VERS LE SOFTWARE DEFINED NETWORKING : INFRASTRUCTURE RÉSEAU UNIVERSELLE ET PROGRAMMABLE

+ LIVRE BLANC +

scasicomp
Your Data Agility

HERVÉ CASTAN
BUSINESS DEVELOPMENT MANAGER
SCASICOMP

SCASICOMP.COM

Sommaire :

1. LA GENESE ..... P. 3
2. SOFTWARE DEFINED NETWORKING ..... P. 6
3. CISCO ACI ..... P. 7
   3.1 TOPOLOGIE PHYSIQUE ..... P. 7
   3.2 L'APPROCHE SDN DE CISCO AVEC ACI ..... P. 8
   3.3 MANAGED INFORMATION TREE (MIT) ET MANAGED OBJECT (MO) ..... P. 9
   3.4 LE POLICY MODEL D'ACI ..... P. 10
4. ACI ANYWHERE ..... P. 11
5. ACI POUR TOUS ..... P. 13
6. PROGRAMMABILITE DES COMPOSANTS D'INFRASTRUCTURE ..... P. 14
7. CONCLUSION ..... P. 15

1. La genèse

L'évolution des solutions pour le Système d'Information des Entreprises s'accélère aujourd'hui de façon exponentielle.

Il y a quelques années, une demande **Métier** nécessitant le développement d'une nouvelle application allait devoir attendre des mois, voire des années selon la complexité de la demande, avant que cette nouvelle application ne bascule en Production.

Le processus était simple et monolithique. Le Métier identifiait un nouveau business potentiel et envoyait une demande pour une nouvelle application à l'équipe Développement. Celle-ci, via une méthode traditionnelle en V, développait puis livrait ce nouveau package applicatif au bout de quelques mois, au mieux. L'équipe Production devait ensuite commander et déployer l'infrastructure nécessaire à cette nouvelle application, ce qui allait nécessiter encore quelques semaines.

Un premier changement a eu lieu avec l'arrivée de la **Virtualisation Serveur** qui permettait à l'équipe Prod de provisionner très rapidement l'environnement de Calcul.

Mais surtout, nous avons assisté à deux évolutions majeures courant des années 2000 :

- En 2006, Amazon lance son service de **Cloud Computing** qui offrait de la ressource de Calcul (grâce entre autres à ce fameux mécanisme de Virtualisation Serveur) et du Stockage en ligne, accessible à la demande et avec une facturation à l'usage.
- L'émergence des **Méthodes de Développement dîtes « Agiles »,** la plus répandue à l'heure actuelle étant la Méthode **Scrum**. La méthode Agile permet de fluidifier les demandes Métier. L'équipe Dév va pouvoir livrer très vite un noyau applicatif immédiatement fonctionnel afin de répondre à la demande Métier initiale. Ce noyau applicatif d'origine sera ensuite enrichi au fur et à mesure des développements successifs. Ainsi, le délai entre la demande Métier et la mise en Production de l'application n'excèdera pas quelques semaines.

Le processus devenait donc le suivant : le Métier envoyait une demande à l'équipe Dév qui pouvait répondre rapidement avec une application de base fonctionnelle et répondant aux attentes des Métiers. Si la production n'était pas en mesure de fournir l'Infrastructure pour supporter cette nouvelle application, la cellule Métier passait commande du service d'Infrastructure chez un Cloud Public.

L'informatique traditionnelle risquait donc de se retrouver marginalisée !

La première étape a consisté à faire évoluer le Système d'Information de l'Entreprise vers le modèle **Cloud Public**, tout en restant internalisé, il s'agit du **Cloud Privé**. Les services offerts entre Cloud Public et Cloud Privé devenant identiques, libre-service et facturation à l'usage, le choix entre ces deux types de Cloud s'effectue désormais selon la criticité des données et des applications, ainsi que la façon dont elles sont utilisées.

Les deux étant pertinents pour une Entreprise donnée, chacun s'oriente donc vers une utilisation conjointe de ces deux modèles de Cloud, d'où le Cloud Hybride.

