Practice and Applications of Data Management
CMPSCI 345

Lecture 19-20: Amazon Web Services
Extra credit: project part 3

- Open-ended additional features.

- Presentations on Dec 7

- Need to sign up by Nov 30!
This week

- No class on Wednesday (enjoy Thanksgiving!)

- Office hours on Tuesday 2-3pm.
Map-Reduce Summary

- Hides scheduling and parallelization details
- However, very limited queries
  - Difficult to write more complex tasks
  - Need multiple map-reduce operations
- Solution:
  - Use MapReduce as a runtime for higher level languages
  - Pig (Yahoo!, now apache project): SQL-like operators
  - Hive (apache project): SQL
  - Scope (MS): SQL! But proprietary...
  - DryadLINQ (MS): LINQ! But also proprietary...
Homework assignment

- Amazon Web Services
  - You need to sign up!
  - Practice large-scale unstructured data processing on Hadoop

- This (and next) week:
  - Overview of AWS in class
  - Guiding through the first steps of the assignment.
Amazon Web Services (AWS)

- A cloud computing platform
Why cloud computing?
What will we learn?

analyze search logs

<table>
<thead>
<tr>
<th>IP Address</th>
<th>Search Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED75271605EBD0C</td>
<td>970916201045 yahoo chat</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184818 garter belts</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184939 lingerie</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185051 spiderman</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185155 tommy hilfiger</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185257 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185513 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185605 exhibitionists</td>
</tr>
</tbody>
</table>

......

......

......
What is Pig?

- An engine for executing programs on top of Hadoop
- It provides a language, Pig Latin, to specify these programs
- An Apache open source project
Why use Pig?

Suppose you have user data in one file, website data in another, and you need to find the top 5 most visited sites by users aged 18 - 25.

Load Users

Filter by age

Load Pages

Join on name

Group on url

Count clicks

Order by clicks

Take top 5
public class MRExample {

    public static class LoadPages extends MapReduceBase {
        implements Mapper<LongWritable, Text, Text, Text> {
            public void map(LongWritable k, Text val, OutputCollector<Text, Text> out, Reporter reporter) throws IOException {
                // If we have a comma, split the line at that position.
                int comma = val.toString().indexOf(',');
                if (comma == -1) {
                    out.collect(val, val);
                } else {
                    out.collect(Text.from(val.toString().substring(0, comma)), Text.from(val.toString().substring(comma + 1))); // output key, new LongWritable((k));
                }
            }
        }
        public static class LoadPagesFilter extends MapReduceBase {
            implements Mapper<LongWritable, Text, Text, Text> {
                public void map(LongWritable k, Text val, OutputCollector<Text, Text> out, Reporter reporter) throws IOException {
                    // If we have a comma, split the line at that position.
                    int comma = val.toString().indexOf(',');
                    if (comma == -1) {
                        out.collect(val, val);
                    } else {
                        out.collect(Text.from(val.toString().substring(0, comma)), Text.from(val.toString().substring(comma + 1))); // output key, new LongWritable((k));
                    }
                }
            }
        }
    }
    public static class Join extends MapReduceBase {
        public static class Join extends MapReduceBase {
            implements Reducer<LongWritable, Text, Text, Text> {
                public void reduce(LongWritable k, Text val, OutputCollector<Text, Text> out, Reporter reporter) throws IOException {
                    // If we have a comma, split the line at that position.
                    int comma = val.toString().indexOf(',');
                    if (comma == -1) {
                        out.collect(val, val);
                    } else {
                        out.collect(Text.from(val.toString().substring(0, comma)), Text.from(val.toString().substring(comma + 1))); // output key, new LongWritable((k));
                    }
                }
            }
        }
    }

    public static class LimitClips extends MapReduceBase {
        public static class LimitClips extends MapReduceBase {
            implements Reducer<LongWritable, Text, Text, Text> {
                public void reduce(LongWritable k, Text val, OutputCollector<Text, Text> out, Reporter reporter) throws IOException {
                    // If we have a comma, split the line at that position.
                    int comma = val.toString().indexOf(',');
                    if (comma == -1) {
                        out.collect(val, val);
                    } else {
                        out.collect(Text.from(val.toString().substring(0, comma)), Text.from(val.toString().substring(comma + 1))); // output key, new LongWritable((k));
                    }
                }
            }
        }
    }
}

