Django & AWS

Evolution of an Architecture

Alan Greenblatt
CTO, PaymentWorks
www.paymentworks.com
blatt@paymentworks.com
@agreenblatt

PaymentWorks

PaymentWorks Overview

ERP-agnostic Supplier Portal that streamlines the invoicing & payment process by enabling large and mid-sized companies to exchange information securely with suppliers

- Have you ever mailed off an invoice and waited for 30-45 days to get paid, hoping everything goes according to plan and you get paid on time?
- Can you imagine what it's like for a company with 10,000 suppliers, a good deal of whom are calling on a weekly basis about the status of invoices?
- Wouldn't it be nice if you could just "friend" your customers and be able to see the status of all your invoices, message the AP department directly when there are problems, and have the option of accepting a small discount and getting paid early on invoices?
- ★Think LinkedIn - but instead of connecting people, we connect accounts payables with accounts receivables

PaymentWorks

Technical Overview

- Backend, public site and FTP file transfer all Django
- REST interface served up by Django Rest Framework
- Subscribers (payers & payees) use fairly large, complex single page application

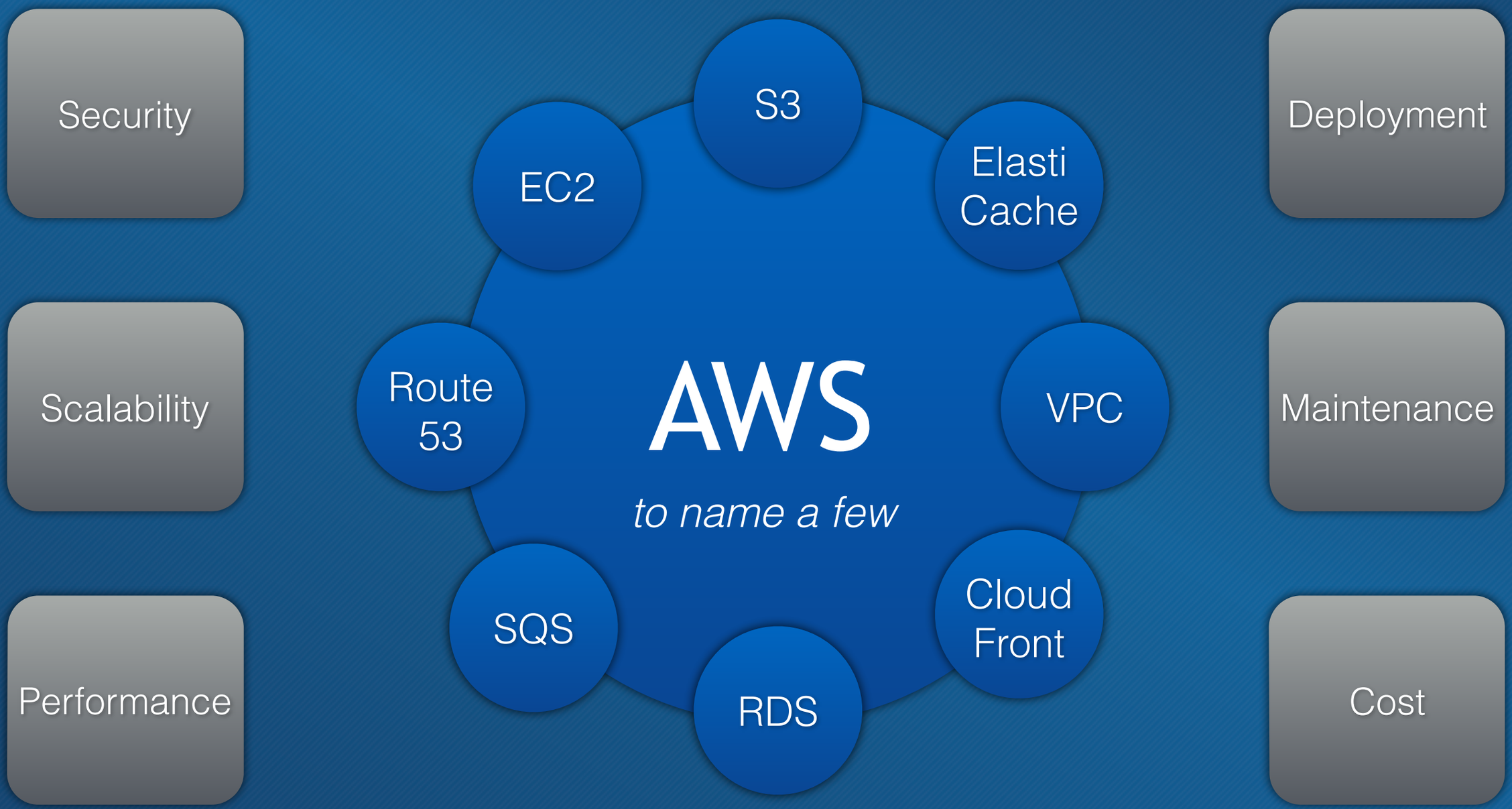
  ➡ Backbone.js/Marionette.js/Require.js
- All hosted on AWS

