



Advanced Approaches to Amazon VPC and Amazon Route 53

Mike Kuentz, Solutions Architect

June 21, 2016



© 2016, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

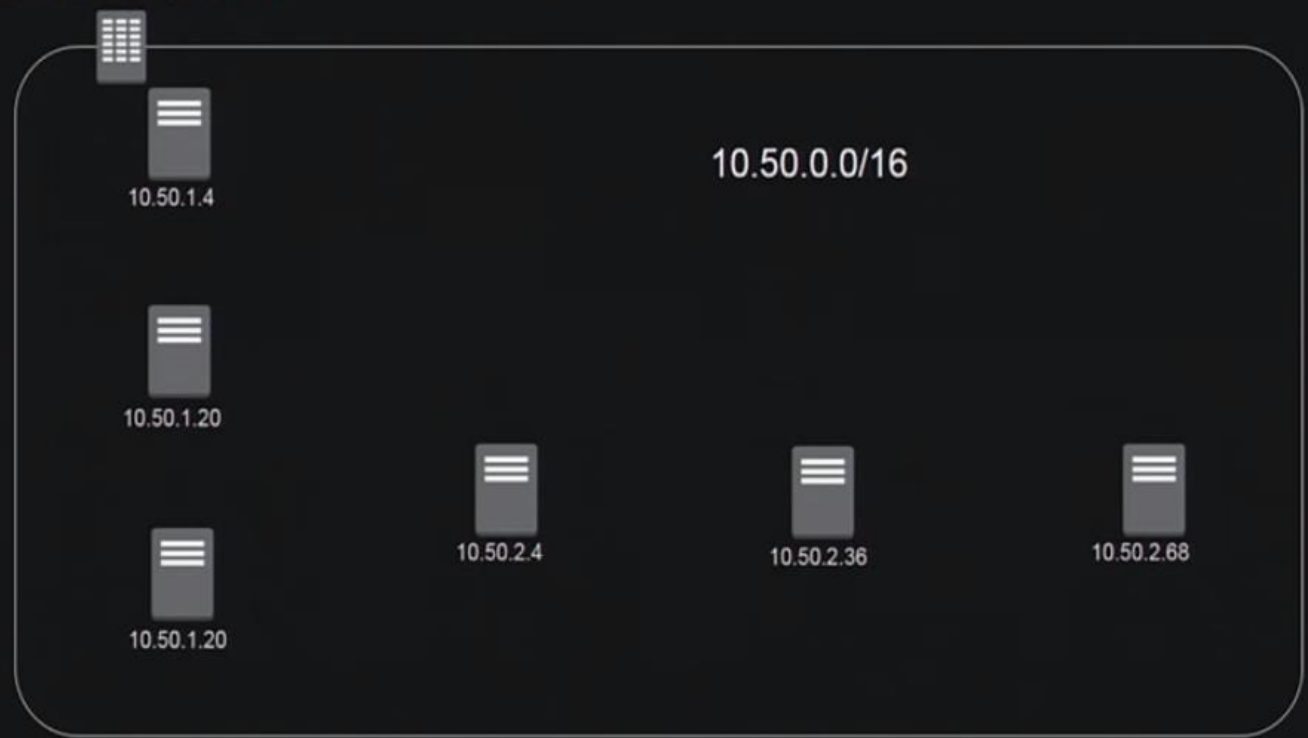
AWS global infrastructure



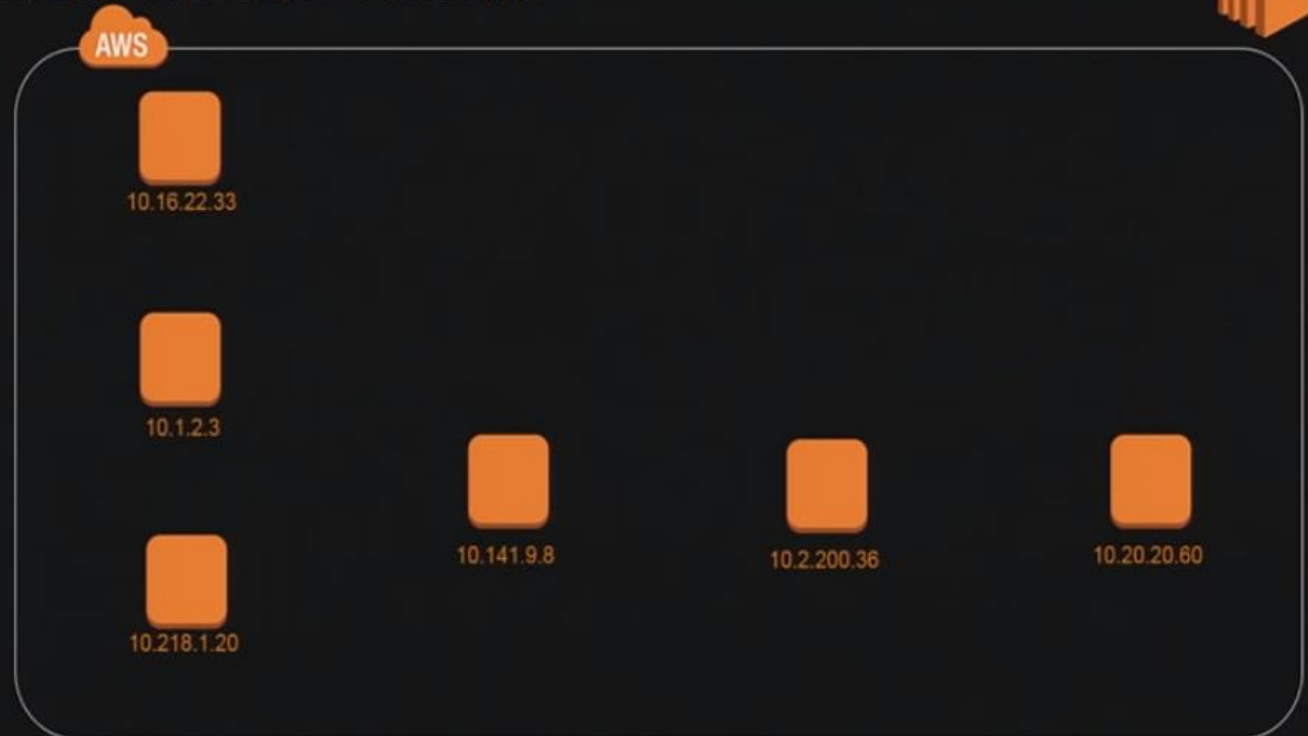
VPC



Data center



Amazon EC2 Classic



Amazon VPC



VPC

10.200.0.0/16

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

Amazon VPC



VPC

10.200.0.0/16

10.200.0.0/16

Availability Zone A

Amazon VPC



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

Amazon VPC



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

Amazon VPC



VPC



10.200.1.4
10.200.1.0/28
Availability Zone A

10.200.0.0/16



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



10.200.1.4
10.200.1.0/28
Availability Zone A

10.200.0.0/16



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

Route tables in a VPC

172.16.0.0
172.16.1.0
172.16.2.0



VPC

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

10.200.0.0

10.200.1.0

10.200.2.0

10.200.0.0/16

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.

