# Data Science and Statistics in the Amazon Cloud with R

Louis J. M. Aslett (aslett@stats.ox.ac.uk)

Department of Statistics, University of Oxford
and Corpus Christi College, Oxford

RGU Computing Research Seminar
4 November 2015

i-like.org.uk

# Introduction

# “The old days …”

Redbus Interhouse, Harbour Exchange Square, London Docklands, 24 March 2005

# R in the cloud

By cloud, I mean an online service which allows users to create and destroy virtual servers remotely without having to worry about initial hardware and OS installation and where billing is in very small increments (e.g. hours).

There are several cloud providers, including:

- Amazon EC2
    - `http://aws.amazon.com/`
- Digital Ocean
    - `http://www.digitalocean.com/`
- Google Compute Engine
    - `http://cloud.google.com/`
- Rackspace Cloud Servers
    - `http://www.rackspace.co.uk/`
- Windows Azure VMs
    - `http://www.windowsazure.com/`

# Why R in the cloud?

- Long analyses
- Collaboration
- Powerful AWS instances, with access to GPUs
- Online data sources
- Full environment with C/C++/Fortran, LaTeX+ Sweave, Git + Subversion ready-to-go.
- Access to the power of Linux without having to dedicate your own machine to running it.

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).
- A permanent storage medium for each machine which can persist independently of the running state of the server.

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).
- A permanent storage medium for each machine which can persist independently of the running state of the server.
- Billing which suspends while the server is stopped, but where the above persistent storage remains alive.

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).
- A permanent storage medium for each machine which can persist independently of the running state of the server.
- Billing which suspends while the server is stopped, but where the above persistent storage remains alive.
- A free tier of instances so everyone can try it without cost.

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).
- A permanent storage medium for each machine which can persist independently of the running state of the server.
- Billing which suspends while the server is stopped, but where the above persistent storage remains alive.
- A free tier of instances so everyone can try it without cost.
- Everything from micro instances to the current state of the art in HPC, including machines with nVidia GPUs for CUDA support. (See also benchmarks)

# Why Amazon Web Services (AWS)?

Today, AWS is the only the service which ticks all the following (subjectively) important boxes for HPC in statistics, though this is a *fast* moving business:

- Repository for community development of images so users can boot ready-to-run machines with more than just bare operating system (bit like package system in R).
- A permanent storage medium for each machine which can persist independently of the running state of the server.
- Billing which suspends while the server is stopped, but where the above persistent storage remains alive.
- A free tier of instances so everyone can try it without cost.
- Everything from micro instances to the current state of the art in HPC, including machines with nVidia GPUs for CUDA support. (See also benchmarks)
- A ‘stock-market’ for unused compute capacity, enabling heavily discounted compute jobs.

# Some cons to AWS

There are some cons to AWS, though again these could be addressed in future:

- Billing is in full hour increments (Google Compute Engine is in minutes)
- There are no guarantees of availability: Amazon explicitly intend users to design for instance failure in the use cases. There have been high profile outages in the US East Coast data centre
- High barrier to learning

# AWS Background

# Amazon Web Services (AWS)

Amazon used to buy in huge server capacity to keep their website up just for the Christmas shopping spree ... rest of the year large parts of server farm sat mostly idle.

Launched 2006. By December 2014, $1,400,000$ servers operating in 28 data centres across 7 countries:

- Dublin, Ireland
- Frankfurt, Germany
- North Virginia, United States
- Oregon, United States
- Northern California, United States
- Singapore, Republic of Singapore
- Tokyo, Japan
- Sydney, Australia
- São Paulo, Brazil

# Some of the relevant AWS ecosystem

- EC2 (Elastic Compute Cloud)
    - is the service which enables launching virtual servers
- EBS (Elastic Block Storage)
    - persistent block level storage for use with EC2
- VPC (Virtual Private Cloud)
    - for creation of virtual LANs within AWS for private networking
- S3 (Simple Storage Service)
    - for resiliant storage of data independent of instances
- RDS (Relational Database Server) / DynamoDB
    - managed SQL and NoSQL databases
- SQS (Simple Queuing Service)
    - highly scalable and reliable atomic messaging service

# EC2 Jargon

- Instance
  - a virtual server running on EC2
- Volume
  - a cloud ‘hard drive’ which is attached to an instance
- AMI (Amazon Machine Image)
  - a bundle of operating system and pre-loaded applications to boot on an instance
- Stop -vs- terminate
  - *stop*: similar to powering down a computer. Cease paying by the hour, just pay by the month for EBS volume ($0.10/GB/mo). Start and instance boots same machine.
  - *terminate*: power down and destroy all data.
- On-demand/Spot instance
  - *on demand*: immediately available, fixed price per hour
  - *spot*: ephemeral instance, price follows Amazon ‘stock-market’

# Demo: launching a virtual server on EC2

**Demo** (without launching)

# Interacting with EC2

- Web interface
- REST API accessible from an array of languages/platforms
  - Android, iOS, *nix, Mac, Windows
  - Javascript, Java, .NET, Node.js, PHP, Python, Ruby, Go, shell
  - interestingly, no official R API yet! However, `http://cloudyr.github.io/` looks promising.

