Deploying your web application with AWS Elastic Beanstalk

Julien Simon
Principal Technical Evangelist
julsimon@amazon.fr
@julsimon

amazon web services

Breaking news!

AWS will open a Region
in France in 2017



amazon web services

AWS Compute

EC2

Amazon Elastic Compute Cloud (EC2) provides resizable compute capacity in the cloud.

Elastic Beanstalk

AWS Elastic Beanstalk is an application container for deploying and managing applications.

Lambda

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you.

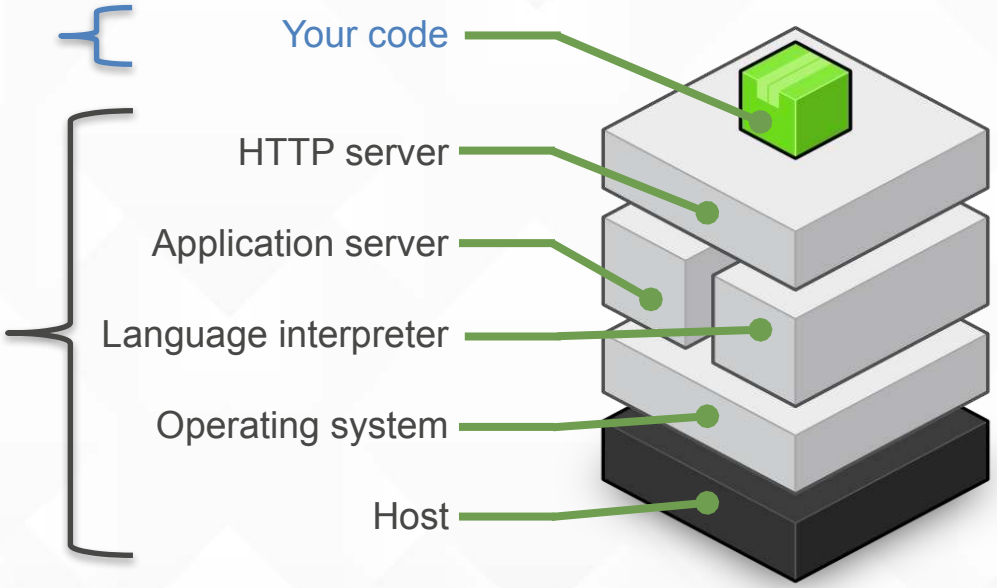
EC2 Container Service

Amazon ECS allows you to easily run and manage Docker containers across a cluster of Amazon EC2 instances.

ElasticBeanstalk vs. DIY

Focus on building your application

Elastic Beanstalk configures each EC2 instance in your environment with the components necessary to run applications for the selected platform. No more worrying about logging into instances to install and configure your application stack.



All you need is code

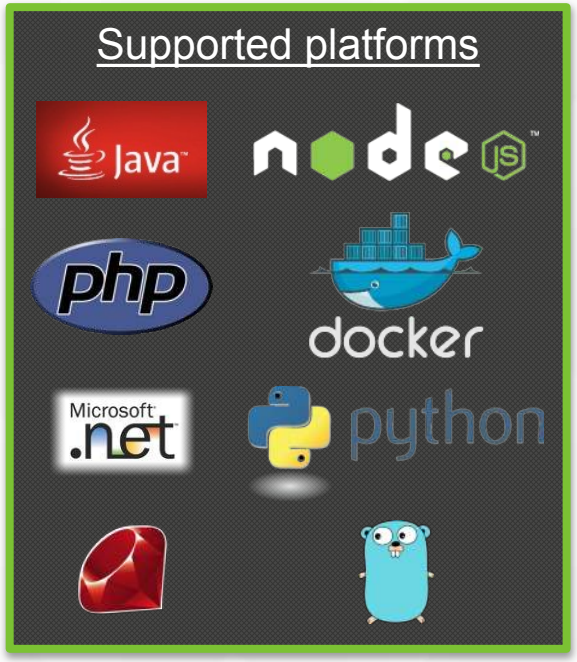
Your code | ‘eb’ CLI

01 Region

02 Stack (container) type

03 Single-instance Or Load balanced with Auto Scaling

04 Database (RDS) Optional



Supported platforms

http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.platforms.html

docker-1.11.2
docker-1.6.2
docker-1.7.1
docker-1.9.1
multi-container-docker-1.11.2-(generic)
multi-container-docker-1.6.2-(generic)

