whole new set of challenges for enterprises as they transition to the cloud:

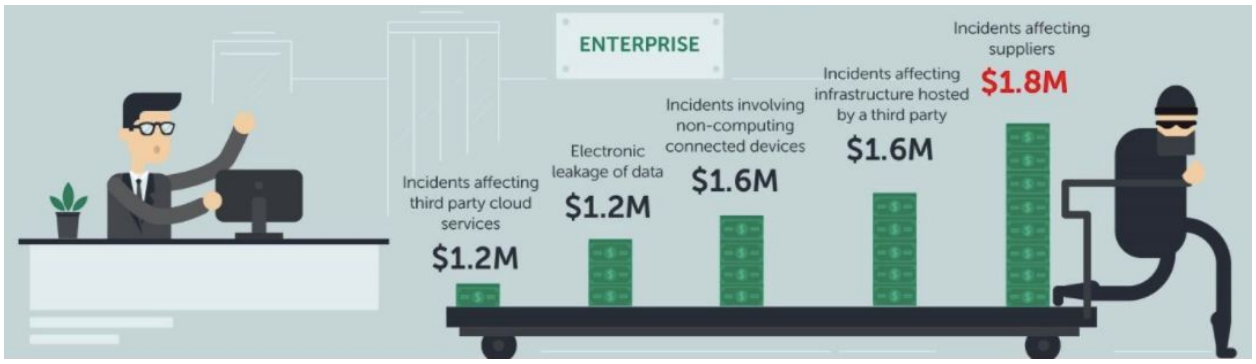
- How to control their user access to cloud resources
- How to control where the data of their application resides
- Which images are used by their users and whether they expose security backdoors

Multi-cloud is now a reality in the typical enterprise, as indicated in one our recent survey which makes these risks and vulnerabilities even greater than ever before.

The lack of good cloud governance and security solutions leads to huge costs according to the report *'IT Security: cost-center or strategic investment'*,

> Enterprises lost nearly two million dollars ($1.8M) as a result of breaches affecting suppliers that they share data with, and **$1.6M because of IaaS-providers' insufficient levels of protection.**

Governments, now well aware of such challenges, introduce new legislation and regulations requiring organization to provide information about how they share and protect personal data.



Current Approaches to Cloud Governance

In a pre-cloud world much of the governance process was applied through explicit delegation of responsibility to a trusted group - usually central IT. The issue with that approach was that central IT quickly became the bottleneck hindering agility, creating a culture where there was a

need to bypass this bottleneck, which was a big catalyst for the onset of cloud in the first place. Sounds like we're going full circle.

The question then becomes how can organizations impose governance and control over how the infrastructure is used by the different users and roles, **without sacrificing organizational agility**?

I will start by examining the challenges with current approaches for achieving governance and control over the way users gets access to the infrastructure resources. I will focus specifically on user control (RBAC), as well as Quota and Security Management.

In the second part i'll suggest a new approach addressing the control vs agility conflict through automation. The general idea is apply the same automation principles used for automating the application build and release cycle a.k.a DevOps to governance process.

Challenges with Quota and Resource Management

Quota and resource management are only a good first step for allowing better control over who and to what extent users get access to infrastructure resources. Quota and resource management is the concept where each user is granted access to a specific infrastructure resource with a limited quota per resource.

This approach defines the boundaries in which users can operate, providing a certain degree of control, however it doesn't really address how these resources should be used in the context of a specific application. Therefore, many times these boundaries are quickly expanded to accommodate the capacity of most demanding use case and as a result makes those boundaries fairly limited.

Challenges with Security Management

In order to achieve improved cloud security governance, many organizations are applying solutions largely as an afterthought, e.g. by monitoring the operating system or network to try and identify anomalies or malicious behavior. The challenge with such approaches is that quite often breaches are discovered after the fact, or even as they happen, relying for the most part on statistical heuristics.

What is Application Defined Governance?

Application Defined Governance, as the name suggests, enables automated control of the use of the application infrastructure resources based on a specific application context. For example, for a QA environment the set of resources that will be assigned will be those that are optimized

for low cost and regular-/low-priority SLA. Production resources on the other hand would be optimized for availability, where Big Data applications would be assigned high memory resources, and the web front-end would be assigned high CPU SLAs, and so forth.

