Cloud Computing

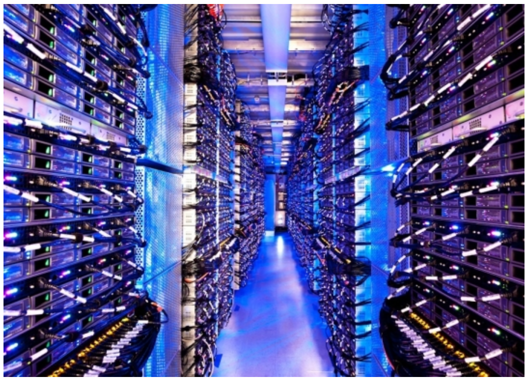
- Part 1: Data centers
- Part 2: Cloud Computing
- Part 3: Kubernetes and Orchestration

Part 1: Data Centers

- Large server and storage farms
  - 1000s of servers
  - Many TBs or PBs of data
- Used by
  - Enterprises for server applications
  - Internet companies
    - Some of the biggest DCs are owned by Google, Facebook, etc
  - Cloud Computing Platforms
- Used for
  - Data processing
  - Web sites
  - Business apps

Inside a Data Center

- Giant warehouse filled with:
- Racks of servers
- Storage arrays
- Cooling infrastructure
- Power converters
- Backup generators

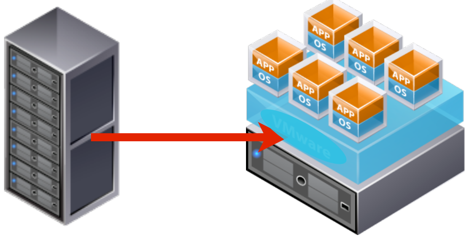


Traditional vs "Modern"

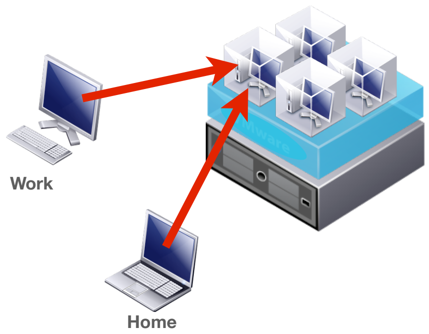
- Data Center architecture and uses have been changing
- Traditional - static
  - Applications run on physical servers
  - System administrators monitor and manually manage servers
  - Use Storage Array Networks (SAN) or Network Attached Storage (NAS) to hold data
- Modern - dynamic, larger scale
  - Run applications inside virtual machines
  - Flexible mapping from virtual to physical resources
  - Increased automation allows larger scale

Virtualization in Data Centers

- Virtual Servers
  - Consolidate servers
  - Faster deployment
  - Easier maintenance

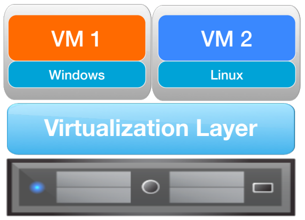


- Virtual Desktops
  - Host employee desktops in VMs
  - Remote access with thin clients
  - Desktop is available anywhere
  - Easier to manage and maintain



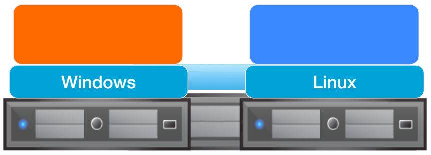
Server Virtualization

- Allows a server to be “sliced” into Virtual Machines
- VM has own OS/applications
- Rapidly adjust resource allocations
- VM migration within a LAN



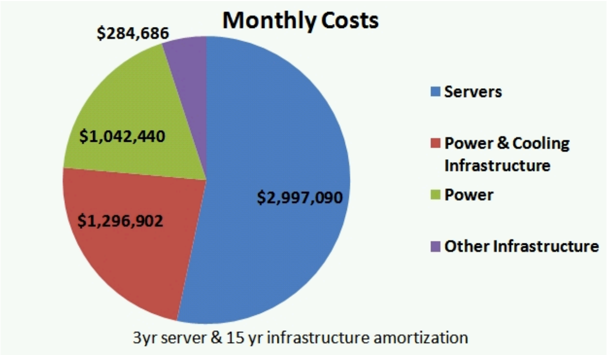
Xen VirtualBox KVM

Parallels vmware



Data Center Costs

- Running a data center is expensive
- Efficiency captured as PUE (Power Usage Effectiveness)
  - Ratio of Total Power / IT Power   (typical: 1.7, Google PUE ~ 1.1)

