Advanced Approaches to Amazon VPC and Amazon Route 53

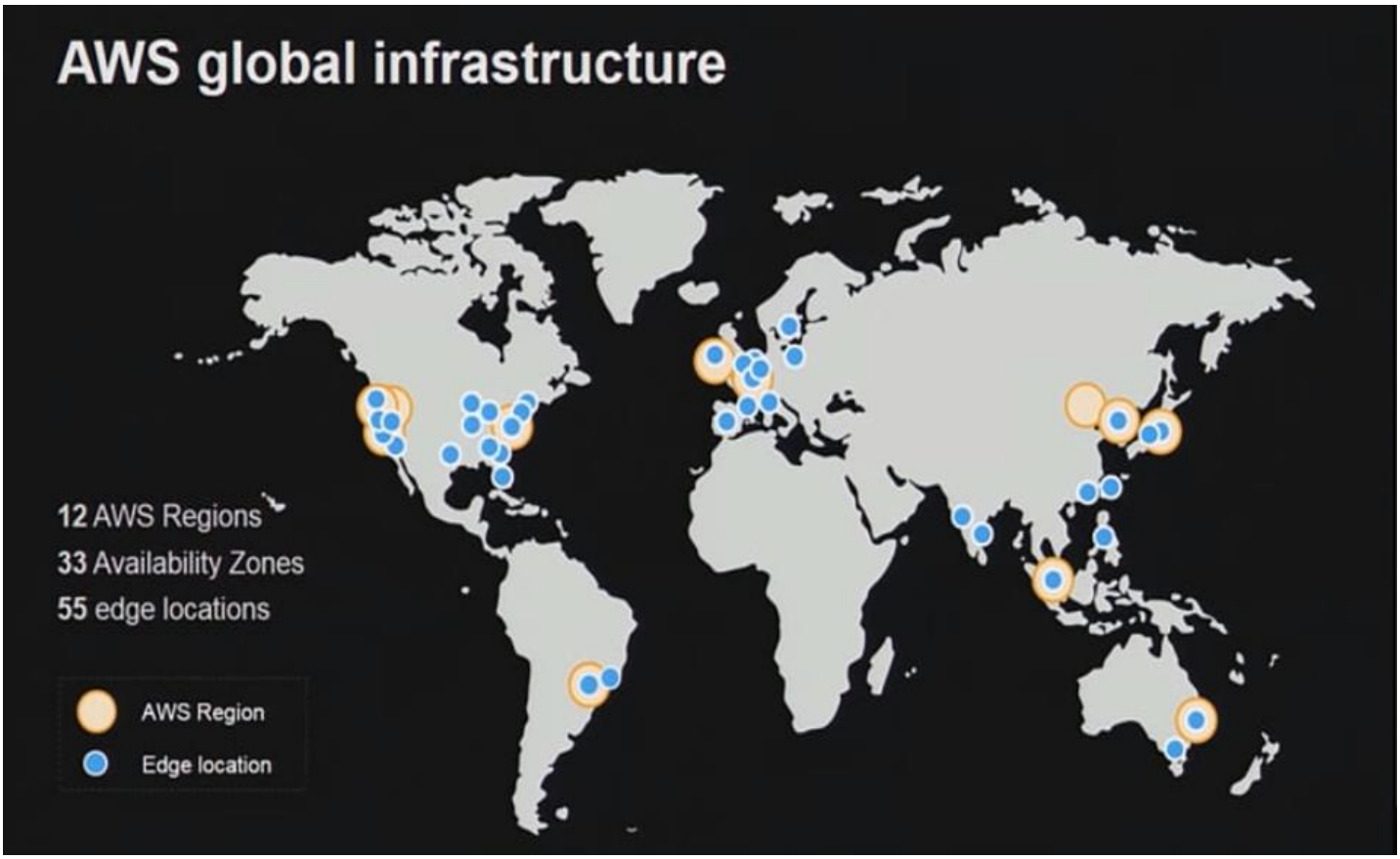
Mike Kuentz, Solutions Architect

June 21, 2016

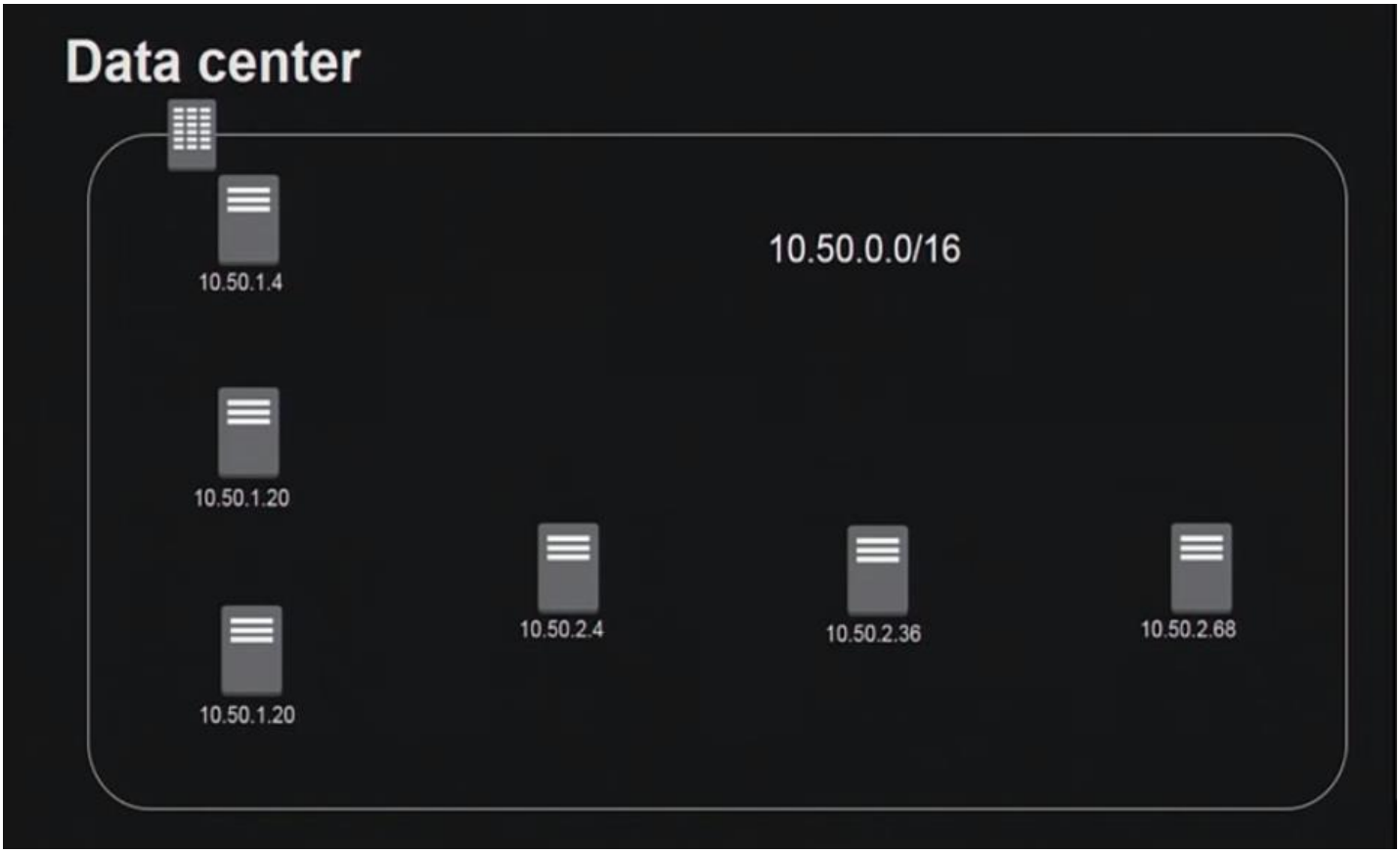
Agenda

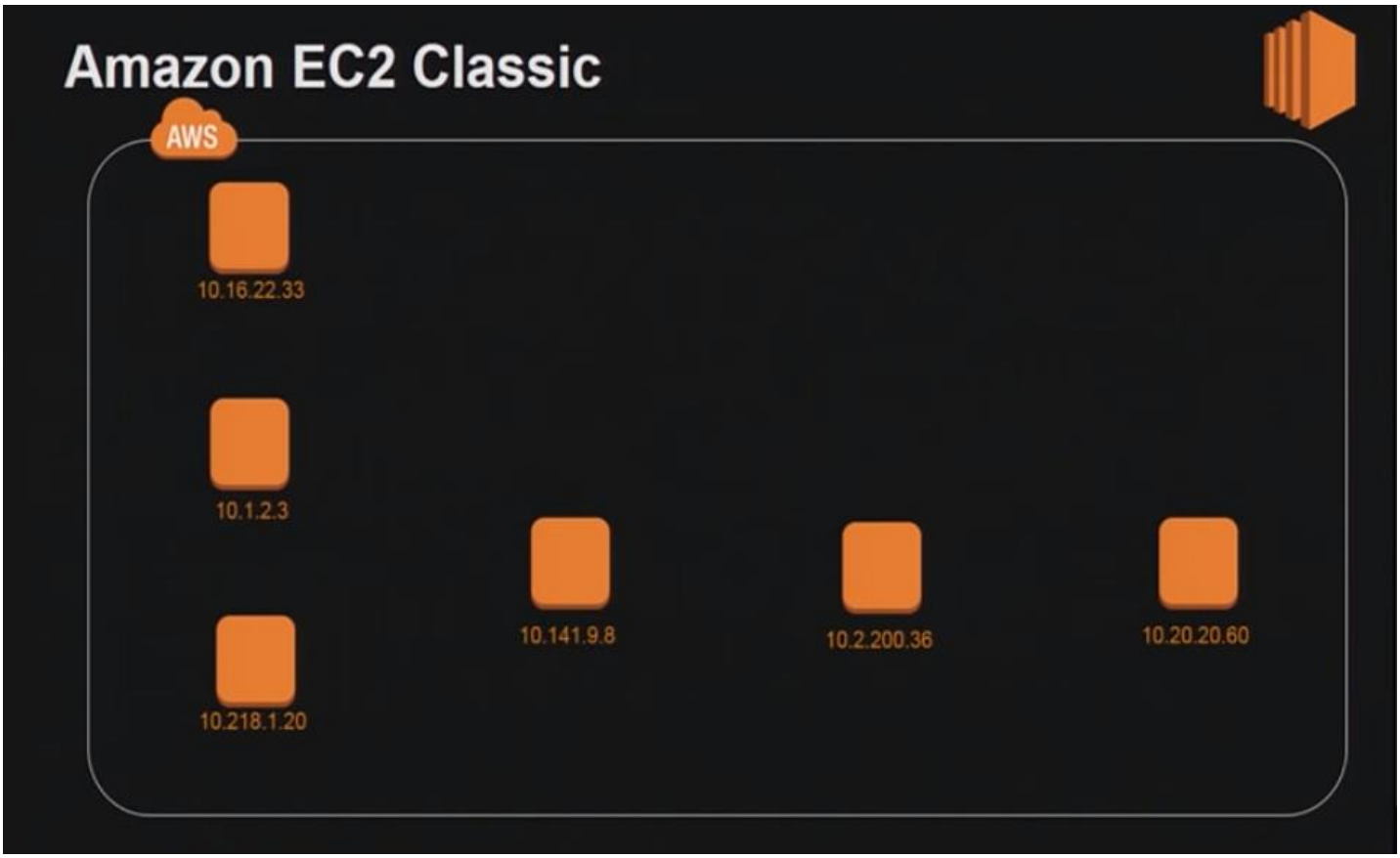
- Amazon VPC concepts
- Basic VPC setup
- Environments with multiple VPCs
- Amazon Route 53 concepts
- Basic Route 53 setup
- Using VPC and Route 53 together

