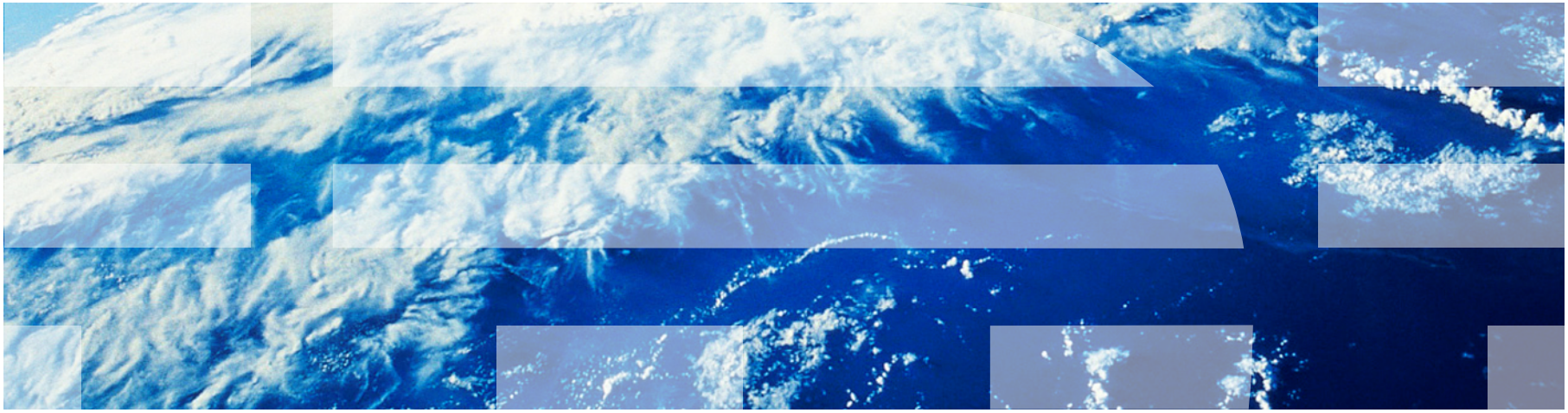
Cloud and Big Data

Sambit Sahu IBM Research



Course Objective

- **Graduate level course on Cloud Computing**
  - Focus is on learning how to *design, build, deploy and manage* extremely large scale systems and applications leveraging Cloud
    - Building blocks and design patterns in designing backend of typical Internet Scale application
  - 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 scalable intelligent systems
- **We shall learn cloud technologies by using real clouds and services - Amazon AWS, Google Cloud, Hadoop/Spark, Kafka, Elastic, Dynamo etc.**
- **Required background**
  - Programming experience with one of the following Java/Python, web services basics
  - Operating Systems concepts, networking concepts would help you understand more

  --> If you are not familiar with web services, take a look at materials on any web application design technologies.

What would you learn in this course…

- HowTo
  - How to use a Cloud as a compute node?
  - How to use cloud to design an Internet scale application?
  - How to process a very large amount of data?
  - How to build your own cloud using open source?
- Concepts: Building Blocks
  - Virtualization, Containers, Serverless
  - Peta-byte scale storage systems
  - Event and messaging systems (Kafka)
  - noSQL datastore (Cassandra, mongo, DynamoDB,…)
  - Elastic Search
  - Compute in a cluster
  - Intelligent AI applications
  - …
- Case studies with real systems/cloud
- Compute Cloud, Storage Cloud, Data Cloud

Main Modules

- **Cloud Platform and Programming**
  - Basic cloud concepts
  - Hands-on experience with Amazon AWS Cloud
  - Virtualization as an enabling technology
  - Virtualization vs Containers vs Serverless
  - ***Build a Web application leveraging cloud***

- **Building Blocks in an Extremely Large Scale Application**
  - Scalable data store and noSQL database
  - Message Queues: Kafka
  - Unstructured data and queries: Elastic Search
  - In-memory data store
  - devOps: Containers, micro-services, logging and monitoring
  - ***Build a scalable application using scalable, event-driven pattern***

- **Private Cloud** *(this module will be shortened)*
  - 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 Computing Platform and Programming**
  - Hadoop eco-system, and batch data processing & storage
  - MapReduce, Hive, Hbase
  - *Spark and Spark Streams*
  - ***Intelligent Real-time system design using Spark***

Course Schedule

Course Schedule

Week	Topic	Reading Paper	Deadline
01/29	Intro to Cloud and AWS		
02/05	Building Scalable Application in Cloud	GFS	A1 release
02/12	Cloud Services and Event driven design pattern	Big Table	Project Proposal Due
02/19	Virtualization, Container and Kubernetes	Borg	
02/26	Cloud DevOps and Containers	Zoo Keeper	A1 due and A2 release
03/05	Scalable Data Pipeline	Kafka	Project Task 1
03/12	Quiz 1	DynamoDB	
03/26	Message Queue	Map Reduce	A2 due and A3 Release
04/02	noSQL database	Spark RDD	Project Task 2
04/09	Computing in Cluster I: Hadoop	TBD	
04/16	Computing in Cluster II: Spark	TBD	A3 Due
04/23	Dataframes and Spark	TBD	
04/30	Quiz 2		Project Task 3
05/07	Intelligent Systems and Cloud		
05/14	Demo		Project Demo

Course Material

- **Lecture Notes**
  - Each lecture will have a theme topic. Lecture slides will be provided for each lecture. Additional reference materials will be specified.

- **Reading List**
  - A set of landmark papers in the area of large scale systems
  - You submit a paper summary by answering the provided questions.

- **Three programming Assignments**
- **A final Course project**
- **Reference Texts**
  - AWS in Action
  - Elastic Search in Action
  - Kafka Definitive Guide
  - Hadoop: The Definitive Guide
  - Learning Spark

Grading and requirements

- 2 Quizzes -- 25%
- Assignments – 35% grade
  - 3 homework stressed on technologies and programming
- Course project -- 40% grade
  - Students may team up
- Submission process – everything to be done using Courseworks and Github

Late submission policy: All the assignments and project deliverables should be submitted by the deadline. First late day will carry 3%, second late day 5% penalty and no submissions will be accepted after that. You will have **total** 3 grace days that you could use towards any late assignment submission.

