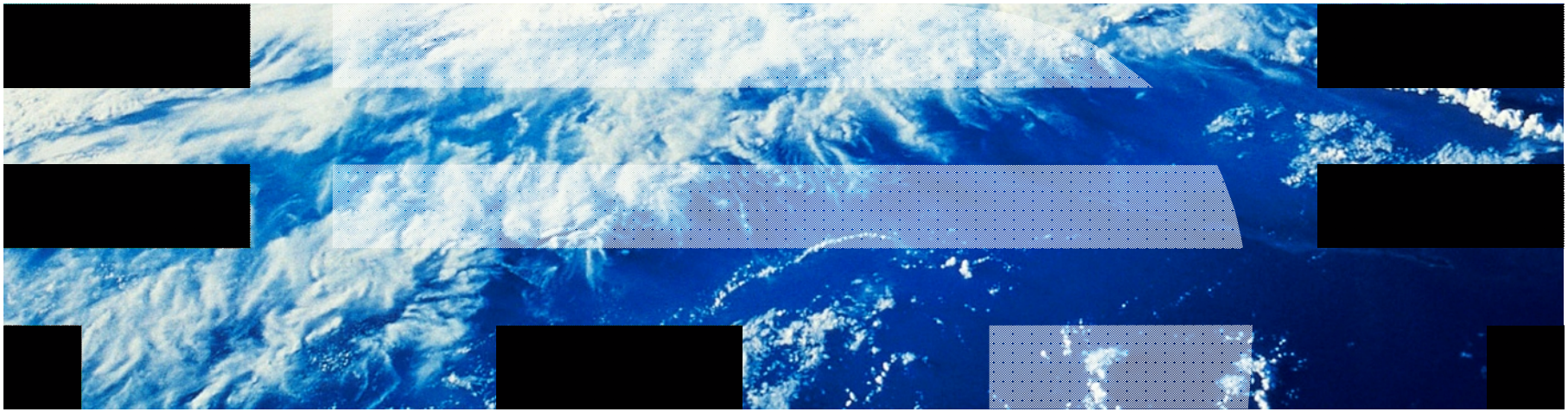
Cloud and Big Data

Sambit Sahu IBM Research



Course Objective

- **Graduate level course on Cloud Computing**
  - Focus is on learning and building extremely large scale systems and applications leveraging Cloud.
  - Learn concepts as well as hands-on experience by using real cloud and cloud technologies.
  - Three key objectives: learn how to use a cloud, leverage cloud to build applications, build your own cloud.
- **We shall learn cloud technologies by using real clouds - Amazon AWS, Google App Engine and Openstack cloud and Hadoop platform.**
- **Required background**
  - Programming experience with one of the following Java/Python/Ruby – Java preferred, web services basics
  - Operating Systems concepts, networking concepts would help you understand more

Four Main Components

- **Cloud Programming**
  - Basic cloud concepts
  - Amazon AWS cloud and services
  - Google App Engine
  - ***Build a large application leveraging cloud***

- **Virtualization**
  - Enabling technology
  - Various virtualization concepts, tools, how-to
  - VMWare ESXi

- **Openstack and Private Cloud**
  - Understand key concepts for building a cloud
  - Use Openstack cloud management stack
  - devops/chef/puppet for private cloud automation
  - ***Build your own cloud***

- **Big Data Platform and Programming**
  - Hadoop eco-system, noSQL database
  - Case study with various well known large scale platform components – FB Cassandra, LinkedIn Voldemort, Google GFS/BigTable,Yahoo Zookeeper
  - Hadoop programming – Map Reduce/Pig/Hive
  - ***Real-world use cases i.e., Twitter sentiment analysis using Hadoop***

Tentative Schedule details

- Tentative Topics and Schedule

1. IaaS/Paas Cloud (AWS and Google App Engine based)
   - Introduction to Cloud: on-demand and elastic infrasturcture
   - Learning IaaS Cloud through AWS Cloud and programming
   - AWS Cloud Services
   - Building and deploying applications in Cloud
   - PaaS Cloud and Google App Engine
   - Netflis OSS and Google Mobile End-point as Cloud Service Examples

2. Cloud and Virtualization
   - Virtualization basics
   - Virtualization and On-demand infrastructure
   - Virtual Appliance, Application images for fast provisioning
   - VMWare vSphere/vCloud, KVM technologies

3. OpenStack and Private Cloud (OpenStack and Chef)
   - OpenStack architecture details
   - Openstack installation concepts
   - Design a private cloud using OpenStack

4. Big Data: Hadoop Eco-system (Hadoop, Cloudera/Horton, EMR)
   - Scalable Cloud Storage and noSQL database
   - HDFS concepts, design and API details
   - Map Reduce concepts and Programming
   - Hadoop administration, cluster design
   - Hive, Pig, Zookeeper
   - Twitter data analysis as a use case

What would you learn in this course…

- HowTo
  - What is a Cloud?
  - How to use a Cloud as a compute node?
  - How to use Cloud programming to use/extend Cloud capability?
  - How to develop new capabilities/services using Cloud?
  - How to build your own cloud using open source?
  - How to use cloud for big data applications?

- Concepts
  - Virtualization
  - Ultra-Large scale file system (GFS, HDFS,..)
  - Large scale systems (Big Table, Chubby, ZooKeeper,..)
  - Large scale storage (Google Storage, Facebook/Cassandra, etc.)
  - …..
- Case studies with real systems/cloud
- Compute Cloud, Storage Cloud, Data Cloud

Course Structure

- **Lecture Structure**
  - Each lecture will have a theme topic. First 1 hour 30 minutes lecture and demonstration by the instructor.
  - Last 30 minutes students present papers from the reading list, discussion and some days quiz.

- **Assignments/Exercises**
  - Reading list – consists of 12-15 land mark papers in the area of large scale systems (Google File System, Map Reduce, Hadoop, Facebook Cassandra, Yahoo Zookeeper, Amazon Dynamo, HBase etc.)
    - Present one paper in the class from the reading list
    - Submit reading summaries for all papers from reading list (about 15 papers)
  - Programming Assignments
    - There would be 4 programming assignments.
  - Final Project
    - Conduct in a group of 3-4 students.