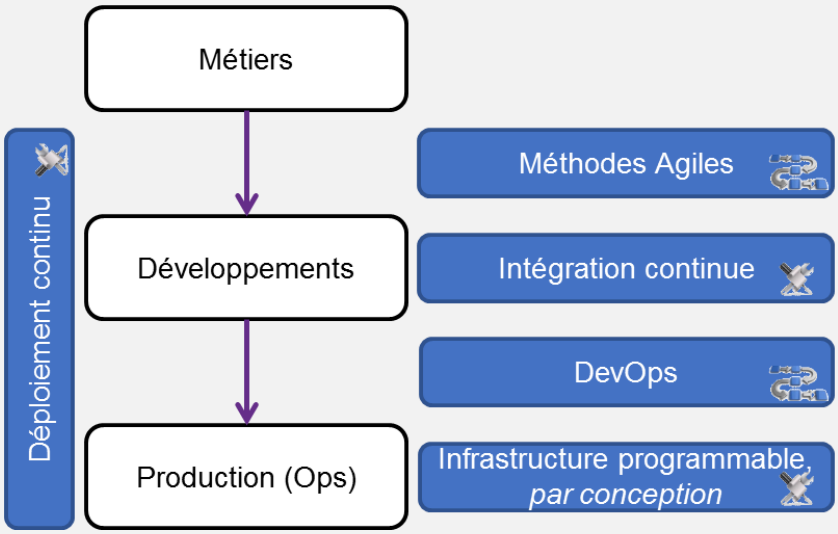
Afin d'être le plus efficace possible dans l'utilisation de ce nouveau modèle d'infrastructure de type Cloud, il faut également réconcilier Dév et Prod. Cela sera le rôle d'une initiative DevOps qui va fluidifier les demandes de l'équipe Dév au même titre que la Méthode Agile le fait pour les demandes émanant des Métiers.

Sans rentrer dans le détail, nous aurons l'occasion d'y revenir dans un autre article, l'initiative DevOps comprend les cycles suivants :

- **Intégration Continue** dont le bénéficiaire principal est l'équipe Dév (mais qui finalement adresse les attentes des métiers).
- **Déploiement Continu** dont le bénéficiaire principal est le commanditaire de cette nouvelle application, soit, à nouveau, le Métier.

Le but recherché pour la Production est de s'intégrer nativement dans les outils utilisés pour le Déploiement Continu. On parle ici « **d'Infra as Code** ». L'Infrastructure est vue comme du code et sera traitée comme tel par les utilitaires DevOps. L'assemblage d'un package applicatif par un Ansible par exemple, va également inclure la programmation de l'Infrastructure par ce même Ansible, afin de provisionner les ressources d'Infrastructure nécessaires au bon fonctionnement de ce nouveau package applicatif.

L'Infrastructure doit donc devenir programmable afin de réaliser le Triptyque du Système d'Information tel que représenté ci-après :



Un troisième élément est à prendre en compte dans l'analyse de l'évolution du Système d'Information, l'environnement dans lequel va s'exécuter une application donnée :

- Les applications tournent majoritairement aujourd'hui dans un environnement de Production virtualisé, selon plusieurs types d'Hyperviseurs.
- Les **méthodes Agiles** avec le développement ultra-rapides de mini-fonctions ont permis l'émergence des Micro-Services, particulièrement adaptés à un nouveau type d'enveloppe applicative, le Container, brièvement décrit ci-après.
- Divers environnements de type **PaaS**.

Les **bénéfices du Container** sont multiples :

- ✓ **Légèreté** : un seul OS par host peut porter des centaines de Containers,
- ✓ **Performances** : le container a un accès direct aux services du kernel de l'OS, aucune émulation,
- ✓ **Micro-services** : un micro-service par Container, ils peuvent être chainés dans une application disposant de centaines de micro-services,
- ✓ **Portable** : le container embarque tous les utilitaires et librairies nécessaires au service applicatif qu'il renferme, il est donc parfaitement autonome, y compris dans un Cloud Public,
- ✓ **Economique** : le royaume de l'Open Source.

Le Container peut nécessiter le déploiement de plusieurs utilitaires, nous en citerons deux parmi les plus répandus :

- ✓ Un gestionnaire du cycle de vie du Container, exemple Docker.
- ✓ Un orchestrateur de Container, afin de corréler l'ensemble des containers représentant une application donnée, exemple Kubernetes (également connu sous le sigle K8S).

Nous avons vu que dès l'origine, le Cloud Public offrait de la puissance de Calcul et du Stockage en ligne. De nos jours, le Cloud Hybride offre les mêmes services, que la ressource de Calcul ou de Stockage soit localisée dans un Cloud Privé ou Public.

