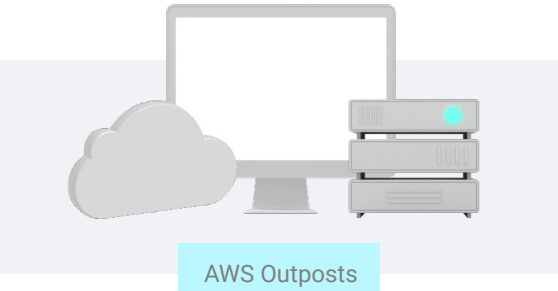
Next Generation Orchestration: Multi Domain, Sites, Clouds, Hyper Scale

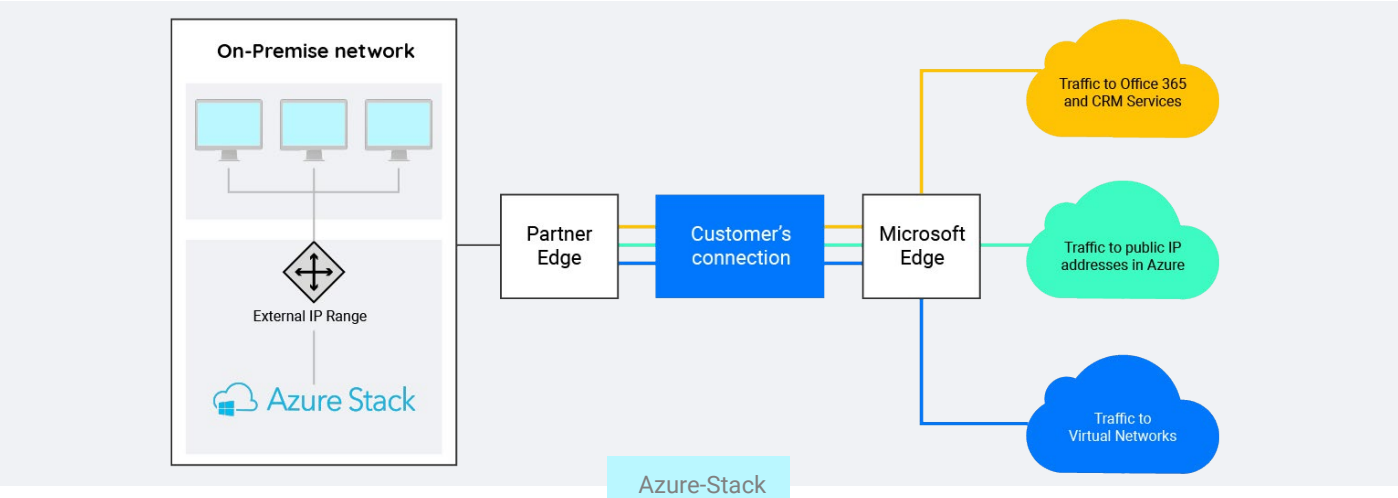
The next generation of orchestration is driven mostly by the need for scale and distribution to support the new edge and IOT use cases. This generation of orchestration needs to support not just orchestration of services within a local data-center or cloud but needs to reach across multiple sites, regions and domains. Here we'll provide an overview of the evolution of the Cloudify orchestration platform -- built to address scaling and federation challenges.

What's Your Edge?

It's no secret that most edge use cases today are largely network driven. Yet cloud providers have started to extend their offerings towards the edge with the introduction of private cloud instances (as per Azure-Stack or AWS Outposts) as well as via new IOT offerings. This new pattern is gradually changing the way we think of edge and SD-WAN today as the data-center becomes the new edge and cloud becomes the new HQ.