Project: Learn how to innovate in this space

- **Objective is to learn how to innovate in this space**
- **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

- Get account on Amazon AWS
  - Get student discount/coupon through AWS Educate

- Course Project
  - Substantial portion of your grade depends on final course project
  - I will provide a set of project categories that you could choose from or come up with your own. But each project category will have a set of criteria that need to be demonstrated
  - You need to have a team and a project proposal by 01/26/21 6:00pm

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

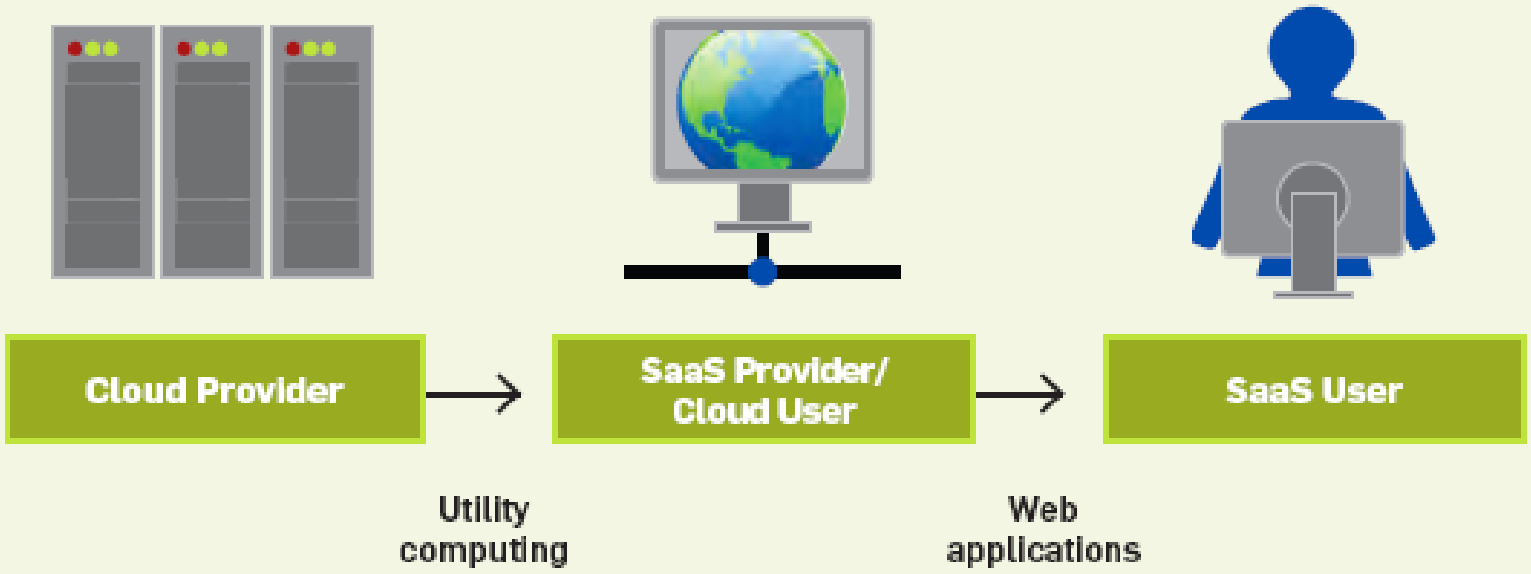
- *Cloud Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email). The consumer 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

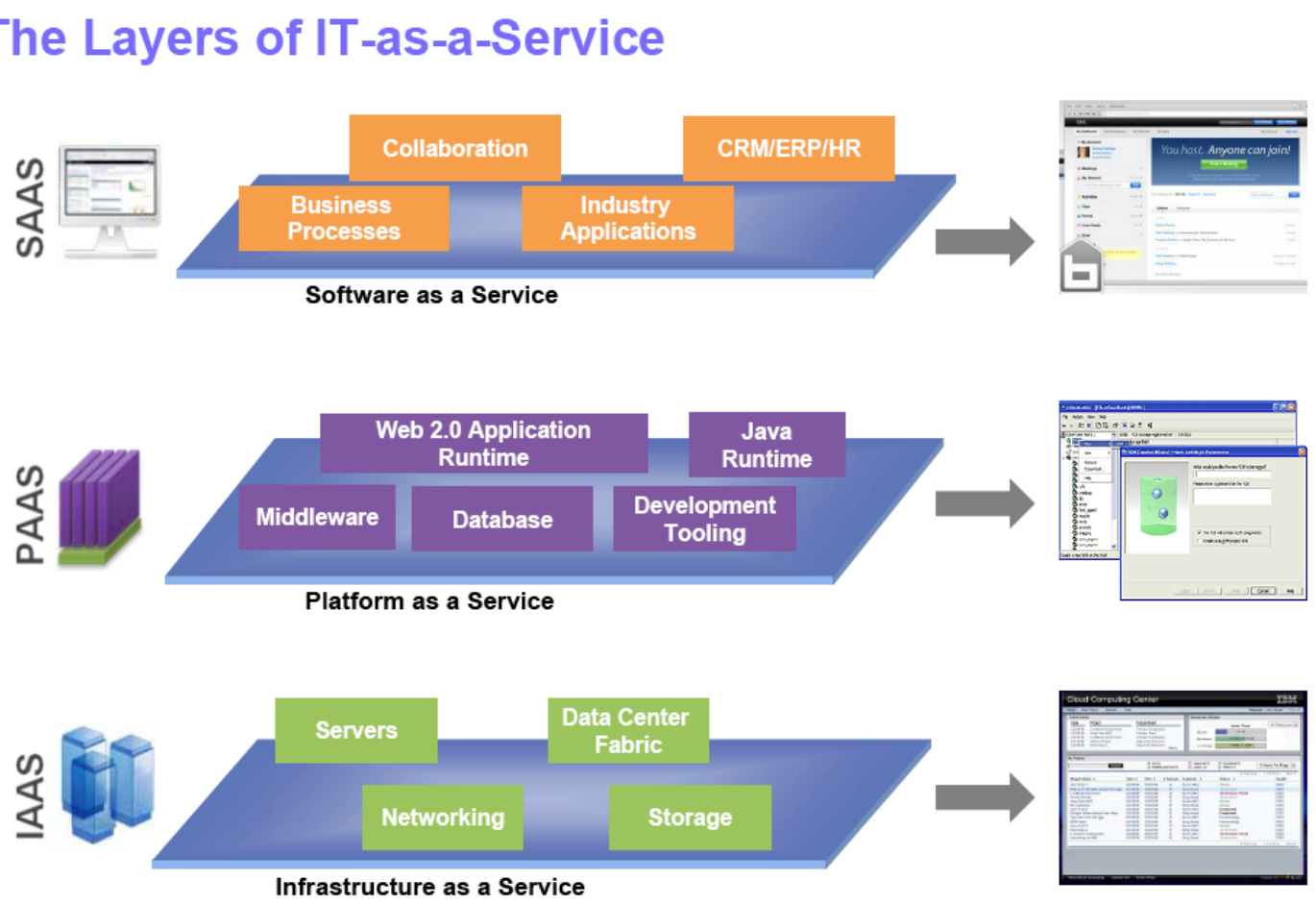
Different types of utility model

- 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



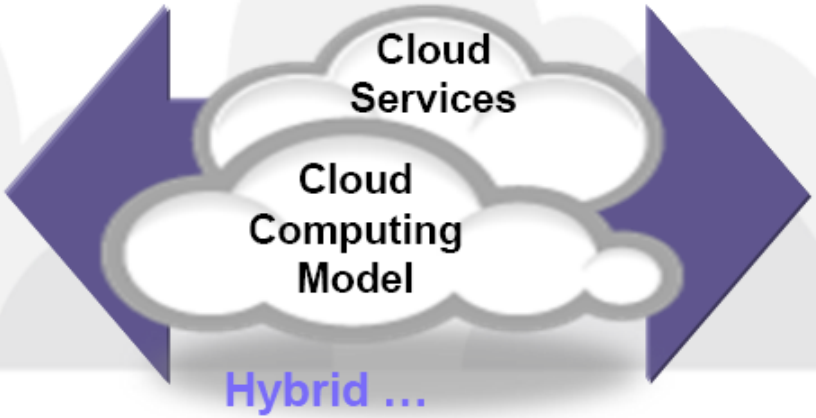
- Higher the stack, less control but more automation for user
- Lower the stack, more control but more responsibility for user