- **Every week**
  - There is a class! I post the lecture material in the course page.
  - There are typically 2 papers for the reading and presentation. Students submit paper reviews before coming to the class.
  - There may be mini-HW – very small exercises that you need to complete and submit.

Grading and requirements

- Class participations (presentation and paper critics/quizzes) -- 30% grade
  - Presenters present the paper
  - Others need to submit a critic for the paper – due on the date the paper is presented
  - There will be standard template for presentation and critic
  - Mini-HW: Non-programming assignments (written assignments)
- Assignments – 30% grade
  - 4 homework stressed on technologies and programming
- Course project -- 40% grade
  - Students may team up
  - Two choices
    - Project: come up with a project idea and carry out the project
    - Research report: choose 10 papers in a particular topic and turn in a detailed report
- Submission process – TAs would email the details

Project: Learn how to innovate in this space

- **Objective is to learn how to innovate in this space**. I would want you to act (*at least pretend*!) as if you are forming a startup.
  - Read few book: **(i) Crossing the Chasm, (ii) The Lean Startup**

- **Four phases to your project**
  1. Concept and business idea
  2. Technology viability and architecture
  3. Execution planning and prototyping
  4. Demo, socialization and review

- **Few suggestion**
  - Don't procrastinate – start early. Motivation: Would help you get A+ (and earn millions!)
  - Form your team carefully – asking, interviewing your team mates. Float around some ideas,, kick the tire. Take a look at lot of recent startups that are bought by Google, Apple, FB, Amazon etc. **Take a look at beta.list**
  - Cloud + Social + Mobile is a good recipe for a perfect storm

What you need to do soon

- Sign up for paper presentations
  - List of papers will be made available by end of the week
- Get account on few popular clouds
  - Amazon AWS (EC2, S3)
  - Google App Engine, Google Storage
  - We are working with Amazon to get free accounts
- Course Project
  - Substantial portion of your grade depends on final course project
  - I shall provide sample project ideas/topics
  - You need to have a team and a project topic by end of one month.
- New Experiment this time…run project team like a startup!
  - Team Composition: Data, mobile/social, UI, backend, machine learning, app
  - Use Hackathon for ideation to rapid prototyping

Reference Books

- Too many topics – so not any particular good book. So attend lectures, read my mind (hopefully the good side of it!), learn from using real cloud and code (a lot of it actually!), and Google!
- Reference books
  - Amazon EC2 Programming
  - Hadoop in Action
  - Hadoop the Definitive Guide
  - Hadoop Operations

What is Cloud?

- Allows users to request computing/storage resources through web interfaces
- You do not need to own or install or manage these resources.
- Pay as you go - Resources on-demand
- Elastic: Use as much as you want or as less as you want
  - Users can assume infinite amount of compute and storage resources are available.
  - Users can request resources when and what they need and release/remove resources when they don't need.
- Compute and storage resources are now treated as software entities. You get access to such resources programmatically – not by physical hardware anymore!
- So what are the Clouds! Where are the Cloud?

- Read this paper: http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext

Why Cloud?

- You can get as many as 1000 machines for an hour for a few dollars to run a complex application!
- You don’t need to manage, maintain or fix any machines!
- You can use as little as 1 machine or as many as 10000 machines depending on what your current needs are!
- Two key focus: on-demand and elastic!

Essential Characteristics

- *On-demand self-service.* A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- *Broad network access.* Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- *Resource pooling.* The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- *Rapid elasticity.* Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- *Measured Service.* Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

Service Models

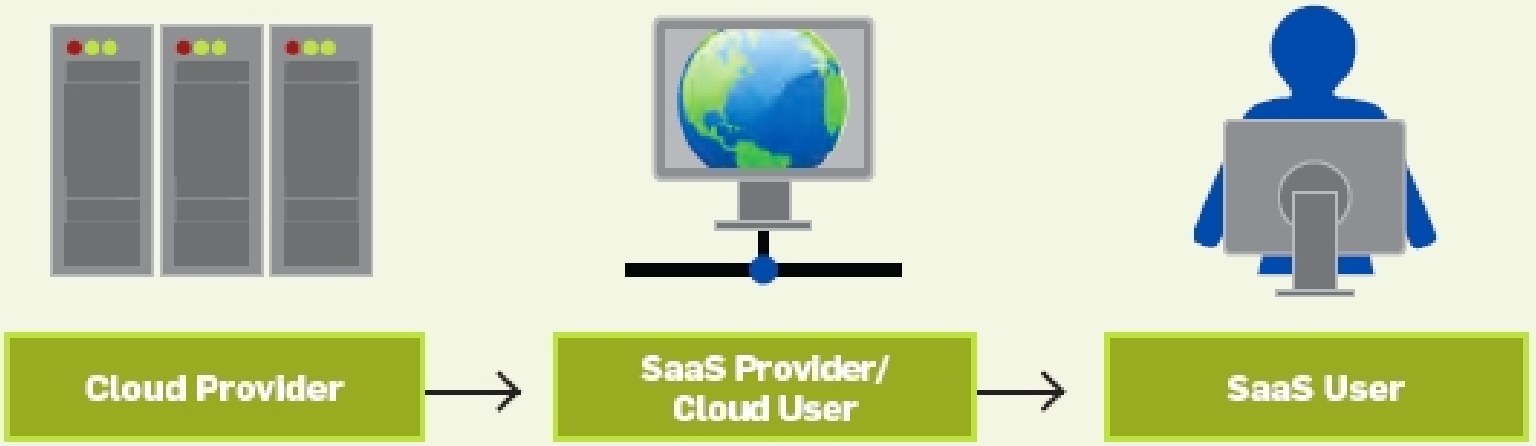
- *Cloud Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.
- *Cloud Platform as a Service (PaaS)*. The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.
- *Cloud Infrastructure as a Service (IaaS).* The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

Deployment Models

- *Private cloud.* The cloud infrastructure is operated solely for an organization. It may be managed by the organization or a third party and may exist on premise or off premise.
- *Community cloud.* The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be managed by the organizations or a third party and may exist on premise or off premise.
- *Public cloud.* The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
- *Hybrid cloud*. The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

Berkeley View of Cloud Definition

Figure 1. Users and providers of cloud computing. We focus on cloud computing's effects on cloud providers and SaaS providers/cloud users. The top level can be recursive, in that SaaS providers can also be a SaaS users via mashups.



