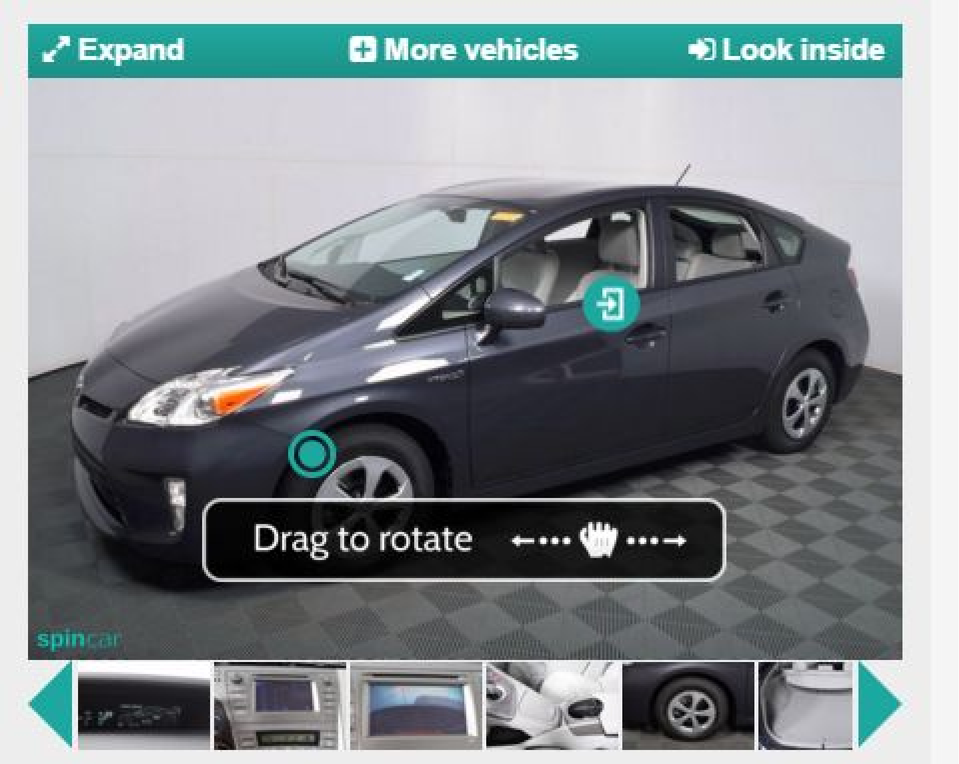
A Comparison of Leased Servers, Cloud Hosting and IaaS

With a focus on Amazon Web Services

Steve Saporta
CTO, SpinCar

Mar 18-19, 2016

SpinCar



You’ve just written a great web application...

Now what?

How do you share your web app...

...with the world?

You need **hosting**.

What is a server?

A server takes a **request** and provides a **response**.

A server is both a running instance of some software that is capable of accepting requests from clients, and the computer that executes such software. -- https://en.wikipedia.org/wiki/Server_(computing)

If what I need is a server, why is it called hosting?

The term "host" is used in several contexts… A host is a computer with a Web server that serves the pages for one or more Web sites... A host can also be the company that provides that service, which is known as hosting.

-- http://searchnetworking.techtarget.com/definition/host

Static or dynamic?

There are no official, consistent definitions of *static* and *dynamic*

- *A static website consists of a series of HTML files, each one representing a physical page of a website. A dynamic website uses server technologies (such as PHP) to dynamically build a webpage right when a user visits the page.* -- https://rocketmedia.com/blog/static-vs-dynamic-websites
- *Websites that only use HTML and CSS are called static websites, and websites with scripting are called dynamic websites.* -- http://www.codeconquest.com/website/static-vs-dynamic-websites/
- So, is JavaScript static or dynamic??

A better question: client-side or server-side?

Client-side

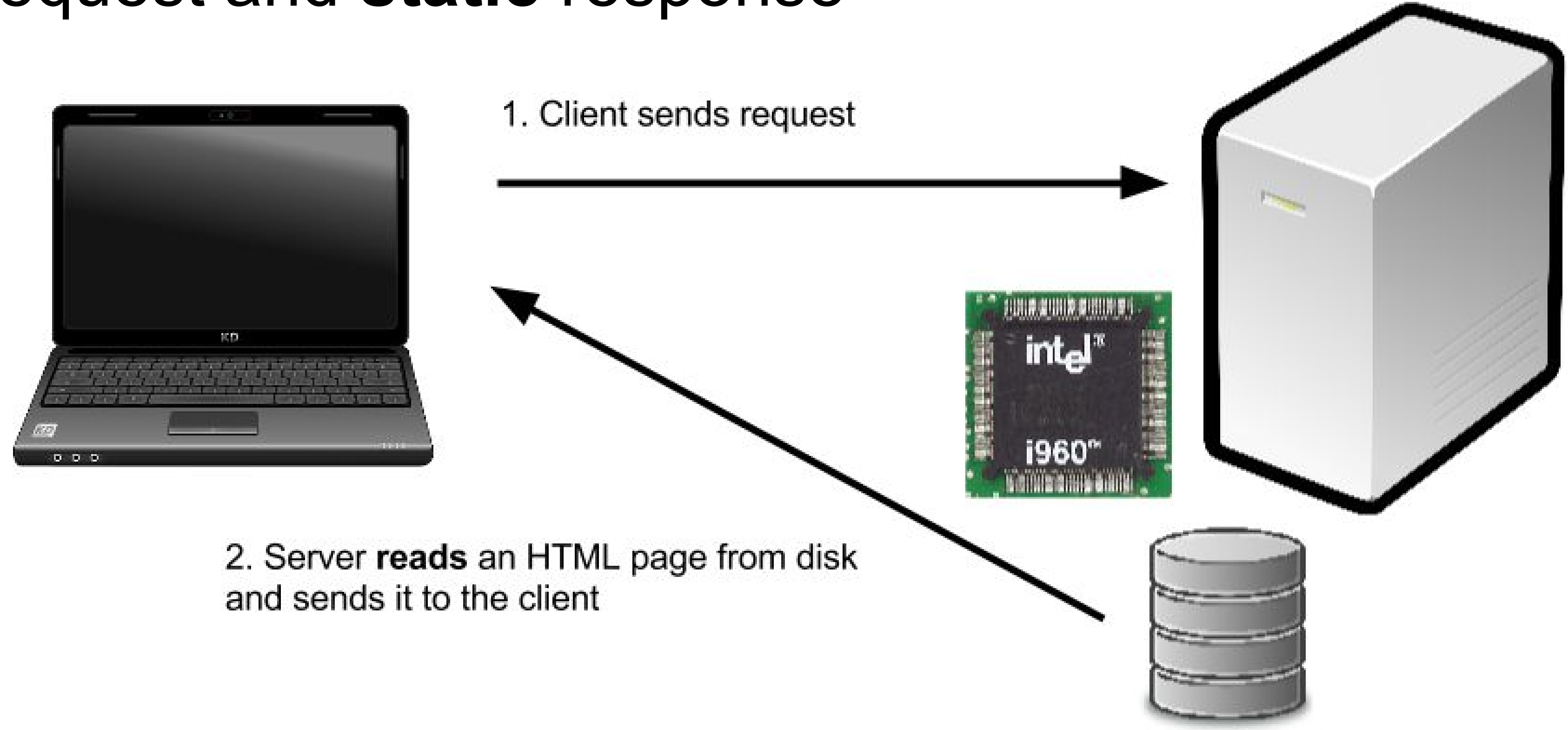
- Everything runs on the client, that is, the web browser
- HTML, CSS, JavaScript, images
- Audio, video, Flash, etc (unless you have special needs)

