Multi-Cloud

* use of two or more cloud computing services (AWS/Azure)
* Mix of public cloud

                Advantages

* + avoid vendor lock-in
  + prevent data loss or downtime
  + use of more price-competitive cloud services
  + Take advantage of the speed, capacity or features offered by a particular cloud provider in a particular geography.
  + data sovereignty - Certain laws, regulations and corporate policies require enterprise data to physically reside in certain locations
  + enables organizations to locate compute resources as close as possible to end users to achieve optimal performance and minimal latency.
  + offers the ability to select different cloud services or features from different providers. This is helpful, since some cloud environments are better suited than others for a particular task.

                Disadvantages

* + public cloud providers offer volume discounts, where prices are reduced as customers buy more of a particular service.
  + requires an IT staff to have multiple kinds of cloud platform or provider expertise
  + Workload or application management in multi-cloud computing can also be a challenge, as information moves from one cloud platform to another.

Hybrid Cloud

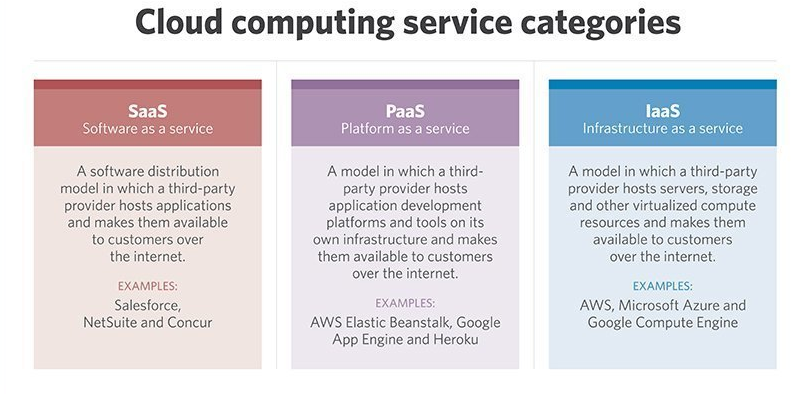
 - Private(on-premises) and public cloud (third party, aws)

- An enterprise often adopts hybrid cloud to achieve a specific task, such as the ability to run workloads in house, and then burst into the public cloud when compute demands spike.

SPI Model

refers to the most common service models of [cloud computing](https://searchcloudcomputing.techtarget.com/definition/cloud-computing):

* software as a service ([SaaS](https://whatis.techtarget.com/definition/SaaS)),
* platform as a service ([PaaS](https://searchcloudcomputing.techtarget.com/definition/Platform-as-a-Service-PaaS)) and
* infrastructure as a service ([IaaS](https://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS))



Cloud Management Platform

Infrastructure as Code (IaC)

                - Also called as programmable or software-defined infrastructure

                Benefits

* + Software developers can use code to provision and deploy servers and applications, rather than rely on system administrators in a [DevOps](https://searchitoperations.techtarget.com/definition/DevOps) environment.
  + With the infrastructure setup written as code, it can go through the same version control, automated testing, and other steps of a [continuous integration](https://searchsoftwarequality.techtarget.com/definition/continuous-integration) and [continuous delivery](https://searchitoperations.techtarget.com/definition/continuous-delivery-CD)(CI/CD) pipeline that developers use for application code.
  + IaC with containers..

Disadvantages

* Need tools (Configuration management), requires learning curve and room for errors
* Any errors can proliferate quickly through servers, so it is essential to monitor [version control](https://whatis.techtarget.com/definition/version-control) and perform comprehensive prerelease testing.

**Approach and Methods**

Declarative programming (Example – AWS Cloud formation)

* + Outlines the desired, intended state of the infrastructure
  + does not explicitly list the steps to reach that state

Imperative approach (Example Java, C++)

* defines commands that enable the infrastructure to reach the desired state
* CHEF – can be used in declarative and imperative as needed

**IaC Tools**

Reference - <https://www.thorntech.com/2018/04/15-infrastructure-as-code-tools/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tool** | **Makers** | **Language** | **OpenSource** | **Cloud-agnostic** |  |
| **Terraform** | Hashicorp | Uses DSL (Domain specific language)  Hashicorp Configuration Language (HCL)  Json compatible | YES | Yes | * infrastructure provisioning tool * describe your infrastructure as code * creates “execution plans” that outline exactly what will happen when you run your code, * builds a graph of your resources * automates changes with minimal human interaction. * cloud-agnostic and allows you to automate infrastructure stacks from multiple cloud service providers simultaneously and integrate other third-party services. * can write [Terraform plugins](https://www.terraform.io/docs/plugins/index.html) to add new advanced functionality to the platform * JSON - used to create these configuration files that describe the infrastructure resources to be deployed |
| **Vagrant** | Hashicorp |  |  |  | * Vagrant focuses on quickly and easily creating development environments that use a small amount of virtual machines, * Vagrant runs on top of virtual machine solutions from VirtualBox, VMware, AWS, and any other cloud provider * works well with tools like Chef and Puppet. |
| **AWS Cloud Formation** | AWS | JSON  YAML |  | NO | * configuration orchestration tool that allows you to code your infrastructure to automate your deployments. * deeply integrated into and can only be used with AWS * allows you to preview proposed changes to your AWS infrastructure stack and see how they might impact your resources, and manages dependencies between these resources. * To ensure that deployment and updating of infrastructure is done in a controlled manner, CloudFormation uses Rollback Triggers to revert infrastructure stacks to a previous deployed state if errors are detected. * You can even deploy infrastructure stacks across multiple AWS accounts and regions with a single CloudFormation template |
| **CHEF** | OpsCode | DSL (Ruby based) |  | Yes | * most popular configuration management tools * recipes and cookbooks specify the exact steps needed to achieve the desired configuration of your applications and utilities on existing servers. * This is called a **Imperative/“procedural”** approach to configuration management, as you describe the procedure necessary to get your desired state. * cloud-agnostic and works with many cloud service providers such as AWS, Microsoft Azure, Google Cloud Platform, OpenStack, and more. * Targets developers |
| **Puppet** | Puppet | DSL (Ruby based) |  | Yes | * configuration management tool that helps engineers continuously deliver software. * you can define the desired end state of your infrastructure and exactly what you want it to do. Then Puppet automatically enforces the desired state and fixes any incorrect changes. * **This “declarative” approach** - declare what you want your configuration to look like. Puppet figures out how to get there * mainly directed toward system administrators * Puppet integrates with the leading cloud providers like AWS, Azure, Google Cloud, and VMware, allowing you to automate across multiple clouds. |
| **Ansible** | RedHat | YAML |  |  | * Infrastructure automation tool * models your infrastructure by describing how your components and system relate to one another, as opposed to managing systems independently. * **Does not use agents** * Ansible paybooks * Can write own plugins and modules |
| **SaltStack** |  | Python | Yes |  | * Takes **“infrastructure as data” approach**, instead of “infrastructure as code.” * Saltstack’s **declarative configuration patterns**, while written in Python, are language-agnostic * more easily read and understood. * supports remote execution of commands, whereas Chef and Puppet’s configuration code needs to be pulled from their servers |
| [**Azure Resource Manager**](https://azure.microsoft.com/en-us/features/resource-manager/) | Microsoft |  |  |  |  |
| [**Google Cloud Deployment Manager**](https://cloud.google.com/deployment-manager/) | Google |  |  |  |  |