Security groups in a VPC



VPC

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

10.200.0.0

10.200.1.0

10.200.2.0

10.200.0.0/16

10.200.2.4

10.200.2.0/27

Availability Zone A

10.200.2.36

10.200.2.32/27

Availability Zone B

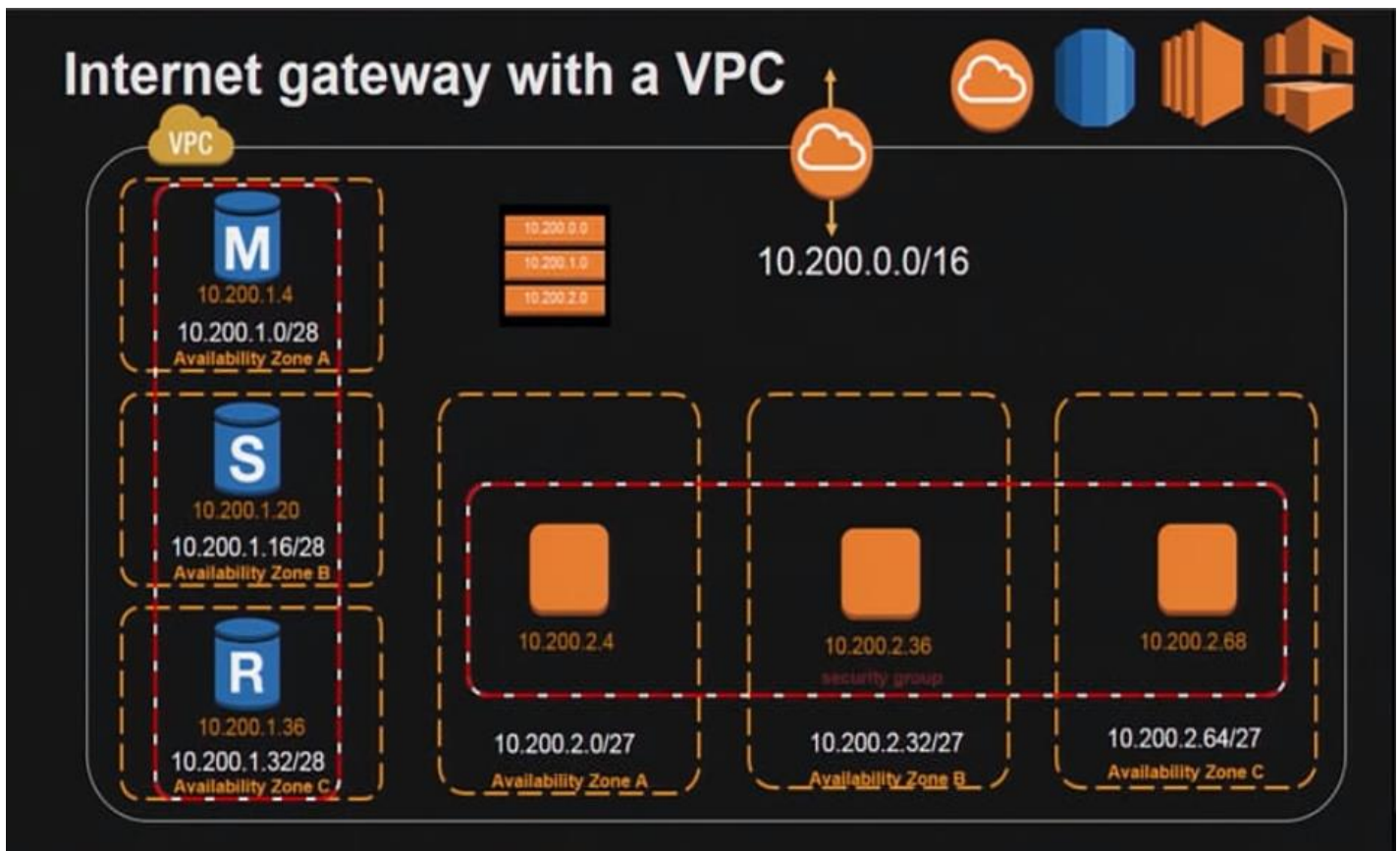
10.200.2.68

10.200.2.64/27

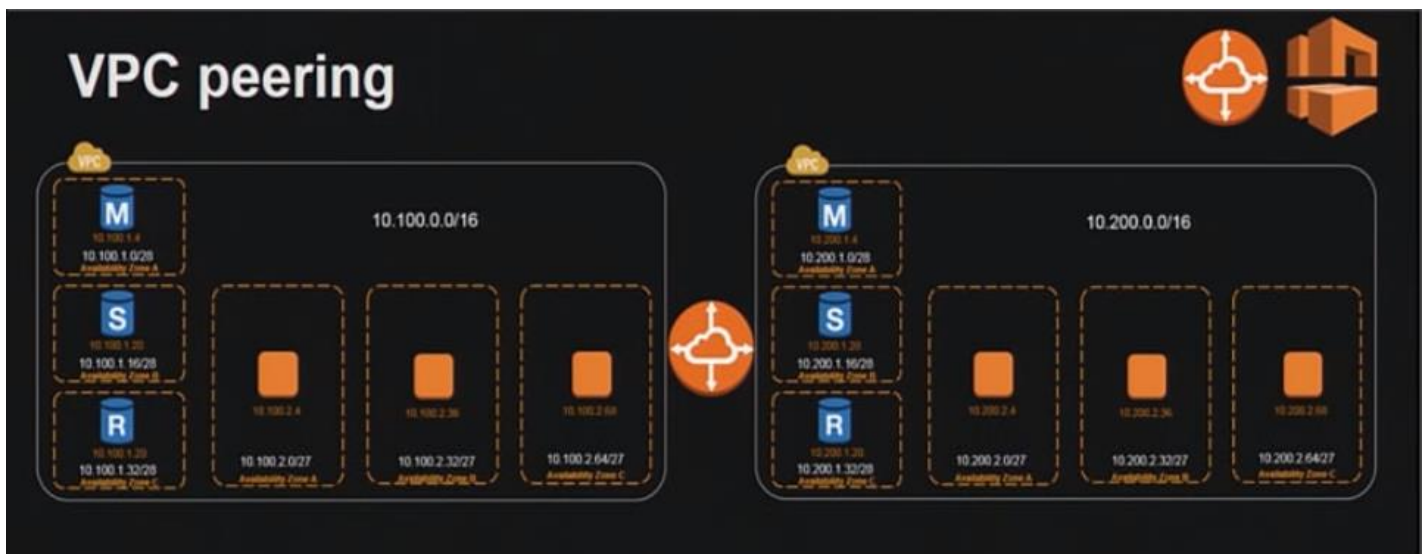
Availability Zone C

security group