- IaaS → SaaS Provider -→ SaaS User

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

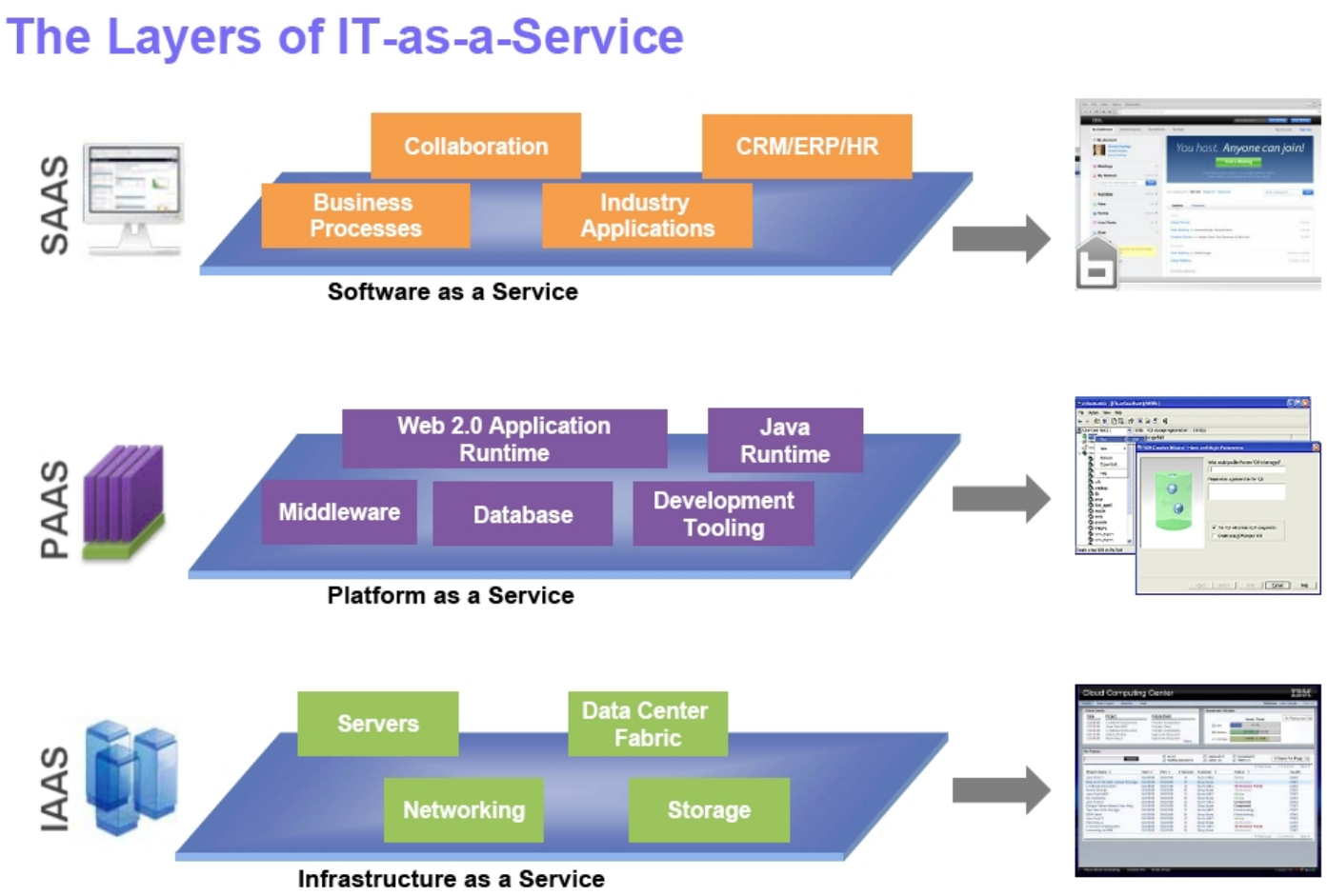
Different types of utility model

- IaaS Cloud (Amazon EC2)
  - Low level of computing resource abstraction
  - Provides a (virtual) machine to users
  - Makes it hard for IaaS providers to support automatic scaling, failover etc.

- Google AppEngine
  - Targeted at web applications
  - Enforces an application structure
  - Clean separation between stateless and stateful storage tier
  - Benefit: makes it possible to handle auto-scaling, fail over/high availability