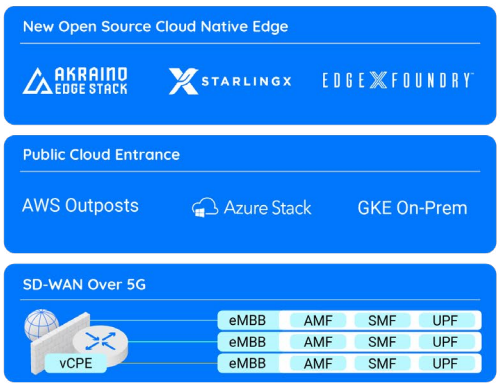


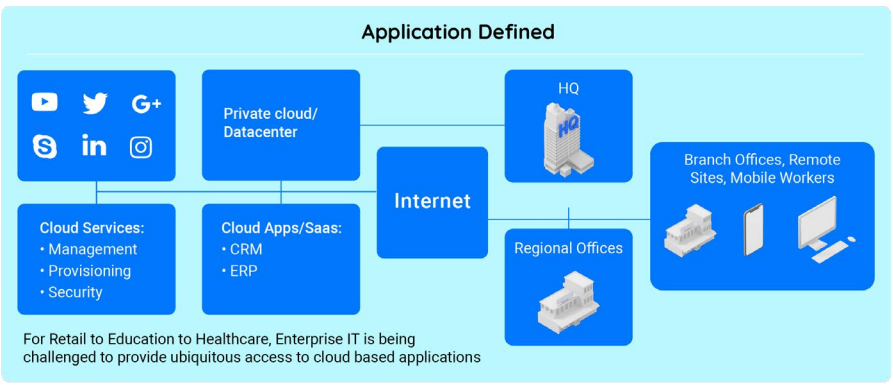
AWS Outposts

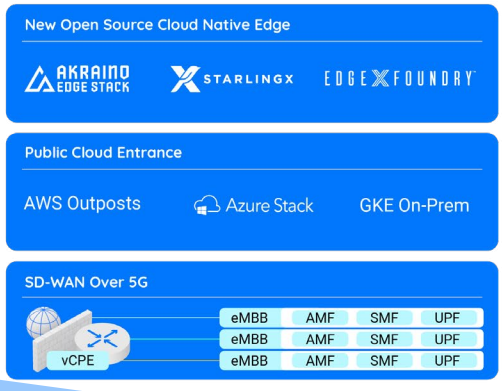


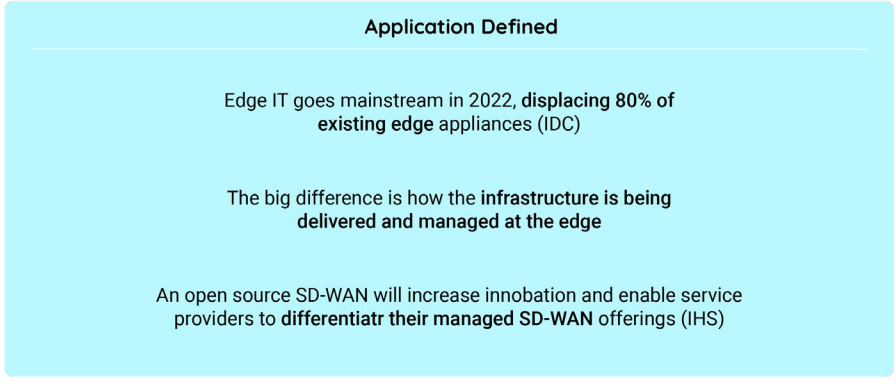
Azure-Stack

Enterprises soon will have to adopt a new and far more universal method of managing applications and services across different edge points, requiring a completely new way to handle SD-WAN and network automation- namely from an IT led model to application driven network automation that fits into the new dev-ops process.