Server-side -- a web **application**, some part of which runs on a server

- Languages like C#, PHP, Perl, Python, Ruby, etc
- Login / authentication / authorization
- Database access

We’ll use “static” as shorthand for “client-side code only”

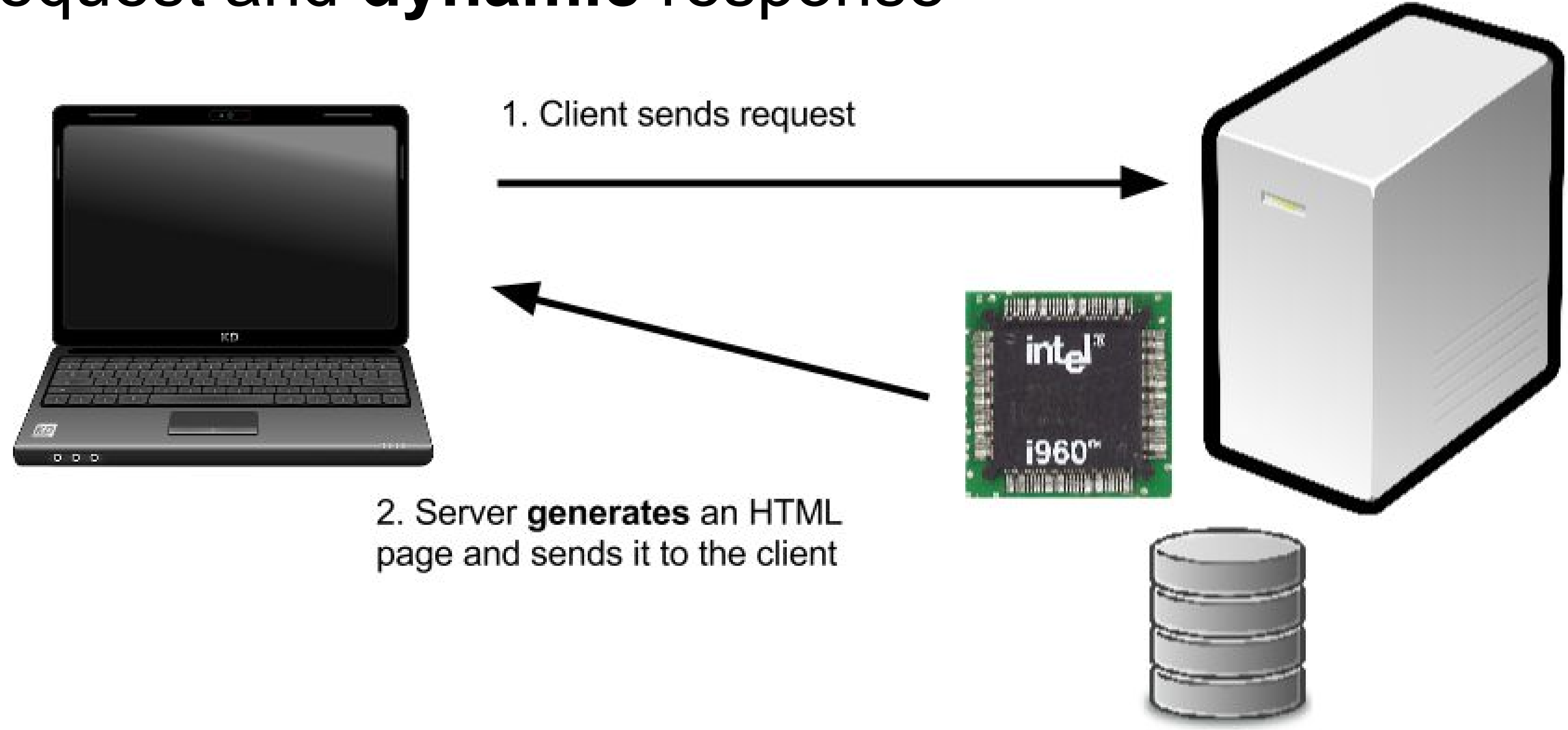
Request and **static** response



1. Client sends request

2. Server **reads** an HTML page from disk
and sends it to the client

Request and **dynamic** response



1. Client sends request

2. Server **generates** an HTML page and sends it to the client

It’s easy to find hosting for static websites

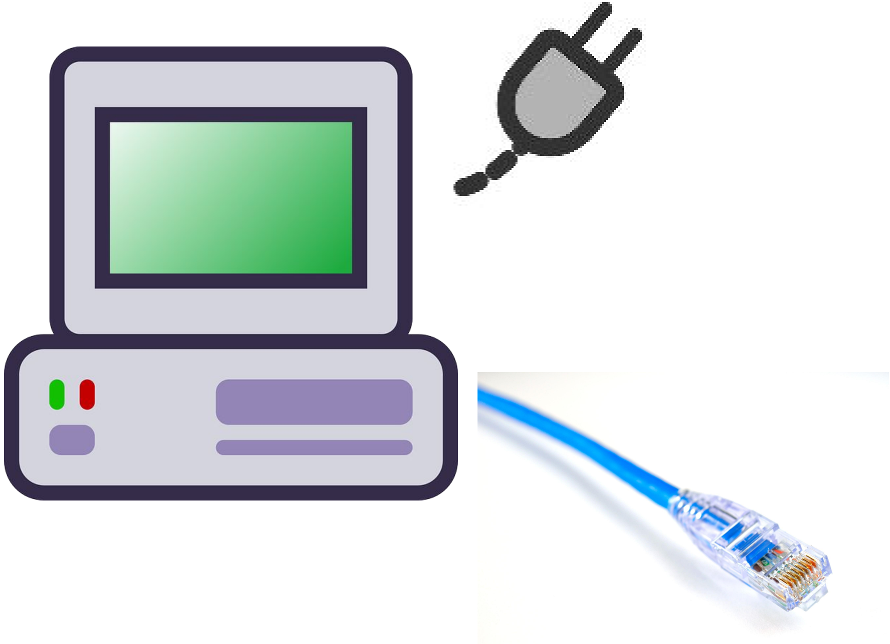
- Traditional web hosting providers: GoDaddy, eHost, etc
- For limited website capabilites: Google Sites, GitHub pages, etc.
- Amazon S3 bucket configured as a static website

Even static hosting becomes complicated if you expect a lot of traffic

We’ll focus on hosting for dynamic web applications

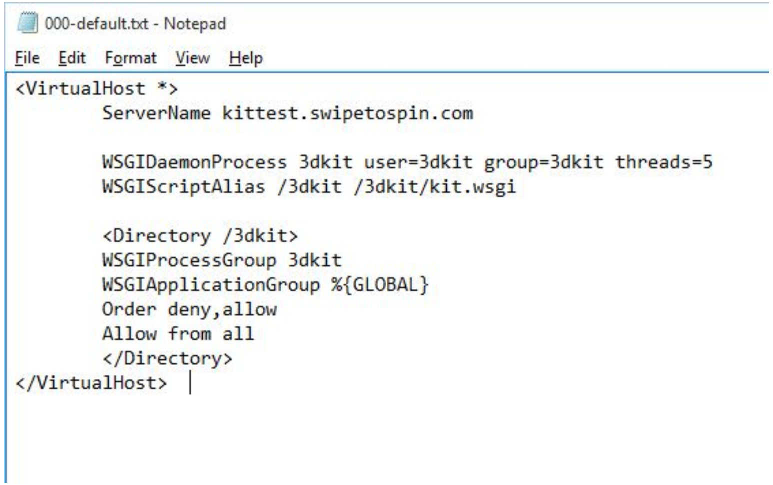
- Server-side code
- Maybe a database
- Real-world concerns
    - Availability
    - Scalability
    - Security

