

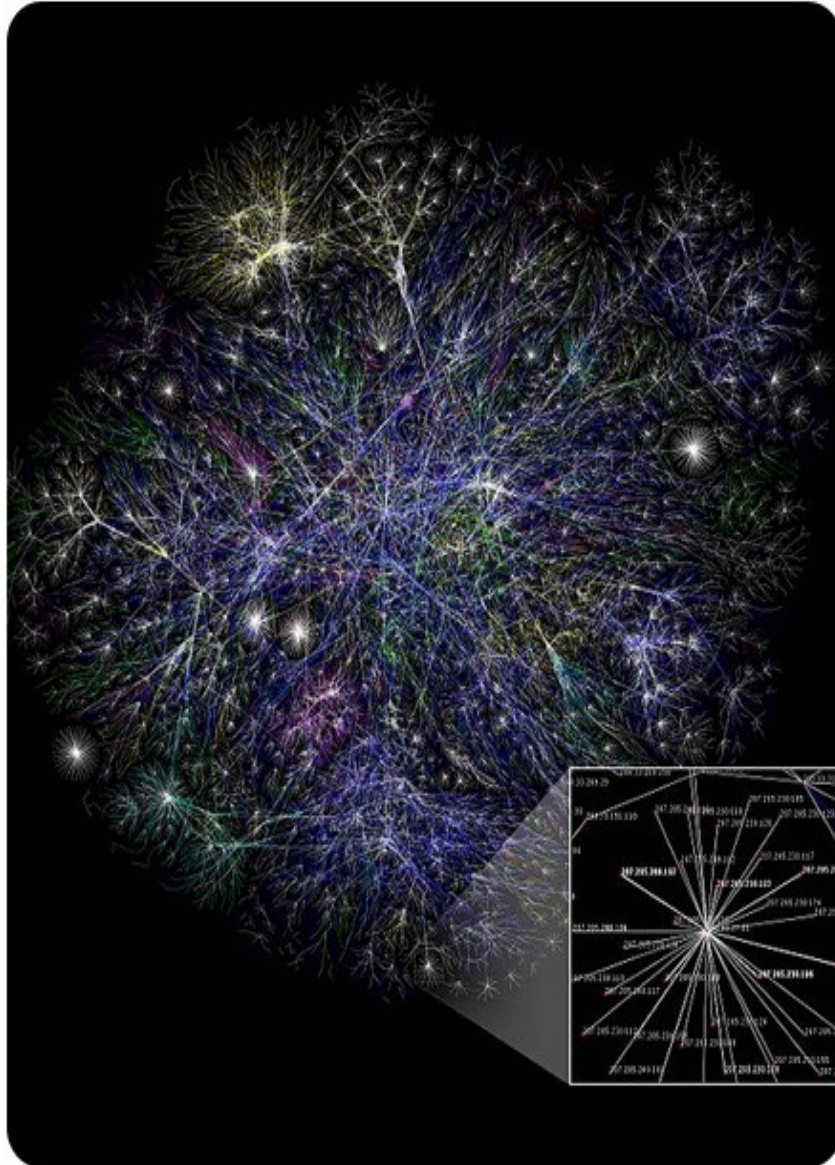
# Sun Cloud API: A RESTful Open API for Cloud Computing

Lew Tucker

CTO, Cloud Computing

Sun Microsystems, Inc.

# Future vision: Global Cloud of Clouds (a.k.a “InterCloud”)



- Inter-connected network of servers, storage, and applications
- Segmented into public and private clouds
  - > For security
  - > For predictability
  - > For regulatory compliance
- Unified and driven by a set of protocols, software APIs, and services
- Global and open

# Three Views on Cloud Interoperability

## 1) We are already there

- Already apps built on top of multiple services
  - Force.com + AWS + Google AppEngine + Maps + Twitter
- Programming language libraries hide service differences
- Agreed upon “language of the net”: HTTP/TCP/IP

## 2) Need much more for application portability

- “Common cloud API” to avoid vendor lock-in and let apps easily move from one provider to another
- Agreement upon a single virtual machine image format
- ...

