

中国对外援助培训项目教材（讲义）
Reference Material for China-Aid Training Programs

发展中国家云环境下大数据的管理及应用研修班

**Seminar on the Management and Application
of Big Data in Cloud Computing
for Developing Countries**

Cloud Computing Architectures and Applications

主办单位：中华人民共和国商务部

Sponsor: Ministry of Commerce of the People's Republic of China

实施单位：贵州科学院

Organizer: Guizhou Academy of Sciences

2015年7月15日 8月4日

From July 15th to August 4th, 2015

中国 · 贵阳

Guiyang·China

发展中国家云环境下大数据的管理及应用研修班

**Seminar on the Management and Application of Big Data in Cloud Computing
for Developing Countries Course**

Cloud Computing

Architectures and Applications

Li Dan

Assoc. Research Prof. of Guizhou Academy of Sciences

July, 2015

Content

1. Introduction of Cloud Computing
2. Techniques of Cloud Computing
3. Demo: Cloud App Development

Auditor: Ma Xinqiang, Assoc. Prof. of Guizhou Academy of Sciences, July,2015

中国 · 贵阳 Guiyang·China

Cloud Computing Architectures and Applications

- 1. Introduction of Cloud Computing**
- 2. Techniques of Cloud Computing**
- 3. Demo: Cloud App Development**

Objectives

- Understand the basic ideas of cloud computing
- Understand the basic concepts and techniques of cloud computing
- Feeling how a cloud application be developed and deployed.

Cloud Architectures & Applications

Introduction of Cloud Computing

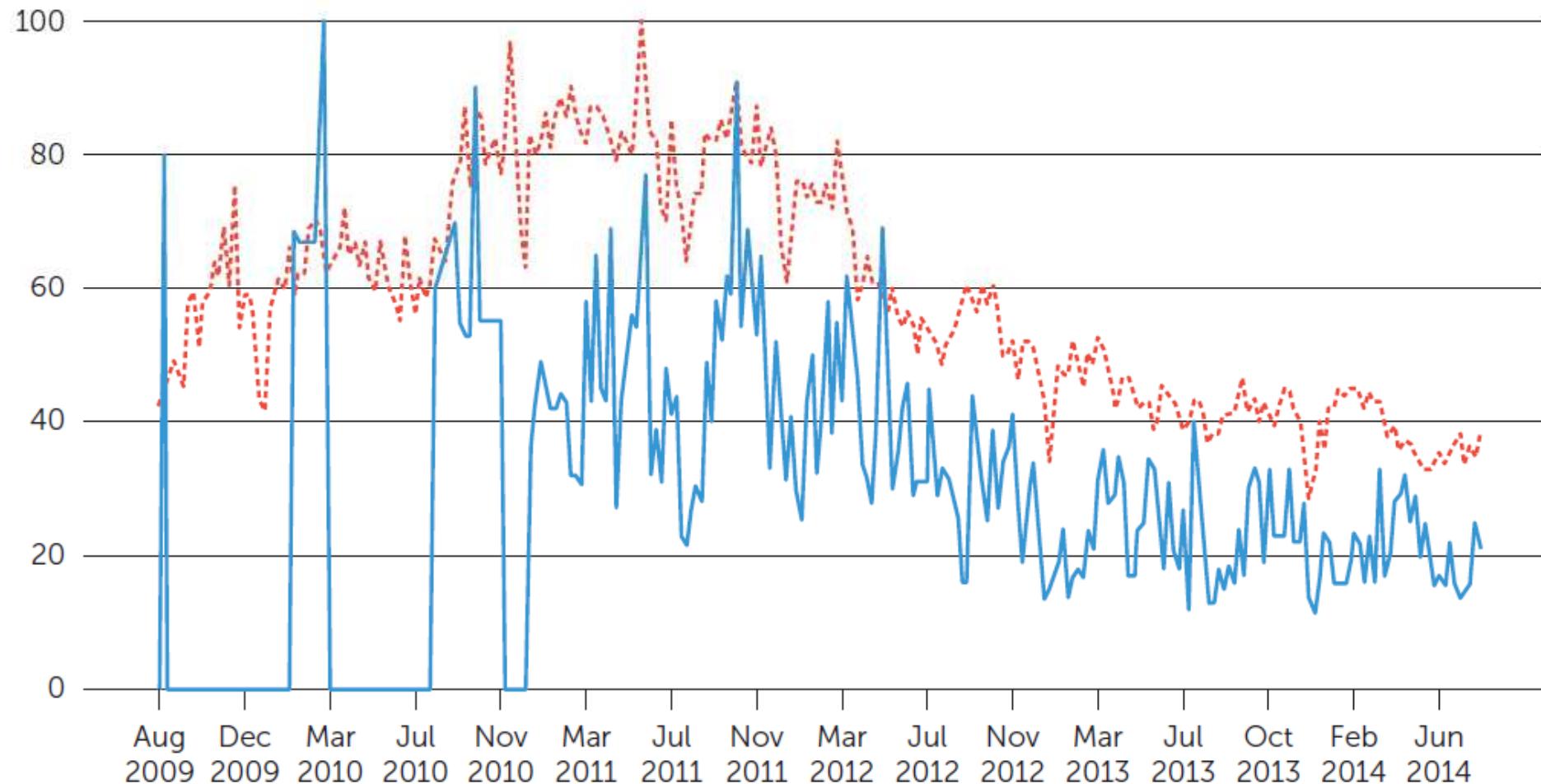
The Cloud is Coming !

- The cloud technologies are becoming inseparable part of our life.
- Cloud is a metaphor for the internet
- The world is moving towards the cloud!
- Software developers will also jump into the cloud: now or later, it will happen
 - This year, or few years later, everyone will develop applications for the cloud

Cloud Computing

- Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. --- [NIST Definition](#)
- Cloud computing has five essential **characteristics**, three **deployment models**, and three **service models**.

Google Search for “Cloud Computing”

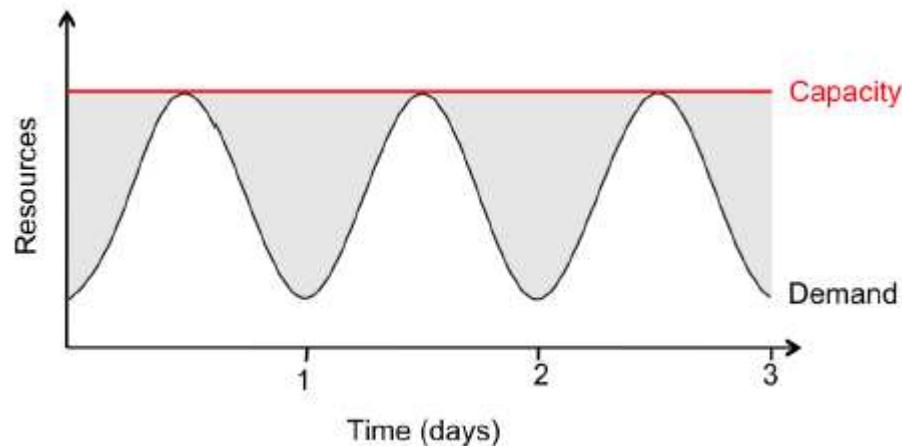


Before the Cloud

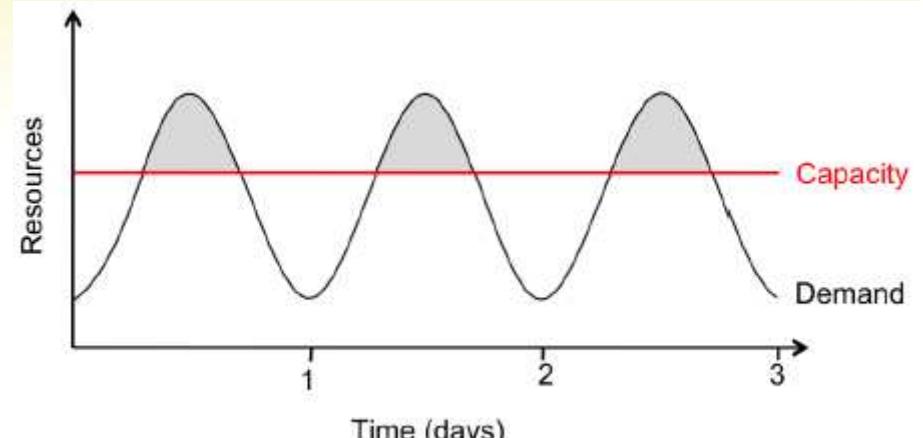
- If you wanted to start an enterprise app, you needed an IT shop
- Massive costs in hardware, software, power, administrative staff
- Prohibitive cost to entry



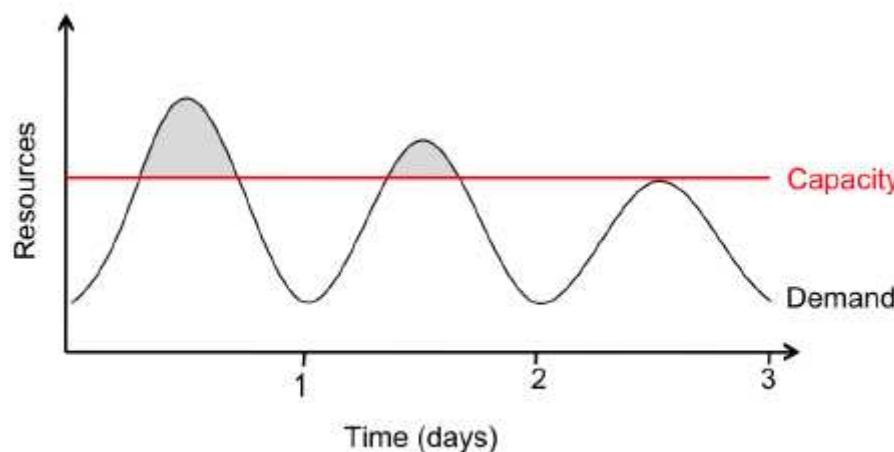
Variability for IT Resources



(a) Provisioning for peak load



(b) Underprovisioning 1

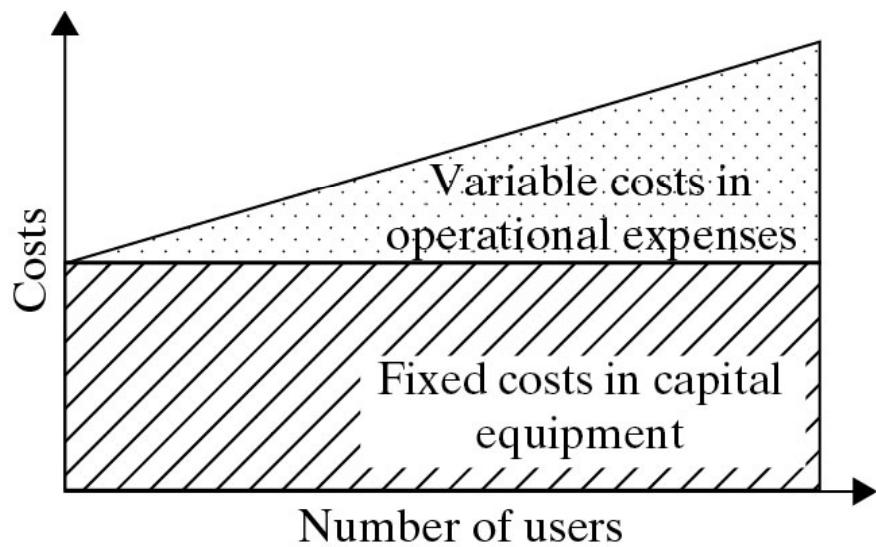


(c) Underprovisioning 2

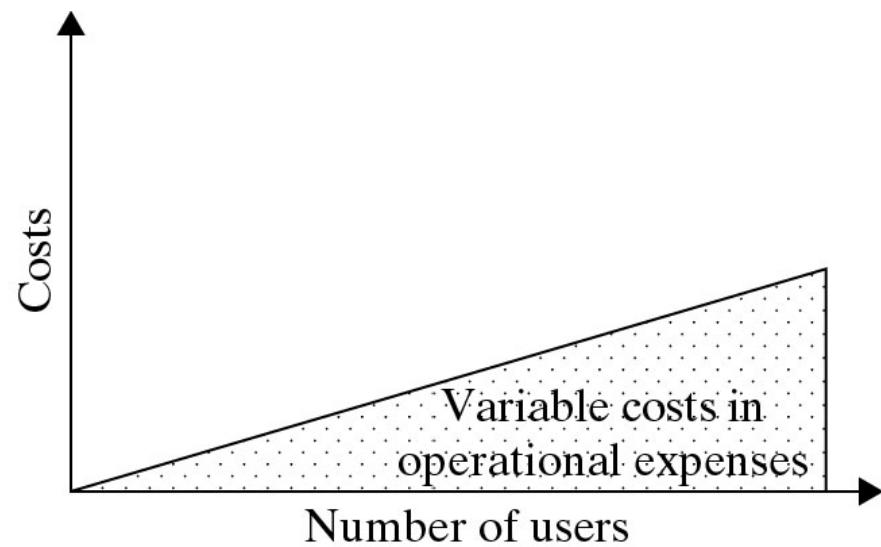
How the Cloud Works?

- In the cloud everyone consumes a portion of the *shared computing resources*
 - CPU, memory, storage, IO, networking, etc.
- If your business is small, you consume less
 - If your business is growing, you consume more resources from the cloud
- Pay as you go
 - Start for free, pay when you grow and need more resources

Cost-Effective of Cloud vs. Traditional

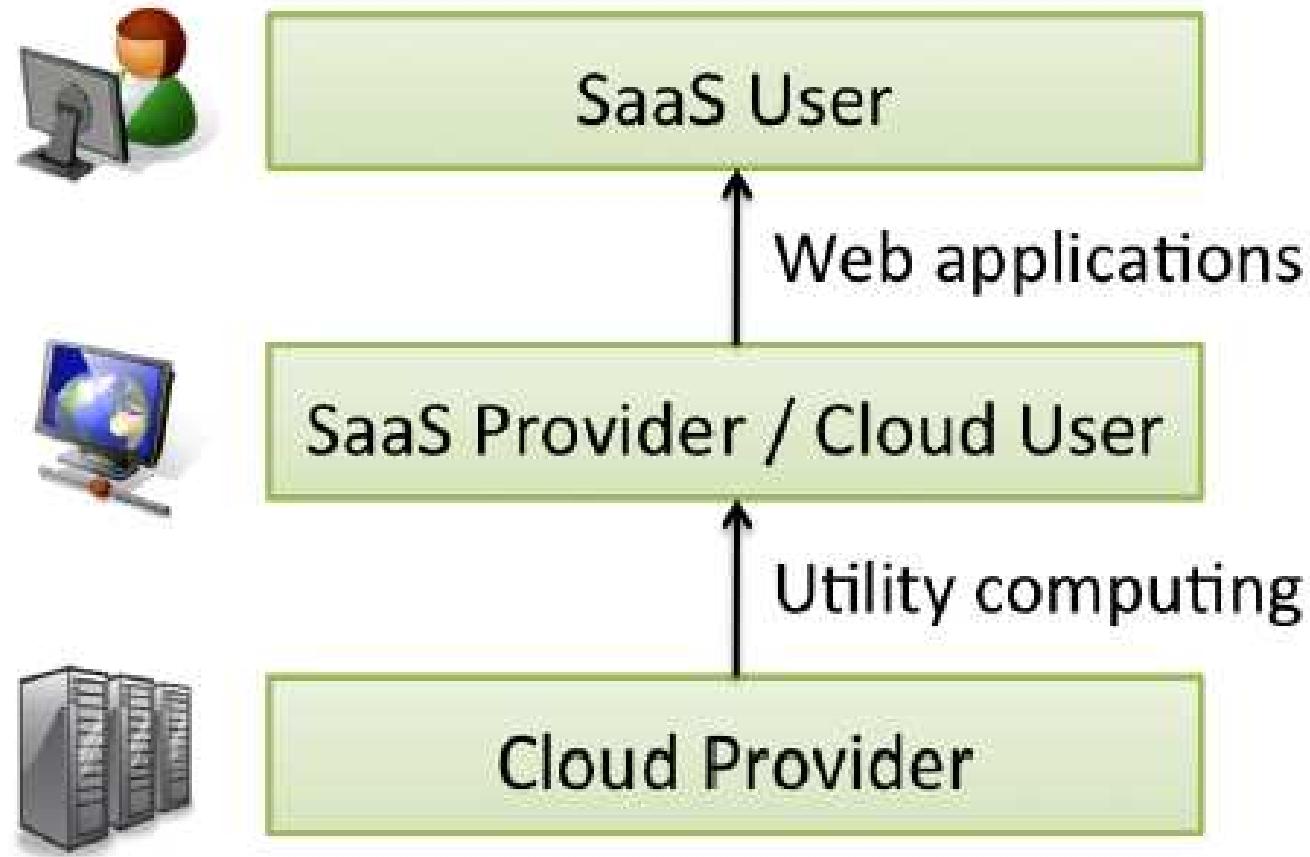


(a) Traditional IT cost model

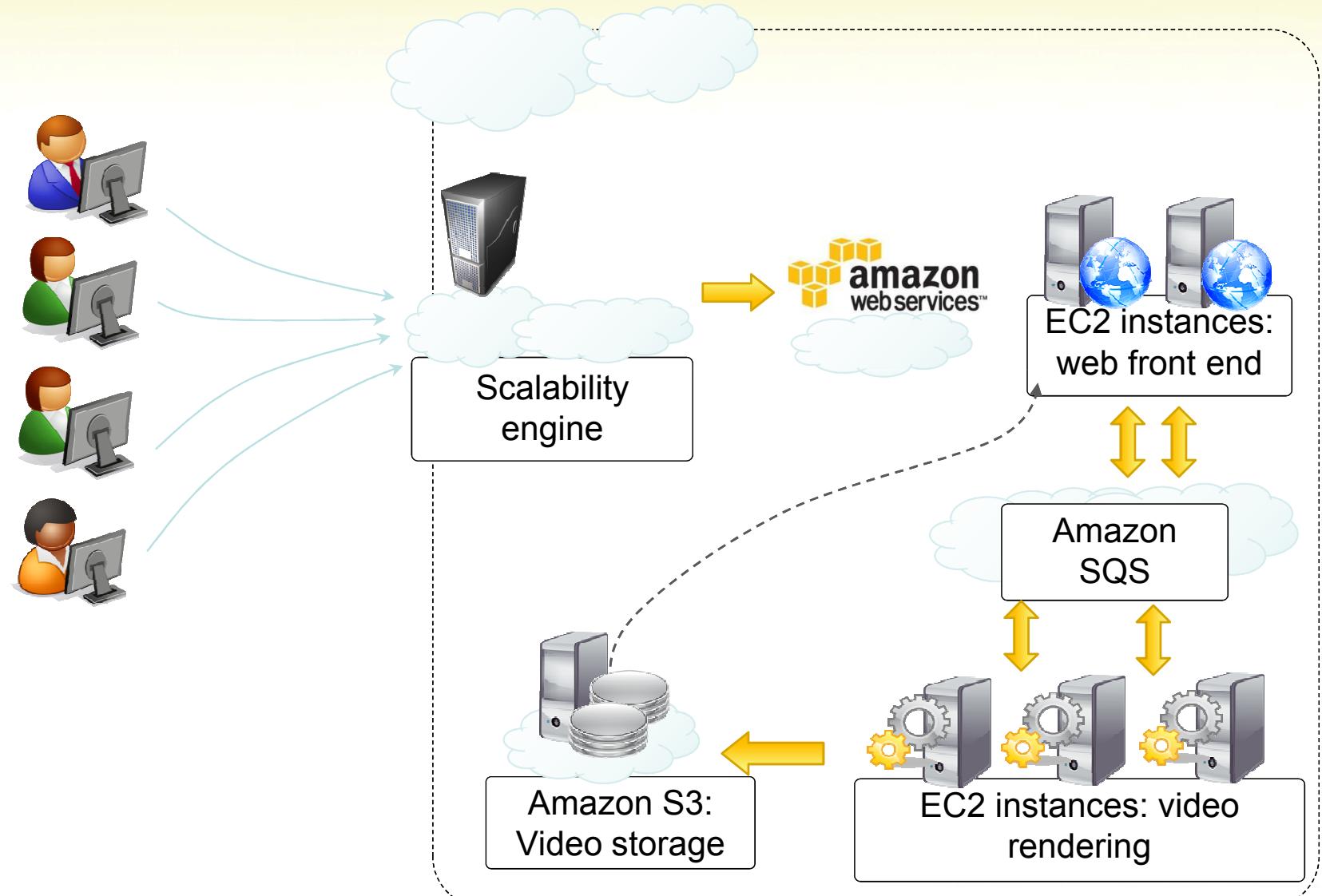


(b) Cloud computing cost model

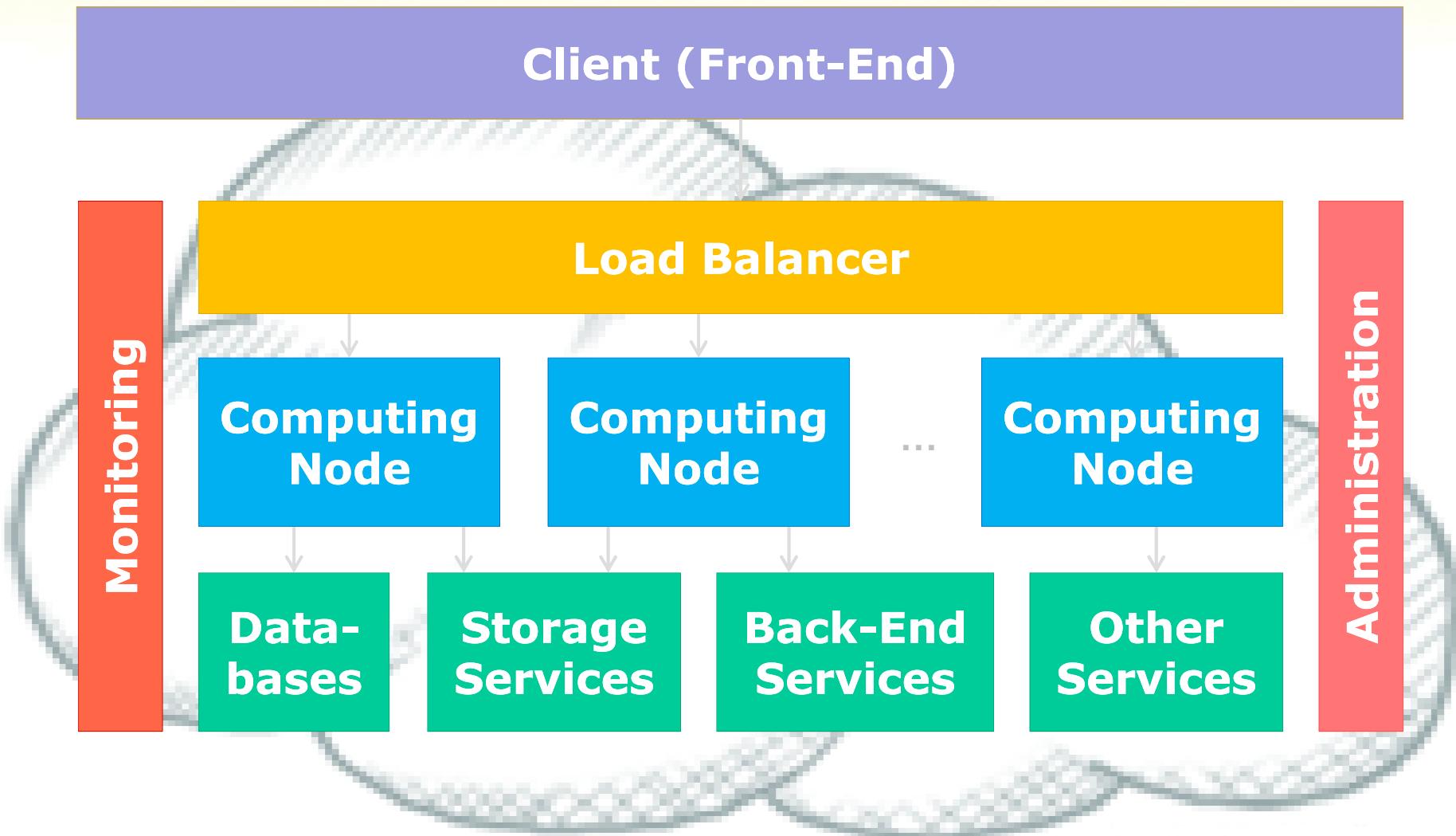
Users and Providers of Cloud



Example Cloud Application



Typical Cloud App Architecture



Why Cloud Computing is Appealing

- Cost
- Consumerization of IT
- Scalability and availability of services
- Sustainability or green IT

Benefits of Cloud Computing

Availability

- Access data anywhere, anytime through standard internet connection

Cost reductions

- Pay as you go model
- Savings on maintenance, repairs, and upgrades

Mobility

- Can access data and application from various devices
- Connecting employees, partners, and suppliers globally

Scalability

- Users have access to resources that scale quickly based on their demand

http://www.businessweek.com/magazine/toc/09_24/B4135cloud_computing.htm

<http://www.sei.cmu.edu/library/assets/whitepapers/Cloudcomputingbasics.pdf>

@2015, Li Dan, Guizhou Academy of Sciences

Green Computing of Cloud

Technology	Cost in Medium-sized DC	Cost in Very Large DC	Ratio
Network	\$95 per Mbit/sec/month	\$13 per Mbit/sec/month	7.1
Storage	\$2.20 per GByte / month	\$0.40 per GByte / month	5.7
Administration	≈140 Servers / Administrator	>1000 Servers / Administrator	7.1

- Data for 2006
- Medium-sized datacenter: 1000 servers
- Very large datacenter: 50,000 servers

Source: Above the Clouds: A Berkeley View of Cloud Computing

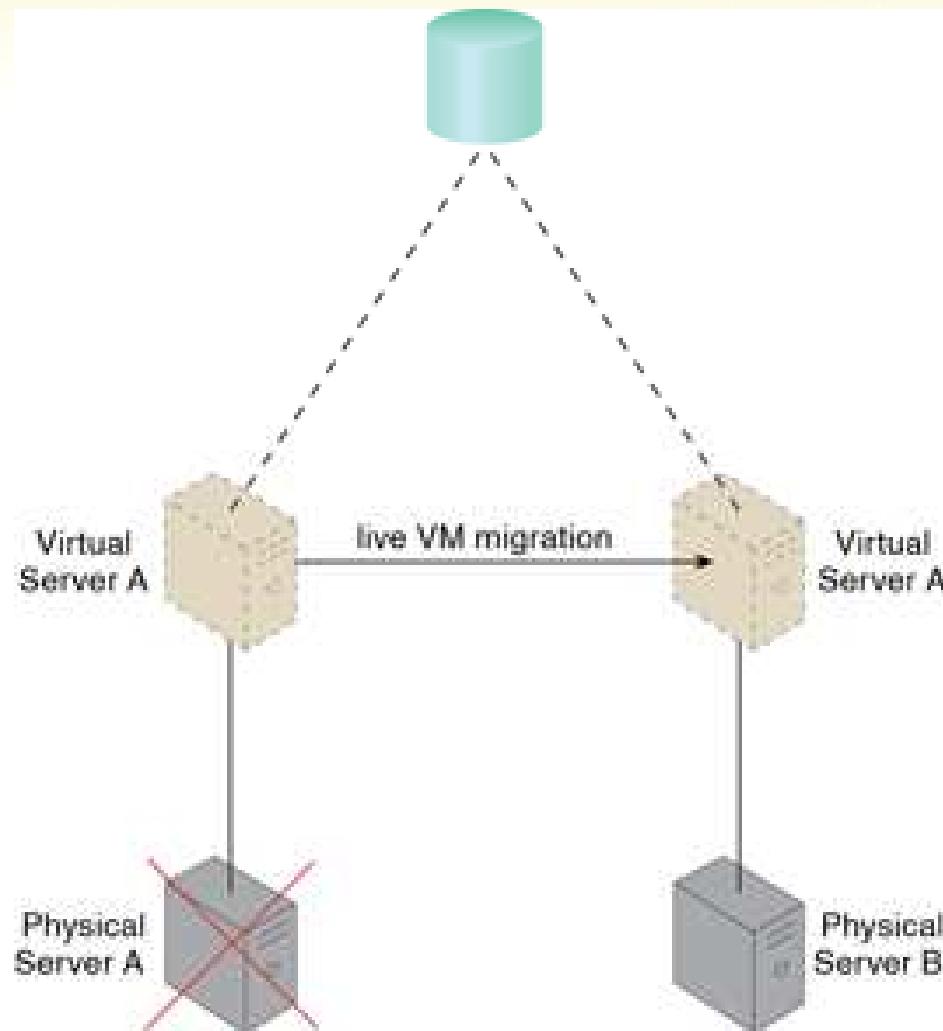
Essential Characteristics of Cloud

1. Broad network access
 - You can access the cloud from anywhere
2. Resource pooling
 - You work with virtual machines that could be hosted anywhere
3. Rapid elasticity
 - Easily go from 5 servers to 50 or from 50 servers to 5
4. On-demand self-service
 - You get elasticity automatically
5. Measured service
 - You pay for what you use

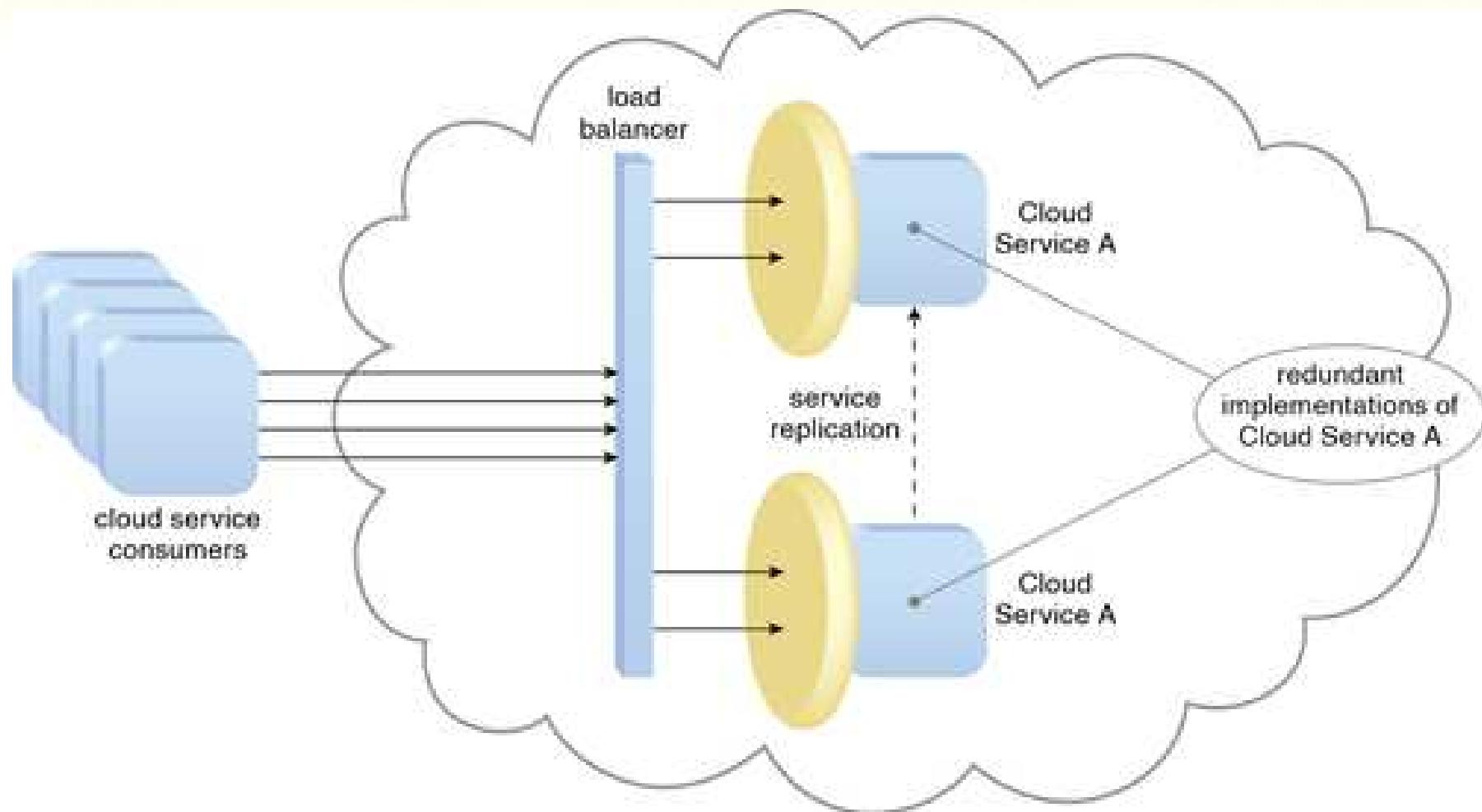
Source: NIST Working Definition of Cloud Computing

@2015, Li Dan, Guizhou Academy of Sciences

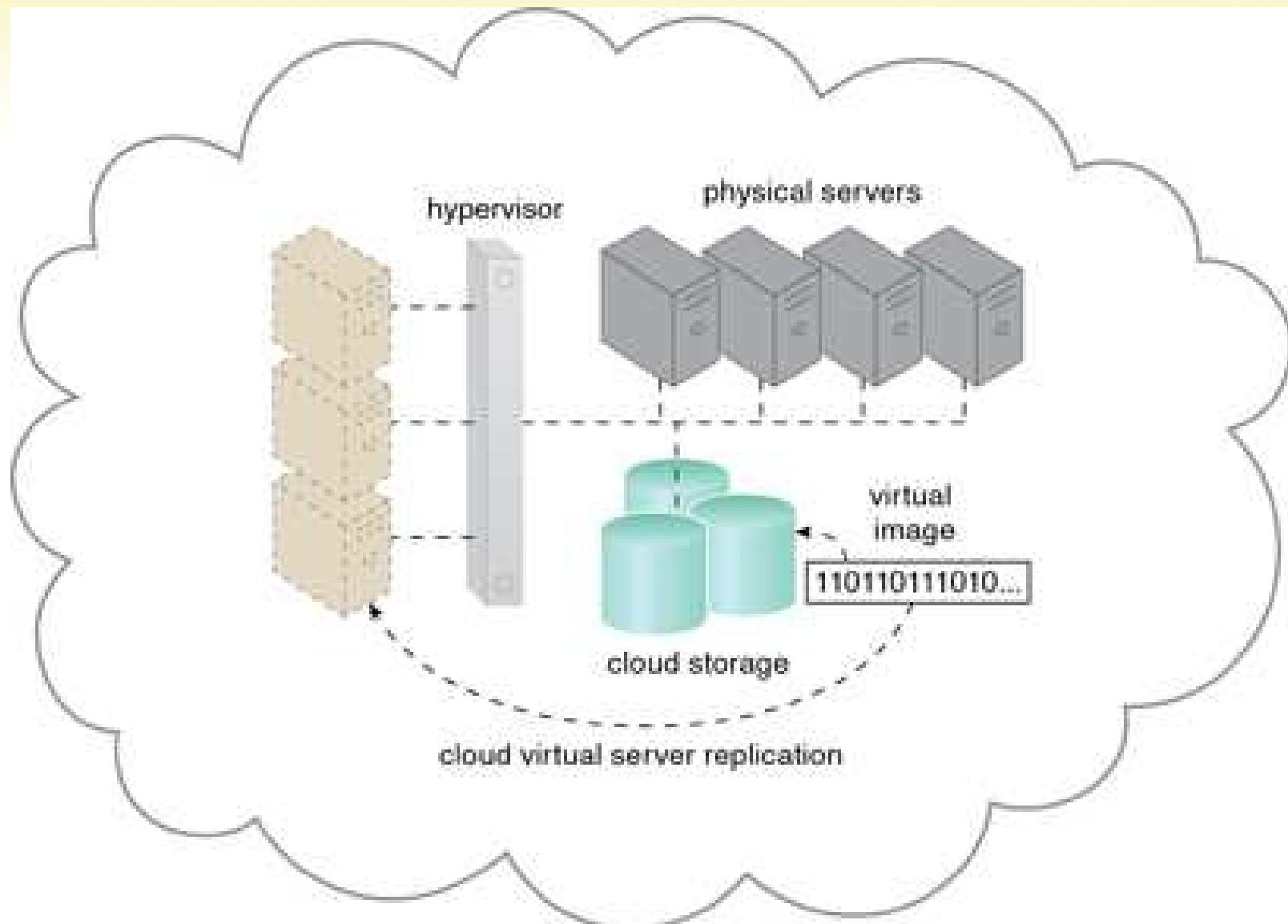
Zero Downtime



Load Balancer



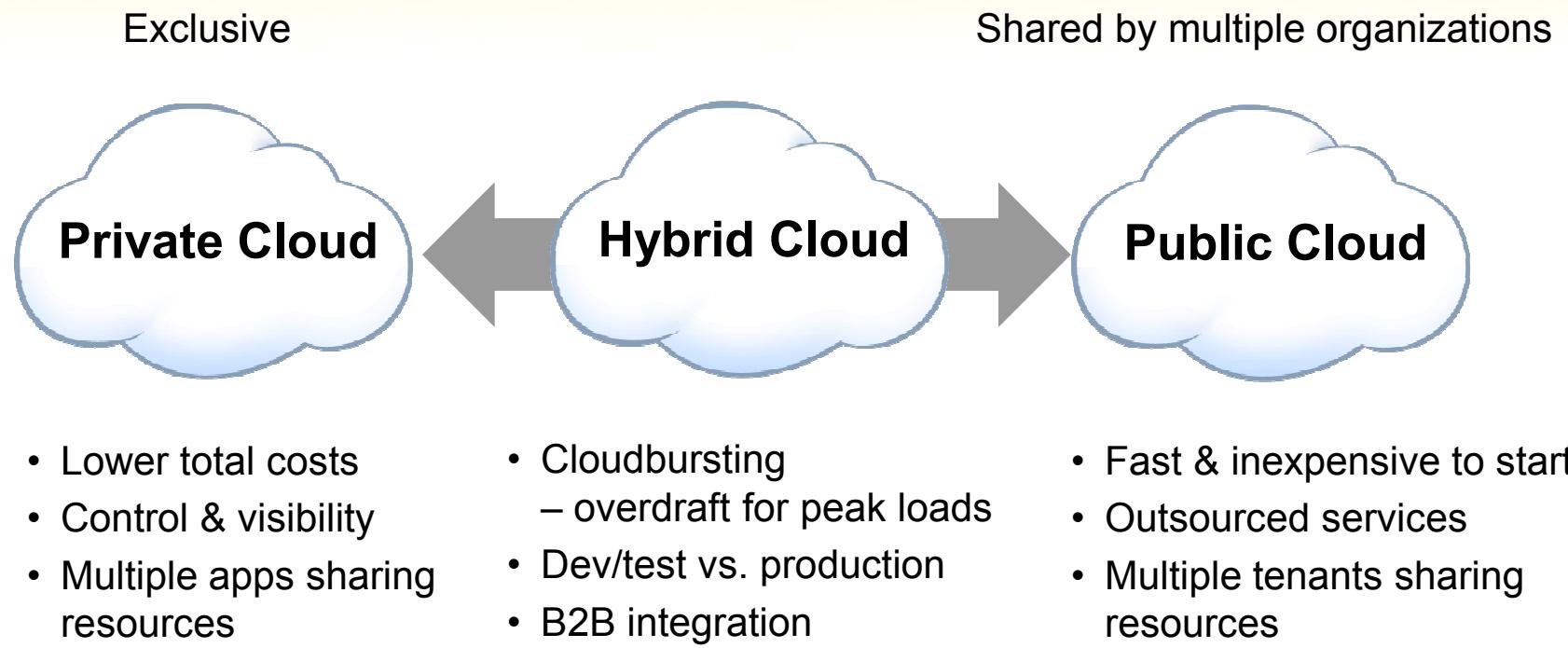
Resource Replication



Deployment Models

- Public clouds
 - Service provider owned and managed.
 - Access by subscription.
 - Standardized services
- Private clouds
 - Users owned and managed.
 - Access limited to client and its partner network.
 - Retaining greater customization and control.
- Hybrid clouds
 - Mix of private and public cloud

Choice of Deployment Models



Example: 12306.cn

Public Clouds

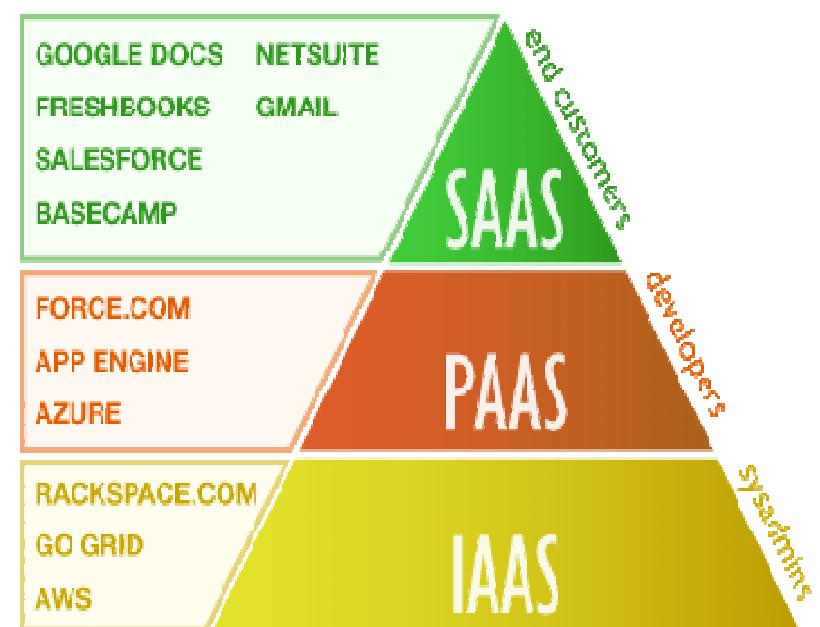
- Provide computing resources on demand
 - Publicly in Internet, for everyone
 - Paid on usage of resources
 - Could be IaaS, PaaS, SaaS or mix of them
- Examples of public clouds
 - **IaaS**: Aliyun, Qingyun, Amazon EC2, ...
 - **PaaS**: OpenShift online, IBM Bluemix, Heroku, Google App Engine, Amazon AWS, Windows Azure, Baidu BAE, JD JAE, Sina SAE...
 - **SaaS**: Google Apps, Microsoft Office 365, Salesforce.com, Adobe Creative Cloud ...

Cloud Service Types

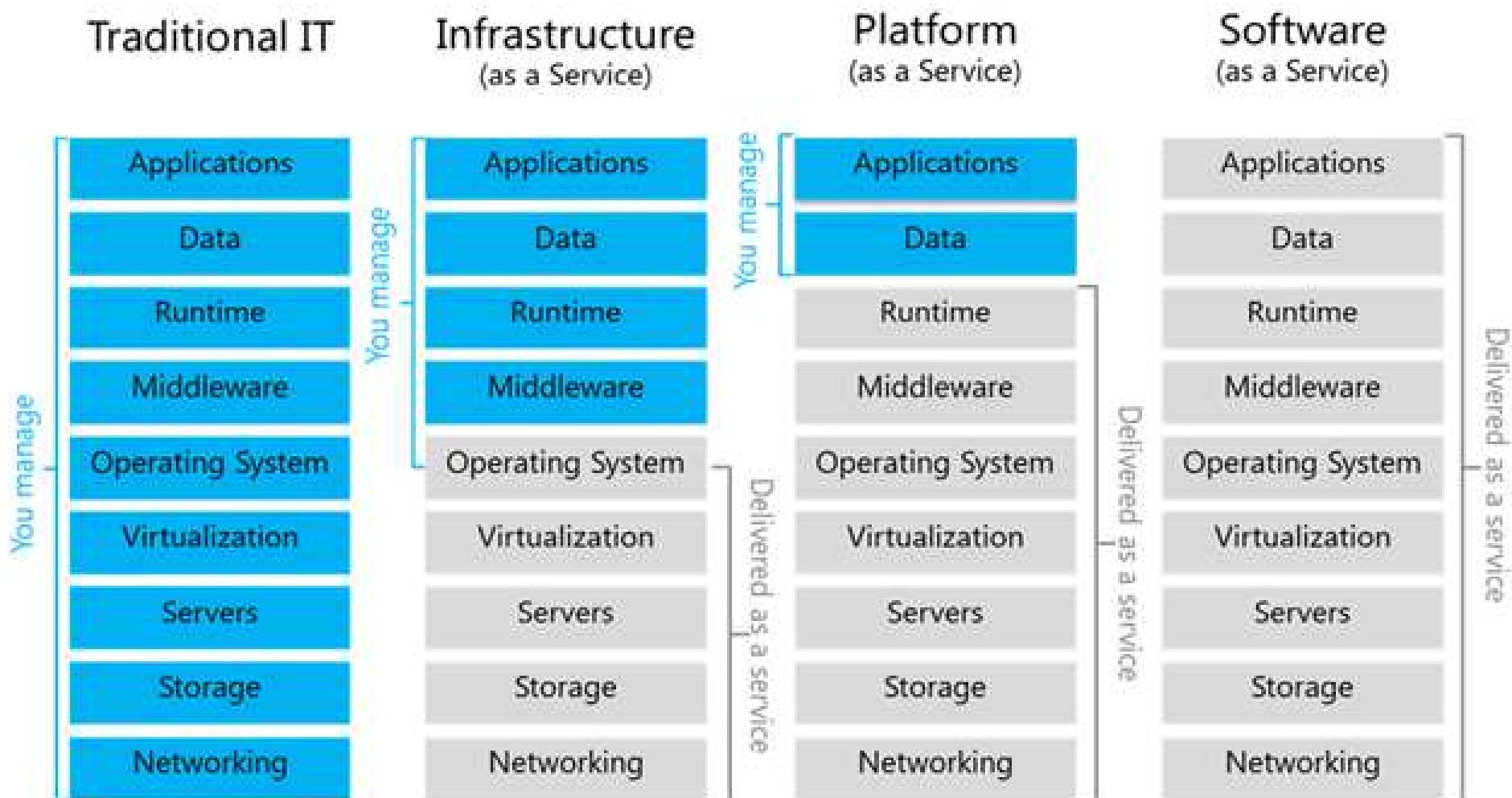
- Infrastructure as a Service (**IaaS**)
 - Virtual machines in the cloud on demand
 - Users install the OS and software they need
- Platform as a Service (**PaaS**)
 - Platform, services and APIs for developers
 - E.g. Java + JBoss + JSF + JPA + MySQL or JavaScript + Node.js + MongoDB + Express
- Software as a Service (**SaaS**)
 - Hosted application on demand (e.g. WordPress)

Types of Cloud Computing

- Software as a Service (SaaS)
 - Access to resources and applications
- Platform as a Service (PaaS)
 - Access to development and operational components
- Infrastructure as a Service (IaaS)
 - Completely outsources needed storage and resources



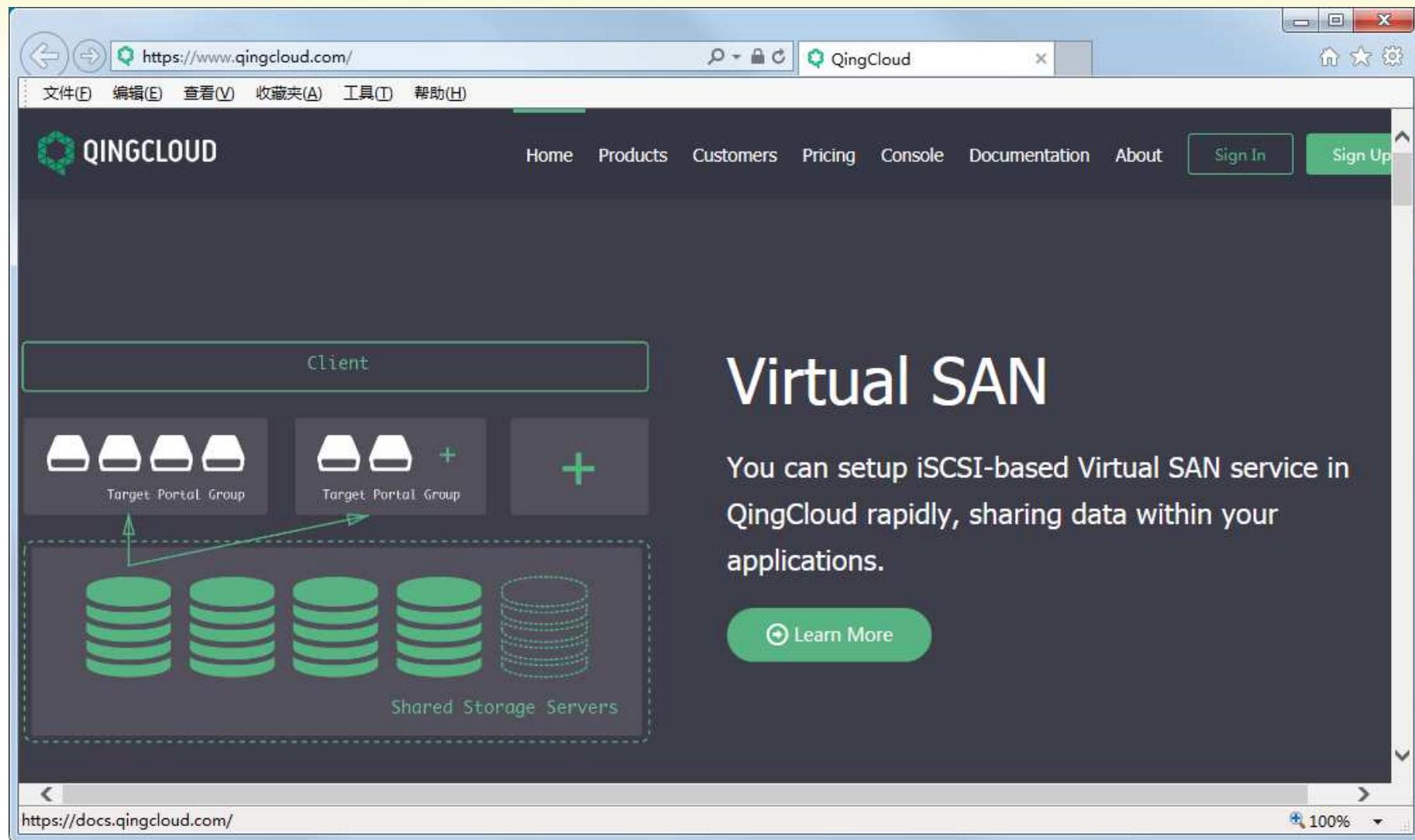
Levels of Cloud Service



IaaS (Infrastructure as a Service)

- IaaS ≈ rent virtual computers and other resources, such as networks and storeages.
- Most basic cloud service model
- Cloud users deploy their applications by then installing operating system images on the machines as well as their application software.
- You could modify your resources as you go
 - E.g. add more 100 GB HDD storage + 2 GB RAM
- IaaS pricing on the amount of resources allocated and consumed.