You could build your own server

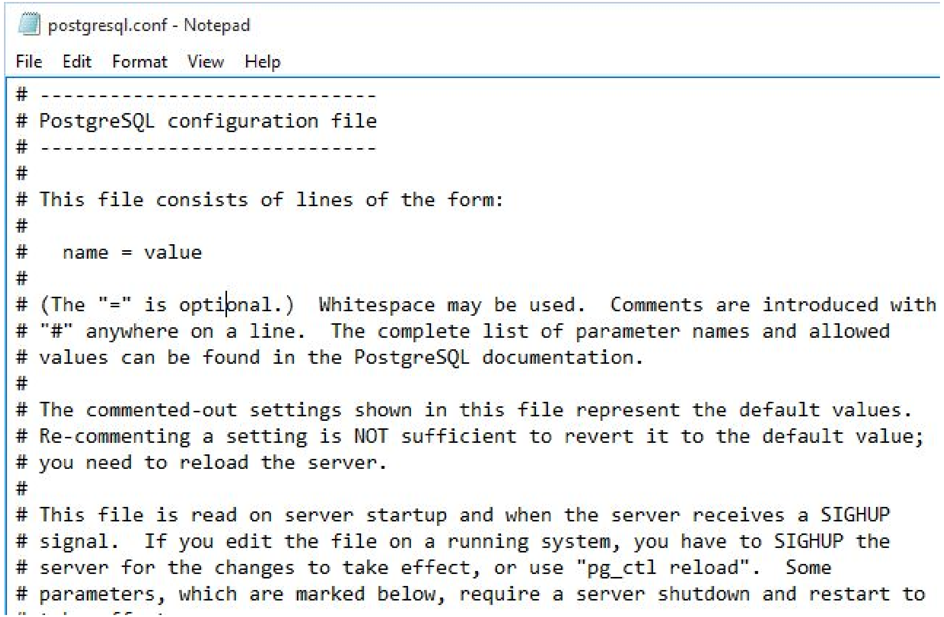


You’ll need to install a bunch of software

And configure the software

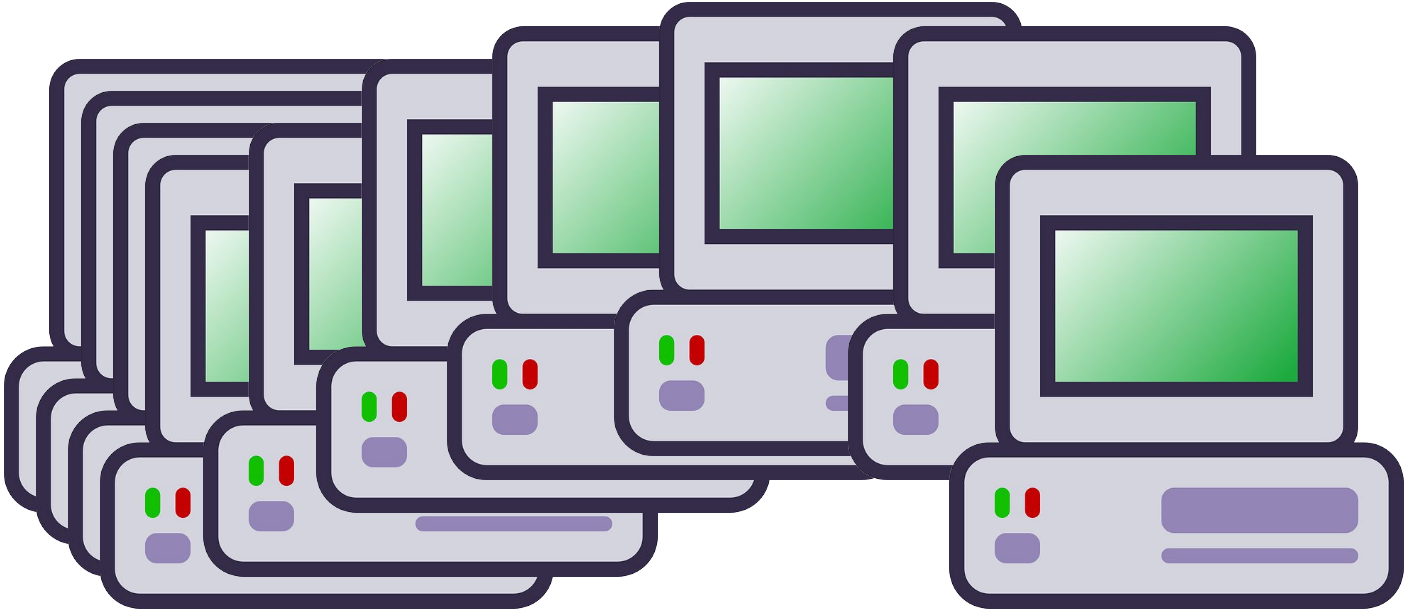


```
<VirtualHost *>
        ServerName kittest.swipetospin.com

        WSGIDaemonProcess 3dkit user=3dkit group=3dkit threads=5
        WSGIScriptAlias /3dkit /3dkit/kit.wsgi

        <Directory /3dkit>
        WSGIProcessGroup 3dkit
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
        </Directory>
</VirtualHost>
```



```
# -----------------------------
# PostgreSQL configuration file
# -----------------------------
#
# This file consists of lines of the form:
#
#   name = value
#
# (The "=" is optional.)  Whitespace may be used.  Comments are introduced with
# "#" anywhere on a line.  