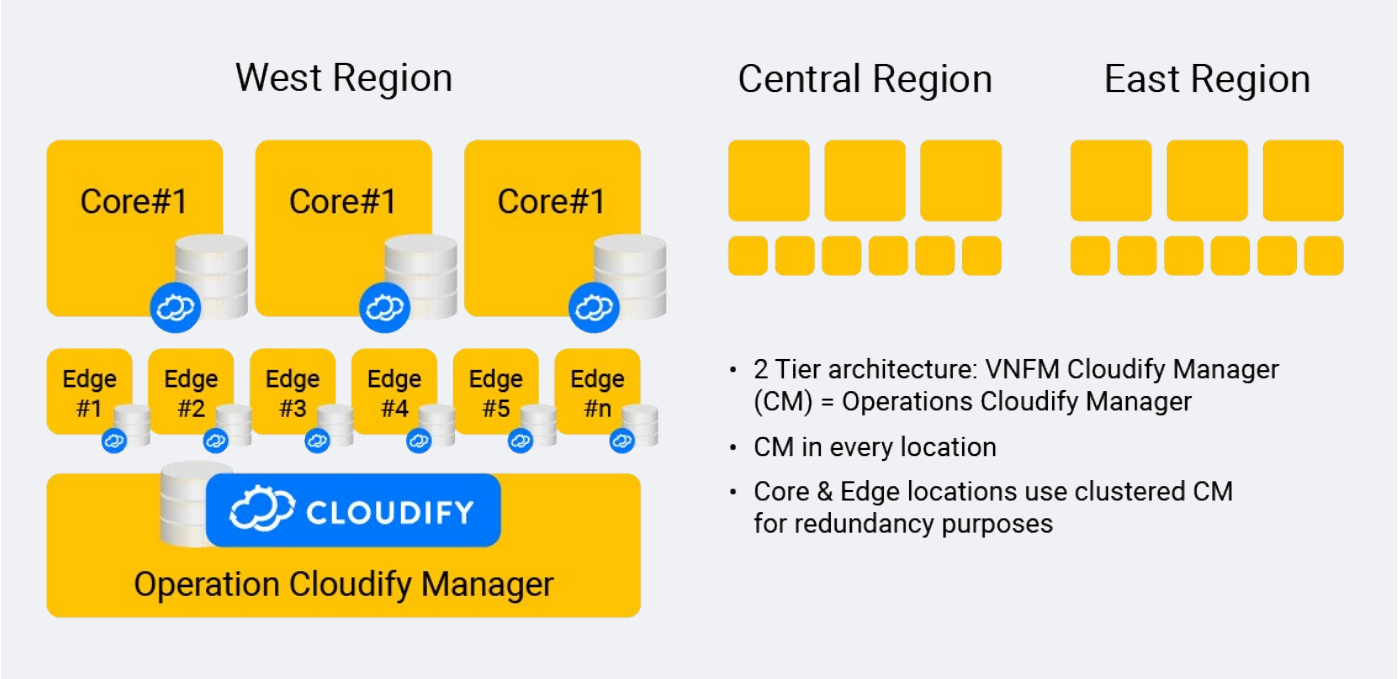


The Evolution of Orchestration

The architecture behind Cloudify Spire evolved over the past two years via a series of client use cases. The first being a satellite network company managing a fleet of ships and airplanes. Each unit held a local CPE device responsible for delivering internet communication. To allow agile control of these devices, it was decided to use Kubernetes as the local control plane and use a hierarchy of orchestrators spread across the globe, responsible for delivering the communication channels.

This was the first time that we came across the need for distributed orchestration and led to our first case study on "The Birth of an Edge Orchestrator"

Later that same year we came across another use case in which we needed to deliver a distributed IP Multimedia Subsystem (IMS) for a US operator. The IMS services needed to run across 39 locations (POP's), and it was important for each location to be completely autonomous. In other words, each site needed to manage its own set of local resources without being dependent on a central orchestrator. The role of the central orchestrator was merely to provide a single point of access to all the sites and to coordinate the workflow. This led us to create the concept of a Manager of Manager (MoM) using a central orchestrator to manage a cluster of multiple micro-orchestrators.



In another use case, one of the world's largest banks ended up using many separate Cloudify clusters to serve different parts of the organization. In some instances, each department had to be completely isolated because of regulation constraints. In this case the use of a MoM was introduced as a way to simplify the operational complexity involved in maintaining such a large number Cloudify clusters and until recently we used a custom plugin that enabled each manager to delegate deployment request to another manager. The Cloudify 4.5.5 release was the first time we provided official support for distributed orchestration (MoM) via a new plugin and UI support.

