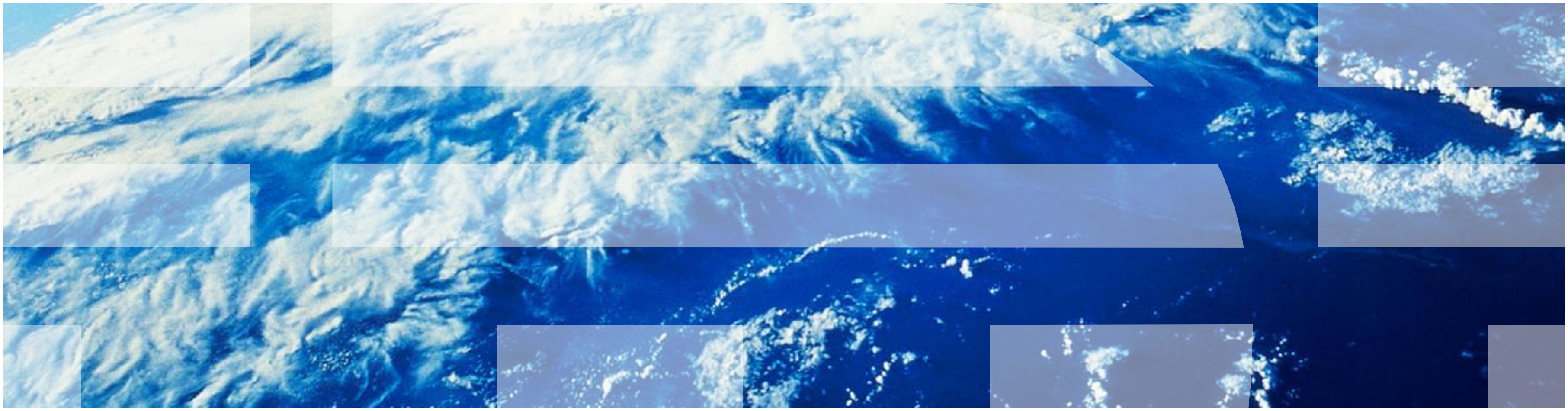
Cloud and Big Data

Dr. Sambit Sahu, VP AI Foundations, Capital One



Course Objective

- **Graduate/Adv Undergrad 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: build applications using cloud, learn building blocks/services required to design large scale applications, computing in a cluster

- **We learn cloud technologies by using real clouds - Amazon AWS, Google Cloud, Hadoop & Spark platform.**

- **Required background**
  - Programming experience with one of the following Java/Python
  - Concepts of web services and applications
  - Optional: Operating Systems fundamentals, networking concepts

Three Main Components

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

- **Cloud building blocks and services**
  - Compute Cloud: Virtualization concepts,
  - Storage Cloud, Cloud Database
  - Message Queues
  - Cloud devOps
  - ***Design pattern in extreme scale backend engineering***

- **Big Data Platform and Programming**
  - Hadoop eco-system, Map-Reduce, HDFS
  - Spark with RDD and dataframes
  - Intelligent systems and pipeline
  - ***Web scale data computing and pipeline***

Tentative Schedule

Date	Lecture	Reading Papers	Assignment/HW
09/05	Intro to Cloud		
09/12	Building Applications using AWS Cloud	GFS	
09/19	Large Scale Systems Design Patterns	Big Table	A1 Release
09/26	Containers, Kubernetes and Micro-services	DynamoDB	
10/03	Cloud DevOps	Kafka	
10/10	Messages Queues (Kafka) and Streams	Borg	A2 release
10/17	Quiz 1	Map Reduce	
10/24	Cluster Computing with Hadoop	Spark RDD	
10/31	Cluster Computing with Spark	Spanner	
11/07	Spark Dataframes and Data Pipelines		A3 release
11/14	Database: SQL, noSQL, Elastic Search		
11/21	Quiz 2		

Course Structure

- **Lecture Structure**
  - Each lecture will have a theme topic. First 1 hour 30 minutes lecture and demonstration by the instructor.
  - Last 20 minutes students to lead discussions from the reading paper lists.
- **Assignments/Exercises**
  - Reading list – consists of 10 landmark papers in the area of large scale systems (Google File System, Map Reduce, Spanner, Hadoop, Amazon DynamoDB, Kafka, Borg, Spark RDD etc.)
    - Submit paper summaries
    - Three Programming Assignments
    - Course Project: you can conduct this in a group of 3-4 students
- **Communication Channel:** Slack, Brightspace

Grading and requirements

- Class participations (paper critics/quizzes) -- 5% grade
  - Paper discussions
  - Paper summaries

- Mid-Term covering concepts, design and coding -- 25%
- Assignments – 35% grade
  - 3 programming assignments stressed on technologies and programming

- Course project -- 35% grade
  - Students may team upto size of four

- Submission process – Brightspace

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**
  - Form your team carefully – asking, interviewing your teammates. 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**
  - I will provide a set of skeleton ideas you could choose from.

What you need to do soon

- Get account on few popular clouds
  - Amazon AWS (EC2, S3)
  - Use AWS coupons if first time user
  - Build a static website using S3: https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html
  - Create a Virtual Machine using AWS EC2

- Course Project
  - Substantial portion of your grade depends on final course project
  - I will provide sample project ideas/topics
  - You need to have a team and a project topic submitted by end of 3rd week

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
  - AWS in Action
  - Learning Spark Programming
  - Kubernetes in Action

What is a WebApp?

Webapp: A web application (web app) is **an application program that is stored on a remote server and delivered over the internet through a browser interface**. Web services are web apps by definition and many, although not all, websites contain web apps

Examples:

- ????
- ????

Key Components:

- backend server stack
  - web server
  - application server
  - database
- frontend/client
- APIs
- network connectivity

https://blog.hubspot.com/website/what-is-web-app

Web Application Primer

Different Components of a WebApp

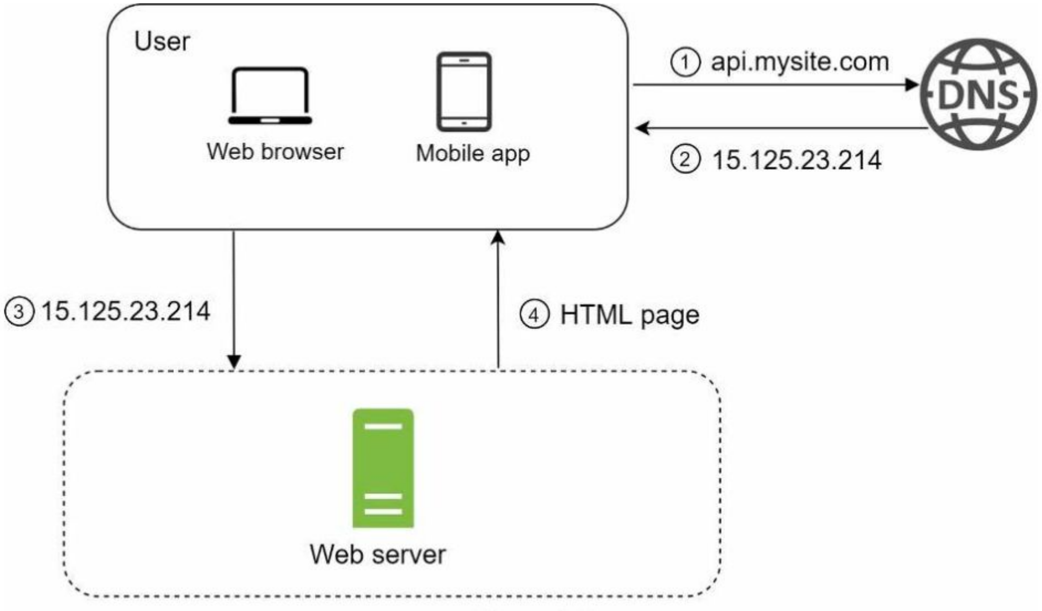
Client and Server Architecture

Client (frontend)

- webapps, mobile apps
- React (FB), Angular 4 etc

Server (backend)