# Interacting with EC2 : example

EC2 CLI tools:
https://aws.amazon.com/developertools/351

```
for Reg in eu-west-1 eu-central-1 us-east-1 us-west-1 \
           us-west-2 sa-east-1 ap-southeast-1 \
           ap-northeast-1 ap-southeast-2
do
    ec2-describe-spot-price-history \
        -t m3.medium \
        --region $Reg \
        -d ”Linux/UNIX (Amazon VPC)” \
        -s ‘date ”+%Y-%m-%dT%H:%M:00”‘ \
        | cut -f 2,3,4,6
done;
```

# The RStudio AMI

# Why?

So,

- AWS enables rapid launching of instances from a set of different operating systems.
- However, they're bare bones systems
  - it's up to you to setup up the system as you want it
  - not hard for the technically literate … but definitely *time consuming*
- Any AWS user can create AMIs of systems so that they can boot directly to a system they have previously setup
  - an AMI is like a bare metal image of a system which can be restored to any new instance.
- The AMIs I provide is a public share of the pre-setup R system I use and have integrated user feedback and requests.

See `http://www.louisaslett.com/RStudio_AMI/`

## What's included?

- 10GB EBS image on SSD backed storage

# What’s included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave

# What’s included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
  - ODBC
  - RMySQL pre compiled

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
  - ODBC
  - RMySQL pre compiled
- Git & Subversion support

## What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
  - ODBC
  - RMySQL pre compiled
- Git & Subversion support
- Probabilistic programming languages Stan and JAGS for MCMC sampling

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
    - ODBC
    - RMySQL pre compiled
- Git & Subversion support
- Probabilistic programming languages Stan and JAGS for MCMC sampling
- Arbitrary precision arithmetic and number theory libraries
    - GMP, MPFR, FLINT

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
  - ODBC
  - RMySQL pre compiled
- Git & Subversion support
- Probabilistic programming languages Stan and JAGS for MCMC sampling
- Arbitrary precision arithmetic and number theory libraries
  - GMP, MPFR, FLINT
- Optimised BLAS for accelerated matrix operations

# What's included?

- 10GB EBS image on SSD backed storage
- 64-bit HVM images for best performance
- Full LaTeX distribution, supporting RMarkdown and Sweave
- GSL and CURL
- Database support
    - ODBC
    - RMySQL pre compiled
- Git & Subversion support
- Probabilistic programming languages Stan and JAGS for MCMC sampling
- Arbitrary precision arithmetic and number theory libraries
    - GMP, MPFR, FLINT
- Optimised BLAS for accelerated matrix operations
- Dropbox integration

# What's not included?

- Every R package (obviously!). But, fast AWS based mirror is automatically set for quick installation of new pacakges.
- Easy cluster setup (yet … some things perhaps harder to automate well). See next section of talk.

# Most important thing to remember …

… is to **allow port 80 through the AWS EC2 firewall!**

This is done by setting up your security group on instance launch.

# Demo

**Demo** (including launch, RMarkdown, Stan)

# Your Own R Cluster

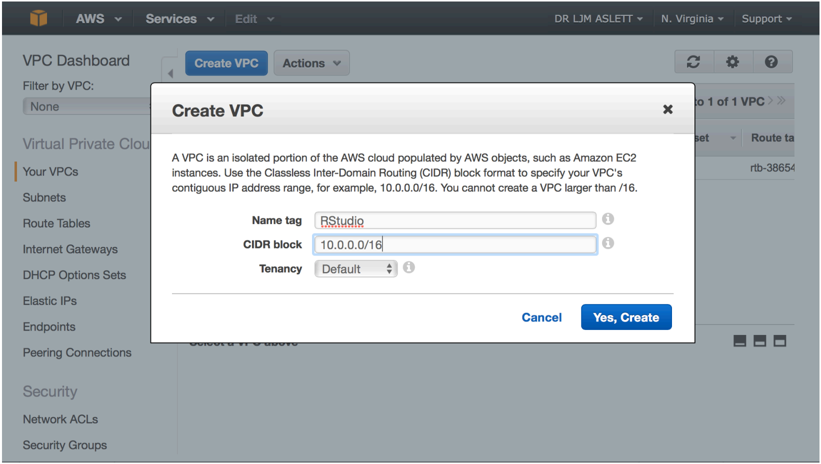
# Cluster setup … if you don’t care about security

Just make sure your security group is open for all ports to the world …

# Cluster setup … if you do care about security

Create a Virtual LAN with private subnet.