PaymentWorks

How would you build all this in the cloud?

PaymentWorks

It can be overwhelming at first…



PaymentWorks

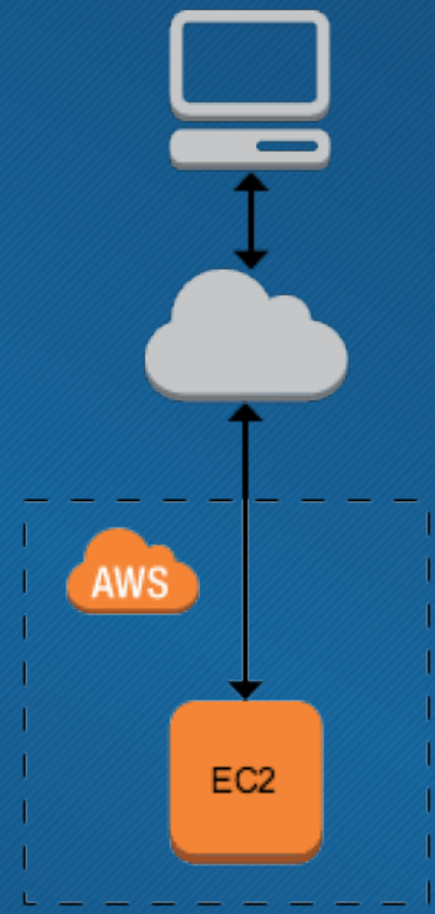
The Polls Tutorial

PaymentWorks

Start in the beginning

EC2 - Elastic Compute Cloud

And EC2 instance is essentially a virtual server



Looks simple!

PaymentWorks

1. Navigate to AWS console (console.aws.amazon.com)

2. Launch new EC2 instance… **BZZZT!**

Right off the bat, you are confronted with new terms, forms to fill out, choices to make, and confusing configuration settings