Cloud Computing Delivery Models

Flexible Delivery Models

Public …

- Service provider owned and managed
- Access by subscription
- Delivers select set of standardized business process, application and/or infrastructure services on a flexible price per use basis



Private …

- Privately owned and managed.
- Access limited to client and its partner network.
- Drives efficiency, standardization and best practices while retaining greater customization and control

Hybrid …

Access to client, partner network, and third party

….Standardization, capital preservation, flexibility and time to deploy

…. Customization, efficiency, availability, resiliency, security and privacy

ORGANIZATION → CULTURE → GOVERNANCE

...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

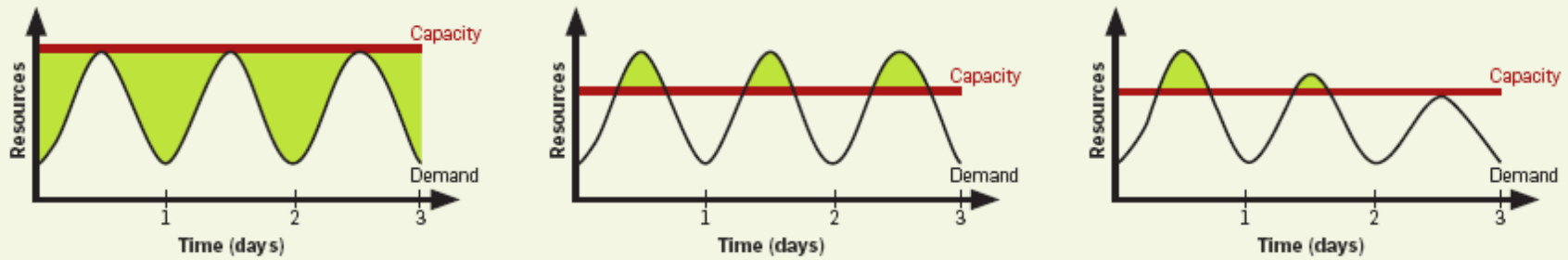
- Machine level abstraction
  - User requests a machine with desired CPU, mem, disk possibly with a preconfigured OS and software
  - IaaS Cloud provides a virtual server with (minimal) pre-installed software such as OS
- Platform level abstraction
  - User writes application using PaaS defined interfaces
  - 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

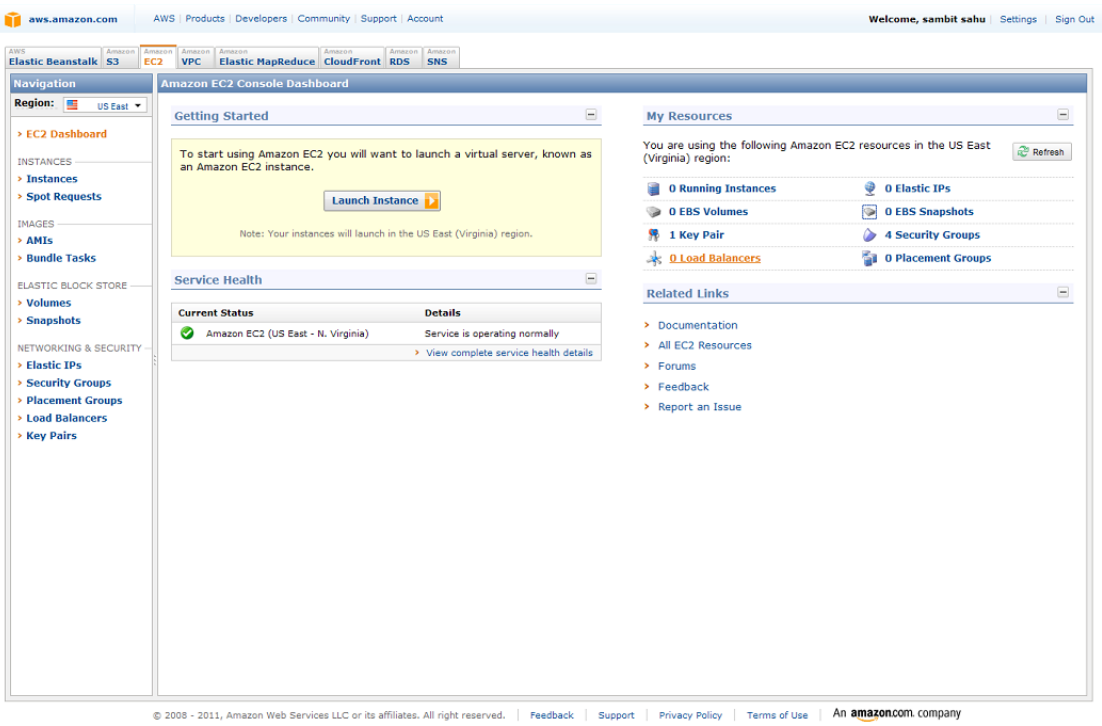
Table 2. Top 10 obstacles to and opportunities for growth of cloud computing.

