Hybrid Cloud DNS Solutions for Amazon VPC

October 2017

Notices

Contents

Abstract

The Domain Name System (DNS) is a foundational element of the internet that underpins many services offered by Amazon Web Services (AWS). The Amazon Virtual Private Cloud (VPC) provides a DNS server to provide this resolution for public, VPC, and Route 53 private hosted zones. Customers with workloads that leverage both the AWS VPC and on-premises resources often need to resolve private DNS records hosted on-premises too. Currently, this can’t be accomplished by the Amazon provided DNS server alone. If your organization needs to perform name resolution of both your VPC and on-premises workloads, you can use the solutions in this paper to achieve that goal.

Introduction

For many organizations with private data centers, operating in a hybrid architecture is a necessary part of their cloud adoption journey. For customers with *hybrid* workloads that are comprised of both on-premises and cloud-based resources, DNS name resolution is an essential service. AWS services that require name resolution could include your Elastic Load Balancing (ELB) load balancer, Amazon Relational Database Service (Amazon RDS), Amazon Redshift, or Amazon Elastic Compute Cloud (Amazon EC2). DNS enables you to decouple the IP address from the resource in question, which is a basic building block that enables us to build highly scalable and available architectures.

The Amazon DNS server (AmazonProvidedDNS), which is available by default in all Amazon VPCs, responds to DNS queries for public records, Amazon VPC resources, and Amazon Route 53 private hosted zones. However, it can't send queries to customer managed authoritative DNS servers hosted on-premises or in Amazon VPC. These are the DNS servers that are authoritative for the records of the customer managed zones and can therefore answer for them. An intermediary solution is required for when a customer needs name resolution for these records in addition to what the AmazonProvidedDNS can answer for. One such solution is to use multiple EC2 instances running a DNS forwarder to either the AmazonProvidedDNS server or your organization's on-premises DNS server. Another solution is to replicate the on-premises DNS domain to a Route 53 private hosted zone.

This paper illustrates several different architectures that you can implement on AWS. These architectures meet the need for name resolution of on-premises infrastructure from your Amazon VPC and address constraints that have only been partially addressed by previously published solutions.

Key Concepts

Before we dive into the solutions, it is important that we establish a few concepts and configuration options that we'll reference throughout this paper.

Amazon VPC DHCP Options Set

The Dynamic Host Configuration Protocol (DHCP) provides a standard for passing configuration information to hosts on a TCP/IP network. The options field of a DHCP message contains configuration parameters such as *domain-name-servers*, *domain-name*, *ntp-servers*, and *netbios-node-type*. In any Amazon VPC, you can create DHCP options sets and specify up to four DNS servers. Currently, these options sets are created and applied per VPC which means that you can't have DNS server list at the Availability Zone level. For more information about DHCP options sets and configuration, see Overview of DHCP Option Sets in the Amazon VPC Developer Guide.[1]

Elastic Network Interfaces

Elastic network interfaces (referred to as *network interfaces* in the Amazon EC2 console and this whitepaper) are virtual network interfaces that you can attach to an instance in a VPC. They're available only for instances running in a VPC. A virtual network interface, like any network adapter, is the interface that a device uses to connect to a network. Each instance in a VPC, depending on the instance type, can have multiple network interfaces attached to it. For more information, see Elastic Network Interfaces in the *Amazon EC2 User Guide for Linux Instances*.[2]

Route 53 Private Hosted Zone

A Route 53 private hosted zone is a container that holds DNS records that are visible to one or more VPCs. VPCs can be associated to the private hosted zone at the time of or after the creation of the private hosted zone. For more information, see Working with Private Hosted Zones in the *Amazon Route 53 Developer Guide*.[3]

Amazon DNS Server

The Amazon DNS Server in a VPC provides full public DNS resolution, with additional resolution for internal records for the VPC and customer-defined Route 53 private DNS records.[4] The AmazonProvidedDNS maps to a DNS server running on a reserved IP address at the base of the VPC network range, plus two. For example, the DNS Server on a 10.0.0.0/16 network is located at 10.0.0.2. For VPCs with multiple CIDR blocks, the DNS server IP address is located in the primary CIDR block.

Connection Tracking

By default, Amazon EC2 security groups use connection tracking to track information about traffic to and from the instance.[5] Security group rules are applied based on the connection state of the traffic to determine if the traffic is allowed or denied. This allows security groups to be stateful, which means that responses to inbound traffic are allowed to flow out of the instance regardless of outbound security group rules, and vice versa.