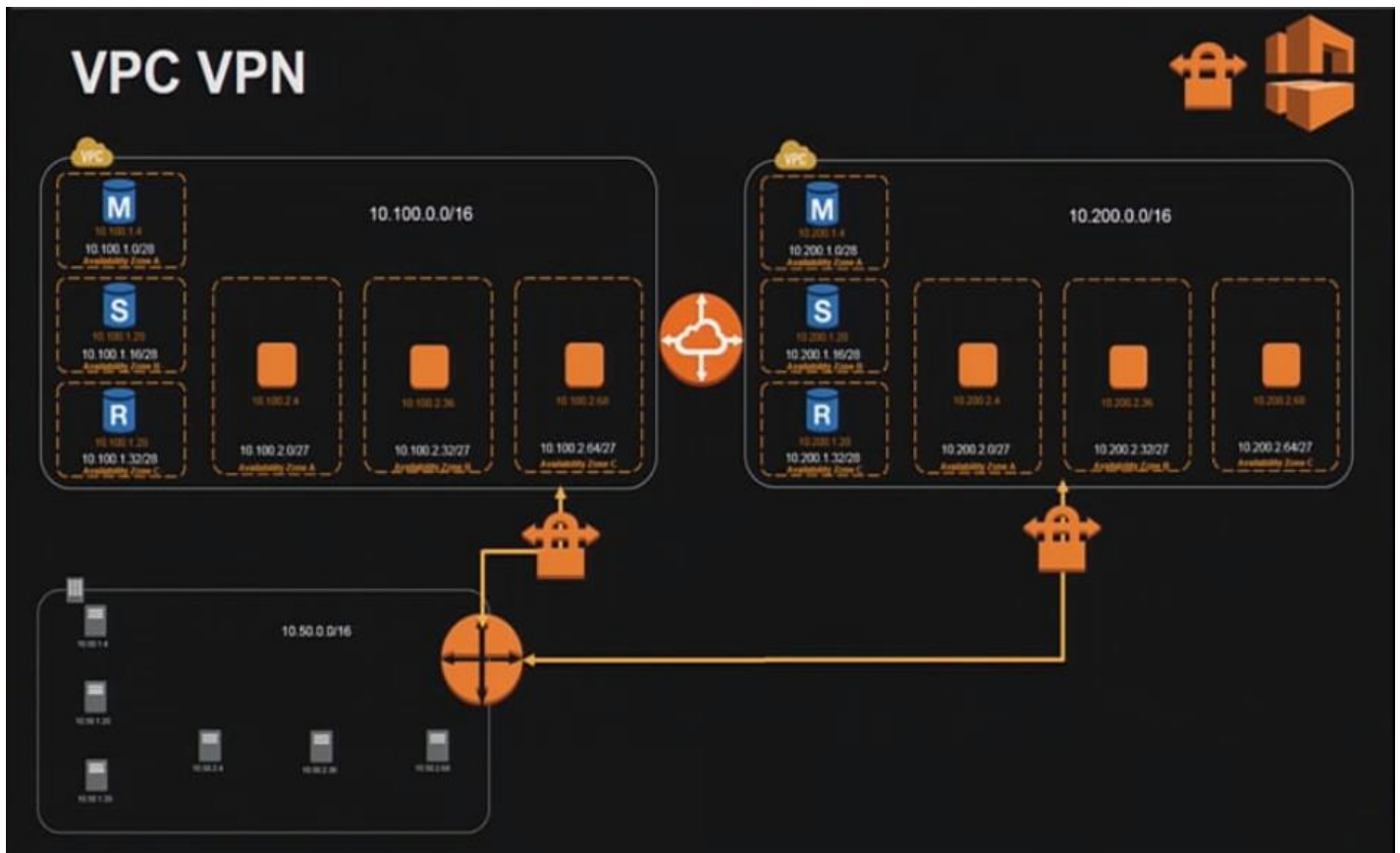
You can limit communication between instances by using security groups. SGs are stateful firewalls (maintain state between your connection sessions) that work around your instances, they go on the Elastic Network Interface ENI on the servers. You can reference another SG from another SG using the SG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