Example of IaaS: QingCloud



Console of QingCloud

The screenshot shows the QingCloud console interface. The top navigation bar includes links for Tickets, Consumptions, Help, and Account Lock, along with user information for 'lidan_gz'. The left sidebar menu is under the 'Guangdong1' region, with 'GD1' selected. The 'Compute & Networking' section is expanded, showing 'Instances' (selected), 'Images', 'VxNets', 'Routers', 'Load Balancers', 'Elastic IPs', and 'DNS Aliases'. Below this are 'Storage', 'Security', and 'Database & Cache' sections.

The main content area displays the 'GUANGDONG1 / INSTANCES' page. A descriptive text block explains that QingCloud provides an elastic computing capability accessible at any time. It lists expected hardware configuration, operating system, and network configuration, noting that any resource request can be accomplished in 10 to 60 seconds, enabling an on-demand computing resource.

Below the text are several control buttons: a refresh icon, a '+ New' button, a 'Start' button, a 'Stop' button, and a 'More Actions' dropdown. To the right of these buttons is a dropdown menu for 'Display per Page' set to 10. A table header follows, listing columns: ID, Name, Status, Image ID, Network, EIP, Type, Alarm Status, Last Snapshot Time, and Created At. The table body contains the message 'No Result'.

A tip at the bottom of the page states: ** Tip: You can click on each resource by "right clicking" for typical operations, and "double-click" to modify the basic attributes.*

PaaS (Platform as a Service)

- PaaS ≈ rent a complete development platform;
- Cloud vendors deliver a computing platform typically including operating system, programming language execution environment, database, and web server.
 - E.g. Linux + Python + Django + MongoDB + Nginx load balancer + web server
- Users develop and run their software on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers.

PaaS vs. IaaS

- IaaS better for migrating existing applications
 - More flexible, you install your environment
- PaaS has lower demands on administration
- PaaS will take care of scaling if applications use correct frameworks, also redundancy and CDN
- PaaS better for new applications
- BUT has dangers of vendor lock in if platform specific functions are used

More about PaaS

- Public solution stacks for web applications
 - OS, web server, language interpreters, provisions for automatic scaling, all shielded from the user
- Each system only has a few supported languages
 - Automatic deployment and scaling not trivial
- Offers development tools
 - Libraries for specific services
 - IDE plugins, deployment tools

Typical PaaS Architecture

Front-End: HTML5, CSS3 JavaScript / Mobile

Middle-Tier Languages and Frameworks:

PHP, Java, C#, Python, Ruby, JavaScript,
Symfony, Zend Framework, JSF, ADF, Django, Rails,
ASP.NET, ASP.NET MVC, **Node.js**

Computing Nodes:

Amazon EC2, Azure
Compute, App Engine

Back-End :

Relational DBs, NoSQL
(MongoDB), Blob Storage,
Message Queues, CDN,
Notifications

Operating Systems: Linux / Windows / other

Classical PaaS Stacks

- Java + JBoss app server + Java ServerFaces + JBoss Rich Faces + Java Persistence API + Oracle database
- Python + Django + MongoDB + Linux cron jobs + Nginx load balancer + Gunicorn web server
- .NET Framework + C# + ASP.NET + WCF + SQL Server + Nginx load balancer + IIS web server
- PHP + Zend Framework + Cassandra DB + Nginx load balancer + Apache web server
- JavaScript + Node.js + MongoDB + RabbitMQ
- Ruby + Ruby on Rails + MySQL + Sphinx + Memcache + Unicorn HTTP server

Find your Platform as a Service

- <http://www.paasify.it>

Find your Platform as a Service!

What's best on your PaaS? Define your needs and get a list of candidates that claim to be your best fit.

Find your PaaS

Comprehensive

More than 70 vendors

...and counting.

Comparable

Distinctive PaaS features

A set of distinctive and intersecting properties to enable comparison and matching of different PaaS offerings.

Current

Continuously updated

Data structures are [publicly available](#) and editable by the community. We also aim at vendors to [verify](#)  their profiles.

SaaS (Software as a Service)

- SaaS ≈ rent an application in cloud.
- Cloud vendors install and operate application software in the cloud.
- One application instance may be serving hundreds of customers.
- Customers access the software from cloud clients.
- A customer can configure the application through metadata

SaaS Concepts

- Simply renting an application instead of setting it up on your own server
- Examples:
 - Exchange hosting (\$10/user/month)
 - Wordpress hosting (\$20-\$150 / month)
 - Web hosting (\$90 / year)
 - Quickbooks (\$50 / month)
 - Salesforce (\$125/user/month)
 - World of Warcraft (\$20/month)
- These are all cloud apps (computing as a utility)

SaaS Example



- **Up to now:** on-demand customer relationship management (CRM), marketing, Web analytics as hosted service
- **Now:** platform to enable developers to create and provide arbitrary business applications on-demand



Why SaaS is Attractive

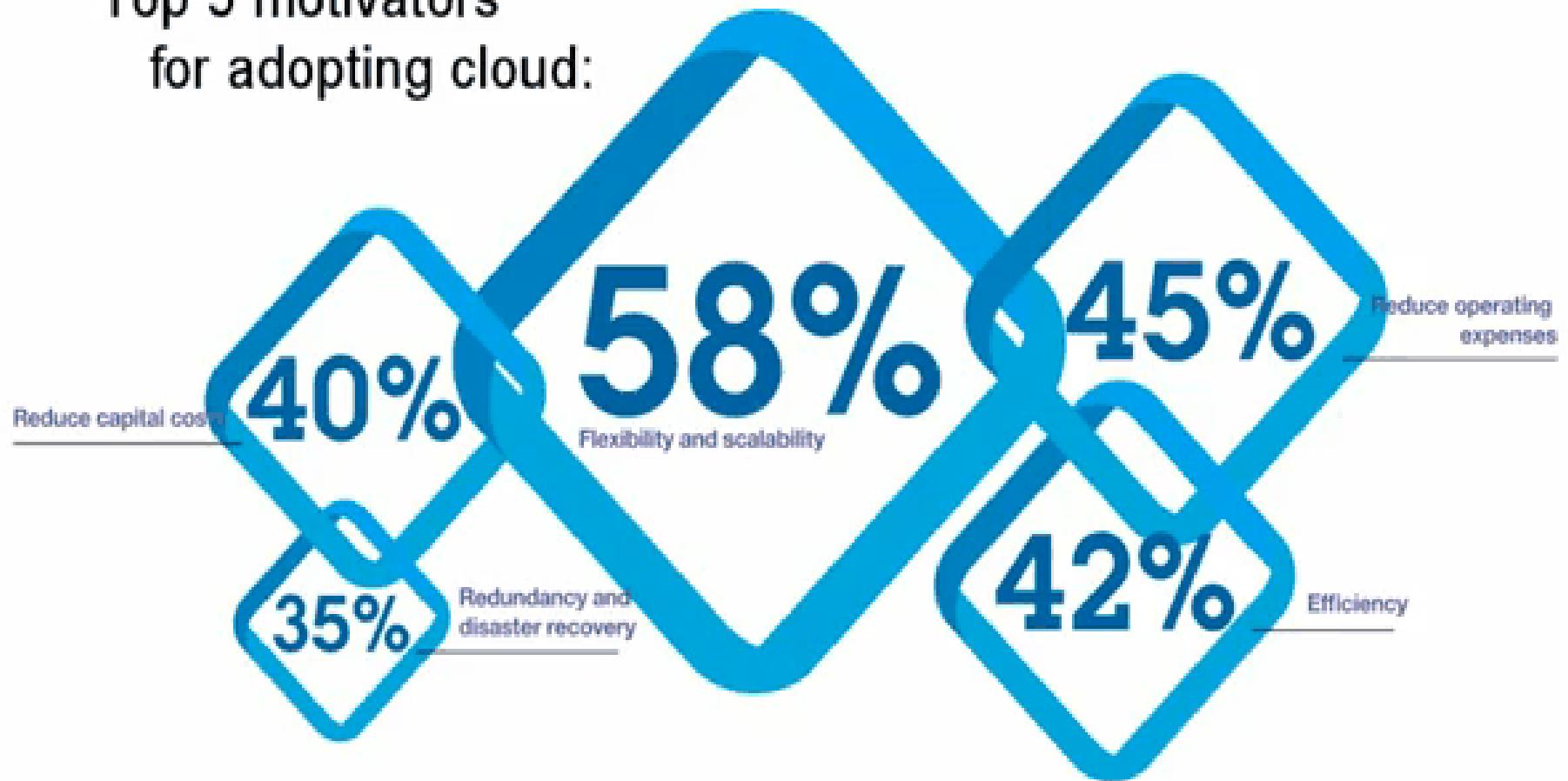
- No software installation on private PC, no upgrades, no patches, no service packs, no maintenance
- No license renewal when computer is exchanged
- All data resides on the Web
- “Pay-as-you-go” business model, i.e., payment based on usage

Periodic Table of SaaS Ecosystem 2015



Top Motivators for Adopting Cloud

Top 5 motivators
for adopting cloud:



Source: IBM Tech Trends Report 2011

@2015, Li Dan, Guizhou Academy of Sciences

Issues with Cloud Computing: Security and Privacy

- Does the cloud back up your data?
- Is your data always safe?
- Can you conduct business abroad?
- Who is given access to your data?
- Who else is on your server?

Data security concerns do not change significantly after adopting cloud computing



Only 25% of companies expressed more concern about data security after adopting cloud services

More
Concerned



47% of businesses in Singapore expressed more concern about data security following cloud adoption

More
Concerned



47% of Brazilian companies said they are less concerned after switching to the cloud

Less
concerned

Enabling the Cloud

- Economic Value
- Full connectivity
- Open Access
- Reliability
- Security
- Privacy
- Interoperability and User Choice
- Sustainability



*Envisioning the Cloud: The Next Computing Paradigm
Marketspace Point of View (March 20, 2009)*

Disadvantages of cloud computing

- Dependent on internet connections
- Users are subject to terms and conditions
- Data in hands of a 3rd party
- No worldwide accepted standards

IaaS Example: Aliyun, Ecosystem

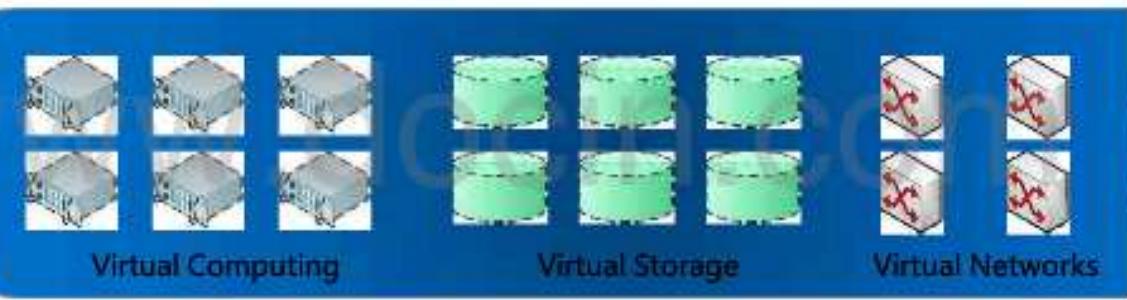
SaaS



PaaS



IaaS



Hardware



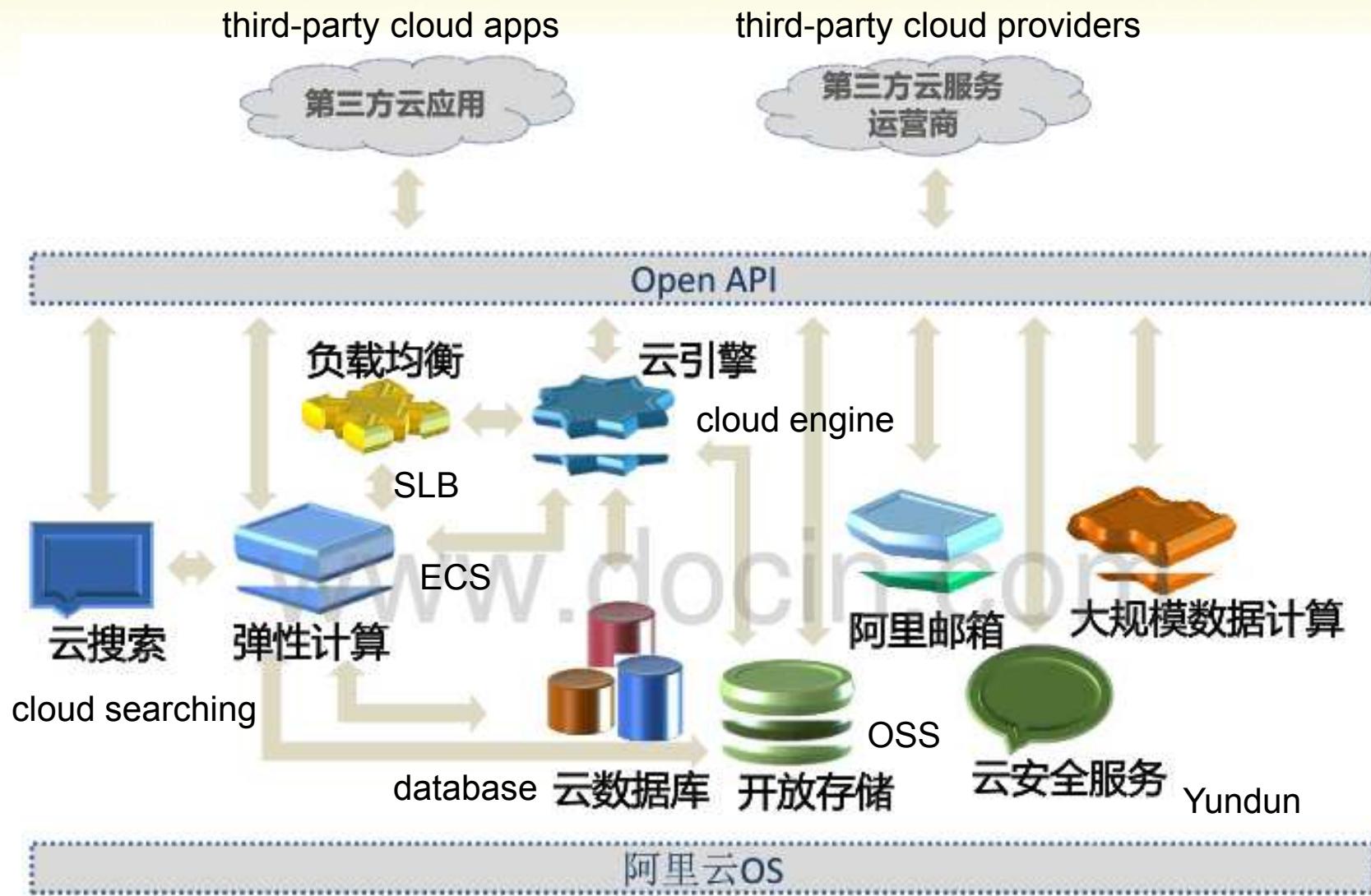
Data Centers of Aliyun



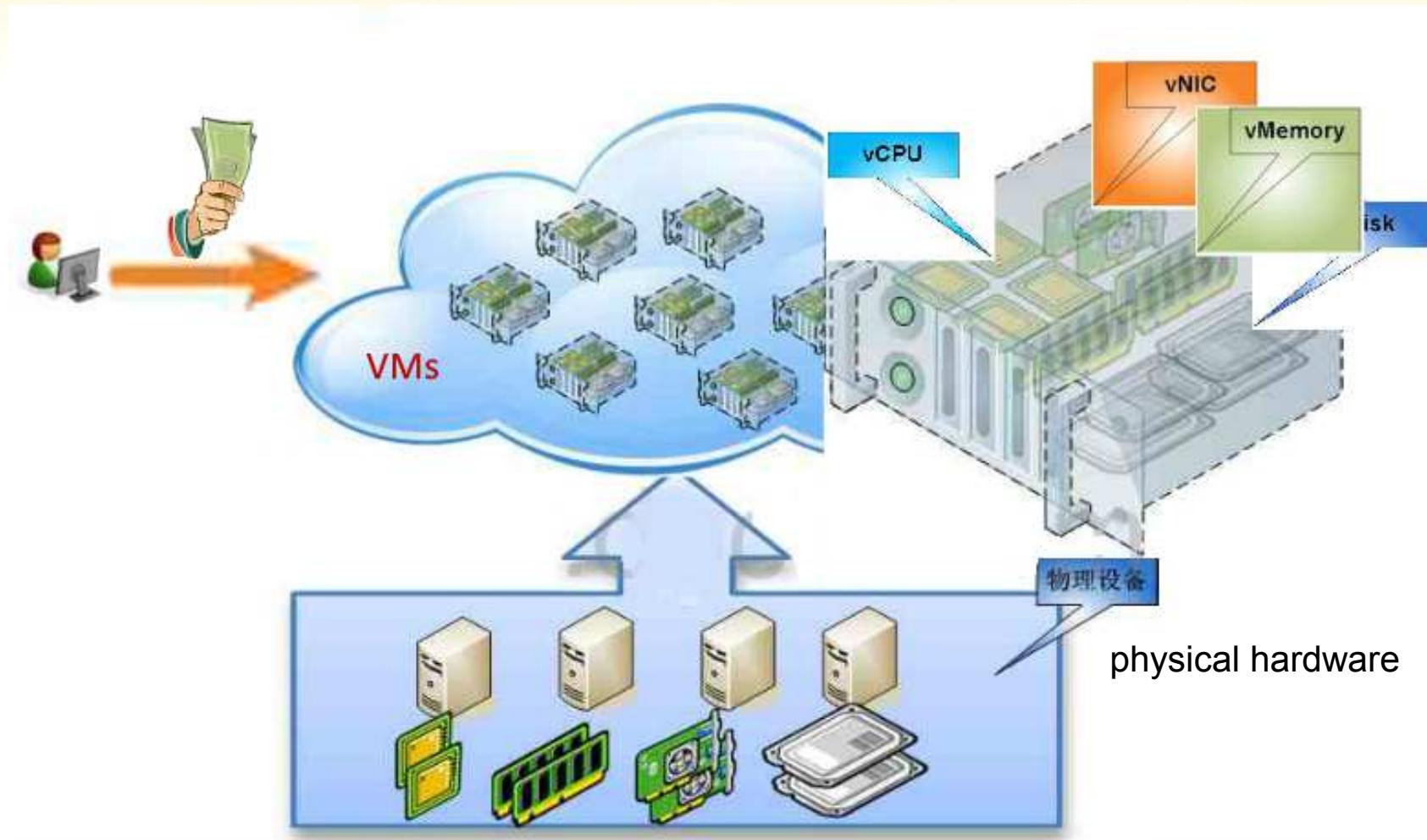
Main Services of Aliyun

- Elastic Computing:
 - **ECS**: Elastic Compute Service. Scalable capacity to build stable and secure applications.
 - **SLB**: Server Load Balancer. Distributes traffic for multiple cloud servers to improve the availability.
- Storage and CDN
 - **OSS**: Open Storage Service.
 - **CDN**: Content Delivery Network
- Database:
 - **RDS** (MySQL), OTS (Open Table Service (OTS), a NoSQL database)
 - **OCS**: Open Cache Service . An online caching service.
- Security and Management
 - **Yundun**: avoid heavy-traffic DDoS attacks
 - **CMS**: Cloud Monitor System. A platform to monitor sites and servers in real time.

Products of Aliyun



ECS: Elastic Compute Service



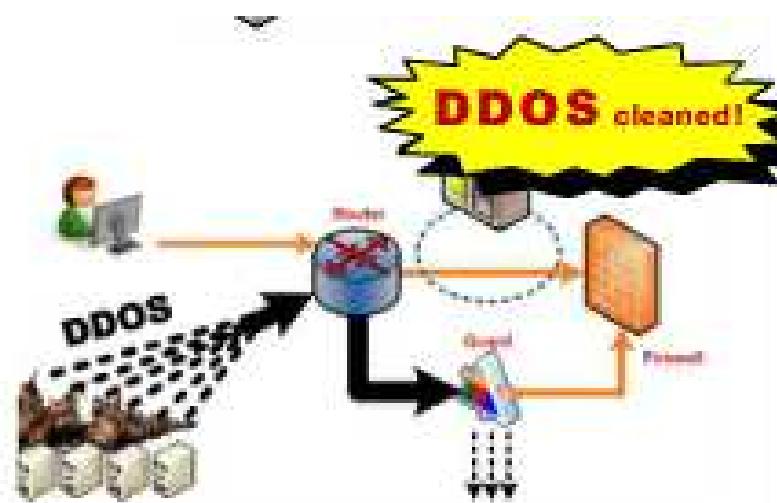
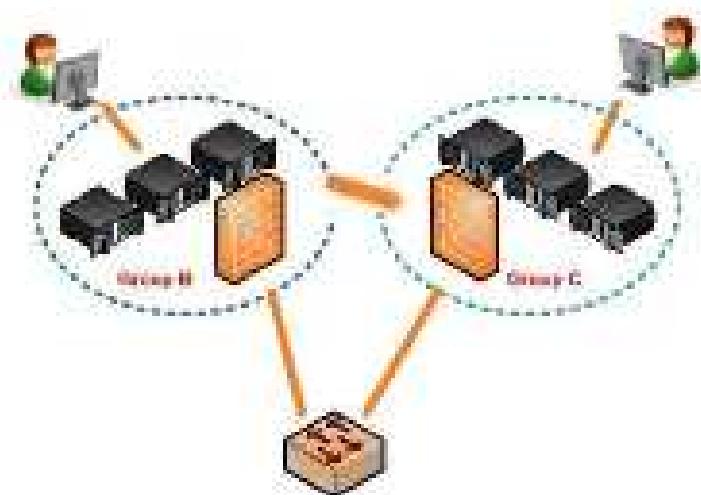
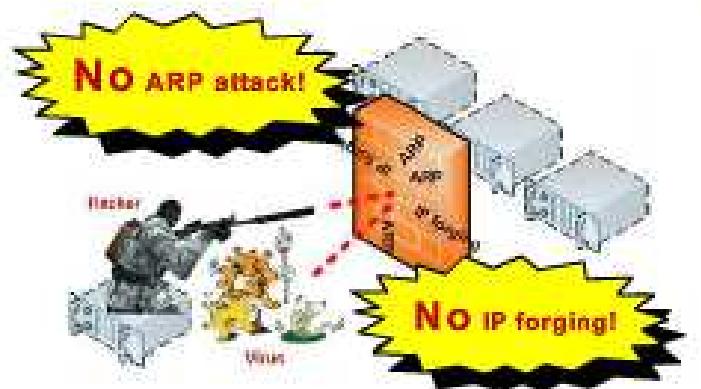
ECS: Elastic Compute Service

- Ability to increase or decrease capacity within minutes
- Ability to commission one, hundreds, or even thousands of server instances simultaneously
- A web service API to control the scaling of instances depending on your needs
- A “pay only for what you use” pricing model

OSS: Open Storage Service.

- Offers secure online storage system for any kind of data
- Automatically replicated 3 copies
- Data is universally accessible over the Web
- No limits on amount of data, longevity or bandwidth for transfer or publish
- Scalable, distributed system
 - Stores data redundantly across geographically separated data centers

Security of Aliyun



Prices of Aliyun

香港区域ECS包年包月定价				
CPU(核)	内存(GB)	香港年价Linux (元)	香港年价Windows (元)	
1	1	1,160	1,760	
1	2	1,550	2,360	
1	4	2,340	3,550	
2	2	2,310	3,520	
2	4	3,100	4,710	
2	8	4,680	7,110	
4	4	4,630	7,030	
4	8	6,200	9,420	
4	16	9,350	14,210	
8	8	9,250	14,060	
8	16	12,400	18,850	
8	32	18,700	28,420	
16	64	37,400	56,850	

计费项	除Windows外的操作系统			Windows操作系统		
	包年包月		按量付费	包年包月		按量付
	月价	年价	小时价	月价	年价	小时价
数据盘 (元/GB)	0.5	5	0.0009	0.7	7	0.001
按固定带宽计费(元/Mbps)	232	2320	0.58	232	2320	0.58
按流出流量付费 (元/GB)	n/a	n/a	1.3	n/a	n/a	1.3

PaaS Example: OPENSHIFT

- <https://www.openshift.com/>
- Red Hat's open source Platform-as-a-Service (PaaS)
- Allow developers to quickly develop, host, and scale applications in a cloud
- Offer online, on premise, and open source project options.

Three Versions of OpenShift

	OpenShift Online	OpenShift Enterprise	OpenShift Origin
What is it?	Hosted PaaS Service	Private PaaS Product	Open Source PaaS Project
How can it help me?	Quickly develop, host, and scale applications in the public cloud.	Accelerate IT service delivery and streamline application development.	Use a free, open source PaaS or help extend OpenShift.
How is it priced?	Free or Premium Plans	Annual software subscription	Free and open source
Who provides support?	Community Support for Free & Bronze Plans or Red Hat Support for Silver Plan	Red Hat	Community
Where does it run?	In the public cloud	On your servers or in your private cloud	Your laptop, your servers, private cloud, or public cloud
Who is it good for?	Startups, developers, small businesses, and even enterprises	Enterprises that want to run their own cloud	Anyone that wants to tinker on the latest thing in open source software
How do I get it?	Sign Up Online	Download a free evaluation	Download from GitHub

OpenShift Online

- Red Hat's public cloud application development and hosting platform
- Automate the provisioning, management and scaling of applications
- Supporting 6 languages: [Java](#), [Ruby](#), [PHP](#), [Node.js](#), [Python](#) and [Perl](#);
- 3 Middleware: [Jboss](#), [Tomcat](#), [Zend server](#)
- 6 Frameworks: [Django](#) ,[Drupal](#) ,[Flask](#),[Rails](#) ,[Switchyard](#) ,[Vert.x](#)
- 7 Native Services: [Jenkins](#), [Mongodb](#), [Mysql](#), [Openshift Metrics](#), [Pgrouting](#), [Postgis](#), [Postgresql](#)

Services and Features

- **World-class Support**
 - Supported by Red Hat's award-winning technical support.
- **Auto Scaling**
 - Automates the scaling of your application as your user traffic increases. Pay-as-you-go access to more and faster servers.
- **Custom SSL**
 - Secure traffic to your custom domains with SSL and your own certificates.
- **Extra Storage**
 - Access to more fast local storage for your applications.

Terminology of OpenShift

- **Application:** OpenShift is focused on hosting web applications.
- **Gear:** a gear is a service container running the application. Gear size: small (512M RAM, 1G disk), medium(1G RAM, 1G disk) and large(2G RAM, 1G disk).
- **Cartridge:** plug-ins a gear to run an application, such as [Tomcat](#), [Node.js](#), [MySQL](#).
- **Scaling:** If you allow your application to scale, a load balancer allocates more gears to handle traffic as your application needs it.

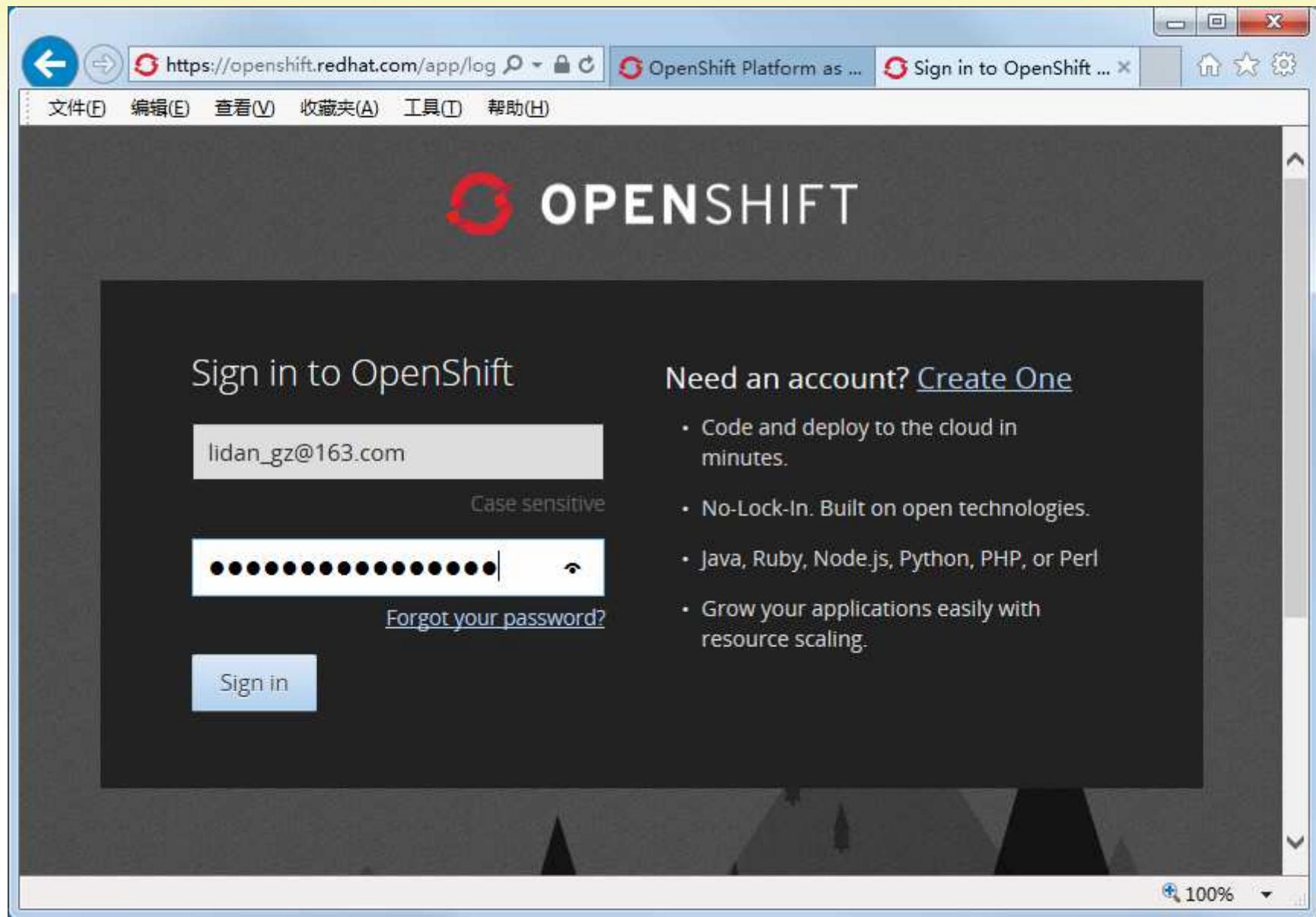
Monthly Pricing

	USD	CAD	EUR	Free Plan	Bronze Plan	Silver Plan
\$ BASE PRICE	Free	Free	Free	Free	Free	\$20/month
⌚ APPLICATION IDLING	24 hours	24 hours	24 hours	Never	Never	Never
⚙️ INCLUDED GEARS	3 small gears	3 small gears	3 small gears	3 small gears	3 small gears	3 small gears
⚙️ MAX GEARS	3	3	3	16	16	16+
↗️ SCALING	Yes (3 min / 3 max)	Yes (3 min / 3 max)	Yes (3 min / 3 max)	Yes (3 min / 16 max)	Yes (3 min / 16 max)	Yes (3 min / 16 max)
⚙️ GEAR SIZES	small	small	small	small (\$0.02/hour) small.highcpu (\$0.025/hour) medium (\$0.05/hour) large (\$0.10/hour)	small (\$0.02/hour) small.highcpu (\$0.025/hour) medium (\$0.05/hour) large (\$0.10/hour)	small (\$0.02/hour) small.highcpu (\$0.025/hour) medium (\$0.05/hour) large (\$0.10/hour)
💾 STORAGE	1GB per gear	1GB per gear	1GB per gear	1GB per gear; \$1.00/month per additional GB	1GB per gear; \$1.00/month per additional GB	6GB per gear; \$1.00/month per additional GB
🔒 SSL	Shared	Shared	Shared	For custom domains	For custom domains	For custom domains
👥 TEAMS	Not included	Not included	Not included	Up to 15	Up to 15	Up to 15
💻 JBOSS EAP 6	Included	Included	Included	3 gears free; \$0.03/hr per additional gear	3 gears free; \$0.03/hr per additional gear	3 gears free; \$0.03/hr per additional gear

Steps to Use OpenShift Online

1. Create an "Application" in OpenShift Online (with the web console, command-line tools, or your IDE)
2. Code the application in your favorite environment, or use one of the available Quickstarts.
3. Push the application code to OpenShift Online (using [Git](#))

Log in to OpenShift web console



Create an Application

1. Choose a web framework or codebase to start from
 - Try JBoss, PHP, Python, Ruby, Node.js
 - or create a new Drupal or Wordpress site instantly.
2. Add **cartridges** like MySQL or MongoDB to your application
 - including databases, cache servers, management tools, and continuous integration servers.
3. Upload your code to OpenShift via Git
 - Your source code is stored with your application in a Git version control repository.

Choose a Type of Application

OPENSHIFT ONLINE

Applications Settings Help OpenShift Hub

1 Choose a type of application 2 Configure the application 3 Next steps

Choose a web programming cartridge or kick the tires with a quickstart. After you create the application you can **add cartridges to enable additional capabilities** like databases, metrics, and continuous build support with Jenkins.

Search by keyword or tag or

Browse by tag... ▾

Instant App see all

-  Jenkins Server CI Java
-  Drupal 7 CMS DRUPAL NOT SCALABLE PHP +
-  Ghost 0.5.10 BLOG CMS GHOST NODEJS +
-  OpenShift Backup Server BACKUP +
-  WordPress 4 BLOG CMS NOT SCALABLE PHP +

xPaaS see all

-  JBoss Data Virtualization 6 JAVA EE 6 +
-  JBoss Enterprise Application Platform 6 JAVA EE 6 \$ +
-  JBoss Unified Push Server 1.0.0.Beta1 JEE FULL PROFILE +
-  JBoss Unified Push Server 1.0.0.Beta2 JEE FULL PROFILE +

Cartridge – A managed runtime for your application.
QuickStart – A quick way to try out a new technology with code and libraries preconfigured. You are responsible for updating core libraries for security updates.
Receives automatic security updates

Configure the Application

1

Choose a type of application

2

Configure the application

3

Next steps

Based On

WordPress 4 Quickstart 

An open source, semantic, blogging and content management platform written in PHP with a MySQL backend focusing on aesthetics, web standards, and usability.

[Learn more](#) OpenShift maintained

Does not receive automatic security updates

Public URL

OpenShift will automatically register this domain name for your application. You can add your own domain name later.

Source Code

Your application will start with an exact copy of the code and configuration provided in this Git repository. OpenShift may expect certain files to exist in certain directories, which may require you to update your repository after creation.

Gears

small

Gears are the application containers running your code. For most applications, the small gear size provides plenty of resources. You can also upgrade your plan to get access to more gear sizes.

Cartridges

PHP 5.4 and MySQL 5.5

Applications are composed of cartridges - each of which exposes a service or capability to your code. All applications must have a web cartridge.

Scaling



This application may require additional work to scale. Please see the application's documentation for more information.

OpenShift automatically routes web requests to your web gear. If you allow your application to scale, we'll set up a load balancer and allocate more gears to handle traffic as you need it.

List of the Applications

The screenshot shows the OpenShift Online web interface. At the top, the URL is https://openshift.redhat.com/app/con and the title bar says "OpenShift Platform as a... Applications | OpenS...". The menu bar includes "文件(F)", "编辑(E)", "查看(V)", "收藏夹(A)", "工具(I)", and "帮助(H)". The header features the OpenShift logo and the text "OPENSHIFT ONLINE". On the right, there are links for "Upgrade Plan" and "lidan_gz@163.com". The main content area has tabs for "Applications", "Settings", and "Help". The "Applications" tab is selected, showing "1 of 3" applications. One application, "gyblog", is listed with "MySQL 5.5, PHP 5.4". A button labeled "Add Application..." is visible. To the right, there's a sidebar with "You may want to..." links: "Add a collaborator", "Use your own domain name", and "Create a scalable application". Below that, under "From the command line", there are links for "The rhc client lets you:", "Access logs", "Save and restore backups", and "Connect directly to internal services". The bottom right corner shows a zoom level of "100%".

Overview of WordPress App

The screenshot shows the OpenShift Online application console interface. At the top, the URL is https://openshift.redhat.com/app/console, and the title bar displays "OpenShift Platform as a S..." and "gyblog | OpenShift Online". The menu bar includes "文件(F)", "编辑(E)", "查看(V)", "收藏夹(A)", "工具(I)", and "帮助(H)". The top right corner shows "Upgrade Plan" and the user "lidan_gz@163.com".

The main content area displays the "OPENSHIFT ONLINE" logo and navigation tabs for "Applications", "Settings", "Help", and "OpenShift Hub".

A card for the application "gyblog-lidangz.rhcloud.com" is shown, indicating it was "Started" 31 minutes ago in the "lidangz" domain and "aws-us-east-1" region. The card includes a "change" link and a gear icon.

The "Cartridges" section lists two components:

- PHP 5.4: Status Started, Gears 1 small, Storage 1 GB
- MySQL 5.5: Database gyblog, User adminLju91uW, Password show

The "Source Code" section provides a SSH URL: ssh://555c98b8e0b8cd235f000031. It also includes instructions: "Pass this URL to 'git clone' to copy the repository locally."

The "Remote Access" section contains a link: "Want to log in to your application?"

At the bottom, there are links for "Continuous Integration" (Enable Jenkins), "Tools and Support" (Add phpMyAdmin 4.0), and a "Delete this application..." button.

A note at the bottom encourages users to "Browse the Marketplace, or see the list of cartridges you can add".

The bottom right corner shows a zoom level of "100%".

WordPress Blog

The screenshot shows a Windows desktop environment. A browser window is open, displaying a WordPress blog. The title of the blog is "2015 Guiyang Training Course". The main content area features a large heading "Hello world!" followed by the text: "Welcome to WordPress. This is your first post. Edit or delete it, then start blogging!". Below this, there is a timestamp "20th May 2015" and a link to "1 Comment". On the left side of the browser window, there is a sidebar with several sections: "RECENT POSTS" (listing "Hello world!"), "RECENT COMMENTS" (listing "Mr WordPress on Hello world!"), "ARCHIVES" (listing "May 2015"), "CATEGORIES" (listing "Uncategorised"), and "META" (listing "Log in", "Entries RSS", "Comments RSS", and "WordPress.org"). The browser's address bar shows the URL "https://gyblog-lidangz.rhcloud.com/". The taskbar at the bottom of the screen shows icons for the browser, file explorer, and other system applications. The system tray indicates a battery level of 100%.

2015 Guiyang Training Course

Just another WordPress site

Search ...

RECENT POSTS

Hello world!

RECENT COMMENTS

Mr WordPress on Hello world!

ARCHIVES

May 2015

CATEGORIES

Uncategorised

META

Log in

Entries RSS

Comments RSS

WordPress.org

https://gyblog-lidangz.rhcloud.com/

2015 Guiyang Training Cour... 2015 Guiyang Training C...

100%

Lab: Start with OpenShift Online

- Sign up a free account in
<https://www.openshift.com/>
- Login in to OpenShift web console, and create a WordPress application.
- Test your WordPress application.

XaaS : Anything as a Service

2010	2011	2012	2009
Classroom-as-a-Service ¹⁾	Business-Integration-as-a-Service ¹⁾	Bespoke applications-as-a-Service ¹⁾	Data-as-a-service ¹⁾
Cluster-as-a-Service ¹⁾	Trust-as-a-service ¹⁾	Triage-as-a-service ¹⁾	Desktop-as-a-service ¹⁾
Infrastructure-as-a-service ¹⁾	Platform-as-a-service ¹⁾	Obsolescence-as-a-service ¹⁾	Format-as-a-Service ¹⁾
Workstation-as-a-service ¹⁾	Telepresence-as-a-service ¹⁾	Zebra-as-a-service ¹⁾	Flexibility-as-a-Service ¹⁾
Maximo-as-a-service ¹⁾	API-as-a-Service ¹⁾	Backup-as-a-Service ¹⁾	Governance-as-a-Service ¹⁾
Code-as-a-Service ¹⁾	Crypto(graphy)-as-a-Service ¹⁾	Unicorns-as-a-service ¹⁾	Leadership-as-a-service ¹⁾
Environment-as-a-Service ¹⁾	Kiss-as-a-Service ¹⁾	Metal-as-a-service ¹⁾	Transparency-as-a-service ¹⁾
Communication-as-a-Service ¹⁾	Connectivity-as-a-Service ¹⁾	Timing-as-a-service ¹⁾	Gaming-as-a-Service ¹⁾
Business-Analysis-as-a-Service ¹⁾	Optical-as-a-service ¹⁾	Case(Management)-as-a-Service ¹⁾	Engineering-as-a-Service ¹⁾
Cloud-as-a-Service ¹⁾	Contact Center-as-a-Service ¹⁾	Femtocell-as-a-Service ¹⁾	Workflow-as-a-service ¹⁾
Unified Communication-as-a-Service ¹⁾	FTTG-as-a-Service ¹⁾	Forensics-as-a-Service ¹⁾	WAN Optimization-as-a-service ¹⁾
Security-as-a-service ¹⁾	Zonal-as-a-service ¹⁾	Kernel-as-a-Service ¹⁾	Transcoding-as-a-service ¹⁾
Testing-as-a-service ¹⁾	Analytics-as-a-Service ¹⁾	Talent-as-a-service ¹⁾	Authentication-as-a-Service ¹⁾
Colocation-as-a-Service ¹⁾	Backend-as-a-Service ¹⁾	Terminology-as-a-service ¹⁾	Fraud-as-a-Service ¹⁾
Grid-as-a-Service ¹⁾	You-as-a-service ¹⁾	Confidentiality-as-a-Service ¹⁾	Business-as-a-Service ¹⁾
Voice-as-a-service ¹⁾	Tokenization-as-a-service ¹⁾	Zoomit-as-a-service ¹⁾	Orchestration-as-a-service ¹⁾
Everything-as-a-service ¹⁾	Emulation-as-a-Service ¹⁾	Foglight-as-a-Service ¹⁾	Fax-as-a-Service ¹⁾
Hudson-as-a-service ¹⁾	Billing-as-a-Service ¹⁾	Deception-as-a-service ¹⁾	Composite-as-a-Service ¹⁾
Content-as-a-Service ¹⁾	GPU-as-a-Service ¹⁾	Core-as-a-Service ¹⁾	Commerce-as-a-Service ¹⁾
Labor-as-a-service ¹⁾	Hadoop-as-a-service ¹⁾	Continuity-as-a-Service ¹⁾	Forms-as-a-Service ¹⁾
Computing-as-a-Service ¹⁾	Zimbra-as-a-service ¹⁾	Web-as-a-service ¹⁾	Freedom-as-a-service ¹⁾
Utopia-as-a-service ¹⁾	Access-as-a-Service ¹⁾	Running-as-a-service ¹⁾	Video-as-a-service ¹⁾
Yahoo-as-a-service ¹⁾	Kindness-as-a-service ¹⁾	Filtering-as-a-Service ¹⁾	Kitchen-as-a-Service ¹⁾
Community-as-a-Service ¹⁾		Fabrication-as-a-Service ¹⁾	Collaboration-as-a-Service ¹⁾
Landscape-as-a-service ¹⁾		Tool-as-a-service ¹⁾	Framework-as-a-Service ¹⁾
Preservation-as-a-service ¹⁾			zhou Academy of Sciences

Cloud is Everywhere



Clouds

Cloud Services

Cloud Consumers

Cloud Architectures & Applications

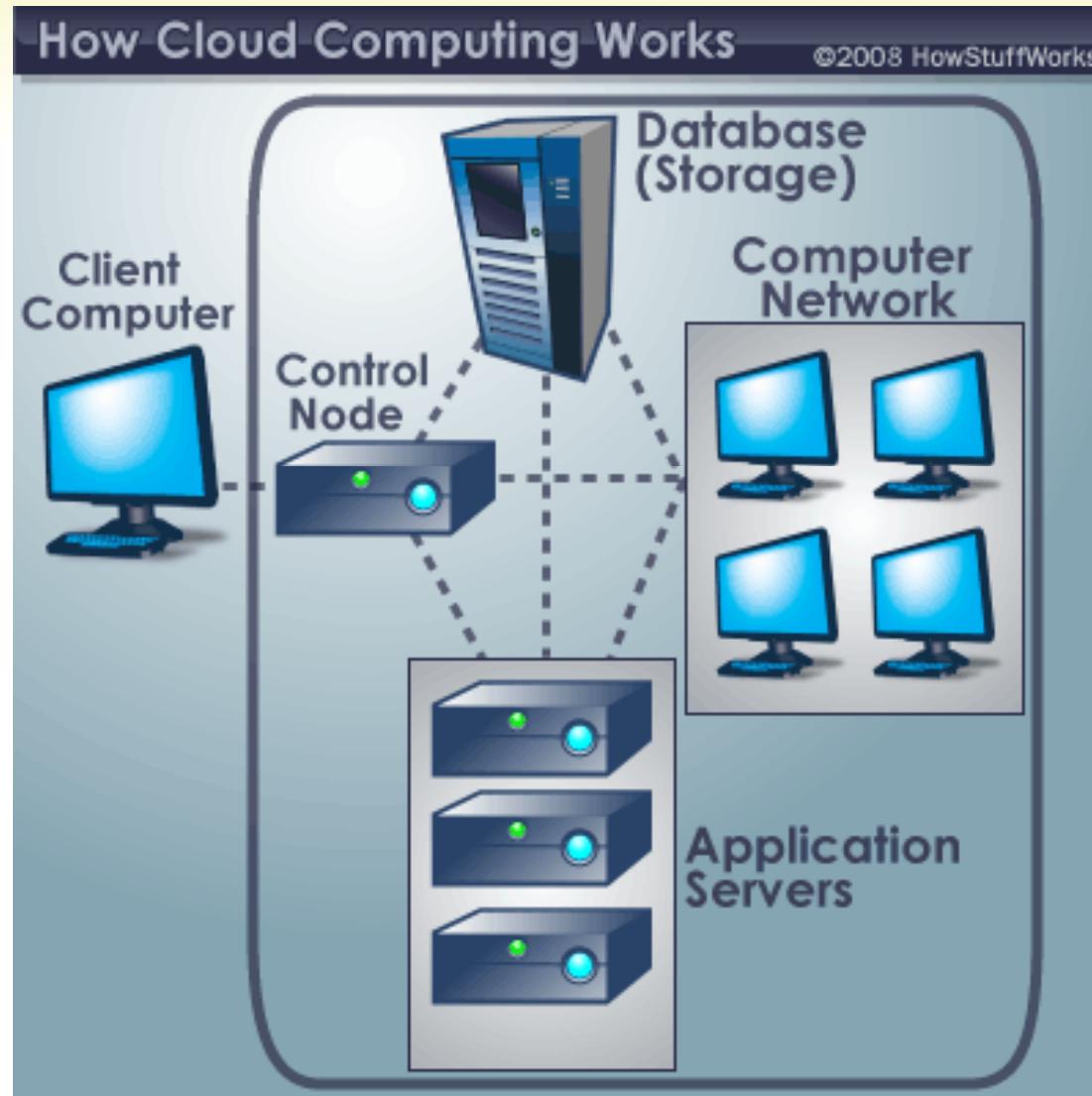
Technologies of Cloud Computing

Table of Contents

- Web Application & HTTP
- Client-Side Techniques
- Server-Side Techniques
- Architecture Techniques

WEB APPLICATION & HTTP

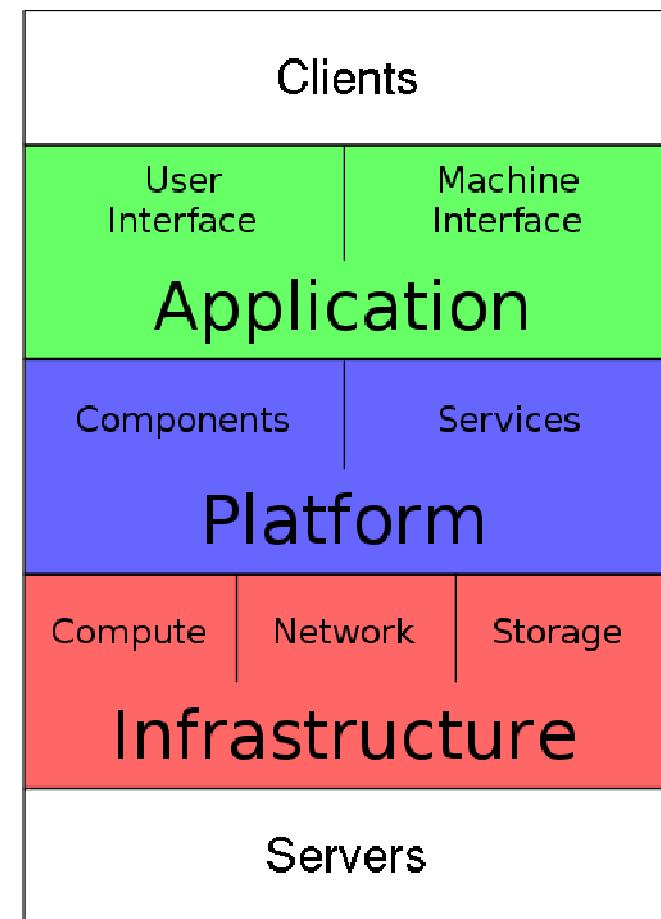
Cloud Computing Architecture



Cloud Computing

Applications: Desktop → Web → Cloud

- SaaS – Software as a Service
- PaaS – Platform as a Service
- IaaS – Infrastructure as a Service



What is a Web Application?