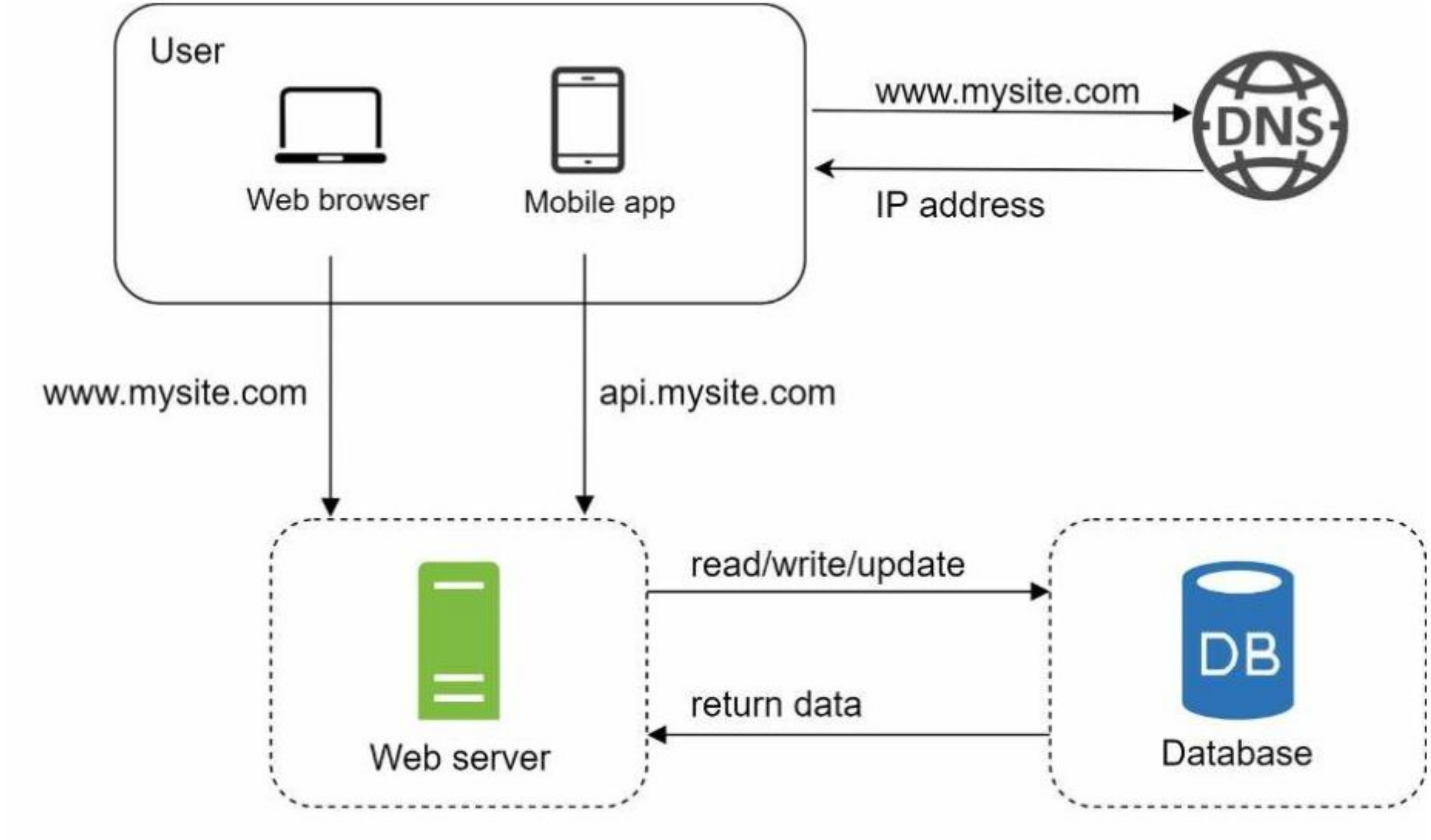
- three tier backend platform
- web server: node.js etc



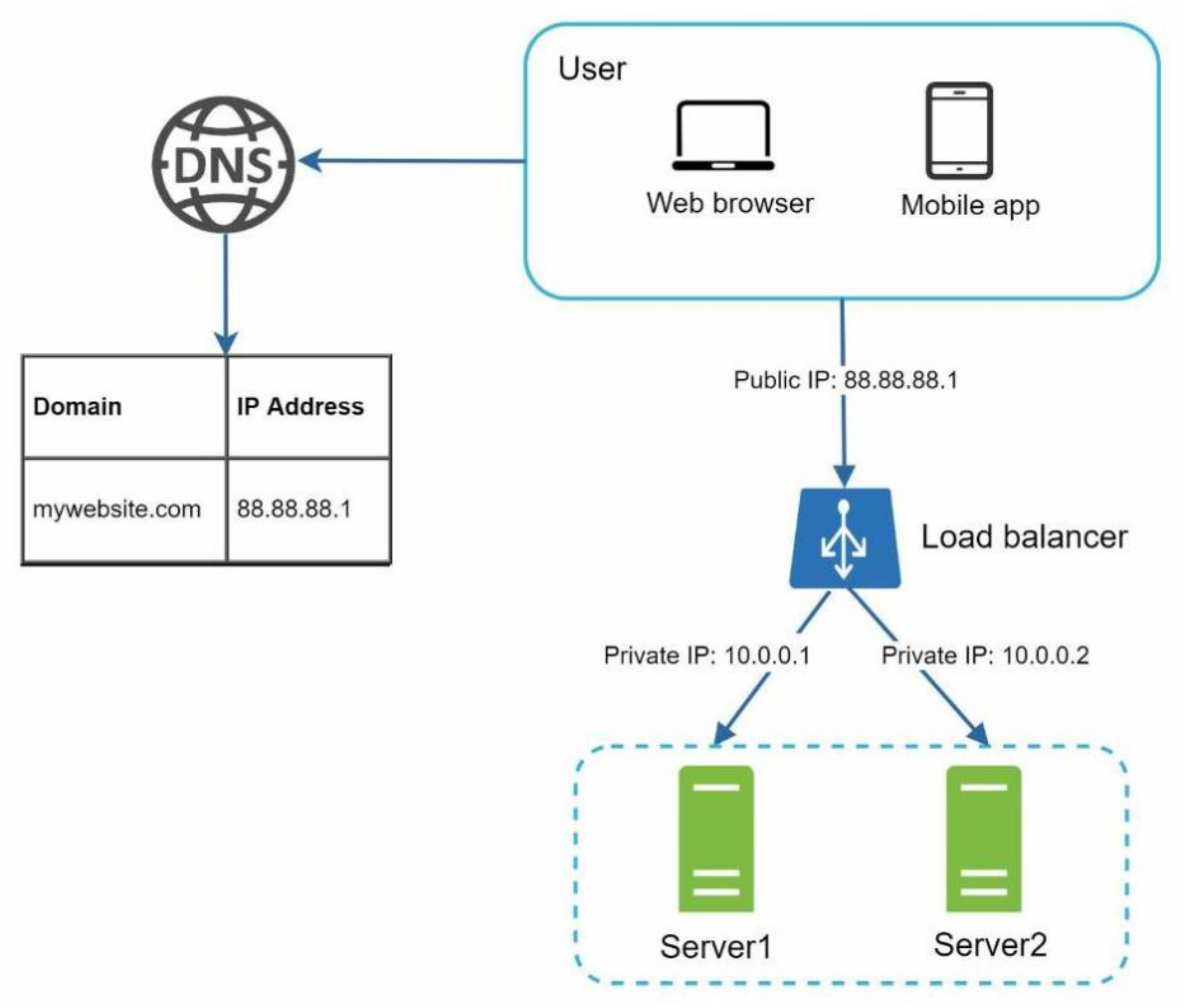
Web APIs

- frontend requests a services running in the backend via a HTML/web request
- REST based APIs