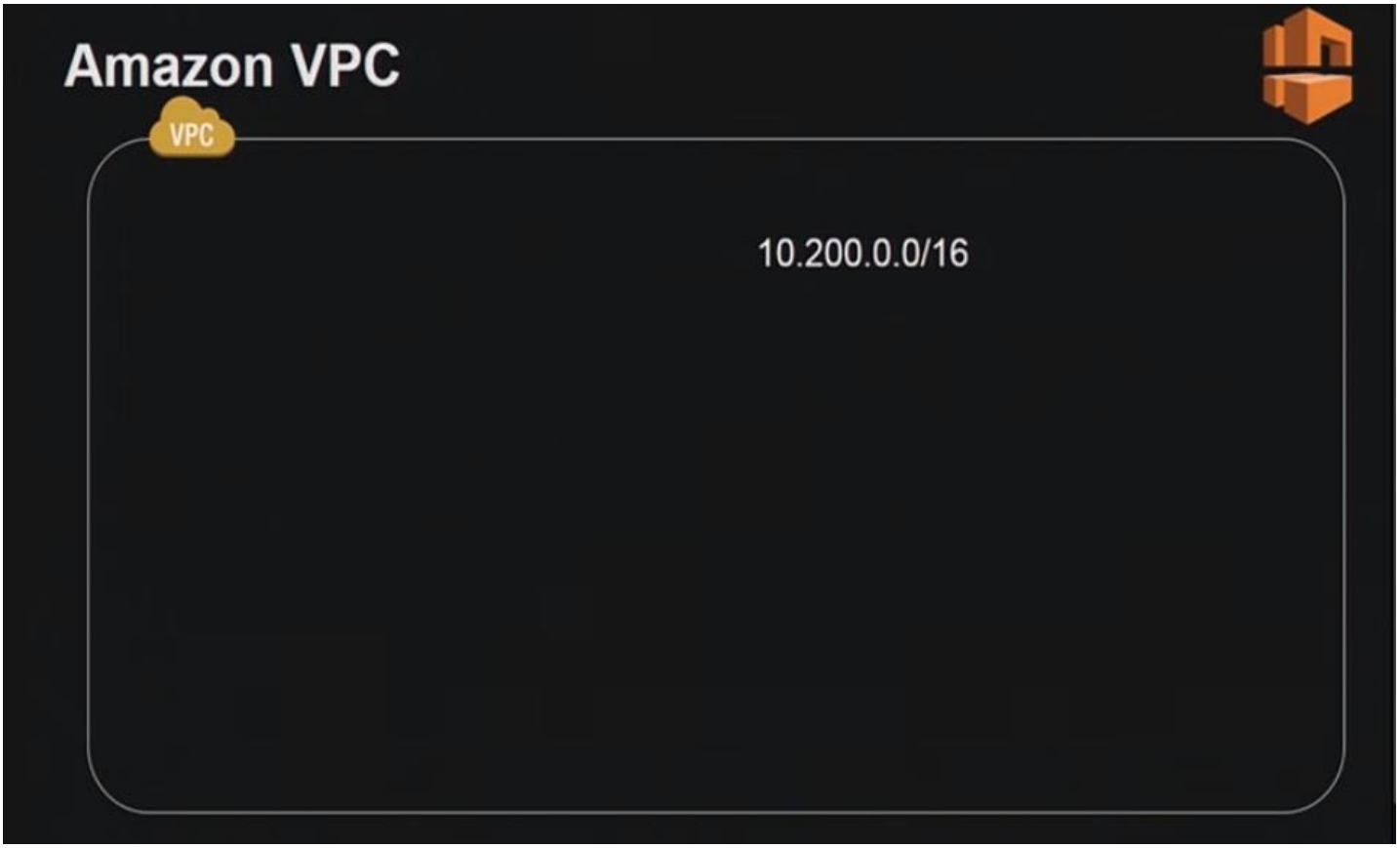
Global infrastructure


AWS global infrastructure
12 AWS Regions
33 Availability Zones
55 edge locations
AWS Region
Edge location

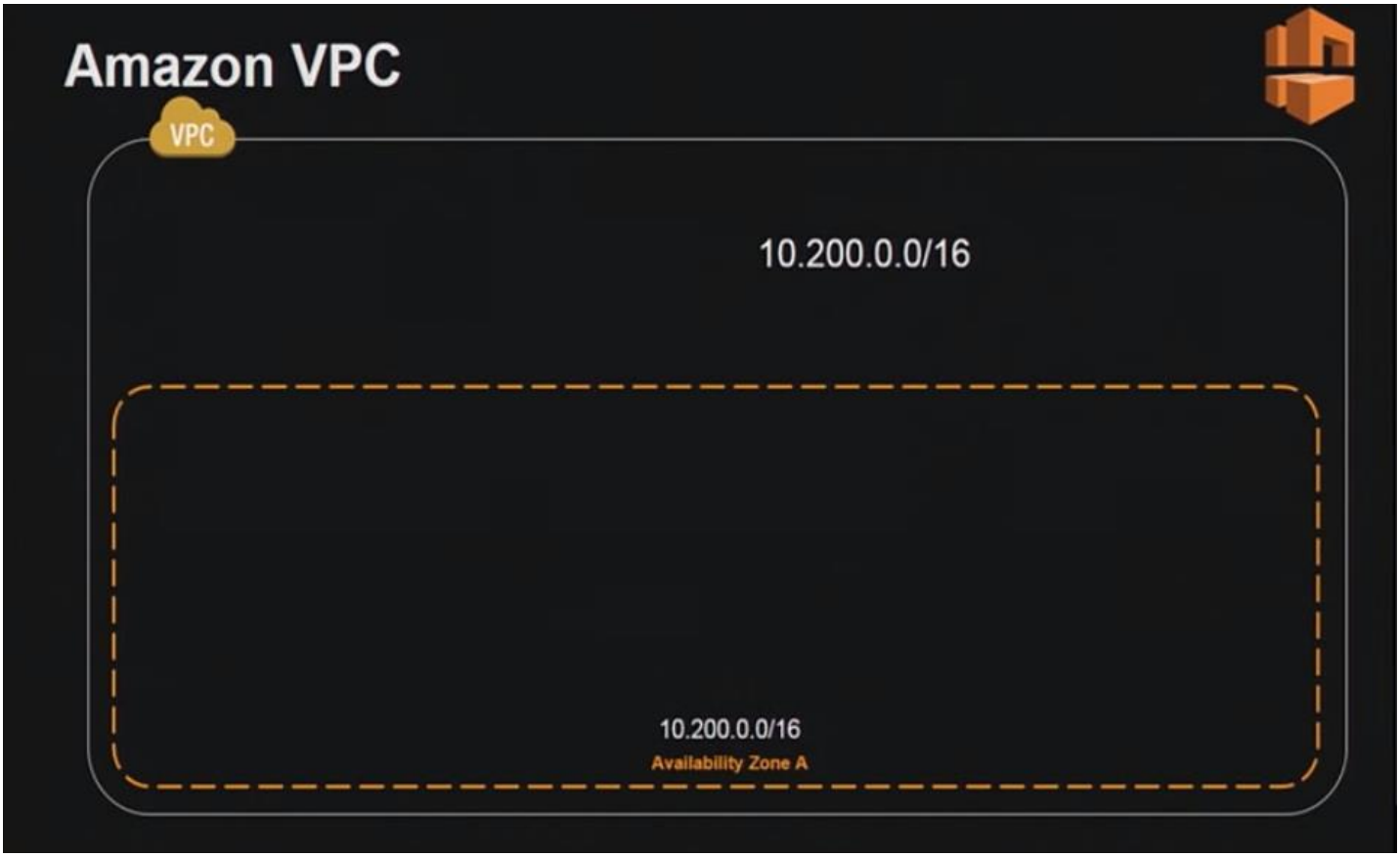

VPC

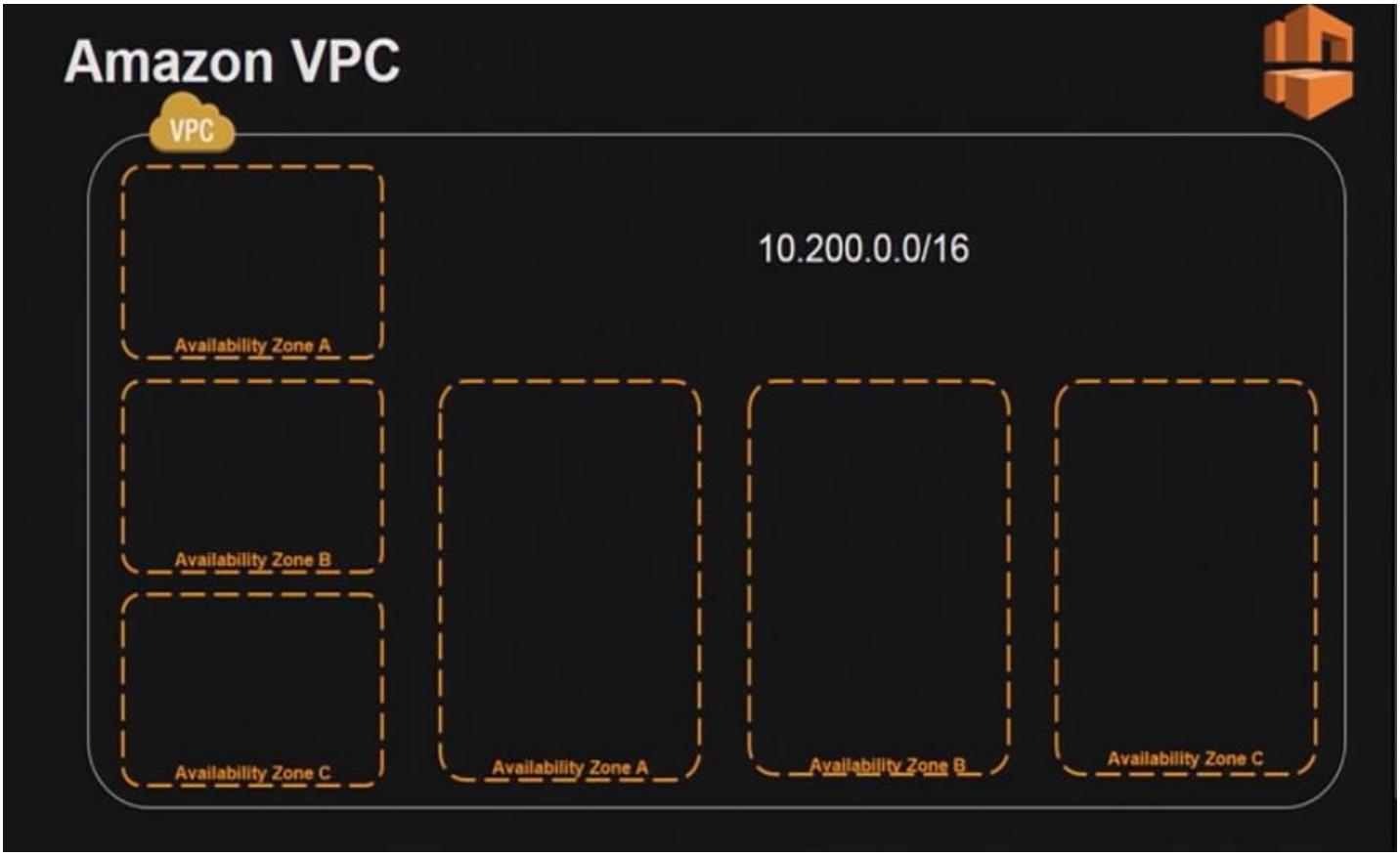

Data center
10.50.0.0/16
10.50.1.4
10.50.1.20
10.50.1.20
10.50.2.4
10.50.2.36
10.50.2.68


Amazon EC2 Classic
AWS
10.16.22.33
10.1.2.3
10.218.1.20
10.141.9.8
10.2.200.36
10.20.20.60