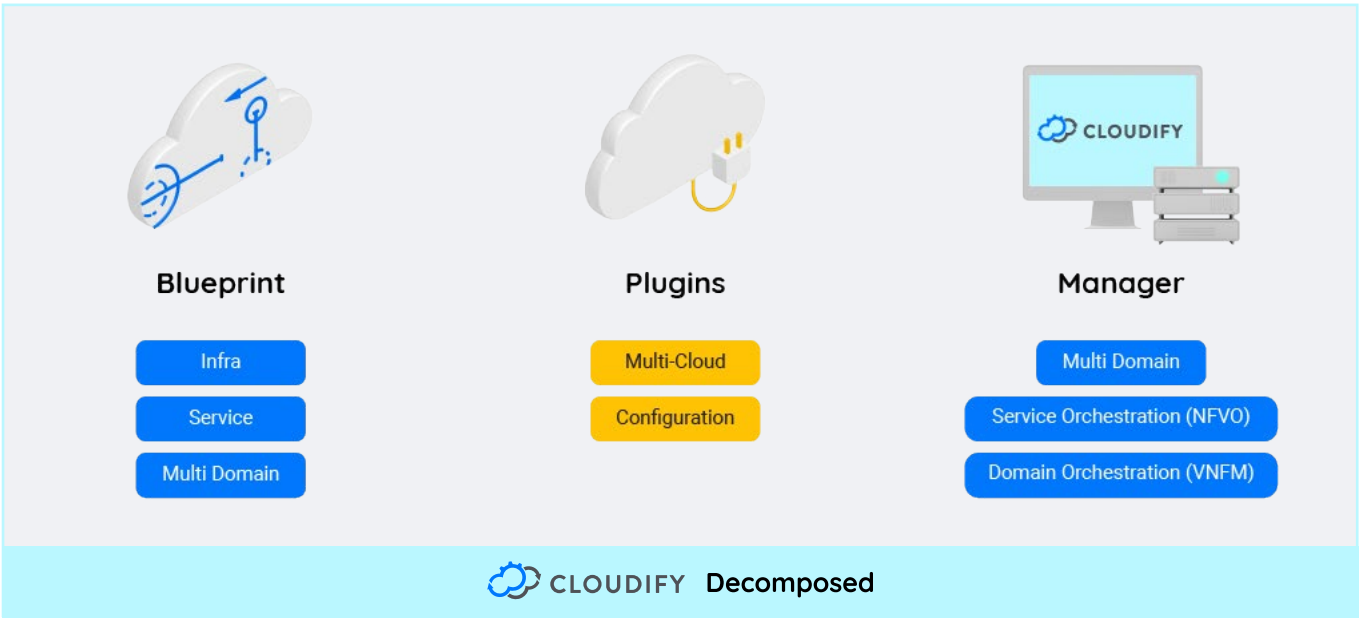
Over the past year we have seen a growing number of edge use cases factoring in open vCPE as well as open SD-WAN which later supported any virtual access device such as OLT and vRAN.

The growing number of edge use cases led us to the realization that we needed a new edition of Cloudify for handling global distribution and scale- hence Cloudify Spire was officially announced at MWC19 as the new Cloudify edge platform.

The Formation of Cloudify Spire

In order to meet the demand of the next generation orchestration, deconstructing and 'rebuilding' Cloudify core components was necessary to establish the best fit in line with demand for scale and multi-site deployments.

The core components of Cloudify can be broken down into the following three elements:



Blueprint

Cloudify uses TOSCA as the basis for its modeling language. One of the key benefits of TOSCA is that it was designed as a domain-specific language and not as a configuration file. As such it supports inheritance, interfaces and sub classing, as well as imports and namespaces, making it relatively flexible. Cloudify uses the same blueprint model to describe low-level elements such as Cloud infrastructure and high-level service composition of multi-domain orchestration which comprises a hierarchy of multi-domain orchestration.

Plugins

Plugins are used as a generic resource of libraries responsible for mapping any API endpoint into a library of TOSCA nodes. There are two kinds of plugins within Cloudify; **Infrastructure plugins** - including Openstack, Azure, AWS, and **configuration plugins** such as Ansible, Script and Terminal.

Manager

The Cloudify manager is used as the daemon responsible for maintaining the current state of the system and execution of the workflow based on that state. The Cloudify manager can have a different role depending on the layer in which it is operating. For example, it can serve as a specific domain orchestration (VNFM) where it is managing the resources of a specific service. At the same time, it can be used to manage multiple orchestrators and domains. The actual role of the manager is defined by the layer of the stack in which it is being placed and the target resources that it is managing.

One of the clear conclusions from this exercise is that the existing structure of Cloudify is often too generic for people to understand exactly how to use it in each layer and use case. Spire has been built with a more structured architecture to leverage those generic building blocks specifically for the layer and role it operates.