Mais qu'en-est-il de la troisième composante d'Infrastructure, à savoir le Réseau ?

Depuis quelques années maintenant, sont apparues des solutions de Virtualisation du Réseau, plus connues sous le terme de SDN pour « Software Defined Networking ».

La solution **SDN de Cisco est ACI,** pour Application Centric Infrastructure, lancée en 2014.

Ce document a pour but de détailler l'état de l'art de la solution ACI de Cisco selon les différents axes majeurs décrits précédemment :

- ✓ L'intégration dans le modèle d'Infrastructure de type Cloud Hybride.
- ✓ L'intégration dans une initiative DevOps par la programmabilité des composants d'Infrastructure selon le modèle « Infra-as-Code ».
- ✓ Le support de tout type d'environnement, multi-hyperviseur, Container, etc.

Avant tout, un rappel sur la structure d'une solution SDN à suivre.

2. Software Defined Networking

Historiquement, les Administrateurs Réseau géraient leur Infrastructure en **CLI** (Command Line Interface), se connectaient sur chaque commutateur ou routeur afin de configurer une nouvelle route IP, ou une nouvelle ACL pour restreindre tel flux de telle source vers telle destination, etc.

Cette approche n'est plus viable de nos jours avec l'avènement des Cloud Publics côté Infrastructure et de la réactivité demandée par les équipes Dév côté Métier !!

Dans le modèle SDN, deux composantes :

- Le **Data Plane** en charge de la commutation / routage des données, soit les muscles.
- Le **Control Plane** responsable de la mise en place et du bon respect des différentes règles en vigueur au sein du Réseau, soit le cerveau.

Dans une approche SDN, le Data Plane reste à la charge des éléments actifs. Le Control Plane est lui centralisé. Il n'existe qu'une seule entité en charge du provisionnement pour la totalité du Réseau, charge à elle de distribuer son intelligence à l'ensemble des éléments constituant le Data Plane.

Pour cela, les différents acteurs du SDN utilisent un « **Overlay** » qui va masquer l'infrastructure sous-jacente. L'objectif est de décorréler, comme décrit précédemment, l'application de l'environnement physique sur lequel elle va s'exécuter.

L'Overlay sera à base de divers protocoles comme Geneve ou VXLAN, mais dans tous les cas, le Control Plane va déployer ce protocole pour bâtir l'environnement Réseau de façon automatique et cachée sans que l'Administrateur Système ou Réseau n'ait besoin d'aucune compétence sur ces protocoles !!

Le Control Plane va ensuite présenter une **Fabric Réseau** utilisable nativement par les Admins Système et Réseau.

En résumé, une solution SDN comprend deux parties au-delà de la distinction présentée ci-dessus :

- L'Infrastructure Réseau bas niveau.
- L'Overlay.

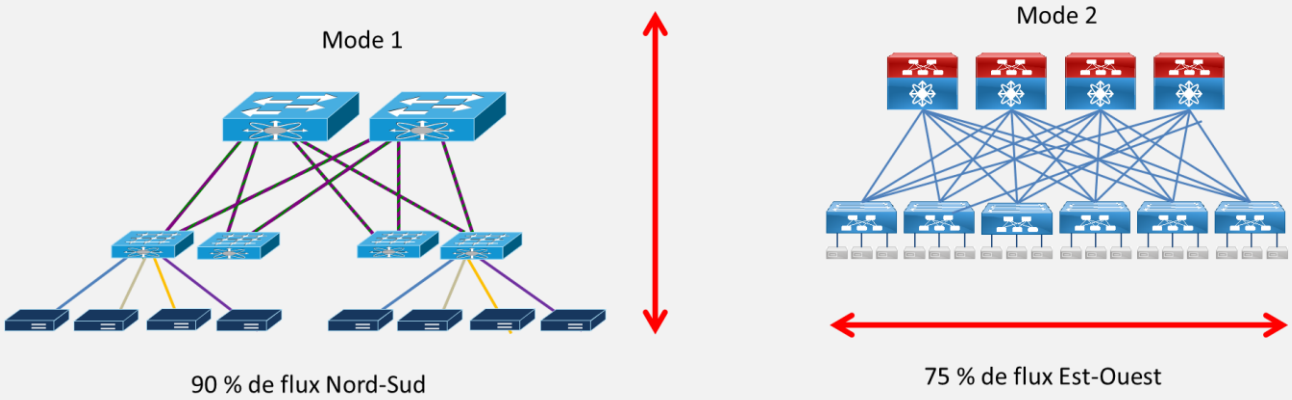
Ces deux éléments peuvent être couplés (d'un même constructeur), ou indépendants.