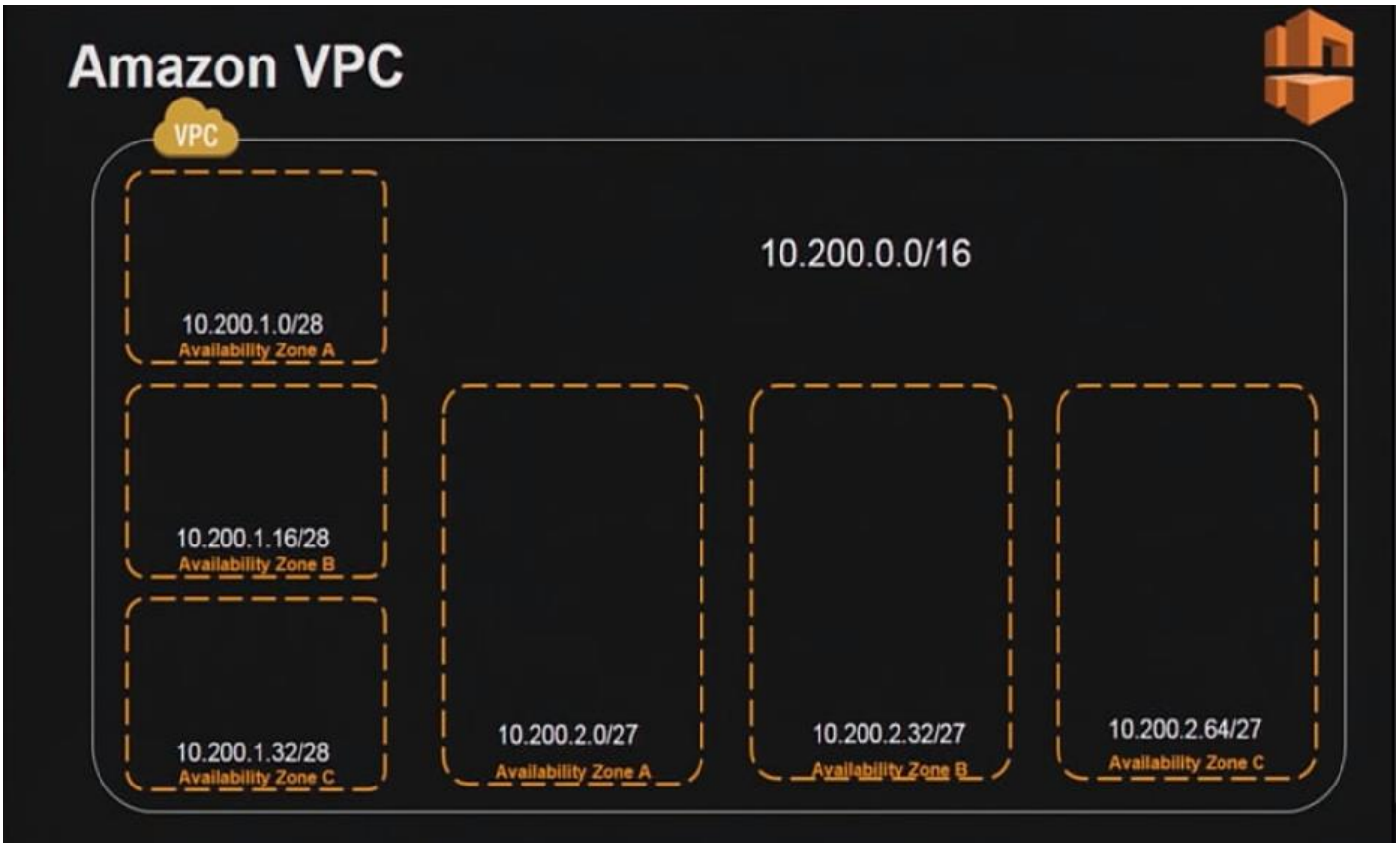


You get a default VPC automatically when you create an AWS account

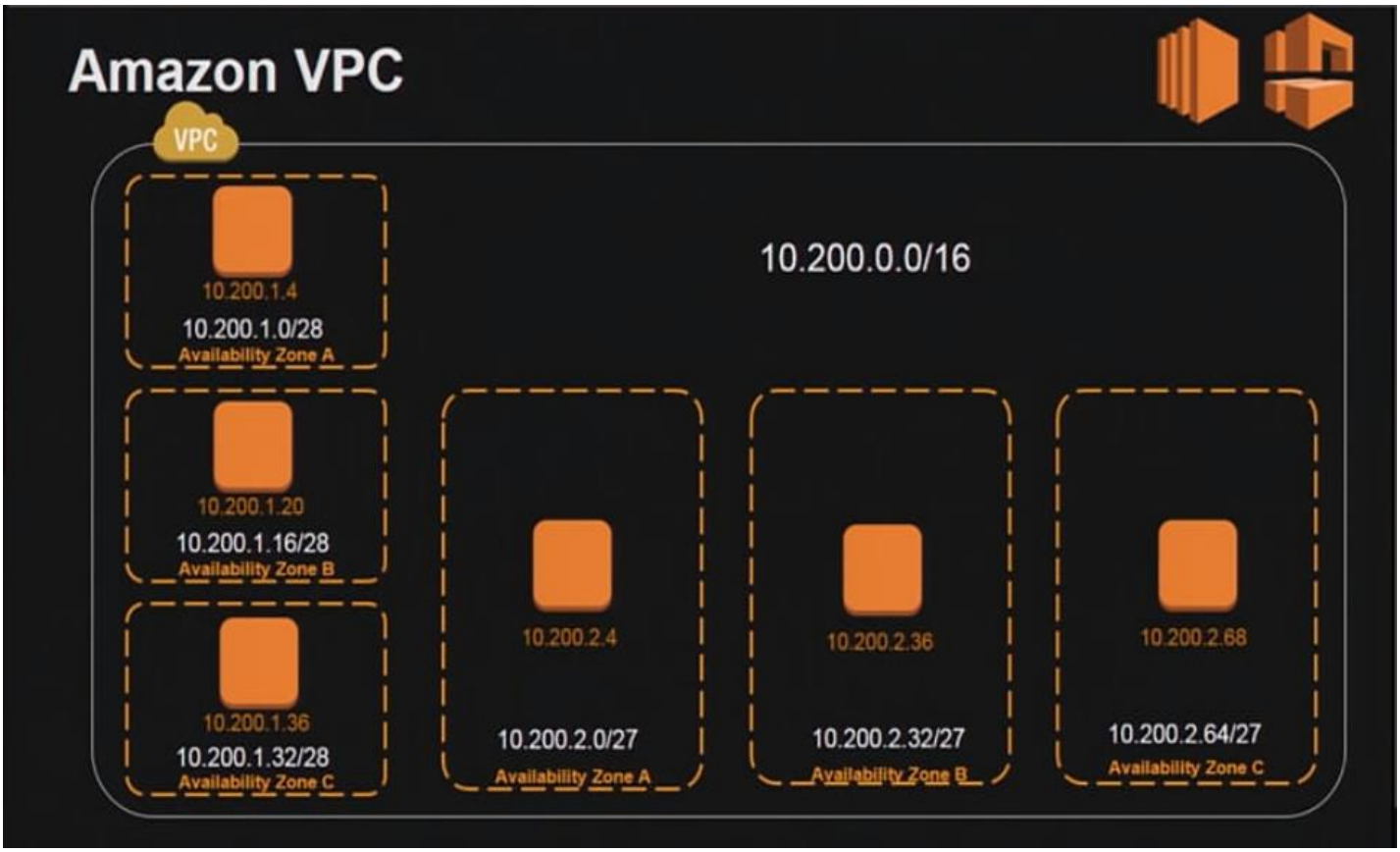


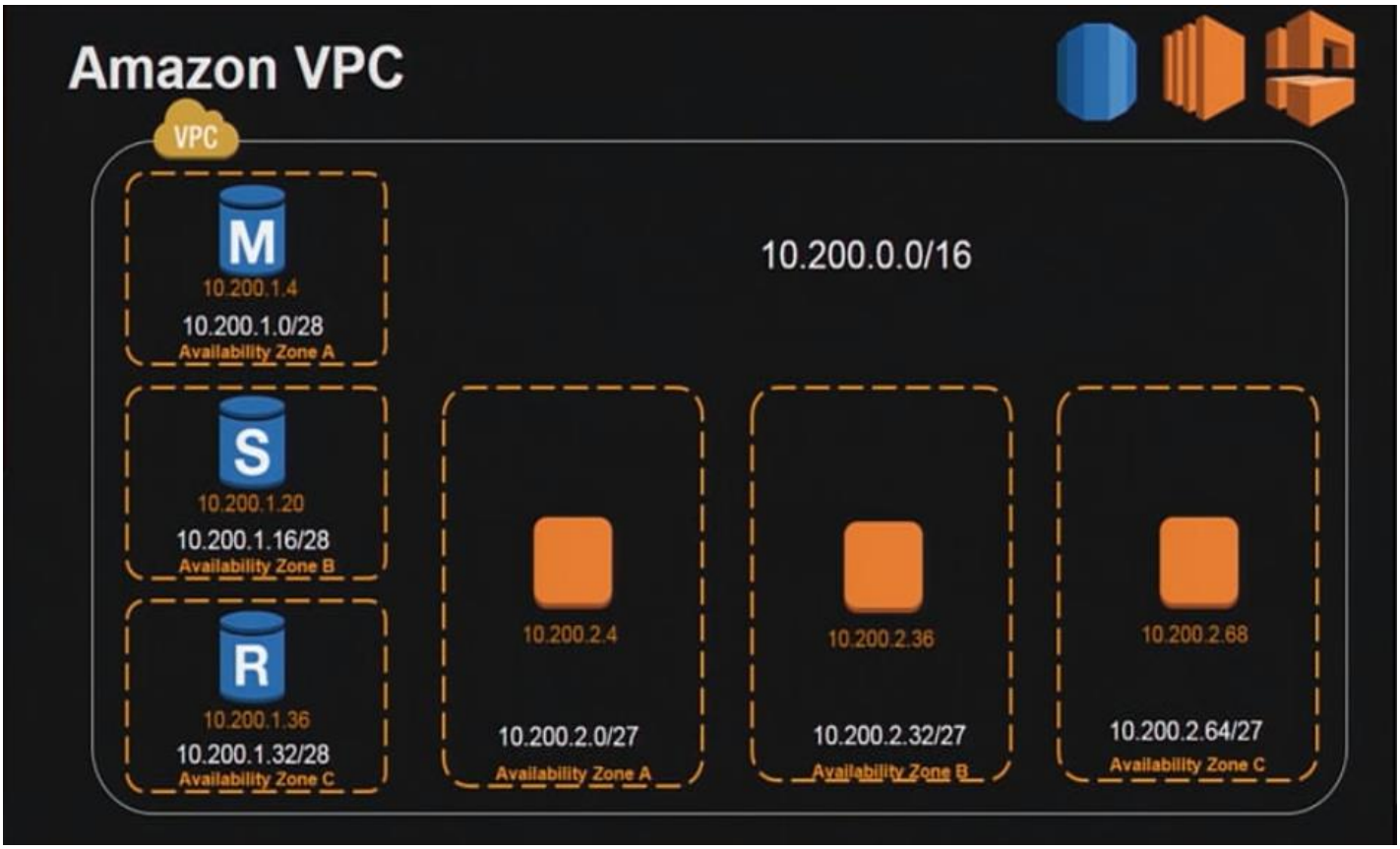


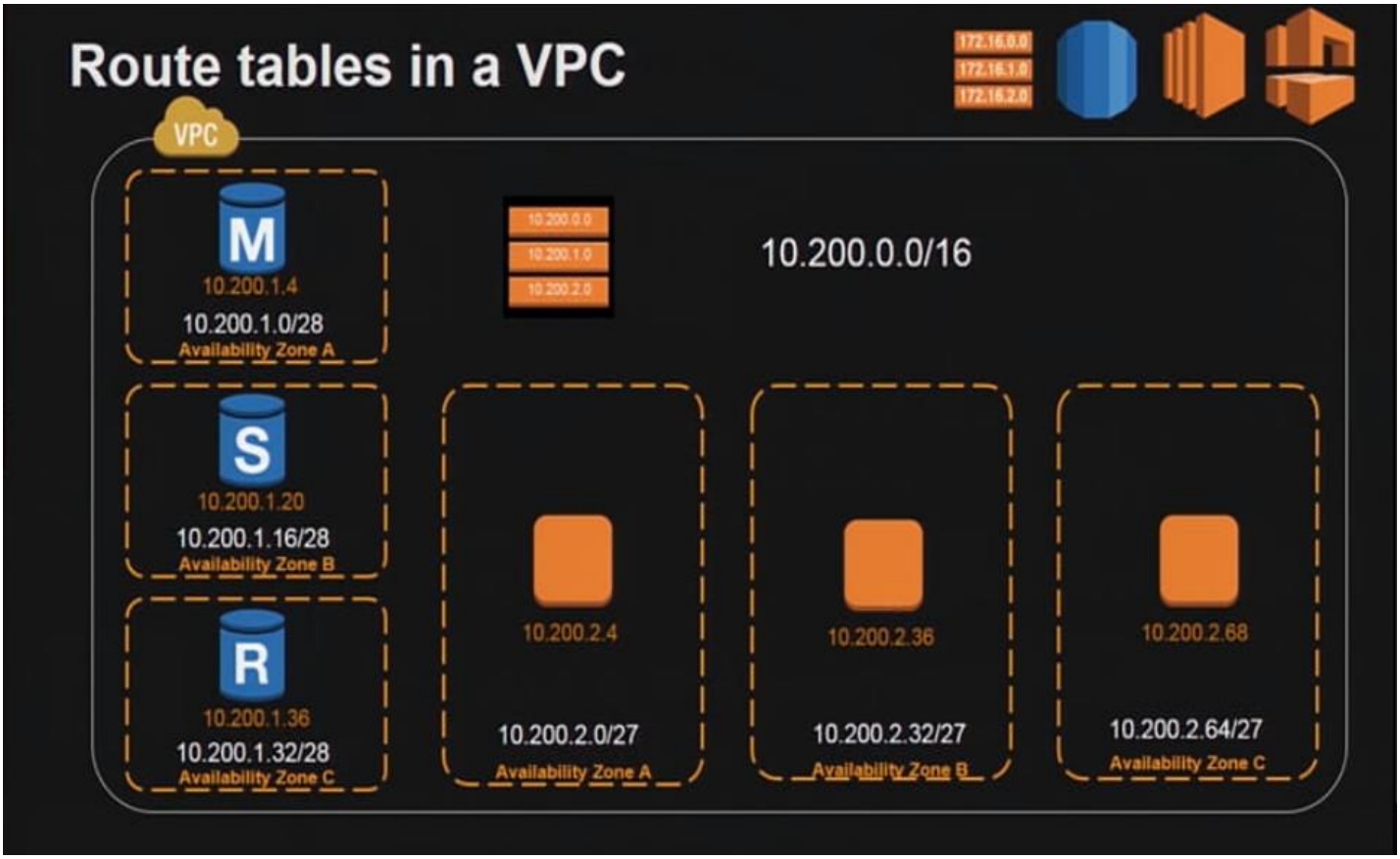
You should actually be creating multiple subnets for multiple AZs and redundancy as above



