5.3 Cloud

Parts of the Cloud

There are many ways to build a Cloud and there are also various levels of clouds. This landscape is constantly evolving so note history of major evaluations,

- Dec to Jan 2016 - value, but not for large enterprise org adoption,
    - Technology is still too immature.
    - Leaders really in infrastructure as code.
    - But missing discovery, network orchestration.
    - Relying on native Cloud provided services that are also not mature.
    - Savings to be had, but large Enterprise should not fully adopt yet.
    - All Cloud providers different enough that there is market for a single cloud orchestrator.
    - Cloud providers or key drivers like IBM should focus on CICD pipeline as a product offering for large Enterprise.
    - Docker is compelling, but more immature than people realize for applications outside of specific Netflix type industry.
    - Kuberneties is compelling.
    - Still big gaps on how to solve for basics and Security as usual is an afterthought.
- Sep to Dec 2017 - still not ready, but Enterprise should think about experimenting in and adopt DevOps + OpsDev and CICD,
    - Cloud Density is something that can save orgs tons of money in adoption, but industry not ready for that concept yet.
    - Terraform is compelling and Cloud agnostics as as Infrastructure as Code platform and Pipeline.
    - More reading I have concerns about Puppet becoming a hinderance over time.
    - Docker by itself is still not ready for Enterprise outside of experimenting, but can be an ok container selection for now due to popularity for stateless applications
    - Security starting to become more important but I don't see any key players yet and native cloud still lacking (but moving in the right direction)
- Jan 2018 - technology is now very close to viable,
    - Configuration management finally distinct via Habitat
    - Finally found things around Network as infrastructure
- ...

Before we get too deep, we should look at the key Cloud Advantages to look at the why to then implement using Path to Cloud.

Driving Cloud Concepts

Infrastructure as Code - ...

Elasticity - to grow and shrink as needed.

Table Legendb

Colour	Note
De Facto Leader Emerged	
Emerging	

This table aims to cover the key aspects and list various options from top down.

Component	Why You Need It	What Does it Do	Driving Cloud Concept	BonsaiFramework Pick	Popular Options
Security Scanning	Scan for viruses, missed hardening and accidental data leakage. Varies depending on your needs, but there are some free scanning services now.				UpGuard (reviewing)
Security Intrusion Detection	Look for any intrusions in the system.				
Synthetic Monitoring					• Dynatrace
Health Check					
System Monitoring					
Application Insight Monitoring	Look inside the code to determine performance and support Production problems inside the code.		n/a	Microsoft Azure - Application Insight (free and powerful) Dynatrace was previous winner for stand alone.	• Cloud Provider Module • Dynatrace • CA APM (previously Wily Introscope)
Integrity Verifciation	Confirm and audit any changes to the system.				
DOS and DDOS Mitigation	There is some argument that going true cloud no longer requires this. I'm not convinced.		n/a	Akamai. However, for smaller implementations Cloud Provider built-in services may be enough.	• Akamai • VeriSign

Customer Caching	Take load off of your system.		Elasticity	Akamai. However, for smaller implementations Cloud Provider built-in services may be enough.	• Akamai • Cloud Providers
Orchestration of Containers & Service Discovery	Unified view and control of containers who should hook themselves up and configure to the larger group.		Elasticity		• etcd used by Kubernetes (started by Google) • Swarm (Native Docker) • ZooKeeper used by Mesos phere + Marathon (preferred by DC/OS) • Chef (to a certain extent through infrastructure automation) • Eureka (by Netflix) used by Spring Cloud but services must be stateless and network non-sticky • HashiCorp Console Comparison (to be made)
Application Packaging	Means to create application packages and manage centrally.		Zero Footprint AppDev Model	Automation and configuration that travels with the application. There is some overlap here with configuration management, but I believe in keeping them separate.	• Zero Footprint with Scripts • Habitat
Software Defined Network			Infrastructure as Code.	Cloud Provider Module or Container Technology	• Microsoft Azure • Amazon AWS • Google Cloud Platform • Rackspace • HashiCorp Console
Virtualization Cloud Provider	No point in running the hardware and base OS yourself. Instead use a provider that will take care of auto-scaling hardware, providing IP addresses, storage and a network infrastructure. Bonus points for instituted caching and monitoring. ++ Bonus points for an proven CICD system. Some of the Bonus items you can implement yourself and are documented higher in this stack.		n/a	At the moment (2016) Microsoft Azure for ease of use.	• Microsoft Azure • Amazon AWS • Google Cloud Platform • Rackspace
Environment Configurator	If you have lots of integration points, centralizing one place to configure those small differences suddenly becomes cost effective. This is not actually service discovery (though having it helps immensely)		Remove infrastructure dependency.		• Habitat (tackles applications provided you use it's packaging) • Ansible
Continuous Code Testing					
Continous Infrastructure Testing					• Chef Kitchen
Code Unit Testing					jUnit
Continuous Integration & Deployment	When build completes auto deploy and hook up. Be the workflow engine to manage CI/CD pipeline from source to delivery				• Jenkins • Bamboo
Continuous Build	Building Application Virtualization from Recipes. Think entire ecosystem (not just code) is built from recipes.				• Jenkins • Bamboo
Source Control for Code				Bitbucket or direclty GitHub	• GitHub • Bitbucket (Atlassian Product fronting GitHub) • Mercurial • Subversion (Does not scale for Agile well)
				Packer	