## 3) Skeptic

- Vendors will always resist out of fear of commoditization

# Sun Cloud Objectives

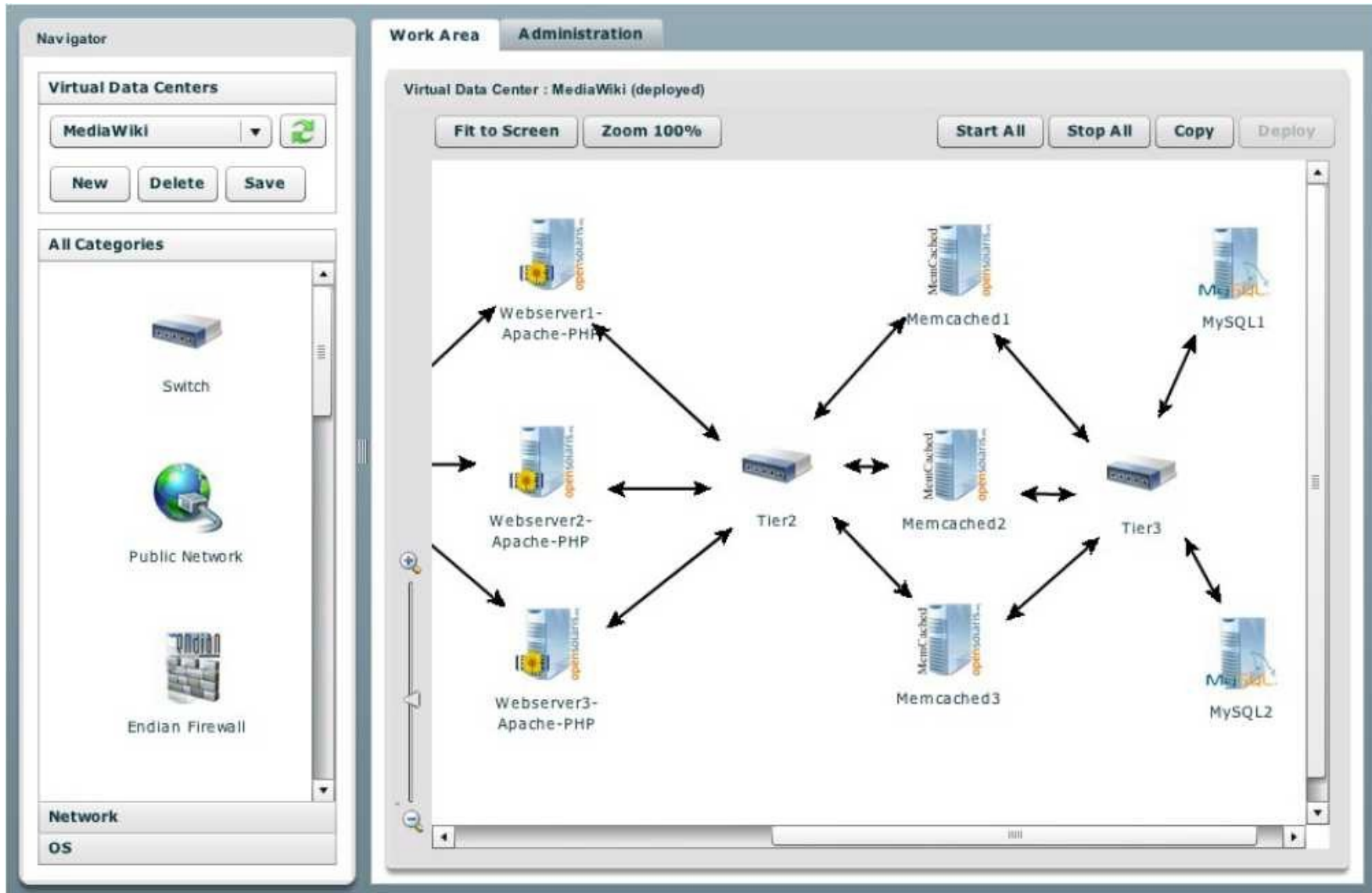
- Build service and APIs covering basic infrastructure services
- Start with IaaS - add additional services over time to build out an application development platform for cloud computing
- **Support many different kinds of clouds**
  - > offered by customers
  - > different service providers
- Simple as possible, easily extended
- Work with standards – use what exists, avoid arbitrary invention

Big question? Why not just follow EC2, S3, ...?

# What we developed

- Storage service – two protocols
  - > WebDAV (standard) + admin APIs
  - > Amazon S3 compatible service
- Compute service – needed something different
  - > Support a rich set of resources including virtual networks
  - > Adopt standard Virtual Machine Image format: OVF (DMTF)
  - > RESTful API
  - > Published under Created Commons