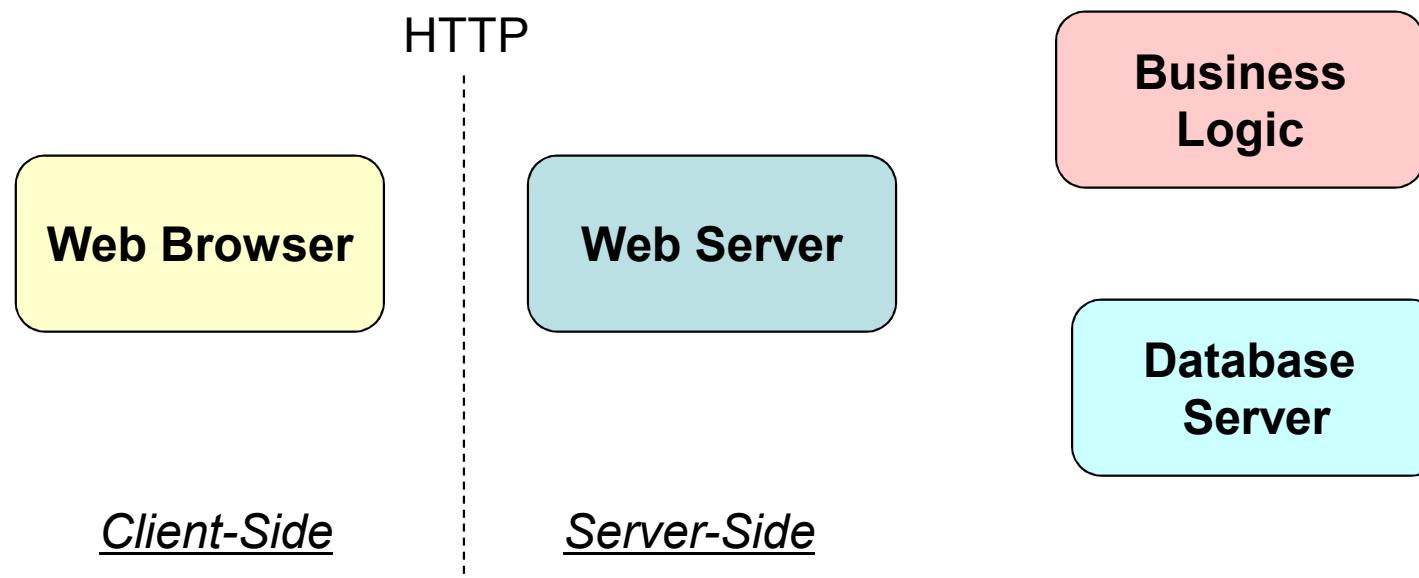
- “*A web application is a software package that can be accessed through the web browser. The software and database reside on a central server rather than being installed on the desktop system and is accessed over a network.*” (NetSity corporate homepage)
- A distributed application that accomplishes a certain business need based on the technologies of WWW and that consists of a set of web-specific resources

Web Applications

- Complex distributed, client/server applications
- High interactivity, high accessibility (Cloud)
- Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role
- Client side run inside a web browser
- Using HTTP for communication
- Rapid development, requires more planning, design, and control than “conventional” projects.

Web Application Components

- Four important components of a web application:



Web Application Components (2)

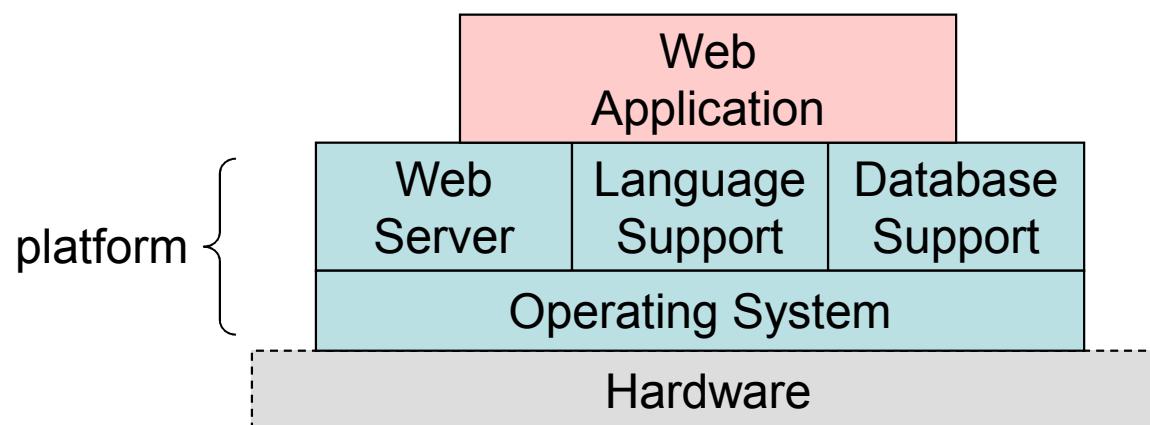
- **Web Browser**: presents the user interface
- **Web Server**: processes HTTP requests
- **Business Logic**: processes requests at the application level by providing a service
- **Database Server**: maintains the database by processing query and update requests from the application

What are the Sides?

- Client side (Front-end):
 - runs after page is displayed
 - page/content generation
 - user interaction
 - send data back to server
 - depends on browser/DOM
- Server side (Back-end):
 - runs before page is displayed
 - page/content generation
 - handle returning data
 - concerned with web server/database

Web App Platform

- A webapp **platform** is the host environment for application development and operation
- The platform includes
 - operating system, web server, language support, database support

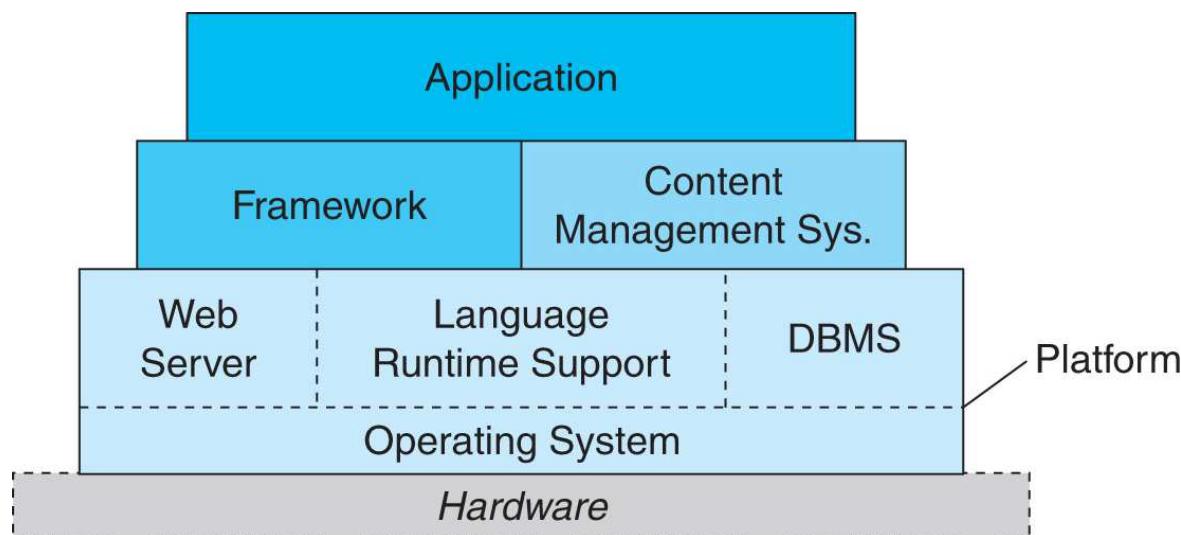


Web App Framework

- A web app **framework** is a set of tools that support web app development with:
 - A standard design model (e.g., MVC)
 - User interface toolkit
 - Reusable components for common functions (authentication, e-commerce, etc.)
 - Database support
 - Support for distributed system integration

Web App Framework (2)

- Frameworks give application developers more powerful building blocks to work with



HISTORY OF WEB FRAMEWORK

The Good Old Days

- Simple, static web pages (with animated gifs).

The screenshot shows the Apple website homepage from 1997. At the top, there's a navigation bar with links for 'Find It', 'Product Information', 'Customer Support', 'Technology & Research', 'Developer World', 'Groups & Interests', 'Resources Online', and 'About Apple'. Below this is a sidebar titled 'Apple Sites Worldwide' with links for Switzerland, Taiwan, Turkey, UK & Ireland, and United States, along with a 'Go' button. The main content area features the Apple logo and the text 'Welcome to Apple 1997'. A large banner in the center says 'Introducing CyberDrive' with a small BMW logo next to it. Below the banner, the text reads 'Register today for a free CD-ROM.' To the right, there's a section titled 'IMATE 300' featuring a picture of a handheld device. Further down, there's a 'MOVIES FROM MARS' section with a thumbnail for 'Interplay VR Takes You Out of This World'.

Find It

Product Information

Customer Support

Technology & Research

Developer World

Groups & Interests

Resources Online

About Apple

Apple Sites Worldwide

Switzerland

Taiwan

Turkey

UK & Ireland

United States

Go

Welcome to Apple 1997

Introducing CyberDrive

BMW

Register today for a free CD-ROM.

IMATE 300

Mobile, Affordable, & Smart

MOVIES FROM MARS

Interplay VR Takes You Out of This World

What's Hot

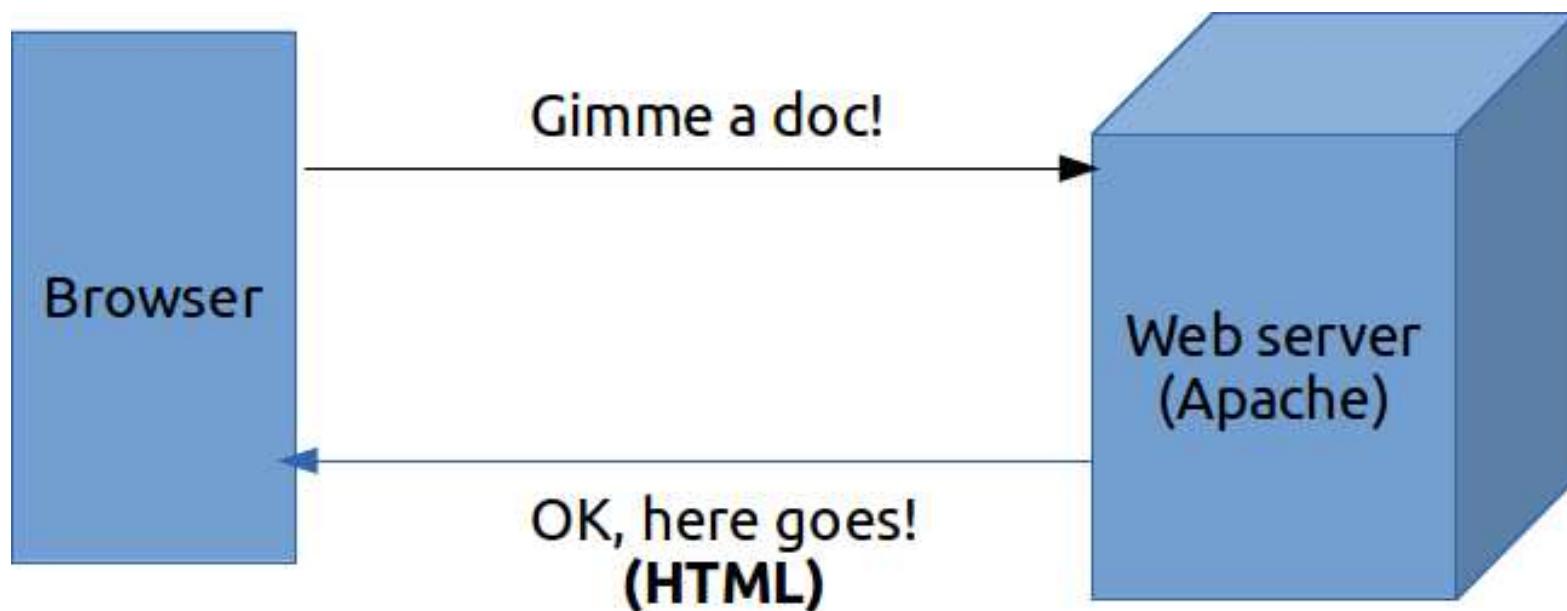
Preorder Mac OS 8

Now you can [preorder Mac OS 8](#), described by Macworld as "the most comprehensive update to the Mac OS in years, sporting a bold new

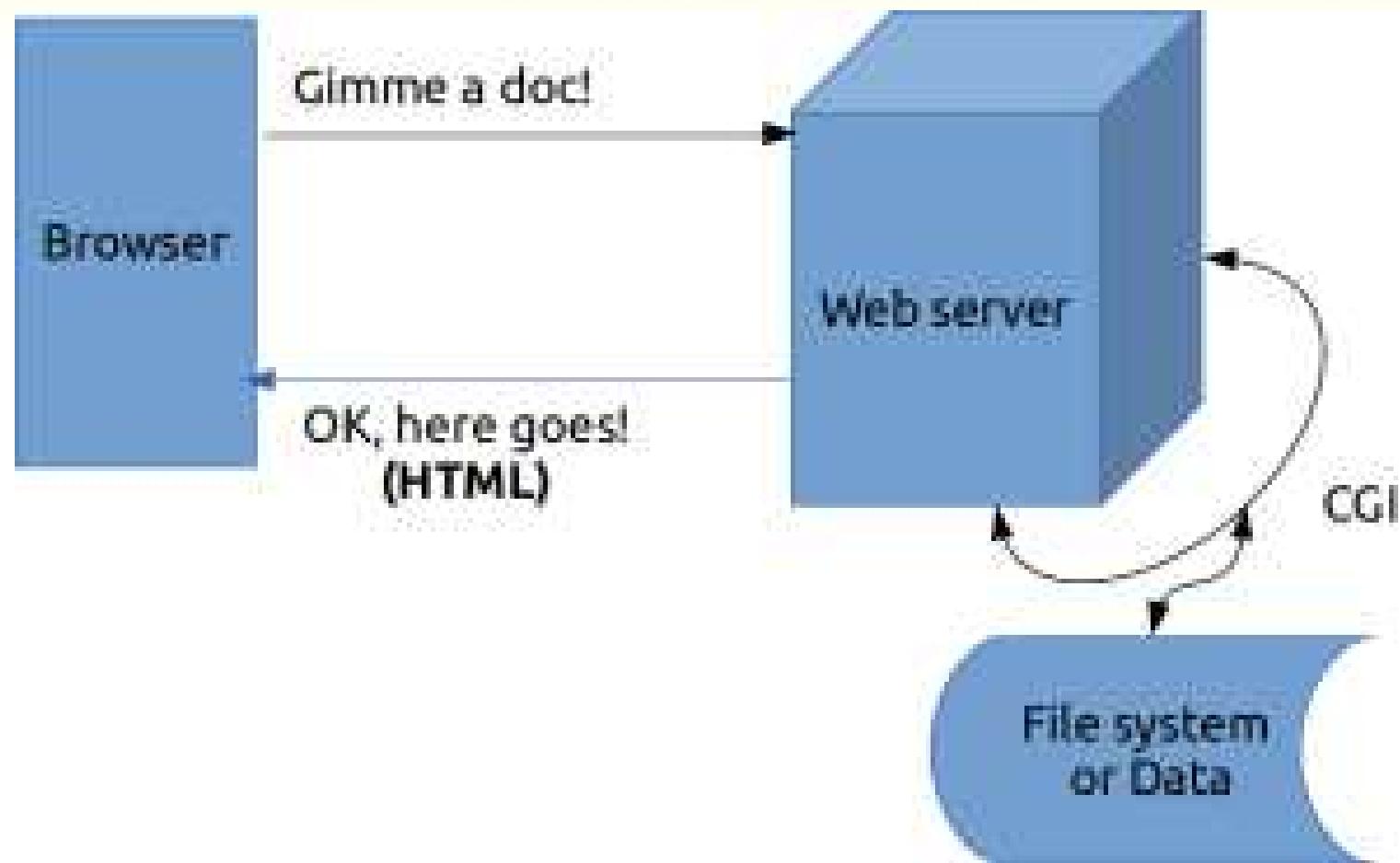
Be the First to Know

Learn about new Macintosh software releases the moment they become available. Check [Hot Mac Products](#) to hear about programs like Speed

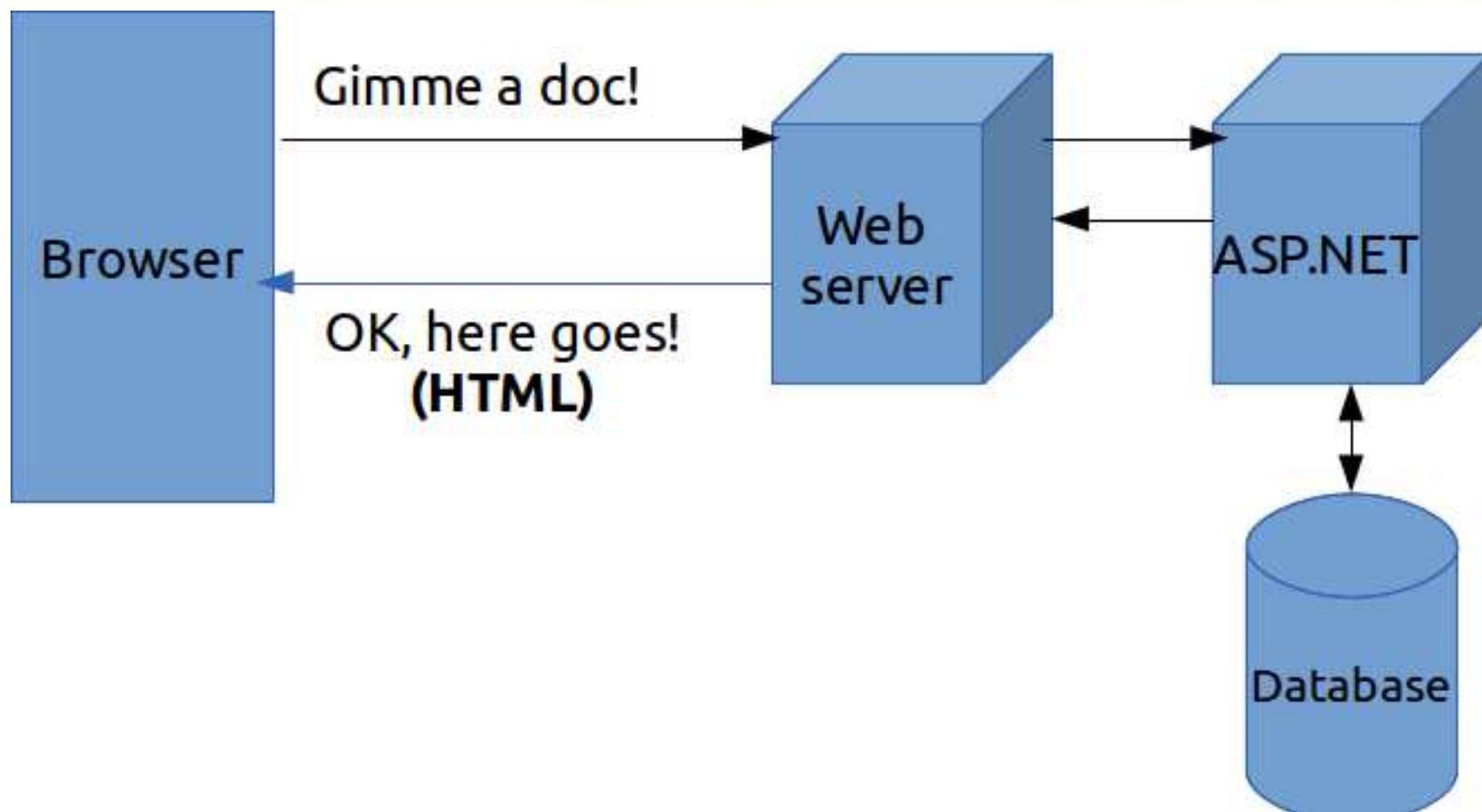
Static Web Site



CGI/Perl → till 2005

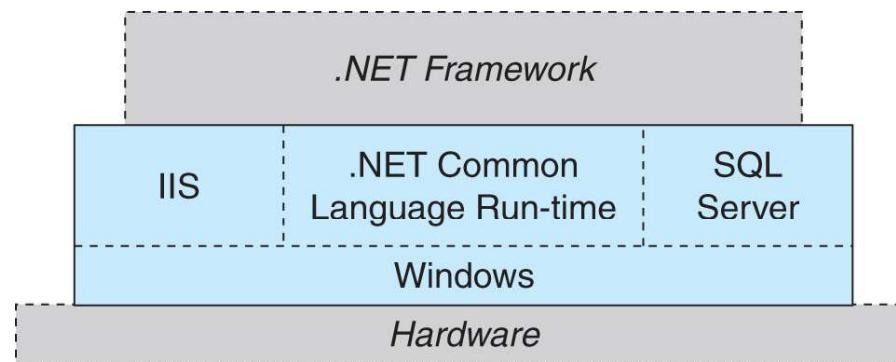


IIS & ASP



Microsoft / .NET

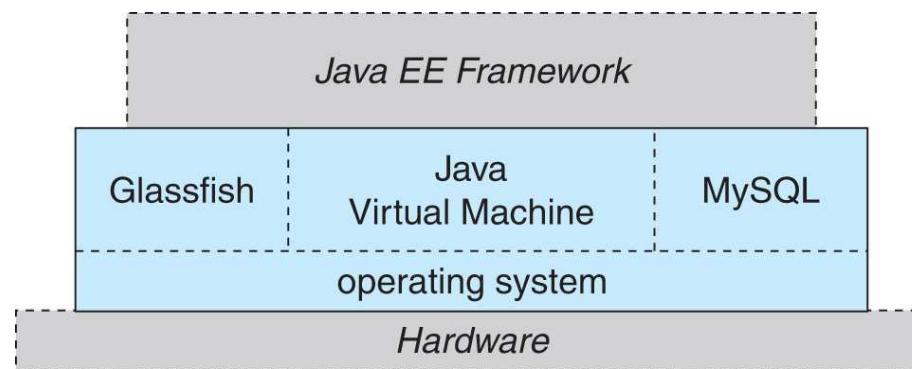
- All Microsoft products (licensed)



- .NET supports multiple languages
- Runs primarily on Windows Server O/S

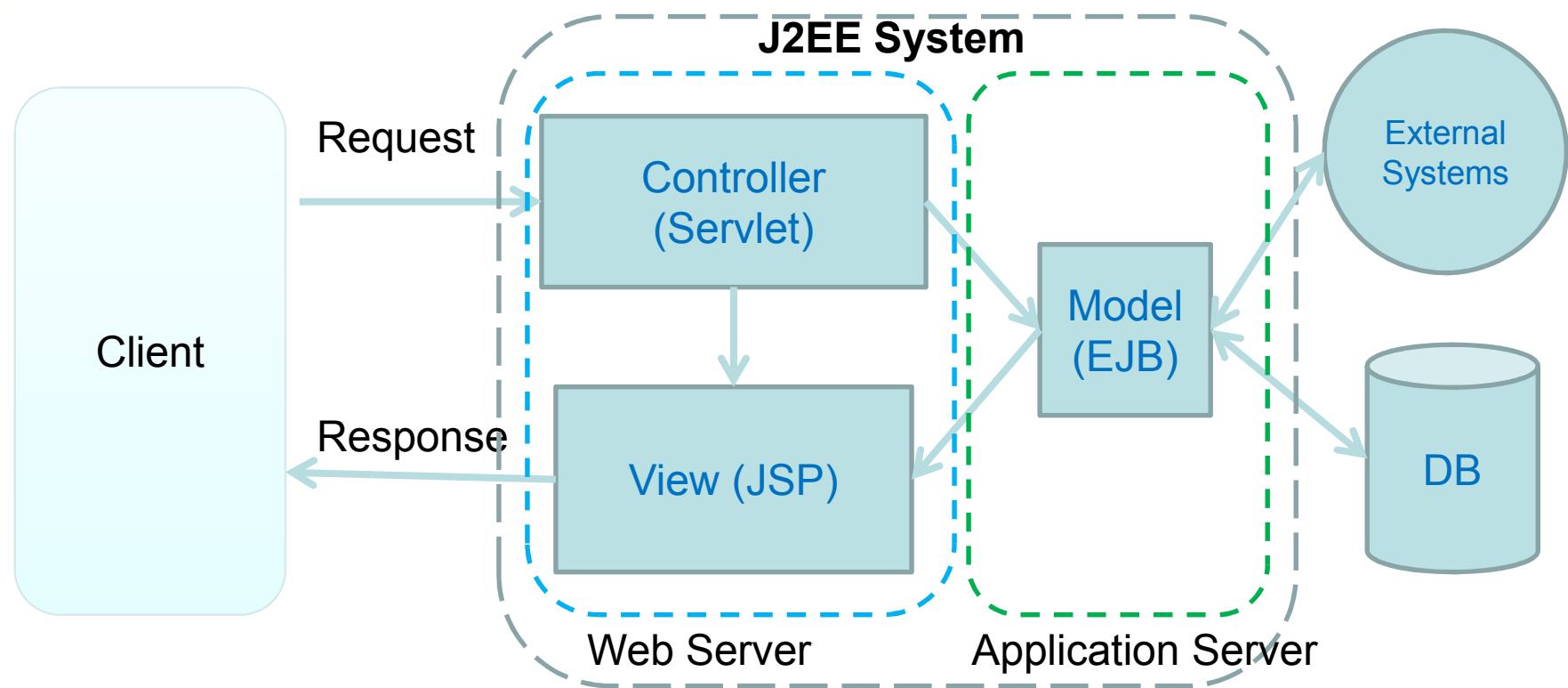
Java EE (J2EE)

- Supports Java language development



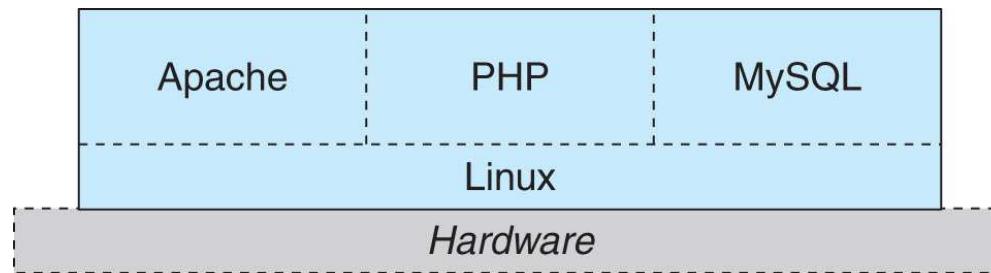
- Supported by multiple operating systems
- Proprietary, free license

J2EE Technology Overview



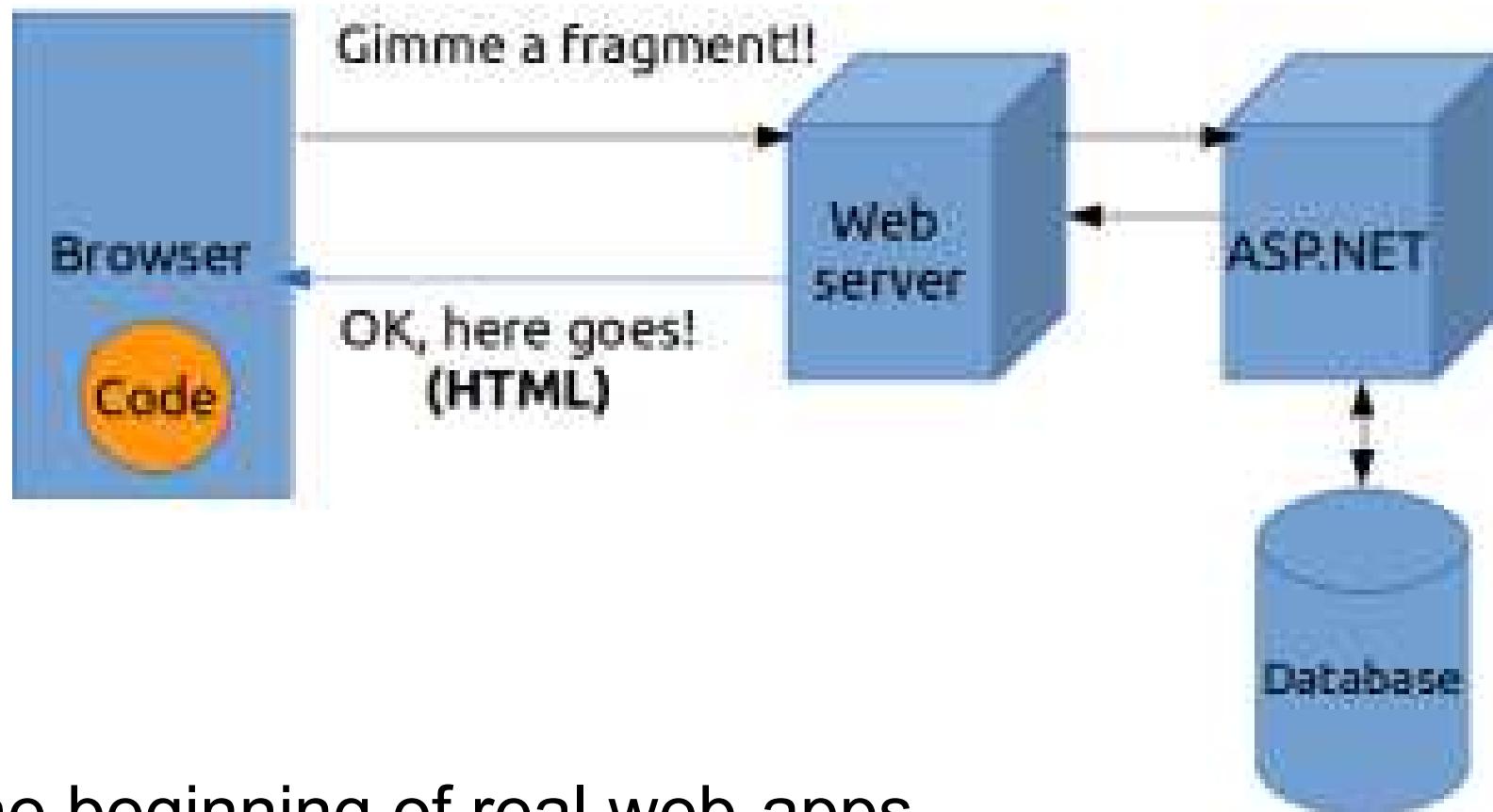
LAMP Stack

- Linux, Apache, MySQL, PHP/Perl/Python



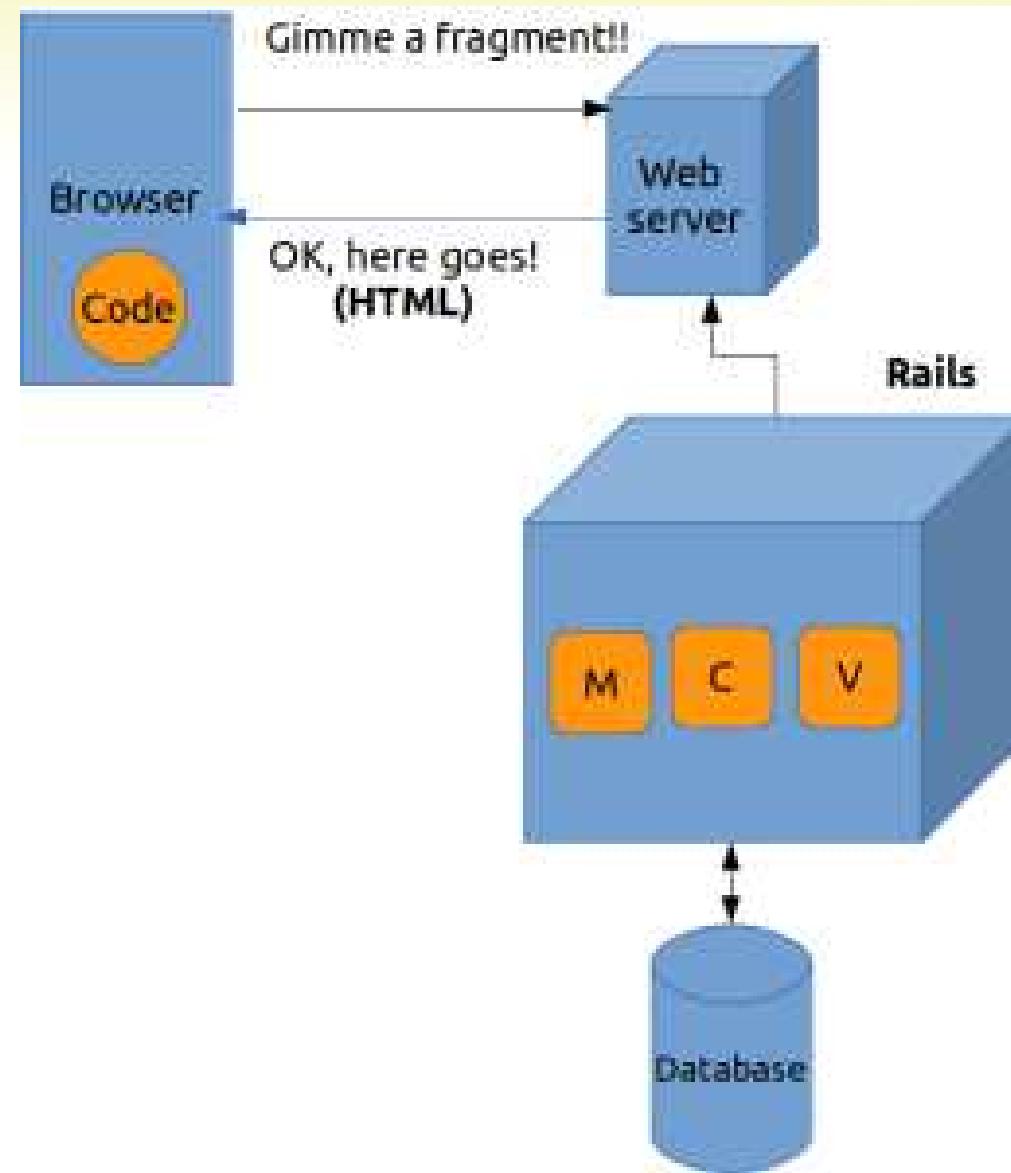
- The LAMP platform appeared in mid 1990's and has become very popular
- LAMP is open-source free software, which is one reason for its popularity

Appearance of Ajax (2005)



The beginning of real web-apps

Ruby on Rails (2007)



Ruby On Rails

- Ruby: a dynamically typed object-oriented programming language
- Rails: a webapp framework, featuring:
 - automatic code skeletons
 - built-in testing features
 - object-relation mapping
 - default implementation of common webapp functions

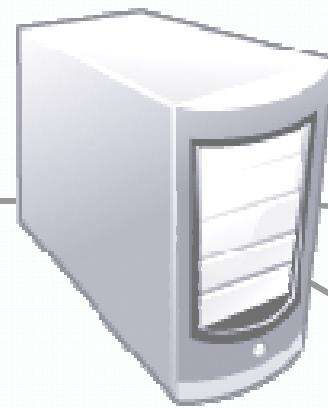
The 3-Tier Architecture Model

Data Tier
(Back-End)



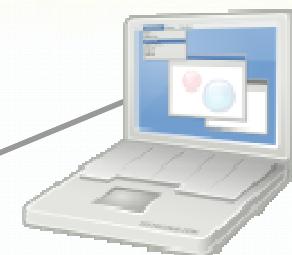
Database

Middle Tier
(Business Tier)



Business
Logic

Client Tier (Front-End)



**Client
Machine**



**Mobile
Client**



**Desktop
Client**

network

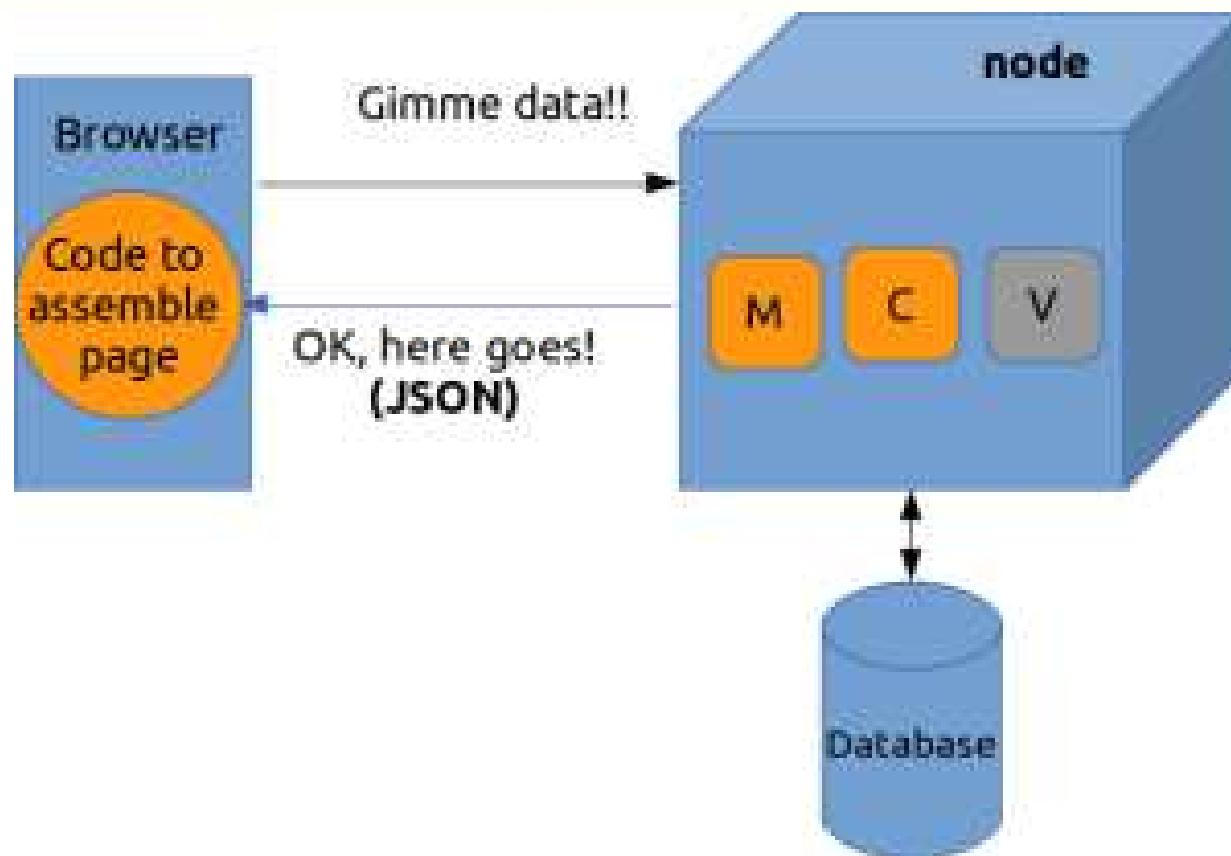
network

network

Three Trends between 2007-2010

- The rise of smart phones and mobile apps. Many applications had a web version and a mobile phone app for it.
- The rise of jQuery – a JavaScript library to build dynamic, beautiful web apps – and made Ajax easy!
- The release of Node.js –high performance JavaScript on the server.