glassfish-4.0-java-7-(preconfigured-docker)
glassfish-4.1-java-8-(preconfigured-docker)

java-7
java-8

tomcat-6
tomcat-7
tomcat-7-java-6
tomcat-7-java-7
tomcat-8-java-8

go-1.3-(preconfigured-docker)
go-1.4
go-1.4-(preconfigured-docker)
go-1.5

iis-7.5
iis-8
iis-8.5

node.js

php-5.3
php-5.4
php-5.5
php-5.6
php-7.0

python-2.7
python-3.4
python-3.4-(preconfigured-docker)

ruby-1.9.3
ruby-2.0-(passenger-standalone)
ruby-2.0-(puma)
ruby-2.1-(passenger-standalone)
ruby-2.1-(puma)
ruby-2.2-(passenger-standalone)
ruby-2.2-(puma)
ruby-2.3-(passenger-standalone)
ruby-2.3-(puma)

amazon web services

Managed environment updates

Elastic Beanstalk | TestApp | Create New Environment

All Applications > TestApp > TestEnv (Environment ID: e-7nghuzgehm, URL: example.sa-east-1.elasticbeanstalk.com) | Actions

- Dashboard
- Configuration
- Logs
- Health
- Monitoring
- Alarms
- Managed Updates NEW
- Events
- Tags

Managed Updates Overview

A new platform version is available. A platform update has been has been scheduled to run during the next maintenance window, between **Wednesday, April 20th 2:30 PM** and **Wednesday, April 20th 4:30 PM (-0700 GMT)**. To perform the update immediately, choose **Apply Now**.

Apply now

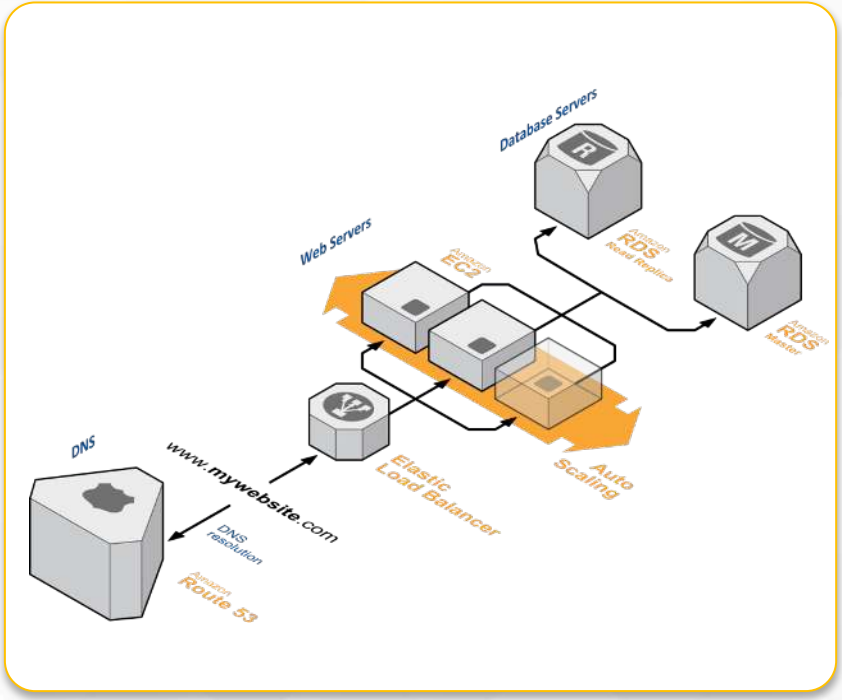
History

Start Time	Durati...	Update Information	Result
2016-04-14 11:42:21 UTC-0700	00:09:34	Environment instance replacement	COMPLETED
2016-04-14 10:45:51 UTC-0700	00:09:15	Environment instance replacement	COMPLETED