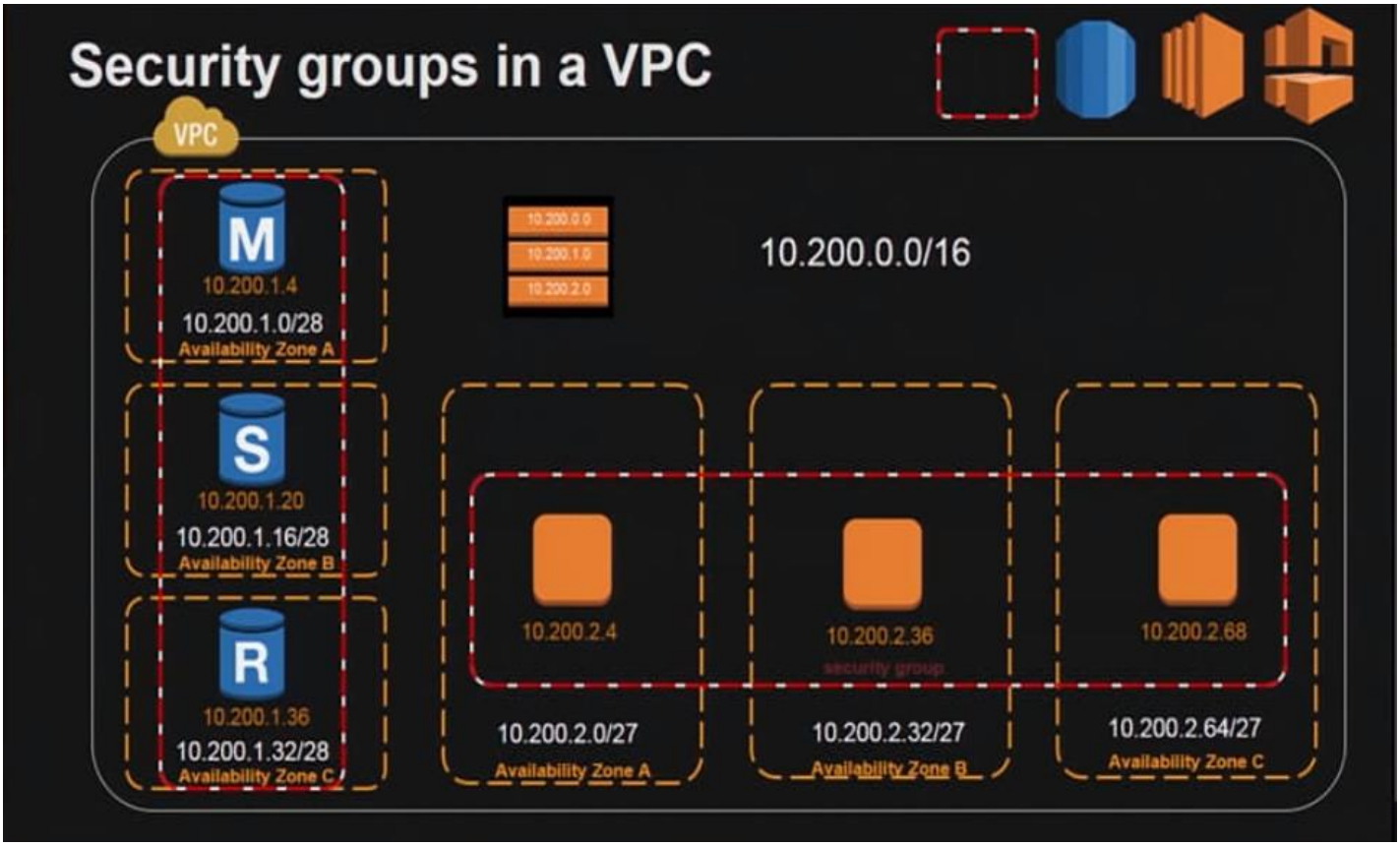
Subnets map to AZs as above, note that AWS reserves 5 addresses between each subnet for DNS, and other uses.


Amazon VPC
VPC
10.200.0.0/16
10.200.1.4
10.200.1.0/28
Availability Zone A
10.200.1.20
10.200.1.16/28
Availability Zone B
10.200.1.36
10.200.1.32/28
Availability Zone C
10.200.2.4
10.200.2.0/27
Availability Zone A
10.200.2.36
10.200.2.32/27
Availability Zone B
10.200.2.68
10.200.2.64/27
Availability Zone C

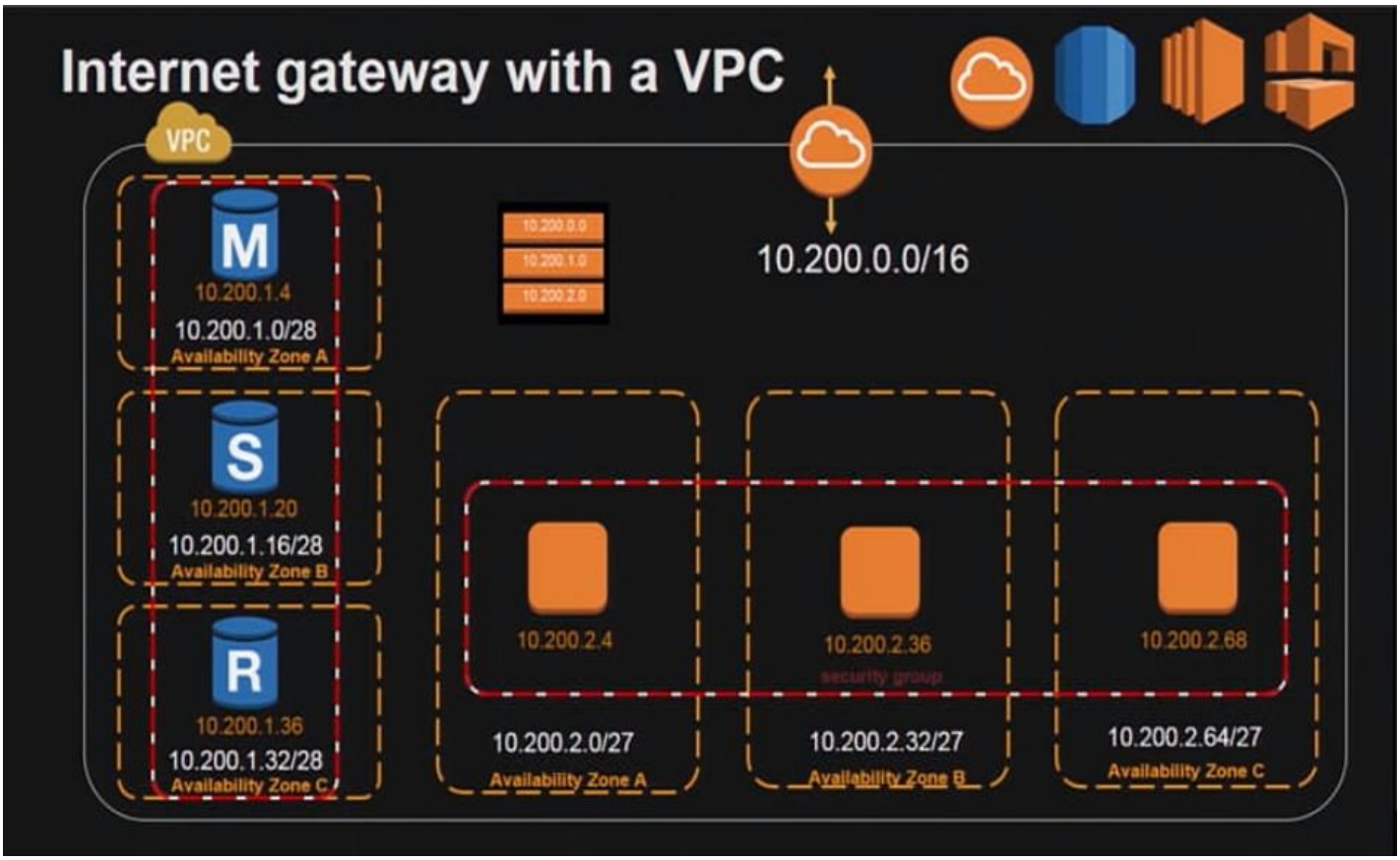

Amazon VPC
VPC
10.200.0.0/16
M
10.200.1.4
10.200.1.0/28
Availability Zone A
S
10.200.1.20
10.200.1.16/28
Availability Zone B
R
10.200.1.36
10.200.1.32/28
Availability Zone C
10.200.2.4
10.200.2.0/27
Availability Zone A
10.200.2.36
10.200.2.32/27
Availability Zone B
10.200.2.68
10.200.2.64/27
Availability Zone C



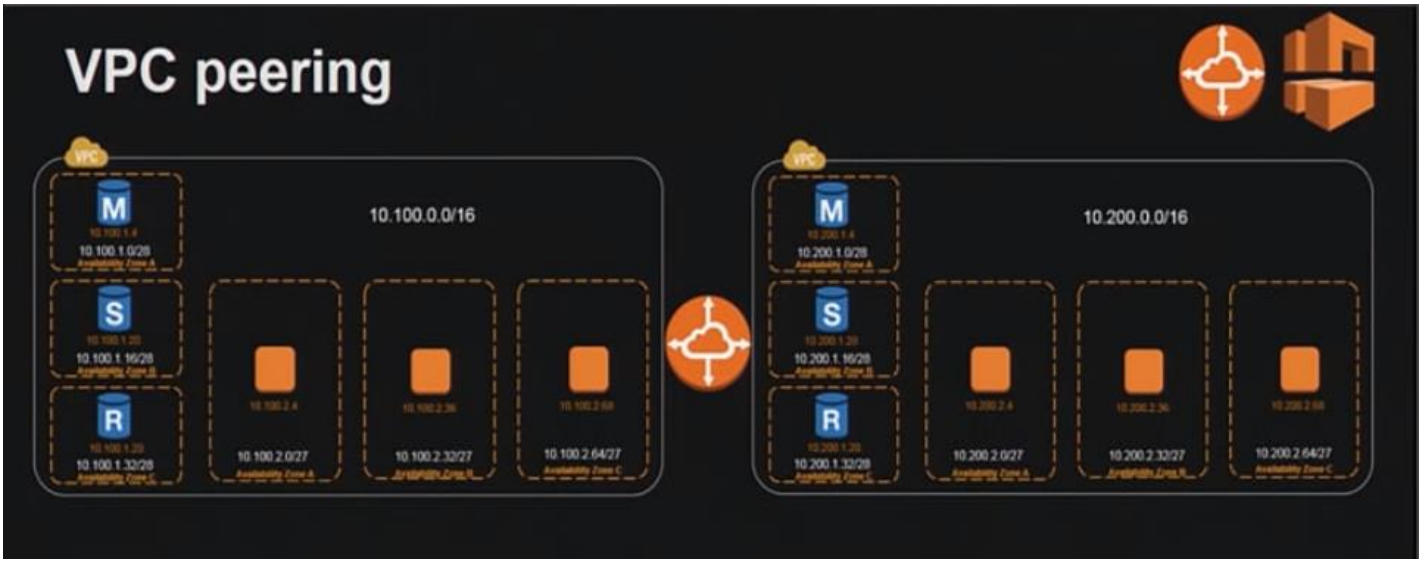
When you create your VPC, you automatically get a Route table by default to use to limit access control between subnets. By default, AWS allows instances in any subnets within a VPC to communicate freely with another instance in any other subnet within that VPC.