Node.js



Focus on Client Side

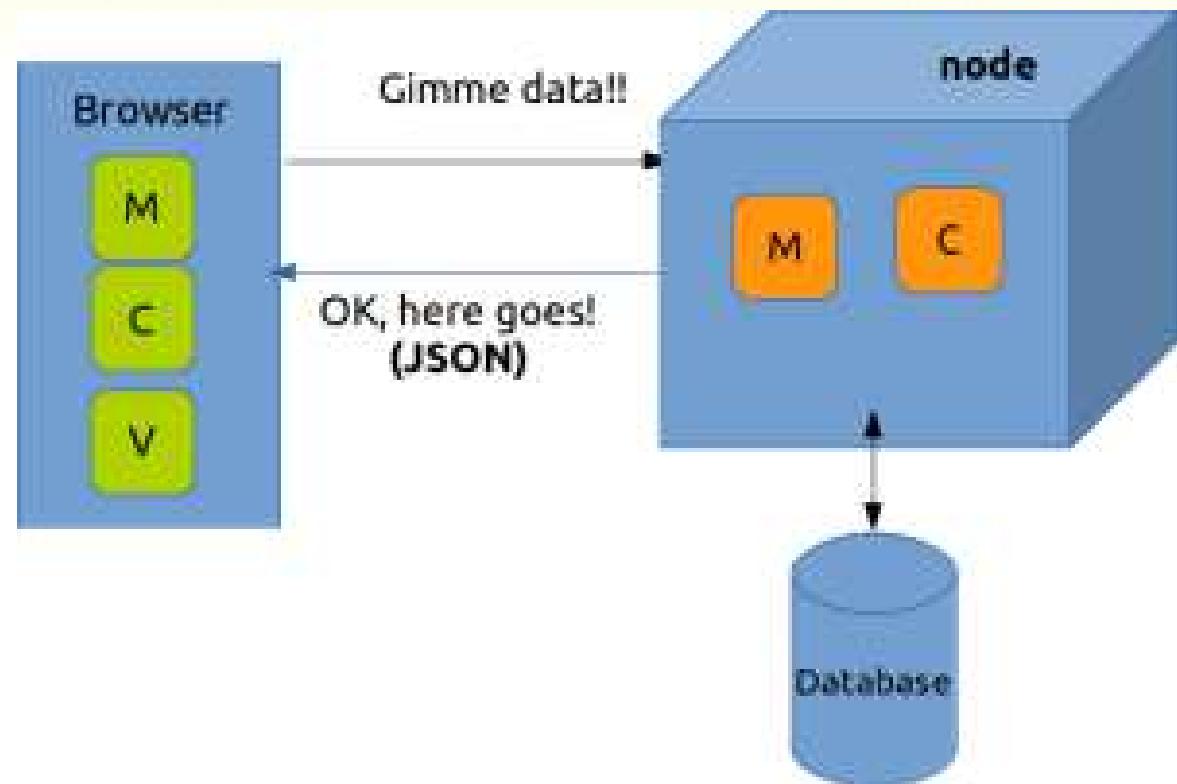
- MVC Frameworks:

✓ Backbone

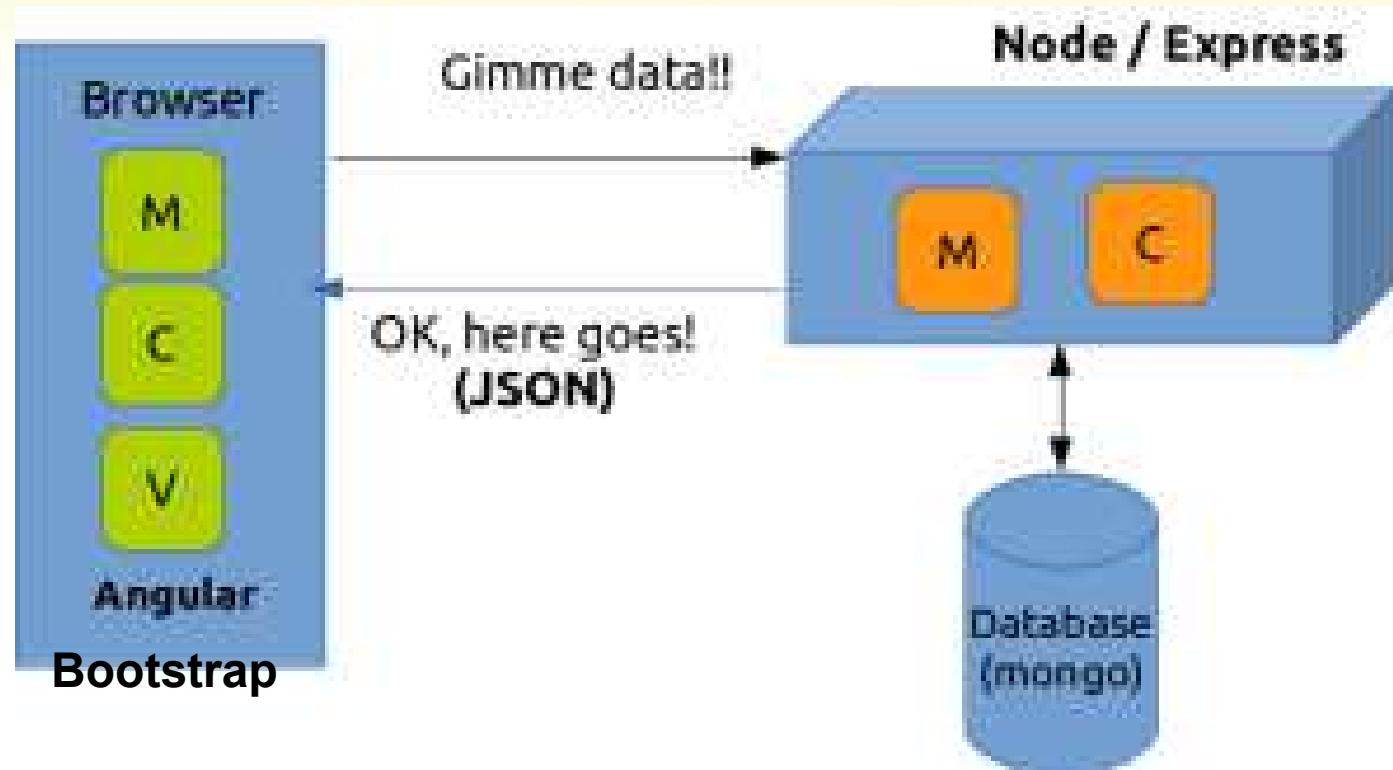
✓ Ember

✓ Knockout

✓ AngularJs



A Modern Web App Architecture



Mobility

- Everything
 - Web sites
 - Information
 - Services
- Everywhere
 - You only need your phone or tablet
- All the time



Cloud Apps



HTTP

Client-Server Communication : HTTP

- Hyper Text Transfer Protocol (HTTP)
 - Client-server protocol for transferring Web resources (HTML files, images, styles, etc.)
- Important properties of HTTP
 - Request-response model
 - Text-based format
 - Relies on a unique resource URLs
 - Provides resource metadata (e.g. encoding)
 - Stateless

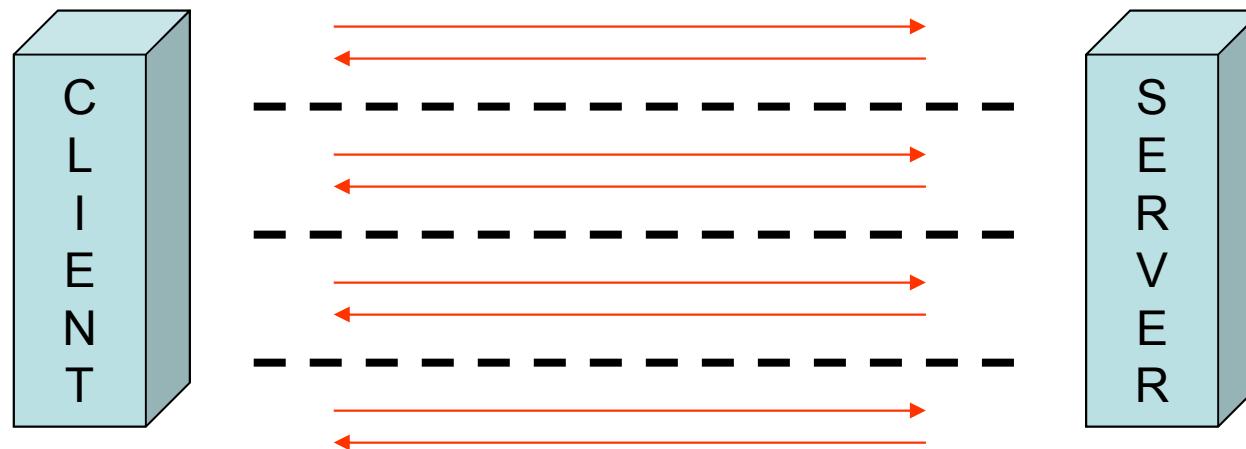
HTTP Transaction: Request-Response

- Client program
 - Running on end host, e.g. Web browser
 - Requests a resource
- Server program
 - Running at the server, e.g. Web server
 - Provides resources



Statelessness

- There is no memory (preservation of state) between HTTP transactions.



- Each HTTP transaction is independent of the one before it and the one after it.
- Cookies can overcome that.

Uniform Resource Identifier (URI)

- Each web resource is identified by a unique URL
- Uniform Resource Locator (URL) is a kind of URI
- General form for a URL:
`<scheme><domain name><port>`
For example,
`http://www.mywebsite.net:80`
(the port is usually omitted and a default is used)

HTTP Example

- HTTP request:

```
GET /academy/about.asp HTTP/1.1
```

```
Host: www.gzas.org.cn
```

```
User-Agent: Mozilla/5.0
```

```
<CRLF>
```

The empty line
denotes the end of
the request header

- HTTP response:

```
HTTP/1.1 200 OK
```

```
Date: Mon, 5 Jul 2013 13:09:03 GMT
```

```
Server: Microsoft-HTTPAPI/2.0
```

```
Last-Modified: Mon, 12 Jul 2010 15:33:23 GMT
```

```
Content-Length: 54
```

```
<CRLF>
```

```
<html><title>Hello</title>
```

```
Welcome to our site
```

```
</html>
```

The empty line
denotes the end of
the response
header

HTTP Request

Request message sent by a client consists of

- **Request line**
 - request method (GET, POST, HEAD, ...)
 - resource URI,
 - and protocol version
- **Request headers**
 - additional parameters
- **Body** – optional data
 - E.g. posted form data, files, etc.

HTTP Request Methods

- Each HTTP request contains a method attribute that identifies its purpose
- Valid methods include
 - ***GET*** retrieve a resource
 - ***POST*** submit data to be processed
 - ***CONNECT*** create a TCP/IP tunnel
 - ***DELETE*** delete a resource
 - ***HEAD*** get response headers only
 - ***OPTIONS*** get a list of supported methods
 - ***PUT*** replace a resource
 - ***TRACE*** echo the request

HTTP Response Codes

- Each HTTP response contains a response code that indicates the general outcome
- Response code categories / examples:
 - **1xx**: informational (e.g., “**100 Continue**”)
 - **2xx**: success (e.g., “**200 OK**”)
 - **3xx**: redirection (e.g., “**304 Not Modified**”, “**302 Found**”)
 - **4xx**: client error (e.g., “**404 Not Found**”)
 - **5xx**: server error (e.g., “**503 Service Unavailable**”)
- “**302 Found**” is used for redirecting the Web browser to another URL

HTTP Response

- Typical HTTP response (includes resource)
 - content type
 - length of file
 - content

```
HTTP/1.1 200 OK
```

```
Date: Fri, 31 Aug 2012 23:18:59 GMT
```

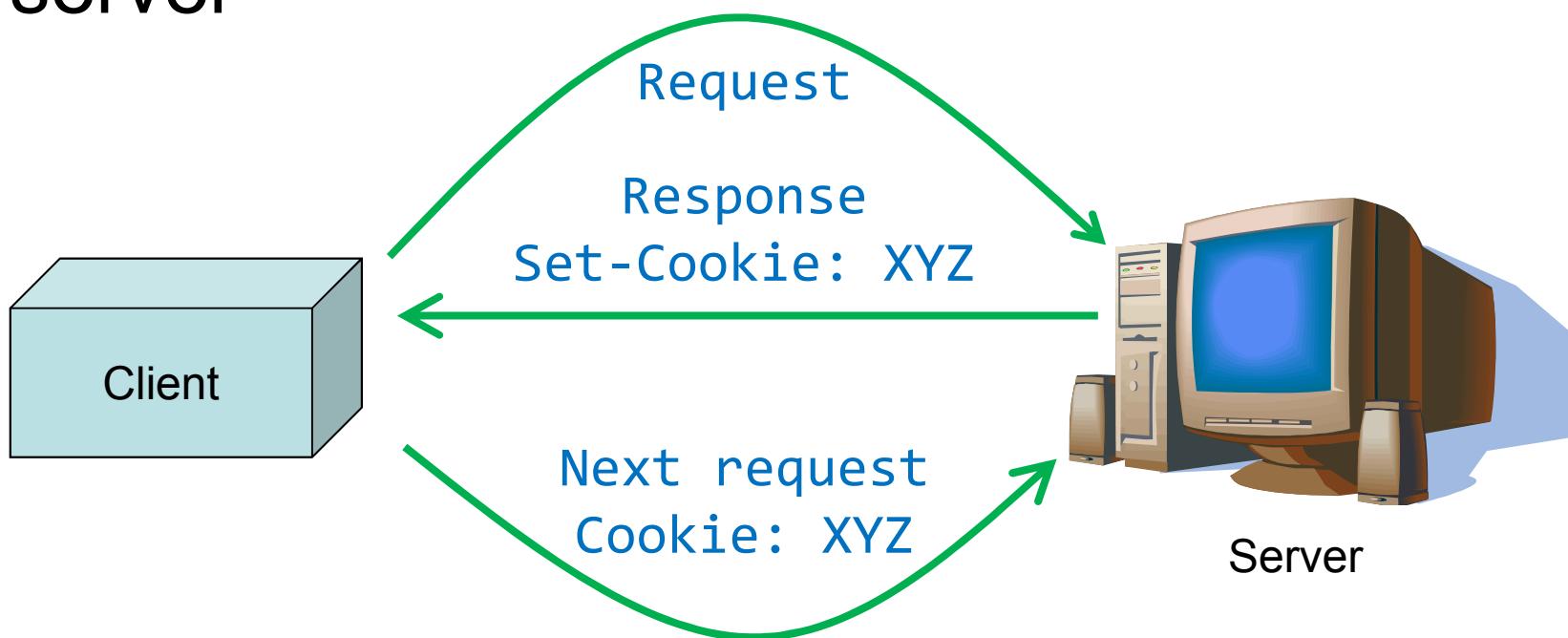
```
Content-Type: text/html
```

```
Content-Length: 1354
```

```
<html>
<body>
. . .
</body>
</html>
```

HTTP Cookies

- Cookie
- Cookies are small pieces of data stored by the client on behalf of the server
- Included in all future HTTP requests to the server



Cookies – Example

- The client requests to Google:

```
GET / HTTP/1.1  
Host: www.google.com
```

- The server sets a cookie in the HTTP response:

```
HTTP/1.1 200 OK  
Set-Cookie: PREFID=c0bf5fd5c3a25209; expires=Wed,  
11-Jul-2013 16:13:22 GMT; domain=.google.com
```

- In further requests to Google the web browser sends the cookie in the HTTP head

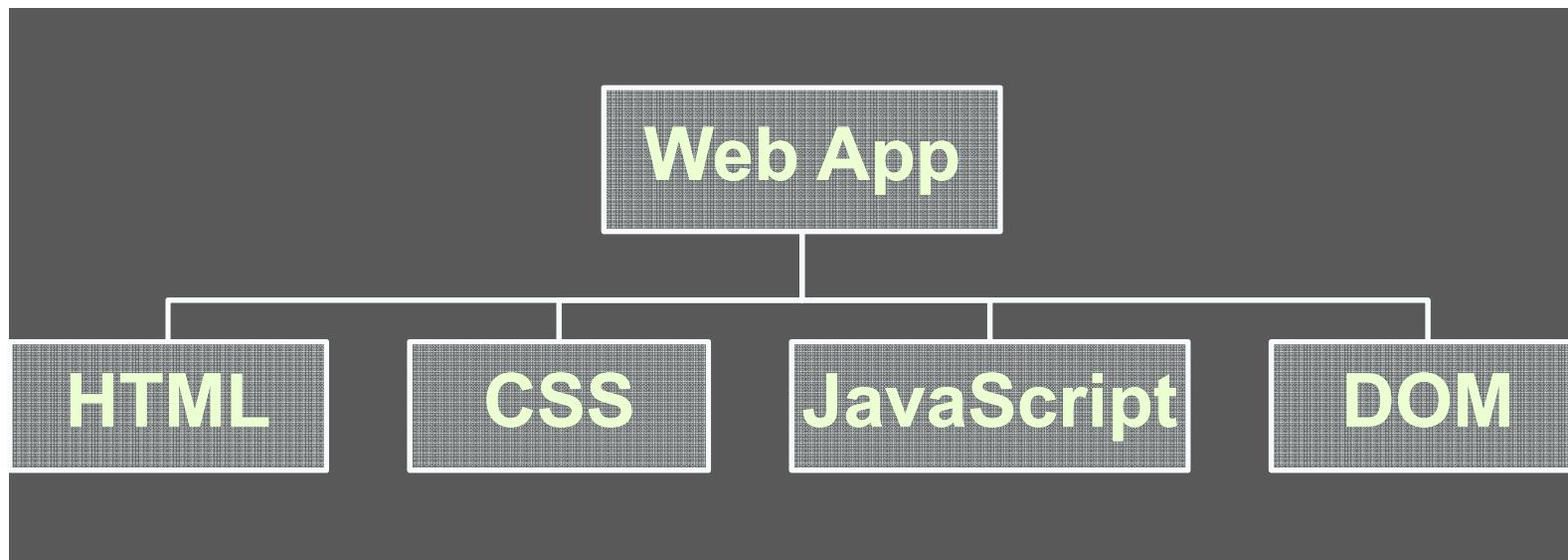
```
GET / HTTP/1.1  
Host: www.google.com  
Cookie: PREFID=c0bf5fd5c3a25209
```

Client-Side Techniques

HTML/JavaScript/DOM/Ajax

Web Apps in Front-end

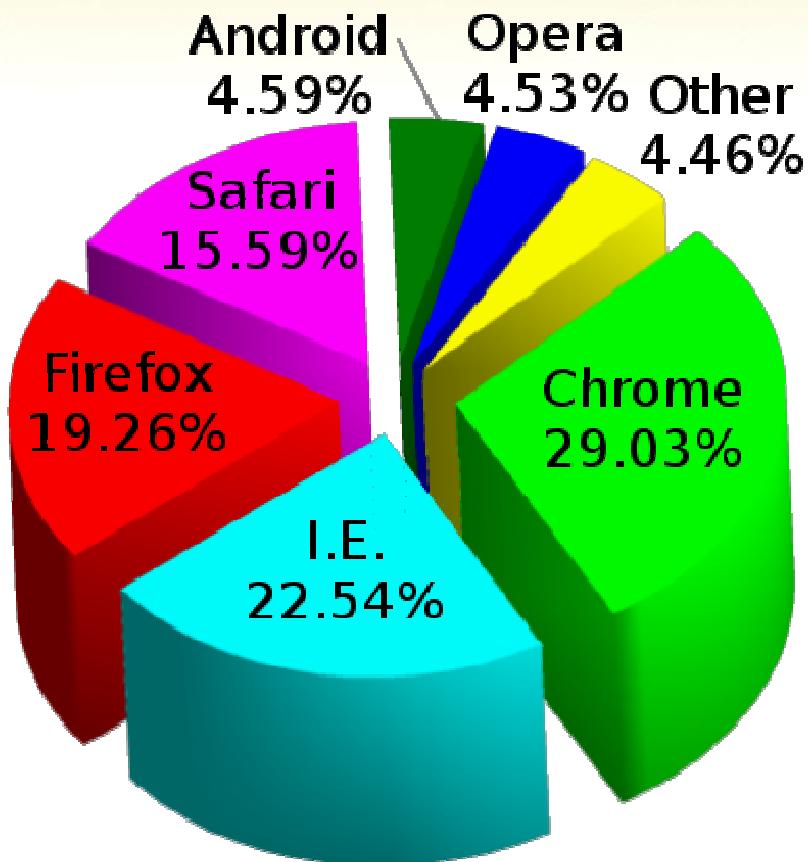
- = HTML + CSS + JavaScript



HTML + CSS + JavaScript

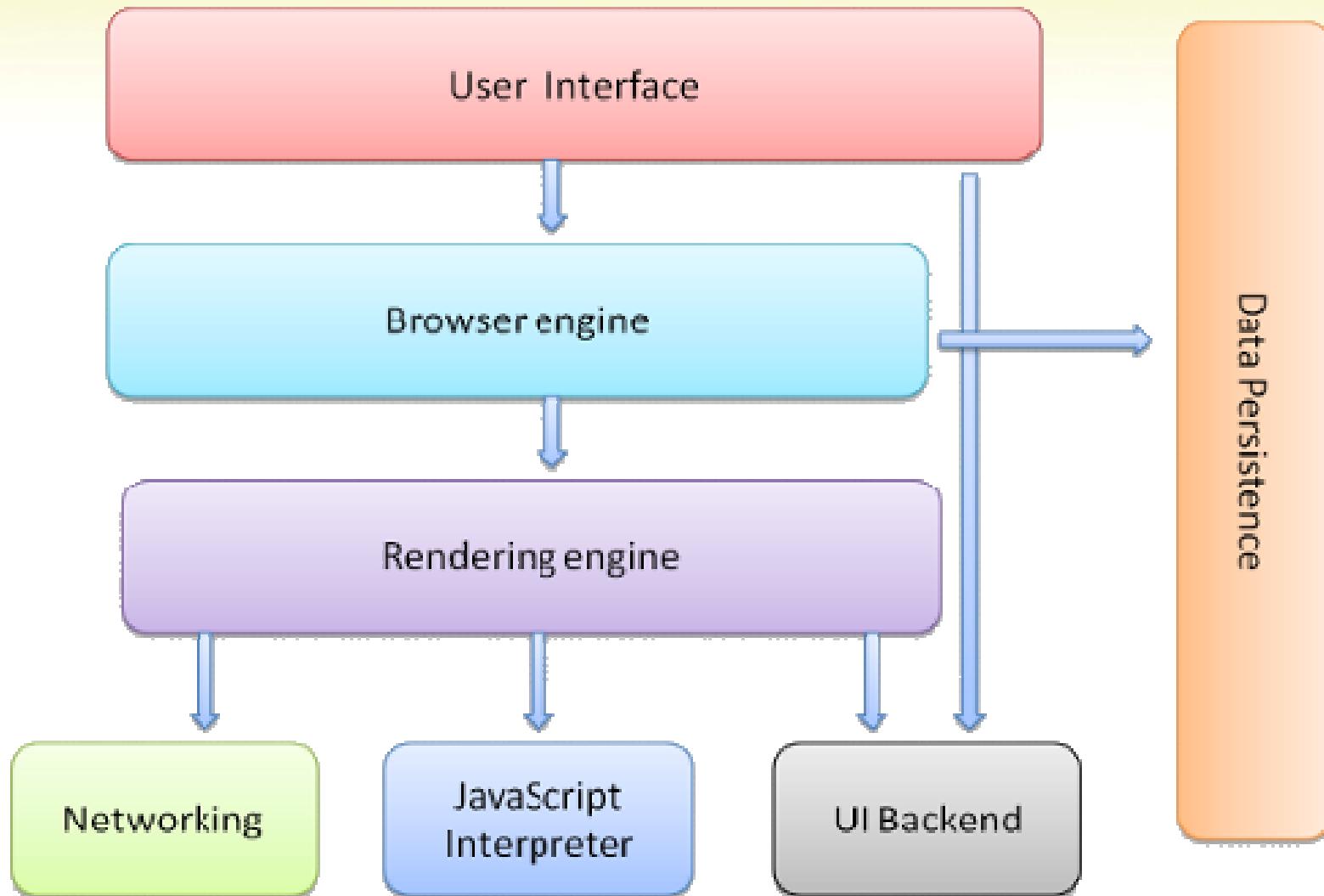
- **HTML** defines web page content through semantic tags (headings, paragraphs, lists, ...)
- **CSS** defines 'rules' or 'styles' for presenting every aspect of an HTML document
 - Font (family, size, color, weight, etc.)
 - Background (color, image, position, repeat)
 - Position and layout (of any object on the page)
- **JavaScript** defines dynamic behavior
 - Programming logic for interaction with the user, to handle events, etc.

Different Browsers

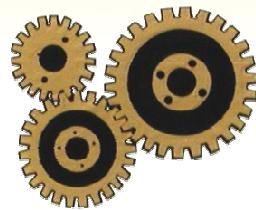


Browser usage on Wikimedia
September 2012

Browser Structure



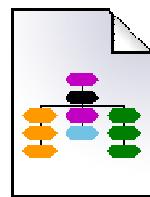
Browsers Components



Scripting engine:
interprets JavaScript



Rendering engine:
draws text, images, etc

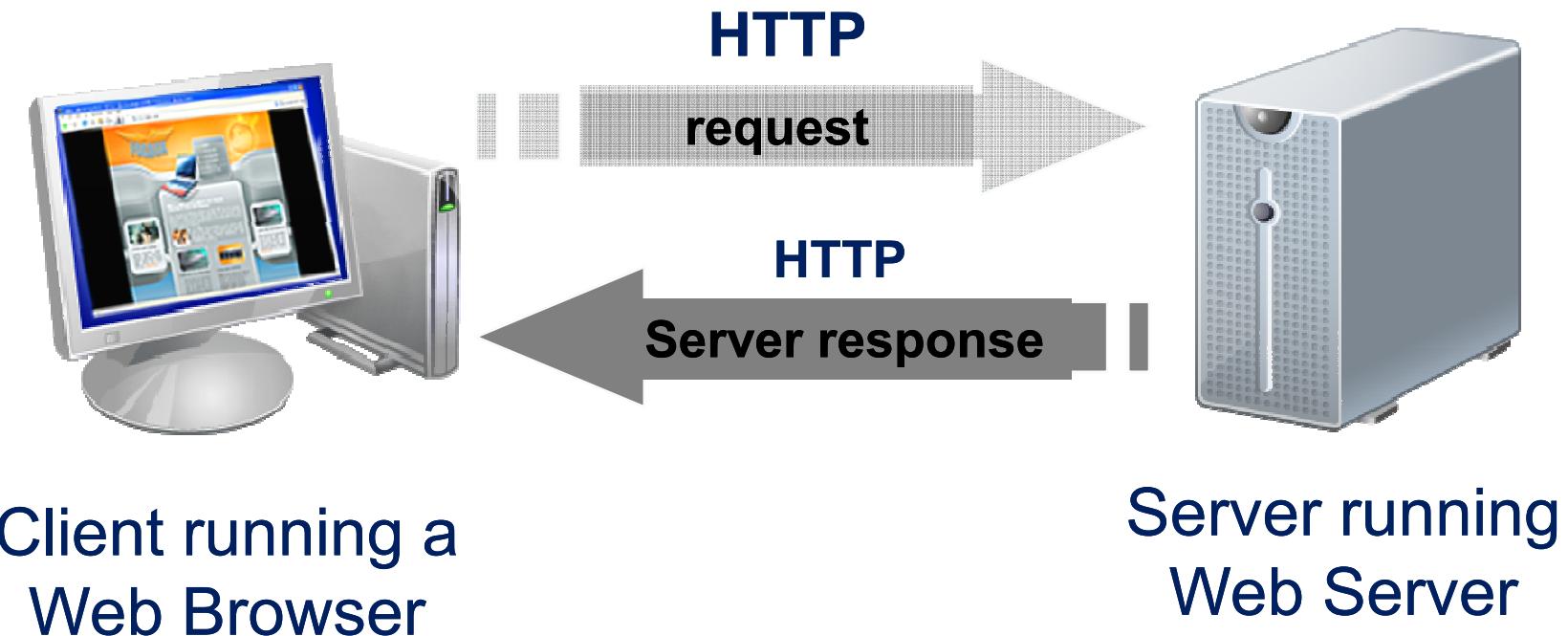


DOM:
Document Object Model

HTML

How the Web Works?

- Use classical client / server architecture
 - HTTP is text-based request-response protocol



HyperText Markup Language (HTML)

- A language for writing web pages
- HTML provides a notation for describing
 - document structure (semantic markup)
 - formatting (presentation markup)
- HTML is only about structure, not appearance
- Browsers tolerate invalid HTML code and parse errors – you should not.
- The current version is HTML 5

HTML-enabled devices

Small screen
smartphones

320 x 240
4" display
16-bit color



large screen
devices

Tablet computer:
1024 x 600
170 ppi
7"



larger screens
(desktop computers)

Desktop HD monitor
1920 x 1080
96 ppi
22" display



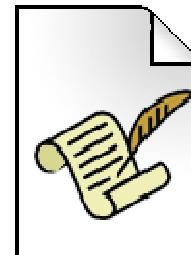
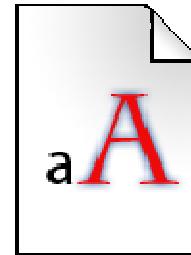
moderate screens
(laptops)

Laptop:
15.6" diagonal
1,366 x 768
96 ppi



Foundations of HTML

- Separation of concerns:
 - structure (.html)
 - presentation (.css)
 - logic (.js)
 - data



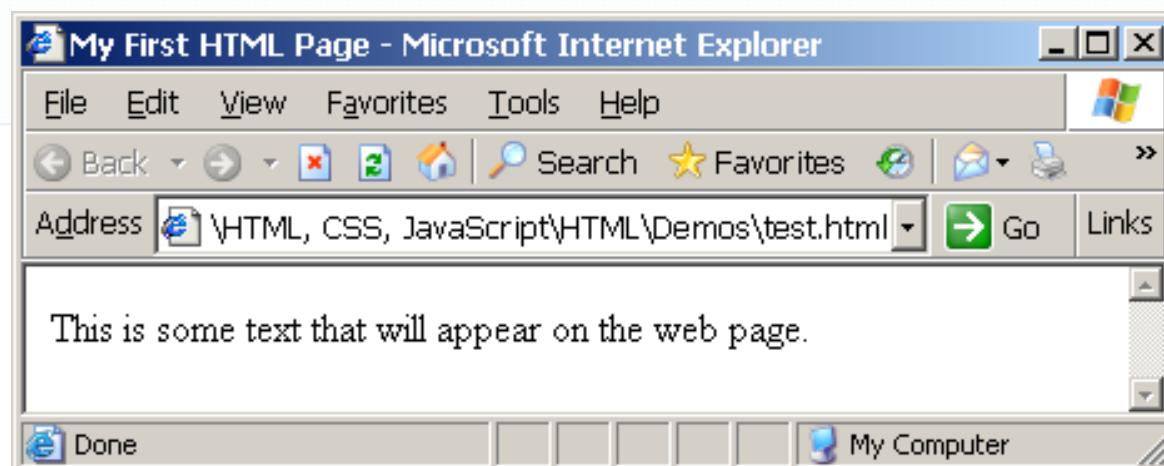
HTML document

3 major groupings/elements:

- **<html>** defines which markup language, DTD
- **<head>** defines properties of page, meta, style and js connect
- **<body>** defines page content

First HTML Page

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>My First HTML Page</title>
  </head>
  <body>
    <p>This is some text...</p>
  </body>
</html>
```

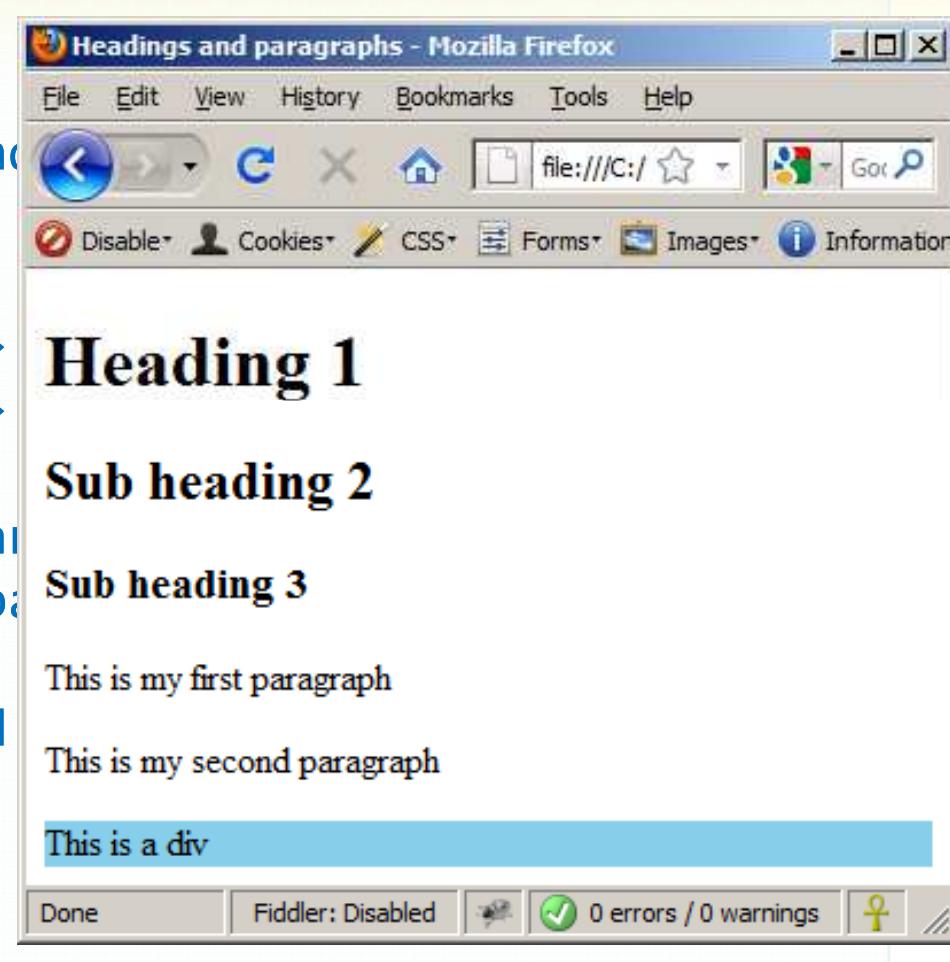


Tag Examples

```
<!DOCTYPE HTML>
<html>
  <head><title>Headings and paragraphs</title>
  <body>
    <h1>Heading 1</h1>
    <h2>Sub heading 2</h2>
    <h3>Sub heading 3</h3>

    <p>This is my first paragraph</p>
    <p>This is my second paragraph</p>

    <div style="background-color: #00FFFF; color: black; padding: 10px;">
      This is a div
    </div>
  </body>
</html>
```



HTML Forms

- HTML Form presents primary method for gathering data from client
- Form controls include
 - *text field, text area, and password box*
 - *select*
 - *check box*
 - *radio buttons*
 - *drop-down menus*
 - *buttons*
- Often used by JavaScript code

HTML Form Example

The “method” attribute tells how the form data should be sent – via GET or POST request

```
<form name="myForm" id="formId"  
method="post" action="path/to/some-script">  
...  
</form>
```

The "action" attribute tells where the form data should be sent

Form Fields

- Single-line text input fields:

```
<input type="text" id="FirstName" value="This  
is a text field" />
```

- Multi-line textarea fields:

```
<textarea id="Comments">This is a multi-line  
text field</textarea>
```

- Hidden fields contain data not shown to the user:

```
<input type="hidden" id="Account" value="This  
is a hidden text field" />
```

Form Input Controls

- Checkboxes:

```
<input type="checkbox" name="fruit"  
value="apple" />
```

- Radio buttons:

```
<input type="radio" name="title" value="Mr." />
```

- Radio buttons can be grouped, allowing only one to be selected from a group:

```
<input type="radio" name="city" value="Lom" />  
<input type="radio" name="city" value="Ruse" />
```

Other Form Controls

- Dropdown menus:

```
<select id="gender" name="gender">
  <option value="Value 1"
    selected="selected">Male</option>
  <option value="Value 2">Female</option>
  <option value="Value 3">Other</option>
</select>
```

- Submit button:

```
<input type="submit" id="submitBtn"
value="Apply Now" />
```

The <script> Tag – Example

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>JavaScript Example</title>
    <script type="text/javascript">
      function sayHello() {
        document.write("<p>Hello World!</p>");
      }
    </script>
  </head>
  <body>
    <script type=
      "text/javascript">
      sayHello();
    </script>
  </body>
</html>
```



What is HTML5?

- Latest standard for HTML
- HTML5 page start with one DOCTYPE declaration

```
<!DOCTYPE html>
<html>
```

- Getting supported by all major browsers (Chrome, Firefox, Internet Explorer, Safari, Opera)

HTML5 - New Features

- Some of the most interesting new features in HTML5 are:
 - The `<canvas>` element for 2D drawing
 - The `<video>` and `<audio>` elements for media playback
 - Support for local storage
 - New content-specific elements, like `<article>`, `<footer>`, `<header>`, `<nav>`, `<section>`
 - New form controls, like `calendar`, `date`, `time`, `email`, `url`, `search`

HTML5 Layout

HTML5 offers new elements to clearly define different parts:

- <header>
- <nav>
- <section>
- <article>
- <aside>
- <figure>
- <figcaption>
- <footer>
- <details>
- <summary>
- <mark>
- <time>



JAVASCRIPT

JavaScript Features (1/2)

- A scripting language
 - informal syntax, minimal coding
- Dynamic variable typing
 - boolean, number, string, function, object
 - implicit type conversions
 - potential problem w.r.t. correctness, security
- First-class functions
 - Functions can be passed as variable

JavaScript Features (2/2)

- Event-driven
 - Programs respond to user interface actions (mouse movement, click, keystroke, etc.)
- Server-Side or Client-Side
 - JavaScript can be used on either platform
- Client-Side functionality
 - JavaScript functions can add, change, or delete any HTML element by using DOM
 - Interaction for user interface

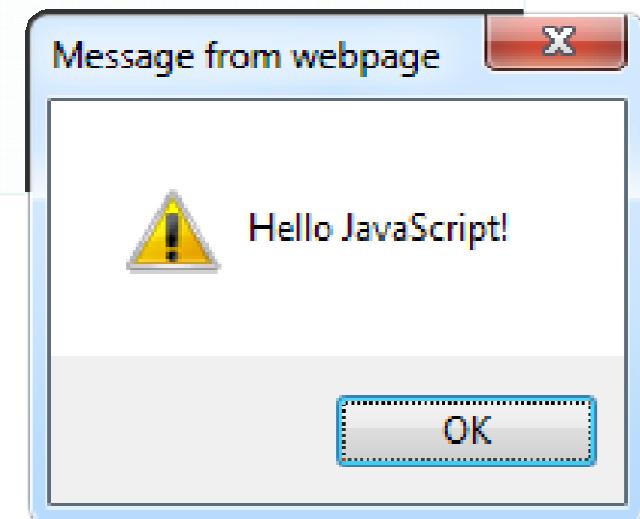
The First Example

first-script.html

```
<html>

<body>
  <script type="text/javascript">
    alert('Hello JavaScript!');
  </script>
</body>

</html>
```



Using JavaScript

- The JavaScript code can be placed in:
 - <script> tag in the head
 - <script> tag in the body
 - External files, linked via <script> tag
 - Files usually have .js extension

```
<script src="scripts.js" type="text/javascript">
<!-- code placed here will not be executed! -->
</script>
```

- Highly recommended
- The .js files get cached by the browser
- Often placed before the <body> tag

Executing JavaScript

- JavaScript code is executed during the page loading or when the browser fires an event
 - Two blocks of code can't run simultaneously
 - Some statements just define functions that can be called later
- Function calls or code can be attached as "event handlers" via tag attributes
 - Executed when the event is fired by the browser

```

```

Calling Function from Event Handler

```
<html>
<head>
<script type="text/javascript">
    function test (message) {
        alert(message);
    }
</script>
</head>

<body>
    <input type="button" value="Press"
          onclick="test('clicked!')"/>
</body>
</html>
```



JavaScript Syntax

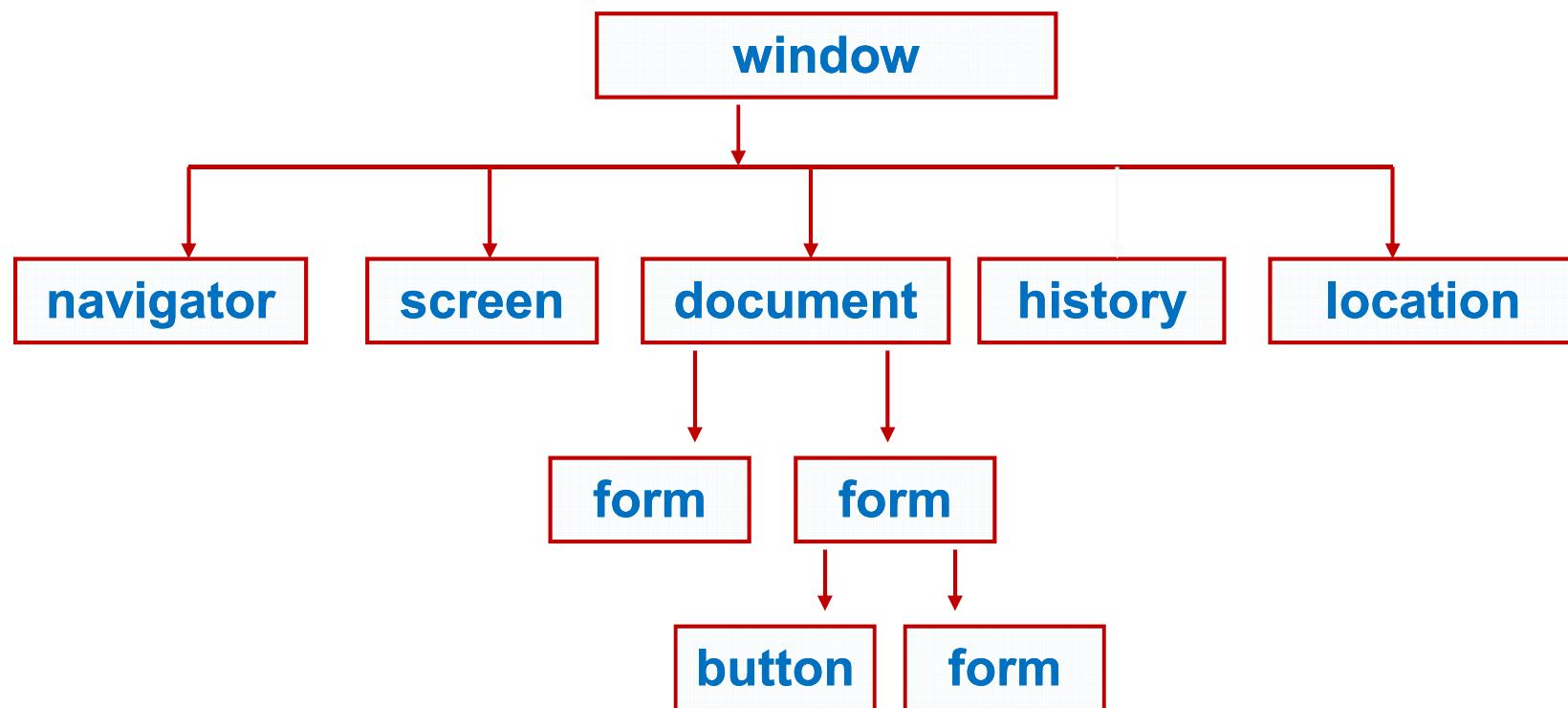
- The JavaScript syntax is similar to C# and Java
 - Operators (`+`, `*`, `=`, `!=`, `&&`, `++`, ...)
 - Variables (typeless)
 - Conditional statements (`if`, `else`)
 - Loops (`for`, `while`)
 - Arrays (`my_array[]`) and associative arrays
(`my_array['abc']`)
 - Functions (can return value)
 - Function variables (like the C# delegates)

DOM

Document Object Model (DOM)

- An interface for manipulating HTML documents
- Document is represented as a tree of objects
- Program can traverse the tree, add/delete/read/write nodes

DOM Hierarchy – Example



Properties of DOM

- Every HTML element is accessible via the JavaScript DOM API
- Most DOM objects can be manipulated by the programmer
- The event model lets a document to react when the user does something on the page
- Advantages
 - Create interactive pages
 - Updates the objects of a page without reloading it

Access Elements by Id and Others

- Access elements via their ID attribute

```
var elem = document.getElementById("some_id")
```

- Via the name attribute

```
var arr = document.getElementsByName("some_name")
```

- Via tag name

```
var imgTags = el.getElementsByTagName("img")
```

- Returns array of descendant elements of the element "el"

Changing the Document

- To put information into the document:
 - Create new elements; replace, or append to, existing nodes

```
// Find thing to be replaced
var mainDiv = document.getElementById("main-page");
var orderForm = document.getElementById("target");
// Create replacement
var paragraph = document.createElement("p");
var text = document.createTextNode("Here is the new
text.");
paragraph.appendChild(text);
// Do the replacement
mainDiv.replaceChild(paragraph);
```

.innerHTML

- A property that gets or sets the text between the start and end tags
- Argument completely replaces elements' existing content
- If the new value contains HTML tags, it is parsed and formatted before being placed into the document.
- Example:

```
document.getElementById('fo').innerHTML='<p>tex</p>'
```

<div> and

- Use to identify an element in the DOM
- Get or update its inner content.

```
<body>
  <h1>Test</h1>
  This is some normal text with
  a <span id="fo">span</span> in it.
  <div id="bar">And this is a div.</div>
  Here is some more normal text.
</body>
```

```
...
document.getElementById('fo').innerHTML = "<p>text</p>";
...
```

DOM Events

- JavaScript can register event handlers
 - Events are fired by the browser and are sent to the specified JavaScript event handler function
 - Can be set with HTML attributes:

```

```

- Can be accessed through the DOM:

```
var img = document.getElementById("myImage");
img.onclick = imageClicked;
```

Common DOM Events

- Mouse events:
 - `onclick`, `onmousedown`, `onmouseup`
 - `onmouseover`, `onmouseout`, `onmousemove`
- Key events: only for input fields
 - `onkeypress`, `onkeydown`, `onkeyup`
- Interface events:
 - `onblur`, `onfocus`
 - `onscroll`

Common DOM Events (2)

- Form events
 - `onchange` – for input fields
 - `onsubmit`
 - Allows you to cancel a form submission
 - Useful for form validation
- Miscellaneous events
 - `onload`, `onunload`
 - Allowed only for the `<body>` element
 - Fires when all content on the page was loaded / unloaded

Review: JavaScript/DOM

- A scripting language for browsers
 - Can make pages interactive w/o sending requests to server
 - Syntax is very similar to Java, but internals are very different
 - Lots of small differences between browser implementations
- JavaScript can interact with the DOM
 - Examples: Read data from forms, replace parts of the page
 - Can register event handlers with DOM elements to respond to clicks, keypresses, mouse movements, selections...

AJAX

What is Ajax?

- **Asynchronous JavaScript and XML**
 - First mentioned by Jesse James Garrett in 2005
- Not a single technology - a mix of technologies for building faster web apps
 - **HTML** and **CSS** for presentation
 - **DOM** for dynamic display
 - **XML** or **JSON** for data interchange
 - **XMLHttpRequest** for asynchronous requests
 - **JavaScript** for binding everything together

Advantages of Ajax

- Rich and responsive user interfaces
- Reduced HTTP traffic, increased response time.
- Merrill (2008)*:
 - 61% speed increase
 - 73% decrease in bytes transferred
- Full browser support
- Many libraries available

*Different researchers come up with different results

@2015, Li Dan, Guizhou Academy of Sciences

Disadvantages of Ajax

- Security; sandbox/server of origin
- Browser navigation tools defeated
- More complex than traditional web pages
- Increases design/development time

XMLHttpRequest

- A JavaScript object that enables web pages to dynamically load more content
- Request can be **asynchronous**
 - Browser performs the HTTP request in the background while the user continues to interact with the web page
 - Script defines a callback function that should be invoked when the requested content has arrived

XMLHttpRequest workflow

1. Instantiate a new XMLHttpRequest object
2. Prepare the object
 - Call `open()` to set the URL and the method (GET, POST, ...)
 - Can add headers, HTTP authentication, ...
 - Define a `callback function` that will be called by the browser when the results are available
3. Send the request
 - Invoke `send()`, optionally with data to submit (for POST)
4. Handle invocations of the callback function

Ajax Example

```
var xmlhttp = new XMLHttpRequest();

xmlhttp.open('GET', url, true);

xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState == 4) {
        if (xmlhttp.status == 200) {
            var mess=document.getElementById("messId");
            mess.innerHTML = xmlhttp.responseText;
        } else
            alert('Ajax failed; status:' + xmlhttp.status);
    }
}

xmlhttp.send(null);
```

Ajax ReadyState Values

Ready State	Significance
0 Uninitialized	Request not yet opened
1 Loading	Not yet sent
2 Loaded	Sent; no information available
3 Interactive	Partial response received
4 Completed	Response complete

HTTP Respond Status

Status codes:

200 = OK

301 = moved permanently

302 = found/redirected

401 = unauthorized

403 = access denied

404 = not found

```
var xmlhttp = new XMLHttpRequest();  
if (xmlhttp.status == 200)
```

Popular JavaScript Libraries

- **jQuery.** One of the most popular libraries. It is lightweight and focused on encapsulating events and Ajax, providing a convenient shortcut to the `getElementById` function, and supplying user interface effects.
- **Prototype.** It has fundamental support for events and Ajax, and has similar—but more extensive—shortcuts for accessing elements of web page; but the differentiator is that Prototype emphasis on classes and it supports object oriented design using inheritance.
- **Dojo.** Dojo starts out with basic support for events and Ajax, and adds widgets—little collections of JavaScript, CSS, and HTML that together create a visual user interface object for displaying or manipulating data. The widgets developed for the library cover a wide range of needs: menus, sortable tables, drag-and-drop trees, calendars, faders, charts, etc.