Linux Resolver

The stub resolver in Linux is responsible for initiating and sequencing DNS queries that ultimately lead to a full resolution. A resolver is configured via a configuration file, /etc/*resolv.conf*. The resolver queries the DNS server listed in the resolv.conf in the order they are listed.

Following is a sample resolv.conf:

```
options timeout:1
nameserver 10.0.0.10
nameserver 10.0.1.10
```

Linux DHCP Client

The DHCP client on Linux provides the option to customize the set of DNS servers that the instance will use for DNS resolution. The DNS servers provided in the AWS DHCP options are picked up by this DHCP client to further update the resolv.conf with a list of DNS Server IPs. In addition, you can use the *supersede* DHCP client option to replace the DNS servers provided by the AWS DHCP options set with a static list of DNS servers. You do this by modifying the DHCP client configuration file, */etc/dhcp/dhclient.conf* as shown following:

```
interface "eth0"
 {
  supersede domain-name-servers 10.0.2.10, 10.0.3.10;
 }
```

This sample supersede statement replaces DNS servers 10.0.0.10 and 10.0.1.10 in the resolv.conf sample discussed earlier with 10.0.2.10 and 10.0.3.10. We will discuss the use of this option in one of the solutions presented in this whitepaper.

Conditional Forwarder – Unbound

A conditional forwarder examines the DNS queries received from instances and forwards them to different DNS servers based on rules set in its configuration, typically using the domain name of the query to select the forwarder. In a hybrid architecture, conditional forwarders play a vital role to bridge name resolution between on-premises and cloud resources. For the solutions presented in this paper we use *unbound*, which is a recursive and caching DNS resolver, in addition to a conditional forwarder. For instructions on how to set up an unbound DNS server, see the How to Set Up DNS Resolution Between On-Premises Networks and AWS by Using Unbound blog post in the AWS Security Blog.[6] Following is a sample unbound.conf:

```
forward-zone:
        name: "."
        forward-addr: 10.0.0.2 # Amazon Provided DNS
forward-zone:
        name: "example.corp"
        forward-addr: 192.168.1.10 # On-premises DNS
```

In this sample, configuration queries to `example.corp` are forwarded to the on-premises DNS server and the rest are forwarded to the Amazon DNS server.

Constraints

In addition to the concepts established so far, it is important that you are aware of some rarely encountered constraints that are key in shaping the rest of this whitepaper and its solutions.

Packet per Second (PPS) per Network Interface Limit

Each network interface in an Amazon VPC has a hard limit of 1024 packets that it can send to the Amazon Provided DNS server every second. Therefore, a computing resource on AWS that has a network interface attached to it and is sending traffic to the Amazon DNS resolver (for example, an Amazon EC2 instance or AWS Lambda function) falls under this hard-limit restriction. In this whitepaper, we refer to this limit as packet per second (PPS) per network interface. When you're designing a scalable solution for name resolution, you need to consider this limit because failure to do so can result in queries to the

Amazon DNS server to go unanswered if the limit is breached. This limit is a key factor that we'll consider for the solutions proposed in this whitepaper.

Connection Tracking

The number of simultaneous stateful connections that an Amazon EC2 security group can support by default is an extremely large value that the majority of standard TCP-based customers never encounter any issues with. In rare cases, customers with restrictive security group policies and applications that create a large amount of concurrent connections, for instance a self-managed recursive DNS server, may run into issues of exhausting all simultaneous connection tracking resources. When that limit is exceeded subsequent connections fail silently. In such cases, we recommend that you have a security group set up that you can use to disable connection tracking. To do this, you need to set up permissive rules on both inbound and outbound connections.

Linux Resolver

The default maximum number of DNS servers that you can specify in the resolv.conf configuration file of a Linux resolver is 3, which means it isn't useful to specify 4 DNS servers in the DHCP options set because the additional DNS server won't be used. This limit further places an upper boundary on some of the solutions discussed in this whitepaper.

Solutions

The solutions in this whitepaper present options and best practices to architect a DNS solution in the hybrid cloud, keeping in mind criteria like ease of implementation, management overhead, cost, resilience, and the distribution of DNS queries directed toward the Amazon DNS server. We cover the following solutions:

- **Secondary DNS in an Amazon VPC** – This solution focuses on using Route 53 to mirror on-premises DNS zones that can then be natively resolved from within VPCs, without the need for additional DNS forwarding resources.
- **Decentralized conditional forwarders** – This solution uses distributed conditional forwarders and provides two options for using them efficiently. While we use unbound as a conditional forwarder in

some of these solutions, you can use any DNS server that supports conditional forwarding with similar features.