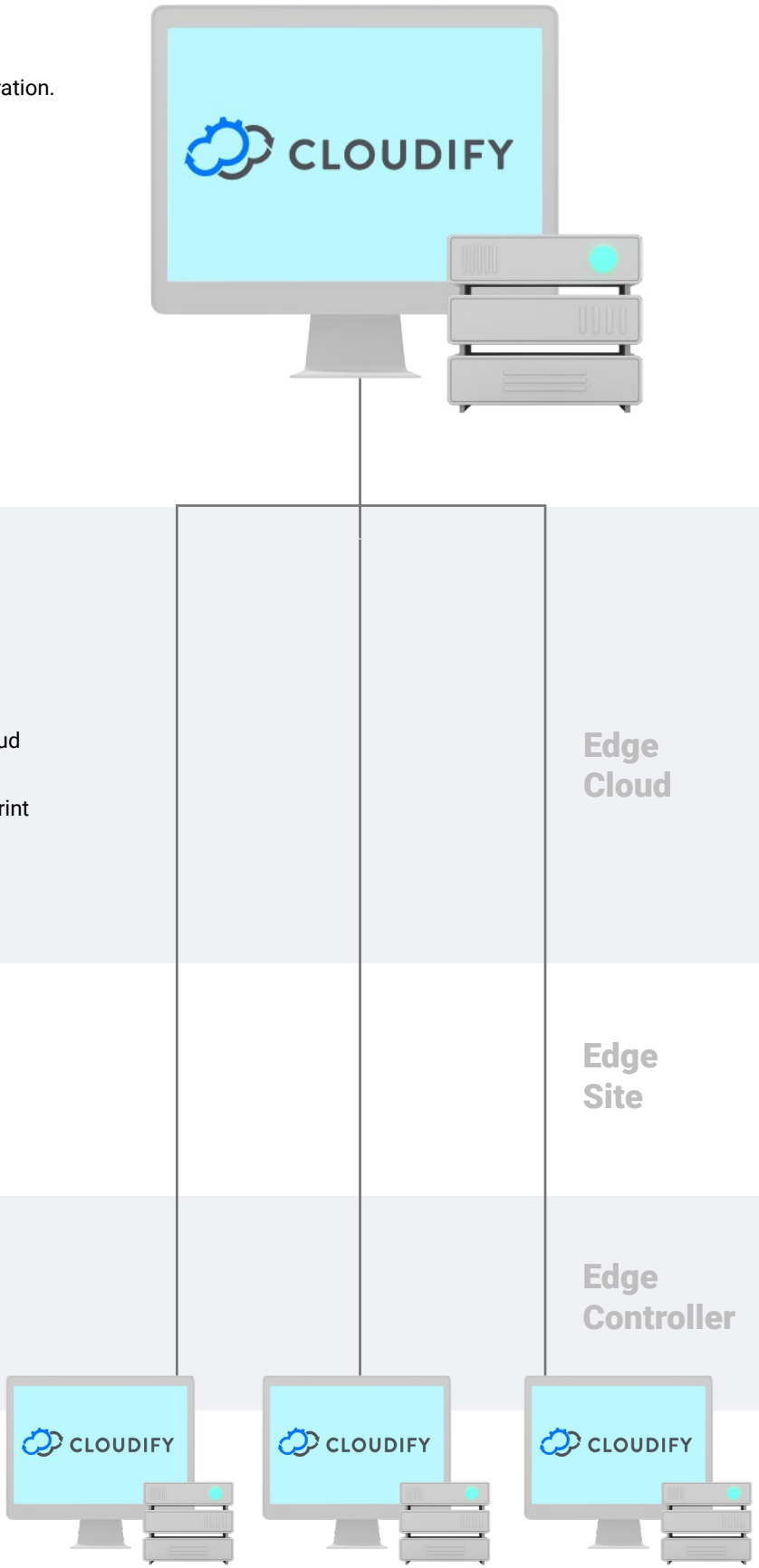
Modifying Cloudify for Multi-Site/ Domain Orchestration

The main target set for Cloudify Spire was to make multi-site/ multi-domain orchestration the primary target architecture.

To meet this goal, it was decided to move from a central orchestration design to a model featuring many micro-orchestrations that are coordinated by one central orchestration. We refer to this model as Manager of Manager (MoM).

Organizing the micro-orchestration into a purpose built-edge cloud:

The move to micro-orchestration was meant to address scaling and multi-site requirements but it was still too generic and could add complexities from an operational and modeling perspective. We therefore decided to arrange the MoM into a purpose-built edge cloud architecture built out of the following three layers:



1. **Edge Cloud:** a group of edge sites targeted for a particular use case where a client can design multiple edge-cloud environments as needed: QA / production environments, Cloud, or edge environment. An edge cloud is a private case of a blueprint in which the components are not the actual services. The edge cloud will inherit all capabilities of a blueprint such as the ability to create multiple instances of that same blueprint (deployments, different versions etc.) The deployment update mechanism can also be used to add/remove sites or change the configuration of a particular site.