170 lines of code, 4 hours to write
In Pig Latin

Users = load ‘users’ as (name, age);
Fltrd = filter Users by
   age >= 18 and age <= 25;
Pages = load ‘pages’ as (user, url);
Jnd = join Fltrd by name, Pages by user;
Grpd = group Jnd by url;
Smmfd = foreach Grpd generate group,
   COUNT(Jnd) as clicks;
Srtd = order Smmfd by clicks desc;
Top5 = limit Srtd 5;
store Top5 into ‘top5sites’;

9 lines of code, 15 minutes to write
But how good is it?
Essence of Pig

- Map-Reduce is too low a level to program, SQL too high
- Pig Latin, a language intended to sit between the two:
  - Imperative
  - Provides standard relational transforms (join, sort, etc.)
  - Schemas are optional, used when available, can be defined at runtime
  - User Defined Functions are first class citizens
  - Opportunities for advanced optimizer but optimizations by programmer also possible
Multi-store script

A = load 'users' as (name, age, gender, city, state);
B = filter A by name is not null;
C1 = group B by age, gender;
D1 = foreach C1 generate group, COUNT(B);
store D into 'bydemo';
C2 = group B by state;
D2 = foreach C2 generate group, COUNT(B);
store D2 into 'bystate';
What are people doing with Pig

- At Yahoo ~70% of Hadoop jobs are Pig jobs
- Being used at Twitter, LinkedIn, and other companies
- Available as part of Amazon EMR web service and Cloudera Hadoop distribution
- What users use Pig for:
  - Search infrastructure
  - Ad relevance
  - Model training
  - User intent analysis
  - Web log processing
  - Image processing
  - Incremental processing of large data sets
What will we learn?

**analyze search logs**

<table>
<thead>
<tr>
<th>Hash</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED75271605EBD0C</td>
<td>970916201045 yahoo chat</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184818 garter belts</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184939 lingerie</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185051 spiderman</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185155 tommy hilfiger</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185257 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185513 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185605 exhibitionists</td>
</tr>
</tbody>
</table>

**analyze small search logs**

<table>
<thead>
<tr>
<th>Hash</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED75271605EBD0C</td>
<td>970916201045 yahoo chat</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184818 garter belts</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184939 lingerie</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185051 spiderman</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185155 tommy hilfiger</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185257 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185513 calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185605 exhibitionists</td>
</tr>
</tbody>
</table>
AWS assignment

Prepare for class: get started right away
This is an easy assignment, but it requires setting up and using AWS, which is usually time consuming. We will go through most of the configuration steps of the assignment together, in class. However, to benefit from this practice, you need to come prepared: You need to make sure that you are set up to use Amazon Web Services. The first thing that you should do is set up an AWS account. The setup process may incur some delay by Amazon, so you want to do it quickly! Once your account is set up, complete the remaining steps of part B. Setting up AWS. You don’t need to worry about parts A and C for now, but you are welcome to try to tackle them before the November 17 lecture if you wish.

Finally, you will need to download putty.exe to gain access to the ec2 instance. Putty is a free remote access and secure shell (SSH) client for Windows, Linux, MacOSX (built-in Linux and OS X). On Windows, you may need to download Putty, the username that you use to connect to AWS. If you use ec2-user as shown in the screenshot.