A Stateful WebApp



Scaling the backend of Webapp



Domain	IP Address
mywebsite.com	88.88.88.1

Scaling the Database

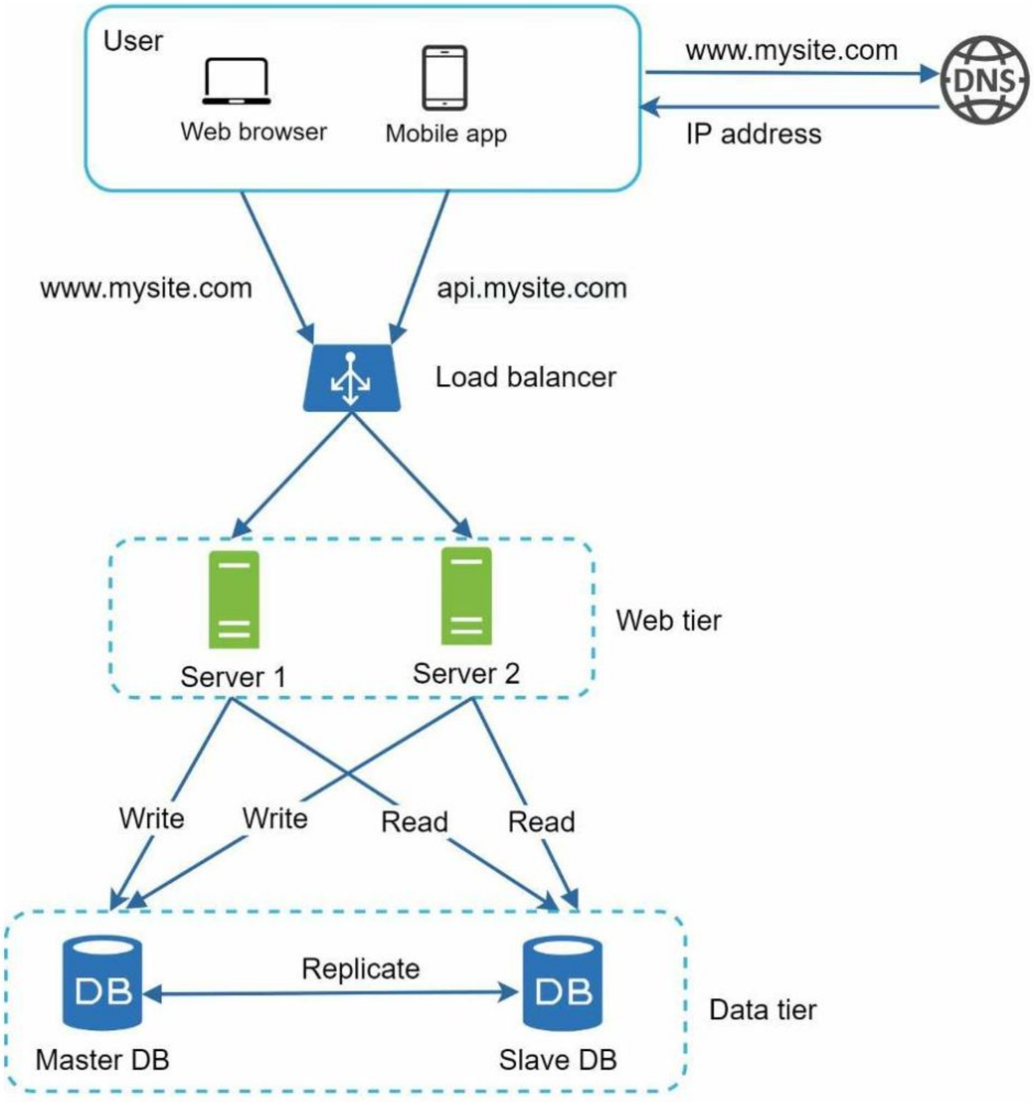
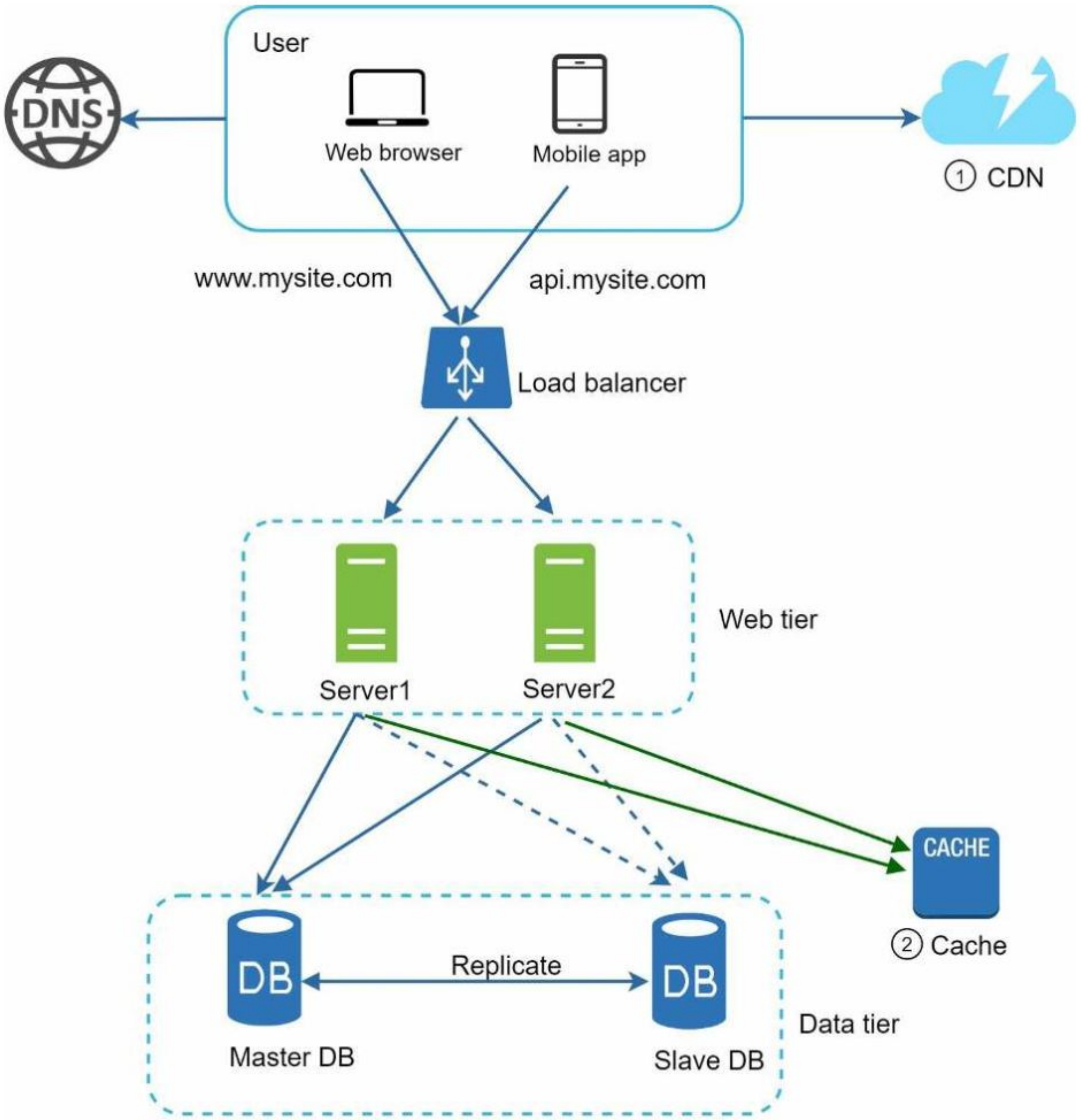


Figure 1-6

Scaling the Performance



What is Cloud?

- Allows users to request computing/storage resources and services 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!
- Example Clouds:
  - AWS: aws.amazon.com
  - Google Cloud: https://console.cloud.google.com/
  - Azure: https://azure.microsoft.com/en-us/products/

Why Cloud?

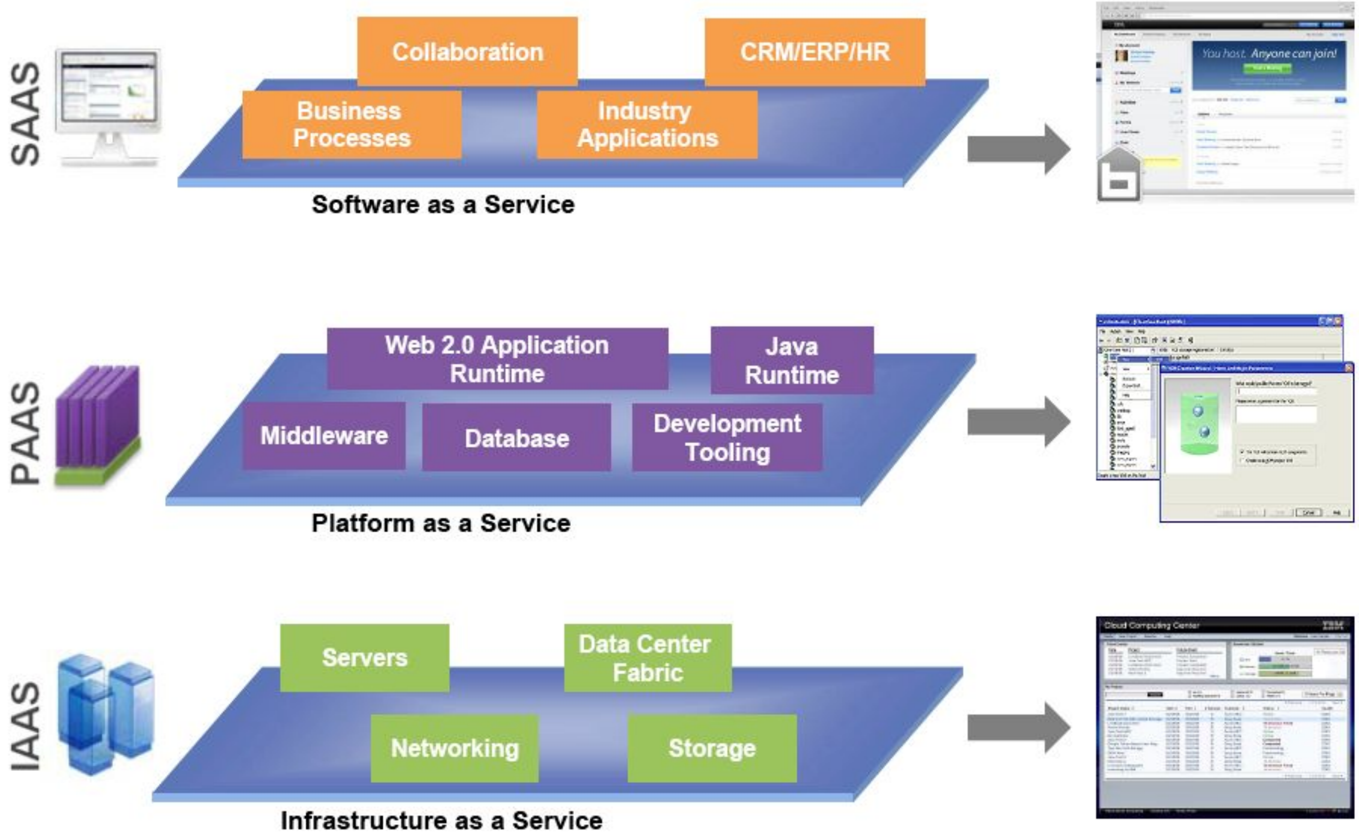
- You can get as many as 1000 machines for an hour for a few dollars to run a complex application!
- You don’t need to manage, maintain or fix any machines!
- You can use as little as 1 machine or as many as 10000 machines depending on what your current needs are!
- Two key focus: on-demand and elastic!