	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



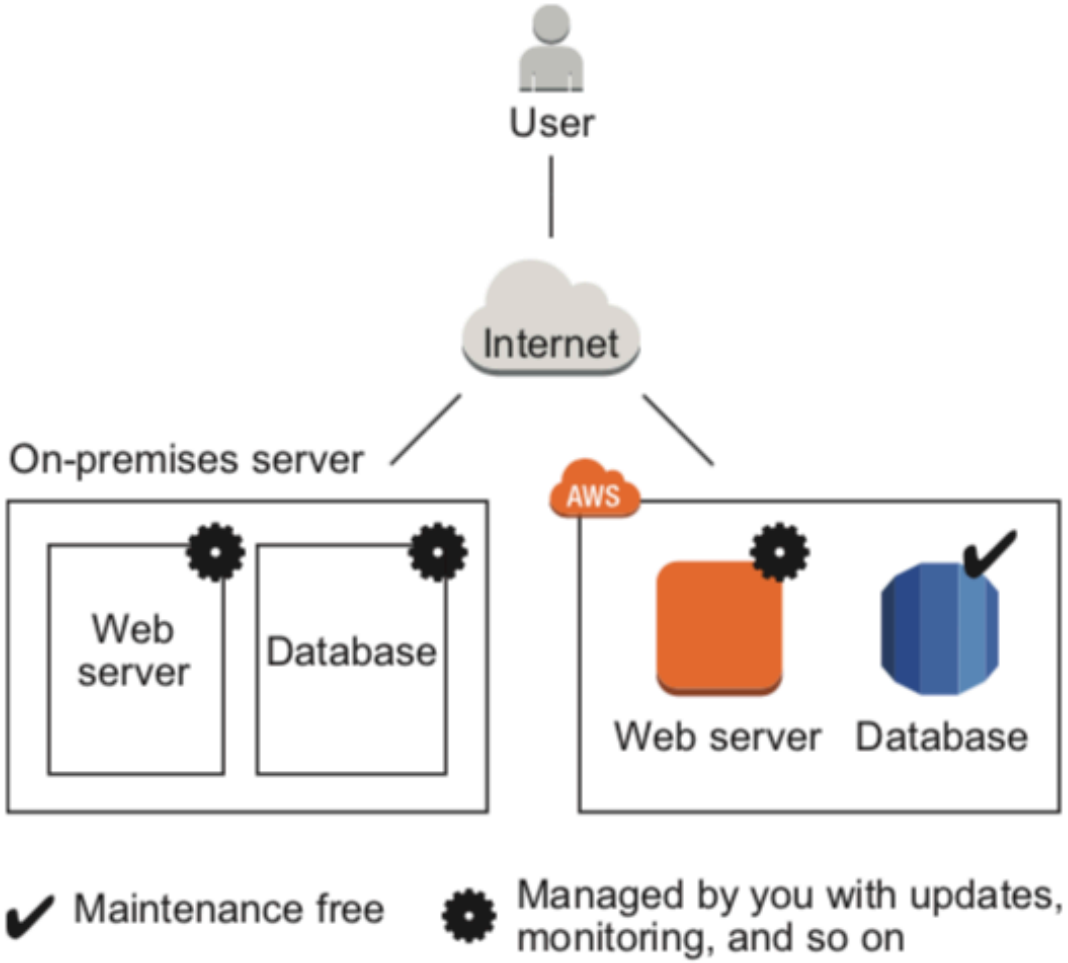
PaaS Cloud: Google App Engine

- 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

Let's use a IaaS Cloud (Amazon EC2)