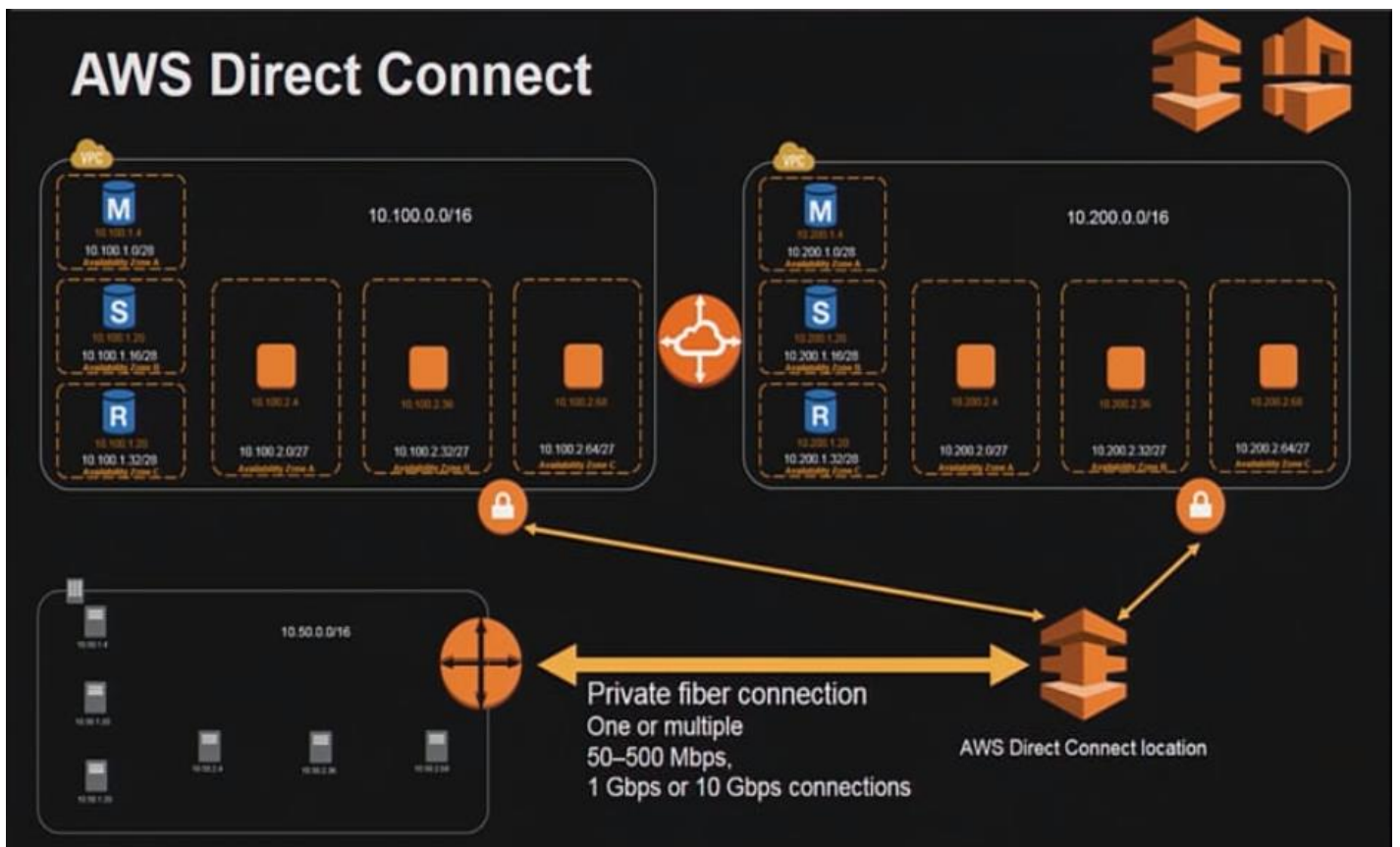
You then need to provide your instances access to internet by using an Internet Gateway and set up routing.