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).

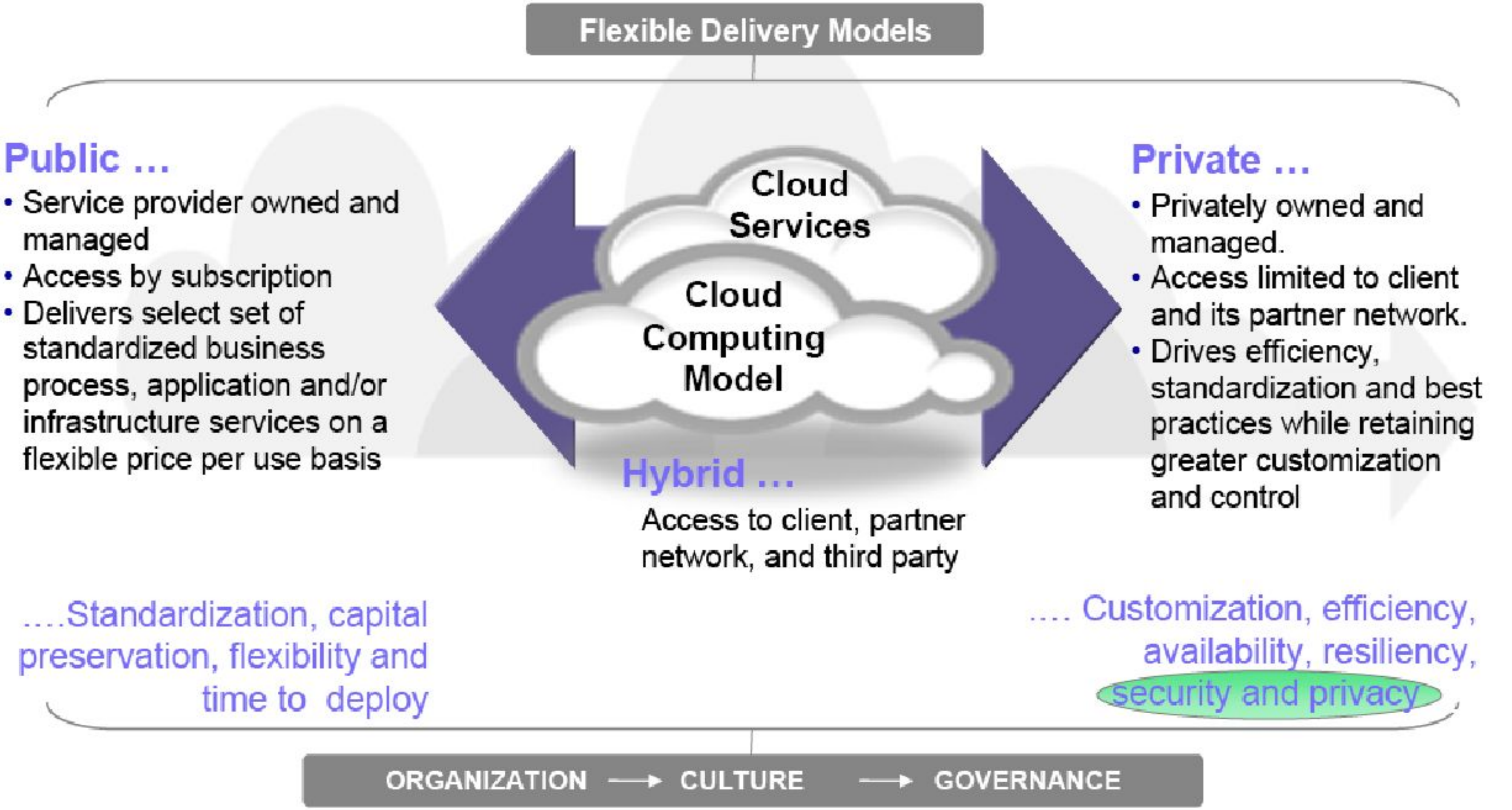
Different Cloud Offerings: A Layered Perspective

The Layers of IT-as-a-Service



- 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

Hybrid …

Access to client, partner network, and third party

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

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

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

ORGANIZATION → CULTURE → GOVERNANCE

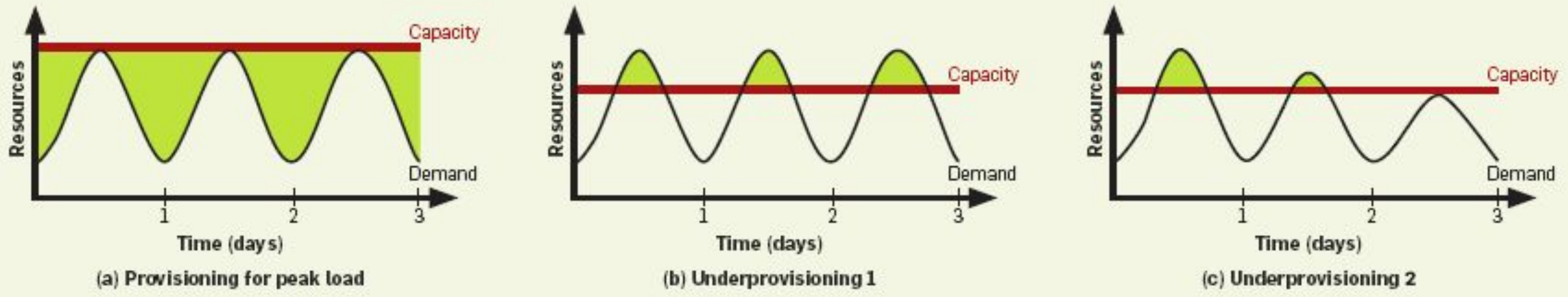
…service sourcing and service value

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.