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

Traditional vs Cloud-based Application



Leveraging Cloud Services to Quickly Build Complex Applications

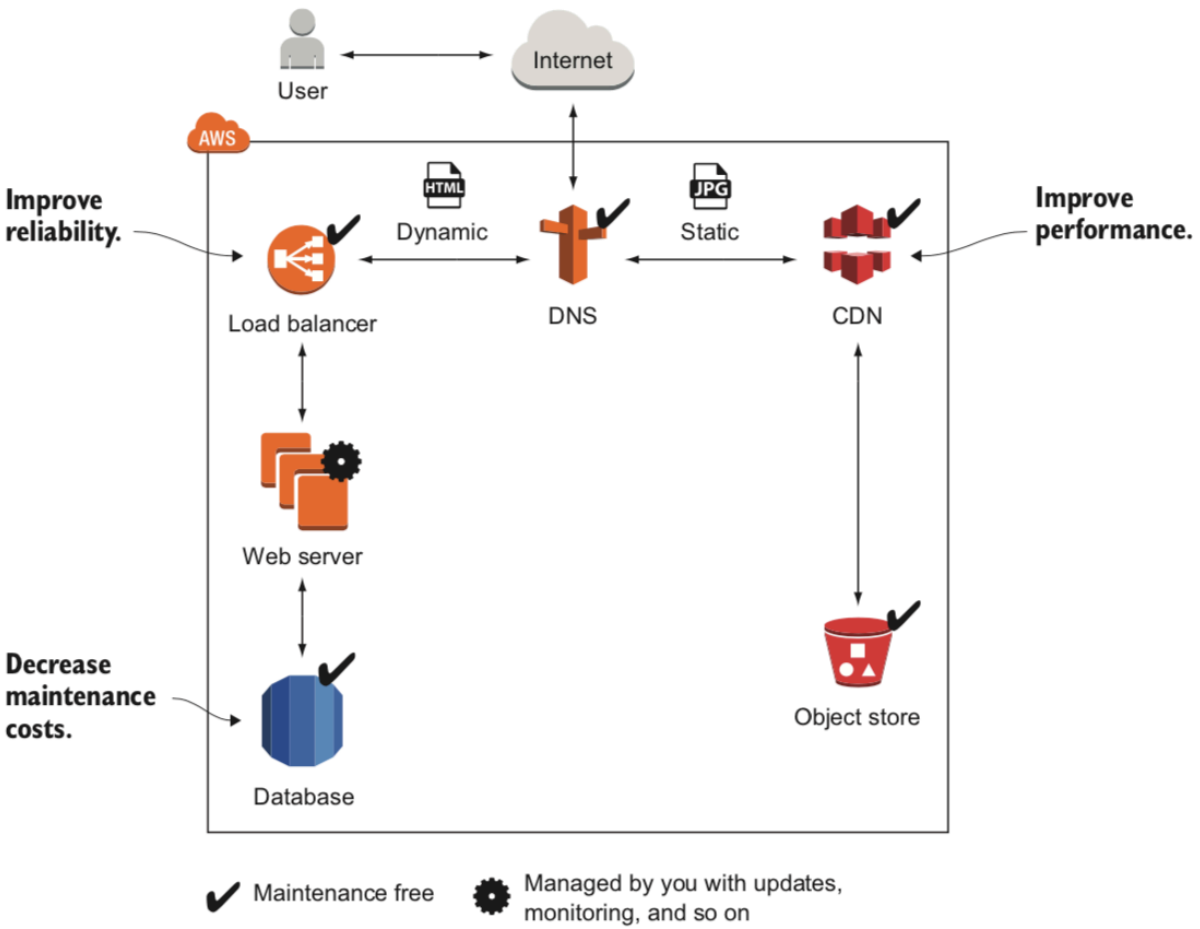
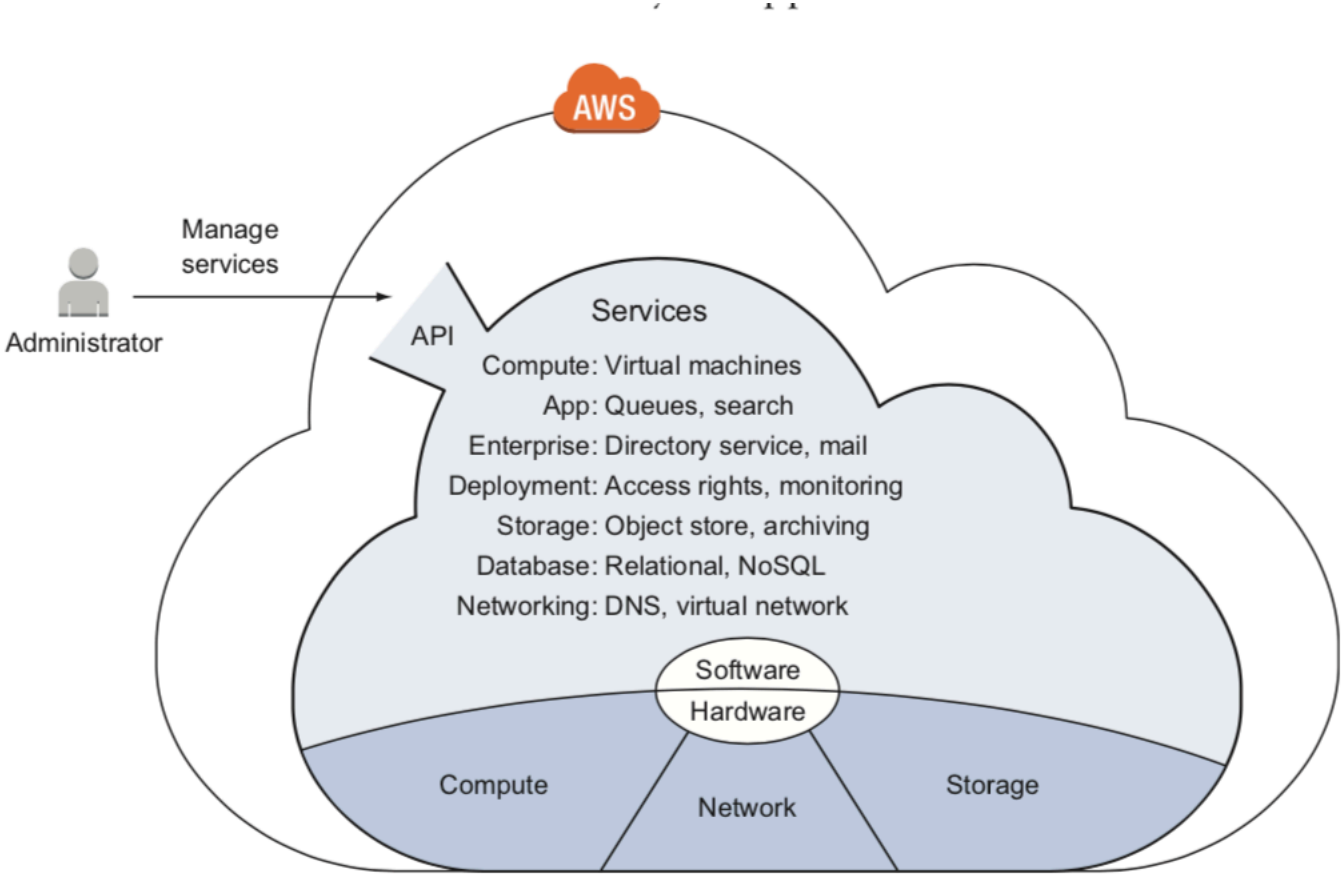
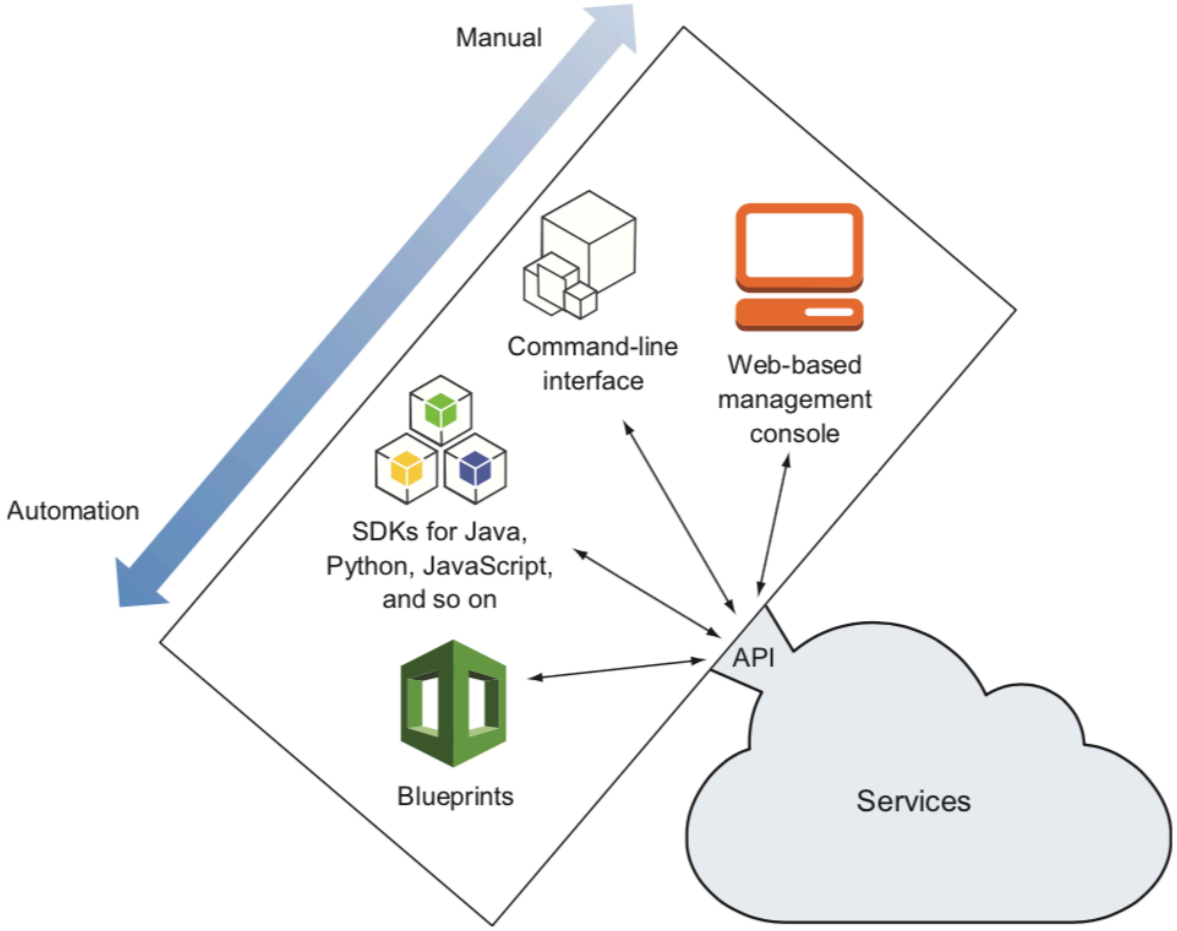