PaymentWorks

Before we do anything

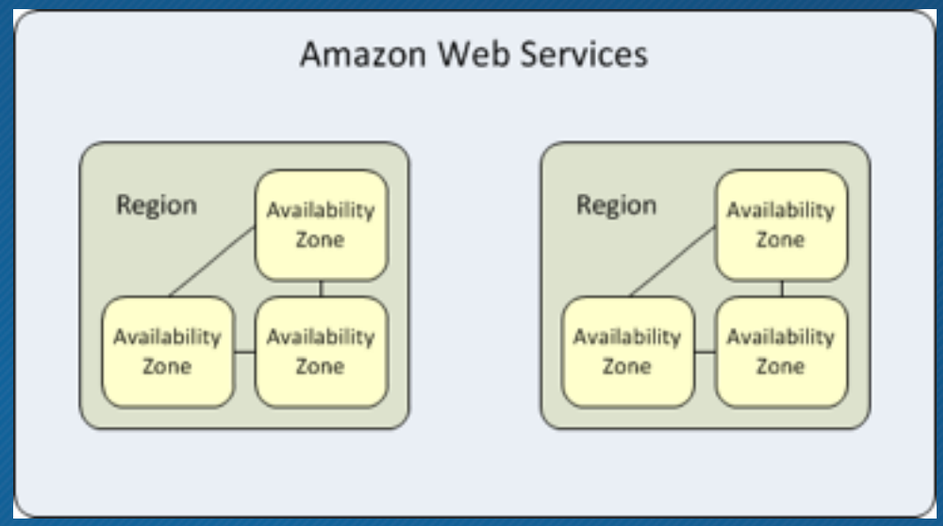
we need to understand a few concepts…

- Regions & Availability Zones
- VPC
- CIDR
- Subnet
- Key Pairs

PaymentWorks

Regions & Availability Zones

- Resources are either global or tied to a region or availability zone.
- Each region is completely isolated from other regions
  - Currently there are 9 distinct global regions, 3 alone in the US. We would use US East (N Virginia).
- Availability zones lie within regions. They are isolated, but are connected through low-latency links. Currently, there are 5 availability zones in the US East region.



PaymentWorks

VPC - Virtual Private Cloud

Allows you to create complex secure networks in the cloud.

You have a default VPC, but can create multiple VPCs, and can even communicate with local IP addresses between VPCs.

Servers, subnets, load balancers, etc are all created in a VPC.

A VPC can span multiple availability zones within a region

You can even VPN into a VPC

CIDR
Classless Inter-Domain Routing

Compact notation for representing an IP address and its associated routing prefix.

Consists of an IP address and a prefix size; the number of leading 1 bits in the routing mask.

E.g. 192.168.100.0/24

The first 24 bits specify the mask. Since there are 8 bits remaining, this specifies a range of up to 256 IP addresses, all with the IP prefix of 192.168.100.x

Subnet

A logical division of an IP network

Useful for grouping a set of IP resources to which you can assign permissions, routing rules, etc.

CIDRs are used to specify the addressable range of a subnet. E.g. 192.168.0.0/16 would specify a range of 65536 possible IP addresses.

Multiple web servers within an availability zone could be placed for instance, in a private (not Internet accessible) subnet.

A load balancer handling access to those web servers, would be placed in a separate public subnet.

Public/Private Key Pairs

Allows for secure communications to/from AWS

- Navigate to EC2 dashboard -> Key Pairs
- Create a Key Pair
- Save downloaded PEM (private key) in a hidden place. Don’t check it into your Git repo!

https://console.aws.amazon.com

Create New VPC

Make it a nice big VPC, allowing for 65K potential IP addresses!

CIDR: 10.0.0.0/16

All IP addresses within this VPC will begin with 10.0.X.X

https://console.aws.amazon.com/vpc

PaymentWorks

Create a public subnet within the VPC

- Specify a name
- Use the previously created VPC
- No preference of availability zone
- CIDR block: 10.0.0.0/24 (256 potential IP addresses)

https://console.aws.amazon.com/vpc

Create EC2 Instance (cont'd)

- Navigate to VPC or EC2 Dashboard
- Launch EC2 Instance
- Pick appropriate image (e.g. free Amazon AMI)
- Minimum storage (8Gb for small)
- Give it a name tag
- Set security group's ssh access from your "My IP"
- Launch
- Choose your previously created keypair
- Launch Instance

PaymentWorks

Internet Gateway

Scaled, redundant and highly available VPC component that allows for communications between VPC resources and the Internet.

- Create an Internet Gateway and attach it to your VPC

https://console.aws.amazon.com/vpc

Elastic IP

Allows you to create public IP addresses and then separately attach them to internal fixed IP address.

Provides flexibility to reassign IP addresses.

- Create an elastic IP address in this VPC and attach it to your newly running EC2 instance. This will be the public IP address of this server.

https://console.aws.amazon.com/vpc

Route Table

Create a new route table in this VPC

Edit the subnet associations and associate the newly created subnet with this route table

Edit the routes and add a route with:

Destination: 0.0.0.0/0
Target: Select your Internet Gateway

Destination	Target	Status	Propagated
10.0.0.0/16	local	Active	No
0.0.0.0/0	igw-f7244392	Active	No

All requests outside of this subnet will go out to the Internet through the gateway

https://console.aws.amazon.com/vpc

SSH into your server!