3. Cisco ACI

Tout d'abord, il convient de décrire ACI selon ses deux principes de base, le MIT avec ses MO et le Policy Model, mais également d'un point de vue topologie physique.

3.1 Topologie physique

La topologie physique des Réseaux de Data Center a fortement évolué ces dernières années du fait du changement de nature des charges applicatives. Cela est matérialisé dans le schéma ci-dessous :



Le **Mode 1** s'applique aux environnements historiques et adresse les besoins des applications traditionnelles. Les flux sont essentiellement de type Client-Serveur et portés par une topologie Réseau en arborescence.

Le **Mode 2** va adresser les besoins des applications de dernière génération, applications distribuées type Big Data, Analytics, Hyperconvergence, Stockage Objet, Cluster de Container, etc. Pour ces nouveaux profils applicatifs, il y a une majorité d'échange entre les différents nœuds applicatifs, ainsi qu'une grande mobilité des charges applicatives au sein de leur propre Cluster.

D'où une topologie de type Fabric qui va apporter les bénéfices suivants :

- ✓ Le Réseau étant entièrement maillé, le nombre de sauts d'un point de vue Réseau est fixe, quels que soient les mouvements de tel ou tel composant applicatif au sein d'un Cluster.
- ✓ La latence est également fixe et garantie pour les mêmes raisons.

Ainsi, la mobilité applicative est assurée et le comportement, notamment au niveau du temps de réponse, ne sera jamais impacté par le déplacement d'un composant applicatif au sein de son Cluster.

ACI est basé sur cette topologie de type Fabric avec les « Spines » (dorsales) en cœur et les « Leafs » (accès) pour connecter l'ensemble des composants de type Serveur physiques ou virtualisés, en VM ou en Container, commutateurs d'accès additionnels, FireWall, Load Balancers, Routeurs d'accès distants, etc.

Pour conclure sur cet aspect topologie, l'architecture de base d'ACI comprendra les éléments suivants :

- ✓ Minimum 2 Spines, l'équivalent du Cœur de Réseau,
- ✓ Minimum 2 Leafs, soit les commutateurs d'accès sur lesquels seront connectées toutes les ressources de type :
  - Serveurs virtualisés, en mode VM ou Container,
  - Serveurs bare-metal,
  - Les commutateurs d'accès sur d'autres ressources,
  - Les routeurs d'accès WAN,
  - Les composants spécialisés type Load Balancer, FireWall, etc. en mode Service Graph.
- ✓ Le Cluster APIC, soit le contrôleur de la Fabric ACI, minimum 3 nœuds.

3.2 L'approche SDN de Cisco avec ACI

Certaines solutions du Marché sont complètement agnostiques au Réseau sous-jacent. Cela étant dit, une Infrastructure Réseau pour poser leur Overlay leur est nécessaire. Par ailleurs, le fait que l'Overlay soit complètement indépendant du Réseau physique présente quelques manques de visibilité et de compréhension entre les deux.

Dans la solution Cisco ACI, l'Infrastructure Réseau et l'Overlay sont intimement liés.