(a) Provisioning for peak load

(b) Underprovisioning 1

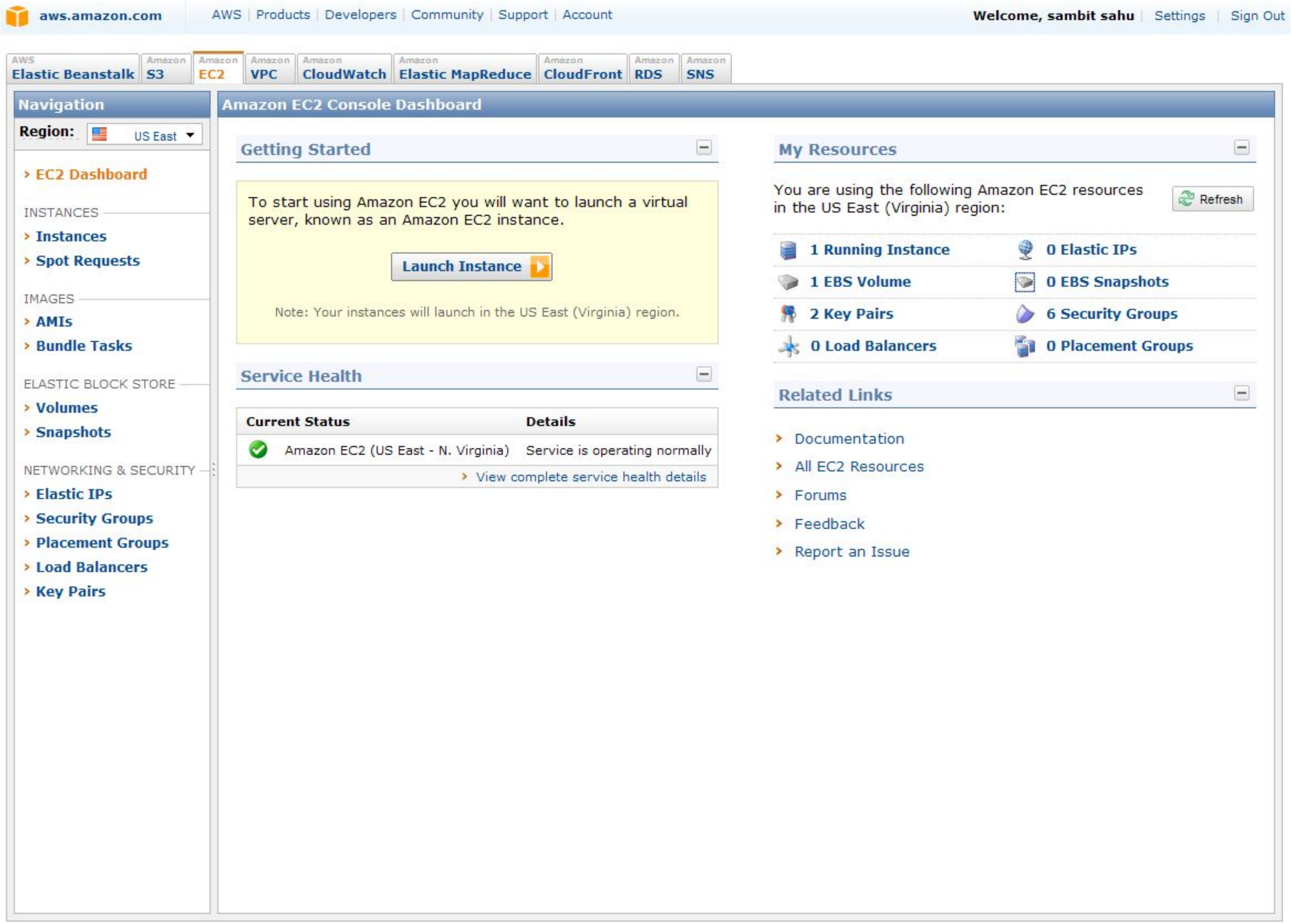
(c) Underprovisioning 2

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

Let’s use a IaaS Cloud (Amazon EC2)

- http://aws.amazon.com/console/
- Amazon EC2 console based provisioning demo
- Build a static website: https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html

Amazon AWS console (EC2 view)



To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Note: Your instances will launch in the US East (Virginia) region.

Current Status	Details
✓ Amazon EC2 (US East - N. Virginia)	Service is operating normally

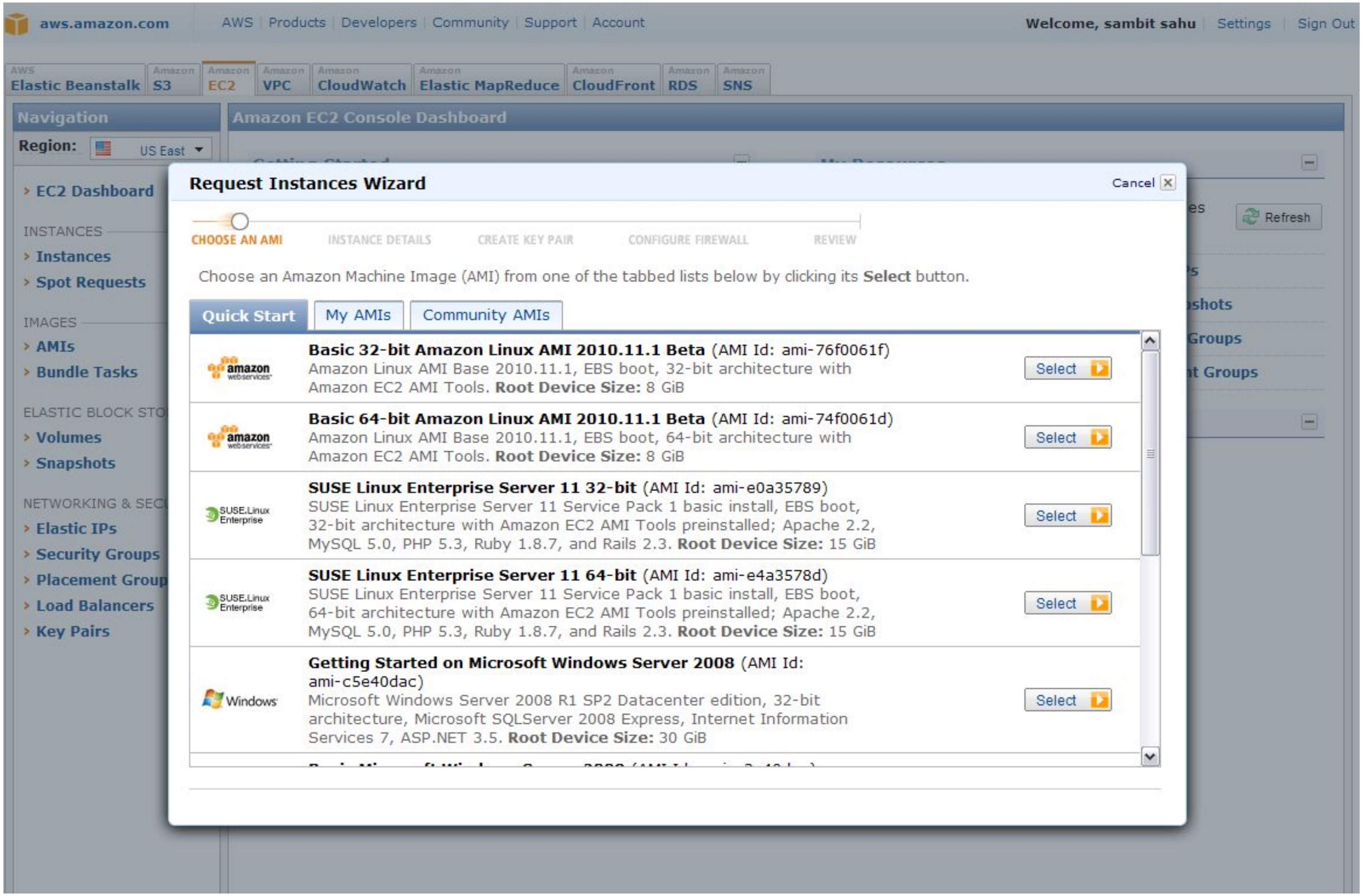
› View complete service health details

You are using the following Amazon EC2 resources in the US East (Virginia) region:

1 Running Instance	0 Elastic IPs
1 EBS Volume	0 EBS Snapshots
2 Key Pairs	6 Security Groups
0 Load Balancers	0 Placement Groups

- User logs in with AWS credentials

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



Request Instances Wizard

Choose an Amazon Machine Image (AMI) from one of the tabbed lists below by clicking its **Select** button.

Basic 32-bit Amazon Linux AMI 2010.11.1 Beta (AMI Id: ami-76f0061f)
Amazon Linux AMI Base 2010.11.1, EBS boot, 32-bit architecture with Amazon EC2 AMI Tools. **Root Device Size:** 8 GiB

Basic 64-bit Amazon Linux AMI 2010.11.1 Beta (AMI Id: ami-74f0061d)
Amazon Linux AMI Base 2010.11.1, EBS boot, 64-bit architecture with Amazon EC2 AMI Tools. **Root Device Size:** 8 GiB