Similarly, we can use the same application-centric approach for controlling where specific parts of our application will run. For example, we could decide that data services can only run in specific geographies, whereas compute services can run on a best effort basis, and the list goes on.

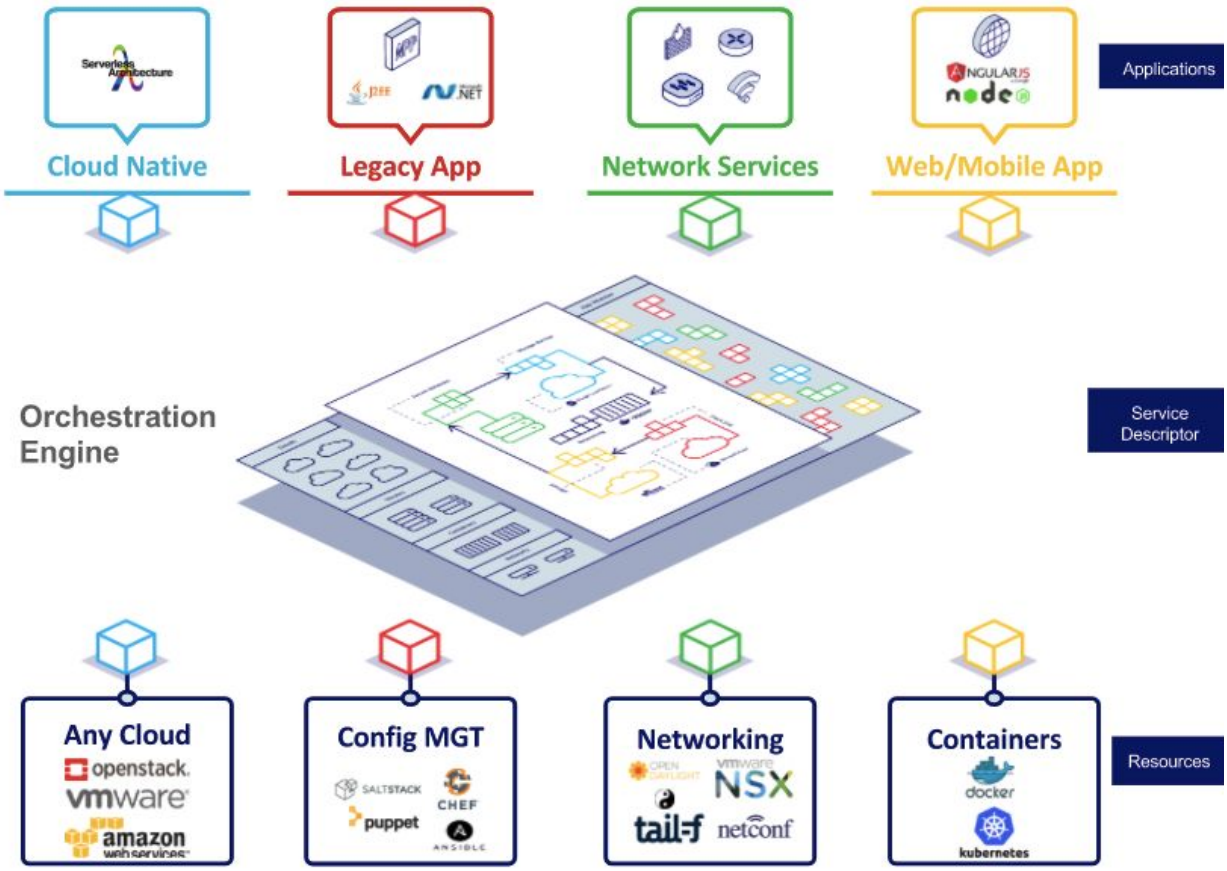
The key with Application Defined Governance is how we define the manner in which resources could, or should, be used based on a specific application context where specific control points are an integral part of the application development and management processes, and not an afterthought.