Recall, 10.0.0.0/16 $\implies$ 10.0.0.0 – 10.0.255.255 (65536 hosts)



Create VPC

A VPC is an isolated portion of the AWS cloud populated by AWS objects, such as Amazon EC2 instances. Use the Classless Inter-Domain Routing (CIDR) block format to specify your VPC's contiguous IP address range, for example, 10.0.0.0/16. You cannot create a VPC larger than /16.

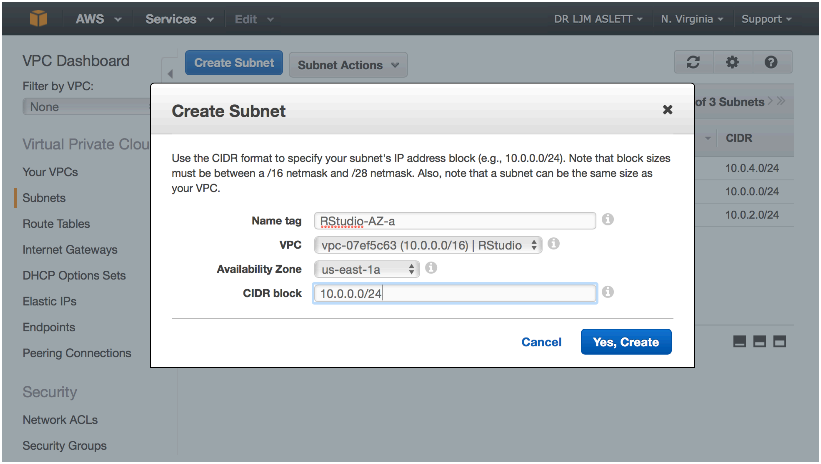
Name tag: RStudio

CIDR block: 10.0.0.0/16

Tenancy: Default

# Split that subnet among the availability zones as you wish

Recall, 10.0.0.0/24 $\implies$ 10.0.0.0 – 10.0.0.255 (256 hosts)



**Create Subnet** ×

Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

**Name tag** RStudio-AZ-a

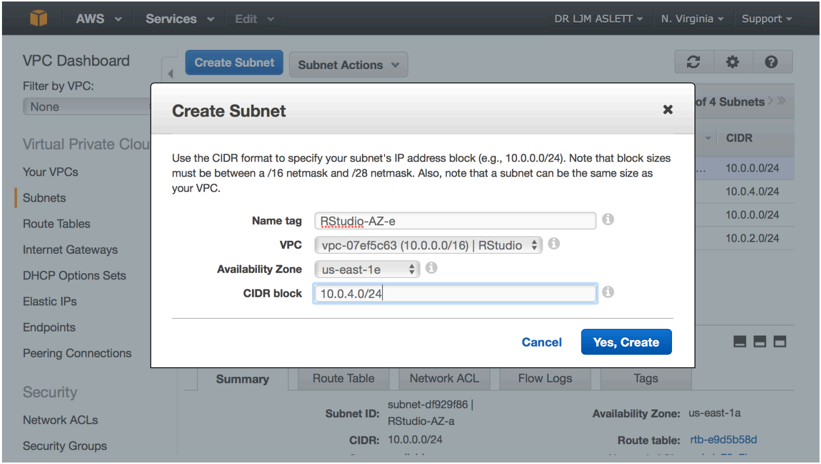
**VPC** vpc-07ef5c63 (10.0.0.0/16) | RStudio