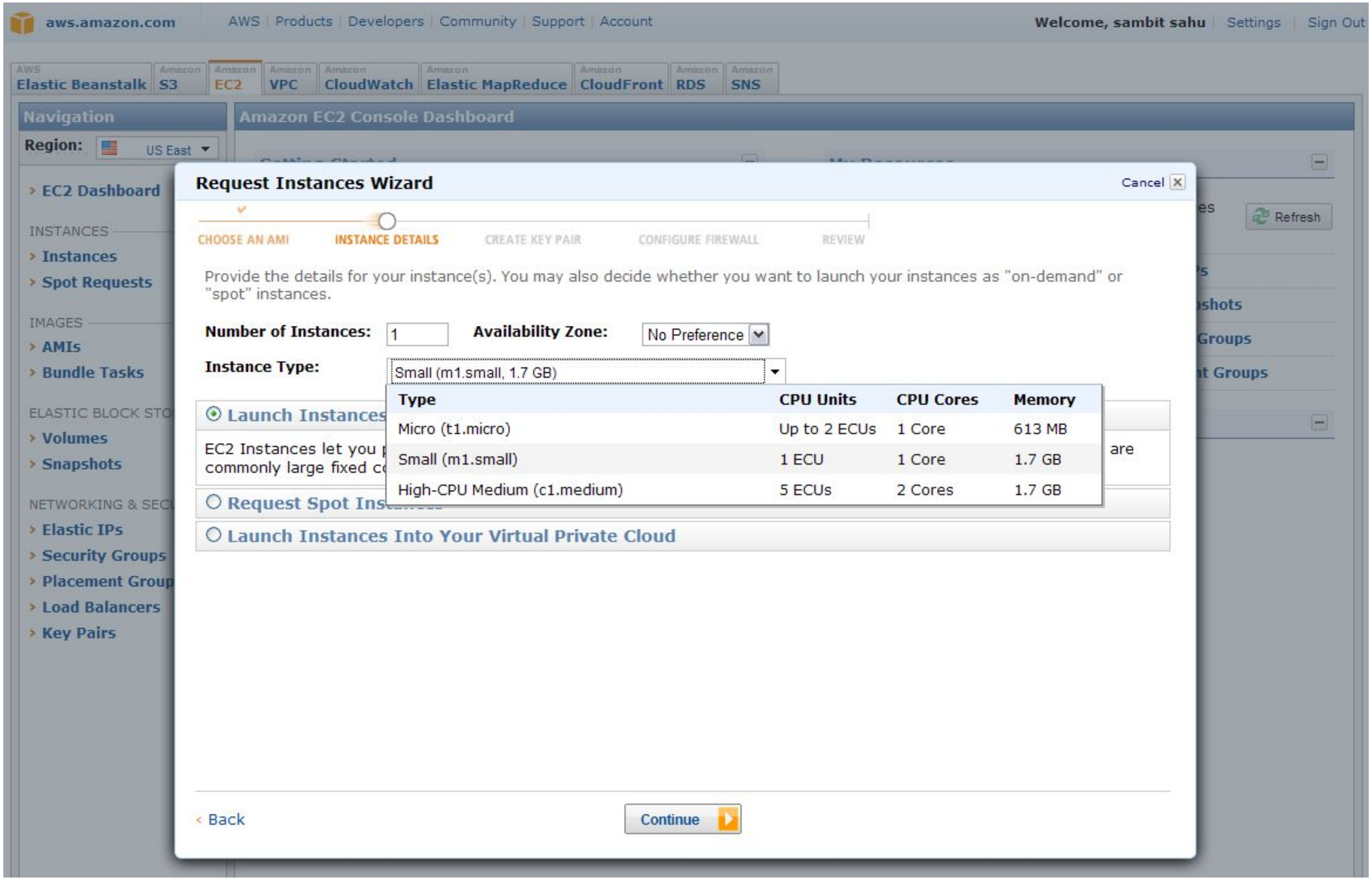
SUSE Linux Enterprise Server 11 32-bit (AMI Id: ami-e0a35789)
SUSE Linux Enterprise Server 11 Service Pack 1 basic install, EBS boot, 32-bit architecture with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, Ruby 1.8.7, and Rails 2.3. **Root Device Size:** 15 GiB

SUSE Linux Enterprise Server 11 64-bit (AMI Id: ami-e4a3578d)
SUSE Linux Enterprise Server 11 Service Pack 1 basic install, EBS boot, 64-bit architecture with Amazon EC2 AMI Tools preinstalled; Apache 2.2, MySQL 5.0, PHP 5.3, Ruby 1.8.7, and Rails 2.3. **Root Device Size:** 15 GiB

Getting Started on Microsoft Windows Server 2008 (AMI Id: ami-c5e40dac)
Microsoft Windows Server 2008 R1 SP2 Datacenter edition, 32-bit architecture, Microsoft SQLServer 2008 Express, Internet Information Services 7, ASP.NET 3.5. **Root Device Size:** 30 GiB

- 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)



Request Instances Wizard

CHOOSE AN AMI — INSTANCE DETAILS — CREATE KEY PAIR — CONFIGURE FIREWALL — REVIEW

Provide the details for your instance(s). You may also decide whether you want to launch your instances as "on-demand" or "spot" instances.

Number of Instances: 1 **Availability Zone:** No Preference

Instance Type: Small (m1.small, 1.7 GB)

Type	CPU Units	CPU Cores	Memory
Micro (t1.micro)	Up to 2 ECUs	1 Core	613 MB
Small (m1.small)	1 ECU	1 Core	1.7 GB
High-CPU Medium (c1.medium)	5 ECUs	2 Cores	1.7 GB

● Launch Instances

EC2 Instances let you p... are commonly large fixed c...

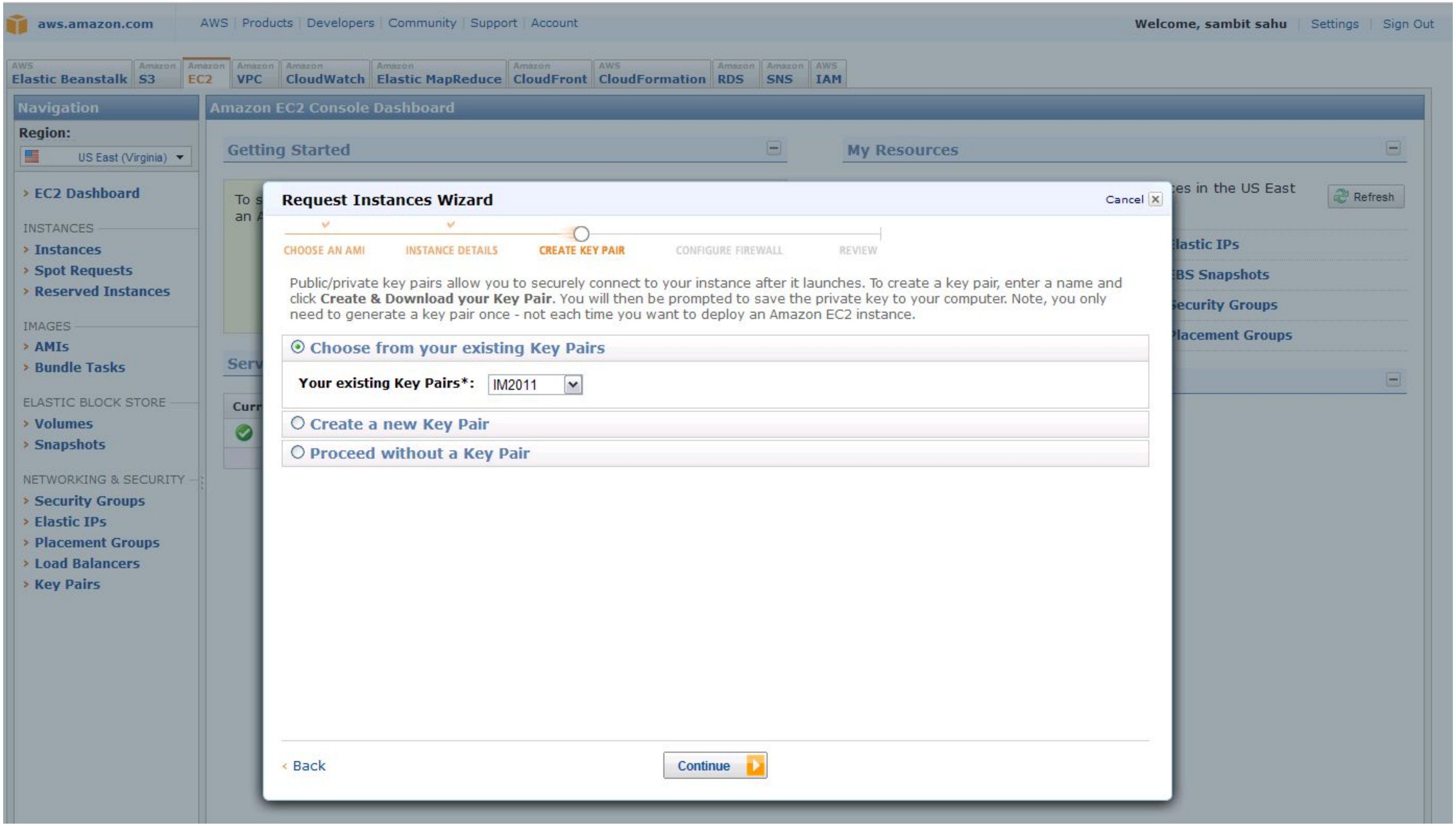
○ Request Spot Ins...

○ Launch Instances Into Your Virtual Private Cloud

< Back Continue

- Instance request wizard guides through resource choices

User specifies security/access configurations



Request Instances Wizard

Public/private key pairs allow you to securely connect to your instance after it launches. To create a key pair, enter a name and click **Create & Download your Key Pair**. You will then be prompted to save the private key to your computer. Note, you only need to generate a key pair once - not each time you want to deploy an Amazon EC2 instance.