Automated Governance

There is an inherent conflict between agility and control/governance. Therefore, the same principles that have been applied to the IT world to reduce operational friction through automation, need to be applied to governance processes as well.

Similar to DevOps automation practices, the automation of governance compliance processes can be achieved through a set of automated compliance tests that each application needs to run as part of its continuous deployment process or by controlling the provisioning, configuration, scaling, and upgrade processes through a set of predefined workflow and policies.

Orchestration provides an automation platform that fits right between the user and the infrastructure, and therefore can automate many of the regulatory processes as an integral part of the application lifecycle, serving as an excellent control point to govern who utilizes the underlying infrastructure resources and how they do so throughout the different stages of the application's lifecycle. These two properties makes orchestration a great framework for Application Defined Governance.



Orchestration: to automate the regulation process as part of the application lifecycle and control how applications consume infrastructure resources

Let's dive into how this model can be achieved with Cloudify.

Applying Application Defined Governance

A good example for Application Defined Governance is the Apple App Store (or any other marketplace for that matter).

In an App Store / application marketplace only the application owner has access to the development version of the application. The developer can iterate through the development stages until the application is ready to be released. Once it is ready to be released, it is pushed into the App Store.

The App Store has a group that verifies that this application complies with App Store rules and regulations. Many of these tests are automated to avoid too much friction during that processes.

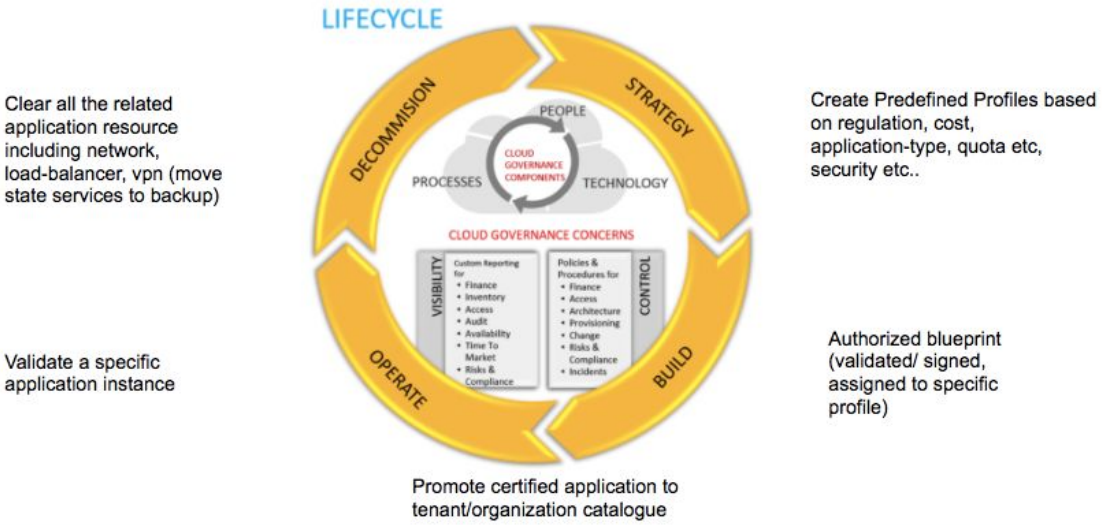
As consumers, we only get to see the end result, the finished application, and are only given access to operate it (i.e. start/stop or control specific configuration elements that the application owner decided to expose to us).

The idea behind Application Defined Governance is to apply the same principles to common enterprise cloud applications, with the same level of simplicity. In this case, the role of central IT would be analogous to the role of the App Store owner. This Admin user defines the environment, provides specific authorizations for each user role, and governs the entire process of certifying and deploying the many applications.

The application owner is analogous to an organizational power user, this can be a DevOps engineer or owner of a specific application, or business unit that is responsible for delivering the application and controlling the way it will run in a way that fits the organizational environment.

The end user can be the developer themselves of additional applications, or a regular consumer that is only interested in getting an instance of a given application in their own environment.

The diagram below describes the different stages and control points that a typical organization has in its application's lifecycle and how these concepts apply in the context of an enterprise organization or service provider.