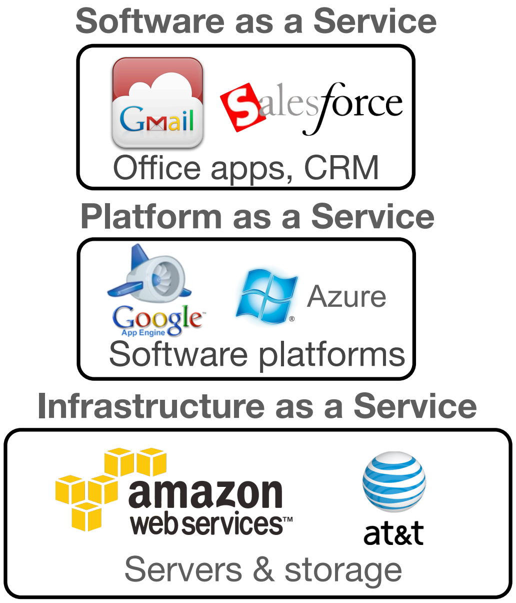


http://perspectives.mvdirona.com/2008/11/28/CostOfPowerInLargeScaleDataCenters.aspx

Part 2: Cloud Computing

- Cloud computing: use of remote servers to run distributed applications
- Cloud computing platform
  - Data center where remote resources can be leased by any user or company
    - No need to create and deploy own data center and IT infrastructure
- Benefits:
  - Remotely available from the Internet
  - Pay as you go
  - Highly scalable: obtain resources on-demand
  - Shared infrastructure and economy of scale

The Cloud Stack



Software as a Service

Office apps, CRM

Hosted applications
Managed by provider

Platform as a Service

Software platforms

Platform to let you run
your own apps
Provider handles scalability

Infrastructure as a Service

Servers & storage

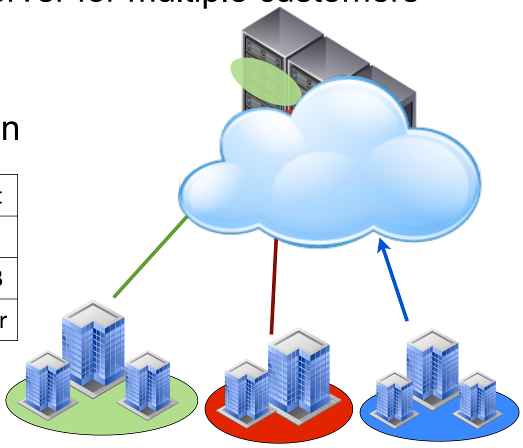
Raw infrastructure
Can do whatever you
want with it

IaaS: Amazon EC2

- Rents servers and storage to customers
  - Uses virtualization to share each server for multiple customers
  - Economy of scale lowers prices
  - Can create VM with push of a button

	Smallest	Medium	Largest
VCPUs	1	5	33.5
RAM	613MB	1.7GB	68.4GB
Price	$0.02/hr	$0.17/hr	$2.10/hr

Storage	$0.10/GB per
Bandwidt	$0.10 per GB



Types of IaaS Instances

- **On-demand** instances
  - Provision on-the-fly
  - Pay by the minute
  - Keep until terminated
- **Reserved** instances
  - Long-term commitment for on-demand server: 1 year, 3 year
  - Discount over on-demand pricing
- **Spot** instances
  - Excess capacity sold by cloud platform at high discount
  - Can be revoked by cloud provided with a warning time
    - Take back server if regular customers need it
    - Cheap method to run large computations in off-peak periods

PaaS Cloud

- Cloud resources offered as highly scalable run-time platform
  - Application developers provide code
  - Platform deploys code, provisions resources,
  - Platform can also autoscale the application
  - Language supported: Python, Java, Node, .NET

  - Users do not need to provision or manage servers resources
  - Billing based on workloads or usage
  - Serverless computing has similarities to PaaS

Serverless Computing

- Developer: deploy code on a platform (eliminate need to provision servers)
- Cloud: platform can each application up or down automatically (elastic scaling)
  - Scale down to zero possible when idle
- One popular form: function-as-a-Service (FaaS)
  - Write code as a set of “functions” and deploy each function
    - functions can be chained together
    - Functions are often stateless
    - More fine-grained than micro-services
    - AWS Lambda: FaaS in the cloud

Public, Private, Hybrid Cloud

- Not all enterprises are comfortable with using **public cloud** services
  - Don’t want to share CPU cycles or disks with competitors
  - Privacy and regulatory concerns
- **Private** Cloud
  - Use cloud computing concepts in a private data center
    - Automate VM management and deployment
    - Provides same convenience as public cloud
    - May have higher cost
- **Hybrid** Cloud
  - Move resources between private and public depending on load
  - Cloud Bursting

Cloud Workloads

- Client/Server
  - Web servers, databases, CDNs, etc
- Batch processing
  - Business processing apps, payroll, etc
- Data processing and analytics
  - Data intensive computing: map reduce, spark
  - Scalability concepts built into programming model
- AI workloads: ML training
  - Use servers with GPUs
- High performance computing: specialized instances

Cloud Storage

- Lease storage from cloud platforms
- Object storage: blobs of storage that use get() and put()
- Block storage / server disk — local storage for IaaS servers
- File Storage: network file system storage
  - Can be shared across machines, not tied to a machine
- Archival storage — used for backups
- Other models
  - Dropbox: cloud storage for end-user machines; automatic sync
  - Google Drive, OneDrive, Box,
  - Cloud backups, Cloud media storage