Public IP address is your Elastic IP

> ssh -i <path to pem file> ec2-user@<ip address>

or specify in ~/.ssh/config:

```
Host <shortcut-server-name>
  HostName <public ip address>
  IdentityFile <absolute path to pem file>
  User ec2-user
```

> ssh <shortcut-server-name>

PaymentWorks

Install Git

> sudo yum install git

Make sure you can read from your remote repo

Make a folder where you are going to install your app

```
> git init
> git remote add origin …
> git pull
```


PaymentWorks

Install Requirements

From your app installation folder:

```
> sudo pip install -r requirements.txt
```


PaymentWorks

Install Apache

An exercise for the reader…

```
> sudo yum install httpd24
```

Unfortunately, mod_wsgi for Amazon AMI is built for python 2.6. You're going to need to build mod_wsgi for python2.7, so:

```
> sudo yum install gcc, python27-devel
> curl -o mod_wsgi-3.4.tar.gz https://modwsgi.googlecode.com/files/mod_wsgi-3.4.tar.gz
> tar xvzf mod_wsgi-3.4.tar.gz
> cd mod_wsgi-3.4
> ./configure --with-python=/path/to/your/python27
> sudo make install
```

* https://forums.aws.amazon.com/message.jspa?messageID=453942

Add new file /etc/httpd/conf.modules.d/02-wsgi.conf which reads:

```
LoadModule wsgi_module modules/mod_wsgi.so
```

Configure Apache to point to wsgi in your app.

```
WSGIApplicationGroup %{GLOBAL}
WSGIDaemonProcess my_process_grp python-path=<path-to-django-app> processes=25
threads=15 display-name=%{GROUP}
WSGIProcessGroup my_process_grp
WSGIScriptAlias / <path-to-django-app>/mysite/wsgi_prod.py
WSGIPassAuthorization On
```


PaymentWorks

Run Django Webserver

Run the server on port 80, accessible from any IP address

> python manage.py runserver 0.0.0.0:80

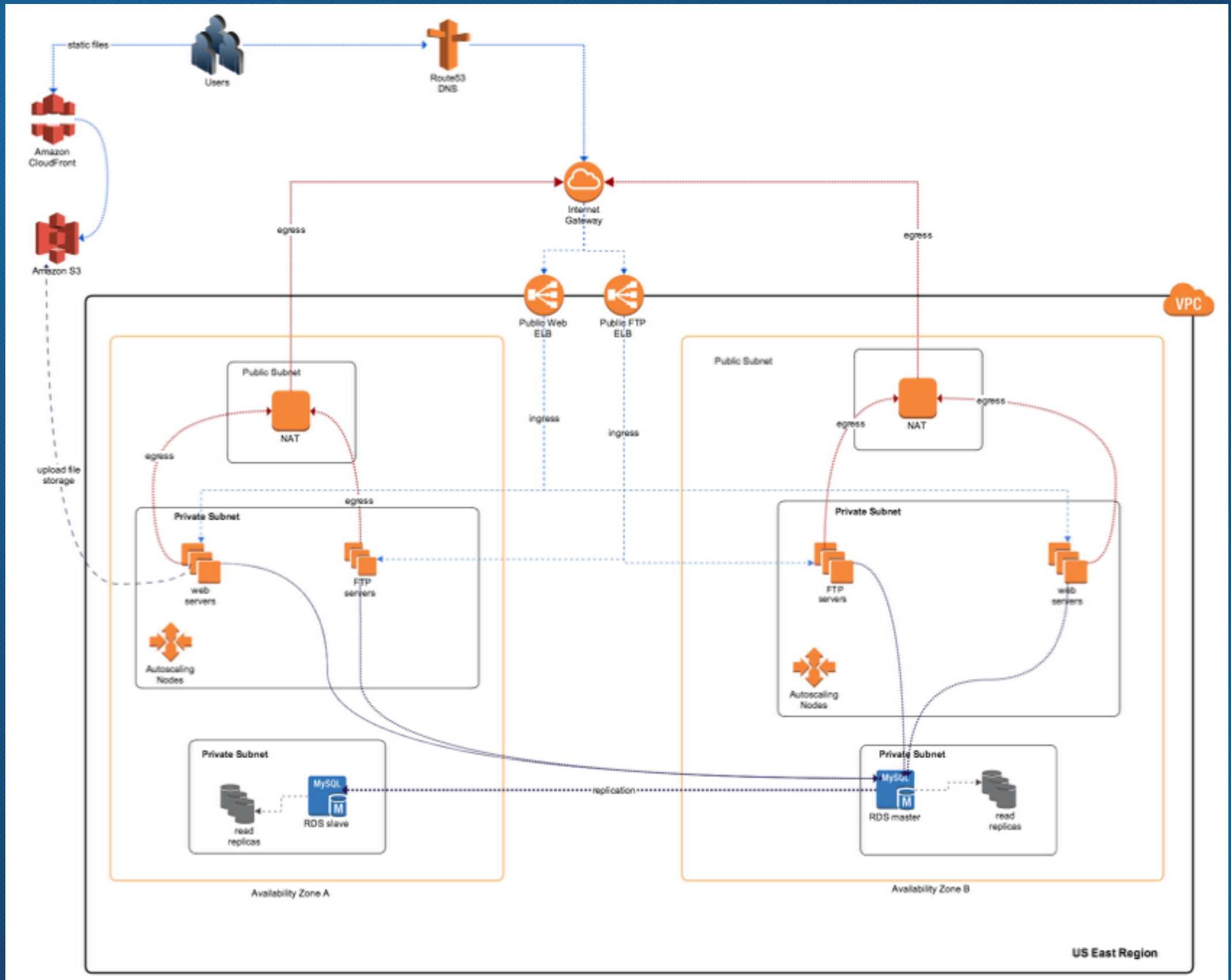
Access the App from a Browser!