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

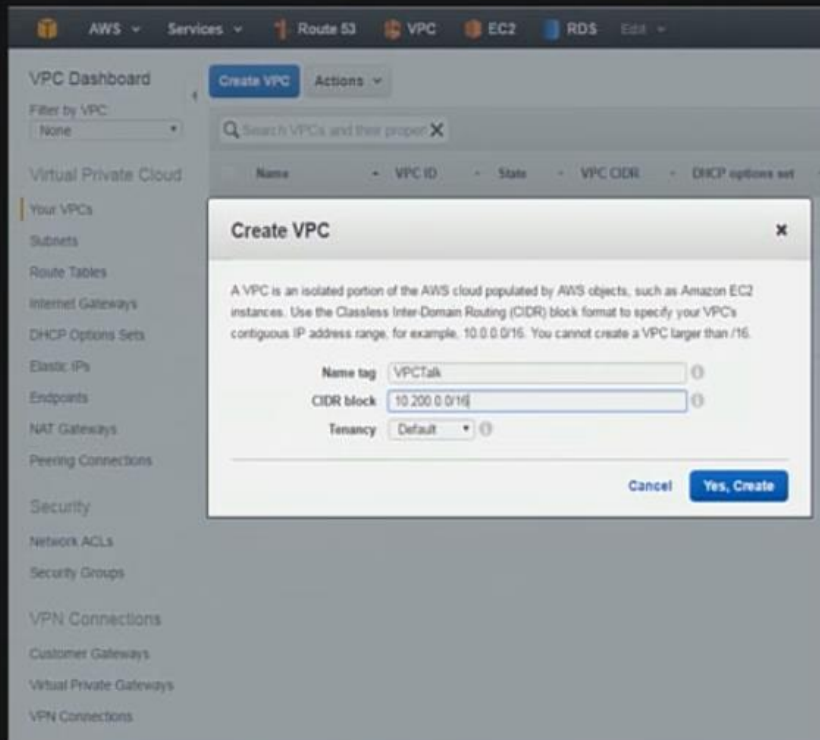


- Secure connection to you network  
- Pair of IPSec tunnels over the internet 
- Dedicated line 
- Lower latency and lower per GB data transfer rates 
- Failover between each  

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

AWS Regions



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

AWS CloudFormation & AWS CLI



```
[ec2-user@nebulous ~]$ aws ec2 describe-regions | grep "RegionName" | awk '{print $2}' | xargs -I '{}' sh -c "aws cloudformation create-stack --template-url https://s3.amazonaws.com/mlk-cfn-templates/webserver.template --stack-name vpcr53talk --region '{}'' || true"
```

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



Choose a domain name

Availability for 'someclouds.ninja'

Domain Name	Status	Price /1 Year	Action
somecloud.ninja	✓ Available	\$18.00	<input type="button" value="Add to cart"/>