Secondary DNS in an Amazon VPC

DNS resolution in a hybrid architecture requires deploying and managing additional DNS infrastructure running on EC2 instances to handle DNS requests either from VPCs or on-premises, where you can use a solution based on AWS managed services instead.

This approach uses Route 53 private hosted zones with AWS Lambda and Amazon CloudWatch Events to mirror on-premises DNS zones. This can then be natively resolved from within a VPC without conditional forwarding and without a real-time dependency on on-premises DNS servers. For the full solution, see the Powering Secondary DNS in a VPC using AWS Lambda and Amazon Route 53 Private Hosted Zones on the AWS Compute blog.[7] The following table outlines this solution.

Table 1: Solution Highlights - Secondary DNS in an Amazon VPC

Use Case	Advantages	Limitations
• Customers that don't want to build or manage conditional forwarder instances • Customers that do not have in-house DevOps expertise • Infrequently changing DNS environment	• Low management overhead • Low operational cost • Highly resilient DNS infrastructure • Low possibility for instances to breach the PPS per network interface limit	• On-premises instances can't query the Amazon DNS server directly for Amazon EC2 hostnames • Works well only when on-premises DNS server records need to be replicated to Route 53 • Requires on-premises DNS server to support full zone transfer query • Requires working with the Route53 API limits

The following image illustrates how the solution works.

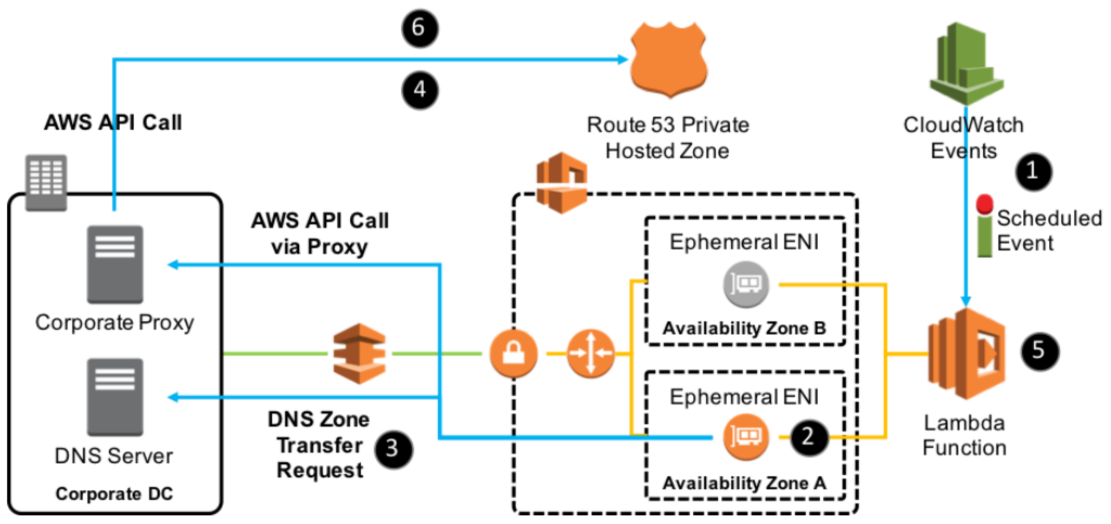

Figure 1: Secondary DNS running on Route 53 private hosted zones

	Description
❶	CloudWatch Events invokes a Lambda function. The scheduled event is configured based on a JSON string that is passed to the Lambda function that sets a number of parameters, including the DNS domain, source DNS server, and Route 53 zone ID. This configuration allows you to reuse a single Lambda function for multiple zones.
❷	A new network interface is created in the VPC's subnets and attached to the Lambda function. This allows the function to access any internal network resources based on the security group that you defined.
❸	The Lambda function transfers the source DNS zone from the IP specified in the JSON parameters. You need to configure DNS servers to allow full zone transfers, which happen over TCP and UDP port 53.
❹	The Route 53 DNS zone is retrieved using the AWS API.
❺	The two zone files are compared, and then the resulting differences are returned as a set of actions to be performed using Route 53.
❻	Updates to the Route 53 zone are made using the AWS API, and then the Start of Authority (SOA) is updated to match the source version.