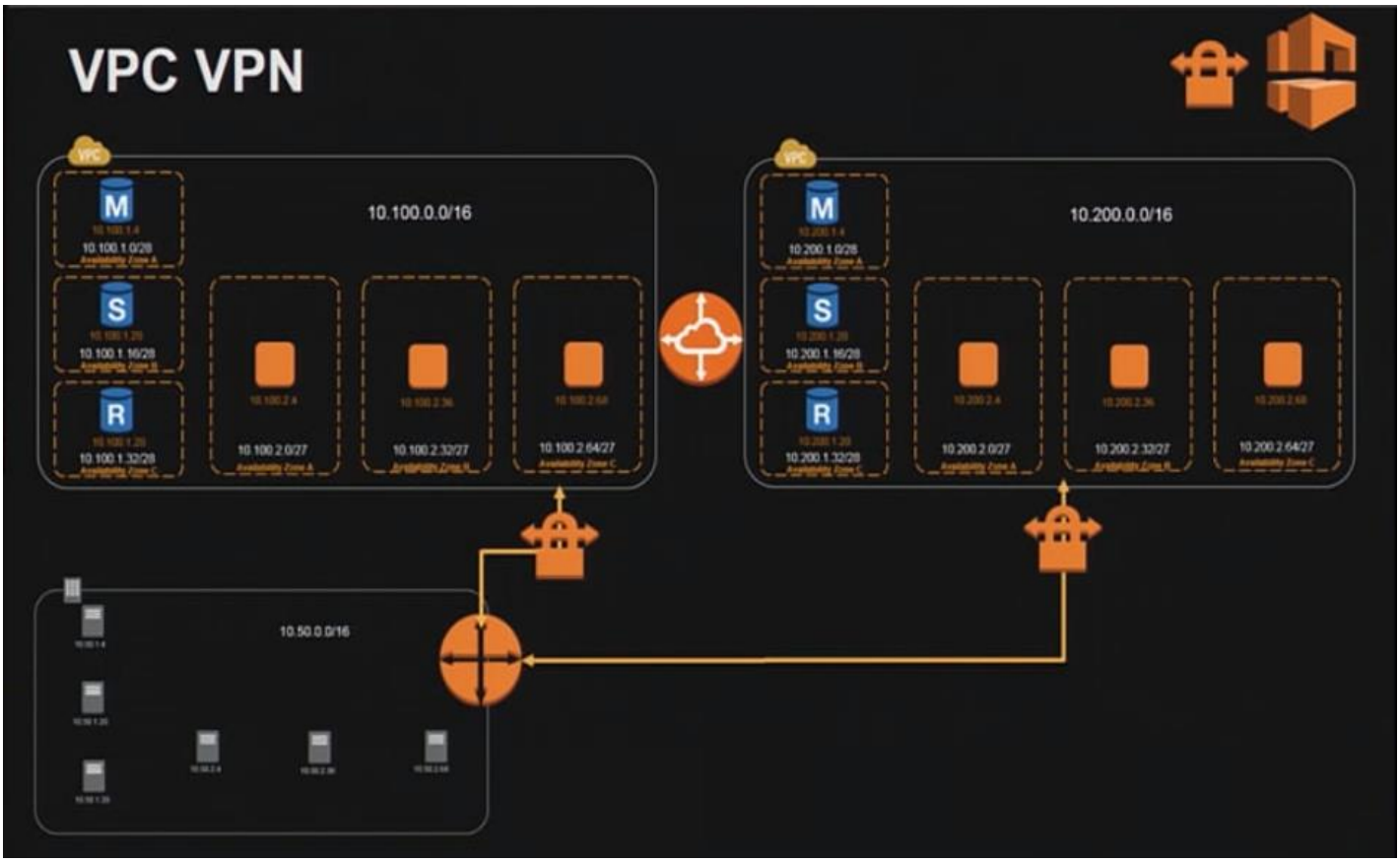
You can limit communication between instances by using security groups. SGs are stateful firewalls (maintain state between your connection sessions) that work around your instances, the go on the Elastic Network Interface ENI on the servers. You can reference another SG from another SG using the SG names.



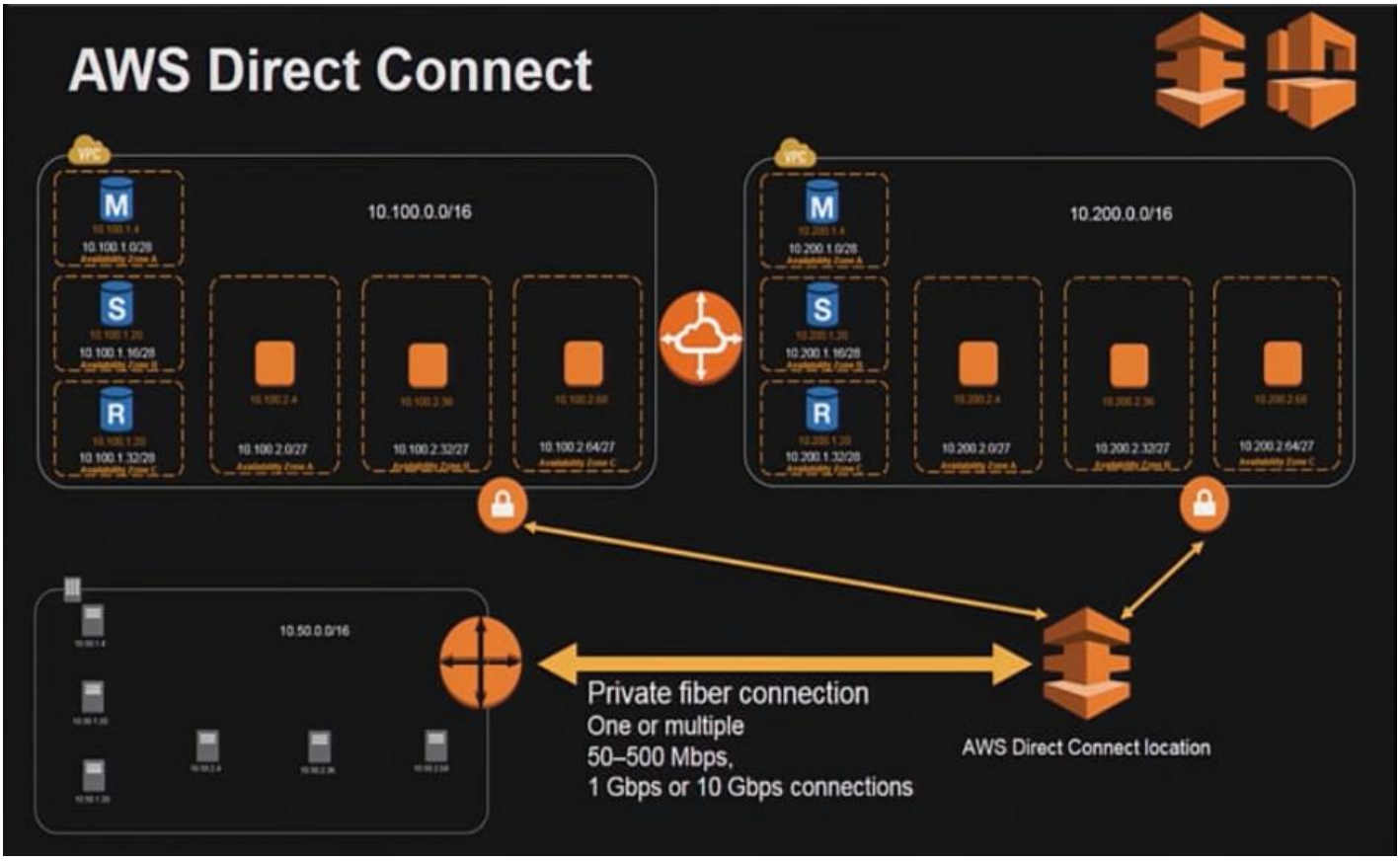
You then need to provide your instances access to internet by using an Internet Gateway and set up routing.



As you start having multiple smaller VPCs per application per customer per department, you are going to need connectivity between them. You can use ***VPC Peering***, this is an invitation model where one VPC sends a peering request to another VPC.



You can create a VPN for communication with your company network, you can use a VPN tunnel for this per VPCs. We have 2 VPN tunnels being used above.

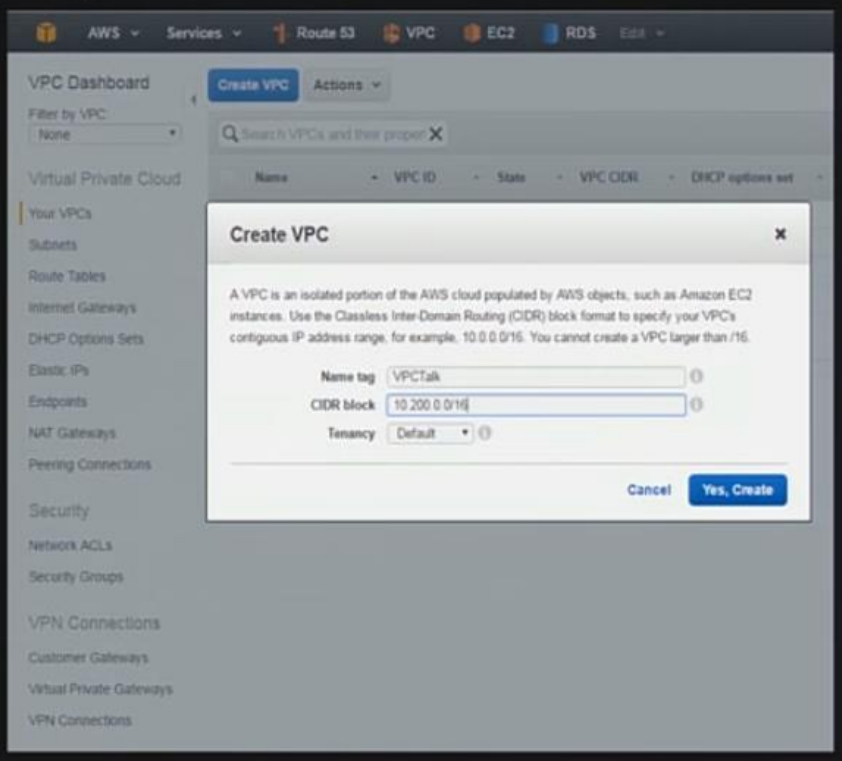


VPN and Direct Connect

- Secure connection to you network
- Pair of IPSec tunnels over the internet
- Dedicated line
- Lower latency and lower per GB data transfer rates
- Failover between each

Amazon VPC

AWS Management Console



AWS Command Line Interface (AWS CLI)

```
[ec2-user@nebulous ~]$ aws ec2 create-vpc --cidr-block 10.200.0.0/16
{
    "Vpc": {
        "VpcId": "vpc-ef33f888",
        "InstanceTenancy": "default",
        "State": "pending",
        "DhcpOptionsId": "dopt-1a504c78",
        "CidrBlock": "10.200.0.0/16",
        "IsDefault": false
    }
}
[ec2-user@nebulous ~]$ aws ec2 create-subnet --vpc-id vpc-ef33f888 --cidr-block 10.200.1.0/28 --
availability-zone us-east-1a
{
    "Subnet": {
        "VpcId": "vpc-ef33f888",
        "CidrBlock": "10.200.1.0/28",
        "State": "pending",
        "AvailabilityZone": "us-east-1a",
        "SubnetId": "subnet-822d55da",
        "AvailableIpAddressCount": 11
    }
}
```