How to proceed
Follow the instructions in the sections below to complete the actual assignment questions:

1. Preliminaries
2. Setting up Pig and AWS
3. Questions

Help with getting set up

Information on Pig, Hadoop, and AWS

Actual assignment

NOTE: You need to keep your AWS use down to avoid using up your AWS credits from us and getting charged real money. The provided codes are way more than what you will need for this assignment, but if you forget jobs running, the costs can rack up before you realize. Once you finish the Pig tutorial on AWS, we recommend you do the following:
Running Hadoop on your machines

Setting up – Part A

1. Extract hw3.zip

2. Extract pigttmp.zip

3. Extract hadoop-0.18.3.zip
Setting up

Make sure hadoop is executable:

$ chmod u+x ~/hw3/hadoop-0.18.3/bin/hadoop
Setting up

Set environment variables:

$ export PIGDIR=~/.hw3/pigtmp
$ export HADOOP=~/.hw3/hadoop-0.18.3
$ export HADOOPSITEPATH=~/.hw3/hadoop-0.18.3/conf/
$ export PATH=$HADOOP/bin/:$PATH

In Windows:
$ set PIGDIR=~/.hw3/pigtmp
...etc...
Setting up

The variable JAVA_HOME should be set to point to your system's Java directory. **System dependent!**

In OS X:
$ export JAVA_HOME=$(/usr/libexec/java_home)

In Windows, it should point to your JDK folder. (You should have that from project part 2.)
The data: search query logs

Excite: old search engine (something like google)
The data

- Take a peak inside excite-small.log

<table>
<thead>
<tr>
<th>User ID</th>
<th>Time</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>BED75271605EBD0C</td>
<td>970916201045</td>
<td>yahoo chat</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184818</td>
<td>garter belts</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916184939</td>
<td>lingerie</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185051</td>
<td>spiderman</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185155</td>
<td>tommy hilfiger</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185257</td>
<td>calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185513</td>
<td>calgary</td>
</tr>
<tr>
<td>824F413FA37520BF</td>
<td>970916185605</td>
<td>exhibitionists</td>
</tr>
</tbody>
</table>
**script1-local.pig**

- **Objective:**
  - Find query phrases that occur with high frequency during certain times of day

- Open script1-local.pig
REGISTER .:/tutorial.jar;

raw = LOAD 'excite-small.log' USING PigStorage('\t') AS (user, time, query);

clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query);

clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.tutorial.ToLower(query) as query;

...
... 
houred = FOREACH clean2 GENERATE user,
org.apache.pig.tutorial.ExtractHour(time) as hour,
query;

ngramed1 = FOREACH houred GENERATE user, hour,
flatten(org.apache.pig.tutorial.NGramGenerator(query)
) as ngram;

ngramed2 = DISTINCT ngramed1;

hour_frequency1 = GROUP ngramed2 BY (ngram, hour);

...
Count the occurrences of each n-gram

`hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0), COUNT($1) as count;`

Generate n-grams from the query string

`uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;`

Use a UDF to compute a popularity score for the n-gram

`uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0),
                flatten(org.apache.pig.tutorial.ScoreGenerator($1));`

Assigns names to the fields

`uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0 as ngram, $2 as score, $3 as count, $4 as mean;`

...
filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;

ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour, score;

STORE ordered_uniq_frequency INTO 'script1-local-results.txt' USING PigStorage();
Execute your Pig script

$ java -cp $PIGDIR/pig.jar org.apache.pig.Main
-x local script1-local.pig

$ ls -l script1-local-results.txt

$ cat script1-local-results.txt
Explore what happens

Start grunt:

```bash
$ java -cp $PIGDIR/pig.jar org.apache.pig.Main
-x local
grunt>
```

Copy and paste commands from the script
Explore the created tables with the commands describe and dump
Sign in the AWS management console

- https://console.aws.amazon.com
Check your S3 storage

- https://console.aws.amazon.com
Go to Elastic MapReduce

- https://console.aws.amazon.com
Software Configuration

Vendor □ Amazon □ MapR
Release 2.4.2

- Hadoop 1.0.3
- Pig 0.11.1.1
- Hive 0.11.0.1
- HBase 0.92.0
- Ganglia

Hue is unsupported on this selected AMI.

AMI versions (3.x and 2.x) are configured differently than EMR Releases (4.x). You must use the configure-Hadoop bootstrap action or application specific configuration for Hadoop and Spark.