Figure 1.3 Running a web shop on AWS with CDN for better performance, a load balancer for high availability, and a managed database to decrease maintenance costs

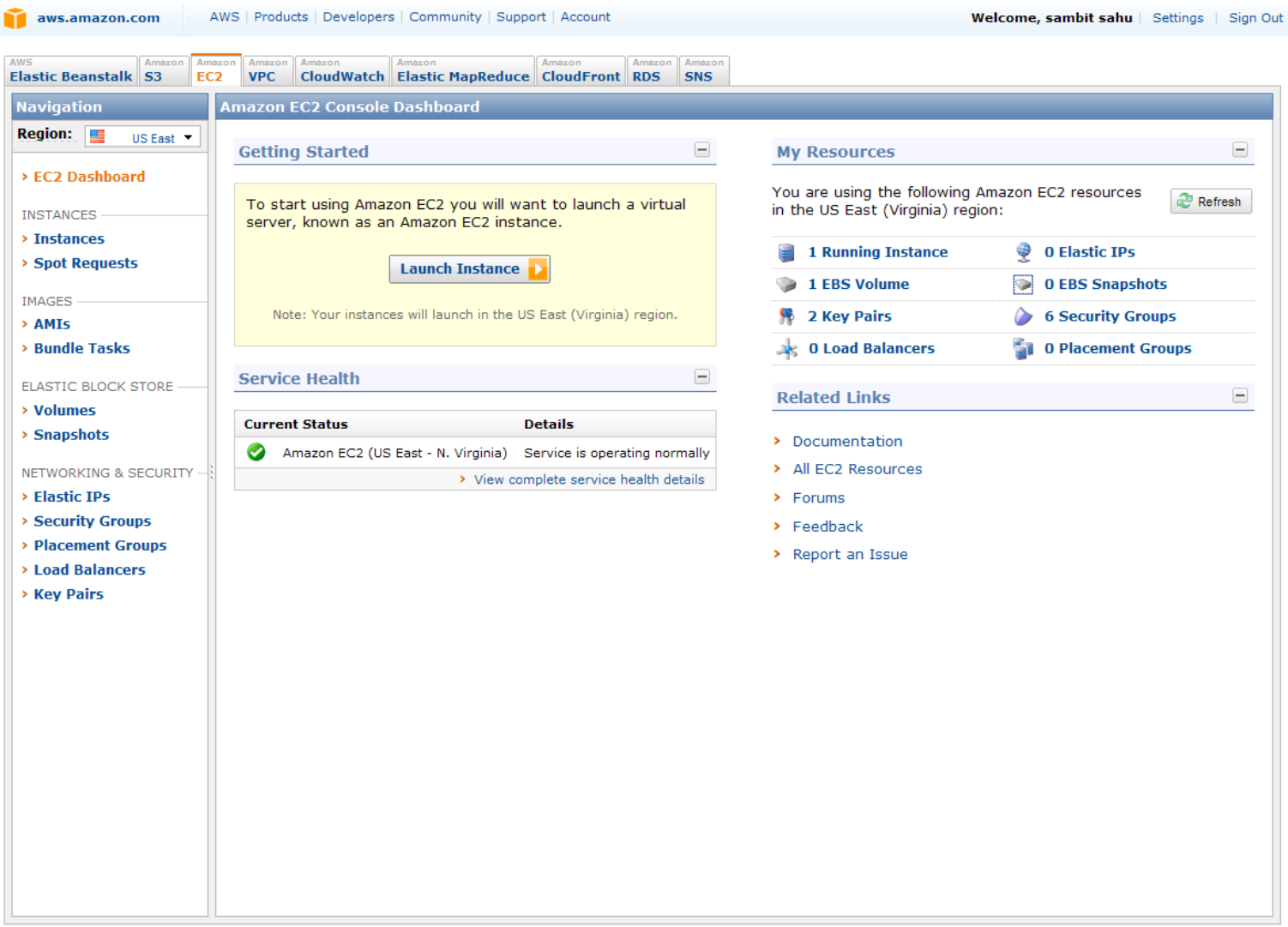
Amazon Cloud Services: Accessing through Web APIs