**Availability Zone** us-east-1a

**CIDR block** 10.0.0.0/24

# Split that subnet among the availability zones as you wish

Recall, 10.0.4.0/24 $\Longrightarrow$ 10.0.4.0 – 10.0.4.255 (256 hosts)



| CIDR |
|---|
| 10.0.0.0/24 |
| 10.0.4.0/24 |
| 10.0.0.0/24 |
| 10.0.2.0/24 |

**Create Subnet**

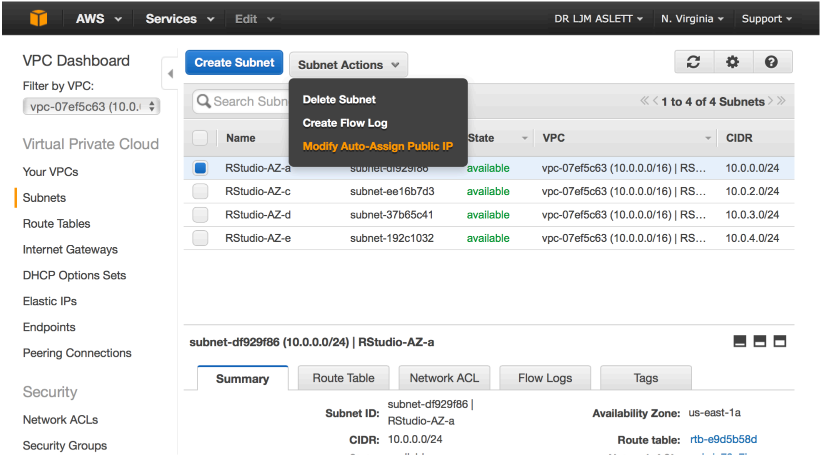
Use the CIDR format to specify your subnet's IP address block (e.g., 10.0.0.0/24). Note that block sizes must be between a /16 netmask and /28 netmask. Also, note that a subnet can be the same size as your VPC.

| | |
|---|---|
| Name tag | RStudio-AZ-e |
| VPC | vpc-07ef5c63 (10.0.0.0/16) \| RStudio |
| Availability Zone | us-east-1e |
| CIDR block | 10.0.4.0/24 |

Subnet ID: subnet-df929f86 | RStudio-AZ-a
CIDR: 10.0.0.0/24
Availability Zone: us-east-1a
Route table: rtb-e9d5b58d

# Enable auto-assignment of public IP (I)

Auto-assigning a public IP will allow you to get in to all hosts remotely for diagnostics etc.



| | Name | | State | VPC | CIDR |
|---|---|---|---|---|---|
| ☑ | RStudio-AZ-a | subnet-df929f86 | available | vpc-07ef5c63 (10.0.0.0/16) \| RS... | 10.0.0.0/24 |
| □ | RStudio-AZ-c | subnet-ee16b7d3 | available | vpc-07ef5c63 (10.0.0.0/16) \| RS... | 10.0.2.0/24 |
| □ | RStudio-AZ-d | subnet-37b65c41 | available | vpc-07ef5c63 (10.0.0.0/16) \| RS... | 10.0.3.0/24 |
| □ | RStudio-AZ-e | subnet-192c1032 | available | vpc-07ef5c63 (10.0.0.0/16) \| RS... | 10.0.4.0/24 |

subnet-df929f86 (10.0.0.0/24) | RStudio-AZ-a

Subnet ID: subnet-df929f86 | RStudio-AZ-a

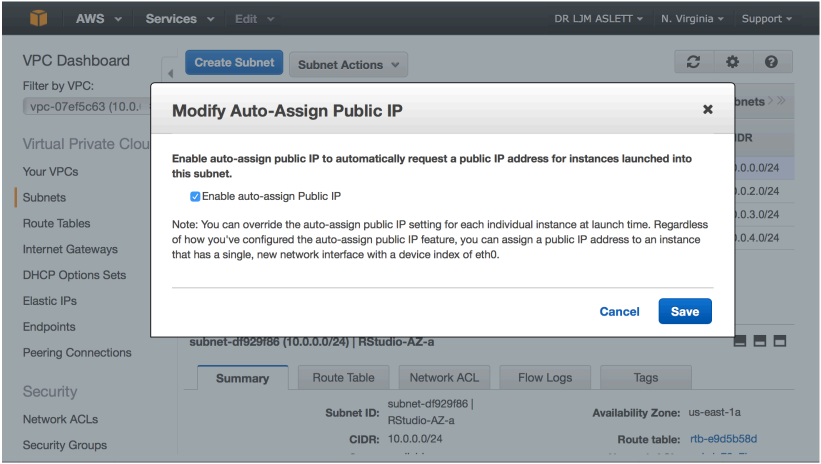
Availability Zone: us-east-1a

CIDR: 10.0.0.0/24

Route table: rtb-e9d5b58d

# Enable auto-assignment of public IP (II)

Auto-assigning a public IP will allow you to get in to all hosts remotely for diagnostics etc.