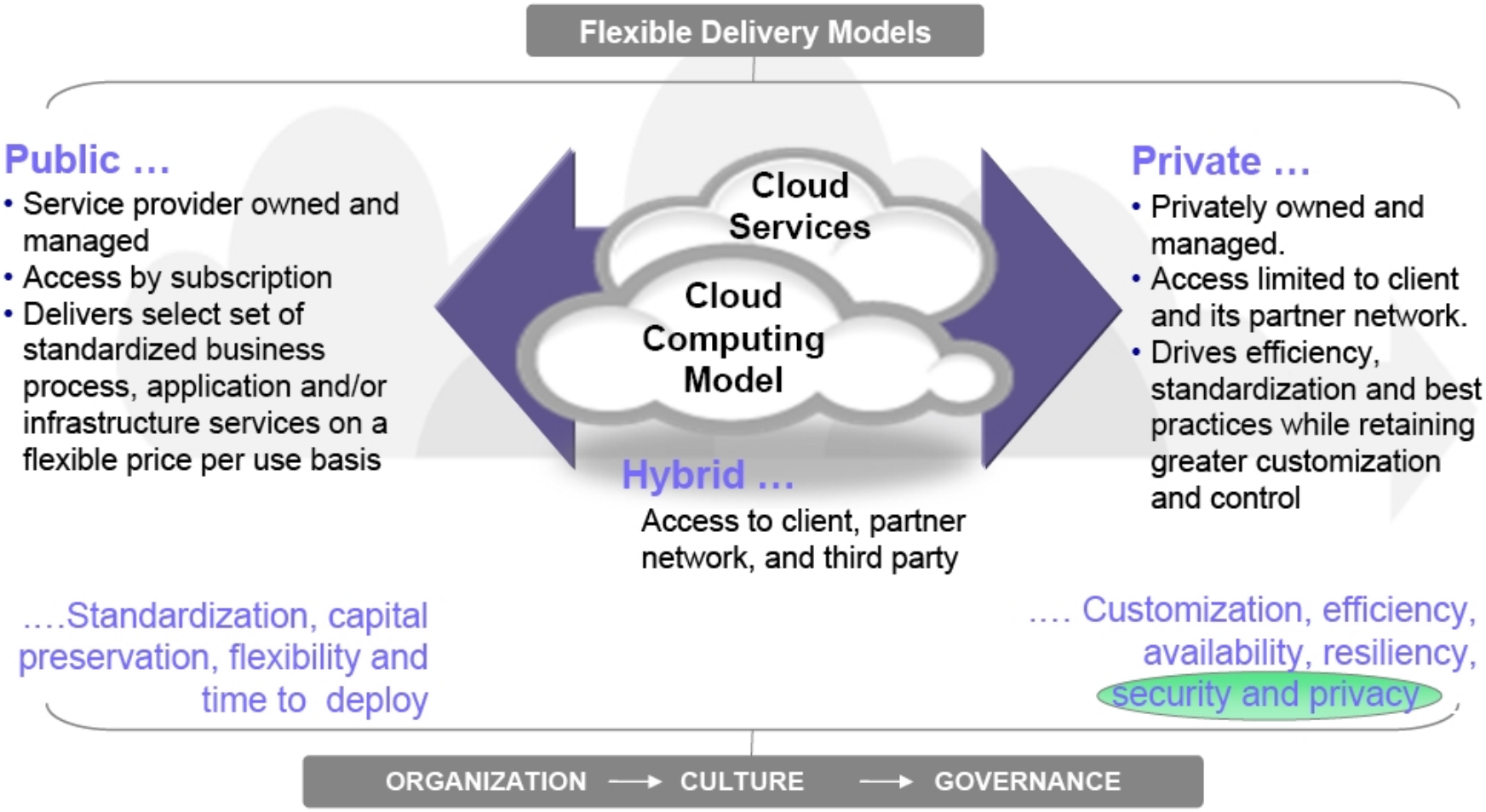
- Microsoft Azure
  - Applications need to be written using .NET libraries
  - More flexible than Google AppEngine
  - Able to provide some automated scaling
  - Between Application framework and hardware virtual machines

Different Cloud Offerings: A Layered Perspective



- Higher the stack, less control but more automation for user
- Lower the stack, more control but more responsibility for user

Cloud Computing Delivery Models



...service sourcing and service value

Example Clouds and Usage Scenario

- IaaS
  - Amazon EC2, Rackspace
- PaaS
  - Google AppEngine
  - Microsoft Azure
- SaaS
  - salesforce.com
- Roll your own
  - Open Source software stack
    - Open Nebula
    - Eucalyptus
    - Openstack

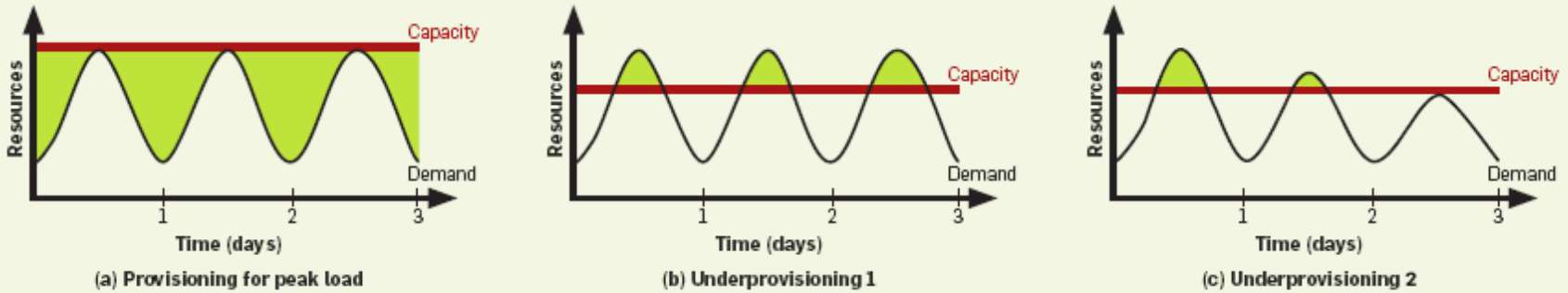
- Machine level abstraction
  - User requests a machine with desired CPU, mem, disk possibly with a preconfigured OS and software
  - IaaS Cloud provides a virtual server with (minimal) pre-installed software such as OS
- Platform level abstraction
  - User writes application using PaaS defined interfaces
  - PaaS provides platform to support the deployment and management of this application
- SaaS
  - salesforce.com
- User installs and adapts to build own Cloud

Cloud Computing Economics

- Three useful usage scenarios
  - Load varying with time
  - Demand unknown in advance
  - Batch analytics that can benefit from huge number of resources for a short time duration

- Why pay-as-you-go model makes sense economically even if costs higher than buying a server and depreciating the h/w
  - Extreme elasticity
  - Transference of risk (of over provisioning)

Figure 2. (a) Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Underprovisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Underprovisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press result in a permanent loss of a portion of the revenue stream.