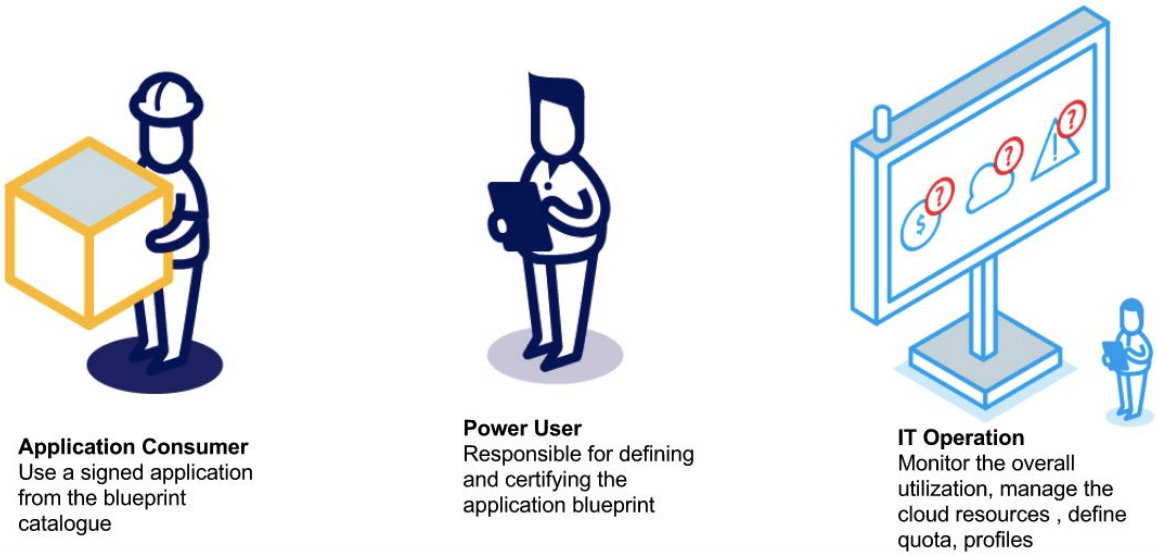
Cloud Application Governance Lifecycle

Let’s go through the these lifecycle stages.

Strategy

A cloud governance *strategy* includes the definition of the processes and roles of users within the organization in the context of the Cloud Application Governance process, as well as the classification process between users/roles/applications and such considerations.

With Cloudify, key enterprise user profiles that are involved in delivering and managing enterprise and network services have been modeled and segmented into three main personas, as described in the diagram below:



The three personas responsible for delivering and managing enterprise and network services

Build, Operate, and Decommission

In the context of Cloudify, the stages *Build*, *Operate*, and *Decommission* map directly to the stages of blueprint and workflow development, and ultimately deployment and execution.

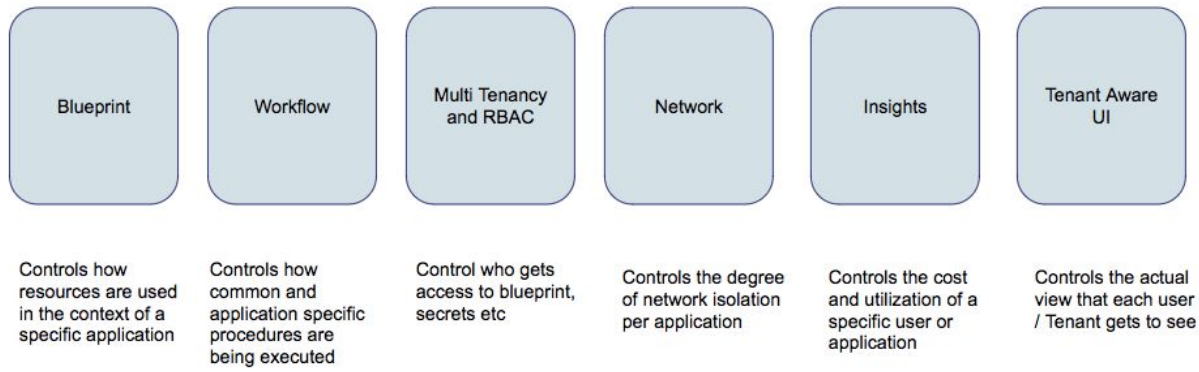
The key concept being that the blueprint essentially acts as a “software-defined contract” between the application owner and central IT. This contract provides the definitions for how the application is allowed to use the infrastructure resources, and therefore, during the build process a primary focus is on embedding the critical hardening parameters into the blueprint. Once the blueprint reaches production maturity it gets promoted for consumption through the application catalogue.