There are several benefits to using this approach. Aside from the initial solution setup, there is little management overhead after the environment is set up as the solution continues working without any manual intervention. Also, there is no client-side setup because the DHCP options that you set configure each instance to use the Amazon DNS server by default.

This solution is one of the most scalable of DNS solutions in a VPC because queries for any domain go directly to the Amazon DNS server from each instance and then to the Amazon Route 53 infrastructure. This ensures that each instance uses its own PPS per network interface limit. There is also no correlation and impact of implementing this solution in one or more VPCs as you choose to associate the Route53 hosted zone with multiple VPCs. The possibility of failure of a DNS component is lower because of the highly available and reliable Amazon Route 53 infrastructure.

The main disadvantage of this solution is that it requires full zone transfer query (AXFR) so it isn't appropriate for customers that run DNS servers that don't support AXFR. Also, because this solution involves working with the Route 53 APIs, you must stay within the Route 53 API limits.[8] It is also worth noting that this solution does not provide a method for resolving EC2 records from on premises directly.

Decentralized Conditional Forwarders

Although the Route 53 solution enables you to avoid the complexities in running a hybrid DNS architecture, you might prefer to configure your DNS infrastructure to use conditional forwarders in your VPCs because you can forward queries meant for on-premises resources to the on-premises DNS server, and then forward the rest of the queries to the Amazon DNS server. For reasons mentioned previously in the Linux Resolver section, we explore three conditional forwarders in these solutions that are configured using the Amazon VPC DHCP options set.

There are two options under this solution. The first option, called *highly distributed forwarders* discusses how to run forwarders on every instance of the environment trying to mimic the scale that the Route53 solution provides. The second option, called *zonal forwarders using supersede* presents a strategy of localizing forwarders to a specific Availability Zone and its instances. The following table highlights these two options followed by their detailed discussion.

Table 2: Solution highlights – Decentralized conditional forwarders

Option	Use Case	Advantages	Limitations
Highly Distributed Forwarders	• Workload generates high volumes of DNS queries • Infrequently changing DNS environment	• Resilient DNS infrastructure • Low possibility for instances to breach the PPS per network interface limit	• Complex setup and management • Investment in relevant skill sets for configuration management
Zonal Forwarders using Supersede	• Customers with existing set of conditional forwarders • Environment that doesn't generate a high volume of DNS traffic	• Fewer forwarders to manage • Zonal isolation provides better overall resiliency	• Complex setup and management as the DNS environment grows • Possibility of breaching the PPS per network interfaces limit is higher than the highly distributed option

Highly Distributed Forwarders

This option decentralizes forwarders and runs a small lightweight DNS forwarder on every instance in the environment. The forwarder is configured to serve the DNS needs of only the instance it is running on, which eliminates bottlenecks and dependency on a central set of instances.

Given the implementation and management complexity of this solution we recommend that you use a mature configuration management solution. The following diagram shows how this solution functions in a single VPC:

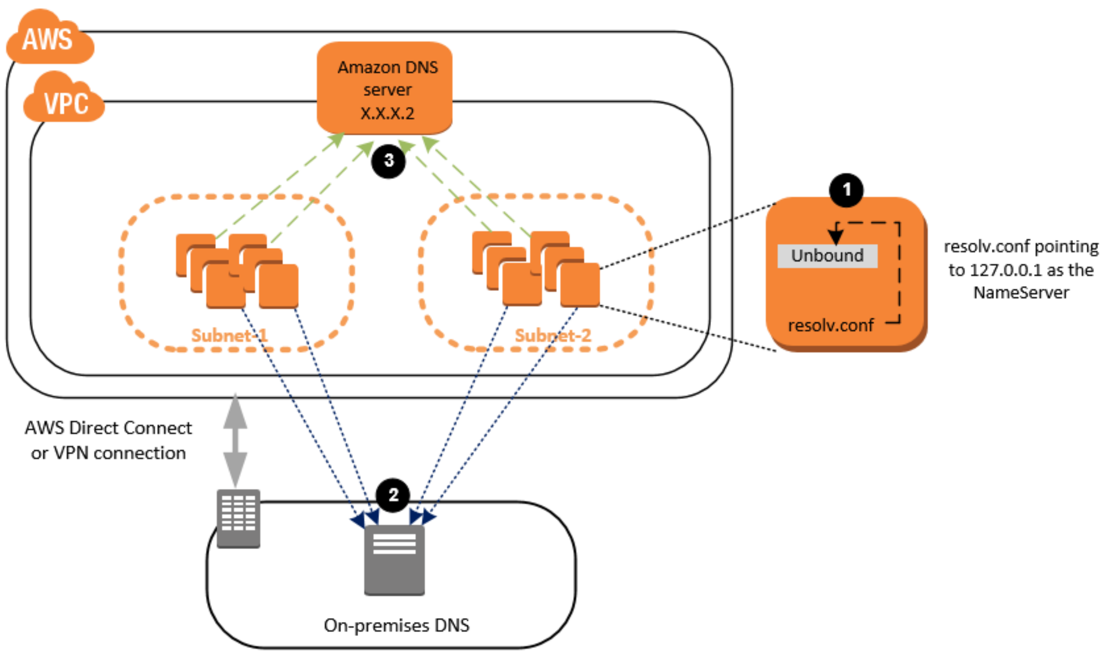
AWS
VPC
Amazon DNS server X.X.X.2
3
1
Unbound
resolv.conf
resolv.conf pointing to 127.0.0.1 as the NameServer
Subnet-1
Subnet-2
AWS Direct Connect or VPN connection
2
On-premises DNS