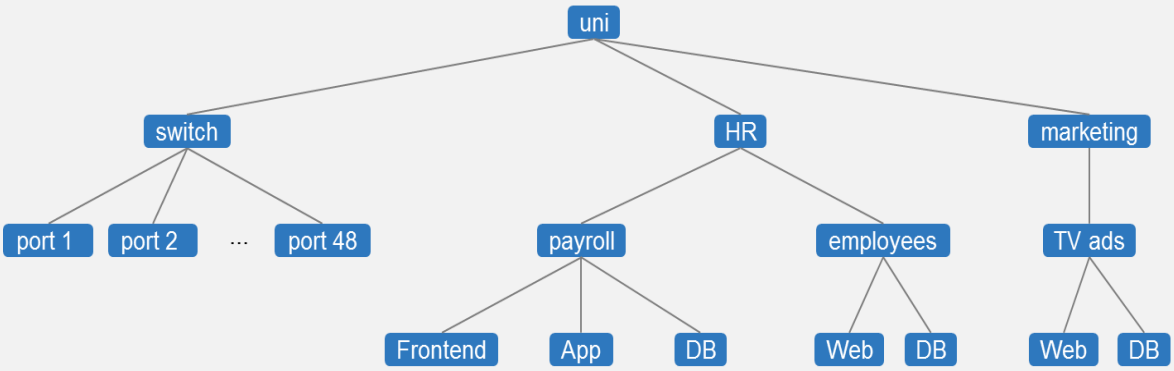
Nous voyons là un des premiers avantages de la solution SDN Cisco ACI : l'Overlay étant couplé au Réseau physique, la visibilité de l'Overlay sur le Réseau de transport est totale :

- Tout déviance de comportement, toute défaillance physique du hardware entre autres est immédiatement perçue par l'Overlay qui va remonter l'information en temps réel.
- D'un point de vue Control Plane, ACI est basé sur un contrôleur centralisé, l'APIC pour Application Policy Infrastructure Controler. L'APIC dispose de son propre GUI et également d'une API au format REST qui offre la possibilité à tout framework du Marché, à tout utilitaire, de s'interfacer avec le Réseau ACI et de le programmer.

3.3 Managed Information Tree (MIT) et Managed Object (MO)

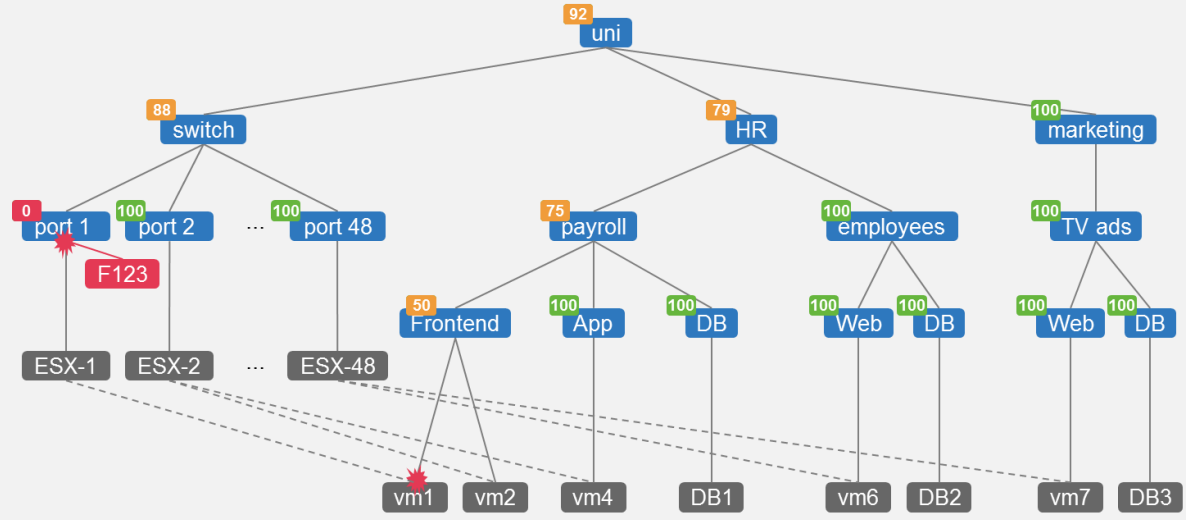
ACI est basé sur un modèle Objet et chaque composant est un **Managed Object** (MO) faisant partie du **Managed Information Tree** (MIT) d'ACI. Tout utilitaire pouvant effectuer une requête REST sur un MO pourra récupérer tout type d'information et également configurer la Fabric ACI via le Contrôleur APIC.

Une représentation ci-dessous du MIT d'ACI :



Nous voyons clairement la distinction à gauche sur les éléments actifs, à droite sur les éléments applicatifs, tous étant des MO dans la vision ACI.

La force d'ACI par rapport à d'autres solutions du Marché étant son couplage fort entre Hardware et Overlay, voilà la matérialisation d'une défaillance d'un composant, en l'occurrence un port de Switch Leaf, dans ACI :



L'impact est instantanément visible au niveau Hardware mais également au niveau Applicatif.