Add steps (optional)

Step type Select a step Configure

□ Auto-terminate cluster after the last step is completed
## Hardware Configuration

If you need more than 20 EC2 instances, complete this form.

**Network**
- Launch into EC2-Classic
- Create a VPC

*Tip: To create a cluster in a VPC, you must first create a VPC. For more information, click here.

**EC2 availability zone**
- No preference

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>EC2 instance type</th>
<th>Count</th>
<th>Request spot</th>
<th>Bid price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>Master instance group - 1</td>
<td>m1.small</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core</td>
<td>Core instance group - 2</td>
<td>m1.small</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task</td>
<td>Task instance group - 3</td>
<td>m1.small</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add task instance group

[Add task instance group]
Security Options

EC2 key pair  ameliKey

Cluster visible to all IAM users in account

Permissions

Default  Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role  EMR_DefaultRole

EC2 instance profile  EMR_EC2_DefaultRole

EC2 Security Groups

Encryption Options

Cancel  Previous  Create cluster
Starting the job

The job may take a few minutes to start

Cluster: Hw3  Starting  Provisioning Amazon EC2 capacity

Connections:  --
Master public DNS:  --
Tags:  --  View All / Edit

Summary

- ID: j-1FMFDEFKZE5SS
- Creation date: 2015-11-30 12:03 (UTC-5)
- Elapsed time: 1 minute
- Auto-terminate: No
- Termination protection: Off

Network and Hardware

Configuration Details

- AMI version: 2.4.2
- Hadoop: Amazon 1.0.3
distribution:
- Applications: Hive 0.11.0.1, Pig 0.11.1.1
- Log URI: --
- EMRFS: Disabled
consistent
view:

Security and Access
Cluster list

![Cluster list](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Status</th>
<th>Creation time (UTC-5)</th>
<th>Elapsed time</th>
<th>Normalized instance hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hw3</td>
<td>j-1FMFDEFKZE5SS</td>
<td>Starting</td>
<td>2015-11-30 12:03 (UTC-5)</td>
<td>2 minutes</td>
<td>0</td>
</tr>
</tbody>
</table>

Monitors elapsed time
Terminates the job

Terminates the job

SSH instructions

DNS name of Master node
Connecting to the Master

Find your Master’s DNS from the console

$ ssh -i </path/to/saved/keypair/file.pem> hadoop@<master.public-dns-name.amazonaws.com>

Use the name of the master, and the path to your EC2 key pair
On the Master

Create a directory on the HDFS system:

```bash
% hadoop dfs -mkdir /user/hadoop
```
raw = LOAD 'excite.log.bz2' USING PigStorage('\t') AS (user, time, query);

STORE ordered_uniq_frequency INTO 'script1-hadoop-results' USING PigStorage();

Change the location of the data to the one on your S3 bucket:
s3n://<name_of_your_bucket>/excite.log.bz2

Change the location of the output:
/user/hadoop/script1-hadoop-results
Upload files to the Master

Again, use the name of the master, and the path to your EC2 key pair
On the Master

Execute the script:

```bash
% pig -l . script1-hadoop.pig
```
Cluster: Hw3  |  Waiting  | Waiting after step completed

Connections: Enable Web Connection – Job Tracker … (View All)
Master public DNS: ec2-54-204-169-47.compute-1.amazonaws.com  |  SSH
Tags: --  |  View All / Edit