Users (devs) can operate only certified blueprints and have no access to the blueprint itself, except for the sole access enabled - usually in the form of being able to define certain parameters, known as inputs, within the blueprint. These inputs are mostly limited to application-specific parameters that need to be entered by the user, such as image id, cloud credentials, and such.

When we decide to close an application, the workflow of that specific application provides the definition of how to decommission its resources. In the case of Cloudify, this workflow is part of the common workflows and is shared across all the blueprints.

Six Pillars for Control and Governance with Cloudify

Incorporated in a Cloudify blueprint are additional components/pillars that allow broader control over various applicative operational aspects.



- The **Blueprint** is the center of everything. This is the “software-defined contract”. It controls how resources are used in the context of a specific application through the definition of relationships, requirements, and capabilities in TOSCA.
- The **Workflow** controls all of the dynamic execution aspects. Workflows provide users with the ability to create common operations such as install/uninstall that will be consistent across all applications, as well as custom workflows, such as snapshots, that will ensure consistent operations of that process in the context of a specific application.
- **RBAC**, or role-based access control, controls who gets the right to create, modify or view each blueprint or receive access to any other resource. Through RBAC we can control which users receive access to infrastructure resources and which users are only able to use them indirectly.
- The **Network** enables users to create their own private network firewall rules, VPN, and other network resources as part of the application lifecycle, and in this manner they can control the network isolation of each application.

- **Insights** provide the operator with the ability to monitor the utilization per application, tenant, or user and control the costs throughout the entire system.
- The **Tenant-Aware UI** is where users can control which widgets can be viewed by each user/tenant. For example, standard users can be granted view-only access to the blueprint catalog, which in turn allows them to only execute applications on a specific resource, where advanced users can also view the blueprint of a specific application, and add/remove applications to/from the catalog.

Managing Continuous Deployment

Once we have deployed an application there's another set of processes that control the continuous deployment of updates into the running instances of the applications.

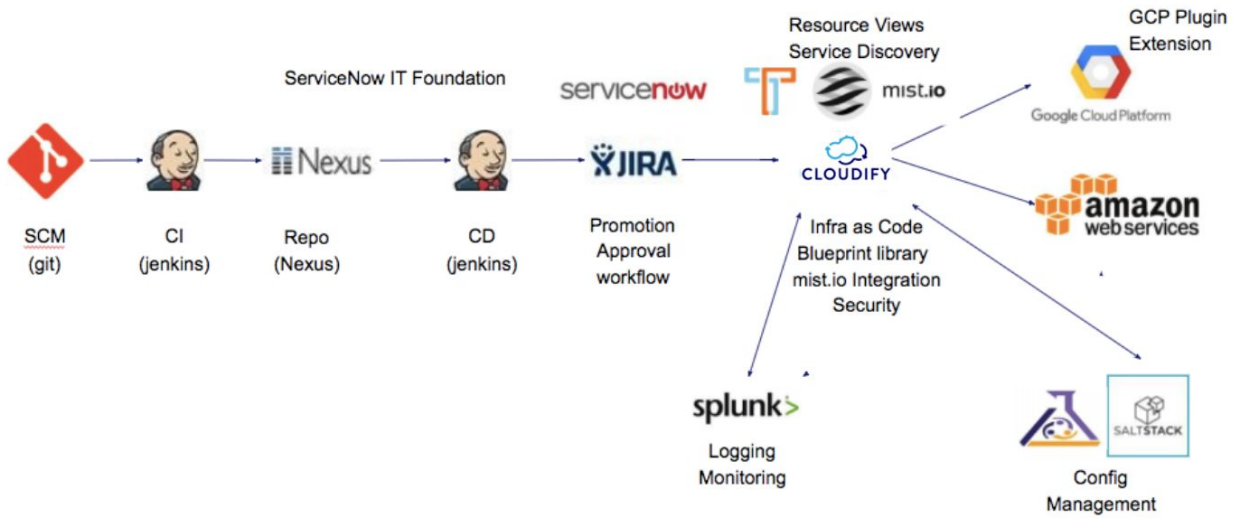
Technically speaking, this process can be fully automated, however, in many organizations the last mile of deploying the actual update into a production system is often gated. With workflows, users can easily customize whether they would like a fully automated or semi-automated process.