Cloud Orchestration

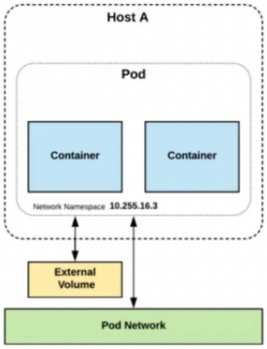
- Cloud controller: similar to K8s controller
  - Customer requests one or more instances
  - Create virtual machines on cloud servers
  - Configure networking and storage
  - Boot VM using specified images
- IaaS platforms now support containers and VMs
  - Container orchestration similar to k8s but for third party users

Part 3: Kubernetes (k8s)

- Cluster management using containers
- Container-based **Orchestration System**
  - Based on Google's Borg /Omega cluster managers
- Applications are containerized
- K8s will deploy them onto machines of the cluster
  - **Replicate** app on multiple machines if requested
  - **load balance** across replicas
  - Can **scale up** or down dynamically (vary replica pool size, a concept similar top dynamic thread/process pools)
  - Provide automated **restart** upon detecting failure (self-healing)

K8s Pods

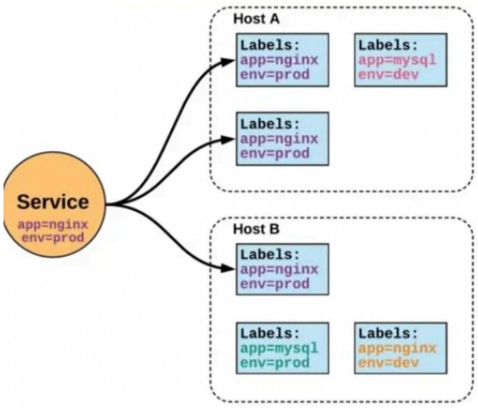
- Pod: contains one or more containers that share volumes and name space
  - Pods: smallest granularity of allocation in k8s.
- Distributed application: multiple components,
  - each component inside a container
  - Each pod consists of one or more components / containers
  - Pod can contain all containers of an application but:
    - If a component needs to be scaled, put each such component in a separate pod
  - Application consists of a set of pods, each independently scalable
    - Pods of an application can span multiple cluster machines



All k8s figures courtesy of
https://www.slideshare.net/rishabhindoria52/introduction-to-kubernetes-139878615

k8s Services

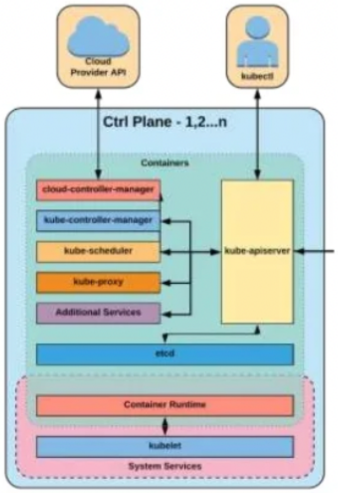
- service: method to access a pods's exposed interfaces
  - static cluster IP address
  - static DNS name
  - Services are not ephemeral
  - collection of pods
- Pods are ephemeral
  - each has its own IP
  - can be migrated to another machine
  - Pods can communicate with one another using this IP



All k8s figures courtesy of
https://www.slideshare.net/rishabhindoria52/introduction-to-kubernetes-139878615

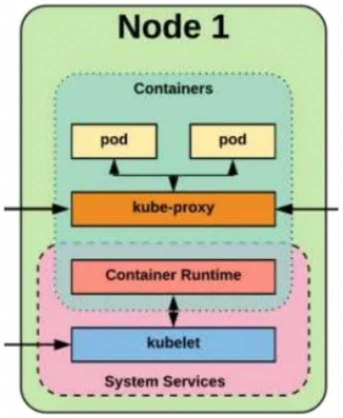
Control Plane

- **apiserver**: REST interfaces for clients to access management interface
- **etcd**: cluster key-value datastore
  - strongly consistent, highly durable (uses RAFT consensus)
- **controller-manager:** replicate pods, monitor for node failures and restart
- **scheduler:** assigns newly created pods to servers based on resource constraints
- **cloud-controller-manager**: interact with cloud platforms



K8s Node

- kubelet: agent on each node
  - ensure containers are running and healthy
- kubelet proxy
  - Manage network rules
  - Load balancing for cluster services
- container runtime
  - runtime for container execution
  - containerd/docker, cri-o, rkt



containerd

- Container orchestration runtime that is basis for docker, k8s and many other systems: for “lifecycle management”
  - Designed to be used as part of a larger system
  - Used by google, amazon, azure, IBM, docker, …

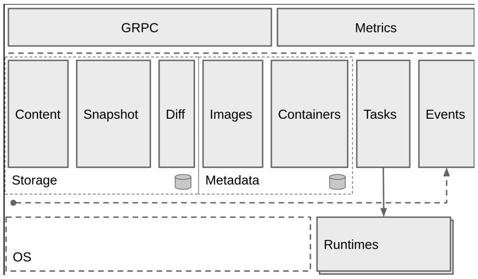


Fig courtesy https://github.com/containerd/containerd