Various Methods to Access AWS

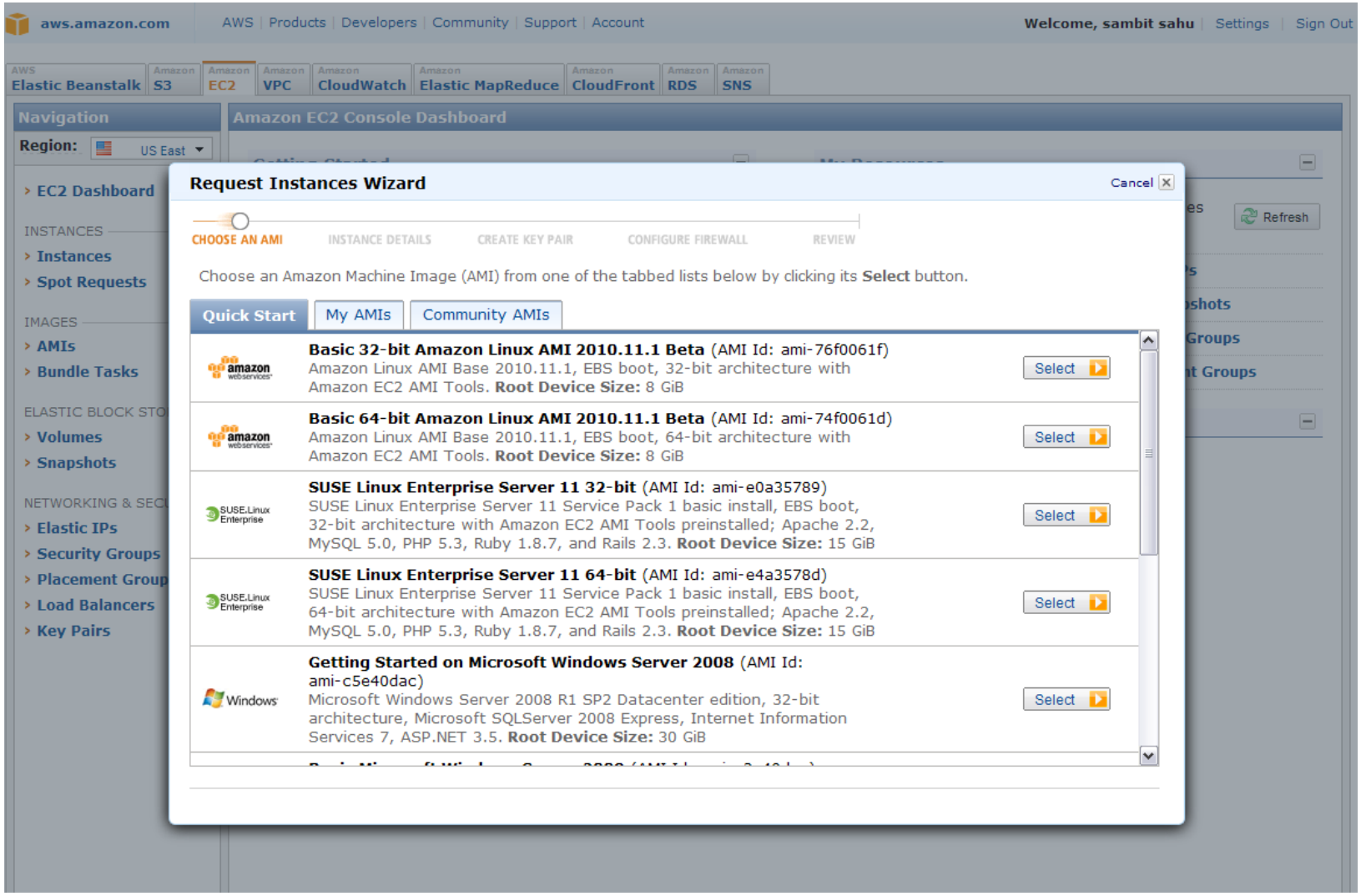


Amazon AWS console (EC2 view)