Route 53 domain registration

☐ Aliases Only☐ Weighted Only

<input type="checkbox"/>	Name	Type	Value	Evaluate Target Health
<input type="checkbox"/>	somecloud.ninja.	NS	ns-1162.awsdns-17.org. ns-922.awsdns-51.net. ns-1951.awsdns-51.co.uk. ns-452.awsdns-56.com.	-
<input type="checkbox"/>	somecloud.ninja.	SOA	ns-1162.awsdns-17.org. awsdns-hostmaster,amazo	-

Website in us-east-1



Create Stack Actions Design template

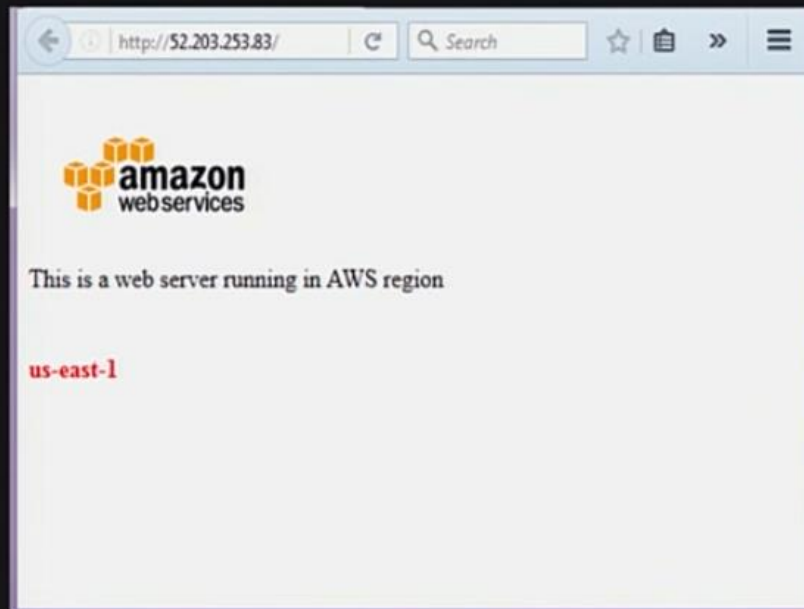
Filter: Active By Name:

Stack Name	Created Time	Status
<input checked="" type="checkbox"/> vpcr53talk	2016-05-26 10:19:31 UTC-0400	CREATE_COMPLETE

Overview Outputs Resources Events Template Parameters Tags Stack P

Key	Value
URL	http://52.203.253.83

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



Configure health check ⓘ

Route 53 health checks let you track the health status of your resources, such as web servers or mail servers, and take action when an outage occurs.

Name ⓘ

What to monitor ☒ Endpoint ⓘ

☐ Status of other health checks (calculated health check)

☐ State of CloudWatch alarm

Monitor an endpoint

Multiple Route 53 health checkers will try to establish a TCP connection with the following resource to determine whether it's healthy. [Learn more](#)

Specify endpoint by ☒ IP address ☐ Domain name

Protocol ⓘ

IP address * ⓘ

Host name ⓘ

Port * ⓘ

Path ⓘ

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 * ?

Recipient email addresses * ?

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 * ?

String matching ☐ No ☒ Yes ?

Search string * ?

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



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 * ?

Recipient email addresses * ?

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

* Required

[Cancel](#)

[Previous](#)

[Create health check](#)

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](#)

<< < 1 to 1 of 1 health checks > >>

	Name	Status	Description
<input type="checkbox"/>	us-east-1-web	<div></div> Healthy 17 minutes ago 2 minutes ago	http://52.203.253.83:80/

Health checks



Name	Status	Description	Alarms	ID
<input checked="" type="checkbox"/> us-east-1-web	<div></div> Unhealthy 10 minutes ago	http://52.203.253.83:80/	1 of 1 in ALARM	0f47719f-6256-4e72-a11c-

[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



Filter by keyword

1 to 1 of

Name	Status	Description	Alarms	ID
us-east-1-web	Healthy	http://52.203.253.83/	1 of 1 in OK	014711f8-6296-4e72-a113-e0c183c1657