2. **Edge Site:** Each Edge Site represents a specific environment, and includes its specific data (secretes, infrastructure setup, Network setup, binaries, security setup etc.)

3. **Edge Controller:** The Edge controller is a private case where the manager is responsible for a particular site or environment.

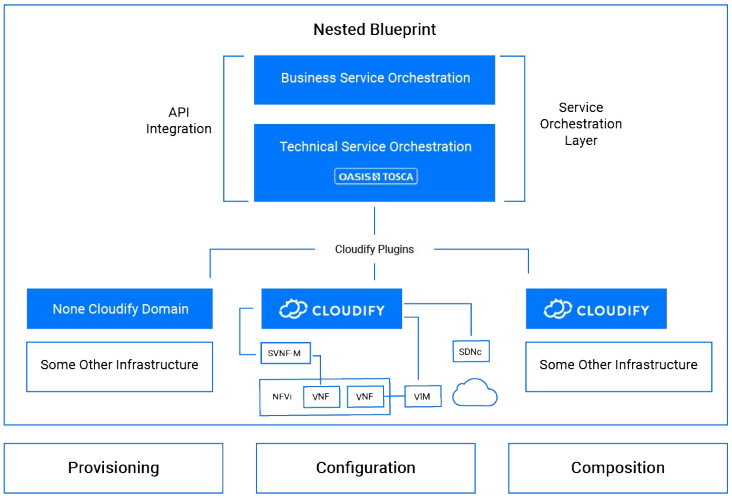
Multi-Site / Domain Service Definition

Automating workloads across multiple micro-orchestrations needs a different way of thinking- particularly on how to model these services.