Ajax without XML

- Despite its name, XMLHttpRequest can handle content other than XML
- JSON is another option to exchange data between client and server

JSON: JavaScript Object Notation

- Data interchange format
- Passes objects as **strings** from server to client
- Basic form: uses *name : value* pairs
- Strings in quotes; numbers not

```
{"firstname": "Wang"}
```

```
firstname = "Wang";
```

```
var student={"name": "Ali", "age": 20, "major": "MBBF" };

var nm1=student.name;
var nm2=student["name"];
```

JSON

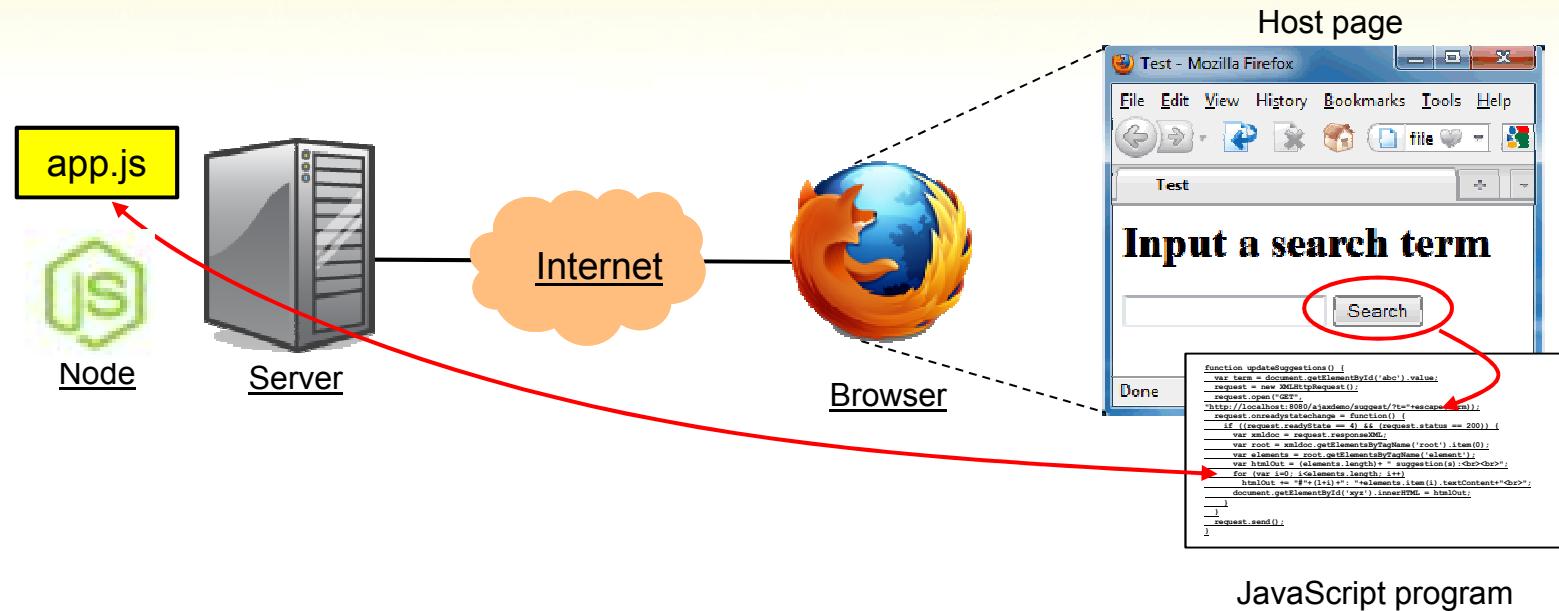
- Curly brackets define entities
- Square brackets define arrays of entities

```
var students = [  
    {"name": "Ali", "age": 20, "major": "MBBF"},  
    {"name": "Bahar", "age": 21, "major": "ITF"}  
];  
  
var ag=students[1].age;
```

Single-Page Web App

- One host (main) HTML file
- One screen at any given time
- Generating (or toggling) other screens dynamically, using JS
- Instead of navigating between different pages, one navigates between different sections of one web page
- Maintaining state – what's currently in the DOM
- Maintaining data – what's needed for generating DOM

Building Web Applications



- **Host page:** A **HTML** page that we'd like to make interactive.
- **CSS** is used to decorate the main page.

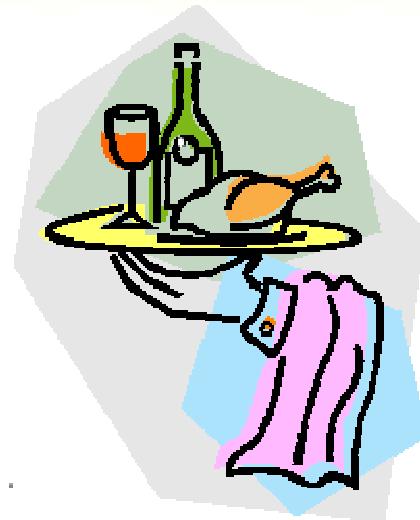
Server-Side Techniques

NodeJS/MongoDB/REST

What is a server?

Software that provides services:

- Web server
- Database server
- File server
- Print server
- Mail server

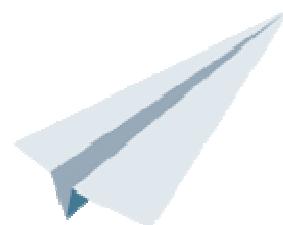


Web Server Responsibilities

- Web servers are software products that handle web requests and manage connections
- These requests are redirected to other software products (ASP.NET, PHP, etc.), depending on the web server settings

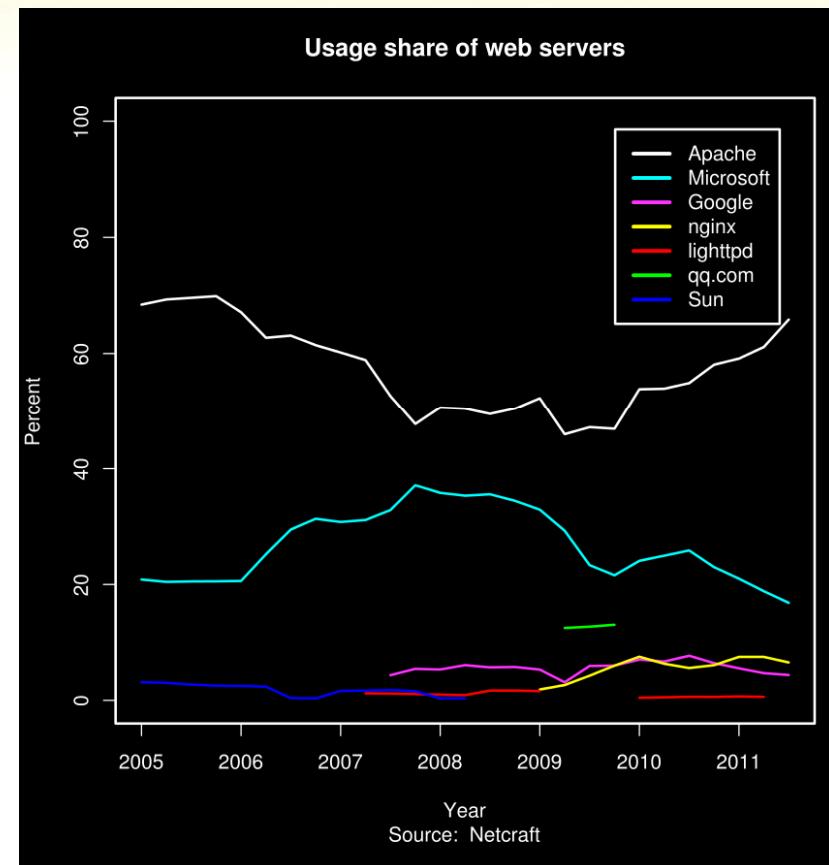
Web Servers

Apache, IIS, Nginx, Lighttpd, etc.



Web Servers Market Share 2014

- Apache - 60.4%
- Nginx - 21.0%
- IIS - 12.3%
- LiteSpeed - 2.0%
- Google Server - 5.09%
- Tomcat – 0.4%
- Lighttpd - 0.3%
- Node.js – 0.1%



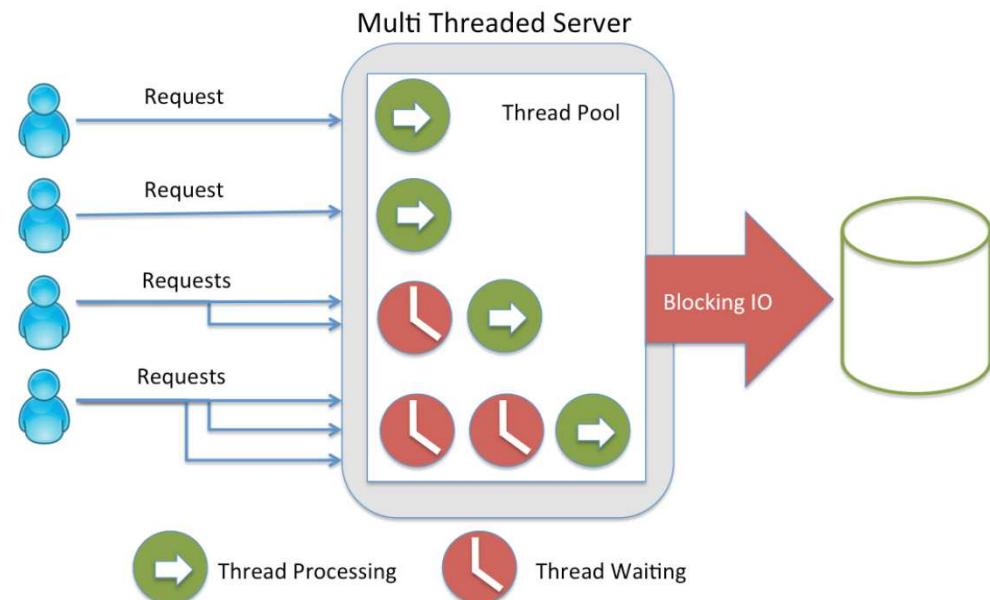
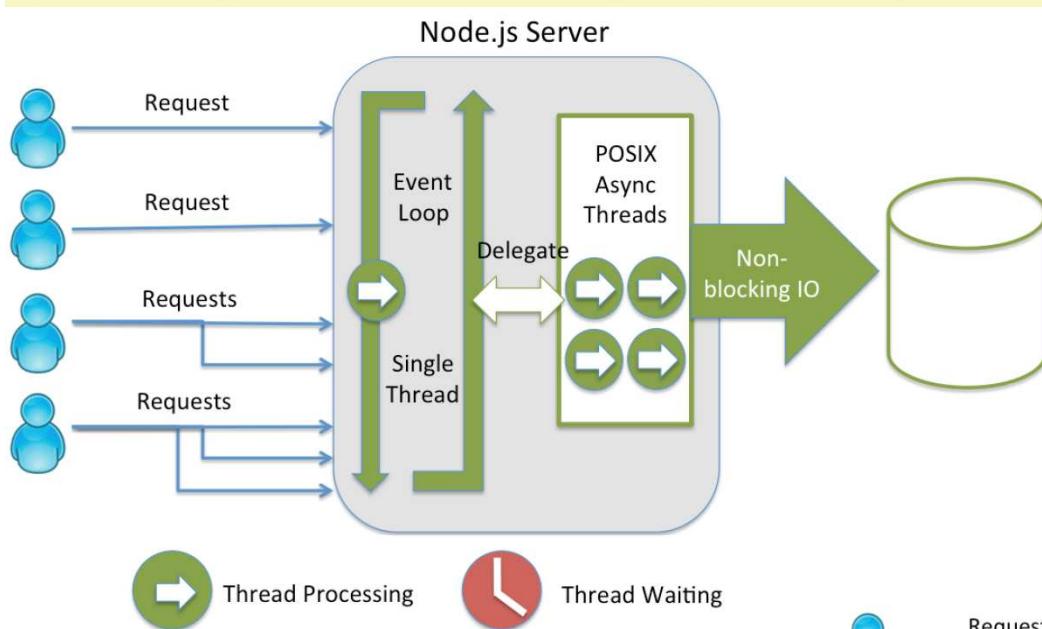
NODE.JS

Introduction to Node.js

“Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.”

- nodejs.org

Threads vs. Event-Driven



Why Node.js Use Event-based?

In a normal process cycle the web server while processing the request will have to wait for the I/O operations and thus blocking the next request to be processed.

Node.JS process each request as events, The server doesn't wait for the IO operation to complete while it can handle other request at the same time.

When the I/O operation of first request is completed it will call-back the server to complete the request.

Node Potential Wins

- JavaScript is the popular language of the web
- Web applications spend most of their time doing I/O
- Can implement the same programming language on client and server. Code can be migrated between server and client more easily
- Common data formats (JSON) between server and client
- Common software tools, testing or quality reporting tools for server and client
- Node.js ships with a lot of useful modules, so you don't have to write everything from scratch.
- Thus, Node.js is really two things: a runtime environment and a library

Node.js vs. Apache

Platform	Number of request per second
PHP (via Apache)	3187,27
Static (via Apache)	2966,51
Node.js	5569,30

Node.js vs. PHP + Nginx

Platform	Static (number of request per second)	Query MySQL (number of request per second)
PHP + Nginx	3624	1293
Node.js	7677	2999

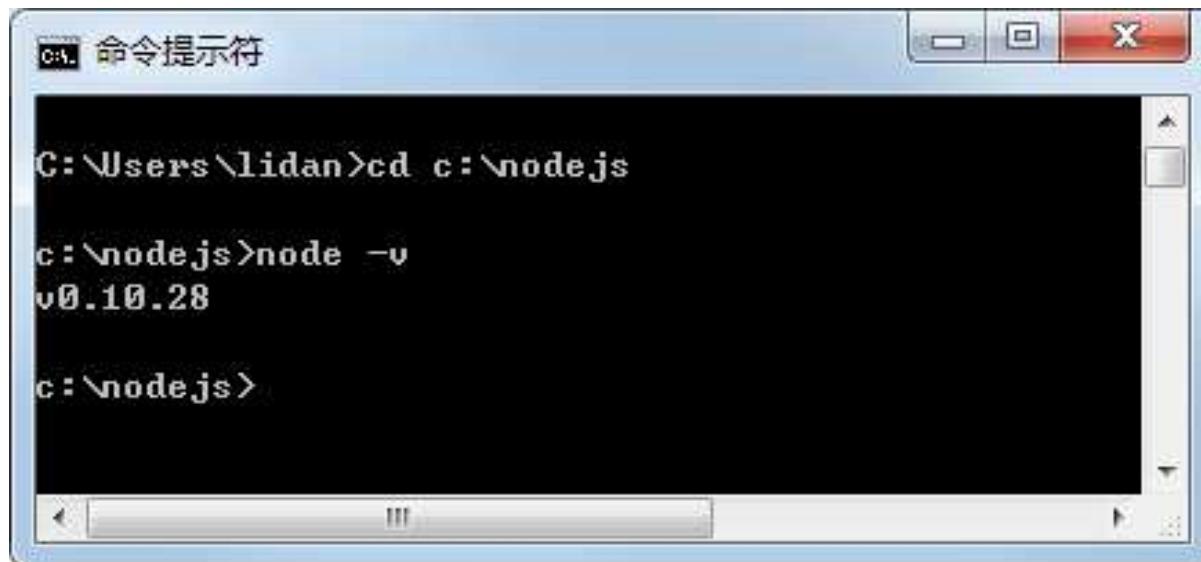
Lab : Install NodeJS

- Download and install node.js : <http://nodejs.org/>
 - Click ‘Install’ or go to the ‘Downloads’ page
 - Once downloaded, run the installer
 - Install Node.js in directory **c:\>nodejs**



Test the Installation

- Open a cmd, go to the **nodejs** directory
- Type **node -v** in command line
- If the version information is shown, Node.js is correctly installed.



```
C:\Users\lidan>cd c:\nodejs

c:\nodejs>node -v
v0.10.28

c:\nodejs>
```

A Basic HTTP Server

- Create `helloworld.js` file in the nodejs folder as the follows

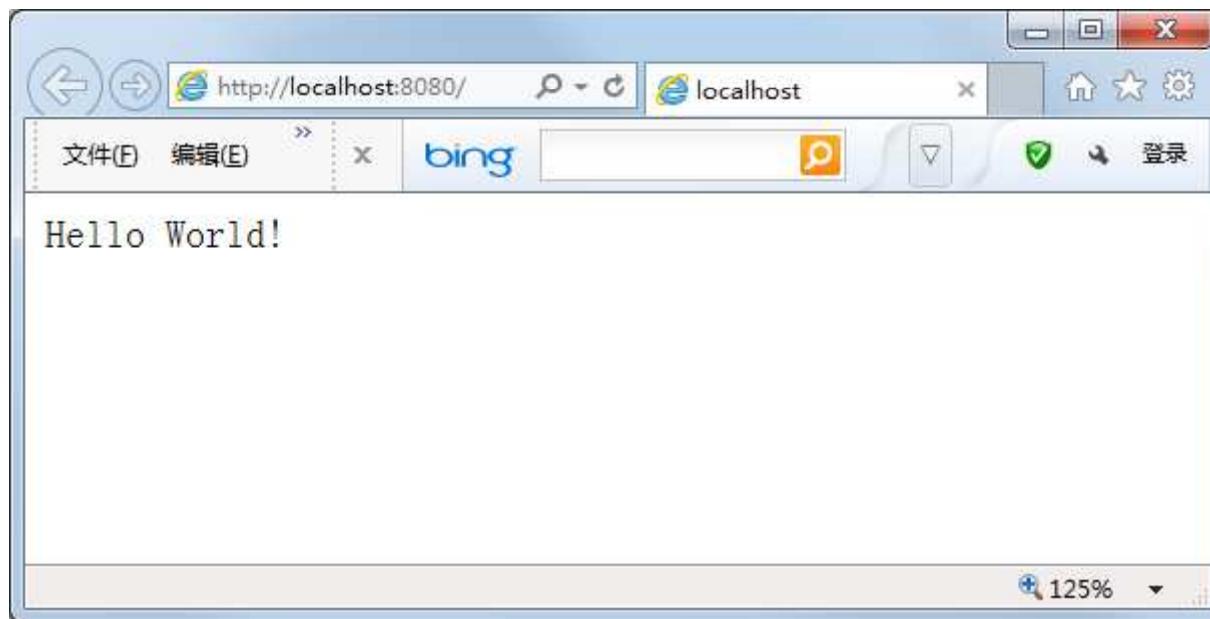
```
var http = require("http");
http.createServer(function(request, response) {
    response.writeHead(200, {"Content-Type": "text/html"});
    response.write("Hello World!");
    response.end();
}).listen(8080);
console.log("Server has started.");
```

- Execute the js file in Node.js:



Test the HTTP Server

- Open your browser and access <http://localhost:8080>. This should display a web page that says “Hello World”.



Analyzing the HTTP Server

```
var http = require("http") ;  
  
http.createServer(function(request, response) {  
    response.writeHead(200, {"Content-Type": "text/html"}) ;  
    response.write("Hello World!") ;  
    response.end() ;  
}).listen(8080) ;  
  
console.log("Server has started.") ;
```

requires the “http” module from Node.js library and assigns it to variable http

the parameter of the createServer : an anonymous function to callback when a http request is arrived

Invokes method listen of the object , listen to a http port

calls function createServer of the http module, returns an object

mongoDB



NoSQL Market Data

2014
8%

% NoSQL Enterprise Adoption

2010
5% 2015
20%



"MongoDB is the new MySQL."

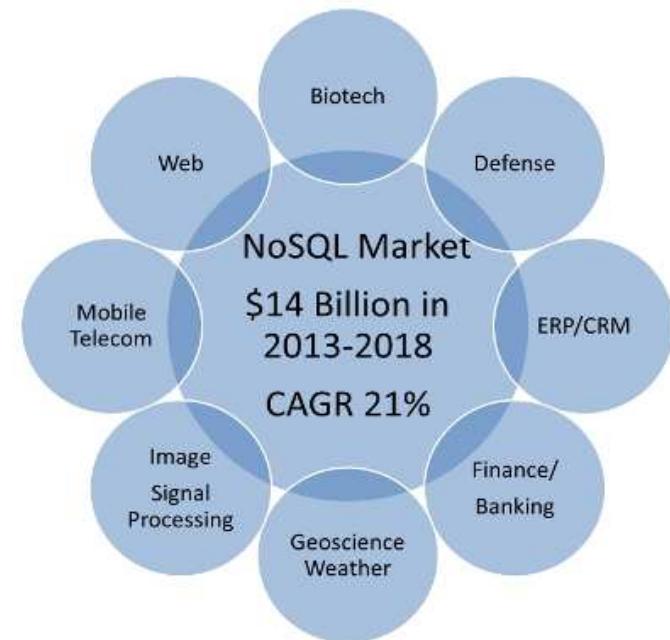
FORRESTER®

"Current adoption of NoSQL in enterprises is 4% in 2010. expected to double by 2014, and grow to 20% by 2015. 10% of existing apps could move to NoSQL. [Sept 2011]

Gartner

"NoSQL is in its infancy, many immature with only community support and limited tool capability; however, expect to see a few used more widely during the next 5 years."

- 2011: Market Research Media noted worldwide NoSQL market was expected to reach ~\$3.4B by 2018, generating \$14B from 2013-2018 with a CAGR of 21%
- Comparison: data implies NoSQL market ~\$895M
 - MySQL in 2011 ~\$100M



MongoDB Overview

- Document oriented, not table/row oriented
- Collection of binary JSON (BSON) documents
- Schemaless
- No relations or transactions native in database
- Scalable and high-performance
- Fast In-Place Updates
- Map/Reduce
- Full index support
- Servers for all major platforms
- Drivers for all major development environments
- Free and open-source, but also commercial support

Why MongoDB

- Rapid Application Prototyping
 - Schema rides as objects inside the JSON Script, not as a separate item.
 - Schemas change quite quickly in modern apps as business needs change.
 - Apps have shorter production lives and even shorter life-cycles as a result
- Scalability
 - Automatic data rebalancing across cheap, inexpensive commodity network hardware during the shard operation.
- Large amount of customers worldwide.
- Free basic edition
- Many related free software downloads

Terminology and Concepts

RDBMS		MongoDB
Database	→	Database
Table, View	→	Collection
Row	→	Document (JSON, BSON)
Column	→	Field
Index	→	Index
Join	→	Embedded Document
Foreign Key	→	Reference
Partition	→	Shard

Schema Free

- MongoDB does not need any pre-defined data schema
- Every document could have different data!

```
{name: "will",  
 eyes: "blue",  
 birthplace: "NY",  
 aliases: ["bill", "la  
 ciacco"],  
 loc: [32.7, 63.4],  
 boss: "ben"}
```

```
{name: "jeff",  
 eyes: "blue",  
 loc: [40.7, 73.4],  
 boss: "ben"}
```

```
{name: "brendan",  
 aliases: ["el diablo"]}
```

```
{name: "ben",  
 hat: "yes"}
```

```
{name: "matt",  
 pizza: "DiGiorno",  
 height: 72,  
 loc: [44.6, 71.3]}
```



CRUD

- Create
 - `db.collection.insert(<document>)`
 - `db.collection.save(<document>)`
 - `db.collection.update(<query>, <update>, { upsert: true })`
- Read
 - `db.collection.find(<query>, <projection>)`
 - `db.collection.findOne(<query>, <projection>)`
- Update
 - `db.collection.update(<query>, <update>, <options>)`
- Delete
 - `db.collection.remove(<query>, <justOne>)`

Example-Querying

```
> db.users.findOne({age:39})  
> db.users.find({'last': 'Doe'})  
// retrieve all users order by last_name:  
> db.users.find({}).sort({last: 1});
```

```
- {"_id" : ObjectId("5114e0bd42..."),  
 "first" : "John",  
 "last" : "Doe",  
 "age" : 39,  
 "interests" : [  
     "Reading",  
     "Mountain Biking ]  
 "favorites": {  
     "color": "Blue",  
     "sport": "Soccer"} }
```

Advanced Querying

```
{ name: "Joe", address: { city: "San Francisco", state: "CA" } , likes: [ 'scuba', 'math', 'literature' ] }

// field in sub-document:
db.persons.find( { "address.state" : "CA" } )

// find in array:
db.persons.find( { likes : "math" } )

// regular expressions:
db.persons.find( { name : /acme.*corp/i } );

// javascript where clause:
db.persons.find("this.name != 'Joe'");

// check for existence of field:
db.persons.find( { address : { $exists : true } } );
```

Insert & Update

- Supports bulk inserts
- Default saves are upserts
- In place updating
- Atomic transactions for single documents
- Server side JavaScript execution

Insert & Update Example

```
> db.user.insert({  
  first: "John",  
  last : "Doe",  
  age: 39  
})
```

```
> db.user.update(  
  {"_id" : ObjectId("51...")},  
  {  
    $set: {  
      age: 40,  
      salary: 7000}  
  })
```

```
> db.user.remove({  
  "first": /^J/  
})
```

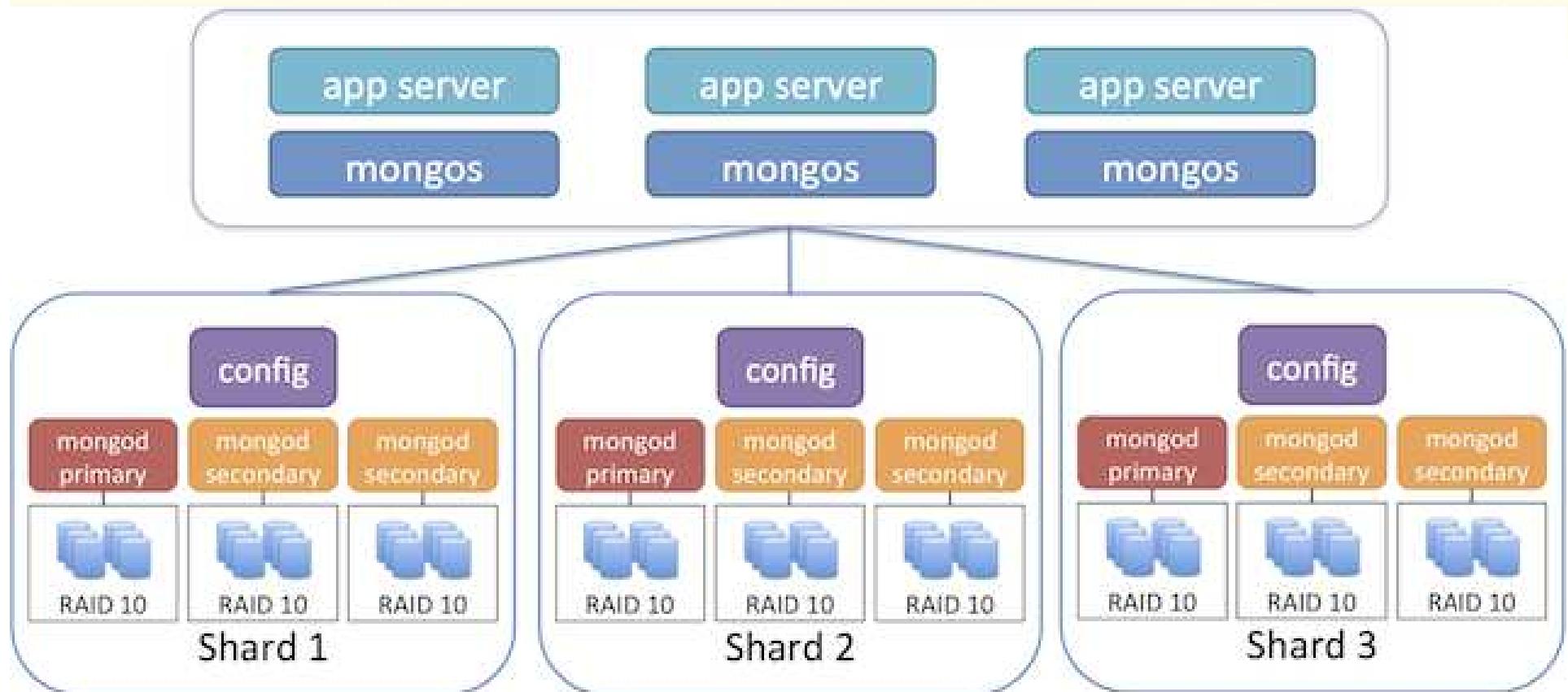
Indexes

- Unique index on primary key (`_id` field)
- Create index from application code (`ensureIndex`)
- Index on embedded documents and fields
- Index on array fields (*multikey index*)
- Geospatial index
- No native full text indexing

Auto-sharding

- Partitions data across *shards*
- Any BSON document resides on only one shard
- Increases write capacity and total data size
- Data automatically distributed
- Sharding transparent to application layer
- Partitioning based on client-defined shard key
- Good shard keys are highly distributed in value and write operations
- Sharding requires config servers (minimal 3) to maintain metadata

Mongo Sharding



Lab : Install MongoDB

- Download mongoDB from
<http://www.mongodb.org/downloads>
(msi for windows)
- Create a new folder *c:\mongodb*, and install mongodb to the folder
- Create a new folder *data* inside the folder *mongodb* to store data

Lab : Start MongoDB

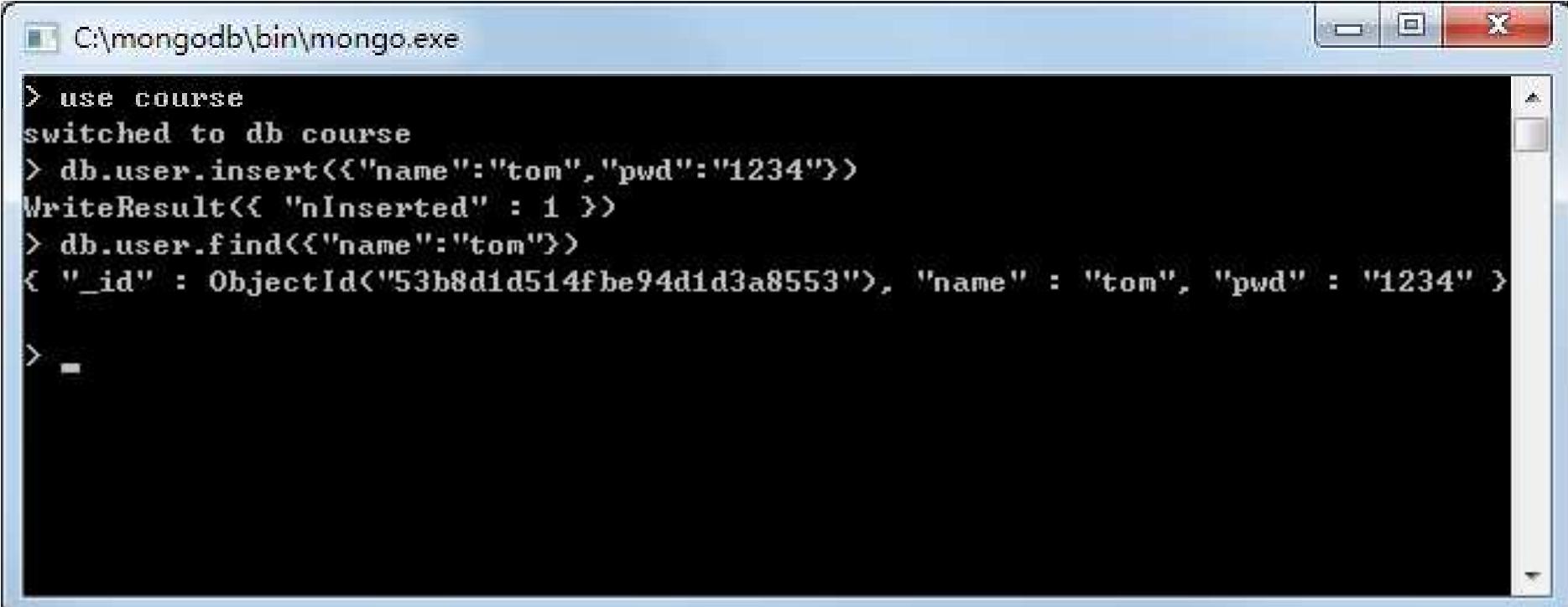
- Open a cmd, goto folder `c:\mongodb\bin`
- Start monfoDB by typing
`mongod.exe --dbpath c:\mongodb\data`
- Open another cmd, goto folder `c:\mongodb\bin` ,
and type `mongo.exe` to open a management
window.



Lab: Add a User to MongoDB

- Go to the management window and type:

```
> use course
> db.user.insert({name:"tom",pwd:"1234"})
> db.user.find({name:"tom"})
```



```
C:\mongodb\bin\mongo.exe

> use course
switched to db course
> db.user.insert({name:"tom",pwd:"1234"})
WriteResult({ "nInserted" : 1 })
> db.user.find({name:"tom"})
{ "_id" : ObjectId("53b8d1d514fbe94d1d3a8553"), "name" : "tom", "pwd" : "1234" }

>
```

REST

What is REST?

- REST stands for *Representational State Transfer*.
- It describes an architecture for distributed information systems
- First described in the 2000 doctoral dissertation “Architectural Styles and the Design of Network-based Software Architectures” by [Roy Fielding](#).
- It’s a description of how the Web works and why it works well

REST is about Architecture

- Software architecture
 - An abstraction of the elements, configurations, constraints, principles, and guidelines that govern a system's design and evolution.
- Software architectural style
 - A set of constraints that restrict a software architecture.

RESTful Examples

- Public services with RESTful APIs:
 - Twitter, Netflix, Dropbox, Flickr, Amazon S3, ...
- Products or tools with RESTful APIs
 - Glassfish Application Server Admin, Selenium WebDriver, ...
- RESTful Frameworks
 - Jersey (JAX-RS), Restlet, Restify, APEX RESTful Services, ...

Why REST?

- Scalable
- Human and machine usable
- Language agnostic
- Globally accessible resources
- Intuitively understandable URLs, resources, and actions.

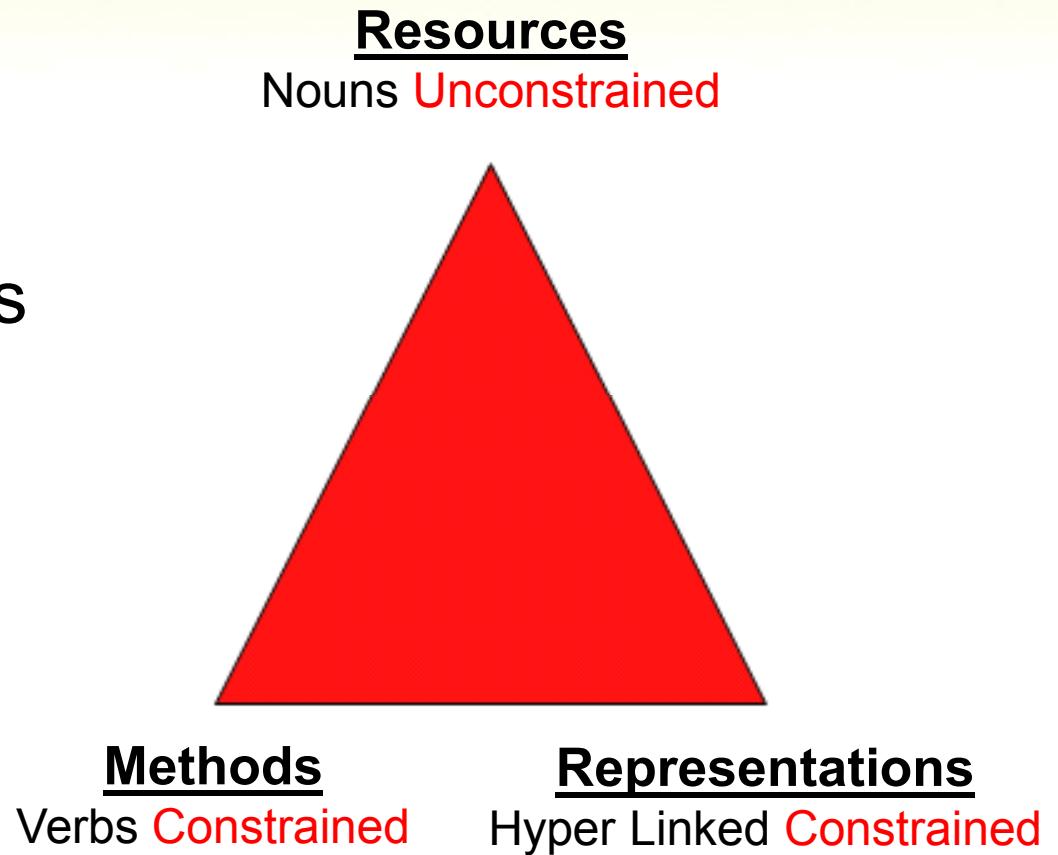
REST Properties

- Client/server model – request/response
- Stateless: no memory of prior communications
- Cacheable
- Layered system: may use intermediary servers
- Code on demand (optional)
- Uniform interface
 - Request response style operations on named resources through self descriptive representations where state changes are via hyperlinks

Uniform Interface

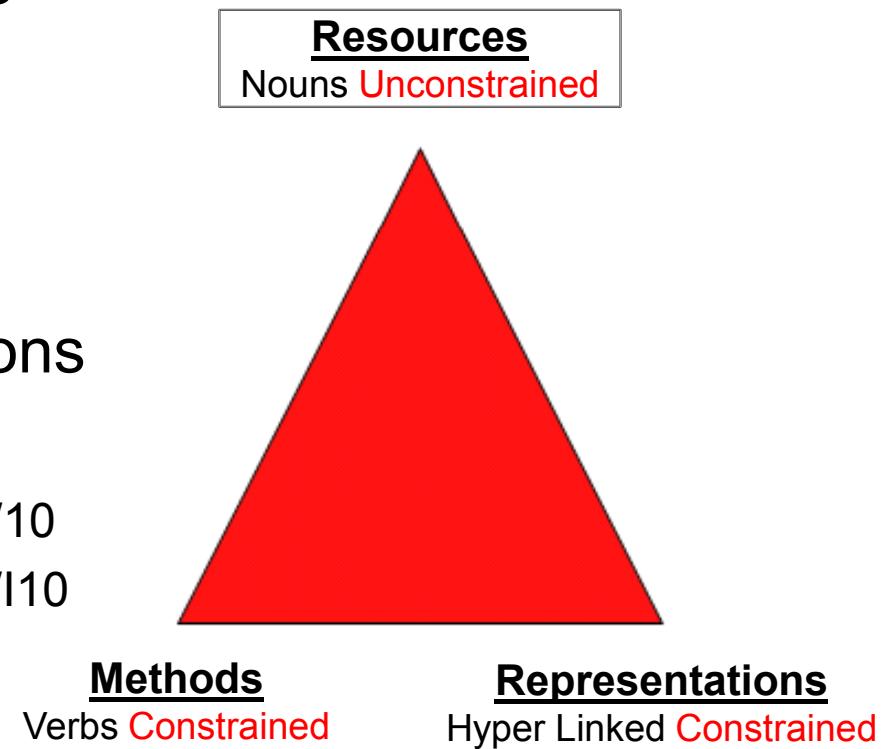
The REST Triangle:

- Resources
- Methods
- Representations



Uniform Interfaces - Resources

- Key abstract concept
- Identified by a URI
- Distinct from underlying storage
- Semantics fixed
- Value may change over time
- Can have multiple URIs
- Can have multiple representations
- Examples:
 - <http://example.org/NewOrleans/traffic/10>
 - <http://example.org/traffic/NewOrleans/l10>
 - <http://foo.com/store/orders>

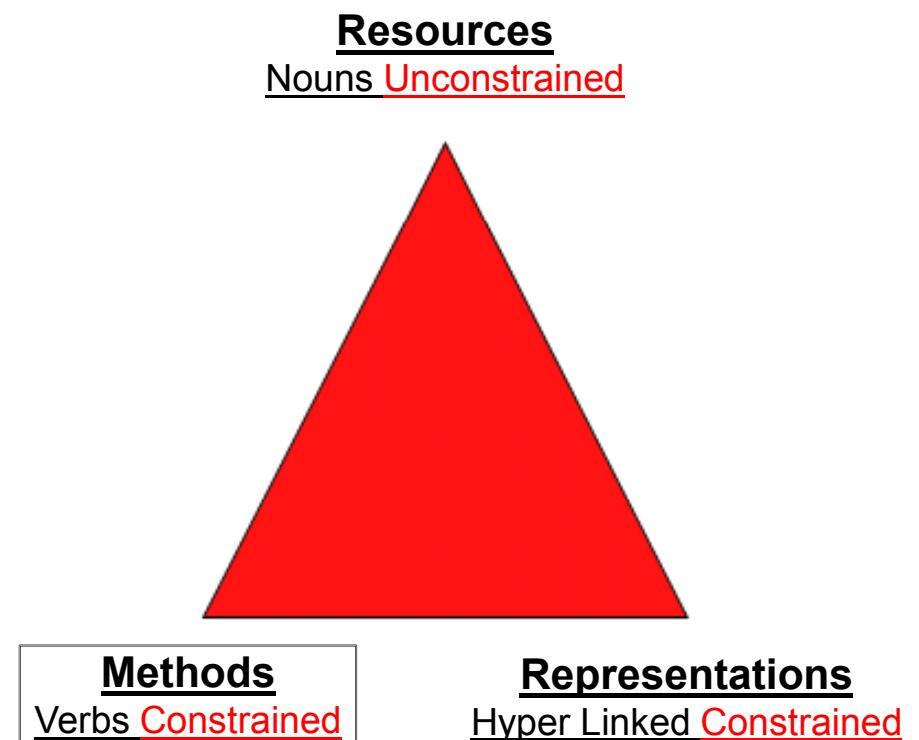


Resource Modeling - URLs

- Human readable (not necessary but it helps)
- Tends to form a hierarchy
- Use the query part appropriately
 - Use to search, filter, or possibly specify a mode
 - Identification of the resource is better in the path
 - (preferred) `http://example.com/orders/100234`
 - `http://example.com/orders?id=100234`
- Don't make them verbs!
 - (bad) `http://example.com/accounts/addaccount`

Uniform Interface - Methods

- HTTP method Applying to the resource
 - **GET** retrieve
 - **PUT** update (or create)
 - **DELETE** delete
 - **POST** create sub resource
- Response codes: HTTP
 - 1xx, 2xx, 3xx, 4xx, 5xx



REST Methods

- HTTP methods POST, GET, PUT, and DELETE are often compared with Create, Read, Update, and Delete (CRUD) operations associated with database functions:

REST	CRUD	SQL
GET	Read	SELECT
POST	Create	INSERT
PUT	Update or Create	UPDATE or INSERT
DELETE	Delete	DELETE

Design REST Methods

Method	Requirement
GET	Retrieve a resource. No modification should be done. No side effects allowed. Keep it safe.
POST	<i>Create or update a resource.</i> For non-safe, non-repeatable changes
PUT	<i>Update a resource.</i> Keep it repeatable with same results (idempotent).
DELETE	<i>Remove a resource.</i> Keep it repeatable with same results (idempotent)

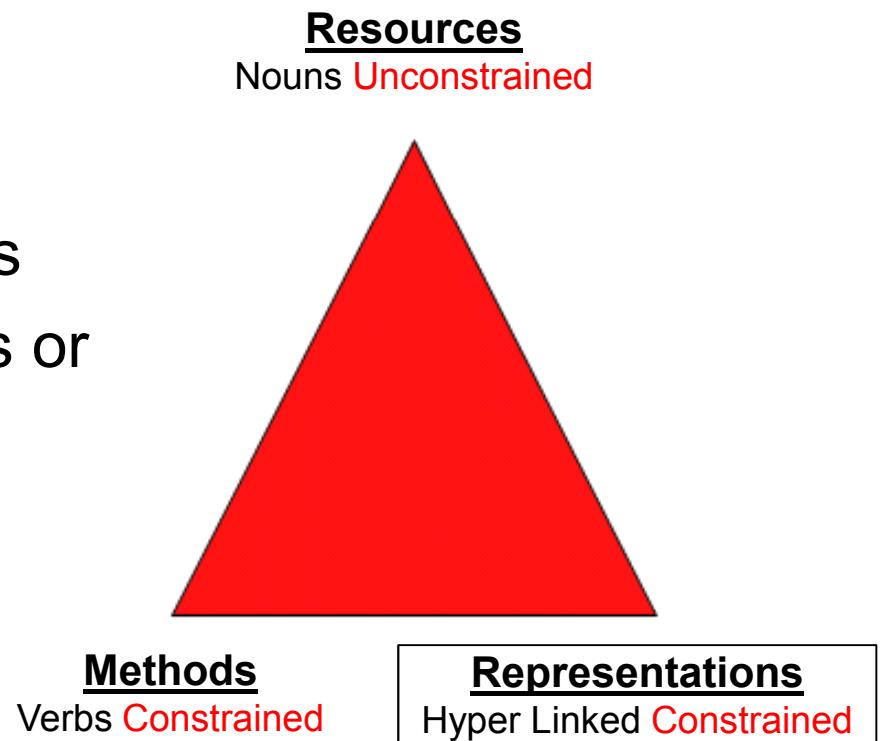
Are They REST ?

- How do these actions map to HTTP and Web pages?

Action	HTTP Method	URL
Create Message	POST	http://bbs/msg.php?m=create
View Message	GET	http://bbs/msg.php?m=view&id=12
Update Message	POST	http://bbs/msg.php?m=update&Id=12
Delete Message	GET	http://bbs/msg.php?m=delete&id=12

Uniform Interface - Representations

- Self-descriptive
- media type (Content-Type)
 - text/html
 - application/json
- Includes metadata
- Understood by all components
- May be for humans, machines or both
- Negotiated



Representation Example

Representation based on request type

- Request for XML output

```
GET /resources/forte HTTP/1.1  
Host: example.com  
Accept: application/xml
```

- Request for HTML output

```
GET /resources/forte HTTP/1.1  
Host: example.com  
Accept: text/html
```

- Could be any format you wish to support.

A Example BBS System

- What message actions are required?
 - Create message
 - View message
 - View message for update
 - Delete message

RESTful Methods

- How do the message actions map in REST?

Action	HTTP Method	URL
Create Message	POST	http://bbs/messages
View Message	GET	http://bbs/messages/12
Update Message	PUT	http://bbs/messages/12
Delete Message	DELETE	http://bbs/messages/12

Common Pattern for Methods

messages /

- GET - Retrieves list of all messages.
- POST - Create a new message .

messages/{mesno}/

- GET - Retrieves details for a specific message .
- PUT - Updates the specific message .
- DELETE - Deletes the message .