L'Administrateur sait instantanément quel est le point de défaillance matérielle et également l'impact sur son environnement applicatif, les métriques d'impact étant ajustables en fonction de la criticité des applications.

3.4 Le Policy Model d'ACI

La représentation en Objet des ressources ACI sert de base au Policy Model.

Le Policy Model (PM) utilise les MO afin de construire un environnement applicatif. Le PM va relier les différents MO d'une même obédience applicative, via des Contrats, afin de définir l'enveloppe Réseau nécessaire à une application donnée.

Prenons l'exemple d'une application 3-Tiers, frontaux Web – Serveurs Applicatifs – Base de Données.

Il convient ici d'introduire la notion d'EPG, pour **End-Point Group**. Un EPG regroupe les différentes ressources servant un même but.

Dans le cas de notre application 3-Tiers, nous allons monter plusieurs EPG :

- Un EPG WEB regroupant les différents frontaux Web,
- Un EPG APP pour les serveurs Applicatifs,
- Un EPG BDD pour la Base de Données en back-end de l'application.

Entre ces différents EPG, nous allons tirer des règles (Contrats) pour valider ou interdire telle ou telle communication. Exemple, seul le port 80 sera ouvert vers l'EPG WEB, et tel autre port pour autoriser le dialogue entre l'EPG APP et l'EPG BDD en fonction des prérequis propres à telle ou telle Base de Données.

La combinaison des EPG avec les Contrats représente un ANP (pour Application Network Profile), soit le Policy Model d'ACI.

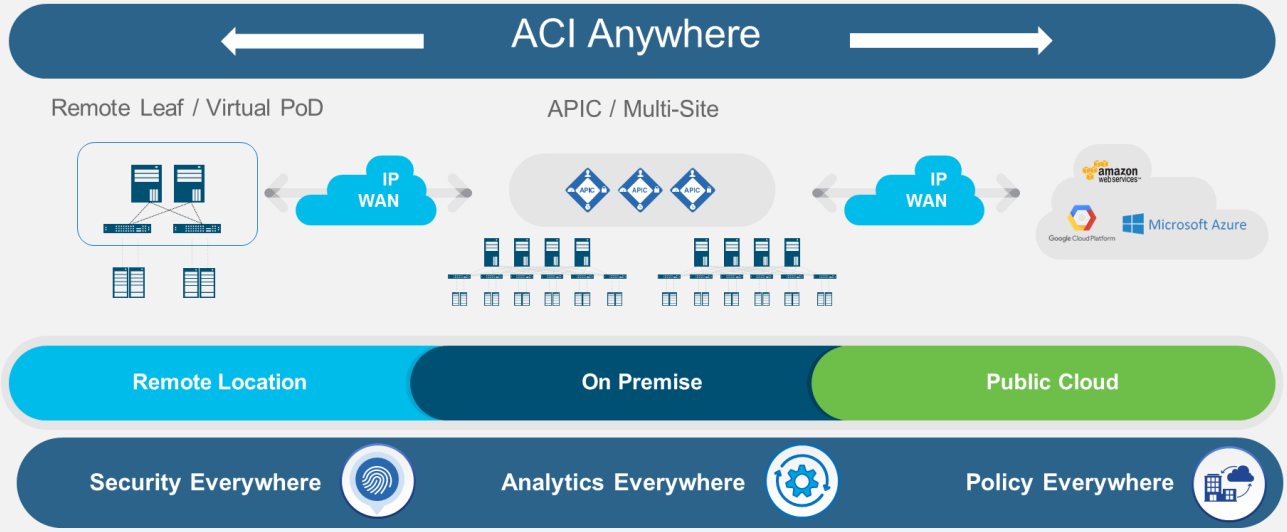
L'intégration Overlay + Physique apporte clairement cette visibilité à l'Administrateur. Une défaillance hardware est signalée non seulement côté physique (tel port sur tel switch), mais l'impact sur le fonctionnel applicatif est aussi immédiatement remonté.

Les éléments ci-dessus présentent l'intérêt de la solution ACI d'un point de vue Pilotage et Supervision du DC. Les chapitres suivants vont étudier les bénéfices d'ACI selon les trois points attendus par le Data Center de nouvelle génération :

- ✓ L'universalité du Réseau (Cloud Hybride),
- ✓ Sa programmabilité (DevOps),
- ✓ Le support des différentes enveloppes applicatives.