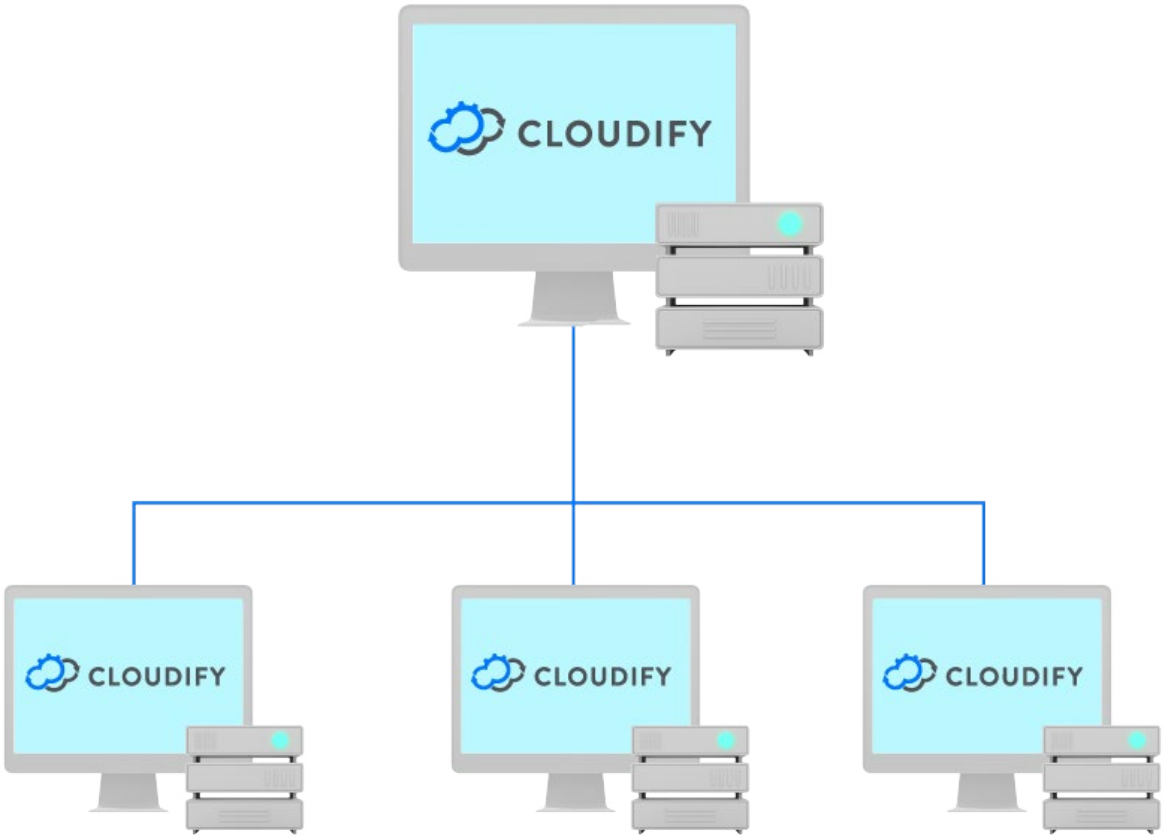
Multi-Layer Modeling

Breaking the modeling into three layers can abstract the complexity of the service modeling as well as allow a higher degree of portability of the service between different edge sites.

- **Infrastructure orchestration layer**: part of each edge-site.
- **Service layer:** will include the definition of the specific service components and configuration.
- **Service composition:** will define the relationship between multiple services and domains and will provide end-to-end definition.

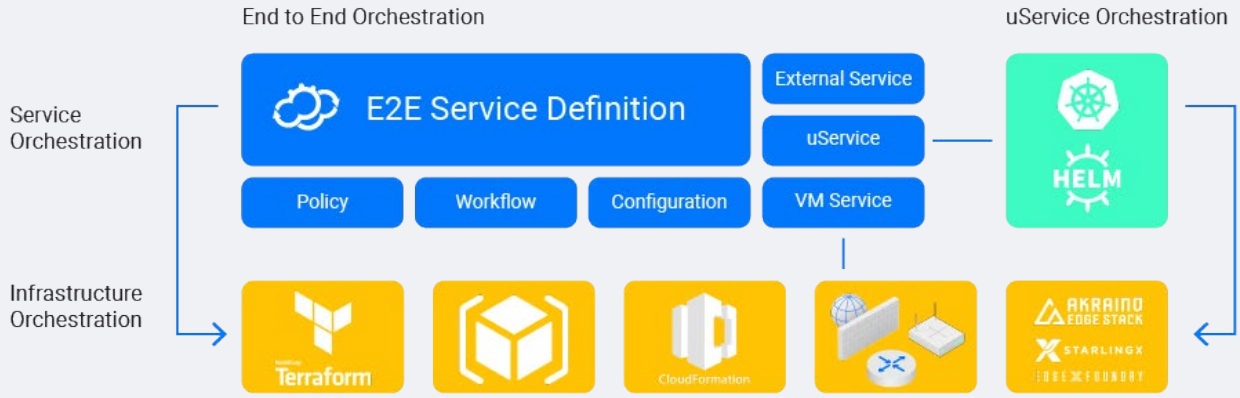


Smart placement through requirements/ capabilities matching



Multi Language Modeling

Creating a layered approach also allows a combination of multiple modeling languages together. For example, Cloud Formation or Azure Arm can now be used to set the infrastructure, Ansible for configuration and TOSCA for service composition. This flexibility can allow a better way to take advantage of the native languages in each cloud infrastructure rather than trying to compromise on the least common denominator.



Opinionated Management Flavors

The current Cloudify manager can be configured in many different flavors (VM, RPM, Container, Clustered etc). The manager can also take many different roles including domain manager and multi-domain manager. This flexibility also leads to more complexities, especially as it relates to the ability to ensure consistent upgrades, or maintenance. These complexities can grow exponentially moving from a central manager architecture to many micro-orchestrations.

To simplify the manager maintenance, it was decided to minimize the number of flavors and standardize on a docker-only flavor. In addition, we chose to provide two flavors for running the manager - Light and Scalable. Light configuration will use a single container model and scalable configuration will be set an Active Active cluster.