Designing RESTful Services

- Key Principles
 - Everything gets a unique URI
 - Link resources together
 - Use standard methods
 - Resources have multiple representations
 - Communicate statelessly

REST Summary

- REST is an architectural style that
 - Focuses on resources
 - Minimizes verbs
 - Leverages existing Web server capabilities
- REST encourages
 - Loose coupling
 - Separation of representation and process
 - Separation of security and logging from application
 - Maximum exploitation of Web servers and HTTP

Cloud Apps Development

- Cloud software development
 - Design and develop an application for the cloud
 - Especially for the public PaaS cloud platforms
- Typical steps in cloud software development
 - Choose a development stack of technologies
 - Choose a cloud platform + services
 - Design the application for the cloud
 - Develop the application using the cloud APIs
 - Deploy and run the application in the cloud

Example: WorkTitle

- A team cooperation tool
- Web-based cross-platforms, visit from browsers
- Single page application. Interface and feeling of native apps, such as drag and drop
- Instant message, every modification in one client will be automatically refreshed in other clients.
- Provide stable service
- Already more than 100 thousands users

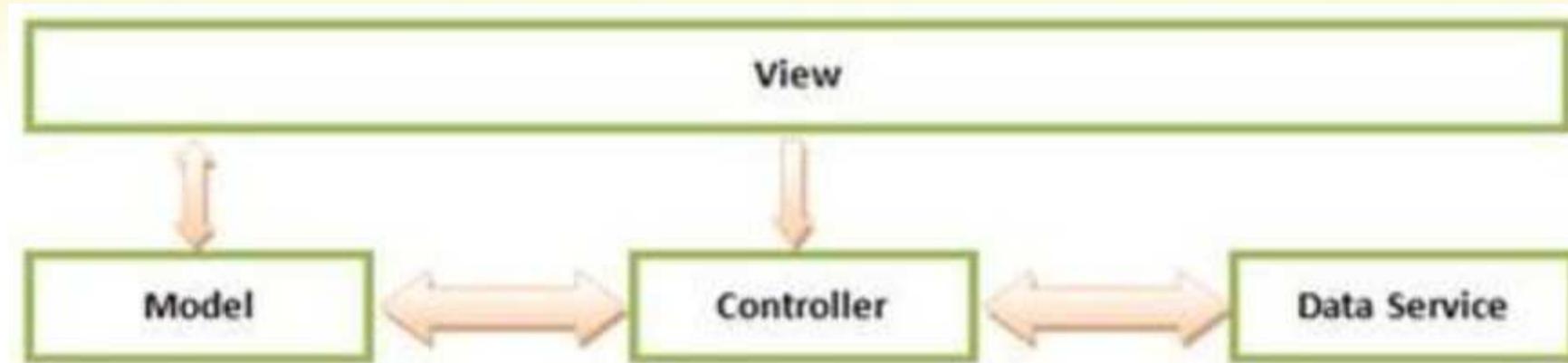
SPA of WorkTitle

The screenshot displays the Worktile product management interface, illustrating a Single Page Application (SPA) structure. The main area is divided into four vertical columns:

- 产品设计 (Product Design):** Contains tasks like "UI设计优化" (UI design optimization), "收费系统、策略" (Fee system, strategy), "换肤和背景" (Skin and background), "设计事件产品原型" (Design event product prototype), and "任务归档" (Task archiving). A "新建任务" (New task) button is located at the bottom.
- 通用 (General):** Contains tasks like "活动时间流" (Activity timeline), "Dashboard优化" (Dashboard optimization), "后台API编写" (Backend API development), and "样式优化" (Style optimization). A "新建任务" (New task) button is located at the bottom.
- 任务模块 (Task Module):** Contains tasks like "拖动成员分配任务" (Drag member to assign task), "日历, 包括全局日历和项目日历, 任务在日历的创建" (Calendar, including global calendar and project calendar, tasks are created in the calendar), "安全性" (Security), and "任务筛选" (Task filtering). A "新建任务" (New task) button is located at the bottom.
- 移动端 (Mobile End):** Contains tasks like "Andriod端" (Android end), "iPad端" (iPad end), and "Android Pad端" (Android Pad end). A "新建任务" (New task) button is located at the bottom.

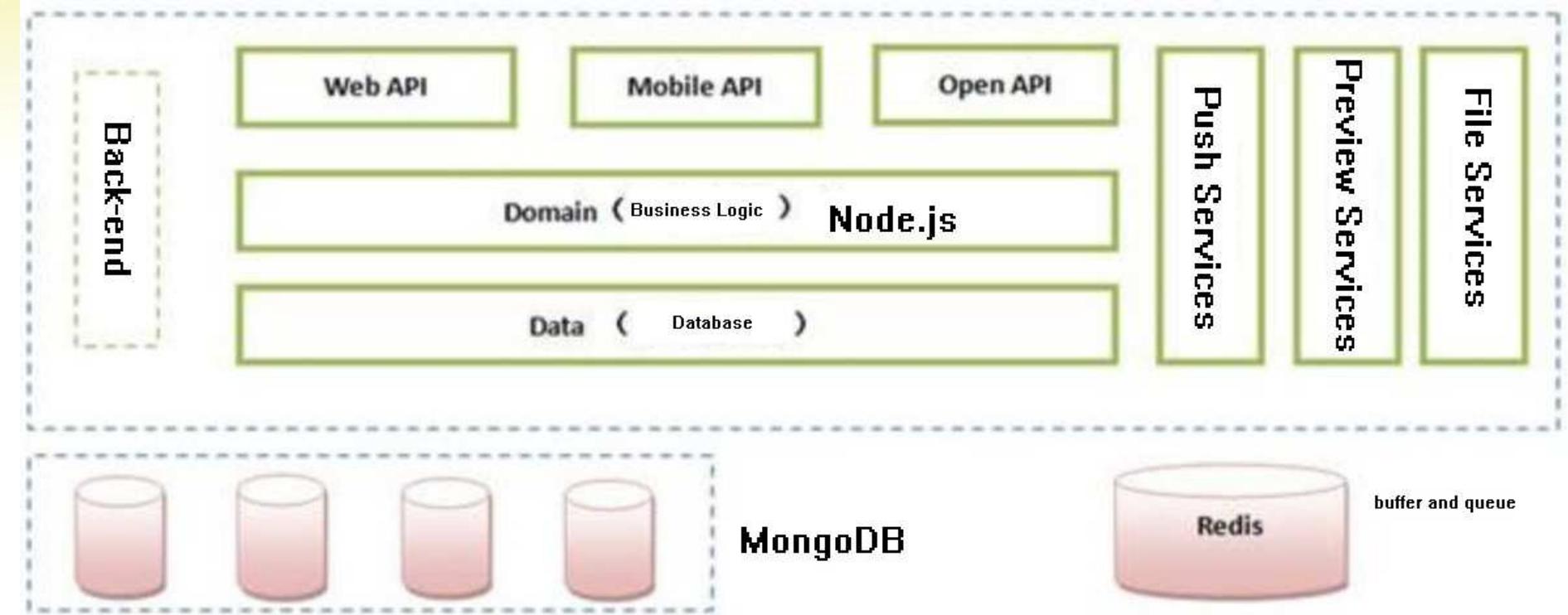
At the top right, there is a user profile for "Jaden" and a sidebar on the right labeled "文" (Wen) which includes icons for Task, Calendar, File, Topic, Document, and Report.

Front-side: AngularJs



- Two-way data binding
- Declarative semantic tags
- Modules
- Dependency injection

Back-End of WorkTitle

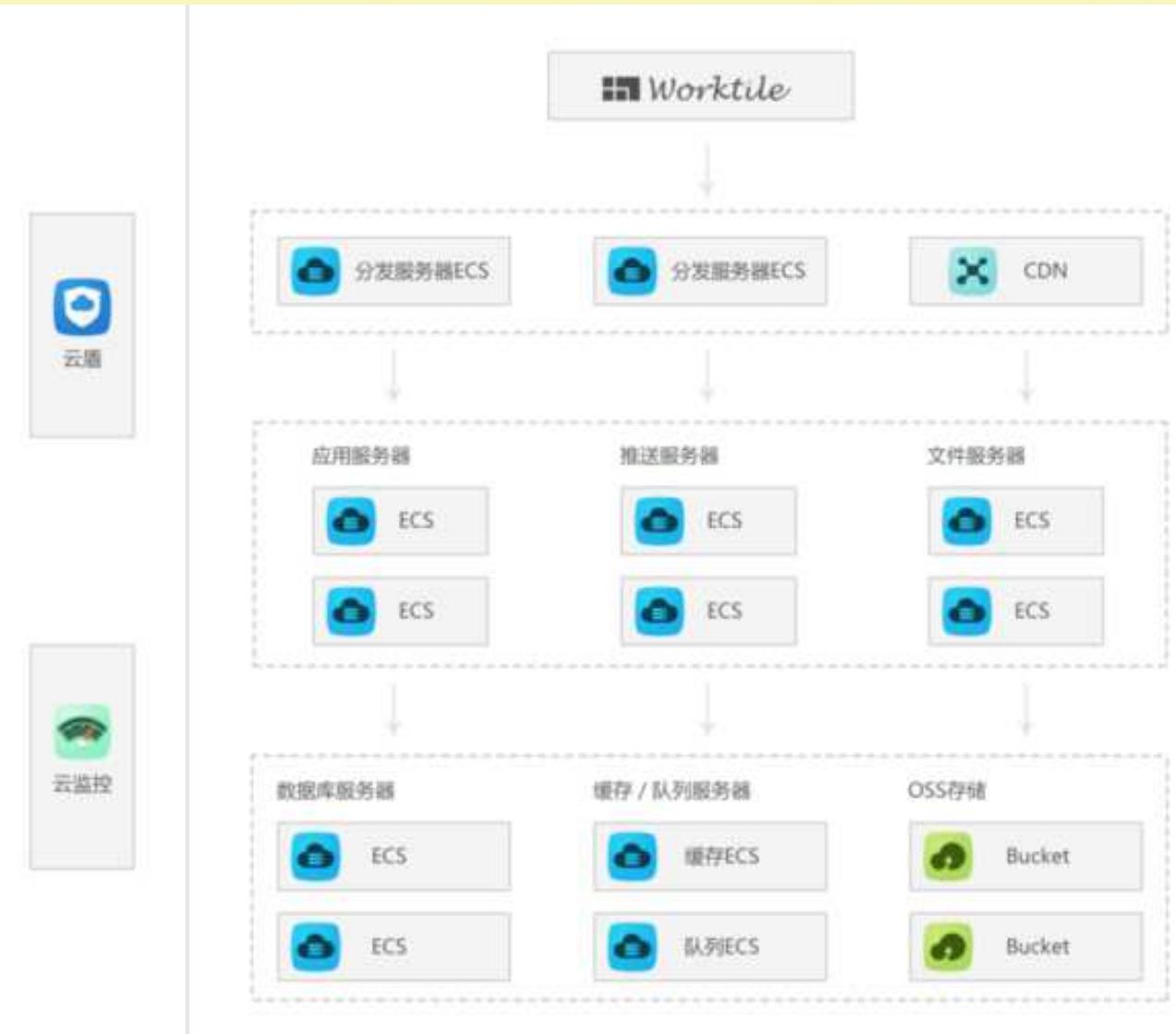


- Preview service: Microsoft Office Web App
- Push service: ejabberd (core technology)

Long Polling vs Short Polling vs Websocket

- Short polling : uses a timer and asks the server if new data is available
- Long Polling: uses an event based algorithm, more realtime.
- Extensible Messaging and Presence Protocol (**XMPP**) is a near-real-time communications protocol for message exchange.
- **ejabberd** is the world's most popular XMPP server. It can

WorkTitle on Aliyun



Architecture Techniques

OpenStack/Docker

What is Virtualization?

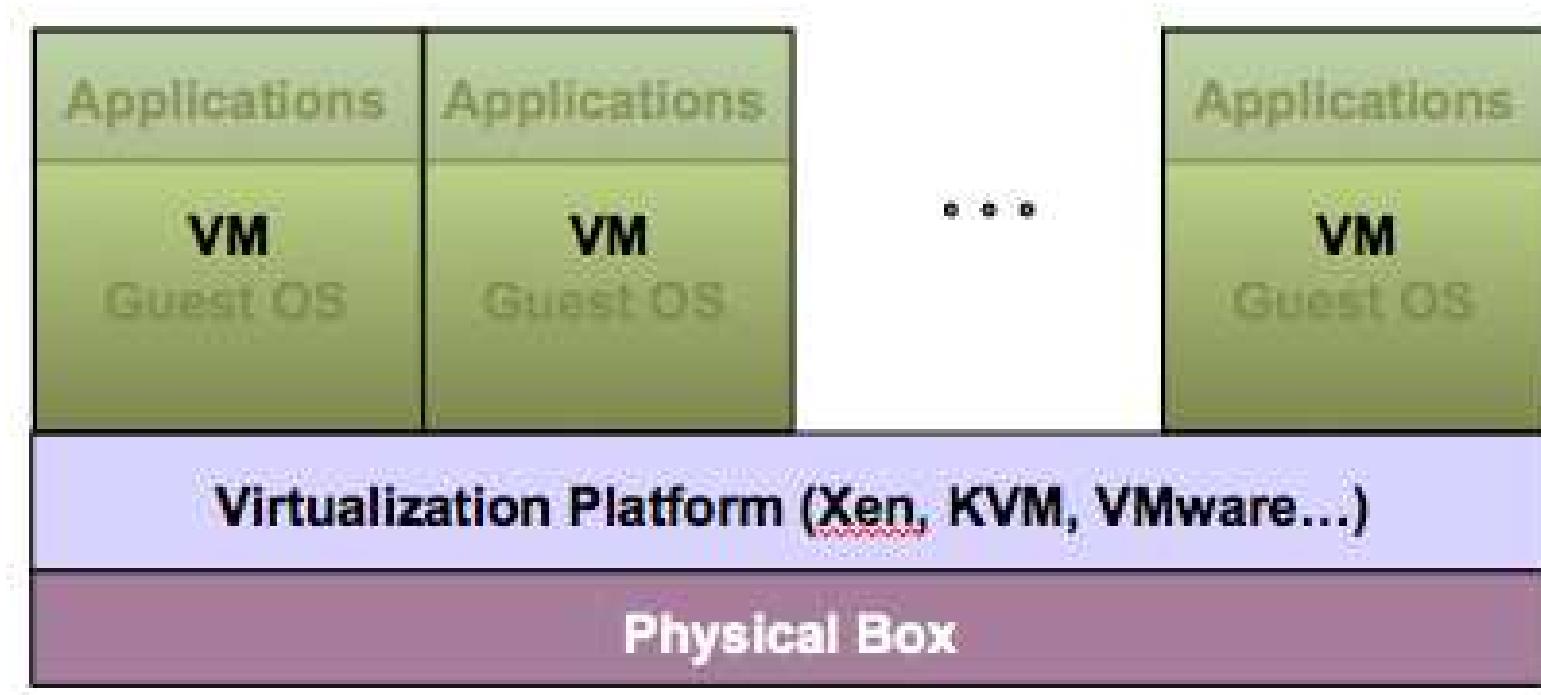
- Virtualization
 - Running several virtual machines (virtual computers) inside a single physical machine
 - Supported by special software called *hypervisor*
 - Uses resources more efficiently
 - E.g. 12 GB physical RAM is shared to 6 virtual machines with 4 GB shared RAM each
 - Most applications use 1-5% of the CPU
 - A single shared CPU can serve thousands of users
 - Reduces costs due to better utilization

Virtualization Definitions

- Virtualization is the ability to run multiple operating systems on a single physical system and share the underlying hardware resources
- Virtualization is used to improve IT throughput and costs by using physical resources as a pool from which virtual resources can be allocated.
- Virtual machine (VM) software is a program that emulates a physical machine
- A VM needs to act *exactly* like its physical machine
- A VM instance is simply a file that represents an actual machine and its state

Virtualization Architecture

- A Virtual machine (VM) is an isolated runtime environment (guest OS and applications)
- Multiple virtual systems (VMs) can run on a single physical system



OpenStack & Docker

Open Frameworks



PaaS
Application Centric



IaaS
Resource
Centric



OPENSTACK

OpenStack

- An open source cloud computing framework
- It is primarily used to enable **infrastructure as a service**
- OpenStack began in 2010 as a joint project of Rackspace Hosting and NASA. Currently, it is managed by the OpenStack Foundation

OpenStack Mission

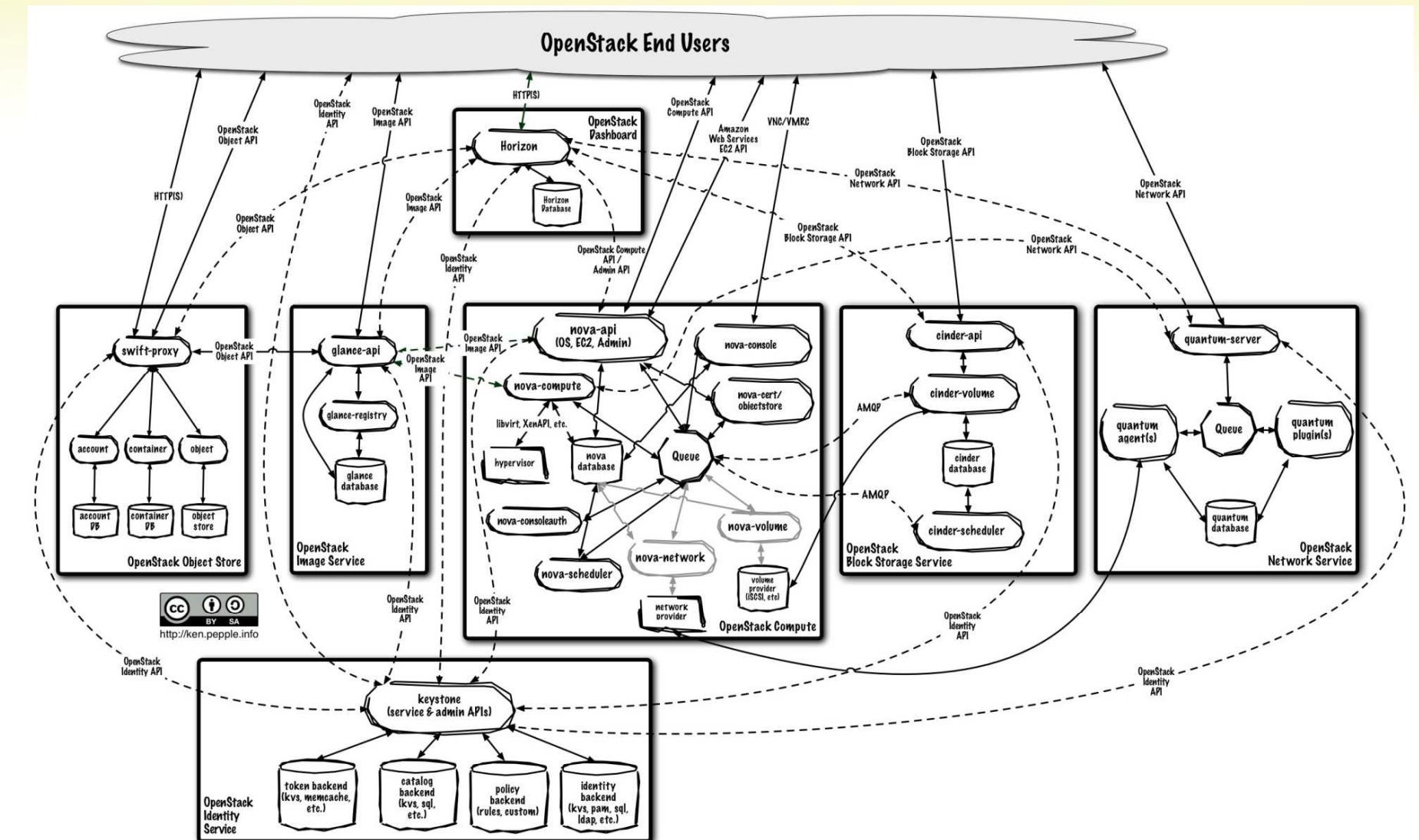
To produce the
**ubiquitous open source cloud computing
platform**

that will meet the needs of public and private
clouds regardless of size, by being simple to
implement and massively scalable.

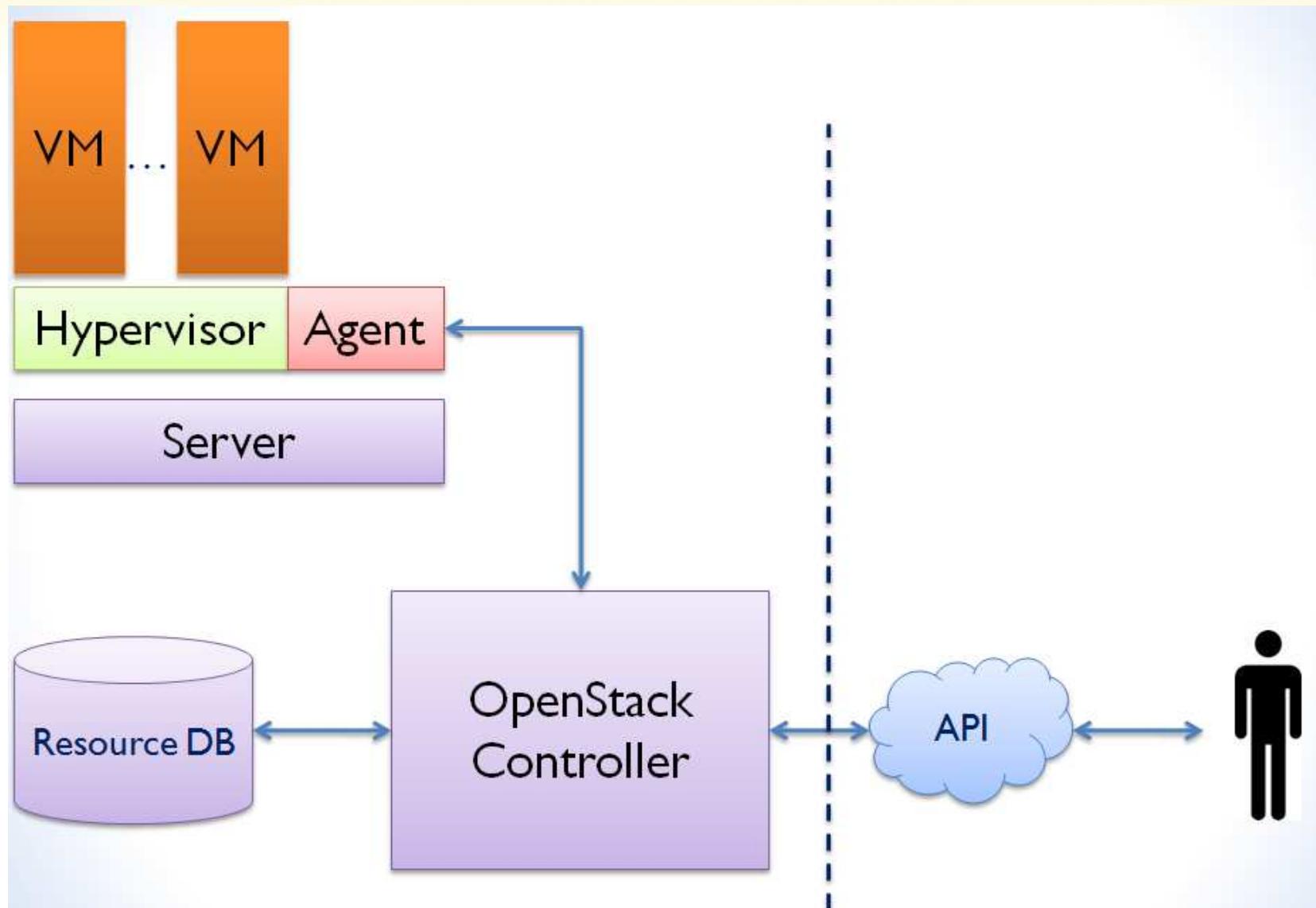
Trends and Themes of OpenStack

- OpenStack Powered Planet – delivering on cloud interoperability and federation across vendors and geographies
- Applications on OpenStack – new community app catalog and first app guide provide more support for developers targeting OpenStack clouds
- Platform for the Next 10 Years – OpenStack gives users the ability to integrate and test new technologies like containers as they emerge

OpenStack is Complicated



But, Basic of OpenStack



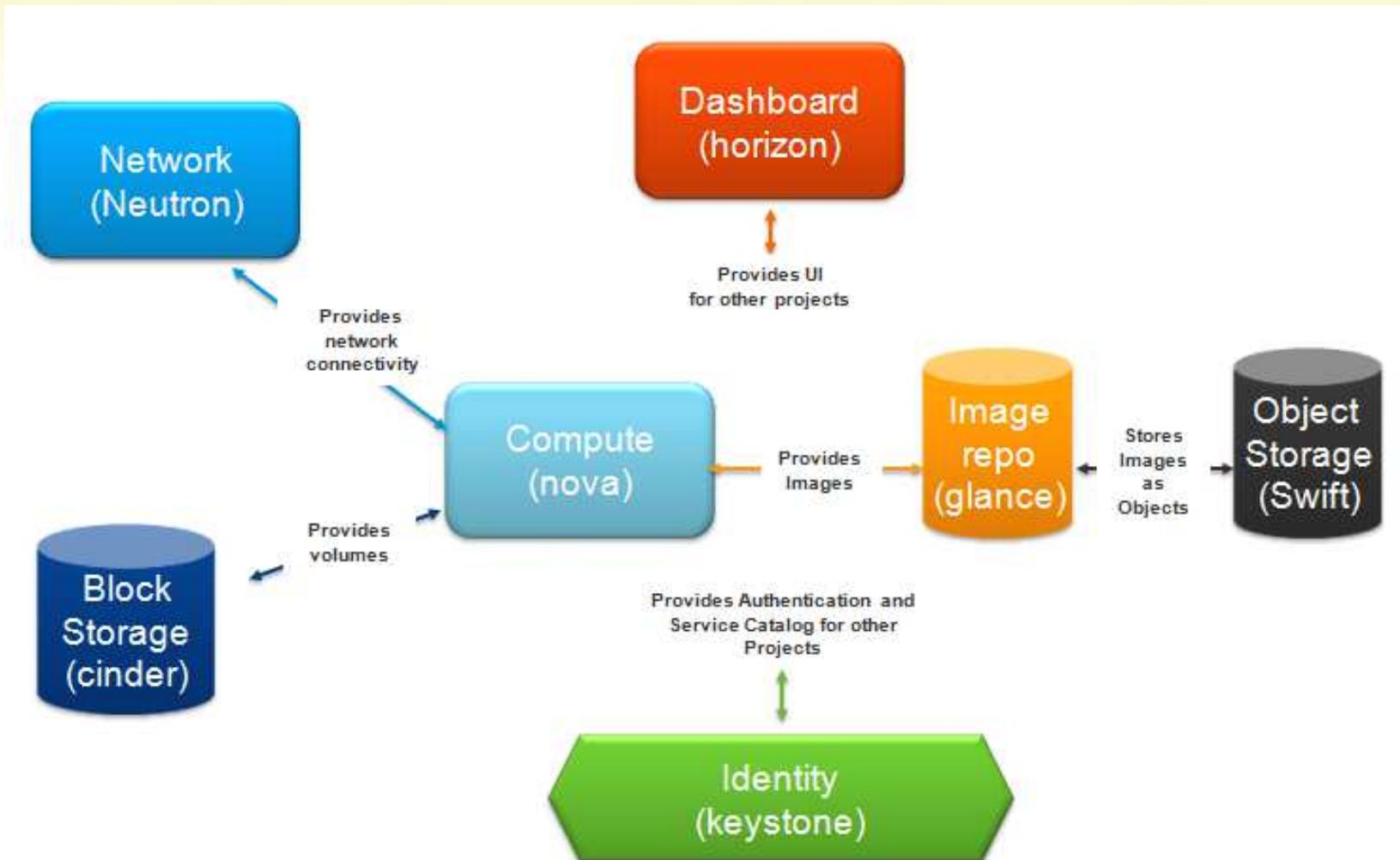
VMs as OpenStack Cloud Resources

- VMs run on top of hypervisors, and Agent process runs on physical machine beside hypervisor.
- User requests VM (resource) from controller, controller finds available resources in pool, contacts appropriate Agent
- Agent interprets command, instructs hypervisor to boot VM with user provided OS image
- Openstack returns handle to the user

Components of OpenStack

- Compute (Nova)
- Object Storage (Swift)
- Block Storage (Cinder)
- Networking (Neutron)
- Dashboard (Horizon)
- Identity Service (Keystone)
- Telemetry (Ceilometer)
- Orchestration (Heat)
- Database (Trove)
- Bare Metal Provisioning (Ironic)
- Multiple Tenant Cloud Messaging (Zaqar)
- Elastic Map Reduce (Sahara)

OpenStack Main Components



Compute (Nova)

- Provision and manages virtual machines. It is a major part of an IaaS system.
 - **Nova-api service**: Accepts and responds to end user compute API calls.
 - **Nova-compute service**: A worker daemon that creates and terminates virtual machine instances through hypervisor APIs.
 - **Nova-scheduler**: Takes a virtual machine instance request from the queue and determines on which compute server host it runs.
 - **Rabbit MQ Server**: A central hub for passing messages between daemons

Storage (Swift)

- Object Storage:
 - Object Storage provides a fully distributed, API-accessible storage platform that can be integrated directly into applications or used for backup, archiving and data retention.
- Block Storage:
 - Block Storage allows block of storages to be exposed and connected to compute instances for expanded storage, better performance and integration with enterprise storage platforms

Networking(Neutron)

- Neutron is an OpenStack project to provide "networking as a service" between interface devices.
- It accepts and routes API requests to the appropriate OpenStack Networking plug-in devices for action.

Dashboard (Horizon)

- Dashboard provides a web based user interface to OpenStack services including Nova, Swift, Keystone, etc.
- The OpenStack dashboard provides administrators and users a graphical interface to access, provision and automate cloud-based resources.

Sample Dashboard

The screenshot shows the OpenStack Dashboard interface. On the left, there's a sidebar with navigation links: Project (selected), CURRENT PROJECT stackato, Manage Compute (Overview, Instances, Volumes), Images & Snapshots, Access & Security, Object Store (Containers). The main content area is titled "Instances" and displays a table of running instances. The table has columns: Instance Name, IP Address, Size, Keypair, Status, Task, Power State, and Actions. There are five entries:

Instance Name	IP Address	Size	Keypair	Status	Task	Power State	Actions
stackato-266.data	10.0.3.2 208.75.128.146	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot
stackato-266.dean	10.0.2.18 208.75.128.145	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot
stackato-266.dean	10.0.5.14 208.75.128.144	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot
stackato-266.dean	10.0.4.10 208.75.128.143	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot
stackato-266-vm1	10.0.3.6 208.75.128.142	m1.medium 4GB RAM 2 VCPU 40GB Disk	-	Active	None	Running	Create Snapshot

A green success message box is visible in the top right corner: "Success: IP address 208.75.128.146 associated." Below the table, it says "Displaying 5 items".

Identity Service (Keystone)

- Keystone is the identity service used by OpenStack for user authentication.
- It performs the following functions:
 - Tracking users and their permissions.
 - Providing a catalog of available services with their API endpoints.

Keystone and Dashboard

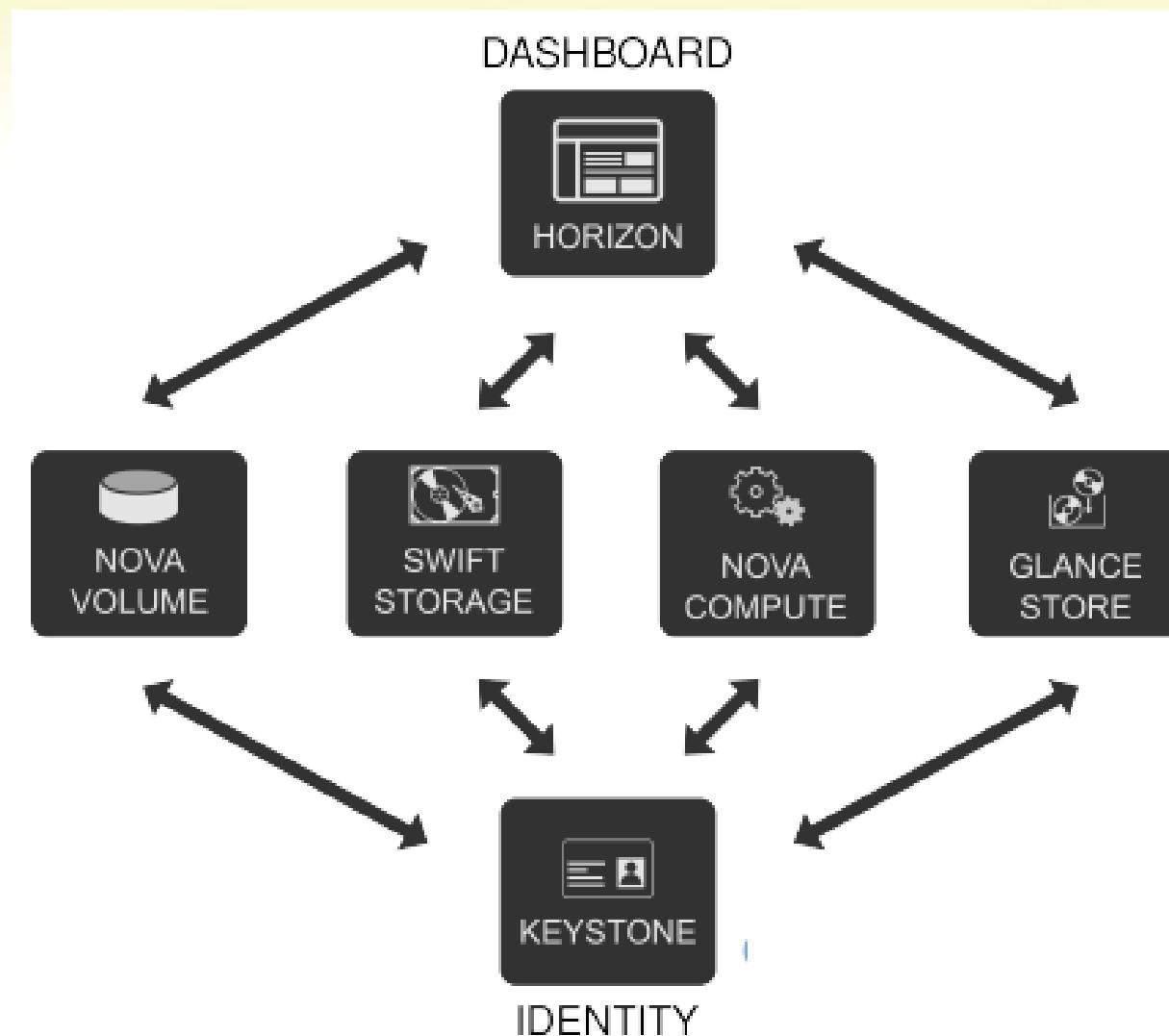


Image Service (Glance)

- The OpenStack Image Service provides discovery, registration and delivery services for disk and server images.
 - glance-api: Accepts Image API calls for image discovery, retrieval, and storage.
 - glance-registry: Stores, processes, and retrieves metadata about images.
 - Storage repository for image files: Various repository types are supported.

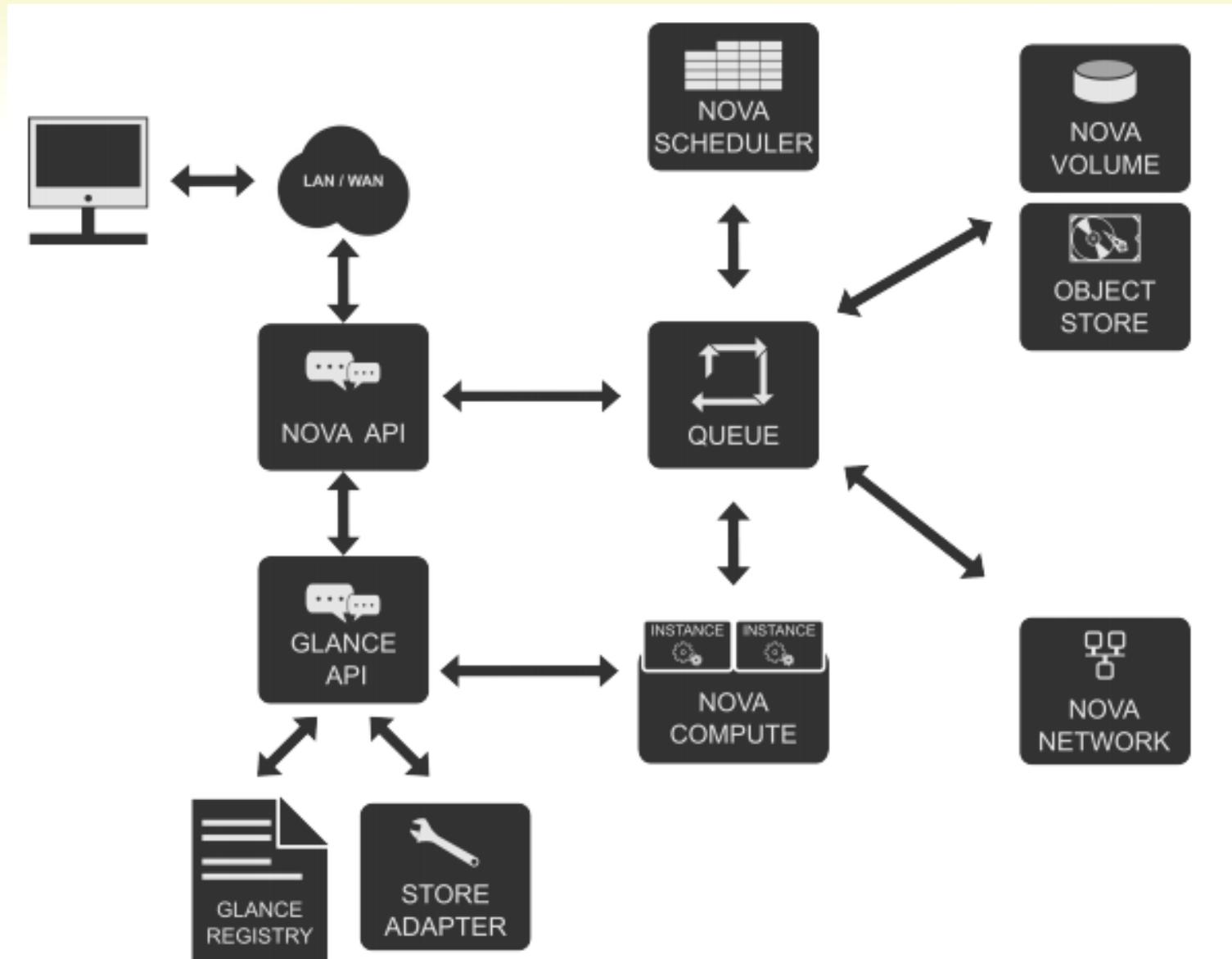
Telemetry (Ceilometer)

- The OpenStack Telemetry service aggregates usage and performance data across the services deployed in an OpenStack cloud.

Orchestration (Heat)

- OpenStack Heat program is to create a human- and machine-accessible service for managing the entire lifecycle of infrastructure and applications within OpenStack clouds.

Architecture of OpenStack



Use Cases of Openstack:

- IaaS compute platform
- Processing big data with tools like Hadoop
- Scaling compute up and down to meet demand for web resources and applications
- High-performance computing (HPC) environments processing diverse and intensive workloads
- Different users of OpenStack are AT&T, HP cloud services, PayPal, Rackspace, Sony online gaming systems, Yahoo, Wikimedia labs, Intel, NASA etc...

Walmart is powered by 140,000+ OpenStack Cores



A man with a beard and a red turban is speaking on stage. To his right is a large blue slide with white text and graphics. The slide features a world map with yellow highlights on North America, South America, Europe, and Australia. Key statistics include:

- 11 eCommerce Websites
- +21% Fiscal Q3 ending 10/14
- 1.5B+ Pageviews Thanksgiving Day - Cyber Monday

"Last Thanksgiving, we did 1.5 billion page views...November and December are the two months where we have to satisfy all the needs of all the customers, and that's what we did – we built about 140,000 cores"

-Amandeep Singh Juneja, WalmartLabs

PayPal' s transactions powered by OpenStack!



Havana
> 12k hypervisors
> 300k cores
10+ availability zones
15+ virtual private clouds
> 1.6 pb block storage
100% KVM
100% OVS

Spring 2015

During the holidays for 2014, we were running 66% of PayPal front and mid-tier on OpenStack. That number is now 100%. If you have doubts about OpenStack running in production, this is the proof point for you to say that OpenStack is doing financial transactions.” -Subbu Allamaraju, eBay Inc



DOCKER

Docker

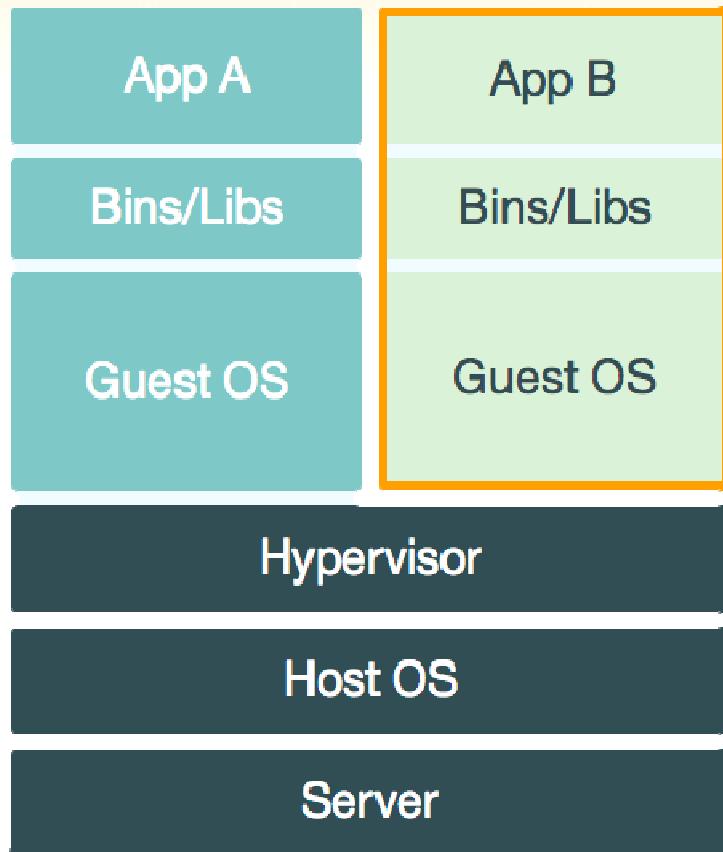
Docker is an open-source project that automates the deployment of applications inside software containers, by providing an additional layer of abstraction and automation of operating system-level virtualization on Linux.

[Source: en.wikipedia.org]

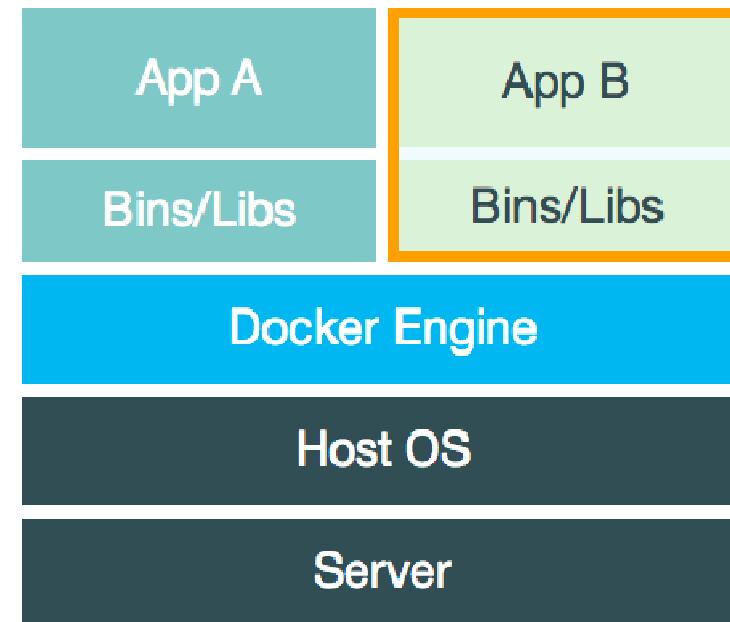
What is Docker?