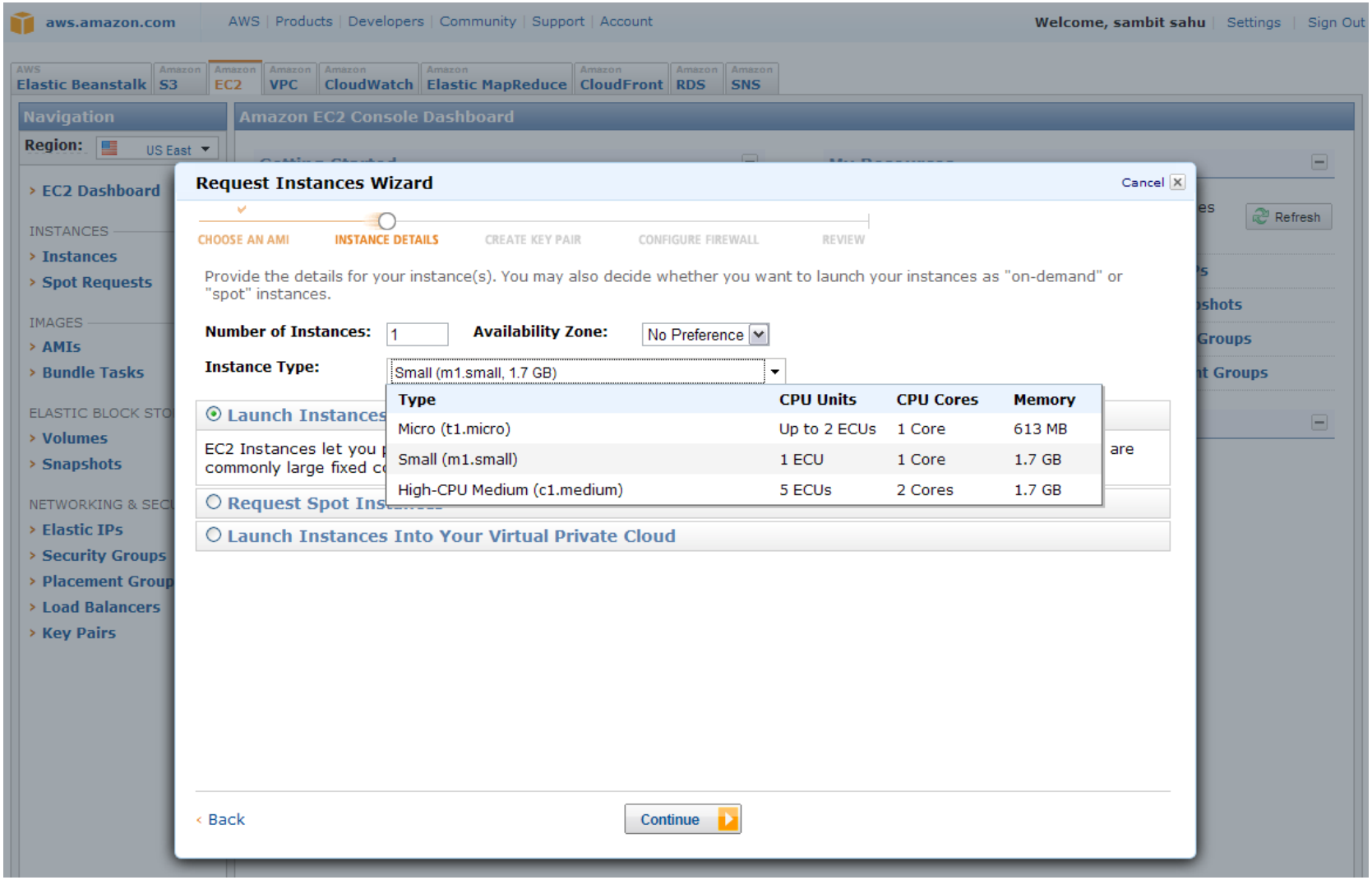
- User logs in with AWS credentials

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



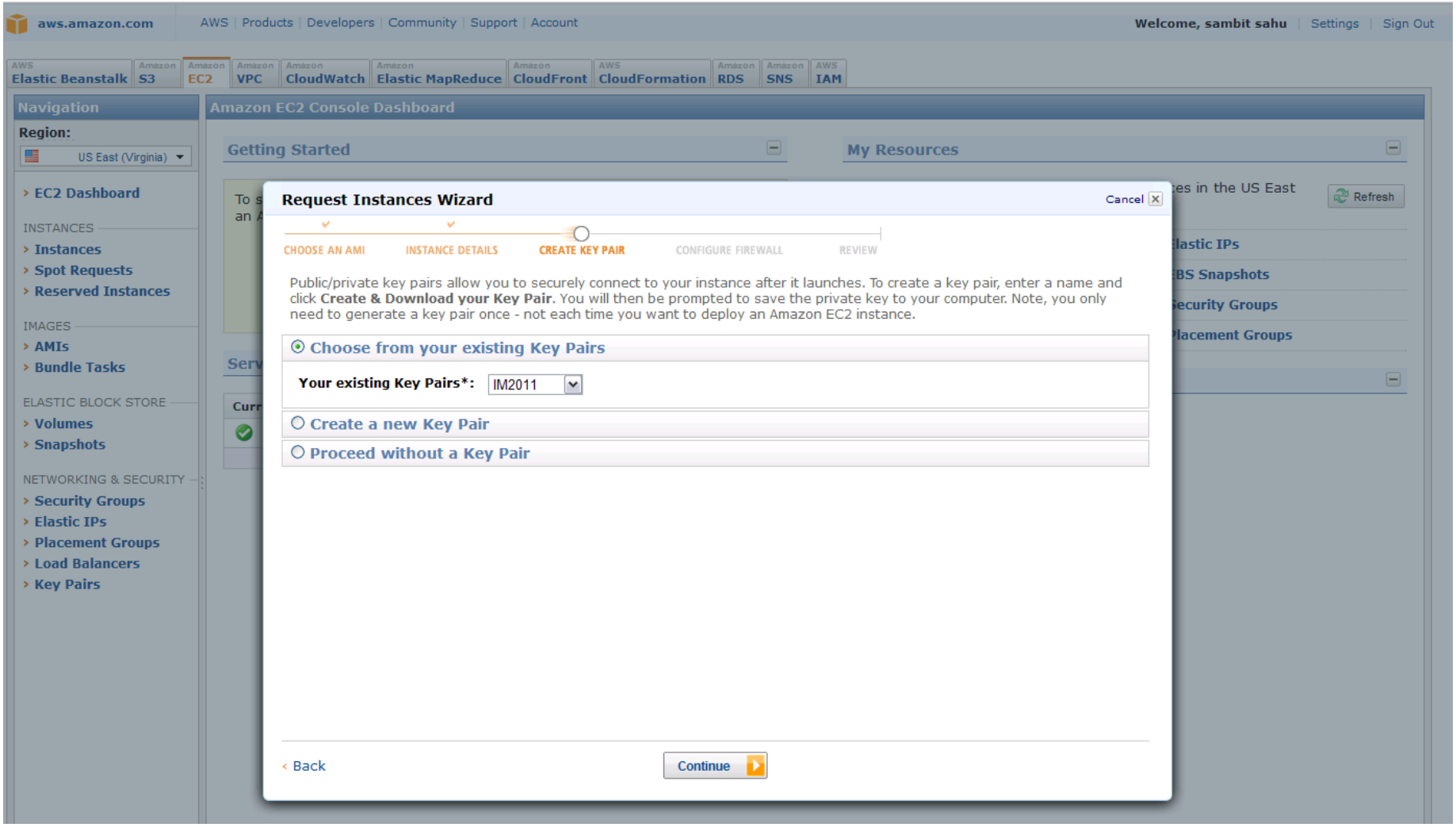
- AWS shows a list of available pre-built base software stack (**called Virtual Appliances**) user may request to add to the machine

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

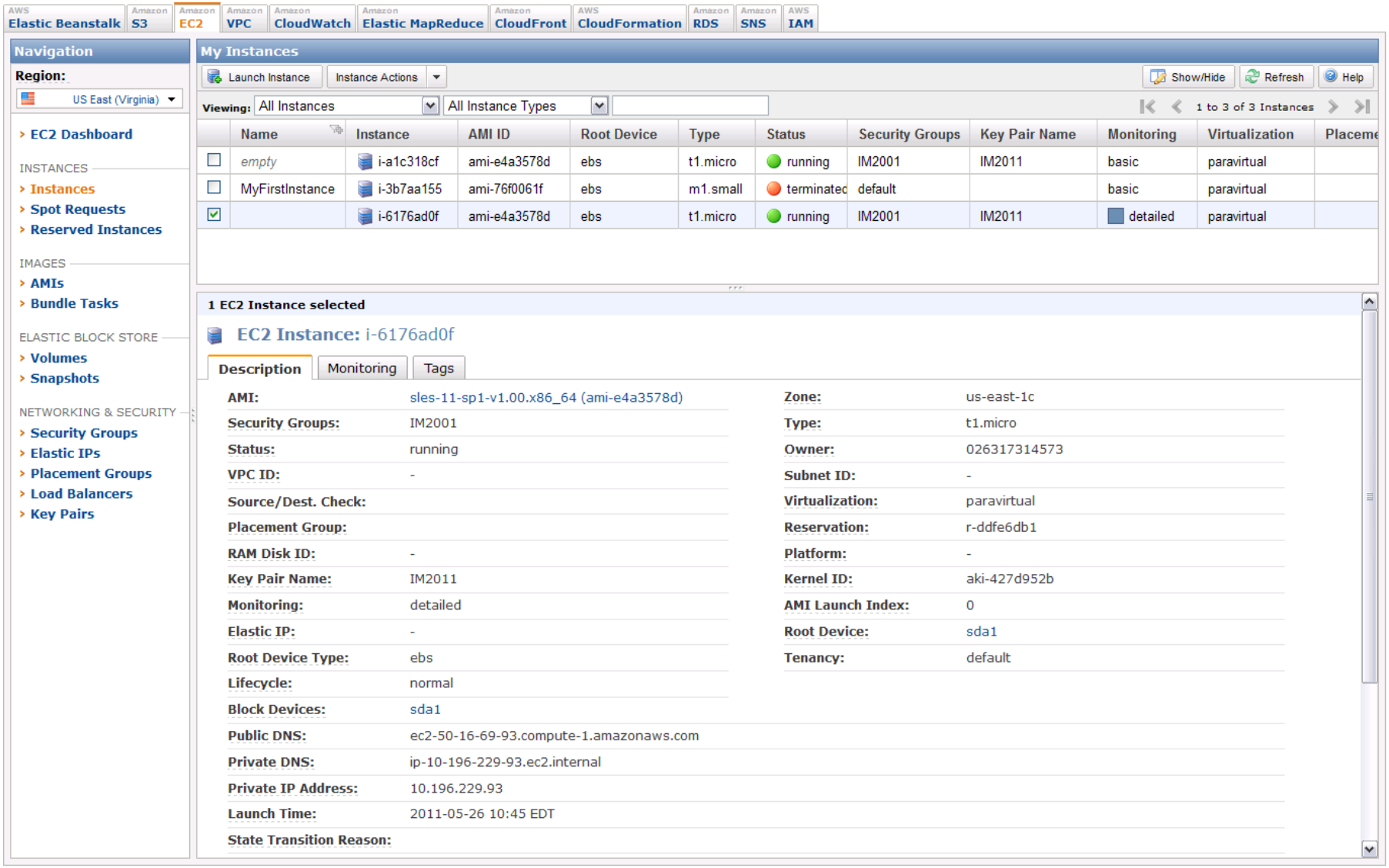


- Instance request wizard guides through resource choices

User specifies security/access configurations



AWS provisions an instance and returns user credentials



TODO This Week

- AWS Account setup and Webapp
  - Sign up for AWS account. Create a VM using EC2 UI console. Log into the created VM and make sure you look at the VM details such as IP address, AMI ID, and other credentials.
  - Complete the full stack webapp in the link: https://aws.amazon.com/getting-started/hands-on/build-web-app-s3-lambda-api-gateway-dynamodb/?e=gs2020&p=fullstack (Links to an external site.)

- You need to understand what is a webapp and various components in a webapp. Please refer to Lecture 0 for that.
  - Complete HW0 to build a webapp if you have never built one.