Navigate to http://<elastic ip>/polls

PaymentWorks

But what about scalability? performance? availability?

PaymentWorks


static files
Users
Route53 DNS
Amazon CloudFront
Amazon S3
upload file storage
Internet Gateway
egress
egress
Public Web ELB
Public FTP ELB
VPC
Public Subnet
NAT
egress
ingress
ingress
egress
Public Subnet
NAT
egress
egress
Private Subnet
web servers
FTP servers
Autoscaling Nodes
Private Subnet
FTP servers
web servers
Autoscaling Nodes
Private Subnet
MySQL
M
RDS slave
read replicas
replication
Private Subnet
MySQL
M
RDS master
read replicas
Availability Zone A
Availability Zone B
US East Region

Hey, you snuck in a bunch of new terms!

- Route 53
- NAT
- Autoscaling
- RDS
- S3
- CloudFront

PaymentWorks

Route 53

Your own private DNS router, with efficient hooks into AWS resources.

PaymentWorks

NAT

Network Address Translation

Outbound requests all go through the NAT before going out on the wild, wild Internet.

Internal IP addresses never get exposed.

Provides a nice way to ssh hop to your web servers once private subnets are really locked down.

PaymentWorks

Autoscaling

You define a launch configuration, and scaling parameters.

Launch configuration includes an AMI image to spin up as necessary.

Scaling parameters define min/max #instances and when to scale up/down

You can define a startup script which pulls the code from the git repo when the instance spins up.

RDS

Relational Data Store

Managed relational database

- MySQL
- Oracle
- SQL Server
- PostreSQL
- Aurora (in preview)

PaymentWorks

CloudFront

Amazon CDN (content delivery network)

Caches resources geographically closer to where they are being requested from.

I’m sure the folks at Akamai hate it.

PaymentWorks

S3

Simple Storage Service

Basically, a key-value store.

Console shows a hierarchical view, but really there is no hierarchy of keys.

Just buckets and key/value pairs in those buckets.

Great way to serve static content and reduce the load on your web servers.

Boto is Your Friend

Python interface to AWS

Code custom commands to be used in Fabric

Collectstatic directly to S3!

PaymentWorks

Thank you!

Alan Greenblatt
blatt@paymentworks.com
@agreenblatt

PaymentWorks