What's Beyond Virtualization?

Derek Collison - @derekcollison
Apcera Inc.

QCon London - March 7, 2014

What does the future of Enterprise IT look like?

What does it look like today?

It Depends!

- Could be very old school - Just physical machines!
- Could be virtualized: CPU/MEM, Storage and Network
- Could be IaaS or IaaS++
- Could be IaaS + PaaS
- Could be off-premise and in the Cloud

Why do we care?

What do these things really buy us?

They remove
undifferentiated heavy lifting!

They speed up
the slow and mundane

Transparent value-add!

What about PaaS?

What about PaaS?

- Tries to speed up deployment
- Very opinionated, still is..
- Is only a piece of the larger puzzle
  - Carrying costs in the delivery pipeline
  - Pipeline is a biz requirement thru value delivered
- BUT PaaS as a standalone technology is not enough..

So..

What do we want?

What do we want..

- Self Service
- Faster iterative development and deployments
- Fault Tolerance, High Availability
- Higher and guaranteed SLAs
- Composeable Systems - legos

Software Defined Everything!

But..

What do we really want?

What do we really want?

- Meta-data driven description of my system
- Extreme Agility
- Transparent Compliance
- Fluid and Abstracted Infrastructure and Services
- Multiple delivery models in a single system

What do we want really?

- Meta-data driven description of my system
- Extreme Agility
- Transparent Compliance
- Fluid and Abstracted Infrastructure and Services
- Multiple delivery models in a single system

Meta-Data Driven

- My app A needs to talk to B and C
- I need 4 instances of A, 2 of B and 3 of C
- It needs X memory and Y CPU
- It needs NNN storage
- It requires I/O SLAs for talking to B and C
- It needs to be available via a URL for trusted identities
- It needs to run on-premise and co-located near B

Is this Autonomic Computing?

How would we do this?

How would we even start?

Getting Started - App A

- What does App A need?
- Where will App A be run?
- How will App A find B and C?
- How do others find my App A?
- What happens on a failures?

Getting Started - App A

- What does App A need? - Packaging and Dependencies
- Where will App A be run? - Provision and Schedule
- How will App A find B and C? - Addressing/Discovery
- How do others find my App A? - External Mapping
- What happens on a failures? - Health Monitoring

Packaging and Dependency

- What does the job need to run?
- What runtimes, OS, libraries?
- What tools can I use for consistency, compliance, audit?
  - SCCS and Chef / Puppet
  - AMIs or VMDKs
  - Docker Images

Packaging and Dependency Challenges

- Do these change when I change from Dev to Production?
- What runtimes, OS, libraries then?
- Who defines what these are?
- Are the existing tools and best practices still sufficient?

Provision and Schedule

- How fast can I provision?
- Can my workload run anywhere and be compliant?
- How do network perimeter security models effect placement?
- What is my unit or work? VM, App, Image?
- Can the system automatically handle compliance and policy?
- Can compliance and deployment be handled independently?
- What new tools exist? Mesos, Fleet?

Addressing and Discovery

- Is DNS sufficient?
- Do we need to change our applications?
- When things get moved, how does the system react?
- Is load balancing handled or is this a manual process?
- What happens when we scale up or down?
- How do others find us?

Monitoring and Management

- What happens when something fails?
- Is this a manual process?
- Who determines failure? Can we trust the system?
- What if they are sick, not dead? Latency vs Chaos
- Do we know if the change even helped?
- Pluggable Health

SO..
How do we get here?

Is it a Bolt-On Solution?

Or is it Bolt-In?

Bolt-On got us into this mess in the first place!

What we need is a Platform OS!

Programmable, pluggable and composeable..

From the inside out..

The OS for the datacenter

The OS for ~~the datacenter~~
Multiple Datacenters

Secure, Trusted, and Hybrid

Multi-Datacenter OS

- Treat all resources as a common pool
- Handle all networking access, addressing and discovery in realtime, and at scale
- Be aware of ontologies and their communication semantics
- Be security and policy aware
- Be purposely built to accept and promote rapid change
- Provide policy compliant resource isolation, connectivity and SLAs

Multi-Datacenter OS

- Virtualization
- SDN - Software-Defined Networking
- Management and Resource Pooling
- Intelligent and Compliant Job Scheduling
- Intelligent canarying, A/B rollouts

Multi-Datacenter OS

- Virtualization
- SDN - Software-Defined Networking
- Management and Resource Pooling
- Intelligent and Compliant Job Scheduling
- Intelligent canarying, A/B rollouts

Virtualization?

- What about speed and weight?
- Google chargeback diversion
- What about containers, e.g. Docker?
- Is there a container equivalent for .NET?
- Micro-task Virtualization?

Multi-Datacenter OS

- Virtualization
- SDN - Software-Defined Networking
- Management and Resource Pooling
- Intelligent and Compliant Job Scheduling
- Intelligent canarying, A/B rollouts

SDN?

- Solve network perimeter security?
- Does it involve application level changes?
- What about layer 7 semantics?
  - How many INSERTS per second from all of App A?
  - Can I disallow DROP and DELETE calls between 1a-3a?
- Can the network be made compliant and transparent?
  - It just works, e.g. mobile

Multi-Datacenter OS

- Virtualization
- SDN - Software-Defined Networking
- Management and Resource Pooling
- Intelligent and Compliant Job Scheduling
- Intelligent canarying, A/B rollouts

intelligent and Compliant Job Scheduling

- Pick the best place to run for a given job and policy
- How does a system rebalance, utilize new resources?
- Centralized or Distributed Algorithms?
- How does policy effect decision making? E.g Geo

Multi-Datacenter OS

- Virtualization
- SDN - Software-Defined Networking
- Management and Resource Pooling
- Intelligent and Compliant Job Scheduling
- Intelligent canarying, A/B rollouts

Intelligent Canarying

- Want to roll out a new version of App A
- Do we know what App A - v2 success looks like?
- How do we do roll in and roll back (if needed)?
- How do we avoid our fingers on the keyboard?
- What is needed for this process to be automated?

Intelligent Canarying

- What data is needed to say if it is ok?
  - resource utilizations - CPU, Mem, Storage
  - communication patterns - cascading effects
  - temporal awareness
- All data feeds into anomaly detection services
  - Utilizes unsupervised deep machine learning

Summary

Summary

- Intelligent, holistic platform technologies - Pluggable and Composeable
- Transparent value add to jobs/workloads - No code changes!
- Packaging and Dependency Management - Policy aware
- Job Scheduling and Provisioning - Also policy aware
- Addressing, Discovery, Networking - Policy again, theme developing
- Monitoring and Management
- Lifecycle Management and Intelligent Canarying

Some Resources

- Docker - https://www.docker.io/
- Mesos - http://mesos.apache.org/
- CoreOS - https://coreos.com/
- Fleet, Etcd - https://coreos.com/using-coreos/etcd/
- Continuum - http://apcera.com/continuum/

Thank You