The complete list of parameter names and allowed
# values can be found in the PostgreSQL documentation.
#
# The commented-out settings shown in this file represent the default values.
# Re-commenting a setting is NOT sufficient to revert it to the default value;
# you need to reload the server.
#
# This file is read on server startup and when the server receives a SIGHUP
# signal.  If you edit the file on a running system, you have to SIGHUP the
# server for the changes to take effect, or use "pg_ctl reload".  Some
# parameters, which are marked below, require a server shutdown and restart to
```

Maybe not just one server



What if...

… There’s a power failure?

What if...

… Your Internet connection fails

What if...

… The hard drive fails while you’re sleeping?

What if...

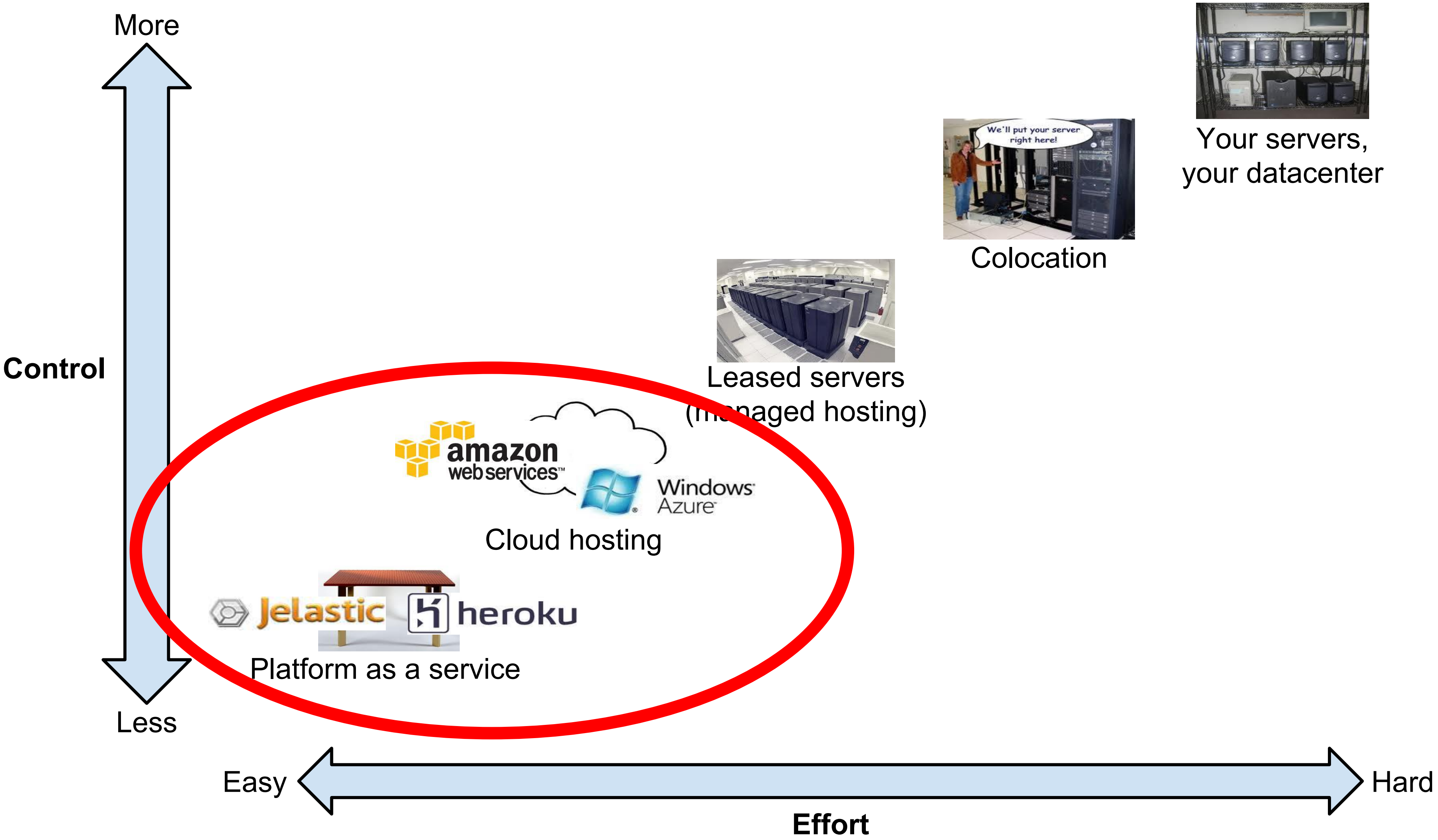
… Someone breaks in and steals your computer?