Subnet ID: subnet-df929f86 | RStudio-AZ-a

CIDR: 10.0.0.0/24

Availability Zone: us-east-1a

Route table: rtb-e9d5b58d

**Modify Auto-Assign Public IP**

**Enable auto-assign public IP to automatically request a public IP address for instances launched into this subnet.**

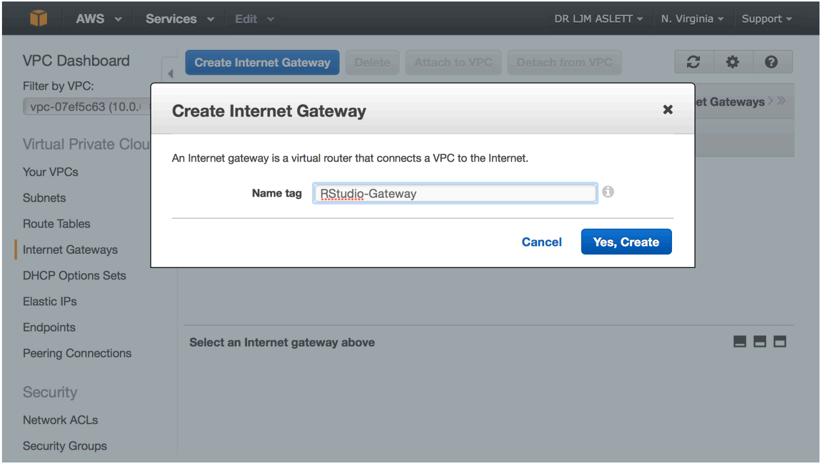
☑ Enable auto-assign Public IP

Note: You can override the auto-assign public IP setting for each individual instance at launch time. Regardless of how you've configured the auto-assign public IP feature, you can assign a public IP address to an instance that has a single, new network interface with a device index of eth0.

Cancel Save

# Add an internet router (I)

Ensure that traffic can be routed between the internet and your LAN …



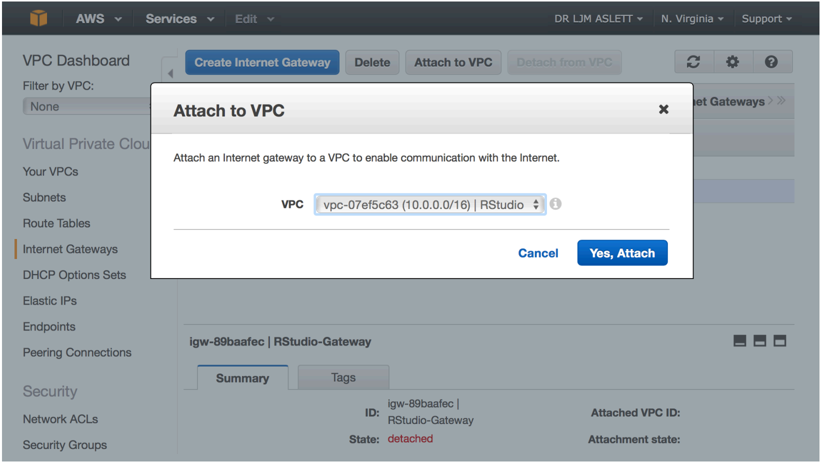
Create Internet Gateway

An Internet gateway is a virtual router that connects a VPC to the Internet.

Name tag: RStudio-Gateway

# Add an internet router (II)

… and attach to your virtual LAN …



Attach to VPC ×

Attach an Internet gateway to a VPC to enable communication with the Internet.

VPC vpc-07ef5c63 (10.0.0.0/16) | RStudio

igw-89baafec | RStudio-Gateway

ID: igw-89baafec | RStudio-Gateway

State: detached

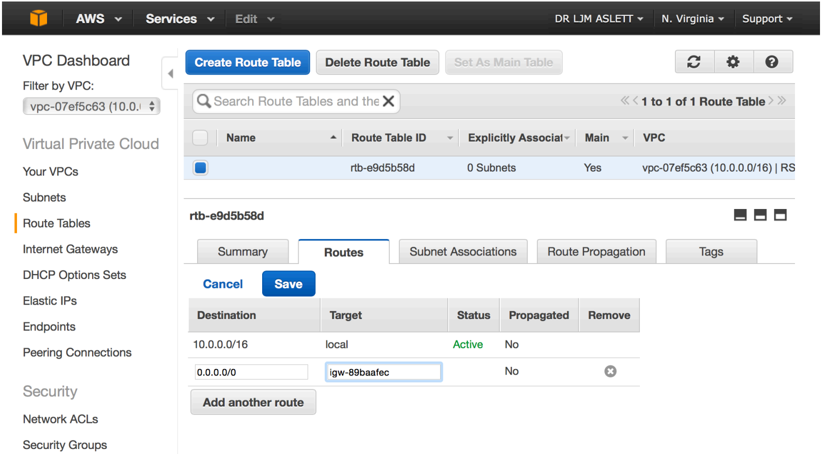
Attached VPC ID:

Attachment state:

i-like.org.uk

# Add an internet router (III)

… and insert to routing table.



| Name | Route Table ID | Explicitly Associat | Main | VPC |
|---|---|---|---|---|
| | rtb-e9d5b58d | 0 Subnets | Yes | vpc-07ef5c63 (10.0.0.0/16) \| RS |

rtb-e9d5b58d

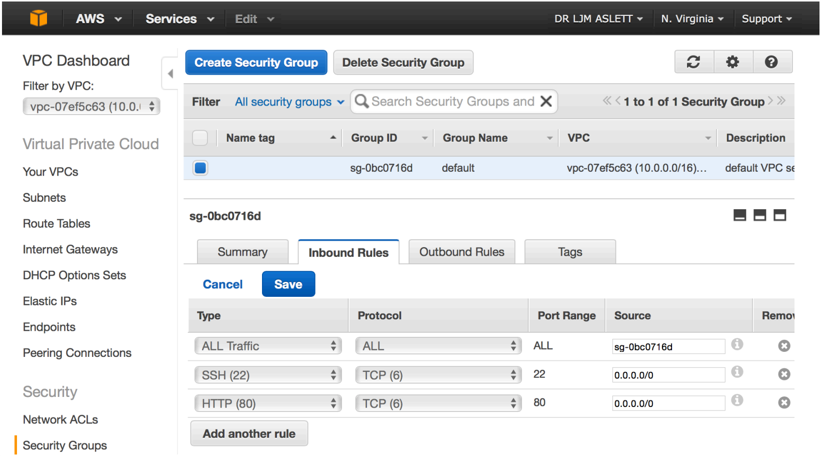
| Destination | Target | Status | Propagated | Remove |
|---|---|---|---|---|
| 10.0.0.0/16 | local | Active | No | |
| 0.0.0.0/0 | igw-89baafec | | No | ⊗ |

i-like.org.uk

# Modify security group

Allow limited inbound traffic over the router so you can SSH and access RStudio web interface.

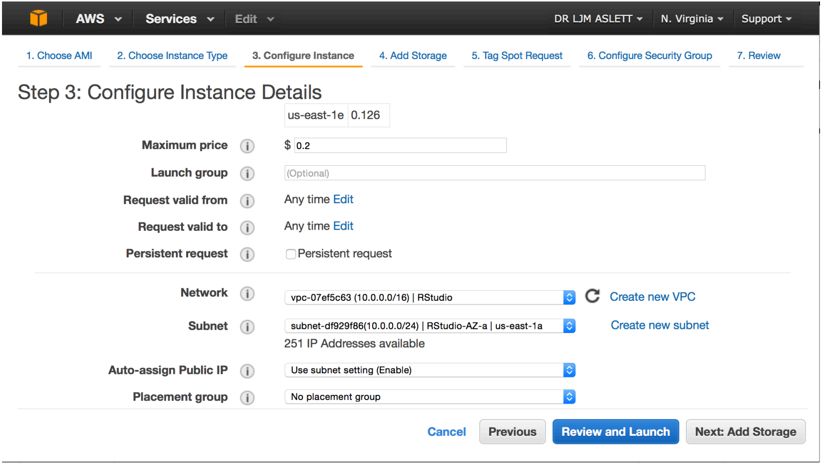
**NOTE:** double check no internal traffic firewalled!



| ☐ | Name tag | Group ID | Group Name | VPC | Description |
|---|---|---|---|---|---|
| ■ | | sg-0bc0716d | default | vpc-07ef5c63 (10.0.0.0/16)... | default VPC se |

| Type | Protocol | Port Range | Source | Remo |
|---|---|---|---|---|
| ALL Traffic | ALL | ALL | sg-0bc0716d | ⊗ |
| SSH (22) | TCP (6) | 22 | 0.0.0.0/0 | ⊗ |
| HTTP (80) | TCP (6) | 80 | 0.0.0.0/0 | ⊗ |