Figure 2: Distributed forwarders in a single VPC

	Description
❶	Each instance in the VPC runs its own conditional forwarder (unbound). The resolv.conf has a single DNS Server entry pointing to 127.0.0.1. A straightforward approach for modifying resolv.conf would be by creating DHCP options set that has 127.0.0.1 as the *domain-name-server* value. You may alternatively choose to overwrite any existing DHCP options settings using the supersede option in the dhclient.conf.
❷	Records requested for on-premises hosted zones are forwarded to the on-premises DNS server by the forwarder running locally on the instance.
❸	Any requests that don't match the on-premises forwarding filters are forwarded to the Amazon DNS server.

Similar to the Route 53 solution, this solution allows every single instance to use the limit of 1024 PPS per network interfaces to the Amazon DNS server to its full potential. The solution also scales up as additional instances are added and works the same way regardless of whether you're using a single or multi-VPC setup. The DNS infrastructure is low latency and the failure of a DNS component such as an individual forwarder does not affect the entire fleet due to the decoupled nature of the design.

This solution poses implementation and management complexities, especially as the environment grows. You can manage and modify configuration files at instance launch using Amazon EC2 user data.[9] After instance launch, you can use the Amazon EC2 Run Command[10] or AWS OpsWorks for Chef Automate[11] to deploy and maintain your configuration files. The implementation of these solutions is outside the scope of this whitepaper, but it is important to know that they provide the flexibility and power to manage configuration files and their state at a large scale. Greater flexibility brings with it the challenge of greater complexity. You need to consider additional operational costs, including the need to have an in-house DevOps workforce.

Zonal Forwarders Using Supersede

If you don't want to manage and implement a forwarder on each instance of your environment and you want to have conditional forwarder instances as the center piece of your hybrid DNS architecture, you should consider this option.

For this option, you localize instances in an Availability Zone to forward queries to conditional forwarders only in the same Availability Zone of the Amazon VPC. For reasons discussed in the Linux Resolver section each instance can have up to three DNS servers in their resolv.conf, as shown in the following diagram.