AWS SDKs

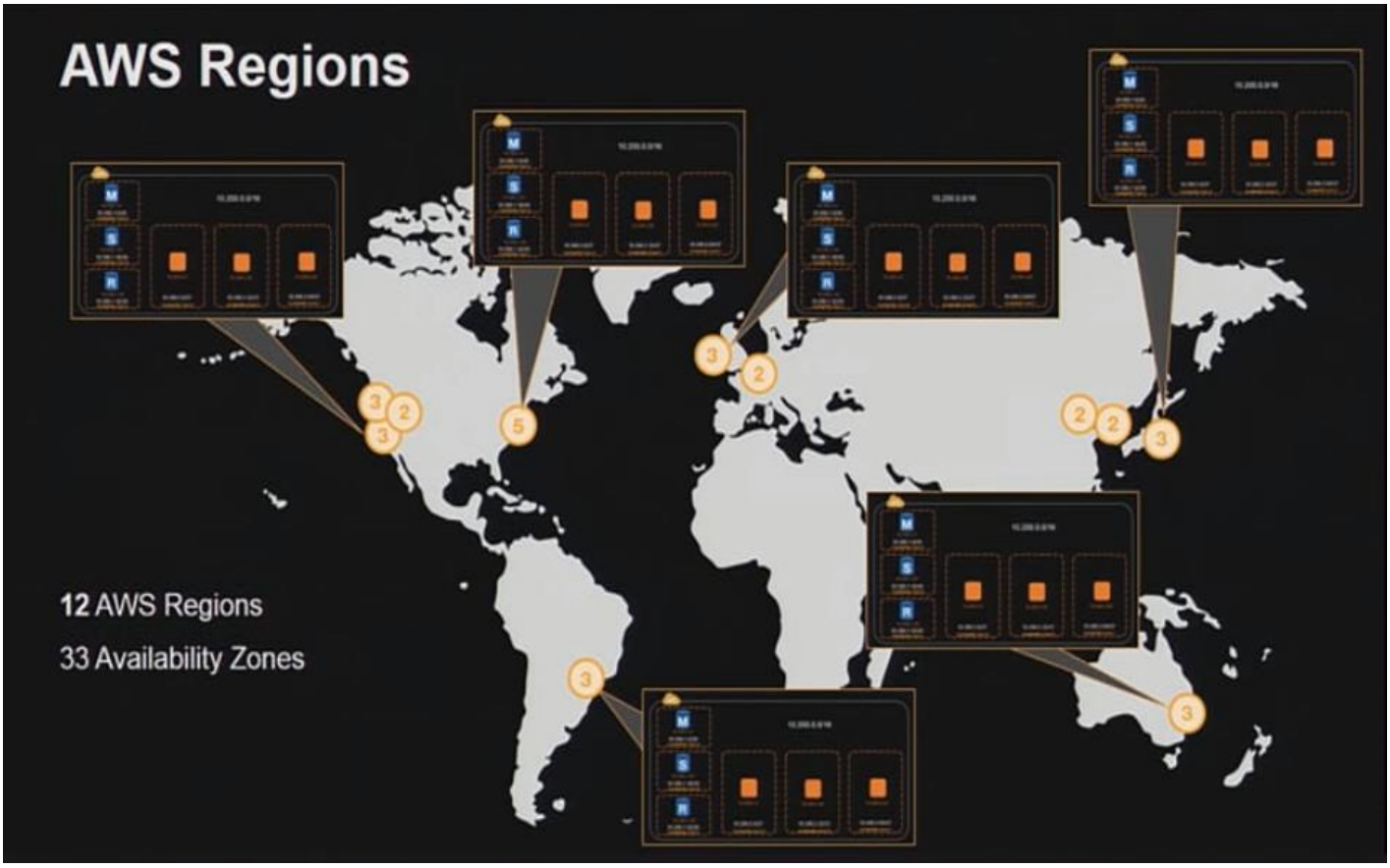
```
var params = {
  CidrBlock: '10.200.0.0/16, /* required */
  DryRun: false,
  InstanceTenancy: 'default'
};
ec2.createVpc(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});

var params = {
  CidrBlock: '10.200.1.0/28', /* required */
  VpcId: ' vpc-ef33f888 ', /* required */
  AvailabilityZone: 'us-east-1a',
  DryRun: false
};
ec2.createSubnet(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

This is an example of creating a VPC using the AWS NodeJS SDK

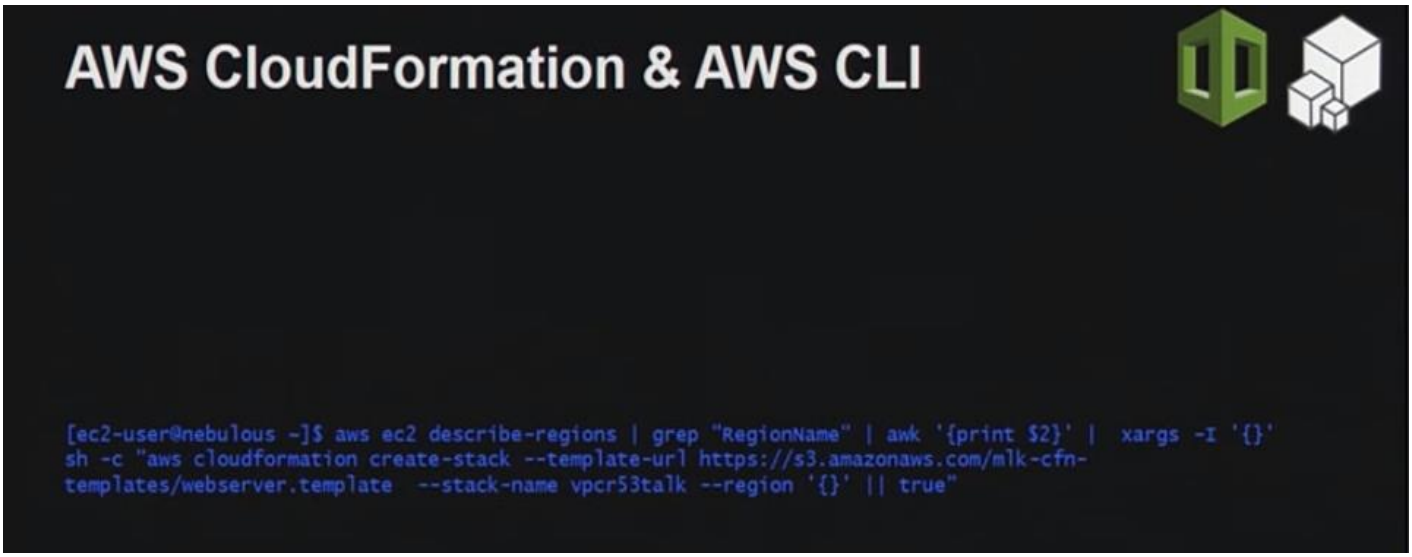
AWS CloudFormation

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Template VPC for VPC Talk",

  "Resources" : {

    "VPC" : {
      "Type" : "AWS::EC2::VPC",
      "Properties" : {
        "CidrBlock" : "10.200.0.0/16",
        "Tags" : [ {"Key" : "Application", "Value" : { "Ref" : "AWS::StackId"} } ]
      }
    },

    "Subnet" : {
      "Type" : "AWS::EC2::Subnet",
      "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "CidrBlock" : "10.200.1.0/28",
        "Tags" : [ {"Key" : "Application", "Value" : { "Ref" : "AWS::StackId"} } ]
      }
    },
```

We can also launch the VPC creation using AWS CloudFormation template that we save as a JSON file



You can then take that same template and launch it in many regions



You can use these 2 together to further build out your VPC infrastructure, the example above uses a CF template that we used to create our web application stack and launches it across all regions our account has access to

Amazon Route 53

Route 53 overview

- Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service
- Distributed globally
- Integrates with other AWS services
- Can be used for on-premises and hybrid setups
- Simple to use

Route 53 features

- Latency based routing
- Geo DNS
- Weighted round robin
- DNS failover
- Health checks
- Private DNS for VPC
- Domain name registration & transfer

Route 53 SLA

100% Available

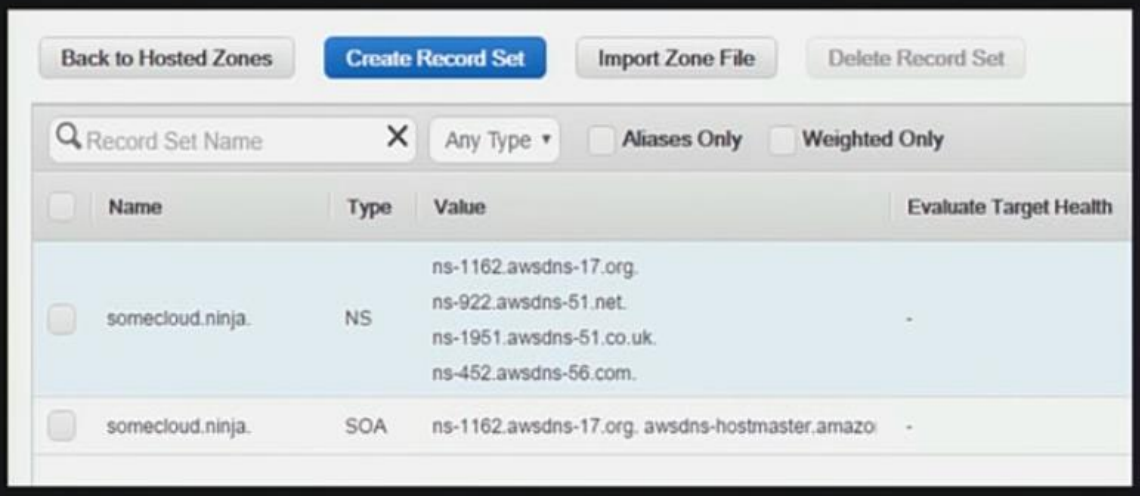
SLA details: https://aws.amazon.com/route53/sla/

Route 53 pricing

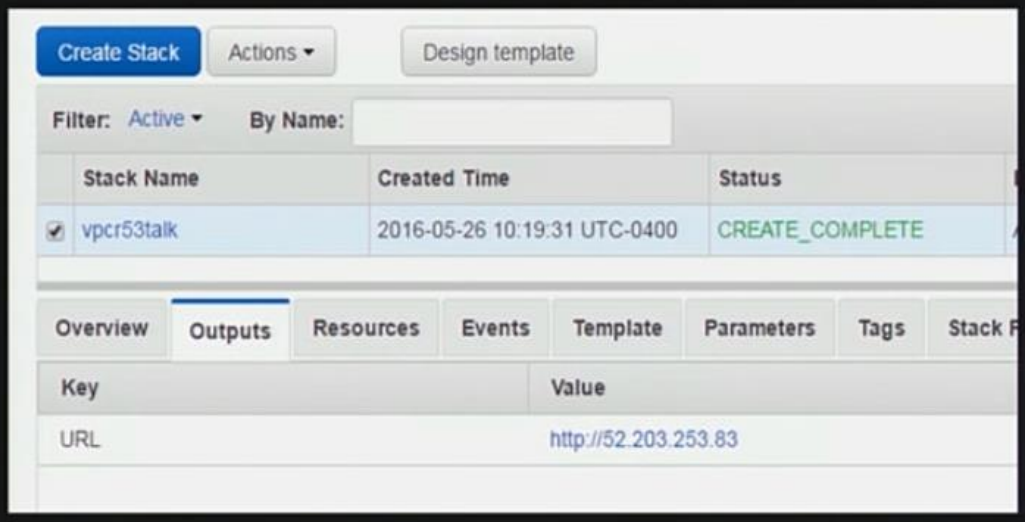
- Hosted zones
  - $0.50 per hosted zone/month for the first 25 hosted zones
  - $0.10 per hosted zone/month for additional hosted zones
- Standard queries
  - $0.400 per million queries—first 1 billion queries/month
  - $0.200 per million queries—over 1 billion queries/month
- Latency based routing queries
  - $0.600 per million queries—first 1 billion queries/month
  - $0.300 per million queries—over 1 billion queries/month
- Geo DNS queries
  - $0.700 per million queries—first 1 billion queries/month
  - $0.350 per million queries—over 1 billion queries/month