# Ensure you select this VPC when launching …



Step 3: Configure Instance Details

us-east-1e 0.126

Maximum price: $ 0.2

Launch group: (Optional)

Request valid from: Any time Edit

Request valid to: Any time Edit

Persistent request: □ Persistent request

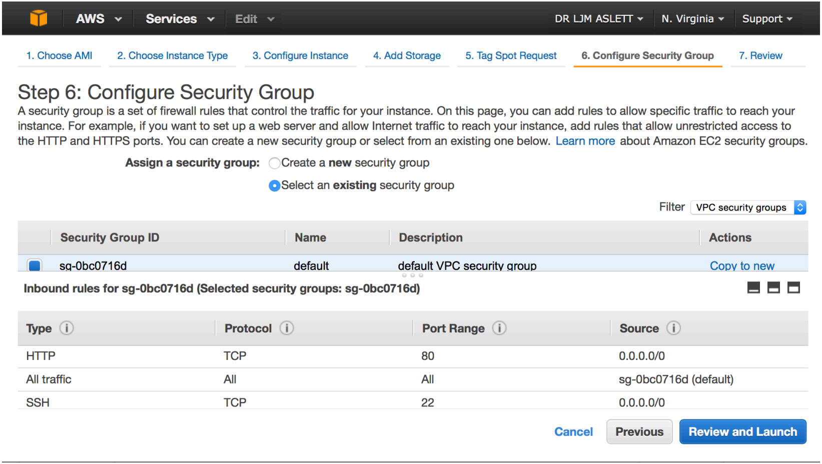
Network: vpc-07ef5c63 (10.0.0.0/16) | RStudio

Subnet: subnet-df929f86(10.0.0.0/24) | RStudio-AZ-a | us-east-1a
251 IP Addresses available

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: No placement group

# … and change security group to the VPC default



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Spot Request **6. Configure Security Group** 7. Review

## Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

**Assign a security group:** ○ Create a **new** security group

● Select an **existing** security group

Filter VPC security groups

| | Security Group ID | Name | Description | Actions |
|---|---|---|---|---|
| ☑ | sg-0bc0716d | default | default VPC security group | Copy to new |

**Inbound rules for sg-0bc0716d (Selected security groups: sg-0bc0716d)**

| Type | Protocol | Port Range | Source |
|---|---|---|---|
| HTTP | TCP | 80 | 0.0.0.0/0 |
| All traffic | All | All | sg-0bc0716d (default) |
| SSH | TCP | 22 | 0.0.0.0/0 |

# Cluster example (I)

Say my private RSA key is in `k.pem`. First, upload to the instance, then make the correct permissions for SSH:

```
system("chmod 600 k.pem")
```

Now, launch R across the cluster with one command:

```
library("parallel")
cl <- makePSOCKcluster(
  rep(c("localhost", "10.0.0.58", "10.0.4.249"), 36),
  rshcmd="ssh -i k.pem -o StrictHostKeyChecking=no",
  user="ubuntu",
  master=system("hostname --all-ip-addresses", TRUE))
```

# Cluster example (II)

Ensure that all the packages you need are installed on all the nodes, then:

- use `clusterEvalQ()`, `clusterApply()` or `clusterApplyLB()` to launch parallel jobs.
- or optionally hook into the foreach framework, for easy to use `%dopar%` construct:

```
library(”doParallel”)
registerDoParallel(cl)
```

# Cluster example (III)

For example, fitting a random forest on large data is then trivial:

```r
library(”randomForest”)
fit <- foreach(nt=rep(5, 36*3), .combine=combine,
               .packages=”randomForest”) %dopar% {
  load(”myDat.RData”)
  randomForest(resp ~ ., data=myDat, ntree=nt)
}
```

# Case Study

# Encrypted statistical machine learning

Modern cryptographic techniques promise to allow privacy to be preserved whilst still allowing computation to be performed, unlike the usual encryption schemes such as AES.

However, these so-called *homomorphic encryption* schemes are currently very computationally demanding and restrictive in the types of computation which can be performed.

Recent work[1] has shown that tailored methods can be built which are competitive with unencrypted machine learning methods, but computational demand is high.

[1] Aslett, L. J. M., Esperança, P. M. and Holmes, C. C. (2015), Encrypted statistical machine learning: new privacy preserving methods. arXiv:1508.06845 [stat.ML].