- Open source platform of light weight container technology for virtualization
- Applications:
 - Are dockerized and
 - Run on Docker Containers
 - from laptops to production servers on cloud
 - Their images are shared on Docker Hub
 - Apps can be linked (node -> mongo)

Docker vs. Virtual Machine



Virtual Machine



Container

Docker vs. Virtual Machine

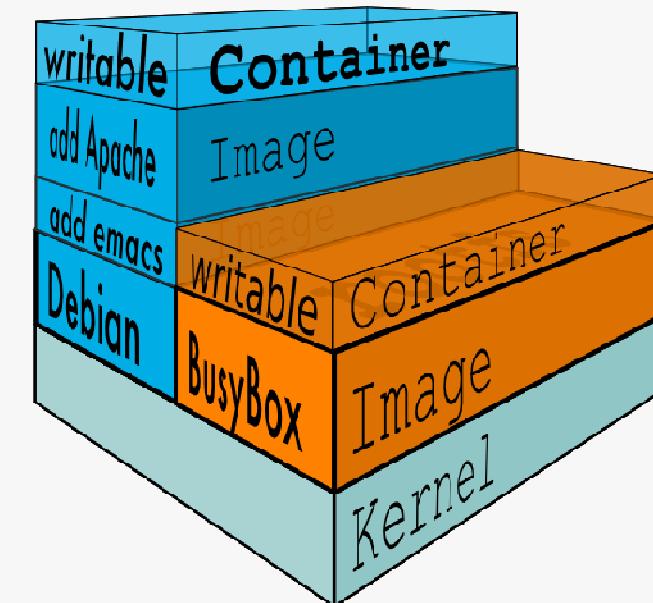
- VMs require apps to be contained within the guest OS
 - Potentially 1:1 on VM:application
- Docker sits between the app and the host OS
 - Takes place of the hypervisor
 - Runs images as an isolated process
 - Only the app and its libraries are compartmentalized instead of requiring the guest OS share that space as well

Why Linux Containers (LXC)

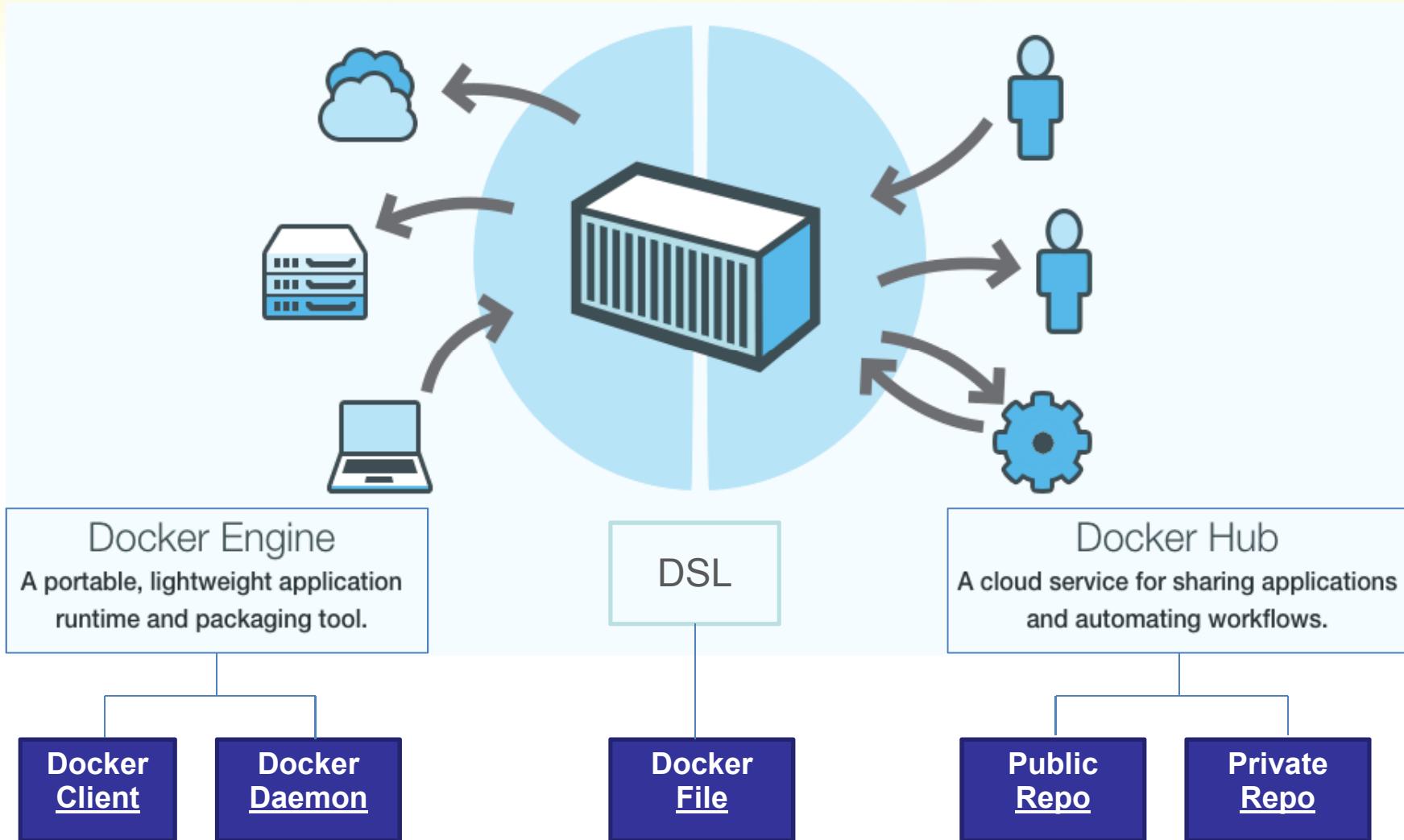
- Fast
 - Runtime performance near bare metal speeds
 - Management operations (run, stop , start, etc.) in seconds / milliseconds
- Flexible
 - Containerize a “system”
 - Containerize “application(s)”
- Lightweight
 - Just enough Operating System (JeOS)
 - Minimal per container penalty
- Inexpensive
 - Open source – free – lower TCO
 - Supported with out-of-the-box modern Linux kernel
- Ecosystem
 - Growing in popularity
 - Vibrant community & numerous 3rd party apps

Docker Technology

- libvirt: Platform Virtualization
- LXC (LinuX Containers):
Multiple isolated Linux
systems (containers) on a
single host
- Layered File System



Docker Components



Docker Parts

- Docker Engine
 - Client-server application
 - Handles running images, registries and containers
- Docker Hub
 - Public Docker registry
 - Holds a collection of existing images to be used publicly
 - Image examples: Ubuntu, Node.js, MongoDB, WordPress

Docker Hub

- Public repository of Docker images
 - <https://hub.docker.com/>
- Integrates with Github and Bitbucket
 - Operates similar to them as well
- Hosts images that can be used and contributed to by the Docker community
 - Location that Docker pulls and pushes to

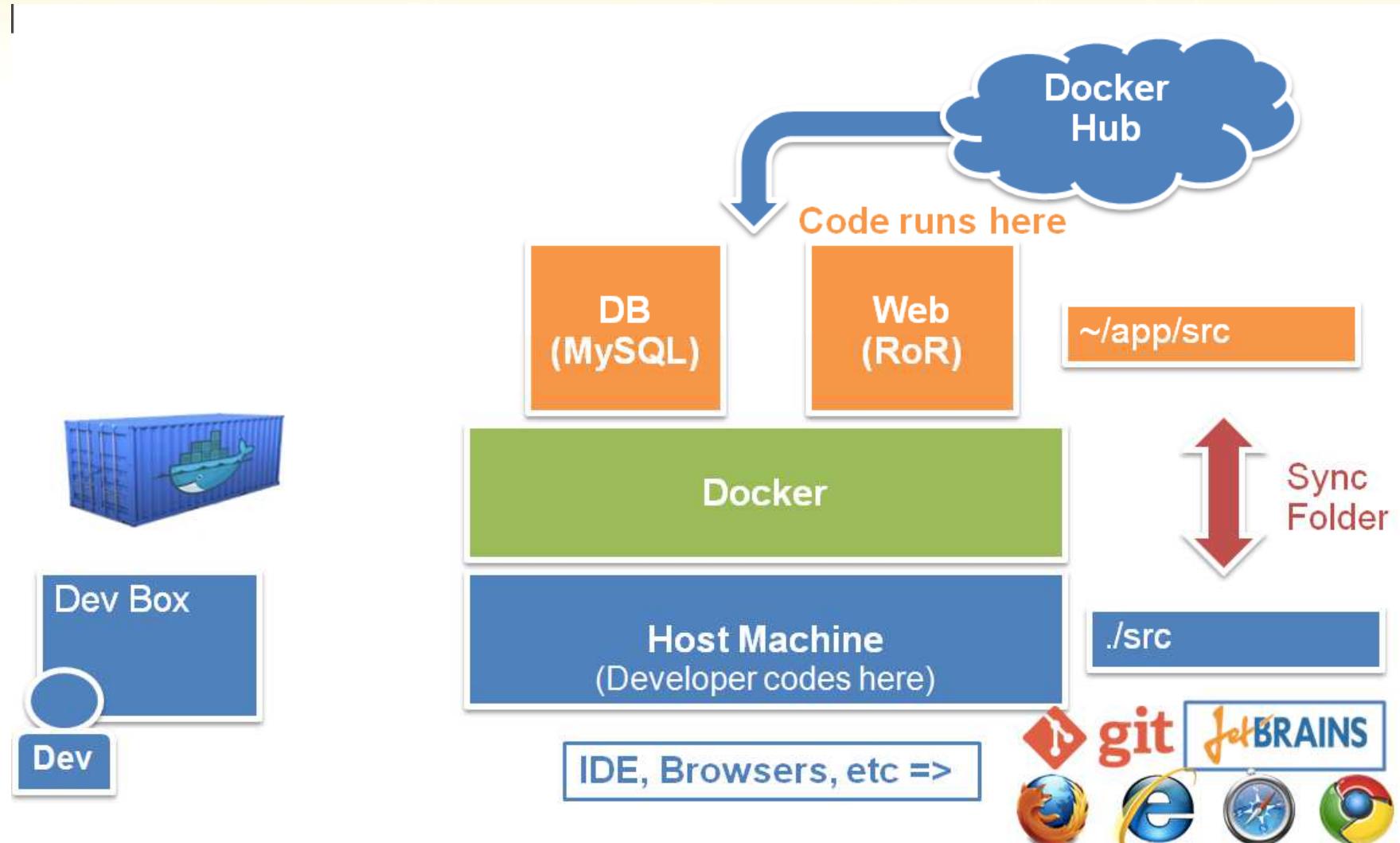
Run Platforms

- Various Linux distributions (Ubuntu, Fedora, RHEL, Centos, openSUSE, ...)
- Cloud (Amazon EC2, Google Compute Engine, Rackspace)
- Microsoft plans to integrate Docker with next release of Windows Server

Running a Web App in Docker

- Similar to any other *nix command line application
- Docker's example runs a Python Flask app:
 - `$ sudo docker run -d -P training/webapp python app.py`
 - **-P**: map required network ports inside the container to the host
 - Check results with `sudo docker ps -l`:
 - CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bc533791f3f5 training/webapp:latest python app.py 5 seconds ago Up 2 seconds 0.0.0.0:49155->5000/tcp nostalgic_morse
 - **-l**: tells docker ps to return the last container started

Development Scenario



MeanStack Use case



Open-Source Full-Stack Solution for MEAN
Applications

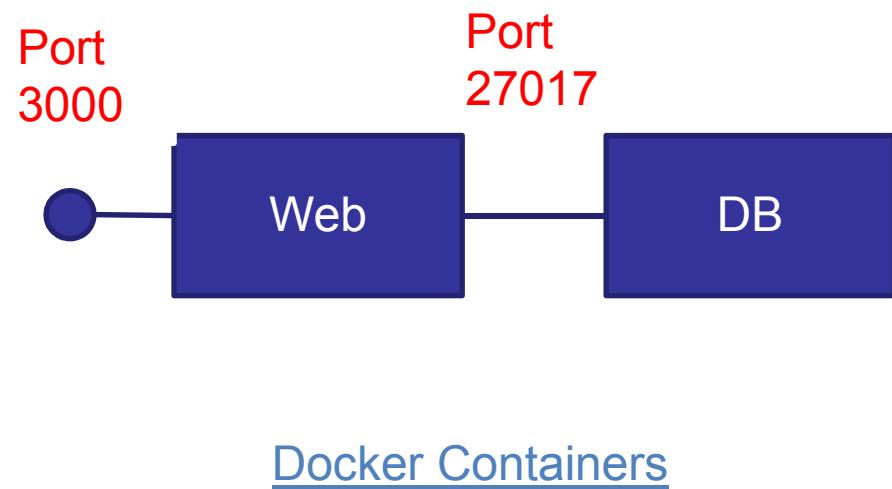
MeanStack

- Mean.js provides
 - Code generator to generate Mean App
 - Mean.js apps typically have
 - Node Js Server
 - Mongo DB database
 - Provides Dockerfile and fig.yml to run the app in Docker Containers
 - One Docker container for Node Js Server
 - One Docker container for Mongo DB Database

MeanStack Configuration

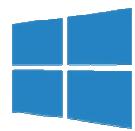
```
fig.yml x  
web:  
  build: .  
  links:  
    - db  
  ports:  
    - "3000:3000"  
  environment:  
    NODE_ENV: development  
db:  
  image: mongo  
  ports:  
    - "27017:27017"
```

[fig.yml](#)



Who uses Docker?

Docker PAAS Providers



Microsoft Azure

Google Cloud Platform



dotCloud



StackDock



DigitalOcean

And many more...

Who uses Docker?

As an Infrastructure Tool along side



Docker Integration

- OpenStack + Docker
- Hadoop + Docker
- Spark + Docker
- Mesos + Docker
-

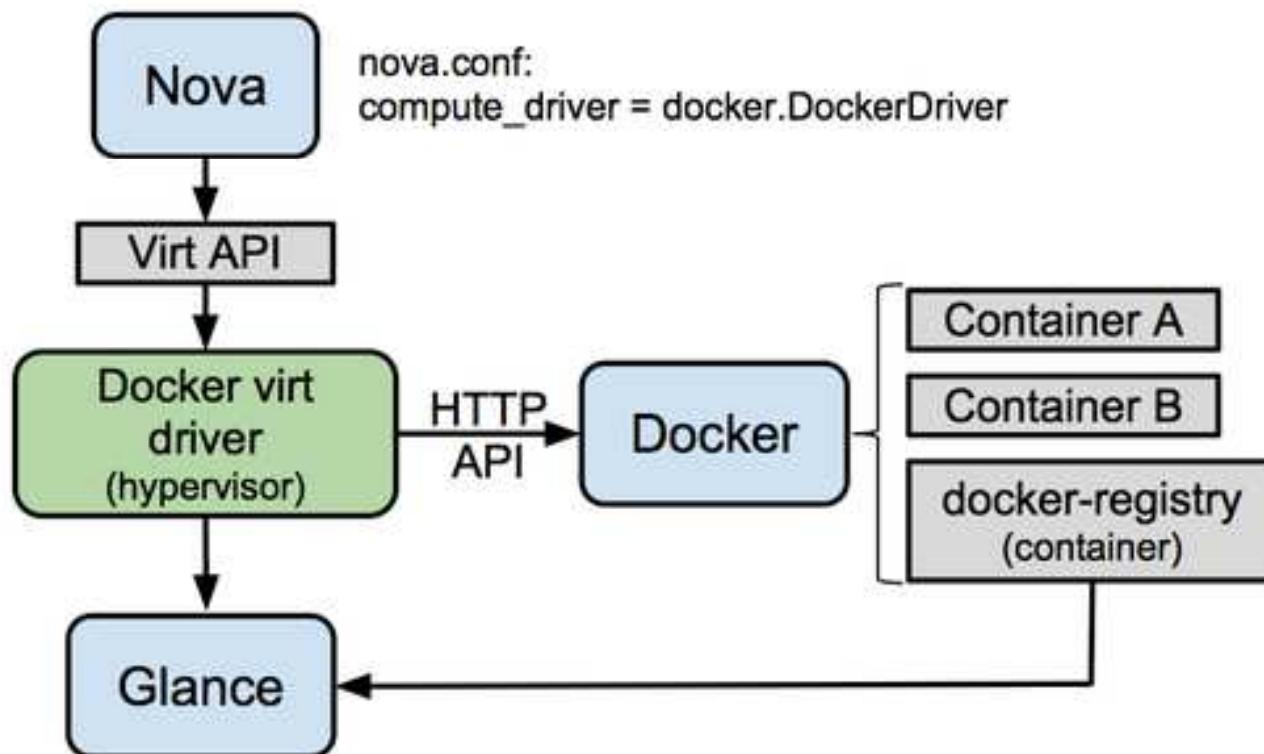
OpenStack Deployment With Docker

- **Nova:** add a new nova docker driver.
<https://github.com/stackforge/nova-docker>
- **HEAT:**
 - Add a new HEAT Resource:
 - HEAT docker driver interacts with docker server
- **Container as a Service**
 - A new service for OpenStack for manage docker container

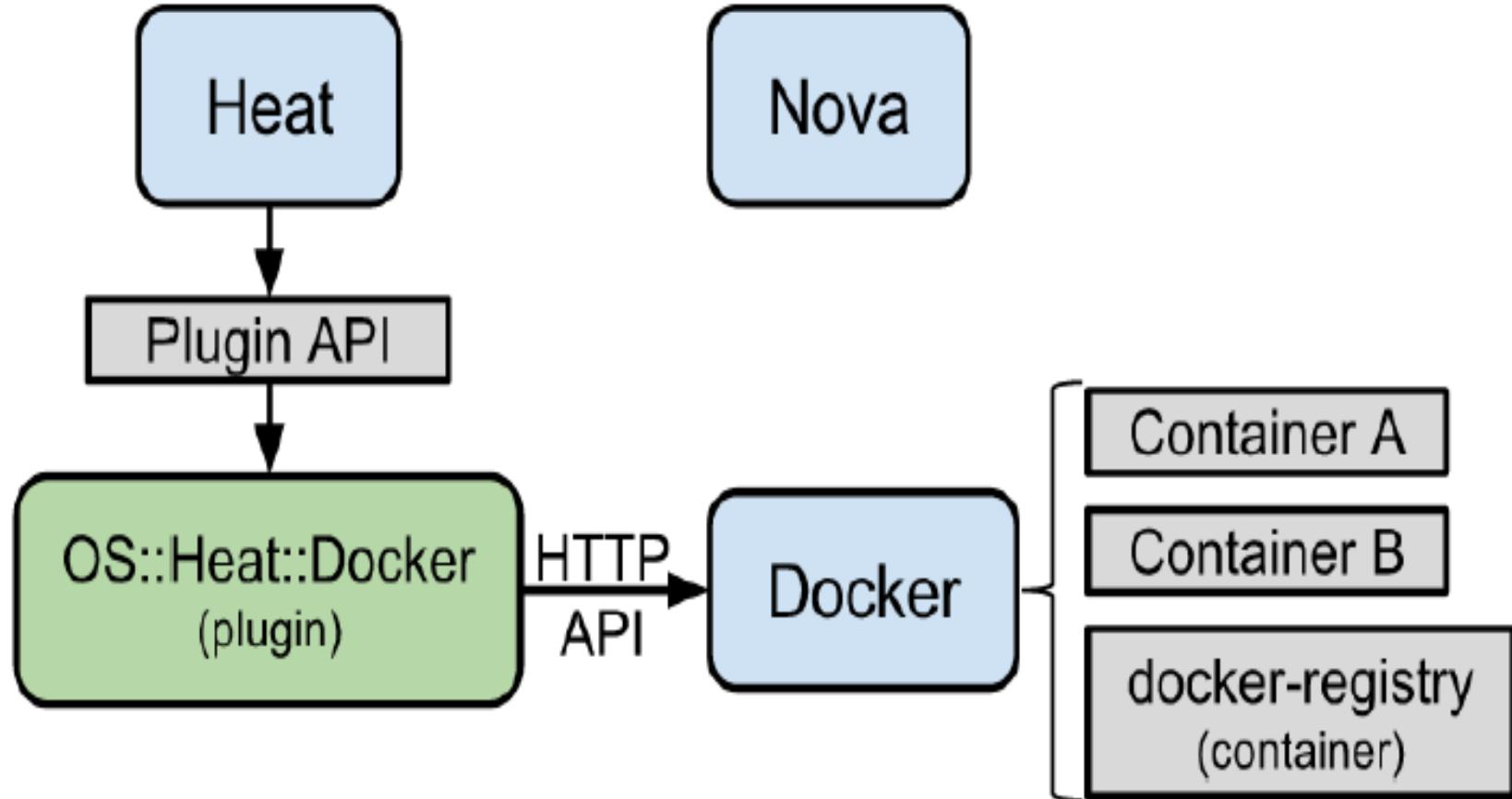
Solutions are still under development

Havana: nova-docker driver

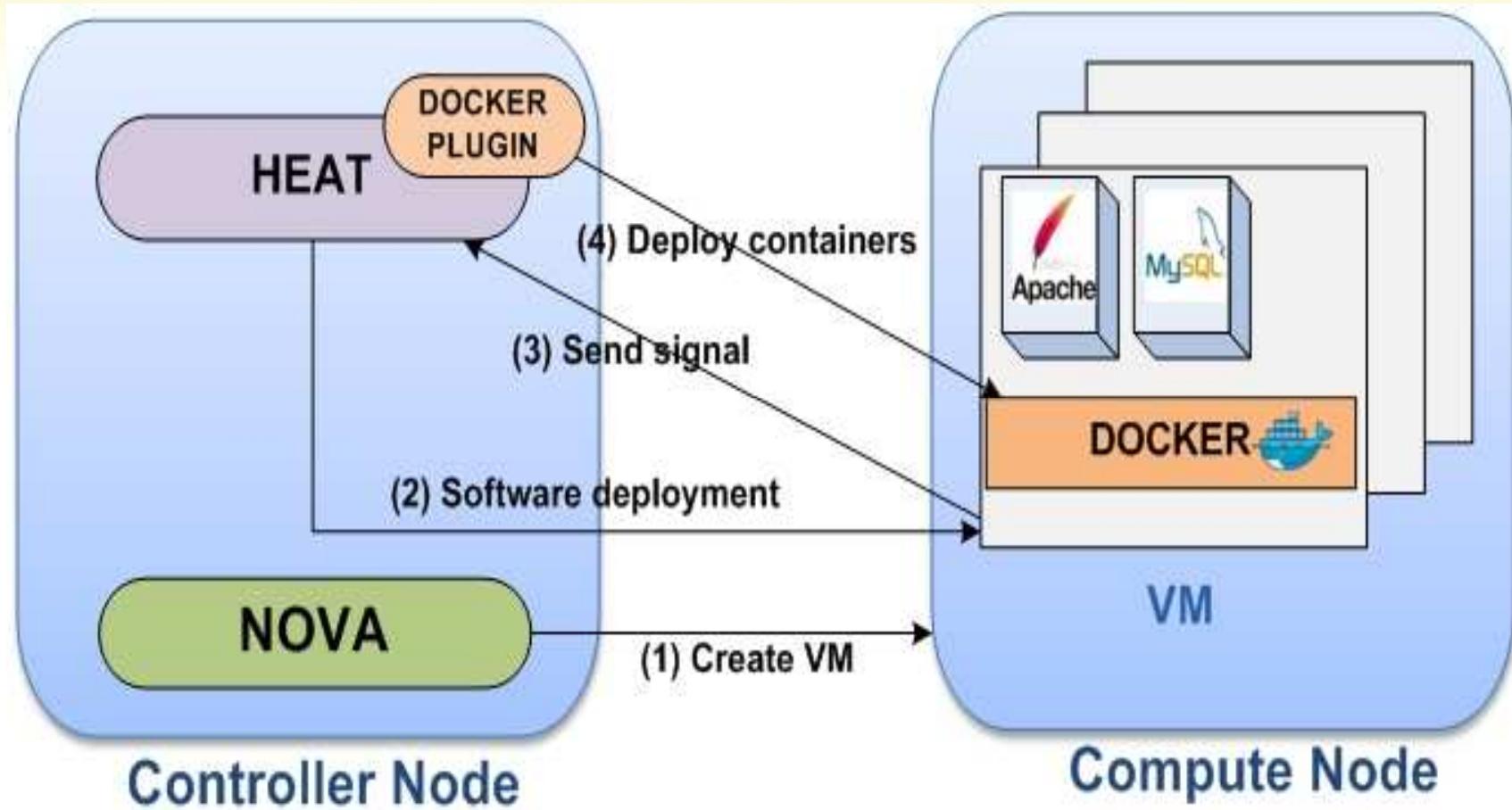
- Nova driver which integrates with docker REST API on backend
- Glance translator to integrate docker images with Glance



Icehouse: heat plugin for docker



HEAT Docker Driver



- <https://github.com/MarouenMechtri/Docker-containers-deployment-with-OpenStack-Heat>
- <http://techs.enovance.com/7104/multi-tenant-docker-with-openstack-heat>



Cloud Architectures & Applications

Cloud App Development Demo

Course Material Sharer

Table of Contents

- Project design
- Install tools
- Implementation
- Deployment

- The source code pieces showed in this slide can be found at :
https://github.com/lidangz/2015CourseMaterials/blob/master/Code_pieces_Part4_Lab.txt
- The source code files can be downloaded from:
https://github.com/lidangz/2015CourseMaterials/blob/master/Code_Part4_Lab.zip
- The full project can be found at:
<https://github.com/lidangz/MaterialSharer>
- The tools used in the lab can also be downloaded from:
<http://pan.baidu.com/s/1AWzZg#path=%252FCourseTools>

Apply an Account for OpenShift

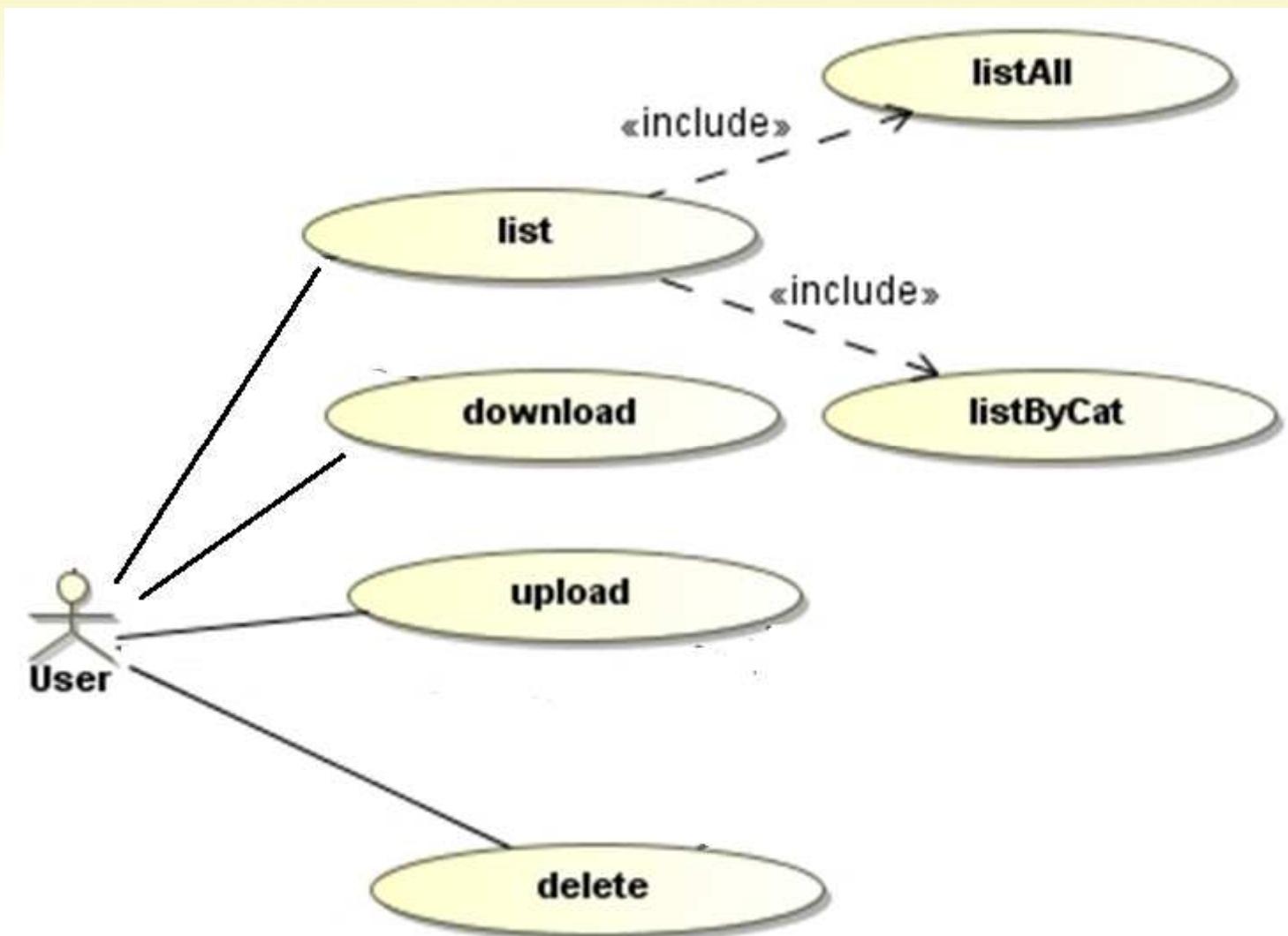
- Sign up a free account from
<https://www.openshift.com/>

PROJECT DESIGN

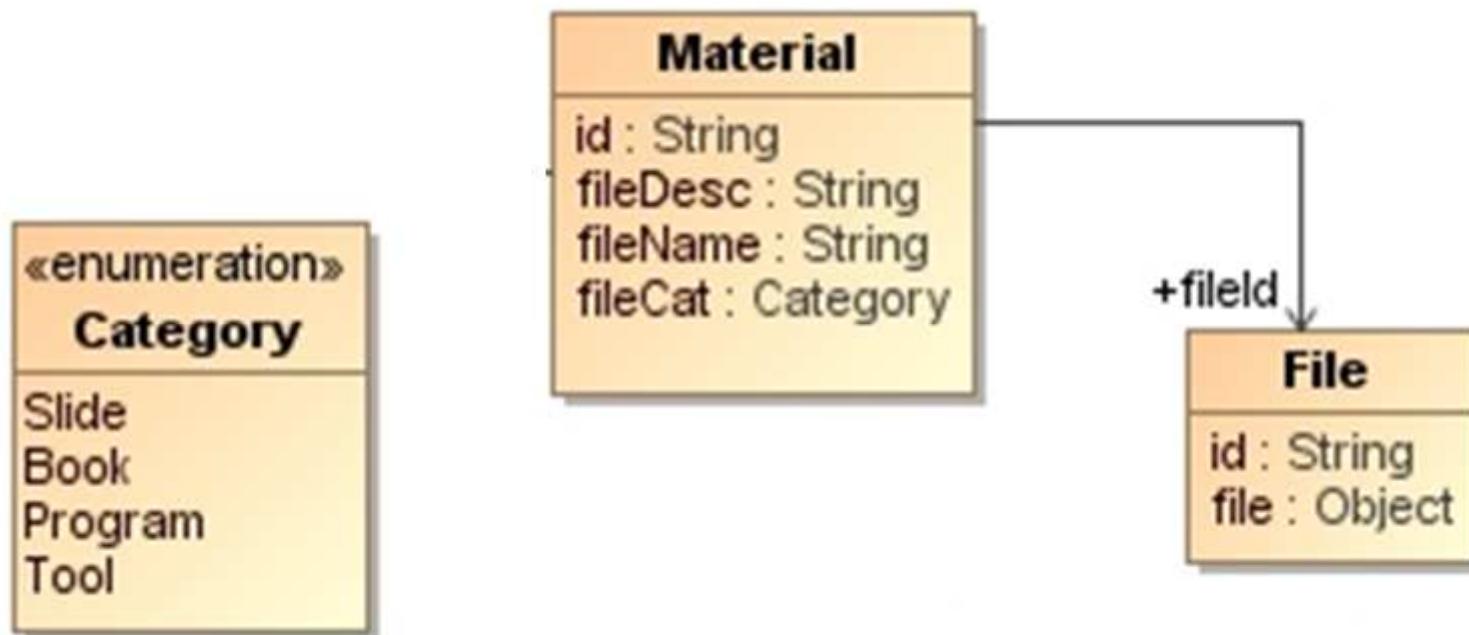
Course Material Sharer : Ideas

- We want to build a course material sharing app. and deploy it in cloud
- A user can **list** the all the sharing materials or a **category** of materials, and **download** the interested one.
- A user can **upload** new material and **delete** the materials.

App Design : Use Case



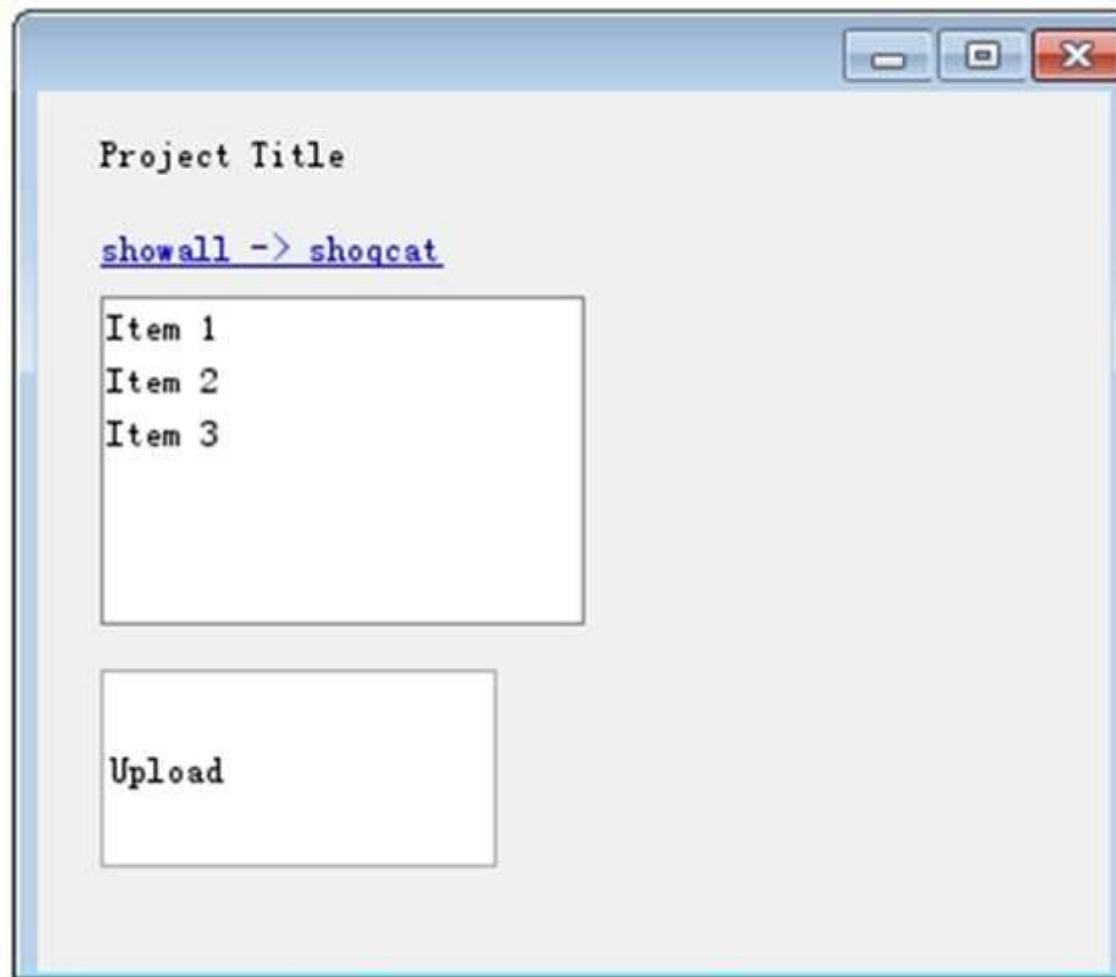
App Design : Class Diagram



Technical Stacks

- Use **HTML + JavaScript + Ajax + REST** in the front-end;
- Use **Node.js + MongoDB + Express** in the back-end;
- The developing version will be installed on **localhost** using the **8080** port.
- The app will be deployed on the OpenShift online.

UI Design of Single-page App



REST API Design

Action	HTTP Method	URL
listAll	GET	http://localhost:8080/materials
listByCat	GET	http://localhost:8080/materials/{catalog}
upload	POST	http://localhost:8080/materials
delete	DELETE	http://localhost:8080/materials/{materialId}
download	GET	http://localhost:8080/files/{fileId}

INSTALL TOOLS

Install MongoDB

- Download mongoDB from
<http://www.mongodb.org/downloads>
(msi for windows)
- Create a new folder *c:\mongodb*, and install mongodb to the folder
- Create a new folder *data* inside the folder *mongodb* to store data

Start MongoDB

- Open a cmd, goto folder *c:\mongodb\bin*
- Start monfoDB by typing

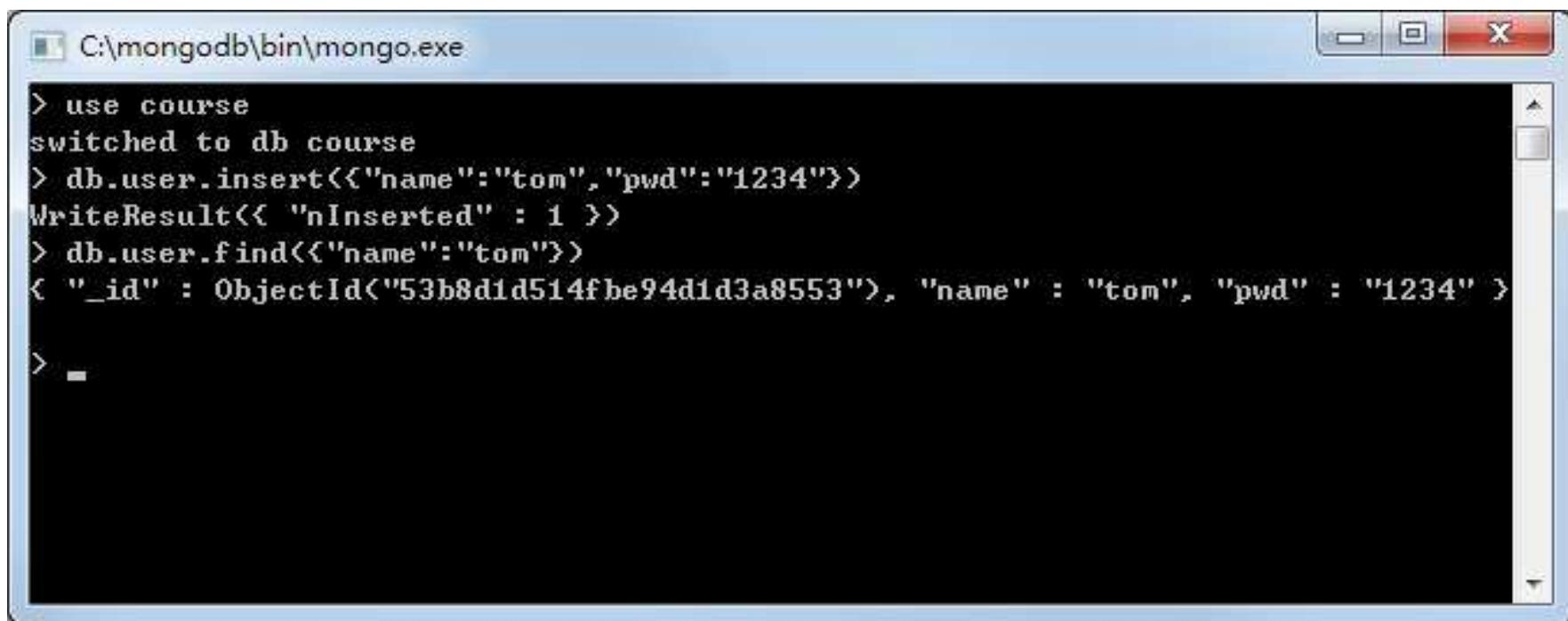
```
mongod.exe --dbpath c:\mongodb\data
```

- Open another cmd, goto folder *c:\mongodb\bin* , and type `mongo.exe` to open a management window.

Add a User to MongoDB

- Go to the management window.
 - Open database `course`
 - Add to collection `user` a user

```
> use course
> db.user.insert({“name”:“tom”,“pwd”：“1234”})
> db.user.find({name:“tom”})
```



The screenshot shows a Windows command-line interface window titled "C:\mongodb\bin\mongo.exe". The window contains the following MongoDB shell session:

```
C:\mongodb\bin\mongo.exe

> use course
switched to db course
> db.user.insert({“name”:“tom”,“pwd”：“1234”})
WriteResult<{ “nInserted” : 1 }>
> db.user.find({“name”:“tom”})
{ “_id” : ObjectId(“53b8d1d514fbe94d1d3a8553”), “name” : “tom”, “pwd” : “1234” }

>
```

Install Node.js

- Create a new folder `c:\nodejs`
- Download and install Node.js : <http://nodejs.org/>
 - Click ‘Install’ or go to the ‘Downloads’ page
 - Once downloaded, run the installer
 - Install Node.js in directory `c:\nodejs`



Test Node.js Installation

- In a cmd window, go to `c:\nodejs` directory
- Type `node -v` in the command line
- If the version information is shown, Node.js is correctly installed.

```
C:\Users\lidan>cd c:\nodejs
c:\nodejs>node -v
v0.10.28
c:\nodejs>
```

Install OpenShift Client Tools

Installing the client tools (**rhc**) on Windows requires three steps:

- **Step 1:** Install **Ruby** with RubyInstaller
 - **Step 2:** Install **Git** version control
 - **Step 3:** Install the **rhc** Ruby gem
-
- <https://developers.openshift.com/en/getting-started-windows.html> for details.

RubyInstaller



RubyInstaller
for Windows

[About](#) [Download](#) [Help](#) [Contribute](#)

The easy way to install Ruby on Windows

This is a **self-contained Windows-based installer** that includes the **Ruby language**, an execution environment, important **documentation**, and more.

[Download](#)

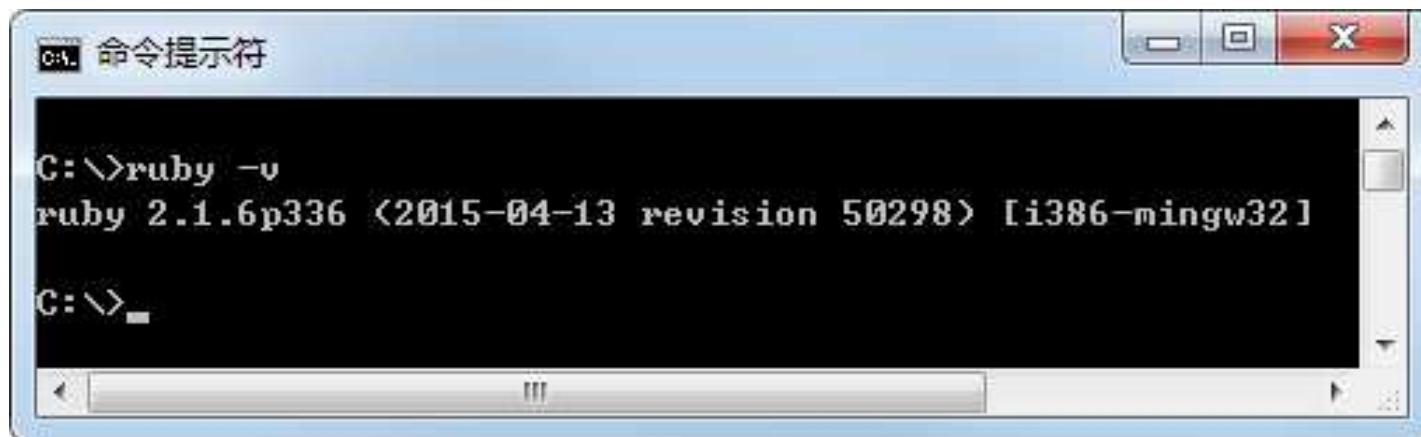
[Add-ons](#)



RubyInstaller provides the best experience for installing Ruby on Windows.

Step 1: Install Ruby

- Download the 2.1.xx version of RubyInstaller from <http://rubyinstaller.org/>)
- Launch the installer , select the **Add Ruby executables to your PATH** check box.
- Verify that the installation:



Step 2: Install Git

- Download Git for Windows from
<http://msysgit.github.io/>
- Install the Git, selecting the ***Run Git from the Windows Command Link Prompt*** checkbox, also, selecting **Checkout Windows-style, commit Unix-style line endings.**
- Verify the installation:



A screenshot of a Windows Command Prompt window titled "命令提示符". The window shows the command "git --version" being run and its output: "git version 1.9.2.msysgit.0". The window has a standard Windows title bar and scroll bars.

```
C:\>git --version
git version 1.9.2.msysgit.0

C:\>
```

Git Basics

- Git is a distributed revision control system. Git keeps a **repository** in your working directory. It can push/pull the source code to/from remote server.
- **git clone <url>** Clone an existing repository from remote server. It creates a working directory indicated by the url, and pulls down all the data for that repository.
- **git add <files>** Specify the files you want to track.
- **git commit -a “<msg>”** commit the changes to local git repository and provide an explanation as <msg>
- **git push** Upload the changed files of local repository to remote repository.
- **git status** Checking the status of local repository

More about git:

<http://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

©2015, Li Dan, Guizhou Academy of Sciences

Step 3: Install OpenShift gem

- Download rhc gem and all its dependency gems from <https://rubygems.org/gems/rhc> into a folder (using http instead of https)*.
- Goto that folder, install rhc locally:

```
gem install rhc --local .\rhc-1.35.3.gem
```

* The gems package can also be downloaded from <http://pan.baidu.com/s/1qWLtjB2>

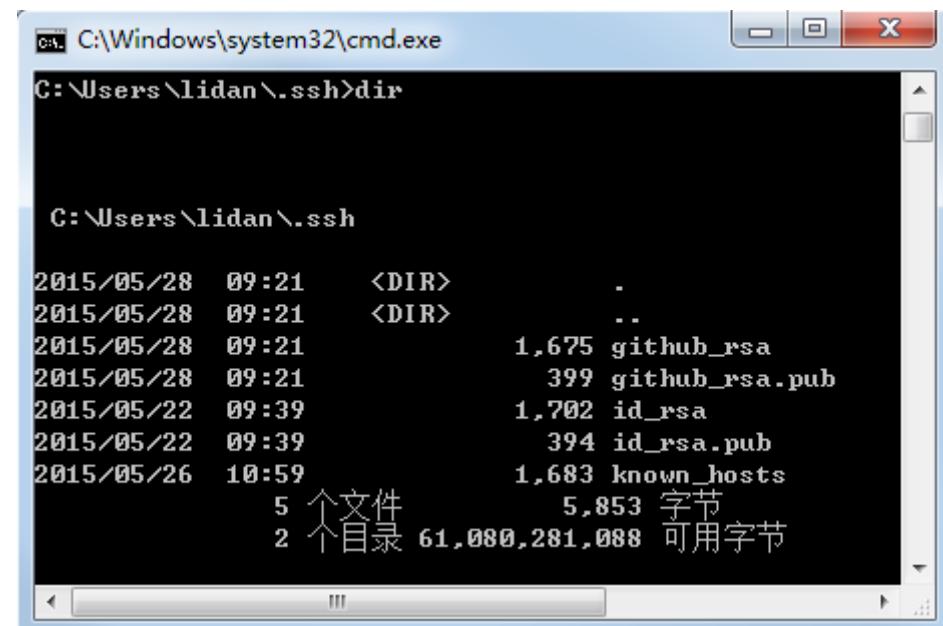
Configure OpenShift gem

- Open a cmd and run:

```
rhc setup
```

- Input OpenShift username and password;
- In this process, a pair of SSH keys will be generated and kept in `.ssh` folder in your personal folder.

Answering yes to
generate the keys,
and yes to upload the
public key.



Secure Shell (SSH)

- A cryptographic network protocol client-server communication in a secure way.
- SSH uses automatically generated public-private key pairs to encrypt a network connection.
- The public key is present on the server end and the matching private key is present on the local machine.

Check SSH Key in OpenShift

- Login the web console (<https://www.openshift.com/>)
- Click on *Settings* to check the **Public Keys**. If there are more than one, delete all others except the one you just uploaded.

The screenshot shows the OpenShift Online web interface. At the top, there is a dark header bar with the "OPENSHIFT ONLINE" logo and a menu icon. Below the header, the word "Settings" is displayed. A green notification bar contains a checkmark icon and the text: "Upgrade your account to get access to additional gear sizes, multiple domains, more storage, and other great features!". Under the "Settings" heading, there is a section titled "Public Keys". The text in this section states: "OpenShift uses a public key to securely encrypt the connection between your local machine and your application and to authorize you to upload code. [Learn more about SSH keys.](#)" Below this text is a table with three columns: "Key name", "Type", and "Contents". There is one row in the table with the following data: "lidangzlidanX201", "ssh-rsa", and "AAAB3Nza..VxPW069n". To the right of the "Contents" column, there is a "Delete" button. At the bottom of the "Public Keys" section, there is a button labeled "Add a new key...".

Key name	Type	Contents
lidangzlidanX201	ssh-rsa	AAAB3Nza..VxPW069n

Add a new key...

Create an App

- Click on **Applications** and
- Add Application...

The screenshot shows the OpenShift Online application dashboard. At the top, there is a dark header bar with the OpenShift logo, the text "OPENSHIFT ONLINE", and user information including "Upgrade Plan" and an email address "lidan_gz@163.com". Below the header, there is a navigation bar with tabs for "Applications", "Settings", "Help", and "OpenShift Hub". The main content area has a title "Applications" with a cube icon and "1 of 3" followed by a gear icon. It lists one application named "gyblog" which is "Available in domain lidangz" and uses "MySQL 5.5, PHP 5.4". There is a blue button labeled "Add Application...". To the right of the application list, there is a section titled "You may want to..." with links for "Add a collaborator", "Use your own domain name", and "Create a scalable application". At the bottom, there is a section titled "From the command line" with instructions about the "rhc" client and links for "Access logs" and "Save and restore backups".

OPENSHIFT ONLINE

lidan_gz@163.com

Applications Settings Help OpenShift Hub

Applications 1 of 3

Available in domain lidangz

gyblog MySQL 5.5, PHP 5.4

Add Application...

You may want to...