Central Manager
(Multi Site/ Domain Manager)

Scalable
(Active Active Cluster)

- Coordinating workfloads between Edge Cloud and Edge Sites
- Automate the management of all edge-sites (single point of access)

Edge Site Manager
(Domain Manager)

Ultra Light (Agent Only)

Light (Single Container)

Scalable (Active Active Cluster)

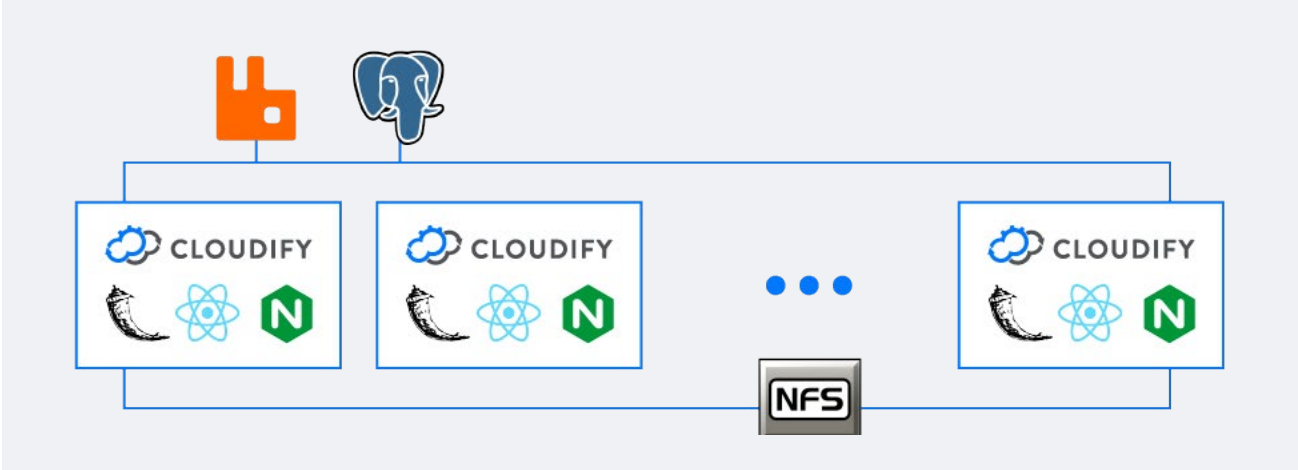
- Autonomous management of a specific Edge Site
- Abstract the environment setup / configuration
- Unit of scale

Opinionated & Purpose built manager packages

Cloud Native Active Active Cluster

To ensure simple upgrades, scaling, and high availability, Cloudify uses a cloud native clustering model where the front-end orchestration functionality is provided as a stateless container that can scale simply by adding more instances (replica-set). Upgrades can be achieved by replacing the front-end container with an updated version.
States are kept in a separate database, messaging and storage clusters.

The adoption of a cloud-native clustering model simplifies the maintenance process across all the micro-orchestration clusters.

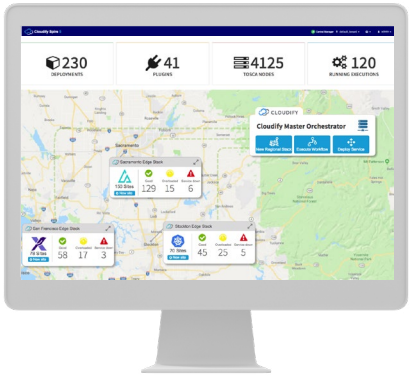


Criteria Based Placement

The decoupling of the service definition (from the infrastructure definition) allows a more policy driven placement. For example, services can be deployed on an edge site that is less occupied in the east coast, a site that fits the QA or production environment, a site that fits a specific department or business , a site that fits a particular cloud such as Azure, AWS or a site that match a specific stack such as Kubernetes or Akraino.

Secured and Reliable Communication

One of the challenges in multi-site management is that the communication channels between the central manager and the edge-site is often less reliable than in a central data-center as they often go through high-latency channels. In addition, each edge site often runs on a private network that is not accessible to the outside world due to security or regulation issues. To allow secured and reliable communication that would address these conditions, all communication from the central manager to the edge-sites is done via a reliable messaging bus. The communication is also done in a pull mode- which means that a master orchestrator can control the edge sites without having access to its internal network. All communication is also encrypted to ensure full privacy of the data channel.



Putting It All Together - Cloudify Spire Demo Orchestration

One of the challenges in designing multi-site orchestration and specifically in moving from a central orchestration model to many distributed micro-orchestrations is the complexity that this change can bring with it - both from an operational and an application modelling standpoint.

To avoid these complexities, we designed Cloudify Spire with a user-first approach, allowing the user experience to lead the underlying architecture and not the other way around.

The result is a far slicker user experience seen by comparing the current Cloudify MoM (4.5 release) to Cloudify Spire (5.0 Release).