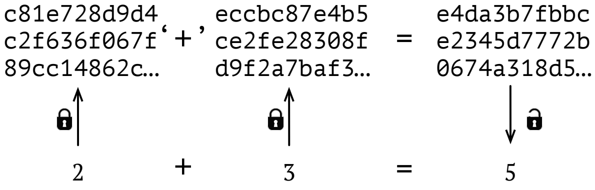
# Homomorphic encryption

## Definition (Homomorphic encryption scheme)

An encryption scheme is said to be *homomorphic* if there is a set of operations $\circ \in \mathcal{F}_M$ acting in message space (such as addition) that have corresponding operations $\diamond \in \mathcal{F}_C$ acting in cipher text space satisfying the property:

$$\texttt{Dec}(k_s, \texttt{Enc}(k_p, m_1) \diamond \texttt{Enc}(k_p, m_2)) = m_1 \circ m_2 \quad \forall\, m_1, m_2 \in M$$

A scheme is *fully homomorphic* if $\mathcal{F}_M = \{+, \times\}$ and an arbitrary number of such operations are possible. Cartoon version:



```
c81e728d9d4      eccbc87e4b5      e4da3b7fbbc
c2f636f067f '+' ce2fe28308f   =   e2345d7772b
89cc14862c...    d9f2a7baf3...    0674a318d5...
```

$$2 + 3 = 5$$

# Fitting a Completely Random Forest (CRF) encrypted

To show the techniques of the paper are practical required showing you can fit a CRF in a reasonable time (while you have lunch, say) and without extreme cost (don't want to write a grant application just for hardware to run it).

Enter Amazon Web Services …

As a proof of concept, we encrypted the Wisonsin breast cancer prognosis data set locally, resulting in 13.8GB of encrypted data.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
  - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
  - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.
- The 13.8GB of encrypted data was uploaded to an S3 bucket in Dublin, Ireland and from there copied to São Paulo, Brazil. It was split into 18 ‘shards’ before upload.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
  - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.
- The 13.8GB of encrypted data was uploaded to an S3 bucket in Dublin, Ireland and from there copied to São Paulo, Brazil. It was split into 18 ‘shards’ before upload.
- 18 spot requests in each region $\implies$ $1,152$ CPU cores.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
    - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.
- The 13.8GB of encrypted data was uploaded to an S3 bucket in Dublin, Ireland and from there copied to São Paulo, Brazil. It was split into 18 'shards' before upload.
- 18 spot requests in each region $\implies 1,152$ CPU cores.
- Each (slighly modified) RStudio AMI instance fitted 50 trees to 1 of the shards.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
    - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.
- The 13.8GB of encrypted data was uploaded to an S3 bucket in Dublin, Ireland and from there copied to São Paulo, Brazil. It was split into 18 ‘shards’ before upload.
- 18 spot requests in each region $\implies$ $1,152$ CPU cores.
- Each (slighly modified) RStudio AMI instance fitted 50 trees to 1 of the shards.
- Upon completion, encrypted subtree uploaded to S3 again.

# The AWS run

- The spot pricing script earlier was used to determine the cheapest region spot prices for the c3.8xlarge instances
    - 32 cores (Intel Xeon E5-2680 v2), 60GB memory.
- The 13.8GB of encrypted data was uploaded to an S3 bucket in Dublin, Ireland and from there copied to São Paulo, Brazil. It was split into 18 ‘shards’ before upload.
- 18 spot requests in each region $\implies$ $1,152$ CPU cores.
- Each (slighly modified) RStudio AMI instance fitted 50 trees to 1 of the shards.
- Upon completion, encrypted subtree uploaded to S3 again.
- Total cost: \$23.86 ($\approx$ £15.66)

# Conclusion

- The opportunity cost to maintain cryptographic security is 2 hours of time and $23.86.
- To buy 36 servers with 32 cores of Intel Xeon Ivy Bridge and 60GB RAM would be prohibitive for the occasional encrypted fit.
- The RStudio AMI reduced the time to working substantially.

# Future plans

# Future plans

- GPU AMI
- more built in cluster tools
  - avoid need to install tools on nodes
  - helper functions to install supporting packages
  - helper function to launch PSOCK clusters
- Migration tools to upgrade when new AMIs released
- Tools to add multiple user accounts and manage server resources

# Feedback

Please give feedback – I would love ideas for what else would make the AMIs more useful!

There is no paper currently, so please use a standard software citation if you use them:

Aslett, L. J. M. (2015), *RStudio AMIs for Amazon EC2 cloud computing*. AMI ID ami-ae05a1d9. **URL:** `http://www.louisaslett.com/RStudio_AMI/`

and simply replace the AMI ID with the version used for reproducibility.

**Thanks**