<table>
<thead>
<tr>
<th>Summary</th>
<th>Configuration Details</th>
<th>Network and HA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID:</strong> j-1FMFDFKZE5SS</td>
<td><strong>AMI version:</strong> 2.4.2</td>
<td><strong>Availability zone:</strong></td>
</tr>
<tr>
<td><strong>Creation date:</strong> 2015-11-30 12:03 (UTC-5)</td>
<td><strong>Hadoop</strong> Amazon 1.0.3</td>
<td><strong>Subnet ID:</strong></td>
</tr>
<tr>
<td><strong>Elapsed time:</strong> 12 minutes</td>
<td><strong>distribution:</strong></td>
<td><strong>Master:</strong></td>
</tr>
<tr>
<td><strong>Auto-terminate:</strong> No</td>
<td><strong>Applications:</strong> Hive 0.11.0.1, Pig 0.11.1.1</td>
<td><strong>Core:</strong></td>
</tr>
<tr>
<td><strong>Termination protection:</strong> Off</td>
<td><strong>Log URI:</strong> --</td>
<td><strong>Task:</strong></td>
</tr>
<tr>
<td>Change</td>
<td><strong>EMRFS</strong> Disabled</td>
<td></td>
</tr>
<tr>
<td>consistent view:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In a new terminal window:

$ ssh -i </path/to/saved/keypair/file.pem> -ND 8157 hadoop@$<master.public-dns-name.amazonaws.com>

Starts a proxy listening on port 8157

Use the name of the master, and the path to your EC2 key pair
Enable FoxyProxy on the browser
**Monitoring jobs**

**Cluster: Hw3**
- **Status:** Waiting
- **Connections:**
  - Job Tracker ...
- **Master public DNS:** ec2-54-204-169-47.compute-1.amazonaws.com
- **SSH**
- **Tags:** View All / Edit

<table>
<thead>
<tr>
<th>Summary</th>
<th>Configuration Details</th>
<th>Network and HA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID:</strong> j-1FMFDEFKZE5SS</td>
<td><strong>AMI version:</strong> 2.4.2</td>
<td></td>
</tr>
<tr>
<td><strong>Creation date:</strong> 2015-11-30 12:03 (UTC-5)</td>
<td><strong>Hadoop</strong> Amazon 1.0.3</td>
<td></td>
</tr>
<tr>
<td><strong>Elapsed time:</strong> 12 minutes</td>
<td><strong>distribution:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Auto-terminate:</strong> No</td>
<td><strong>Applications:</strong> Hive 0.11.0.1, Pig 0.11.1.1</td>
<td></td>
</tr>
<tr>
<td><strong>Termination protection:</strong> Off</td>
<td><strong>Log URI:</strong> --</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>EMRFS</strong> Disabled</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>consistent view:</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Access monitoring URLs**
Load the jobtracker


### Running Jobs

<table>
<thead>
<tr>
<th>Jobid</th>
<th>Started</th>
<th>Priority</th>
<th>User</th>
<th>Name</th>
<th>Map % Complete</th>
<th>Map Total</th>
<th>Maps Completed</th>
<th>Reduce % Complete</th>
<th>Reduce Total</th>
<th>Reduces Completed</th>
<th>Job Scheduling Information</th>
<th>Diagnostic Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>job_201304161658_0002</td>
<td>Tue Apr 16 17:19:16 UTC 2013</td>
<td>NORMAL</td>
<td>hadoop</td>
<td>PigLatin:script1-hadoop,pig</td>
<td>54.41%</td>
<td>1</td>
<td>0</td>
<td>0.00%</td>
<td>1</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kind</th>
<th>% Complete</th>
<th>Num Tasks</th>
<th>Pending</th>
<th>Running</th>
<th>Complete</th>
<th>Killed</th>
<th>Failed/Killed Task Attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>100.00%</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>reduce</td>
<td>85.72%</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 / 0</td>
</tr>
</tbody>
</table>
Retrieving results

On the Master:

```
% hadoop dfs --copyToLocal /user/hadoop/script1-hadoop-results script1-hadoop-results
```

On your machine:

```
$ scp -i </path/to/saved/keypair/file.pem>
-r hadoop@<master.public-dns-name.amazonaws.com>:/~script1-hadoop-results/ .
```
Terminate all jobs when you are done!

If you forget jobs running, costs will rack up.