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

Top obstacles and opportunities for Cloud

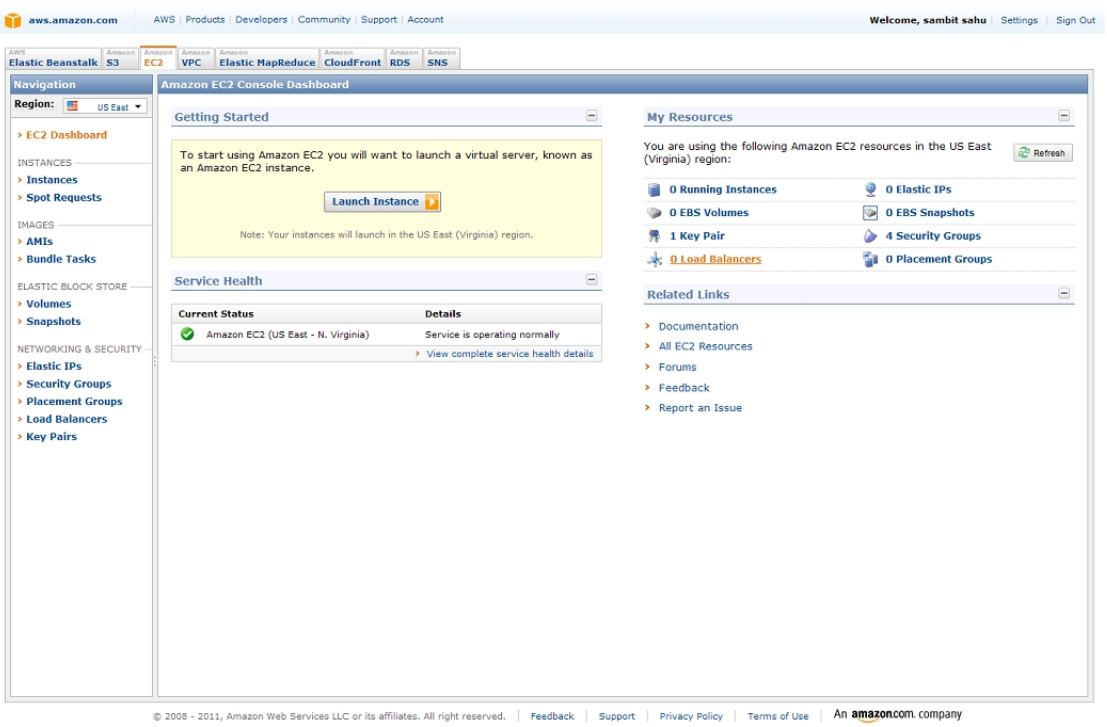
Table 2. Top 10 obstacles to and opportunities for growth of cloud computing.

	Obstacle	Opportunity
1	Availability/Business Continuity	Use Multiple Cloud Providers
2	Data Lock-In	Standardize APIs; Compatible SW to enable Surge or Hybird Cloud Computing
3	Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls
4	Data Transfer Bottlenecks	FedExing Disks; Higher BW Switches
5	Performance Unpredictability	Improved VM Support; Flash Memory; Gang Schedule VMs
6	Scalable Storage	Invent Scalable Store
7	Bugs in Large Distributed Systems	Invent Debugger that relies on Distributed VMs
8	Scaling Quickly	Invent Auto-Scaler that relies on ML; Snapshots for Conservation
9	Reputation Fate Sharing	Offer reputation-guarding services like those for email
10	Software Licensing	Pay-for-use licenses

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

IaaS Cloud Example: Amazon EC2

- Amazon EC2 provides public IaaS Cloud
- User uses a portal to request a machine with specific resource
  - CPU, memory, disk space
  - Pre-built OS and possibly middleware



PaaS Cloud: Google App Engine

- PaaS model
- Provides a platform to host web applications
- App Engine SDK for programming (Python and Java support)
- A set of primitives (datastore, URL fetch, memcache, JavaMail, Images, authentication..)
- User focuses on developing the application in this framework
- Once deployed, scaling, availability etc. are handled by Google AppEngine platform

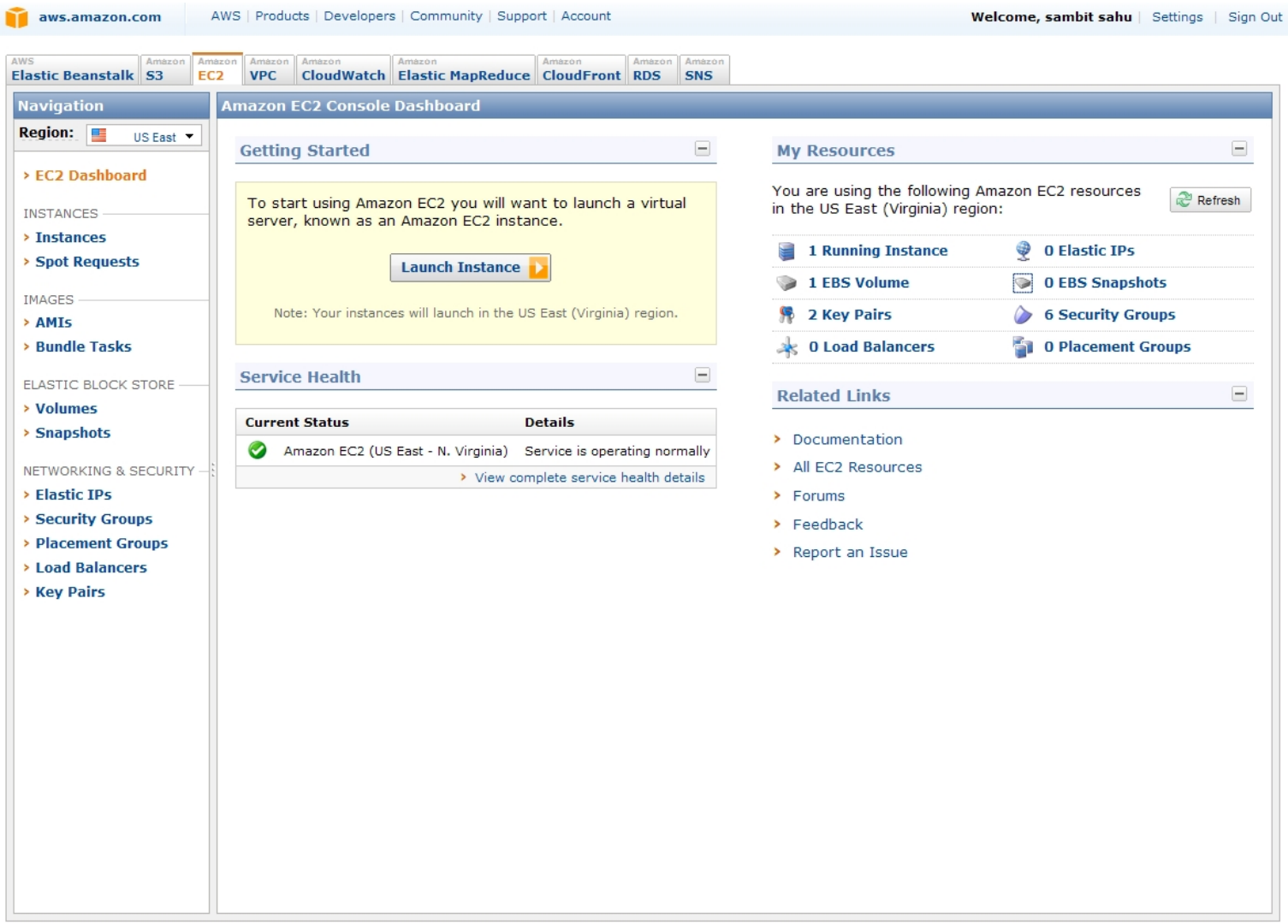
IaaS Cloud Basic Steps: Users Perspective