# Sun Cloud: Each customer gets their own “virtual data center” in the cloud



# Sun's Set of Cloud Services

## Compute Service

Virtual  
Machines  
Networking  
Storage

## Virtual Datacenter

Resources, People,  
Graphical UI

## Open API

Public, RESTful,  
Java, Python, Ruby

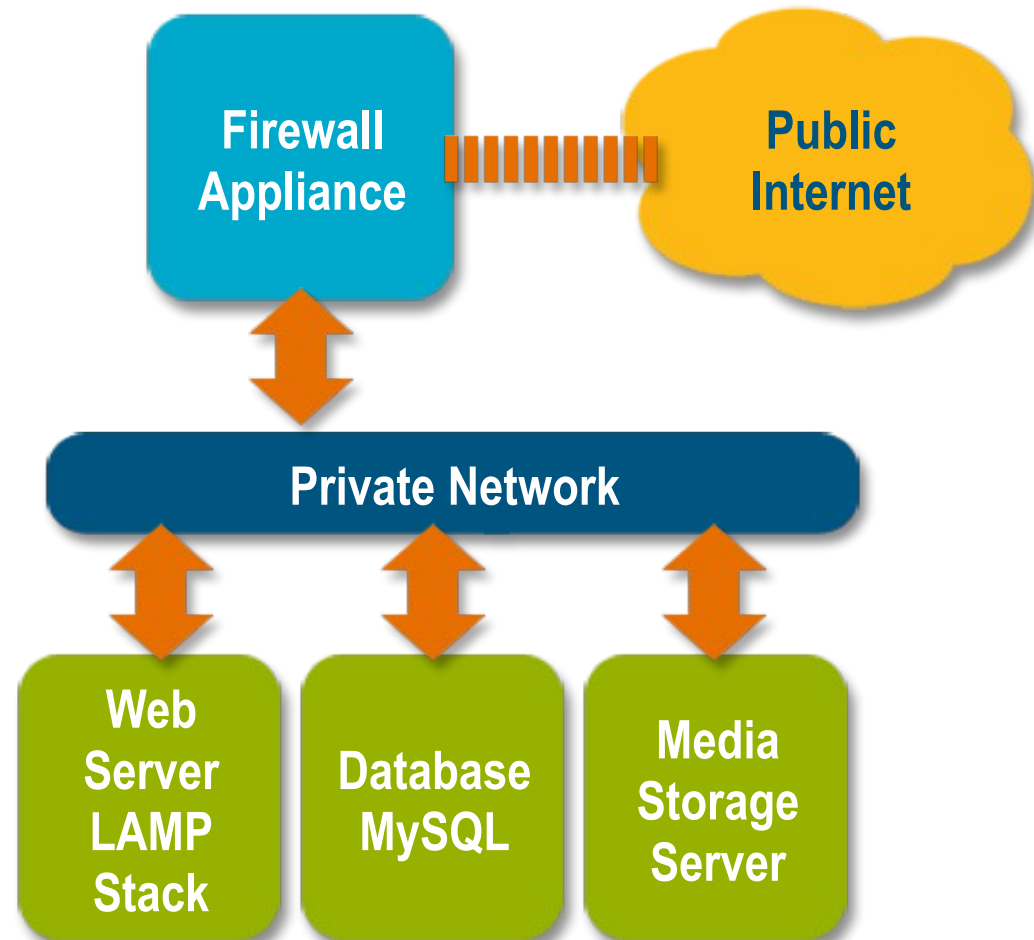
## Storage Service

Volumes  
Objects

Protocols:  
WebDAV  
S3

# Sun Cloud RESTful API

- Virtual model of a real data center for developers, system operators, QA-test
- Highly elastic and dynamic
- Easy to create, save load, stop, start entire applications
- Released to community for input on March 18th, 2009



# Sun Cloud API use of REST

- Uses basic verbs of HTTP
  - > GET (Read), POST (Create), PUT (Update), DELETE (Delete)
- URIs are discovered at run-time and opaque
  - > User start with a single URI for their own cloud
  - > Field values returned in resource representations hold service-defined URIs
  - > Allows each service provider to define their own URI space
- HTTP protocol using JSON
  - > Allows usage of any programming language

# Virtual Resources

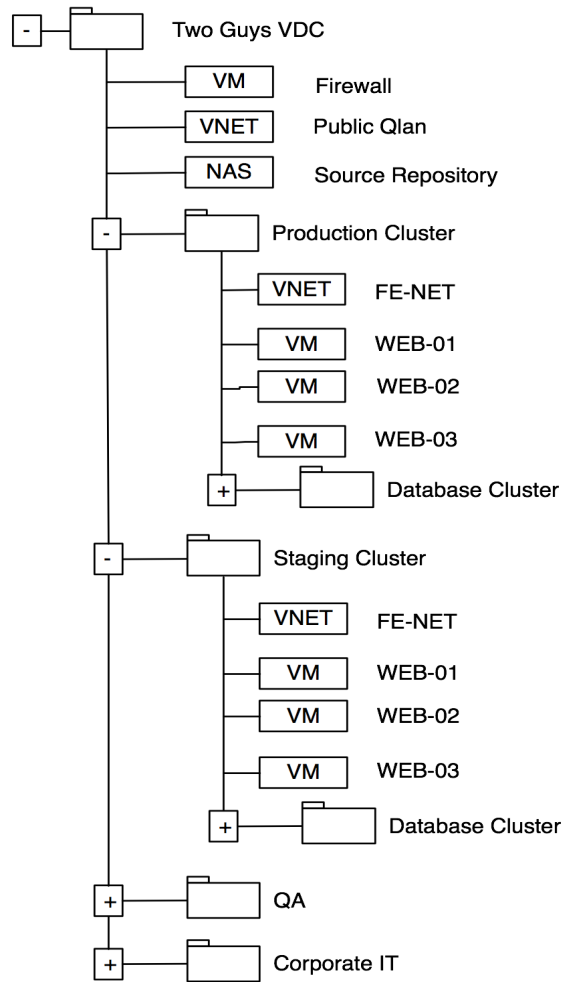
- Resource types:
  - > Cloud (user account)
  - > Virtual Data Center
  - > Cluster
  - > Virtual Machine
  - > Private Virtual Network
  - > Public Addresses
  - > Storage Volumes
  - > Volume Snapshots
- CLI + Language bindings for Java, Ruby, Python