Leased servers

- Someone else buys the hardware
- Someone else provides electrical and network connections
- Someone else provides a secure datacenter
- You pay rent, usually by the month
- Web server (e.g. Apache) is probably included
- You might still have to provide your own application server and database
- You definitely still have to install and configure your own application

Leased servers can be dedicated or shared

Dedicated	**Shared**
All yours	Shared with other customers
Your own CPU, RAM and disk storage	A virtual machine with a share of the available CPU, RAM and disk storage
Higher cost	Lower cost
More control	Less control - you might not be able to configure the web server, install packages, etc to your liking



More

Control

Less

Your servers, your datacenter

Colocation

Leased servers (managed hosting)

Cloud hosting

Platform as a service

Easy

Hard

Effort

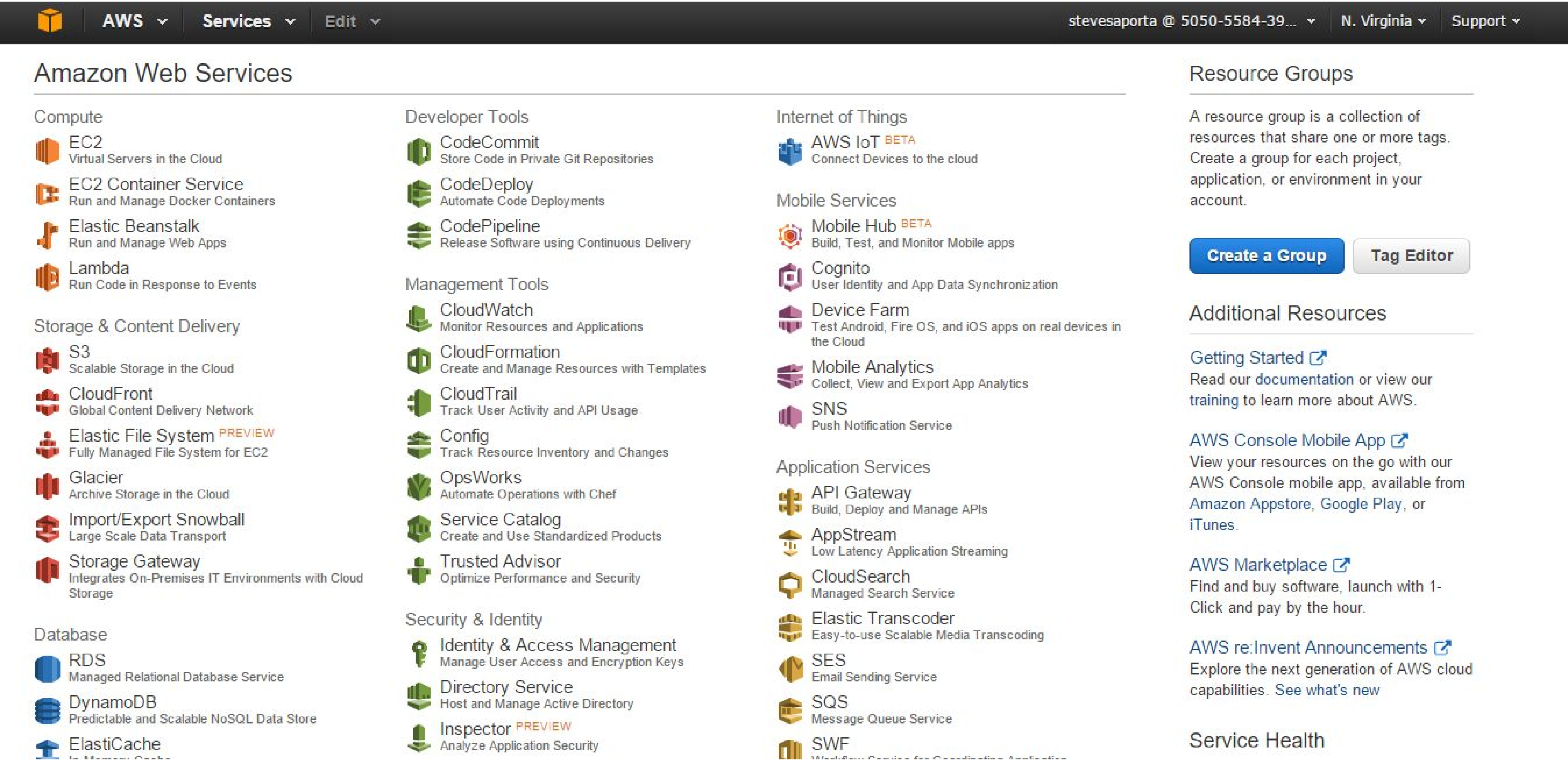
What is cloud hosting?