In this specific example, we use the same blueprint and workflow model that we used in the previous “App Store” example, only this time we will use it to control the development, QA, and production of the same application.

The RBAC associated with the user and blueprint allows us to provide different sets of control points for the blueprint creation and deployment processes. For example, a dev user has full access to create and modify blueprints, while a QA user only has view and execution rights for that same blueprint, without any modification rights.

The diagram below shows an example of a specific customer environment that utilizes Github as the source control, Jenkins for the build system, and Jira/ServiceNow for controlling the overall process.

Cloudify is used as the backend execution engine that is responsible for deploying the application, based on the blueprint definition, on the appropriate target cloud or clouds (Google and AWS in this specific case).



Continuous Deployment Example - Automation with Full Control

What makes this method valuable is that, even though there's a great deal of control required over the entire process, it can still be fully automated.

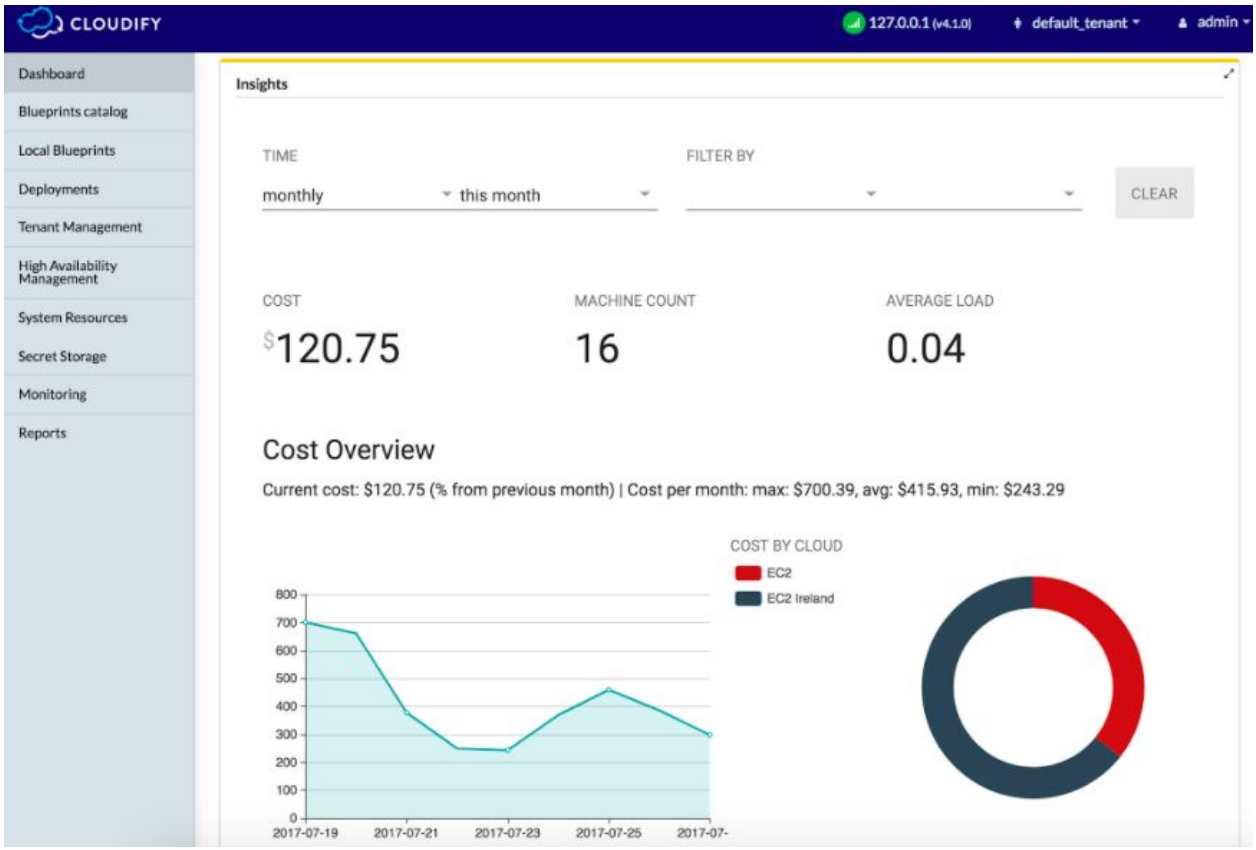
Cloudify 4.2 Enhancements

Cloudify 4.2 was built with, and for, some of the most demanding enterprises and service providers, and comes with major enhancements to enable better granularity on how enterprises and service providers can control their application governance.

Those features include:

- **Fine-grained multi-tenancy and RBAC -** With Cloudify 4.2 the same user can now have different roles per tenant to which they belong. These new roles include Tenant Manager, Tenant Viewer, and Tenant Operations, in addition to the existing User and System Administrator roles. Roles can be assigned to both individual users or groups of users. For example, an operator may be granted a manager role in *tenant-1* which belongs to his own department, and only a viewer role in *tenant-2* which belongs to a different department. Each role allows the users a different set of permissions to Cloudify resources per the tenant they are assigned to.
- **Global Resources -** Global resources such as blueprints, plugins, and cloud credentials (known as secrets) can now be shared tenant-wide or globally throughout the organization. Tenant-wide resources can be shared among all of the users in that same tenant. Global resources can be shared among all users of all tenants.
- **SSO** - Cloudify now enables SAML Single Sign-On (SSO) through integration with Okta systems.

- **Blueprint Validation** - The new two-way editing feature in Cloudify Composer (a simple drag and drop graphical blueprint composer), provides real time validation of TOSCA blueprints during the design phase, and thus reduces syntactical errors during the deployment phase.