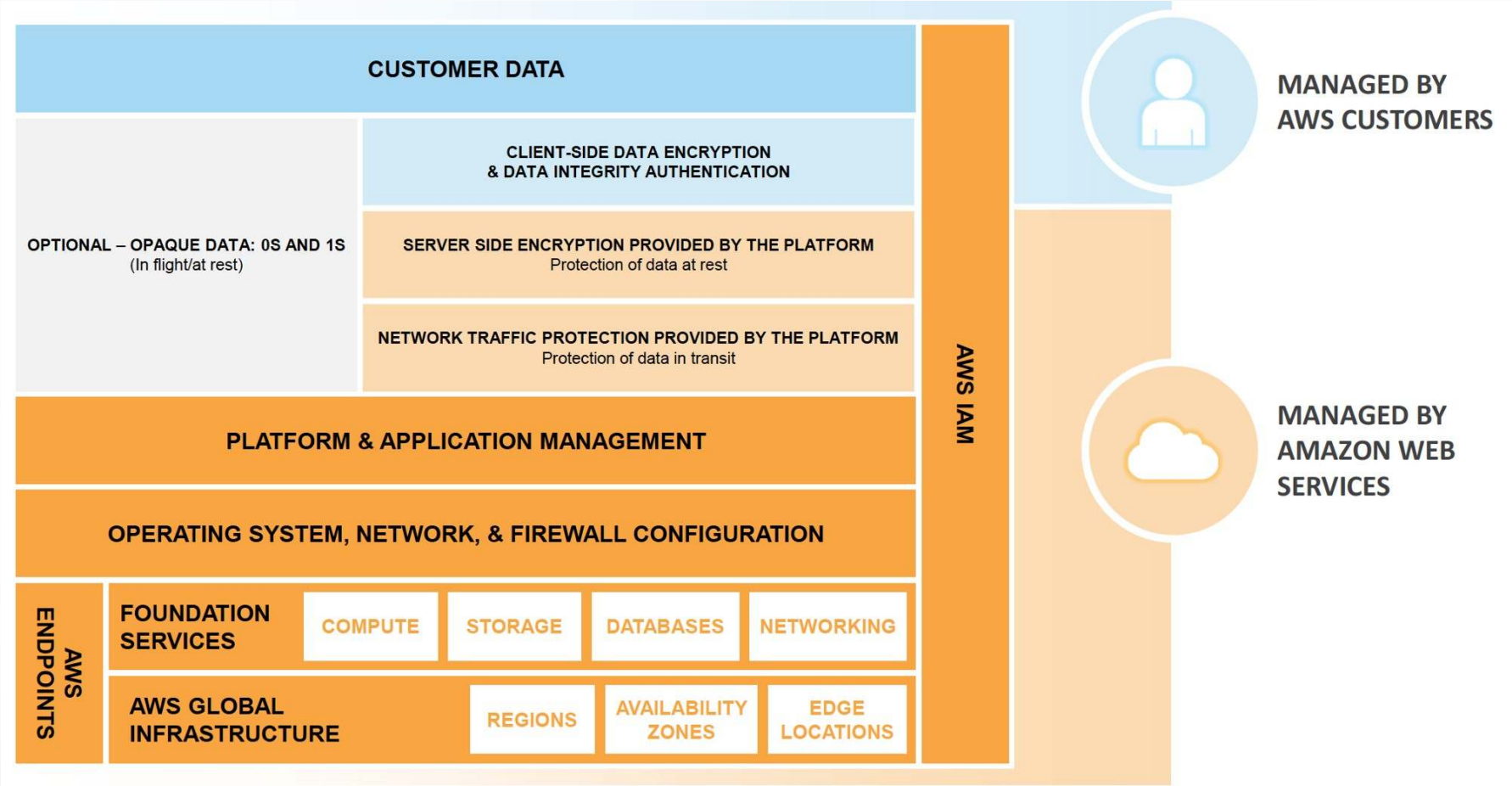
amazon web services

Managed infrastructure

- Preconfigured infrastructure:
  - Single-instance (dev, low cost)
  - Load-balanced, Auto Scaling (production)
- Web and worker tiers
- Elastic Beanstalk provisions necessary infrastructure resources, such as the load balancer, Auto Scaling group, security groups, database (optional), etc.
- Provides a unique domain name for your application (e.g., yourapp.elasticbeanstalk.com)



Managed security



Demo: Ruby app

Create a Git repository with AWS CodeCommit

```
$ aws codecommit create-repository
--repository-name blog --region us-east-1
--repository-description "ElasticBeanstalk
demo"

$ git clone ssh://git-codecommit.us-
east-1.amazonaws.com/v1/repos/blog
```

amazon web services

Create a new Rails application

```
$ rails new blog

$ cd blog

$ git add .

$ git commit -m "Initial version"
```

amazon web services

Create a new Rails application

```
$ rails new blog

$ cd blog

$ git add .

$ git commit -m "Initial version"
```

amazon web services

Add a ‘post’ resource to the application

```
$ rails generate scaffold post title:string body:text

$ bundle exec rake db:migrate

$ git add .

$ git commit -m "Add post resource"

$ rails server

$ open http://localhost:3000/posts
```