- Log into a portal
- Request a machine
  - CPU, mem, disk, OS details
- Cloud returns user machine address with login credentials
- User can securely login into the machine
- User may also be able to request a software stack to be pre-installed into a machine
- User may be able to load a pre-built stack into this machine on initialization

Let’s use a IaaS Cloud (Amazon EC2)

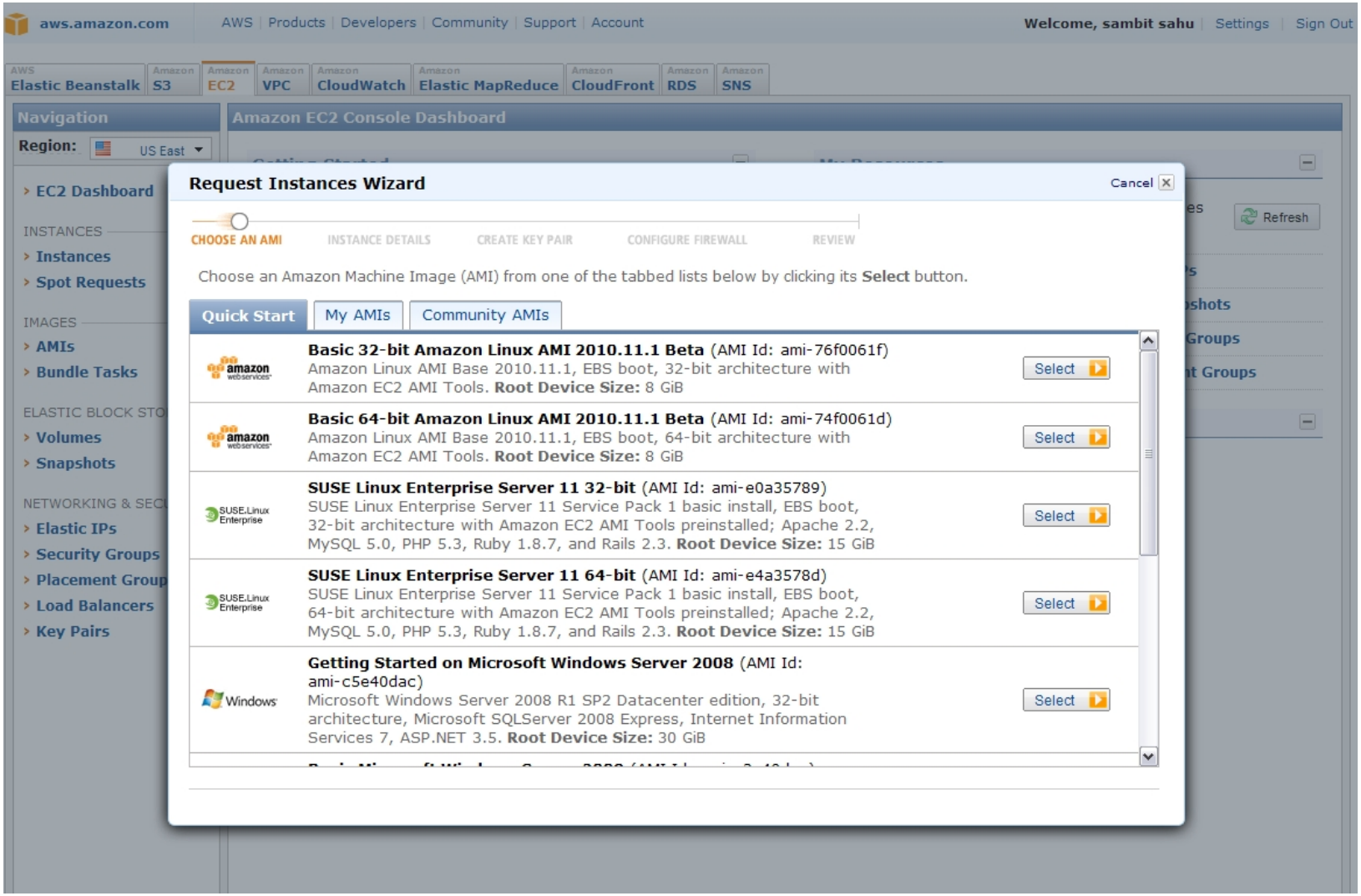
- http://aws.amazon.com/console/
- Amazon EC2 console based provisioning demo

Amazon AWS console (EC2 view)