Route 53 domain registration

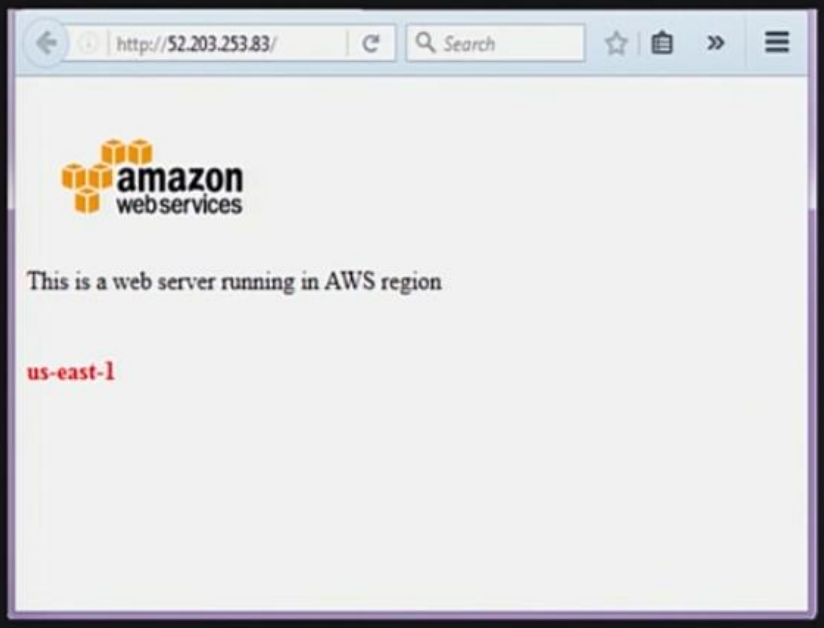
Route 53 domain registration



Website in us-east-1



Sample website

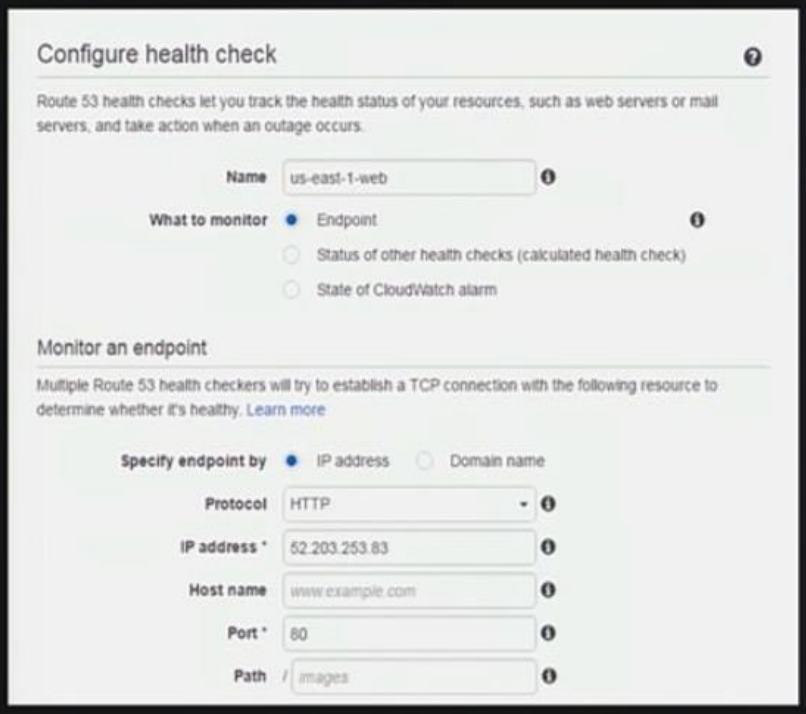


AWS CloudFormation

```
[ec2-user@nebulous ~]$ aws ec2 describe-regions | grep "RegionName" | awk '{print $2}' |  xargs -I '{}'  sh -c "aws
cloudformation describe-stacks --region '{}' || true" | grep "OutputValue" | awk '{print $2}'
"http://54.72.210.244"
"http://52.77.119.167"
"http://52.62.2.174"
"http://52.58.203.28"
"http://52.78.4.248"
"http://52.196.172.135"
"http://52.203.253.83"
"http://52.67.33.11"
"http://52.9.240.65"
"http://52.40.118.107"
```

We now need to make some DNS records for each of this server endpoints that we have for our web servers launched in each region

Health checks



First, we create/configure health checks for each of the DNS records that we will be creating for our 10 application endpoints

Health checks

Get notified when health check fails

If you want CloudWatch to send you an Amazon SNS notification, such as an email, when the status of the health check changes to unhealthy, create an alarm and specify where to send notifications.

Create alarm ● Yes ○ No

CloudWatch sends you an Amazon SNS notification whenever the status of this health check is unhealthy for one minute.

Send notification to ○ Existing SNS topic ● New SNS topic

Topic name * ServerAlarm

Recipient email addresses * kuentzm@amazon.com

Separate multiple addresses with a comma, a semicolon, or a space

* Required

Cancel | Previous | Create health check

Health checks

Advanced configuration

Request interval ○ Standard (30 seconds) ● Fast (10 seconds)

Failure threshold * 3

String matching ○ No ● Yes

Search string * web server running

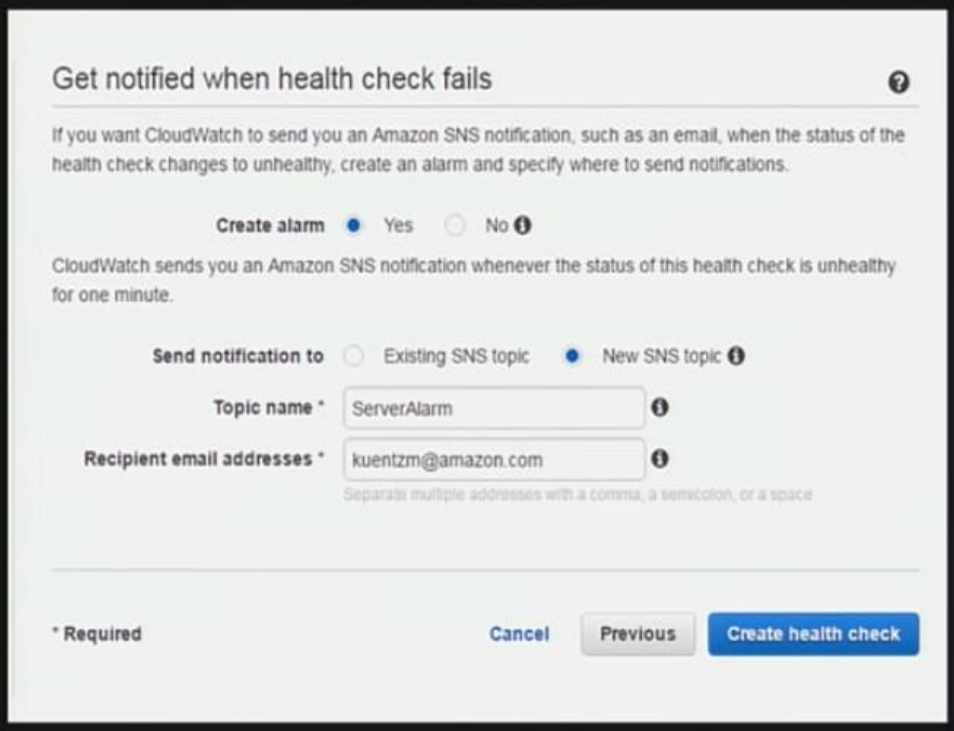
Latency graphs ☑

Invert health check status ☐

Health checker regions ○ Customize ● Use recommended

US East (N. Virginia)
US West (N. California)
US West (Oregon)
EU (Ireland)
Asia Pacific (Singapore)
Asia Pacific (Sydney)
Asia Pacific (Tokyo)
South America (São Paulo)

Health checks



Health checks

```
[ec2-user@nebulous ~]$ aws route53 create-health-check --caller-reference $RANDOM  --health-check-config
IPAddress=52.203.253.83,Port=80,Type=HTTP_STR_MATCH,SearchString="web server
running",RequestInterval=10,FailureThreshold=3,MeasureLatency=true,Inverted=false,EnableSNI=false
```


```
{
    "HealthCheck": {
        "HealthCheckConfig": {
            "SearchString": "web server running",
            "IPAddress": "52.203.253.83",
            "EnableSNI": false,
            "Inverted": false,
            "MeasureLatency": true,
            "RequestInterval": 10,
            "Type": "HTTP_STR_MATCH",
            "Port": 80,
            "FailureThreshold": 3
        },
        "CallerReference": "1008",
        "HealthCheckVersion": 1,
        "Id": "0f779143-14ff-4ff0-9476-12a2467f0f1a"
    },
    "Location": "https://route53.amazonaws.com/2015-01-01/healthcheck/0f779143-14ff-4ff0-9476-12a2467f0f1a"
}
```

Health checks

Create health check | Delete health check | Edit health check

Filter by keyword

1 to 1 of 1 health checks

Name	Status	Description
us-east-1-web	17 minutes ago 2 minutes ago Healthy	http://52.203.253.83:80/

Health checks

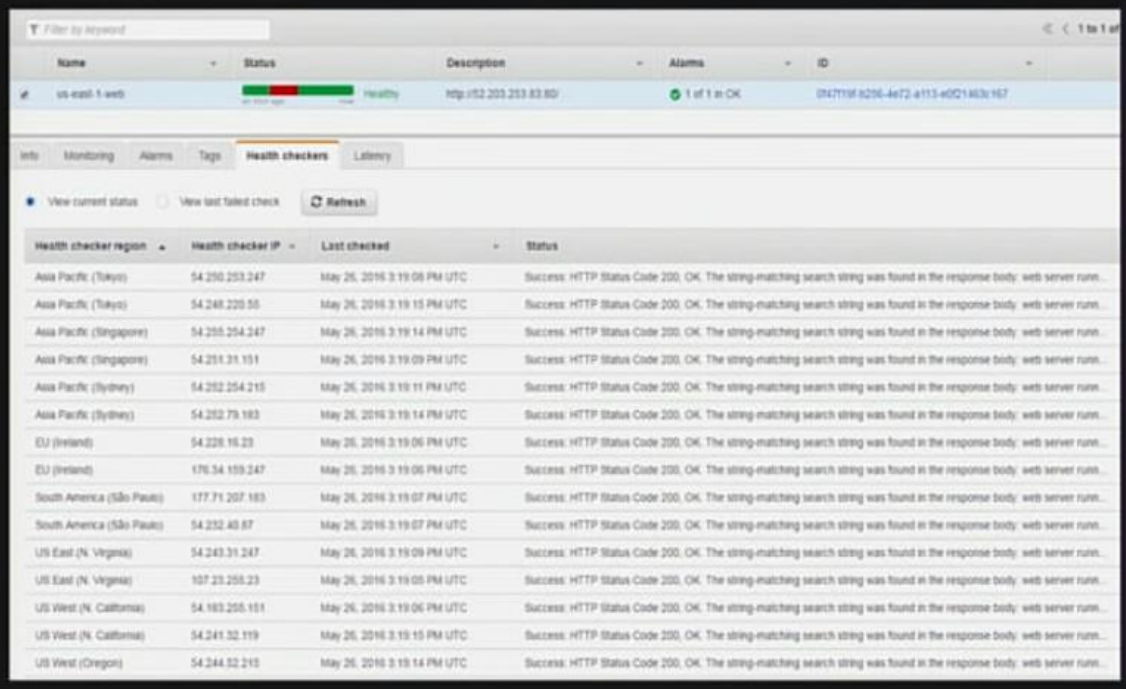
Name	Status	Description	Alarms	ID
us-east-1-web	10 minutes ago now Unhealthy	http://52.203.253.83:80/	1 of 1 in ALARM	0f47f19f-b256-4e72-a11