4. ACI Anywhere

D'un point de vue Cloud Hybride, le positionnement de la solution Cisco ACI est représenté ci-dessous :



Les possibilités d'ACI Anywhere sont listées ci-après :

- ✓ ACI on prem est disponible depuis l'origine (Spines x 2 + Leafs x 2 ou plus), un seul Cluster APIC.
- ✓ ACI multi-DC est disponible, un cluster APIC mono-site ou réparti (solution Multi-PoD).
- ✓ ACI multi-sites est disponible, plusieurs Cluster APIC avec MSO pour fédérer le tout,
- ✓ ACI sur site distant, une ou plusieurs Remote Leaf, un seul Cluster APIC,
- ✓ ACI sur site non-Cisco avec Virtual PoD, mode Multi-PoD, un seul Cluster APIC,
- ✓ ACI multi-Cloud en mode MSO, plusieurs Clusters APIC dont le Cloud APIC pour AWS et Azure à ce jour.

Clarifions de façon simple tous ces termes propres à ACI :

Un PoD représente un Data Center. Un Cluster APIC va gérer ce PoD.

Plusieurs Data Centers peuvent être gérés par un même Cluster APIC, c'est le Multi-PoD.

MSO est le Multi-Site Orchestrator, soit un contrôleur d'APIC au même titre qu'un UCS Central va connecter plusieurs Domaines UCS entre eux. Avec MSO, plusieurs Cluster APIC sont interconnectés avec une hiérarchie apportée par le MSO qui apporte ainsi une isolation de configuration entre les différents Sites.

La Remote Leaf est une Leaf distante (un commutateur de la gamme nexus9000) connectée par un Réseau IP sur les Spines du Site Central, gérée par l'APIC du Site Central. On peut ainsi provisionner des EPG entre site central et site distant et déplacer des charges applicatives comme si la Leaf était locale aux Spines. Idéal pour des sites ROBO.

Le vPoD va encore plus loin, il émule la Fabric ACI en mode virtualisé, soit des vSpine et des vLeaf sous forme de VM sur un site ROBO disposant de tout type de commutateurs, non-Cisco ou Cisco non-ACI ready. Nous sommes dans le cas d'un PoD virtualisé, donc du multi-PoD géré par le Cluster APIC en central.

Le mode Cloud ACI est la dernière évolution apportée avec ACI 4.1, c'est le Cloud APIC disponible sur AWS et Azure à ce jour. Dans ce mode, l'APIC est émulé sur les Cloud Public et connecté sur les APIC du site central via MSO.

L'interconnexion vers les Clouds Publics est assurée par Réseau IP, via des tunnels VPN portés par des CSR, le CSR1000v étant disponible dans la Market Place AWS ou en mode BYOD.

Le Cloud APIC va réceptionner les demandes ACI et les transformer en Constructs AWS. Exemple :

- ✓ Un EPG ACI sera transformé en Security Group AWS,
- ✓ Un contrat ACI sera transformé en SG Rule,
- ✓ Une VRF ACI sera un VPC AWS,
- ✓ Un PoD ACI sera une Région AWS,
- ✓ Etc.

ACI adresse donc le Data Center on-prem, le site distant avec Remote Leaf Nexus9000, le site distant sans switch ACI avec le vPoD, ainsi que les Clouds Publics.

5. ACI pour tous

L'Universalité Réseau d'ACI a été démontrée dans le Chapitre précédent.

Il importe aussi de savoir si ACI est multi-environnement.

Aujourd'hui, la solution ACI s'insère dans les environnements suivants :

- ✓ Virtualisation Serveur : par connexion directe sur la vCenter VMWare ou la SCVMM Microsoft.
- ✓ Virtualisation de Container : par plug-in dans K8S,
- ✓ Serveur Bare-Metal : au niveau de la Leaf,
- ✓ Serveur non-x86 : même chose que ci-dessus.

Tout type de workloads applicatifs, virtualisé, en Container ou physique sont donc portés par un réseau ACI.

Dans un environnement VMware, il suffit de renseigner l'adresse de la vCenter pour qu'ACI découvre l'ensemble de l'environnement virtualisé. A noter qu'il existe également un plug-in ACI pour VMware qui permet de piloter ACI depuis la vCenter.