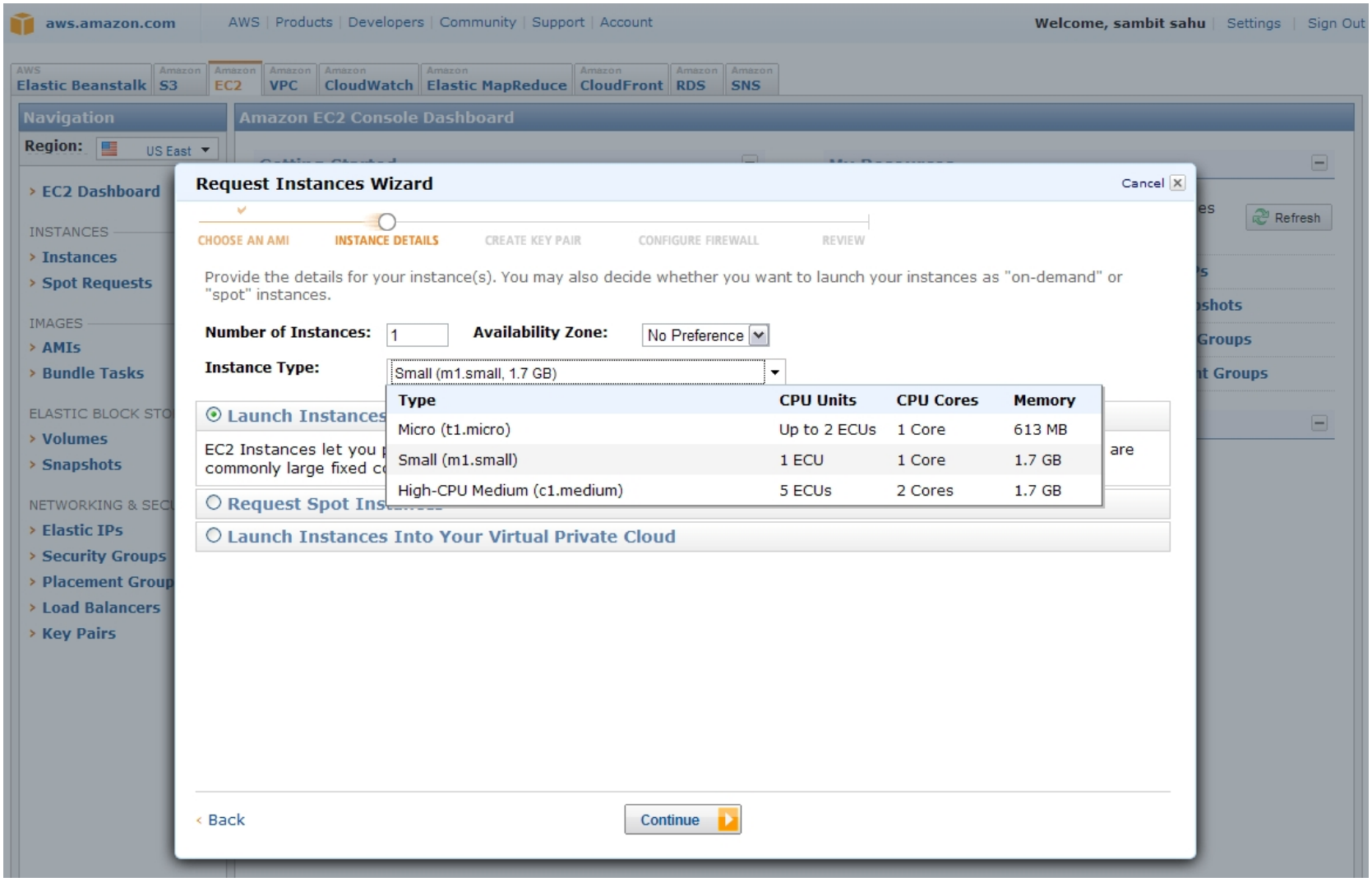
- User logs in with AWS credentials

User launches request instance → a list of prebuilt stack is provided



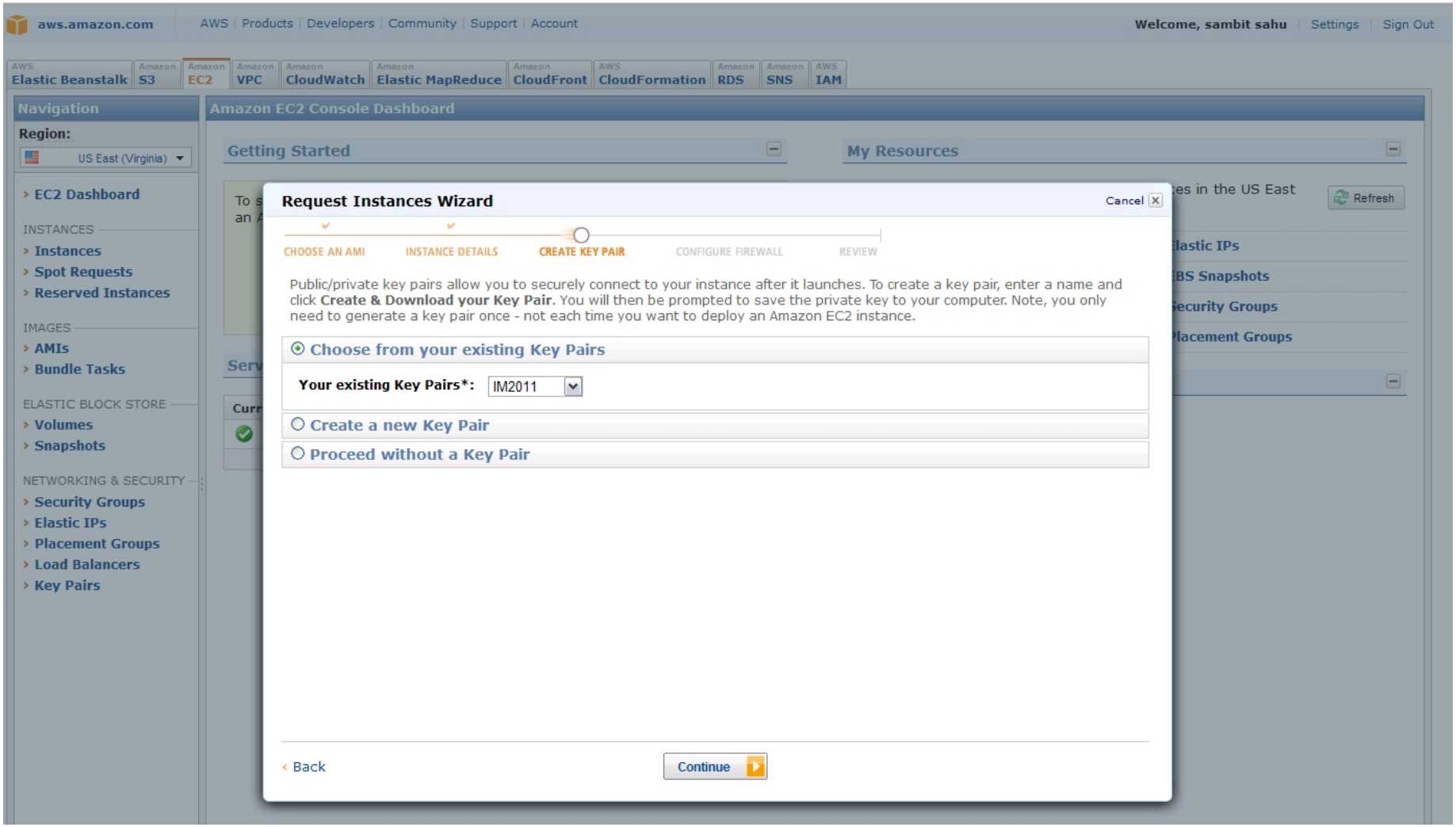
- AWS shows a list of available pre-built base software stack (**called Virtual Appliances**) user may request to add to the machine