You are responsible for your usage.
Relational DB on AWS

- https://console.aws.amazon.com
Resources

You are using the following Amazon RDS resources in the US East (N. Virginia) region:

- **DB Instances (1)**
- **DB Snapshots (1)**
  - Manual (1)
  - Automated (0)
- **DB Parameter Groups (1)**
- **Security Groups (2)**
- **Reserved DB Purchases (0)**
- **Recent Events (3)**
- **Supported Platforms**: EC2, VPC
- **Default Network**: none

Create Instance

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud.

![Launch a DB Instance](button)

Note: Your DB Instances will launch in the US East (N. Virginia) region:
Create DB Security Group

Pick a name a description
<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>No Authorizations</td>
<td>default</td>
</tr>
<tr>
<td>testgroup</td>
<td>No Authorizations</td>
<td>postgres group</td>
</tr>
</tbody>
</table>
This DB Security Group has no authorizations. You will not be able to connect to DB Instances associated with this security group until you add an authorization. Select a Connection Type below to add an authorization.

<table>
<thead>
<tr>
<th>Connection Type</th>
<th>Details</th>
<th>Status</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIDR/IP</td>
<td>CIDR/IP: 128.119.246.224/32</td>
<td>authorized</td>
<td>Remove</td>
</tr>
</tbody>
</table>

CIDR/IP of your current machine: 128.119.246.224/32

CIDR/IP to Authorize*: 128.119.246.224/32

Authorize
Amazon Relational Database Service (RDS) is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. We currently offer MySQL, SQL Server, Postgres and Oracle engines, allowing you to use the code, application and tools you already use with your existing database with Amazon RDS. You can find pricing information for RDS [here](#) Click the Launch DB Instance button to get started.

Note: Your DB Instances will launch in the US East (N. Virginia) region.
Select Engine

To get started, choose a DB Engine below and click Select.

- MySQL
- PostgreSQL
- PostgreSQL
- ORACLE
- Microsoft SQL Server
Do you plan to use this database for production purposes?

For databases used in production or pre-production we recommend:

- Multi-AZ Deployment for high availability (99.95% monthly up time SLA)
- Provisioned IOPS Storage for fast, consistent performance

Billing is based upon the RDS pricing table.
An instance which uses these features is not eligible for the RDS Free Usage Tier.

- Yes, use Multi-AZ Deployment and Provisioned IOPS Storage as defaults while creating this instance

- No, this instance is intended for use outside of production or under the RDS Free Usage Tier
Specify DB Details

Instance Specifications

- **DB Engine**: postgres
- **License Model**: postgresql-license
- **DB Engine Version**: 9.4.4
- **DB Instance Class**: db.t1.micro — 1 vCPU, 0.613 GiB RAM
- **Multi-AZ Deployment**: No
- **Storage Type**: General Purpose (SSD)
- **Allocated Storage**: 5 GB

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. 
Click here for more details.

Settings

- **DB Instance Identifier**
  - testinstance
- **Master Username**
  - ameli
- **Master Password**
  - ********
- **Confirm Password**
  - ********
VPC: Not in VPC
Availability Zone: No Preference
DB Security Group(s): Create new Security Group
default
testgroup

Database Options

Database Name: cloud_db
Database Port: 5432
DB Parameter Group: default.postgres9.4
Option Group: default:postgres-9-4
Copy Tags To Snapshots: No
Enable Encryption: No

The selected Engine or DB Instance Class does not support storage encryption.

Backup

Backup Retention Period: 0 days
Connect to the cloud database

Use the DB instance address from your console

Type the command in a single line

Type the username you chose
Import data to RDS

In your phpExample code

```bash
psql
  -f initialize.sql
  --host=<your_RDS_instance>
  --port=5432
  --username=<username>
  --password
  --dbname=cloud_db
```
Update your configuration file

Enter the proper values in config.php
Start a local http server.
E.g., with php 5.4: `php -S localhost:8000`

Retrieving data from a database

The div element below populates a table from the data in a database. DB configuration is in config.php.

field
Hello World!

Entering data to a database

After you finish entering something in the field below, the data will be pushed to the database, and the table above will reload.

Enter new data here: 
This is what you entered:
Remember to delete your instance when you no longer need it.