- **Insights** - Cloudify now offers insights and analytics as a service which allows operators to monitor their machine and application utilization, as well as cost over time, and get recommendations on how they can control and reduce usage and costs for multi- and hybrid cloud infrastructure.



Cloudify 4.2 new Insights as a Service

Final Notes

The current approaches to cloud governance are very infrastructure-centric and are mostly enabled as an afterthought. They are lacking the application context on how resources should be used for a given environment such as QA or Production, or application profiles such as big data, web applications, on top of multiple roles per user.

Application Defined Governance addresses these challenges by giving the application owner and the central IT common tools in which users can control how their application consumes infrastructure resources as part of the design and development stages of the application, and in a way that fits the application's needs alongside organizational policies.

Orchestration provides an excellent control point that is shared between the developers and central IT. The application blueprint provides a software defined "contract" that afford the application owner flexibility to use the resources that they need, while simultaneously giving central IT more effective ways to govern the application through blueprint certification and control. This model makes it possible to bake in many of the governance processes at the very early development stages of the application, in order to be fully automated as part of a continuous deployment process. In this way, we gain even greater control without sacrificing agility.

Cloudify 4.2 was built and designed with the most demanding enterprises and service providers as technology partners, and comes with a comprehensive set of features that provide an end-to-end solution for delivering Application Defined Governance, starting from the blueprint design phases and workflows, through fine-grained multi-tenancy and RBAC, network security, and cost and utilization insights. All this is integrated into a single management portal which allows IT operators and users alike to control not just how an application consumes infrastructure resources, but also who has visibility to which resources.

This concept significantly simplifies formerly complex application management and governance processes, while simultaneously making them more robust and secure. Moreover, this entire process can integrate with current DevOps processes, and can be fully automated, creating a whole new way to strike the right balance between control and agility.