User can choose the resource size (CPU, mem choices)

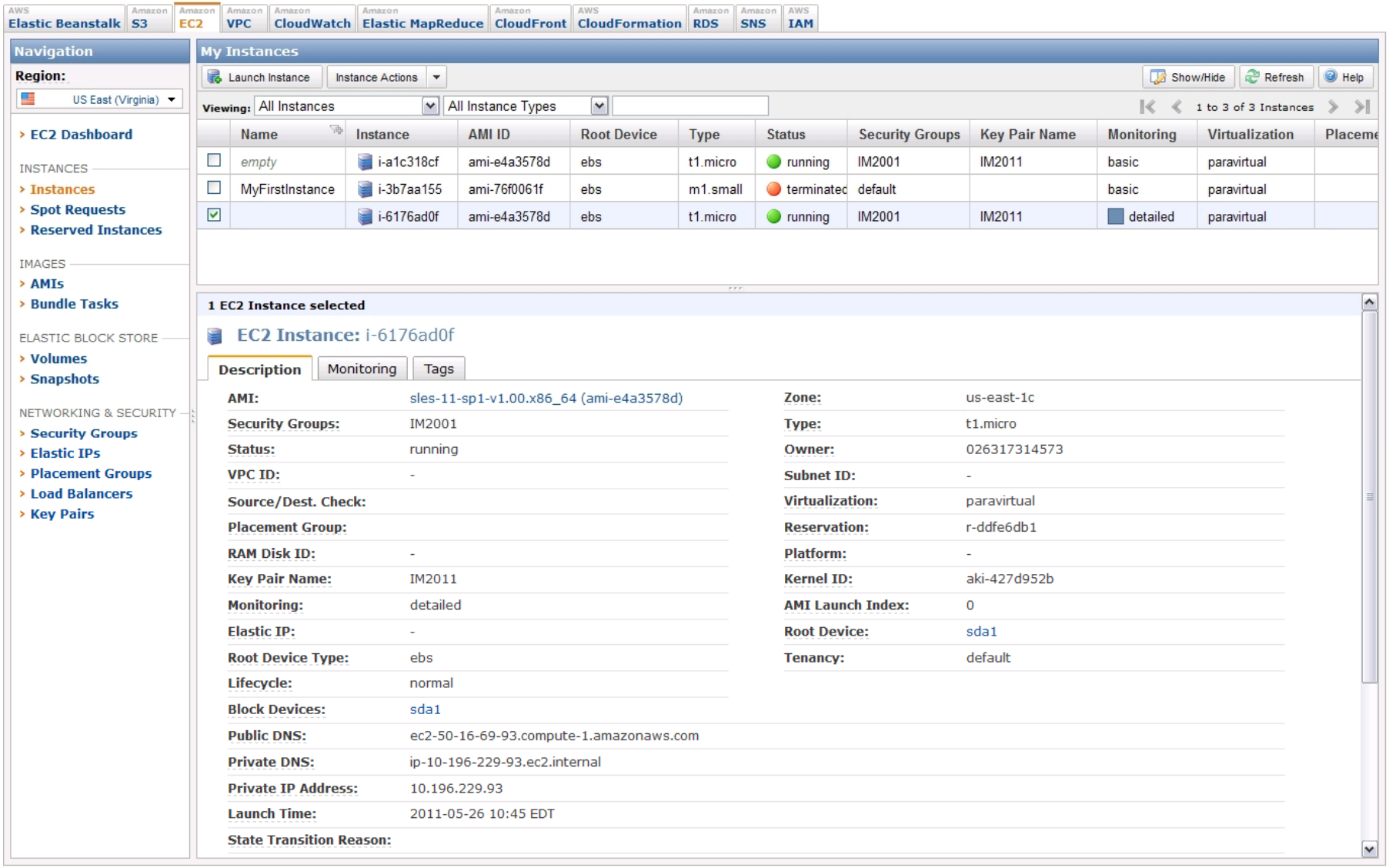


- Instance request wizard guides through resource choices

User specifies security/access configurations



AWS provisions an instance and returns user credentials



Next Week

- Reading List
  - A View of Cloud Computing
    - http://cacm.acm.org/magazines/2010/4/81493-a-view-of-cloud-computing/fulltext
  - CloudCmp: Comparing Public Cloud Providers
    - http://research.microsoft.com/en-us/UM/people/srikanth/data/cloudcmp_imc10.pdf

- Mini Homework 1.1
  - Sign up for AWS account. Sign up for AWS EC2 and S3 services.
  - Create a micro instance with Amazon Linux stack with appropriate keys and access control.
  - SSH into the instance you created. Take a screenshot and submit it. You submit in the courseworks under miniHW1 link in Assignments.

Some additional links

- https://aws.amazon.com/solutions/case-studies/
- http://aws.amazon.com/awscredits