amazon web services

Initialize a Ruby application in Elastic Beanstalk

```
$ eb init blog -p Ruby -r eu-west-1

$ git add .gitignore

$ git commit -m "Ignore .elasticbeantalk
directory"
```

amazon web services

Create a ‘blog-dev’ environment

Single instance: no Auto Scaling, no load balancing

Size: t2.micro instance (default value)

```
$ eb create blog-dev
--single
--keyname aws-eb
--envvars SECRET_KEY_BASE=`rake secret`
```

amazon web services

Update a page and redeploy on ‘blog-dev’

```
$ vi app/views/posts/index.html.erb
$ git add app/views/posts/index.html.erb

$ eb use blog-dev
$ eb deploy –staged
$ git commit -m "Add message on post page"
$ eb deploy
```

amazon web services

Create a production branch for the blog

```
$ git branch prod
$ git checkout prod
```

Now we have to modify 3 files to add support for Postgres:

```
Gemfile
config/database.yml
.ebextensions/packages.config
```

Gemfile

```
group :development, :test do
  # Use sqlite3 as the database for Active Record
  gem 'sqlite3’
end

group :production do
    # Use PostgreSQL as the database for Active Record
    gem 'pg', '~> 0.18.1’
end
```

amazon web services

config/database.yml

```
production:
    <<: *default
    adapter: postgresql
    encoding: unicode
    database: <%= ENV['RDS_DB_NAME'] %>
    username: <%= ENV['RDS_USERNAME'] %>
    password: <%= ENV['RDS_PASSWORD'] %>
    host: <%= ENV['RDS_HOSTNAME'] %>
    port: <%= ENV['RDS_PORT'] %>
```

These environment variables will be automatically declared by Elastic Beanstalk when we create an RDS instance

amazon web services

.ebextensions/packages.config

```
packages:
    yum:
      postgresql94-devel: []
```

This directive will install the postgres94-devel package on your instances.
It is required to install the ‘pg’ Ruby gem.

.ebextensions provides lots of options to configure and customize your Elastic BeansItalk applications. The documentation is your friend ☺

https://docs.aws.amazon.com/fr_fr/elasticbeanstalk/latest/dg/ebextensions.html

amazon web services

Commit these changes to the production branch

```
$ git add Gemfile config/database.yml .ebextensions

$ git commit -m "Use Postgres for production”

$ git push
```

Now let’s create a proper production environment :
running in a VPC, auto-scaled, load-balanced,
with larger instances and backed by RDS Postgres.

Ready? ☺

amazon web services

Create a ‘blog-prod’ environment

```
$ aws ec2 describe-subnets
$ export VPC_SUBNETS=subnet-63715206,subnet-cbf5bdbc,subnet-59395b00

$ eb create blog-prod -k aws-eb
--vpc.id=vpc-def884bb --vpc.elbpublic --vpc.publicip
--vpc.elbsubnets $VPC_SUBNETS
--vpc.ec2subnets $VPC_SUBNETS
--vpc.dbsubnets $VPC_SUBNETS
--instance_type m4.large
--database.engine postgres --database.version 9.4.5
--database.instance db.m4.large --database.size 5
--database.username YOUR_USERNAME --database.password YOUR_PASSWORD
--envvars SECRET_KEY_BASE=`rake secret`
```

amazon web services

Accessing other AWS services

```
// Login to one of our app servers

$ eb ssh

// Connect to our RDS instance

[ec2-user] psql -h RDS_ENDPOINT -p 5432 -U user -d ebdb
ebdb=> \dt
ebdb=> select * from posts

// Connect to our ElastiCache cluster

[ec2-user] echo "stats" | nc ELASTICACHE_ENDPOINT 11211
[ec2-user] printf "set mykey 0 60 4 \r\ndata\r\n" | nc ELASTICACHE_ENDPOINT 11211
[ec2-user] echo "get mykey" | nc ELASTICACHE_ENDPOINT 11211
```

amazon web services

More CLI

```
$ eb status
$ eb health
$ eb scale
$ eb logs
$ eb terminate

$ aws elasticbeanstalk …
```

amazon web services

Best practices

Testing and tuning your application

- Pick performance metrics you want to optimize for (e.g., latency, concurrent users, number of web requests, etc.)
- Load test your application:
  - Start with Auto Scaling minimum and maximum of 1 to understand how your application degrades under an overload condition
  - Understand available metrics and how they correspond to your performance metric
- Configure Auto Scaling to optimize for performance metrics:
  - Number of instances to add on scale out
  - Breach duration
  - Metric to scale on
- Tune the back end (DynamoDB, RDS, etc.) for optimal performance; leave enough headroom for full scale out

amazon web services

Deployment option (rolling)

Pros:

- Deployments are fast (20-60 sec.)