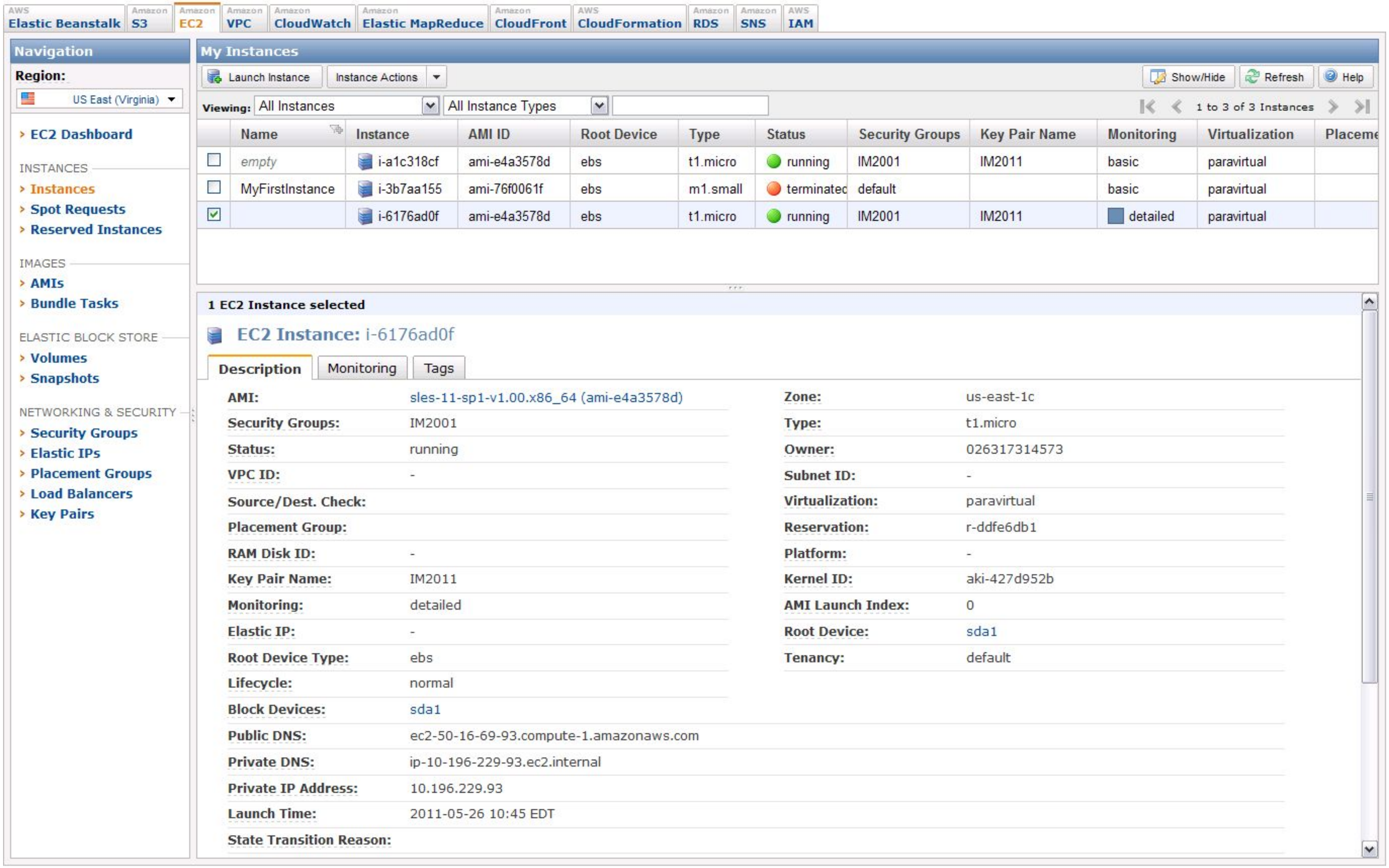
● Choose from your existing Key Pairs

Your existing Key Pairs*: IM2011

○ Create a new Key Pair

○ Proceed without a Key Pair

AWS provisions an instance and returns user credentials



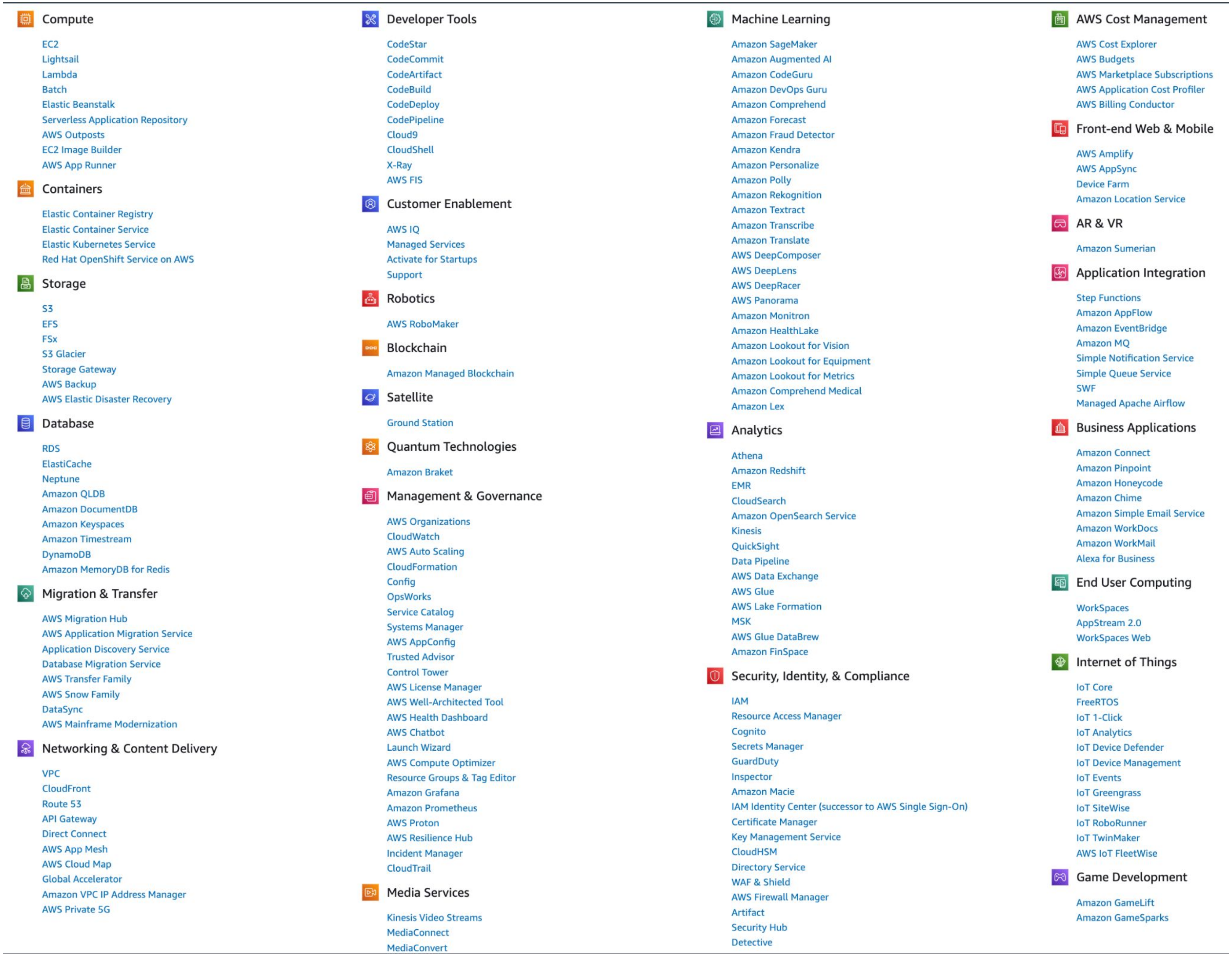
	Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name	Monitoring	Virtualization	Placeme
□	*empty*	i-a1c318cf	ami-e4a3578d	ebs	t1.micro	● running	IM2001	IM2011	basic	paravirtual	
□	MyFirstInstance	i-3b7aa155	ami-76f0061f	ebs	m1.small	● terminatec	default		basic	paravirtual	
☑		i-6176ad0f	ami-e4a3578d	ebs	t1.micro	● running	IM2001	IM2011	detailed	paravirtual	