Info | Monitoring | Alarms | Tags | Health checkers | Latency

View current status | View last failed check | Refresh

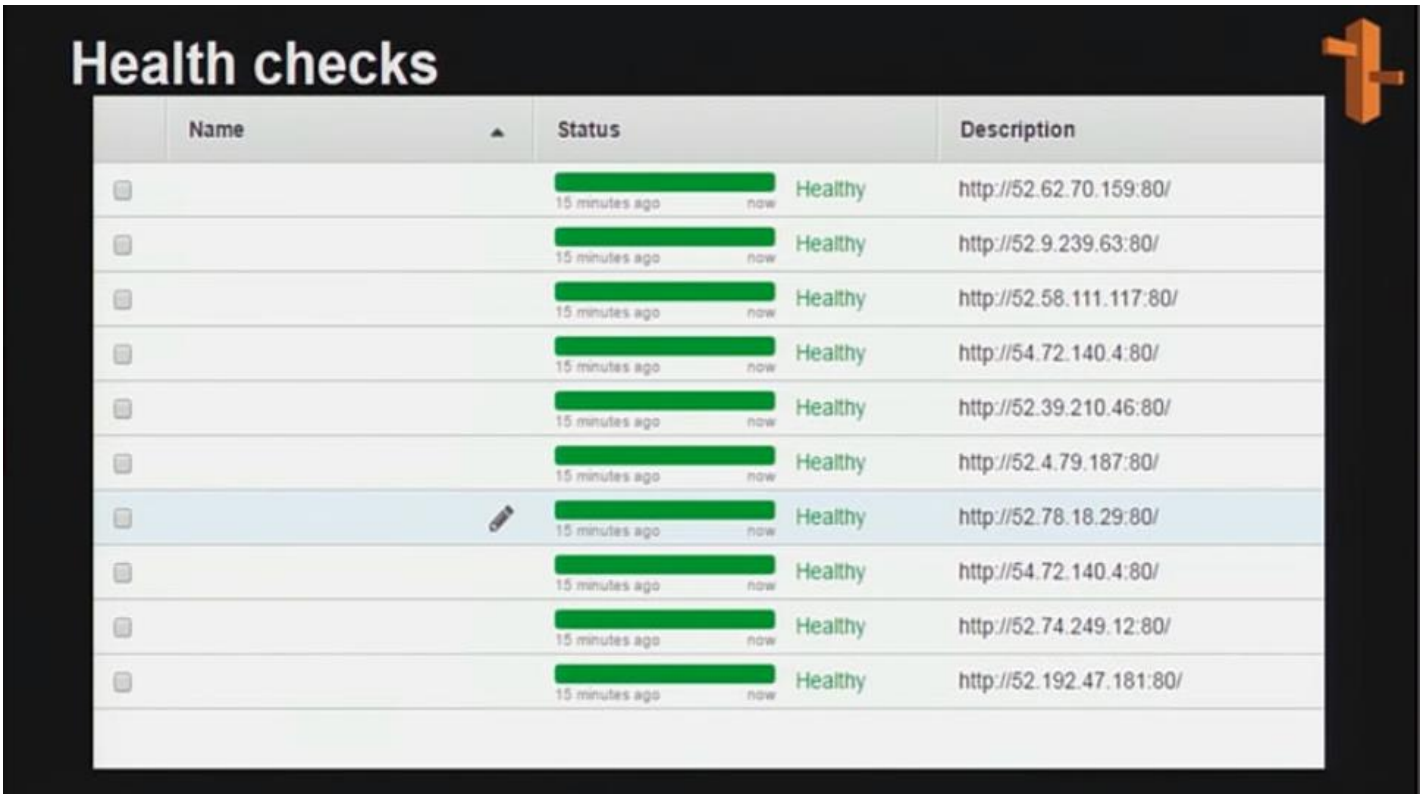
Health checker region	Health checker IP	Last checked	Status
Asia Pacific (Tokyo)	54.250.253.247	May 26, 2016 3:16:41 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
Asia Pacific (Tokyo)	54.248.220.55	May 26, 2016 3:16:37 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
Asia Pacific (Singapore)	54.255.254.247	May 26, 2016 3:16:34 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
Asia Pacific (Singapore)	54.251.31.151	May 26, 2016 3:16:39 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
Asia Pacific (Sydney)	54.252.254.215	May 26, 2016 3:16:41 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
Asia Pacific (Sydney)	54.252.79.183	May 26, 2016 3:16:34 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
EU (Ireland)	54.228.16.23	May 26, 2016 3:16:42 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
EU (Ireland)	176.34.159.247	May 26, 2016 3:16:43 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
South America (São Paulo)	177.71.207.183	May 26, 2016 3:16:42 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
South America (São Paulo)	54.232.40.87	May 26, 2016 3:16:40 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
US East (N. Virginia)	54.243.31.247	May 26, 2016 3:16:35 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
US East (N. Virginia)	107.23.255.23	May 26, 2016 3:16:42 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
US West (N. California)	54.183.255.151	May 26, 2016 3:16:43 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
US West (N. California)	54.241.32.119	May 26, 2016 3:16:38 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.
US West (Oregon)	54.244.52.215	May 26, 2016 3:16:37 PM UTC	Failure: The health checker could not establish a connection within the timeout limit.

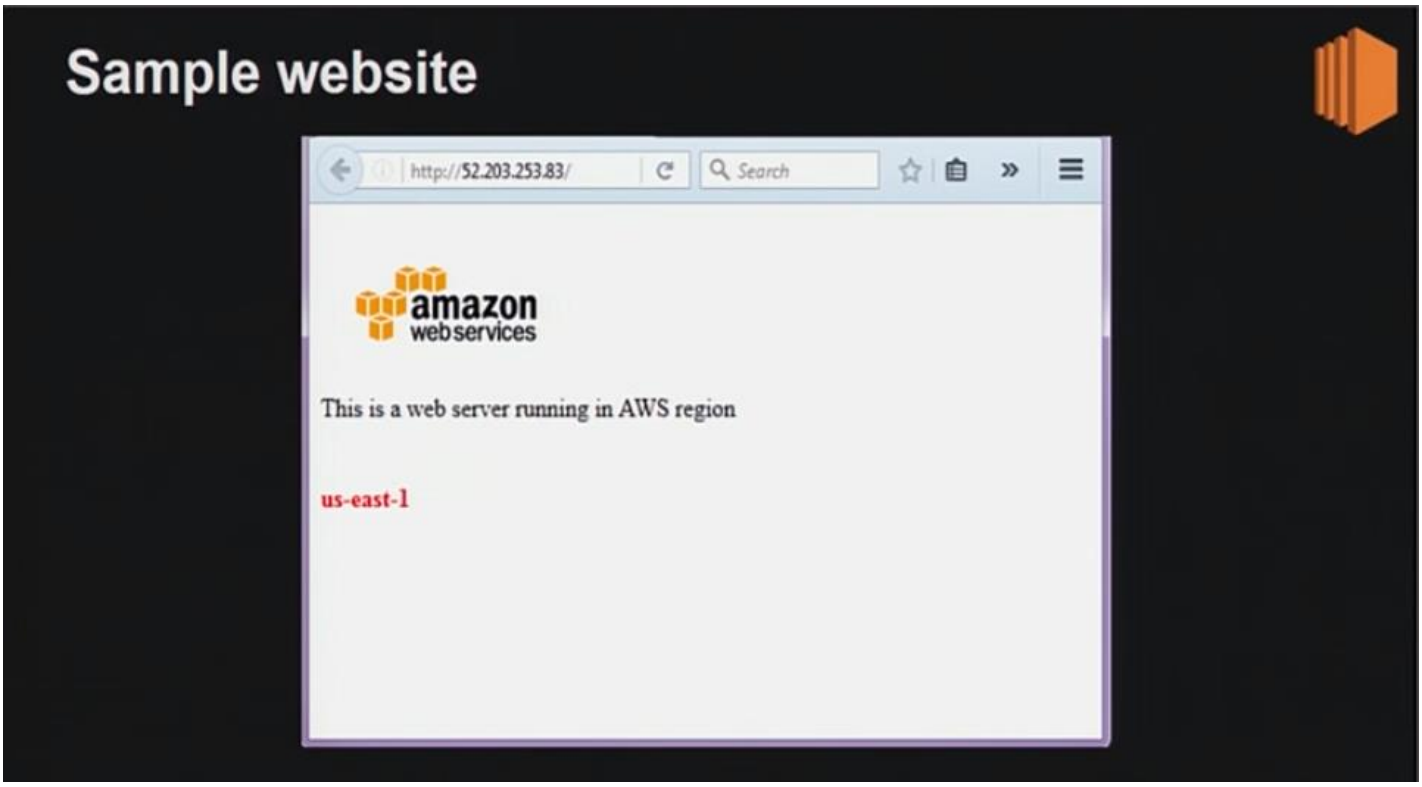
Health checks



Health checks

```
[ec2-user@nebulous ~]$ aws ec2 describe-regions | grep "RegionName" | awk '{print $2}' |  xargs -I '{}'
sh -c "aws cloudformation describe-stacks --region '{}' || true" | egrep "OutputValue" | awk '{print $2}'
| tr 'http:/"' ' ' | awk '{$1=$1};1' | xargs -I '{}' sh -c "aws route53 create-health-check --caller-
reference '{}'  --health-check-config IPAddress='{}',Port=80,Type=HTTP_STR_MATCH,SearchString="web server
running",RequestInterval=10,FailureThreshold=3,MeasureLatency=true,Inverted=false,EnableSNI=false"
```





Now we can use Route53 for the DNS to replace the IP address above

Supported DNS record types

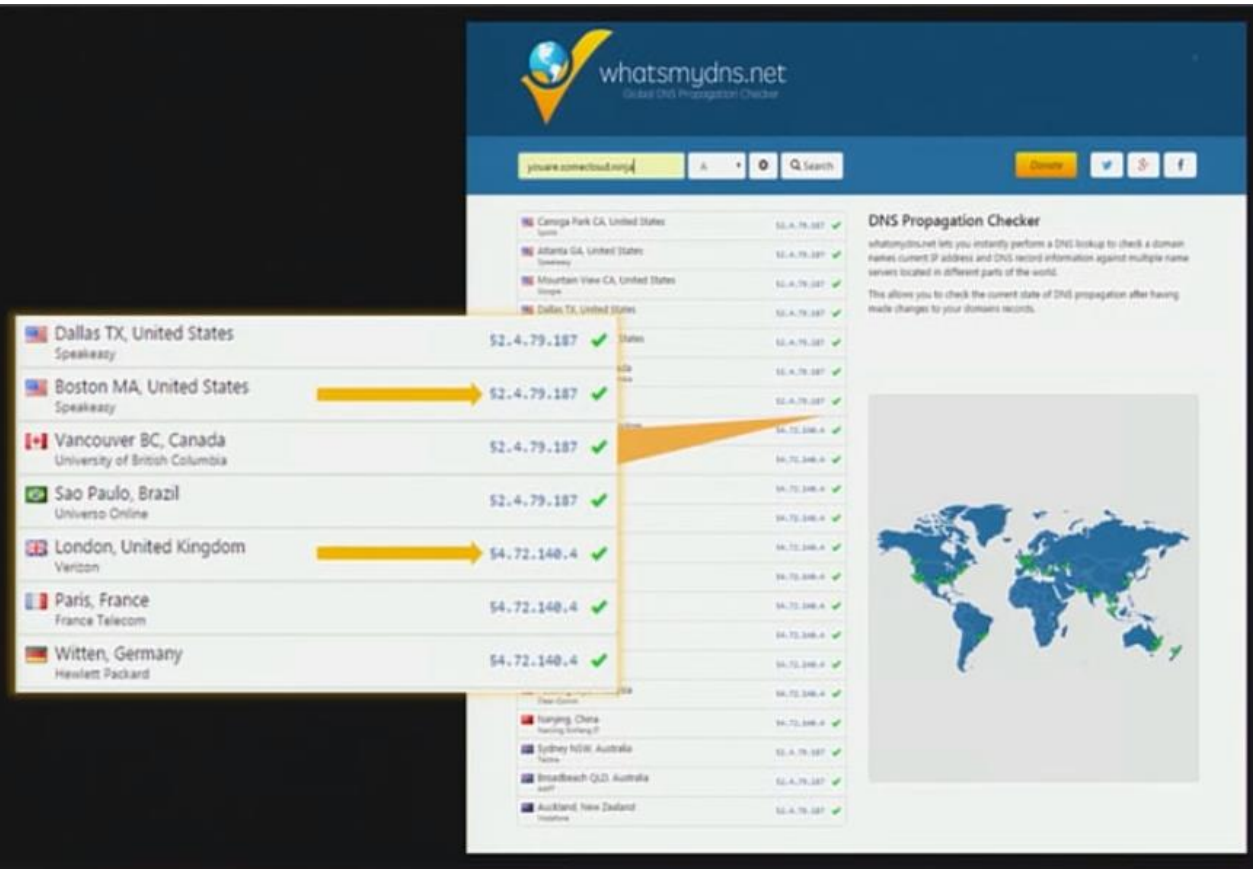
- A
- AAAA
- CNAME
- MX
- NS
- PTR
- SOA
- SPF
- SRV
- TXT

The A record allows us to make a URL for our web app

Latency based record with health check

Latency based record with health check

Type	Value	Evaluate Target Health	Health Check ID	TTL
A	54.72.140.4	-	50c2becb-28f6-4b73-a7e8-4f7c0abfffe8	1
A	52.4.79.187	-	d9ce5e26-6c7e-43ac-8b6f-80c0cf25e6ca	1



Thank you!

amazon
webservices