Cloud computing is a model for enabling ubiquitous, convenient, on-demand access to a shared pool of configurable computing resources… The network elements representing the provider-rendered services are invisible, as if obscured by a cloud. -- https://en.wikipedia.org/wiki/Cloud_computing

Imagine racks of servers, humming along in a data center. Together, these servers become a massive pool of resources. Divide this "pool" into multiple virtual servers, and you create a "cloud." -- http://www.rackspace.com/cloud/what_is_cloud_computing

Why cloud hosting?

- Rent by the hour, not by the month.
- Increase or decrease capacity on a moment’s notice
  - Manually or automatically
- Access a variety of services, not just web servers, designed and maintained by experts
  - Storage
  - Databases
  - Queues
  - Email
- Almost infinitely scalable



Amazon Web Services

Compute

EC2
Virtual Servers in the Cloud

EC2 Container Service
Run and Manage Docker Containers

Elastic Beanstalk
Run and Manage Web Apps

Lambda
Run Code in Response to Events

Storage & Content Delivery

S3
Scalable Storage in the Cloud

CloudFront
Global Content Delivery Network

Elastic File System PREVIEW
Fully Managed File System for EC2

Glacier
Archive Storage in the Cloud

Import/Export Snowball
Large Scale Data Transport

Storage Gateway
Integrates On-Premises IT Environments with Cloud Storage

Database

RDS
Managed Relational Database Service

DynamoDB
Predictable and Scalable NoSQL Data Store

ElastiCache
In-Memory Cache

Developer Tools

CodeCommit
Store Code in Private Git Repositories

CodeDeploy
Automate Code Deployments

CodePipeline
Release Software using Continuous Delivery

Management Tools

CloudWatch
Monitor Resources and Applications

CloudFormation
Create and Manage Resources with Templates

CloudTrail
Track User Activity and API Usage

Config
Track Resource Inventory and Changes

OpsWorks
Automate Operations with Chef

Service Catalog
Create and Use Standardized Products

Trusted Advisor
Optimize Performance and Security

Security & Identity

Identity & Access Management
Manage User Access and Encryption Keys

Directory Service
Host and Manage Active Directory

Inspector PREVIEW
Analyze Application Security

Internet of Things

AWS IoT BETA
Connect Devices to the cloud

Mobile Services

Mobile Hub BETA
Build, Test, and Monitor Mobile apps

Cognito
User Identity and App Data Synchronization

Device Farm
Test Android, Fire OS, and iOS apps on real devices in the Cloud

Mobile Analytics
Collect, View and Export App Analytics

SNS
Push Notification Service

Application Services

API Gateway
Build, Deploy and Manage APIs

AppStream
Low Latency Application Streaming

CloudSearch
Managed Search Service

Elastic Transcoder
Easy-to-use Scalable Media Transcoding

SES
Email Sending Service

SQS
Message Queue Service

SWF
Workflow Service for Coordinating Application

Resource Groups

A resource group is a collection of resources that share one or more tags. Create a group for each project, application, or environment in your account.

Additional Resources

Getting Started
Read our documentation or view our training to learn more about AWS.

AWS Console Mobile App
View your resources on the go with our AWS Console mobile app, available from Amazon Appstore, Google Play, or iTunes.

AWS Marketplace
Find and buy software, launch with 1-Click and pay by the hour.

AWS re:Invent Announcements
Explore the next generation of AWS cloud capabilities. See what's new

Service Health

Cloud hosting providers

- Amazon Web Services (AWS)
- Microsoft Azure
- Rackspace (also offers leased servers)
- Many others
- AWS is by far the leader
  - Top position in Gartner “quadrant” ratings
  - 10x the capacity of 14 other providers combined
  - “Coolness” factor

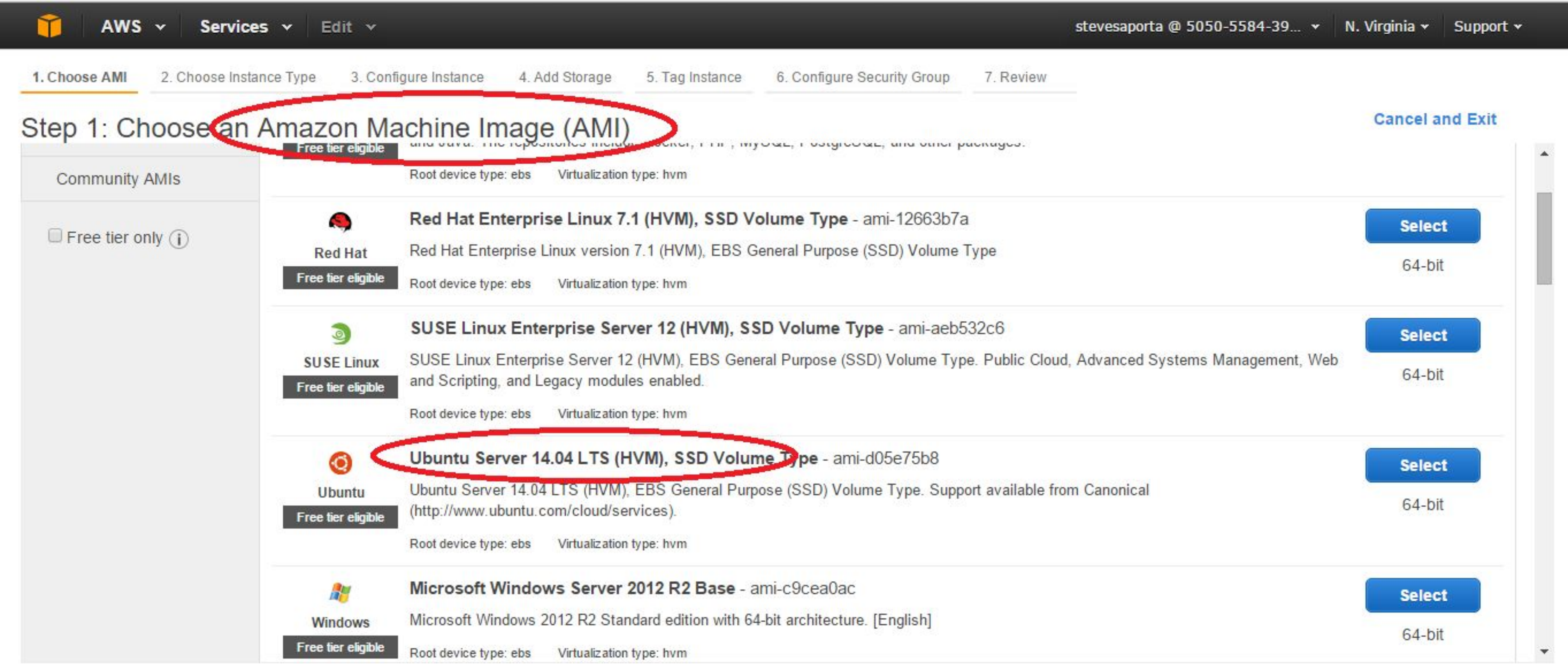
Why did Amazon get into the cloud hosting business?