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 3:19:08 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
Asia Pacific (Tokyo)	54.248.220.55	May 26, 2016 3:19:15 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
Asia Pacific (Singapore)	54.255.254.247	May 26, 2016 3:19:14 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
Asia Pacific (Singapore)	54.254.31.151	May 26, 2016 3:19:09 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
Asia Pacific (Sydney)	54.252.254.215	May 26, 2016 3:19:11 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
Asia Pacific (Sydney)	54.252.79.183	May 26, 2016 3:19:14 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
EU (Ireland)	54.228.16.23	May 26, 2016 3:19:06 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
EU (Ireland)	176.54.159.247	May 26, 2016 3:19:06 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
South America (São Paulo)	177.71.207.183	May 26, 2016 3:19:07 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
South America (São Paulo)	54.252.40.87	May 26, 2016 3:19:07 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
US East (N. Virginia)	54.243.31.247	May 26, 2016 3:19:09 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
US East (N. Virginia)	107.23.258.23	May 26, 2016 3:19:05 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
US West (N. California)	54.183.255.181	May 26, 2016 3:19:06 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
US West (N. California)	54.241.32.119	May 26, 2016 3:19:15 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.
US West (Oregon)	54.244.52.219	May 26, 2016 3:19:14 PM UTC	Success: HTTP Status Code 200, OK. The string-matching search string was found in the response body: web server running.

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"
```

Health checks



	Name		Status	Description
			<div><div></div></div> 15 minutes ago now Healthy	http://52.62.70.159:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.9.239.63:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.58.111.117:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://54.72.140.4:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.39.210.46:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.4.79.187:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.78.18.29:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://54.72.140.4:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.74.249.12:80/
			<div><div></div></div> 15 minutes ago now Healthy	http://52.192.47.181:80/

Sample website



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



The screenshot shows the 'Edit Record Set' interface in the AWS Route 53 console. The form is configured as follows:

- Name:** youare.somecloud.ninja
- Type:** A - IPv4 address
- Alias:** ☐ Yes ☒ No
- TTL (Seconds):** 1 | 1m | 5m | 1h | 1d
- Value:** 52.4.79.187
- Routing Policy:** Latency
- Region:** us-east-1
- Set ID:** us-east-1
- Associate with Health Check:** ☒ Yes ☐ No
- Health Check to Associate:** us-east-1

Below the 'Health Check to Associate' dropdown, there is explanatory text: 'When responding to queries, Route 53 can omit resources that fail health checks. [Learn More](#)'.

Latency based record with health check



Type ^	Value	Evaluate Target Health ^	Health Check ID	TTL ^
A	54.72.140.4	-	50c2becb-28f6-4b73-a7e8-4f7c0abffe8	1
A	52.4.79.187	-	d9ce5e26-6c7e-43ac-8b6f-80c0cf25e6ca	1

The screenshot shows the 'whatsmydns.net' website, a 'Global DNS Propagation Checker'. The search bar contains 'yourare.comcloud.org'. The results are displayed in a table with columns for location, IP address, and status. A world map on the right shows the global distribution of the test servers.

Location	IP Address	Status
Canada Park CA, United States	52.4.79.187	✓
Atlanta GA, United States	52.4.79.187	✓
Mountain View CA, United States	52.4.79.187	✓
Dallas TX, United States	52.4.79.187	✓
Dallas TX, United States (Speakaz)	52.4.79.187	✓
Boston MA, United States (Speakaz)	52.4.79.187	✓
Vancouver BC, Canada (University of British Columbia)	52.4.79.187	✓
Sao Paulo, Brazil (Univero Online)	52.4.79.187	✓
London, United Kingdom (Verizon)	54.72.140.4	✓
Paris, France (France Telecom)	54.72.140.4	✓
Witten, Germany (Hewlett Packard)	54.72.140.4	✓
Hong Kong, China (Hawking)	54.72.140.4	✓
Sydney NSW, Australia (Telstra)	52.4.79.187	✓
Broadbeach QLD, Australia (apt)	52.4.79.187	✓
Auckland, New Zealand (Hewlett Packard)	52.4.79.187	✓

DNS Propagation Checker
whatsmydns.net lets you instantly perform a DNS lookup to check a domain. Applies current IP address and DNS record information against multiple name servers located in different parts of the world. This allows you to check the current state of DNS propagation after having made changes to your domains records.



Thank you!