Cons:

- Slower rollback because the previous application version must be redeployed
- Possibility of state build-up on long-running environments

Deployment option (blue/green)

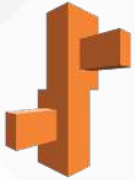
Pros:

- Fast rollback because the environment running the previous version has not been modified in any way
- Ensures no state build up on environments

Cons:

- Deployments take longer than rolling deployments (2-5 min.) because a new environment must be created
- Clients (i.e., mobile devices) might cache DNS addresses and not respect DNS TTL, resulting in staggered/non-deterministic traffic rollover to the new environment

AWS Elastic Beanstalk



- Platform as a Service
- Supports PHP, Java, .NET, Node.js, Python, Go, Ruby, IIS, Glassfish, Tomcat and Docker

- Developer-friendly CLI : '*eb*'
- Built-in monitoring (Amazon Cloudwatch), networking (Amazon VPC), load balancing (Amazon ELB) and scaling (Auto Scaling)
- Relational data tier is available through Amazon Relational Data Service (RDS)
- Can also integrate with many AWS services

The simplest and most intuitive way to deploy your applications

This should really be your default option for deployment

https://aws.amazon.com/fr/elasticbeanstalk/
https://aws.amazon.com/fr/blogs/aws/category/aws-elastic-beanstalk/https://aws.amazon.com/releasenotes/AWS-Elastic-Beanstalk