Myth: Amazon had extra server and network capacity for their retail operation, and decided to rent it out by the hour.

Reality: Jeff Bezos realized cloud hosting was a profitable business opportunity and decided to launch Amazon Web Service.

AWS’s core service - EC2

- EC2 = “Elastic Compute Cloud”
- Virtual servers
- Rent a server by the hour
- Your choice of operating system, capacity and many other settings



Step 1: Choose an Amazon Machine Image (AMI)

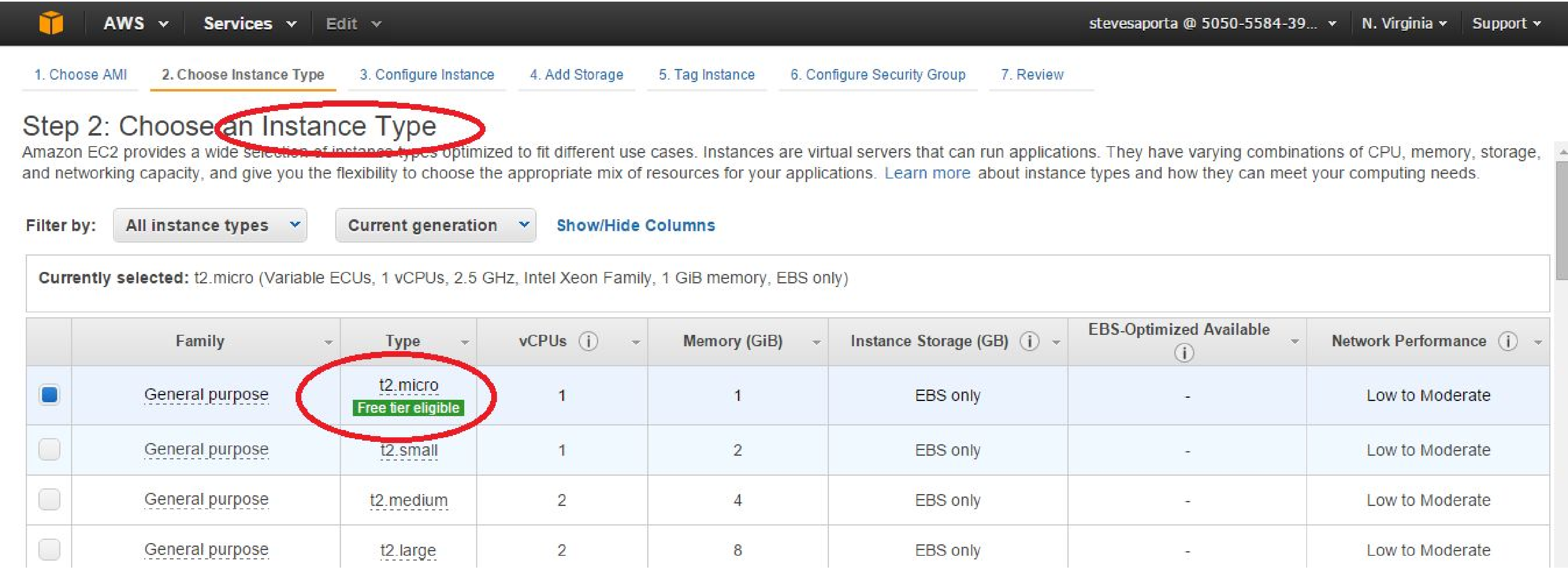
Free tier eligible — and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm

Red Hat Enterprise Linux 7.1 (HVM), SSD Volume Type - ami-12663b7a
Red Hat — Free tier eligible — 64-bit
Red Hat Enterprise Linux version 7.1 (HVM), EBS General Purpose (SSD) Volume Type
Root device type: ebs Virtualization type: hvm

SUSE Linux Enterprise Server 12 (HVM), SSD Volume Type - ami-aeb532c6
SUSE Linux — Free tier eligible — 64-bit
SUSE Linux Enterprise Server 12 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.
Root device type: ebs Virtualization type: hvm

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d05e75b8
Ubuntu — Free tier eligible — 64-bit
Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services).
Root device type: ebs Virtualization type: hvm

Microsoft Windows Server 2012 R2 Base - ami-c9cea0ac
Windows — Free tier eligible — 64-bit
Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]
Root device type: ebs Virtualization type: hvm



Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. Learn more about instance types and how they can meet your computing needs.