Centralized Log Aggregation and Alerting	Simplification of adapters to be pipeline will likely emerge as part of Cloud Providers and container technology.		Remove infrastructure dependency.	Splunk	
Application Caching System	Lots of noSQL databases in this space.	• saves application data apart from the application instance so data is still available for application <X + 1> when application <X> goes down			
Messaging System	Guarantee delivery and integrity of key transactions across systems.			Depends on your specific messaging needs. Will break this up later.	• Kafka • RabbitMQ
Application Virtualization	Microservices concept of running ephemeral containers at the focusing on escalating a single immutable application.	• Building from Recipes • Linking of Containers		Docker	• Docker
Configuration Management and Building Applications and Integration from Recipe	Often initiated by the CI/CD pipeline control to build the operating system, setup users, install software and apply configuration.		Configuration Management and Infrastructure as Code. This includes SDN (Software Defined Networking) which is still a growing space as the what's available is still rudimentary.	Chef and Puppet are leading (2017) configuration management tools. However, they don't solve (without fiddling) stateful applications requiring workflow deployment, ie upgrade of a database.	• Chef • Puppet • Docker (do for more than just the os and base?) • Ansible • SaltStack • Terraform Comparison
Automation of Cloud Infrastructure	The big cloud providers provide true infrastructure as code to provision (build and manage) all your resources (virtual machines, network, ect...). Often tightly paired and confused with with Configuration Management and CICD tools.		Infrastructure as Code		• Azure ARM (Azure Resource Manager Templates) • Amazon AWS CFN (Cloud Formation) Templates Higher level, Terraform and Vagrant (for Devs)
Optimized Operating System for Containers	Newish concept of lightweight transactionally updated operating system. Solaris had the transactional aspect a while back.				• Google CoreOS • Ubuntu Snappy • RedHat Atomic

Distributed Operating System for Containers	Similar in concept to what Hadoop technology solves for databases.		Elasticity		• Mesosphere Enterprise DC/OS
					  • Apache Mesos (distributed systems kernel)
					  • Apache ZooKeeper (distributed coordination)
					  • Apache Marathon (container orchestration)

Operating Virtualization	Docker focuses on ephemeral container's and single process as a practice for application isolation. However, LXC now LXD, diverged to focus on overall system density by isolation of the OS itself. Because of this, in my view, LXD lends itself to vendor packaged and data enterprise solutions. The technologies are designed to be compatible so you can take advantage of OS isolation through LXC with Docker running inside.		Cloud Density	LXD (LXC)	

Research

To watch this video - https://mesosphere.com/product/

Rackspace now provides consulting and support to build your own private cloud on OpenStack - http://www.rackspace.com/cloud/private_edition/

Rackspace even provides their Reference Architecture online - http://www.referencearchitecture.org/

Ubuntu has a program called Jumpstart for $9,000 for 5 days to help you build your own private cloud with UEC (Ubuntu Enterprise Cloud) previously powered by Eucalyptus now powered by OpenStack at http://www.ubuntu.com/cloud

This might be a worthwhile setup tutorial - http://cssoss.wordpress.com/2011/04/27/openstack-beginners-guide-for-ubuntu-11-04-installation-and-configuration/

https://mesosphere.com/ - Dickson recommended

https://www.ansible.com - Dickson recommended

Best Practices for Cloud from IBM - http://www.ibm.com/developerworks/websphere/techjournal/1404_brown/1404_brown.html

Cloud Infrastructure design strategies - http://realscale.cloud66.com/cloud-server-scaling-strategies/

MicroServices strategies - http://www.kennybastani.com/2016/04/event-sourcing-microservices-spring-cloud.html

Service Discovery Discussion - https://www.nginx.com/blog/service-discovery-in-a-microservices-architecture/

Very good article on IAC and differences btw Configuration Management and Provisioning, also declarative vs procedural tools - https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c

Looks at state challenges in relation to container technology - https://dzone.com/articles/containerizing-stateful-applications

12-Factor App... to read - https://12factor.net/

Good 2017 overview of Puppet and Chef - https://www.upguard.com/articles/puppet-vs.-chef-revisited

Adds on the above tools but not clear on what exactly - https://xebialabs.com/products/