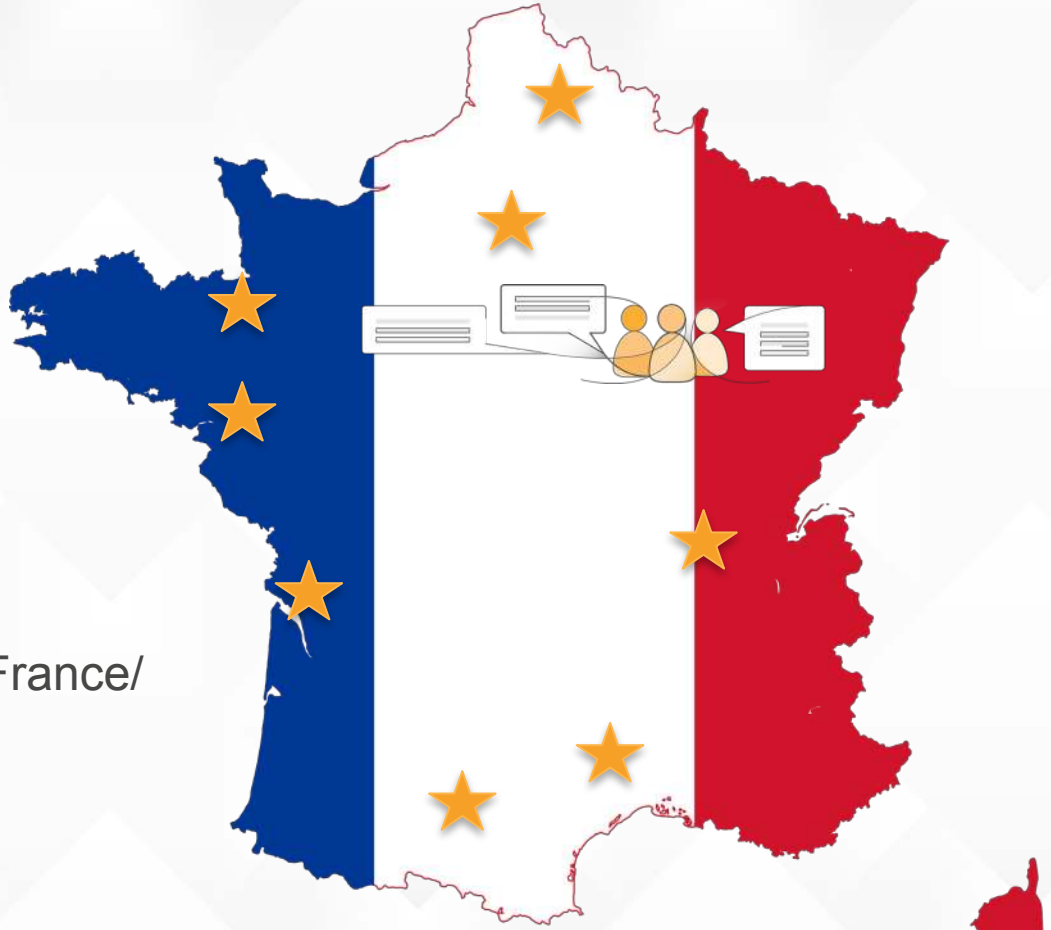
amazon web services

AWS User Groups

Lille
Paris
Rennes
Nantes
Bordeaux
Lyon
Montpellier
Toulouse

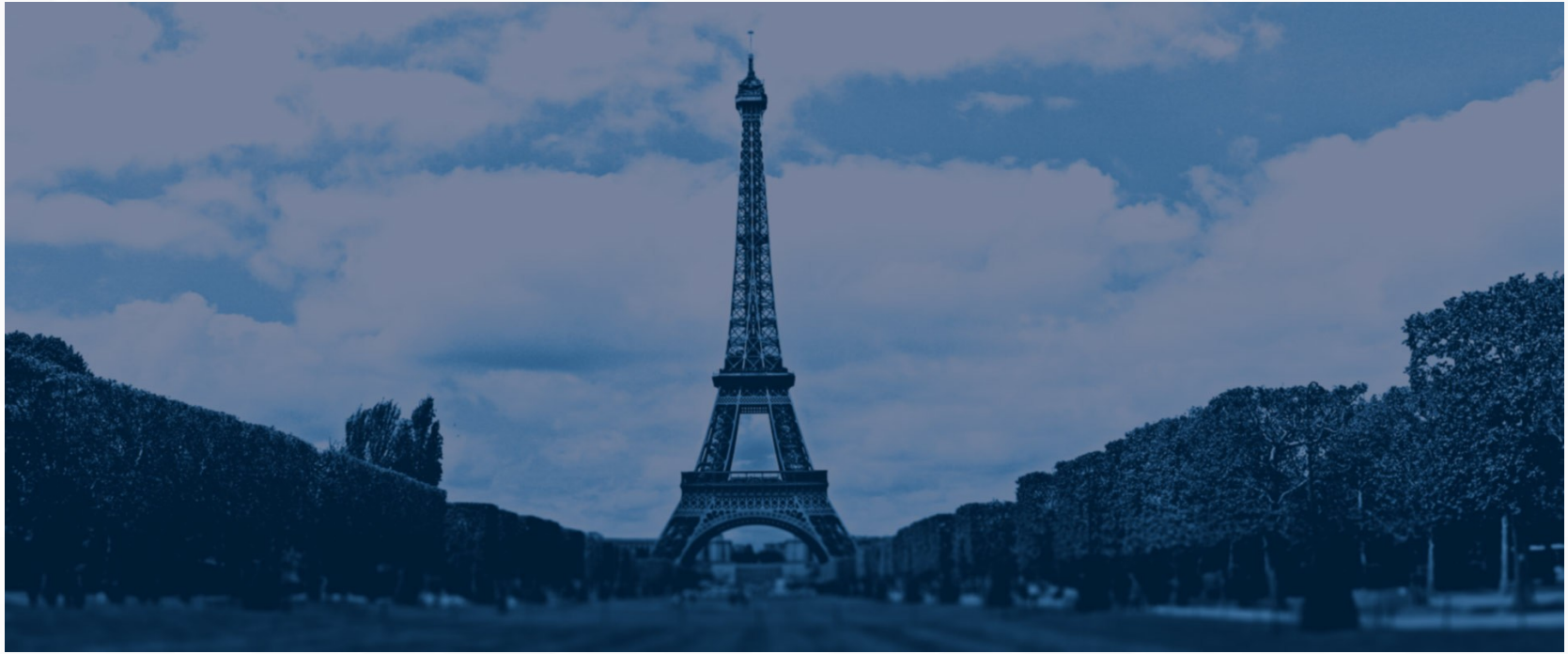
facebook.com/groups/AWSFrance/

@aws_actus



amazon web services

AWS Enterprise Summit – 27/10/2016, Paris



http://amzn.to/1X2yp0i

amazon web services

Thank you!

Julien Simon
Principal Technical Evangelist
julsimon@amazon.fr
@julsimon

amazon web services