En Kubernetes, il suffit de fournir le plug-in ACI dans K8S pour qu'ACI découvre l'ensemble des Containers gérés par K8S.

L'apport d'ACI en environnement Container est essentiel lorsque l'on envisage de déployer des applications containerisées en Production pour la raison simple suivante.

K8S dispose de règles de Sécurité. Mais ces règles sont une volonté, dans le sens ou K8S dit : je ne veux pas que tel Container ne discute pas avec tel autre Container, sauf sur tel protocole IP. Mais K8S ne dispose d'aucun mécanisme pour autoriser ou interdire telle ou telle discussion. Il s'agit juste d'un désir.

D'où l'intérêt d'ACI sur ces environnements containerisés car ACI va gérer, tout comme avec les VM, des EPG regroupant des Containers de caractéristiques équivalentes et des contrats entre EPG afin de restreindre la discussion entre Containers aux seuls flux autorisés.

6. Programmabilité des Composants d'Infrastructure

Il est intéressant de savoir qu'une solution Réseau est Universelle en termes de localisation (on-premise, Cloud) et d'enveloppe fonctionnelle (VM, bare-metal, Container).

Il est tout autant intéressant de savoir si elle va s'insérer dans une initiative DevOps, autrement dit dans de l'Infra-as-Code.

L'Infrastructure n'est pas forcément concernée par la partie purement Dev du DevOps, à savoir un Maven pour les builds, un Jenkins pour l'Intégration Continue, un Git pour le repository de code source et le versionning ou un Nexus pour le repository d'exécutables et d'Articfacts.

L'Infrastructure sera forcément concernée lors de la partie Déploiement Continu du DevOps.

Ansible assurera l'assemblage des Packages Applicatifs issus du Développement Continu ci-dessus et de leur mise à disposition auprès de l'équipe Prod pour une bascule en Production.

Cependant, cette nouvelle application va nécessiter des ressources matérielles afin de s'exécuter. Il serait intéressant qu'Ansible puisse également provisionner le hardware.

ACI peut être piloté par Ansible afin de provisionner les besoins d'Infrastructure Réseau ainsi que les règles de Routage et de Sécurité afférentes.

Il existe de nombreux outils d'assemblage de packages applicatifs, c'est le royaume de l'Open Source. Si les Chef, Puppet, CFengine avaient pignon sur rue il y a quelques années, les récentes mises en œuvre d'outils CD se font à plus de 90% sur base Ansible, son gros différenciateur étant qu'il est agent-less.

Si le choix d'Entreprise s'est porté sur Terraform, car c'est un outil foncièrement Infra-as-Code, un Terraform est également apte à piloter une Infrastructure Réseau ACI.

Au-delà des outils cités ci-dessus, ACI dispose via son contrôleur APIC d'une API Nord au format REST qui permet à tout utilitaire de piloter le Réseau ACI, aussi bien en termes de monitoring et supervision, qu'en termes de provisionnement.

L'API au format REST d'ACI peut être formattée de multiples façons. Chacune des déclinaisons de cette API peut être validée via POSTMAN, utilitaire très répandu dans les API Factory.

7. Conclusion

L'objectif de ce document est de présenter les apports des solutions SDN en général et de l'offre Cisco en particulier en regard des besoins d'un Système d'Information de nouvelle génération.

Les attentes sont décrites ci-après et la solution ACI de Cisco y répond de la façon suivante :

- ✓ **Cloud Hybridation :** ACI peut être déployé on-prem, sur site ROBO avec commutateur ACI (Remote Leaf) ou non, en vPoD et sur Cloud Public (AWS et Azure à ce jour).
- ✓ **Multi-environnements :** ACI s'insère dans les environnements virtualisés VMware ou Microsoft, dans les Clusters de Containers Docker / Kubernetes et vers tout serveurs physiques x86 ou non.
- ✓ **Programmabilité dans une initiative DevOps :** le contrôleur d'ACI, l'APIC, peut être programmé par les deux outils majeurs utilisés dans la partie Déploiement Continue d'une initiative DevOps aujourd'hui, Ansible et Terraform. L'API au format REST de l'APIC lui permet de s'insérer dans tout autre framework du Marché.

La solution SDN de Cisco est parfaitement mature et complète pour adresser toutes les attentes des Centres Informatiques de dernière génération.