Add a collaborator Use your own domain name Create a scalable application

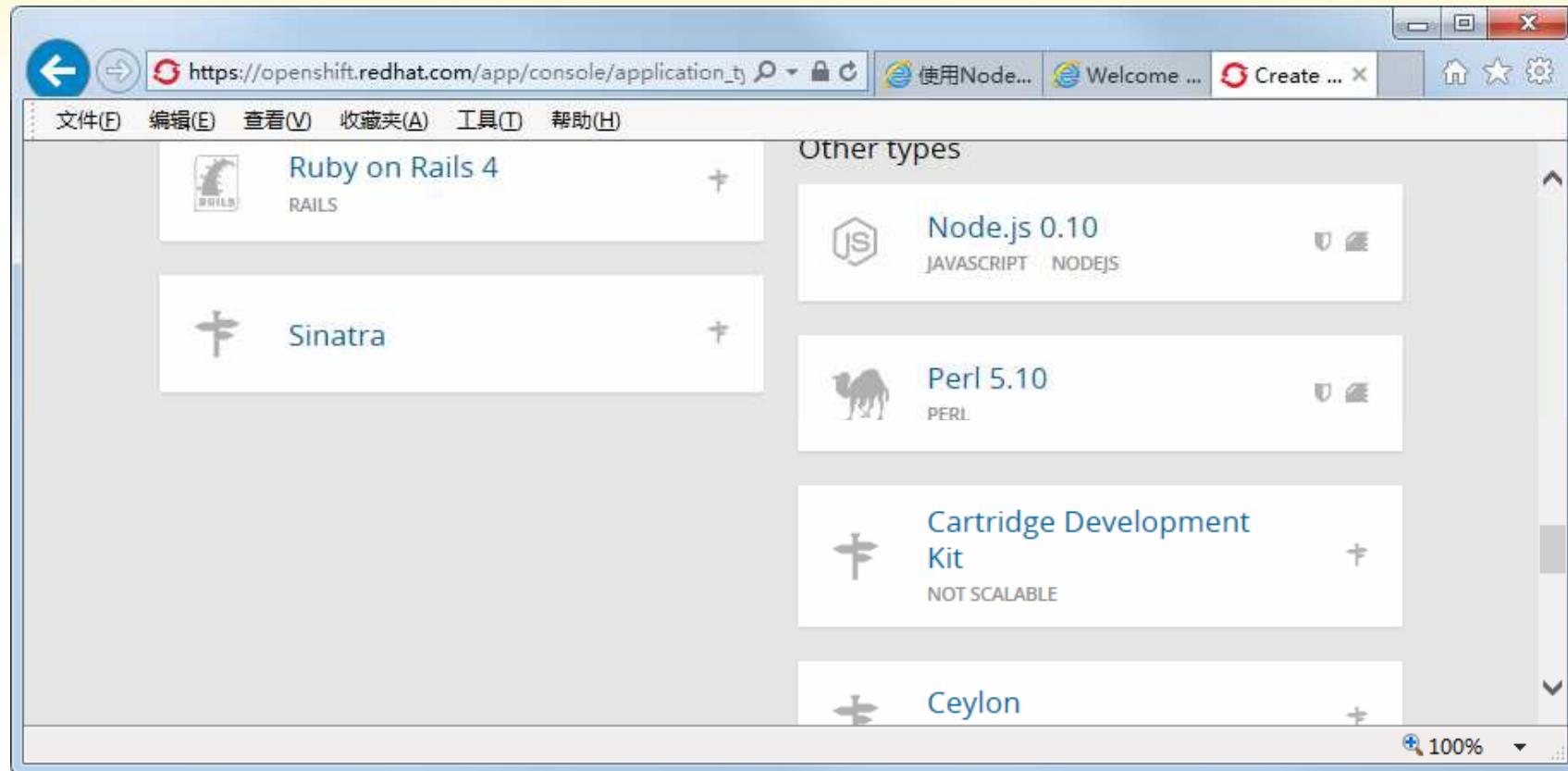
From the command line

The rhc client lets you:

Access logs Save and restore backups

Choose Type of the Application

- Select cartridge *Node.js 0.10*



- In the next page, input, for example “*mycourse*” to *Public URL*
- Press button *Create Application*

Configure the Application

- Input **mycourse** in *Public URL*
- Press button *Create Application*

The screenshot shows the configuration interface for creating a new application on OpenShift. At the top, there are two status indicators: a star icon followed by "OpenShift maintained" and a shield icon followed by "Receives automatic security updates". Below these are two input fields: "Public URL" containing "http:// mycourse -lidangz.rhcloud.com" and "Source Code" containing "Optional URL to a Git repository" and "Branch/tag". A note below the URL field states: "OpenShift will automatically register this domain name for your application. You can add your own domain name later." A note below the Source Code fields states: "We'll create a Git code repository in the cloud, and populate it with a set of reasonable defaults. If you provide a Git URL, your application will start with an exact copy of the".

☆ OpenShift maintained
🛡️ Receives automatic security updates

Public URL http:// mycourse -lidangz.rhcloud.com

OpenShift will automatically register this domain name for your application. You can add your own domain name later.

Source Code Optional URL to a Git repository Branch/tag

We'll create a Git code repository in the cloud, and populate it with a set of reasonable defaults. If you provide a Git URL, your application will start with an exact copy of the

Add NongoDB to the App

- *Continue to the application overview page*
- *Click Add MongoDB 2.4, and Add Cartridge in next page*

The screenshot shows the OpenShift Online application overview for the domain `mycourse-lidangz.rhcloud.com`. The application was created 9 minutes ago in the domain `lidangz` and the `aws-us-east-1` region. It is currently **Started** with 1 gear and 1 GB of storage. The application uses a **Node.js 0.10** cartridge, which is also listed under Cartridges. The Source Code URL is `ssh://5563d521e0b8cd9e7d0000a`. Under Databases, there are links to add MongoDB 2.4, MySQL 5.5, and PostgreSQL 9.2. Under Continuous Integration, there is an option to enable Jenkins. A "Delete this application..." button is visible at the bottom right.

OPENSHIFT ONLINE

Applications Settings Help ▾ OpenShift Hub

mycourse-lidangz.rhcloud.com change

Created 9 minutes ago in domain lidangz and the aws-us-east-1 region

Started 1 gear 1 GB

Cartridges

Node.js 0.10	Status Started	Gears 1 small	Storage 1 GB	Source Code <code>ssh://5563d521e0b8cd9e7d0000a</code>
--------------	----------------	---------------	--------------	---

Databases

- Add MongoDB 2.4
- Add MySQL 5.5
- Add PostgreSQL 9.2

Continuous Integration

- Enable Jenkins

Remote Access

Want to log in to your application?

Delete this application...

App's url

Overview of your App.

ssh url

The screenshot shows the Heroku app overview for 'mycourse-lidangz'. It includes:

- App's url:** mycourse-lidangz.rhcloud.com
- Cartridges:**
 - Node.js 0.10: Status Started, Gears 1 small, Storage 1 GB
 - MongoDB 2.4: Database: mycourse, User: admin, Password: show
- Source Code:** ssh://5563d521e0b8cd9e7d0000a
- Remote Access:** Want to log in to your application?
- Continuous Integration:** Enable Jenkins
- Tools and Support:**
 - Add 10gen Mongo Monitoring Service Agent
 - Add RockMongo 1.1
- Delete this application...**

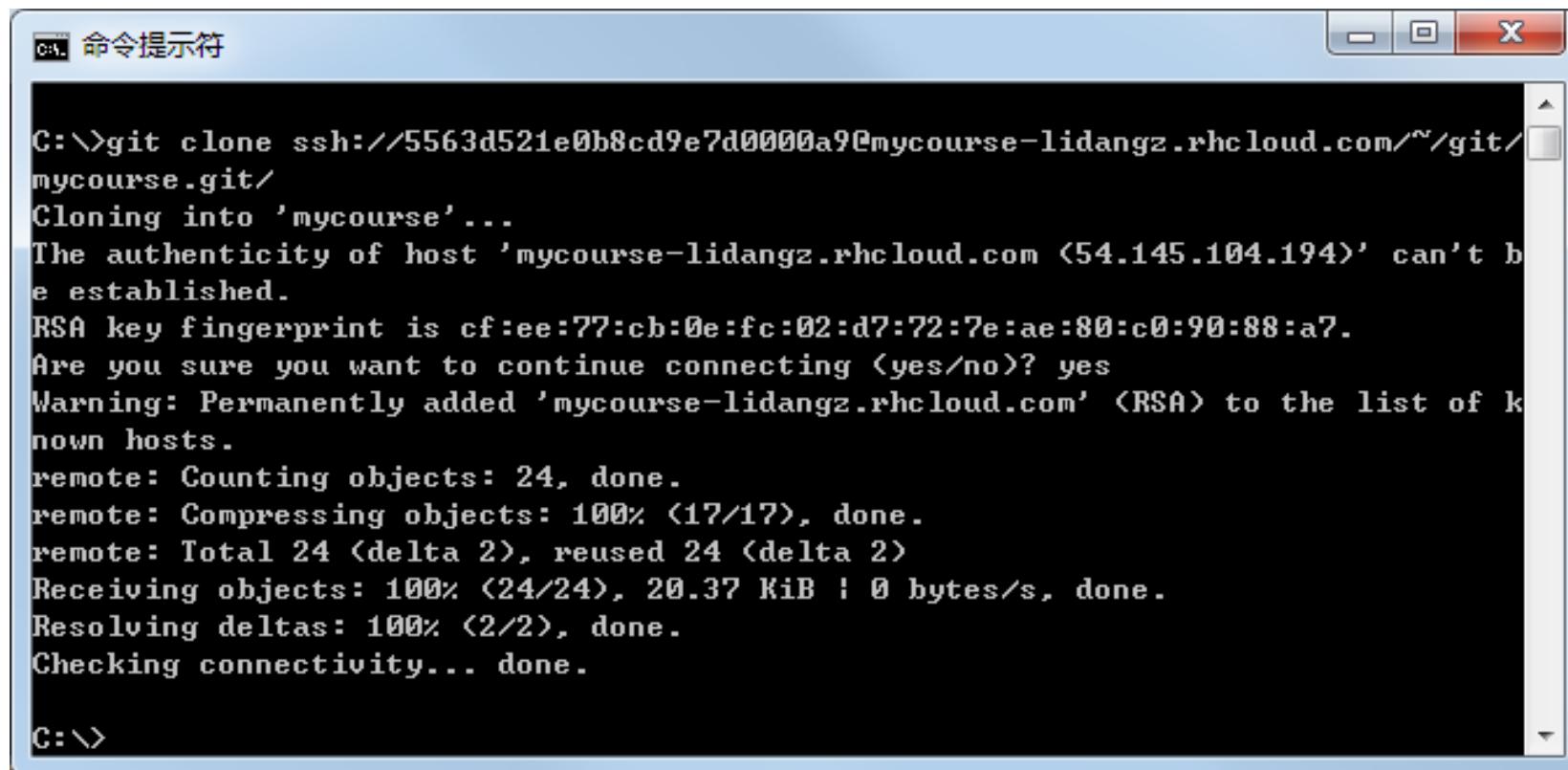
Name of your database

- Copy your ssh url under the **Source Code**

Download Source Code

- Open a cmd, goto `c:\`, and run

```
git clone <sshUrl>
```



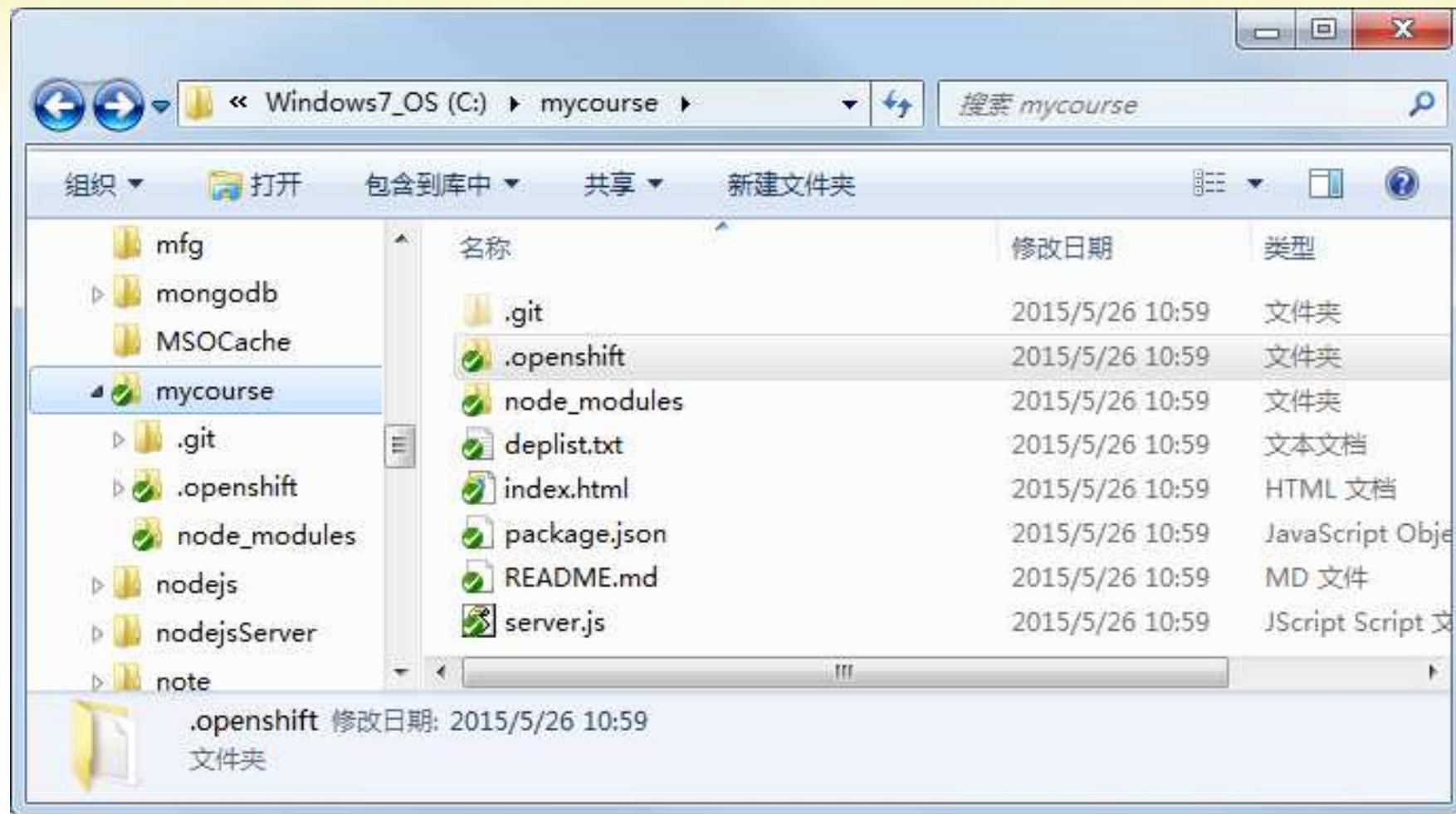
A screenshot of a Windows Command Prompt window titled "命令提示符". The window shows the following command and its execution:

```
C:\>git clone ssh://5563d521e0b8cd9e7d0000a9@mycourse-lidangz.rhcloud.com/~git/mycourse.git/
Cloning into 'mycourse'...
The authenticity of host 'mycourse-lidangz.rhcloud.com (54.145.104.194)' can't be established.
RSA key fingerprint is cf:ee:77:cb:0e:fc:02:d7:72:7e:ae:80:c0:90:88:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'mycourse-lidangz.rhcloud.com' (RSA) to the list of known hosts.
remote: Counting objects: 24, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 24 (delta 2), reused 24 (delta 2)
Receiving objects: 100% (24/24), 20.37 KiB / 0 bytes/s, done.
Resolving deltas: 100% (2/2), done.
Checking connectivity... done.

C:\>
```

- Now, the `c:\mycourse` is your project folder.

Structure of the Project Folder



Modify package.json

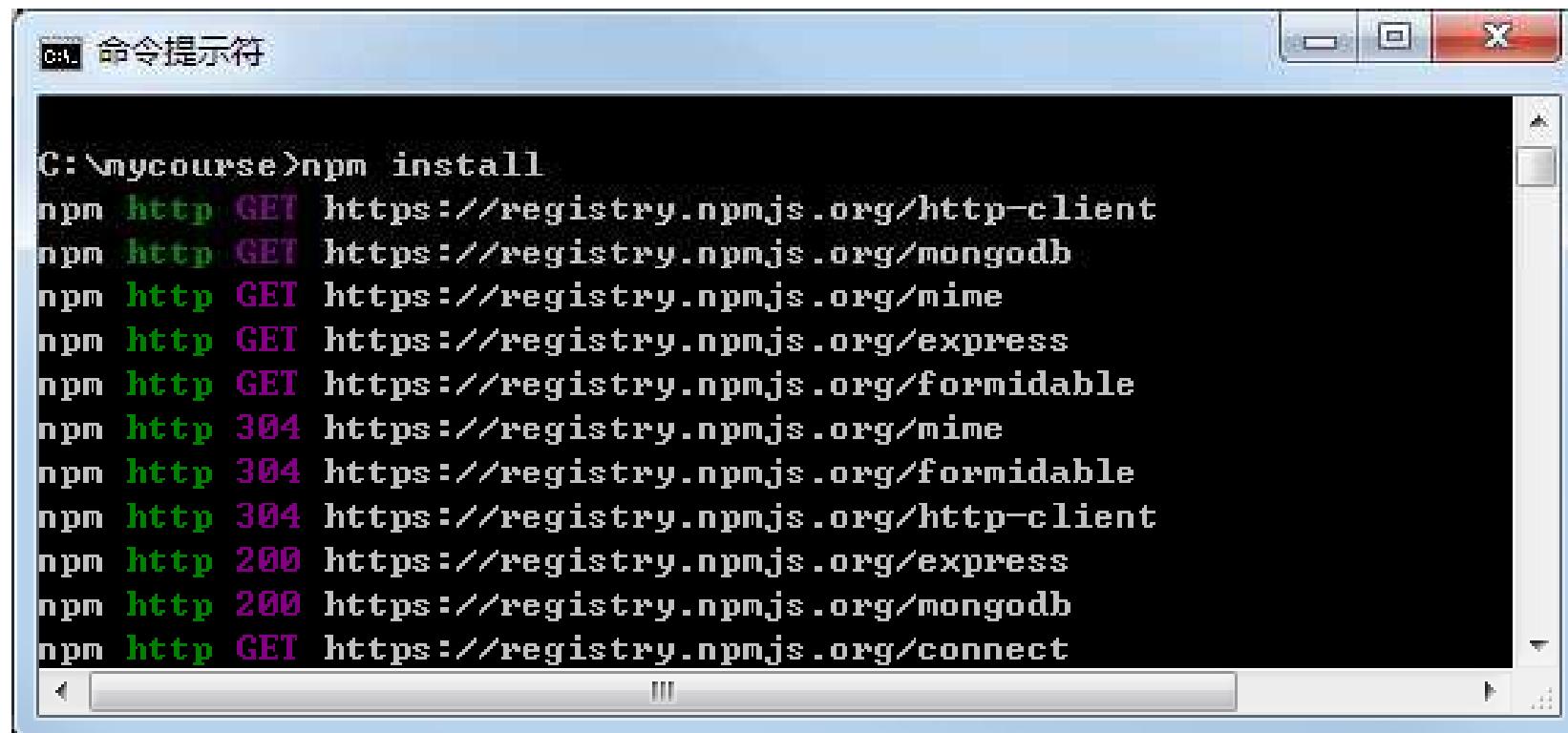
- Open the `package.json` file with an editor, and change it so it looks like this:

```
"dependencies": {  
    "express": "~3.4.4",  
    "formidable": "~1.0.15",  
    "mime": "~1.2.11",  
    "mongodb": "~1.4.7",  
    "http-client": "~1.0.0"  
},  
"devDependencies": {},  
"bundleDependencies": [],  
"private": true,  
"scripts": {  
    "start": "supervisor server.js"  
},  
"main": "server.js"  
}
```

Install Dependencies

- In the cmd, goto the project folder, run:

```
npm install
```

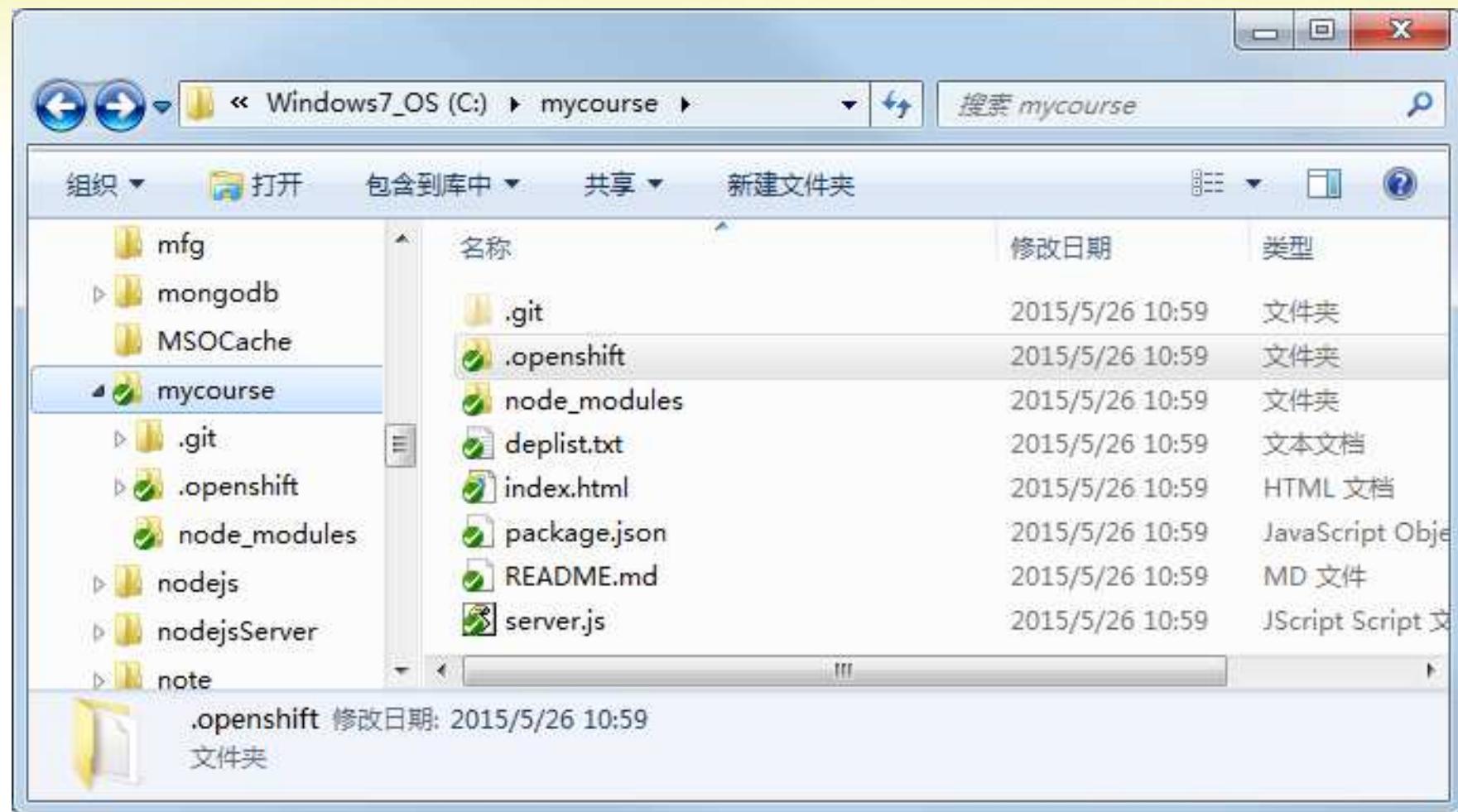


The screenshot shows a Windows Command Prompt window titled "命令提示符" (Command Prompt) with the path "C:\mycourse". The window displays the command "npm install" followed by a series of HTTP requests to the npm registry. The output is as follows:

```
C:\mycourse>npm install
npm http GET https://registry.npmjs.org/http-client
npm http GET https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/mime
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/mime
npm http 304 https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/http-client
npm http 200 https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/connect
```

IMPLEMENTATION

Files in the Project Folder



Start the Development

- Create the following two files in the project folder.

handlers.js

dbutils.js

- Now the project folder has the following files.

– Server-side

- *server.js* -- app main entry, route the requests
- *handlers.js* -- actually deal with the requests
- *dbutils.js* -- database connection
- *package.json* -- configuration file

– Client-side

- *Index.html* -- front-end html file

Design an Empty Client-side Html

- Replace the contents of `index.htm` with the following code:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Course material sharer</title>
    <script type="text/javascript">

    </script>
  </head>
  <body>
    <h2>Training Course Material Sharer</h2></p>
    <div id="catDiv">

    </div>
    <div id="listDiv">

    </div>
    <p> =====
    <div id="upLoadDiv">

    </div>
  </body>
</html>
```

Server-side Design: *server.js*

- Here we start the app, and route the requests to functions in *handler.js*. Replace the contents of *server.js* with the following code:

```
var express = require('express');
var handlers = require('./handlers');

var ip_address = process.env.OPENSHIFT_NODEJS_IP || '127.0.0.1';
var port      = process.env.OPENSHIFT_NODEJS_PORT || 8080;

var app = express();
app.configure(function () {
    app.use(express.logger('dev'));
    app.use(express.bodyParser());
});

```

server.js (2)

- Followed by the code below:

```
console.log ('registering event routes with express');

app.get('/', handlers.start);
app.get('/materials', handlers.listAll);
app.get('/materials/:cat', handlers.listByCat);
app.post('/materials', handlers.upload);
app.get('/files/:fileId/:fileName', handlers.download);
app.delete('/materials/:id', handlers.delete);

console.log ('About to start listening');
app.listen(port,ip_address);
console.log('Listening on port: ', port, ' of ', ip_address);
```

handlers.js

- Here we write the logic to handle the requests .
Update *handlers.js* as below:

```
var fs = require('fs');
var mime = require('mime');
var formidable = require("formidable");
var mongo = require('mongodb');
var BSON = mongo.BSONPure;
var dbutils = require('./dbutils');
var mongoUrl = dbutils.getMongoUrl();

exports.start = function(req, res) {
  console.log("Request handler 'start' was called.");
  fs.readFile('index.html', 'utf-8', function (err, data) {
    if (err) {return console.dir(err);}
    res.setHeader('Content-Type', 'text/html');
    res.send(data);
  });
}
```

Deal with *upload* in *handlers.js*

```
exports.upload = function(req,res) {
    console.log("Request handler 'upload' was called.");

    var form = new formidable.IncomingForm();
    form.parse(req, function(err, fields, files) {
        mongo.Db.connect(mongoUrl, function (err, db) {
            var idobj=new mongo.ObjectID();
            var fileId=idobj.toString();
            var gridStore = new mongo.GridStore(db, fileId, 'w');
            gridStore.writeFile(files.upload.path, function(err, fileInfo) {
                db.collection('material',{safe:true},function(err,collection){
                    fields["fileName"]=files.upload.name;
                    fields["fileId"]=fileId;
                    collection.insert(fields,{safe:true},function(err,result){
                        fields["_id"]=result._id;
                        db.close();
                        res.setHeader('Content-Type', 'text/html');
                        res.send(JSON.stringify(fields));
                    });
                });
            });
        });
    });
}
```

Functions listAll, listByCat in *handlers.js*

```
exports.listAll = function(req, res) {
    console.log("Request handler 'listAll' was called.");
    readList(res,null);
}

exports.listByCat = function(req, res) {
    console.log("Request handler 'listByCat' was called.");
    var cat = req.params.cat;
    readList(res,cat);
}
```

Function Called by listAll, listByCat

```
function readList(res,cat) {  
    var qstr = {};  
    if (cat!=null) {  
        qstr=fileCat : "+ cat +"});  
    };  
  
mongo.Db.connect(mongoUrl, function (err, db) {  
    db.collection('material', function(err, collection) {  
        collection.find(qstr).toArray(function(err, items) {  
            res.setHeader('Content-Type', 'text/html');  
            res.send(JSON.stringify(items));  
            db.close();  
        });  
    });  
});  
}
```

Deal with *download* in *handlers.js*

```
exports.download = function(req,res) {
    console.log("Request handler 'download' was called.");

    var fileId = req.params.fileId;
    var fileName = req.params.fileName;
    mongo.Db.connect(mongoUrl, function (err, db) {
        var gridStore = new mongo.GridStore(db, fileId, 'r');
        gridStore.open(function(err, gridStore) {
            var stream = gridStore.stream(true);
            stream.on("end", function(err) {
                db.close();
                res.end();
            });
            var contentType=mime.lookup(fileName).toString();
            res.setHeader('Content-Type', contentType);
            stream.pipe(res);
        });
    });
}
```

Deal with *delete* in *handlers.js*

```
exports.delete = function(req, res) {
  console.log("Request handler 'delete' was called.");
  var id = req.params.id;
  mongo.Db.connect(mongoUrl, function (err, db) {
    db.collection('material', {safe:true}, function(err, collection) {
      collection.findOne({'_id':new BSON.ObjectID(id)}, function(err, doc) {
        var fileName=doc.fileName;
        collection.remove({'_id':new BSON.ObjectID(id)}, {safe:true}, function(err, result) {
          if (doc.fileId) {
            var gridStore=mongo.GridStore;
            gridStore.unlink(db, doc.fileId, function(err) {
              console.log('delete '+fileName);
              db.close();
              readList(res,null);
            });
          } else { db.close();
            readList(res,null);
          };
        });
      });
    });
  });
}
```

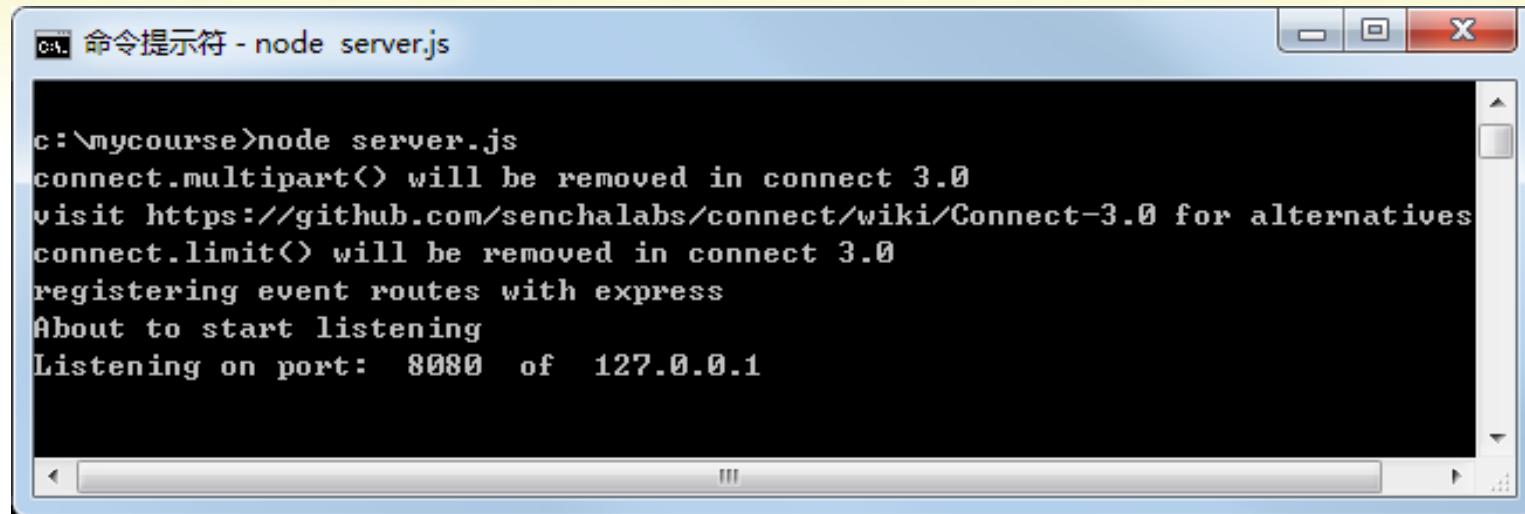
Add the Follows to *dbutils.js*

```
var mongostr = {                                     // local machine
    "hostname": "localhost",
    "port": 27017,
    "username": "tom",
    "password": "1234",
    "name": "",
    "db": "course"
}
if(process.env.OPENSIFT_NODEJS_PORT){           // OpenShift
    mongostr = {
        "hostname": process.env.OPENSIFT_MONGODB_DB_HOST,
        "port": process.env.OPENSIFT_MONGODB_DB_PORT,
        "username": process.env.OPENSIFT_MONGODB_DB_USERNAME,
        "password": process.env.OPENSIFT_MONGODB_DB_PASSWORD,
        "name": "",
        "db": "mycourse"
    }
}
exports.getMongoUrl = function() {
    return "mongodb://" + mongostr.username + ":" +
        mongostr.password + "@" + mongostr.hostname + ":" +
        mongostr.port + "/" + mongostr.db;
}
```

Change to name of your database

Test the App in Local

- Cd to the project folder, and start the server



```
c:\mycourse>node server.js
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
registering event routes with express
About to start listening
Listening on port: 8080 of 127.0.0.1
```

- Access <http://localhost:8080/> from a browser
- Check the log message showed by Nodejs

Client-side Design

Upload in index.html

- Replace the `uploadDiv` by :

```
<div id="upLoadDiv">
  <fieldset><legend>Select a file to upload:</legend>
    <p>File Cat:
      <select size="1" id="fileCat">
        <option selected>Slide</option>
        <option>Book</option>
        <option>Program</option>
        <option>Tool</option>
      </select> </p>
      <p>File Desc. : <input type="text" id="fileDesc" size="40"> </p>
      <p>Select File: <input type="file" id="upload"></p>
      <p><input type="button" value="Upload File" onclick="uploadFile()"/>
      </p>
    </fieldset>
  </div>
```

Design Upload in index.html (2)

- Add function `uploadFile` to the script section:

```
function uploadFile(){
    var urlstr = "/materials";
    var fileObj = document.getElementById("upload").files[0];
    var fileName=document.getElementById("upload").value;
    var fileDesc=document.getElementById("fileDesc").value;
    var fileCat=document.getElementById("fileCat").value;

    if (validate(fileName,fileDesc)){
        var form = new FormData();
        form.append("fileDesc", fileDesc);
        form.append("fileCat", fileCat);
        form.append("fileName", document.getElementById("upload").value);
        form.append("upload", fileObj);
```

Design Upload in index.html (3)

- Second part of function `uploadFile`:

```
xmlhttp = new XMLHttpRequest();
xmlhttp.open("post", urlstr, true);
xmlhttp.setRequestHeader("Content-type", "multipart/form-data");
xmlhttp.onreadystatechange = function(){
    if(xmlhttp.readyState === 4){
        if(xmlhttp.status === 200){
            var json = JSON.parse(xmlhttp.responseText);
            var ihtm = getOneLi(json);
            document.getElementById("listDiv").innerHTML += ihtm;
            document.getElementById("upload").value =null;
            fileDesc=document.getElementById("fileDesc").value=null;
        }else{
            alert('Error: '+xmlhttp.status);
        }
    }
}
xmlhttp.send(form);
}
```

Design Upload in index.html (4)

- Two more functions to the script section:

```
function validate(fileName, fileDesc) {  
    if (fileName == null || fileName == "") {  
        alert("Please select a file to upload.");  
        return false;  
    }  
    if (fileDesc == null || fileDesc == "") {  
        alert("Please input the file description.");  
        return false;  
    }  
    return true;  
}  
  
function getOneLi(json) {  
    var fpath= '/files/'+json["fileId"]+'/'+json["fileName"];  
    var str='<li><u><a onclick="listAll(\''+json["fileCat"]+'\')">' +  
            json["fileCat"]+'</a></u> :&nbsp; <a href="'+fpath+'">' +  
            json["fileDesc"]+ '&nbsp;(' +json["fileName"]+')</a> &nbsp; '+  
            '<a onclick="delItem(\''+_id+'\','+_id+')">[delete]</a></li>';  
    return str;  
}
```

Design List in `index.html`

- Replace the `catDiv` by :

```
<div id="catDiv">
    <u><a onclick="listAll(null)">All </a></u>
</div>
```

- Add event to the `body`:

```
<body onload="listAll(null)">
```

Design List in index.html (2)

- Add function `listAll` to the script section:

```
function listAll(cat){  
    var urlstr = "/materials";  
    if (cat!=null && cat !='') {  
        urlstr = urlstr+"/"+cat;  
    }  
    xmlhttp = new XMLHttpRequest();  
    xmlhttp.open("get", urlstr, true);  
    xmlhttp.onreadystatechange = function(){  
        if(xmlhttp.readyState === 4){  
            if(xmlhttp.status === 200){  
                showItems(xmlhttp.responseText,cat);  
            }else{  
                alert('Error: '+xmlhttp.responseText); // An error occurred  
            }  
        }  
    };  
    xmlhttp.send(null);  
}
```

Design List in index.html (3)

- Add function `showItems` to the script section:

```
function showItems(jsontxt,cat) {  
    var alljson = JSON.parse(jsontxt);  
    var ihtm = '<ul>';  
    for(var i=0;i<alljson.length;i++){  
        ihtm += getOneLi(alljson[i]);  
    }  
    var titlelstr='<u><a onclick="listAll(null)">All</a></u>';  
    if (cat!=null && cat !='') {  
        titlelstr += '<u>-->' +cat+ '</u>';  
    }  
  
    document.getElementById("listDiv").innerHTML=ihtm+'</ul>';  
    document.getElementById("catDiv").innerHTML=titlelstr;  
}
```

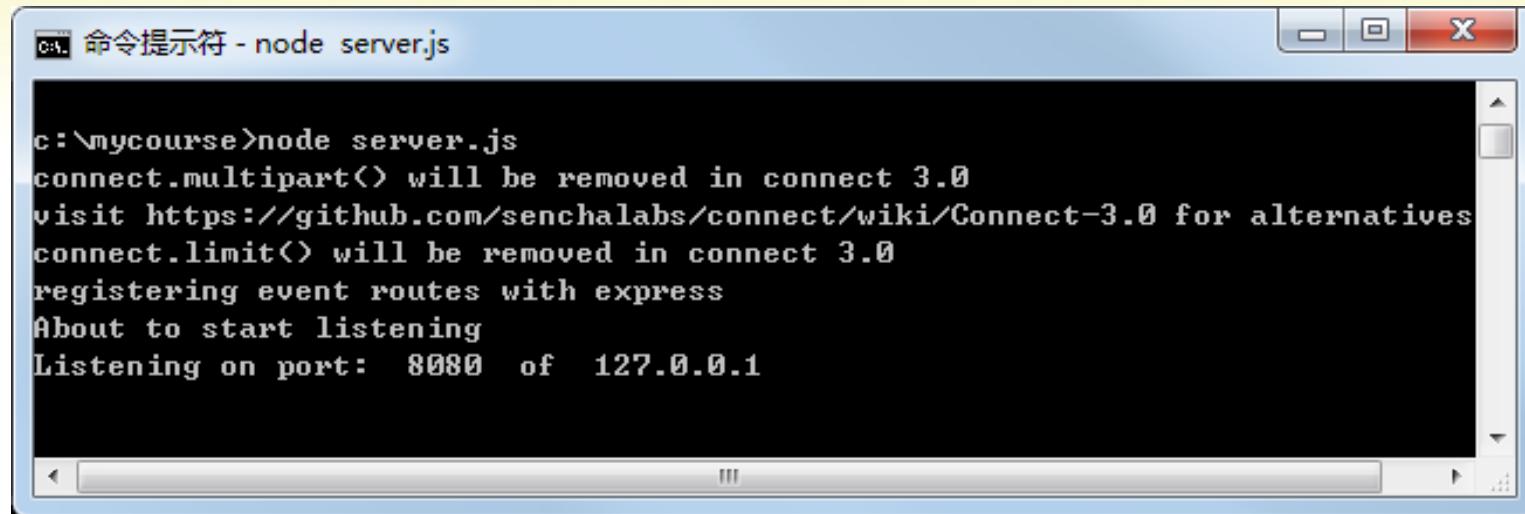
Design Delete in index.html

- Add function `delItem` to the script section:

```
function delItem(mid, fileName) {  
    if(confirm('Do you want to delete file '+fileName+' ?')){  
        var urlstr = '/materials'+'/'+mid;  
        xmlhttp = new XMLHttpRequest();  
        xmlhttp.open("delete", urlstr, true);  
        xmlhttp.send();  
        xmlhttp.onreadystatechange = function(){  
            if(xmlhttp.readyState === 4){  
                if(xmlhttp.status === 200){  
                    showItems(xmlhttp.responseText,null);  
                } else {  
                    alert('Error: '+xmlhttp.responseText);  
                }  
            }  
        };  
    };  
}
```

Again Test the App

- Cd to the project folder, and start the server



```
c:\mycourse>node server.js
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
registering event routes with express
About to start listening
Listening on port: 8080  of 127.0.0.1
```

- Access <http://localhost:8080/> from a browser
- Check the log message showed by Nodejs

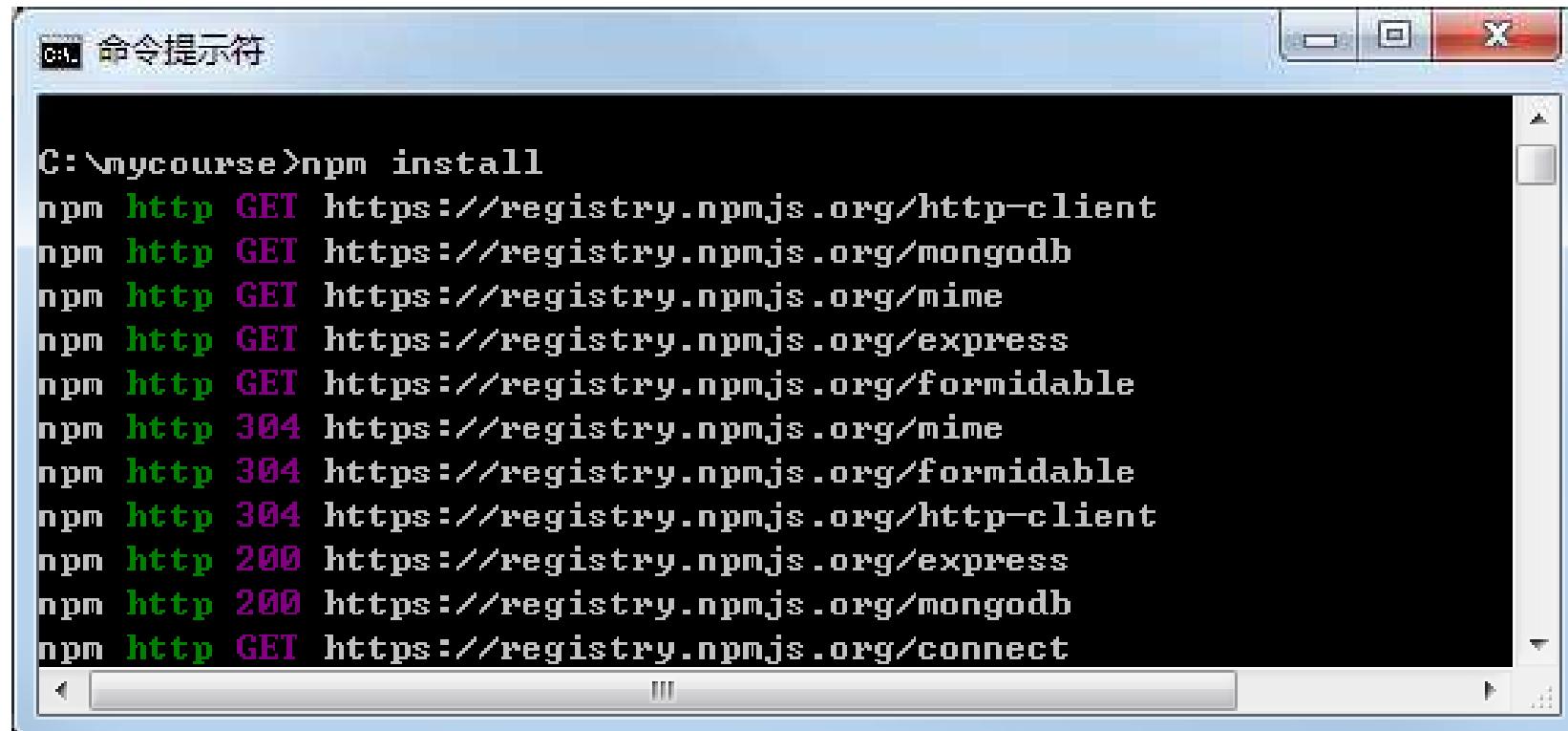
A Short-Cut for Development

- Download [Code Part4 Lab.zip](#) and unzip all files from to the project folder `c:\mycourse`
- Now the project folder has the following files.
 - Server-side
 - `server.js` -- app main entry, route the requests
 - `handlers.js` -- actually deal with the requests
 - `dbutils.js` -- database connection
 - `package.json` -- configuration file
 - Client-side
 - `Index.html` -- front-end html file

Install Dependencies

- In the cmd, goto the project folder `c:\mycourse`, run:

```
npm install
```



A screenshot of a Windows Command Prompt window titled "命令提示符". The window shows the command `npm install` being run in the directory `C:\mycourse`. The output consists of several lines of text, each starting with "npm http" followed by either "GET" or "200" and a URL from the npm registry. The text is color-coded, with "http" in green and the URLs in purple.

```
C:\mycourse>npm install
npm http GET https://registry.npmjs.org/http-client
npm http GET https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/mime
npm http GET https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/mime
npm http 304 https://registry.npmjs.org/formidable
npm http 304 https://registry.npmjs.org/http-client
npm http 200 https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/connect
```

Check the *dbutils.js*

```
var mongostr = {                                // local machine
    "hostname": "localhost",
    "port": 27017,
    "username": "tom",
    "password": "1234",
    "name": "",
    "db": "course"
}
if(process.env.OPENSIFT_NODEJS_PORT){           // OpenShift
    mongostr = {
        "hostname": process.env.OPENSIFT_MONGODB_DB_HOST,
        "port": process.env.OPENSIFT_MONGODB_DB_PORT,
        "username": process.env.OPENSIFT_MONGODB_DB_USERNAME,
        "password": process.env.OPENSIFT_MONGODB_DB_PASSWORD,
        "name": "",
        "db": "mycourse"
    }
}
exports.getMongoUrl = function() {
    return "mongodb://" + mongostr.username + ":" +
        mongostr.password + "@" + mongostr.hostname + ":" +
        mongostr.port + "/" + mongostr.db;
}
```

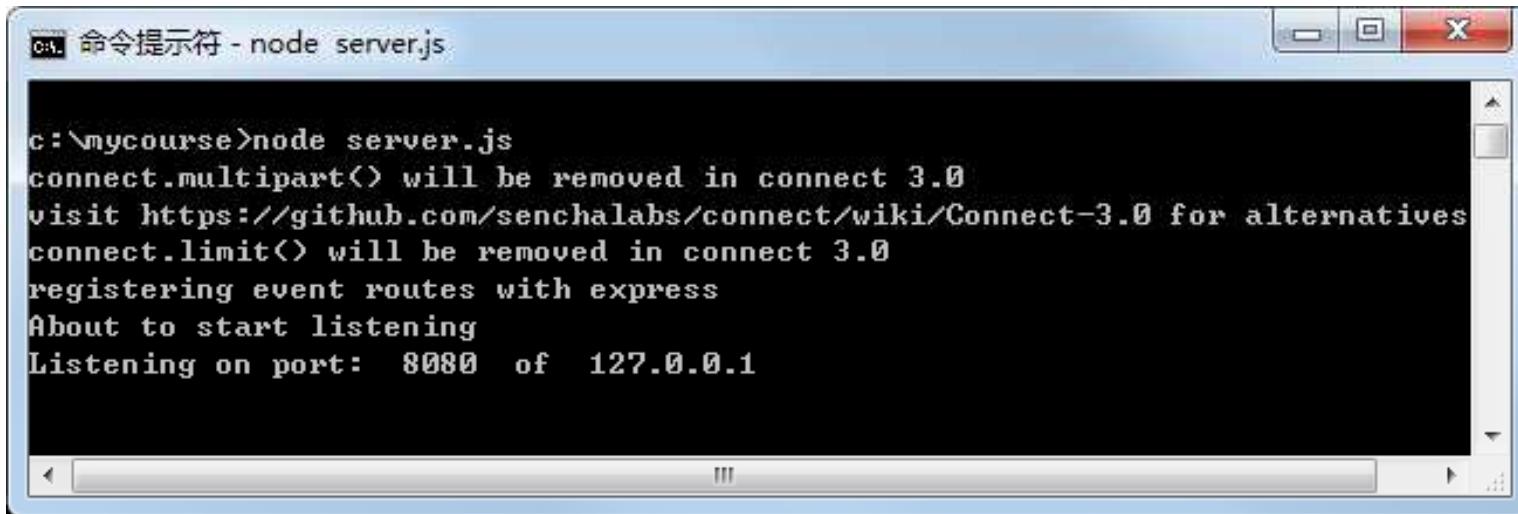
Change the name and password

Change to name of your database

DEPLOYMENT

Test the App in Local

- Be sure the mogondb is started
- Open a cmd, cd to the project folder, and start the Node.js server



```
c:\mycourse>node server.js
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
registering event routes with express
About to start listening
Listening on port: 8080 of 127.0.0.1
```

- Access <http://localhost:8080/> from an IE
- Check the log message showed by Nodejs

Upload code to OpenShift

- Open a cmd, cd to the project folder, runing

1. Add the two new js files to git control

```
git add handlers.js dbutils.js
```

2. Commit the changes to local git repository

```
git commit -am "My first change"
```

3. Push the code to OPENSIFT server*

```
git push
```

* Try several times if errors

Test the App in OpenShift