# Need collections of resources in a VDC

- computers, networks, storage

- Emphasis is on groups of resources which serve a particular application or business purpose
  - > Web site, Dev-test, HR application, DB tier
- Owned and managed by different people
  - > Engineering QA, Development
  - > Production, Finance
- Apply single action to a group
  - > Copy
  - > Backup
  - > Shutdown

# Hierarchy of Resources: “clusters”



Folder-based organization of all resources within a customer's virtual data center

- Supports organizing resources by
  - > organization (mktg, dev, finance)
  - > application (complex, multi-server app)
  - > cluster (production, staging, test)
- Access rights assigned to set of resources
- Stop, start, suspend, or other operations applied to set of resources
- Cloning and distribution of entire applications

# In cloud computing, resources may change state

- Breaks usual REST resource model
  - > Use “controller” resources to represent requests to change state
  - > Think of pushing “reset”, “power”, etc., on the resource
- Many operations are asynchronous
  - > Immediately return standard HTTP status codes
  - > Plus a URI for checking job progress
  - > Poll this progress URI to know when job is done
- All URI's except for the first, are dynamically generated
  - > Exposes only those operations that are allowed

# Example: GET VM Properties and CTRLR

**GET /vdc/vm001**

Host: cloud.sun.com

**// RESPONSE: resource description**

HTTP/1.1 200 OK

Content-Type: application/vnd.com.sun.cloud.VM+json

{

"name" : "Database"

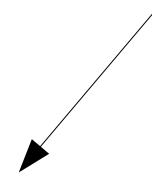
"uri": "/vdc/vm001",

"run\_status" : "HALTED",

"description" : "MySQL host",

"tags" : [ "sql" ]

**Controller  
URI  
(opaque)**



"controllers" : [ "start": "/vdc/ops/23:32:2x:3" ]

"interfaces" : [

{ "vnet": "/vnets/10.31.145.0",

"mac\_address": "00:16:3E:08:00:92",

"ip\_address": "144.31.100.15",

"nic": "eth0"

}]}

# Example: Create VM based on a template

**POST /vdc/**

Host: xrgy.cloud.sun.com

Authorization: Basic xxxxxxxxxxxxxxxxxxxxxx

```
{
  "name" : "Database"
  "from_template" :
    "http://cloud.sun.com/resources/vmtemplates/003",

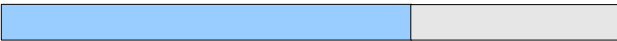
  "description" : "MySQL host",
  "tags" : [ "sql" ]
}
```

**// RESPONSE: request accepted - returns job status URI to poll**

HTTP/1.1 202 Accepted

Content-Type: application/vnd.com.sun.cloud.Status+json

```
{
  "op": "new-VM",
  "progress": 20,
  "target_uri": "/vdc/vm001",
  "status_uri": "/vdc/status?op=new-vm&vm=001"
}
```



# VM Provisioning – interesting questions

- Should all VMs be internet accessible?
- Should users have any control over placement?
- How to override machine image defaults?
- How to pass in startup-parameters, startup scripts, etc.?
- Role of scripting and recipes in provisioning?

# Closing thoughts

- APIs are long lived
  - > All permutations will be tried
  - > Always used in ways never imagined
- Security issues are still the #1 inhibitor
  - > Automation removes human “reasonableness” test
  - > Developers often make stupid mistakes
- We all have nearly identical conceptual models at the lowest, IaaS level
- Need Open APIs to grow adoption
  - > Plenty of room for innovation by service providers in what they may offer

# More information

<http://kenai.com/projects/suncloudapis/>

Lew Tucker  
Sun Microsystems, Inc