Filter by: All instance types | Current generation | Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
□	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
□	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
□	General purpose	t2.large	2	8	EBS only	-	Low to Moderate

Some other AWS services

S3

- Simple Storage Service
- Like a hard disk in the cloud

RDS

- Relational Database Service
- Your choice of MySQL, PostgreSQL, Oracle, etc
- Tuning, patching and backups are taking care of for you

Some other AWS services (cont’d)

SES

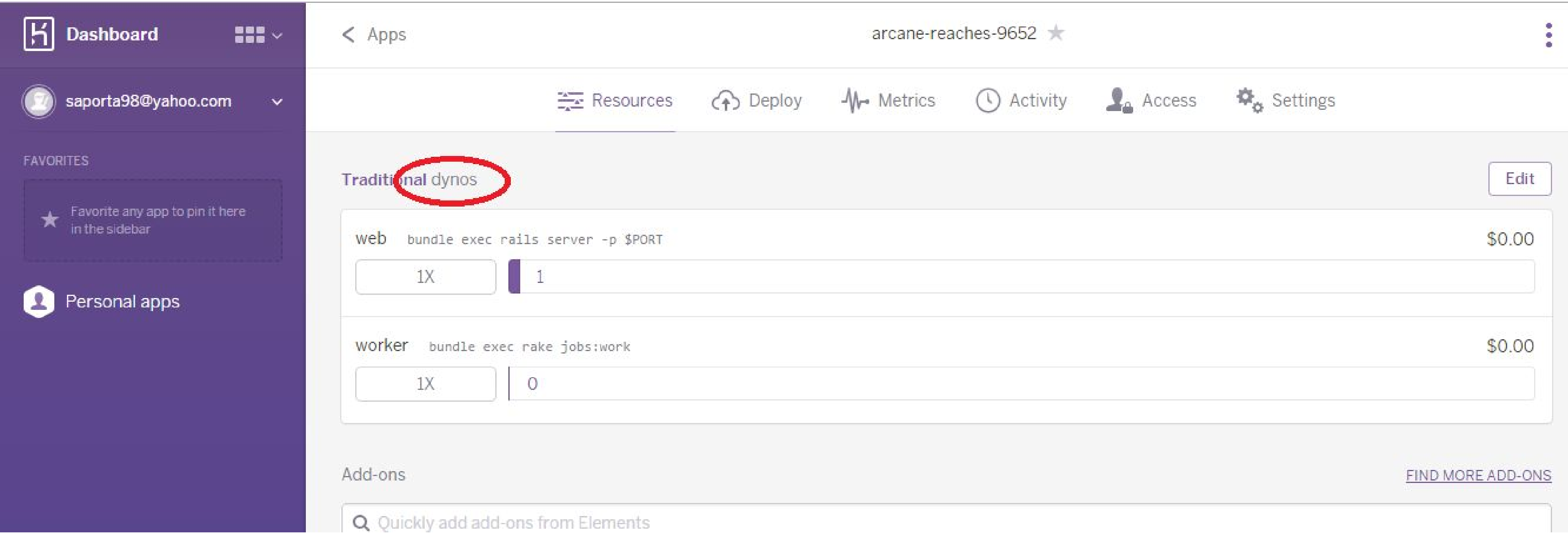
- Simple email service
- Send email by calling an API

SQS

- Simple queue service
- Read and write a queue in the cloud
- Almost infinitely scalable -- ensures data doesn’t get lost while waiting for your app to have time to process it

Platform as a Service (PaaS)

- Hosting provider provisions the whole stack, not just hardware and OS
  - Web server
  - Application server
  - Application framework (e.g. Ruby on Rails)
  - Database (e.g. PostgreSQL)
- Very convenient, but less flexible
- Best-known example: Heroku
- AWS has its own PaaS offering, Elastic Beanstalk



Traditional dynos

web `bundle exec rails server -p $PORT` $0.00

1X 1

worker `bundle exec rake jobs:work` $0.00

1X 0

Add-ons

How is cloud hosting priced?

- Most major cloud hosting providers have similar pricing
- We’ll focus on AWS
- Each service has its own pricing model
- Data transfer (network bandwidth) is never free

EC2 pricing

- Hourly rental fee for each virtual server (called an “Instance”). Some examples:
  - t2.micro - $0.013 / hr - 1 GB RAM, 1 virtual CPU
  - m3.2xlarge - $0.532 / hr - 30 GB RAM, 8 virtual CPUs
- That’s for Linux. Windows Instances are higher, because of Microsoft’s license fee
- More expensive Instances also include more storage and I/O capacity
- Reserve pricing provides discounts of up to 60% if you prepay for 12 or 36 months
- Spot pricing offers the potential for huge discounts when excess Instances are available

EC2 pricing (cont’d)

- Data transfer
  - Data transfer IN is free
  - Data transfer OUT is $0.09 / GB
- Additional storage is $0.05 to $0.125 / GB / mo
- Prices change (decrease!) frequently

EC2 pricing (cont’d)

Free Tier

- Lasts for 12 months
- If your hosting needs are modest, you probably won’t pay anything!
- Even after 12 months some free services are included every month

AWS auto-scaling

- You can define an auto-scaling group
  - Min # of Instances
  - Max # of Instances
  - Rule for increasing # of Instances (time of day, CPU, network traffic, etc)
  - Rule for decreasing # of Instances
- More Instances when you need them, fewer when you don’t
- Pay only for what you use

“Infinitely” scalable services

- S3 has no fixed capacity limit and no need for auto-scaling. Within reason, it handles however many requests the universe sends to it.
- SQS similarly has no fixed limit and no need for auto-scaling.
  - A good architecture is to send requests for long-running services to a queue, with a separate process that consumes the queue
  - Protects your application against spikes, and your users against brief downtime

Summary

- You need hosting if you want to share your applications with the world
- Hosting dynamic, high-traffic, high-availability apps is challenging
- You could operate your own datacenter or lease servers by the month, but cloud hosting or PaaS is an easier, more cost-effective solution for most people
- You can get started with cloud hosting for free

Questions?