1 EC2 Instance selected

EC2 Instance: i-6176ad0f

AMI:	sles-11-sp1-v1.00.x86_64 (ami-e4a3578d)	Zone:	us-east-1c
Security Groups:	IM2001	Type:	t1.micro
Status:	running	Owner:	026317314573
VPC ID:	-	Subnet ID:	-
Source/Dest. Check:		Virtualization:	paravirtual
Placement Group:		Reservation:	r-ddfe6db1
RAM Disk ID:	-	Platform:	-
Key Pair Name:	IM2011	Kernel ID:	aki-427d952b
Monitoring:	detailed	AMI Launch Index:	0
Elastic IP:	-	Root Device:	sda1
Root Device Type:	ebs	Tenancy:	default
Lifecycle:	normal		
Block Devices:	sda1		
Public DNS:	ec2-50-16-69-93.compute-1.amazonaws.com		
Private DNS:	ip-10-196-229-93.ec2.internal		
Private IP Address:	10.196.229.93		
Launch Time:	2011-05-26 10:45 EDT		
State Transition Reason:			

AWS Services Catalog



Compute

- EC2
- Lightsail
- Lambda
- Batch
- Elastic Beanstalk
- Serverless Application Repository
- AWS Outposts
- EC2 Image Builder
- AWS App Runner

Containers

- Elastic Container Registry
- Elastic Container Service
- Elastic Kubernetes Service
- Red Hat OpenShift Service on AWS

Storage

- S3
- EFS
- FSx
- S3 Glacier
- Storage Gateway
- AWS Backup
- AWS Elastic Disaster Recovery

Database

- RDS
- ElastiCache
- Neptune
- Amazon QLDB
- Amazon DocumentDB
- Amazon Keyspaces
- Amazon Timestream
- DynamoDB
- Amazon MemoryDB for Redis

Migration & Transfer

- AWS Migration Hub
- AWS Application Migration Service
- Application Discovery Service
- Database Migration Service
- AWS Transfer Family
- AWS Snow Family
- DataSync
- AWS Mainframe Modernization

Networking & Content Delivery

- VPC
- CloudFront
- Route 53
- API Gateway
- Direct Connect
- AWS App Mesh
- AWS Cloud Map
- Global Accelerator
- Amazon VPC IP Address Manager
- AWS Private 5G

Developer Tools

- CodeStar
- CodeCommit
- CodeArtifact
- CodeBuild
- CodeDeploy
- CodePipeline
- Cloud9
- CloudShell
- X-Ray
- AWS FIS

Customer Enablement

- AWS IQ
- Managed Services
- Activate for Startups
- Support

Robotics

- AWS RoboMaker

Blockchain

- Amazon Managed Blockchain

Satellite

- Ground Station

Quantum Technologies

- Amazon Braket

Management & Governance

- AWS Organizations
- CloudWatch
- AWS Auto Scaling
- CloudFormation
- Config
- OpsWorks
- Service Catalog
- Systems Manager
- AWS AppConfig
- Trusted Advisor
- Control Tower
- AWS License Manager
- AWS Well-Architected Tool
- AWS Health Dashboard
- AWS Chatbot
- Launch Wizard
- AWS Compute Optimizer
- Resource Groups & Tag Editor
- Amazon Grafana
- Amazon Prometheus
- AWS Proton
- AWS Resilience Hub
- Incident Manager
- CloudTrail

Media Services

- Kinesis Video Streams
- MediaConnect
- MediaConvert

Machine Learning

- Amazon SageMaker
- Amazon Augmented AI
- Amazon CodeGuru
- Amazon DevOps Guru
- Amazon Comprehend
- Amazon Forecast
- Amazon Fraud Detector
- Amazon Kendra
- Amazon Personalize
- Amazon Polly
- Amazon Rekognition
- Amazon Textract
- Amazon Transcribe
- Amazon Translate
- AWS DeepComposer
- AWS DeepLens
- AWS DeepRacer
- AWS Panorama
- Amazon Monitron
- Amazon HealthLake
- Amazon Lookout for Vision
- Amazon Lookout for Equipment
- Amazon Lookout for Metrics
- Amazon Comprehend Medical
- Amazon Lex

Analytics

- Athena
- Amazon Redshift
- EMR
- CloudSearch
- Amazon OpenSearch Service
- Kinesis
- QuickSight
- Data Pipeline
- AWS Data Exchange
- AWS Glue
- AWS Lake Formation
- MSK
- AWS Glue DataBrew
- Amazon FinSpace

Security, Identity, & Compliance

- IAM
- Resource Access Manager
- Cognito
- Secrets Manager
- GuardDuty
- Inspector
- Amazon Macie
- IAM Identity Center (successor to AWS Single Sign-On)
- Certificate Manager
- Key Management Service
- CloudHSM
- Directory Service
- WAF & Shield
- AWS Firewall Manager
- Artifact
- Security Hub
- Detective

AWS Cost Management

- AWS Cost Explorer
- AWS Budgets
- AWS Marketplace Subscriptions
- AWS Application Cost Profiler
- AWS Billing Conductor

Front-end Web & Mobile

- AWS Amplify
- AWS AppSync
- Device Farm
- Amazon Location Service

AR & VR

- Amazon Sumerian

Application Integration

- Step Functions
- Amazon AppFlow
- Amazon EventBridge
- Amazon MQ
- Simple Notification Service
- Simple Queue Service
- SWF
- Managed Apache Airflow

Business Applications

- Amazon Connect
- Amazon Pinpoint
- Amazon Honeycode
- Amazon Chime
- Amazon Simple Email Service
- Amazon WorkDocs
- Amazon WorkMail
- Alexa for Business

End User Computing

- WorkSpaces
- AppStream 2.0
- WorkSpaces Web

Internet of Things

- IoT Core
- FreeRTOS
- IoT 1-Click
- IoT Analytics
- IoT Device Defender
- IoT Device Management
- IoT Events
- IoT Greengrass
- IoT SiteWise
- IoT RoboRunner
- IoT TwinMaker
- AWS IoT FleetWise

Game Development

- Amazon GameLift
- Amazon GameSparks

How to build the backend using resources from cloud

Let’s see how to leverage on-demand and elastic resources from cloud to build an extreme scale backend platform for a given application

- Let’s first build a webapp
- Next let’s create that webapp using resources from cloud
- Next we will progress towards building an extremely large scale application backend
-

Demos/Videos/Links

- Static website using AWS S3 service: https://docs.aws.amazon.com/AmazonS3/latest/userguide/HostingWebsiteOnS3Setup.html
-

Next Week

- Reading List
  - GFS: The Google File System
    - http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf

- Mini Homework
  - 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.

Lecture 2: IaaS Cloud and Cloud Programming

- API and CLI based access
  - http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
- AWS access using Java SDK and CLI
- Learn how to use EC2 and S3 as example services
- Breaking down the steps - how AWS provided on-demand resource
- Create a web server and deploy your web application using AWS
- How to use on-demand infrastructure for regular applications?

Amazon EC2 Programming

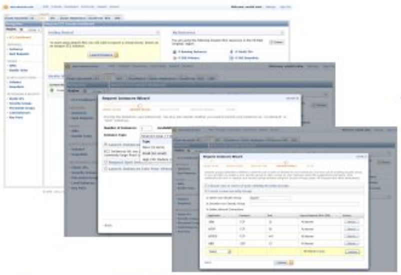
- Amazon EC2 SDK for java on Eclipse
  - http://aws.amazon.com/eclipse/
  - https://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-install.html
- AWS SDK with Python: https://aws.amazon.com/sdk-for-python/

- RESTful APIs for invoking EC2 APIs from Java

Deconstructing Amazon EC2 request machine API

- User goes to Amazon EC2 portal and specifies desired parameters for a machine
  - Resource: CPU, mem, disk
  - Stack: OS and possibly with additional software
- Amazon AWS Cloud manager (resource pool manager) provisions the user request
  - Finds appropriate physical resource
  - Dispatches the request to virtualization manager on the identified resource
  - Cloud Manager invokes EC2 API to provisions the request
- Virtualization manager on physical server
  - Copies the pre-built software stack (virtual appliance)
  - Provisions a guest VM and configures parameters (IP address, access rules,…) at run/boot time
- Cloud manager returns login credentials to user

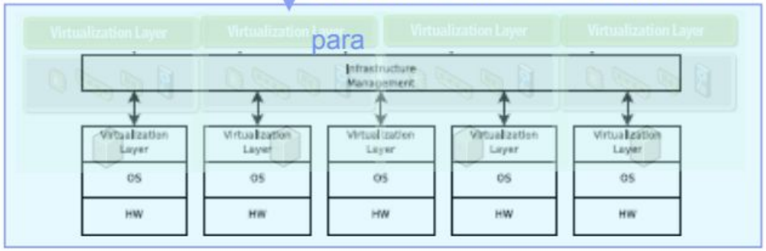
1. User requests a machine with a desired Software stack, access rules



2. Cloud manager processes request

3. Identifies physical server where to instantiate

4. Virtualization mgr on the server launches a VM, copies virtual appliance and boots the VM with appropriate run-time configuration



Physical Resource Pool

5. Login credentials for user

6. User is provided instance details