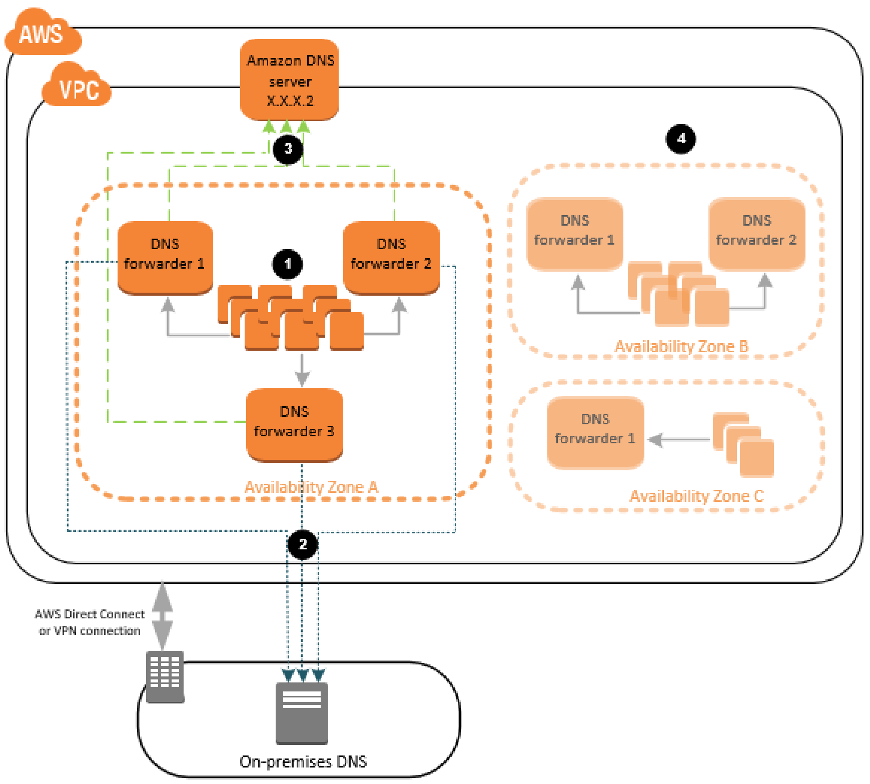

Figure 3: Zonal forwarders with supersede option

	Description
❶	Instances in Availability Zone A are configured using the supersede option, which uses a list of DNS forwarders that are local to that Availability Zone. To avoid burdening any specific forwarder in the Availability Zone, randomize the order for the DNS forwarders across instances in the Availability Zone.
❷	Records requested for on-premises hosted zones are directly forwarded to the on-premises DNS server by the DNS forwarder.
❸	Any requests that don't match the on-premises forwarding filters are forwarded to the Amazon DNS server. This illustration doesn't depict the actual flow of traffic. It's presented for representation purposes only.
❹	Similarly, other Availability Zones in the VPC can be set up to use their own set of local conditional forwarders that serve the respective Availability Zone. You determine the number of conditional forwarders serving an Availability Zone based on your need and the importance of the environment.

If one of the three instances in Availability Zone A fails, the other two instances will continue serving DNS traffic. It is important to note that there is no way to guarantee that the forwarders are not running on the same parent hardware, which is a single point of failure. To ensure separate parent hardware, you may setup and take advantage of Dedicated Hosts, which lets you run your EC2 instances on dedicated physical server.[12]

If all three DNS forwarders in Availability Zone A fail at the same time, the instances in Availability Zone A will fail to resolve any DNS requests because they are unaware of the presence of forwarders in other Availability Zones. This prevents the impact from spreading to multiple Availability Zones and ensures that other Availability Zones continue to function normally.

Currently, the DHCP options that you set apply to the VPC as a whole. Therefore, you must self-manage the list of DNS servers that are local to instances in each Availability Zone. In addition, we recommend that you don't use the exact same order of DNS servers in your resolv.conf for all instances in the Availability Zone because it would burden the first server in the list and push it closer to breaching the PPS per network interfaces limit. While each Linux instance can only have three resolvers, if you're managing the resolver list yourself, you can have as many resolvers as you wish per Availability Zone. Each instance should be configured with 3 random resolvers from the resolver list.

Selecting the Best Solution for Your Organization

There are various advantages and trade-offs with each of these solutions. Choosing the right solution for your organization depends on the specific needs of the workload. You might choose to run different solutions in different VPCs to meet the needs of your specific workloads. The following table summarizes the criteria that you can use to evaluate what will work best for your organization. These include the complexity of the implementation, the management overhead, the availability of the solution, probability of hitting the PPS per network interface limit, and the cost of the solution.

Table 4: Solutions selection criteria

	Secondary DNS in a VPC	Highly Distributed Forwarders	Zonal Forwarders
Implementation complexity	Medium	High	High
Management overhead	Low	High	Medium
DNS infrastructure resiliency	High	High	Medium
PPS limit breach	Low	Low	Medium
Cost*	Low	High	Medium

***** Cost is a combination of the infrastructure and operational expense.

Additional Considerations

While the solutions discussed so far work well for most customers, there are additional scenarios and optimizations that are worth exploring depending on your requirements.

Custom EC2 DNS Resolver

You can choose to host your own custom DNS resolver on Amazon EC2 that leverages public DNS Servers to perform recursive public DNS resolution instead of using the Amazon DNS Server. The inclination to do so could be due to the nature of the application and the ability to have more control and flexibility over the DNS environment. You could also think of doing so if the PPS per network interface limit is a hindrance in your ability to scale and none of the solutions discussed thus far suit your needs.

Although, we will not get into the details of architecting such a solution, we wanted to point out some caveats that will help you plan better in such a scenario. The following diagram illustrates an approach of a hybrid VPC DNS setup where you have your own DNS resolver on Amazon EC2.

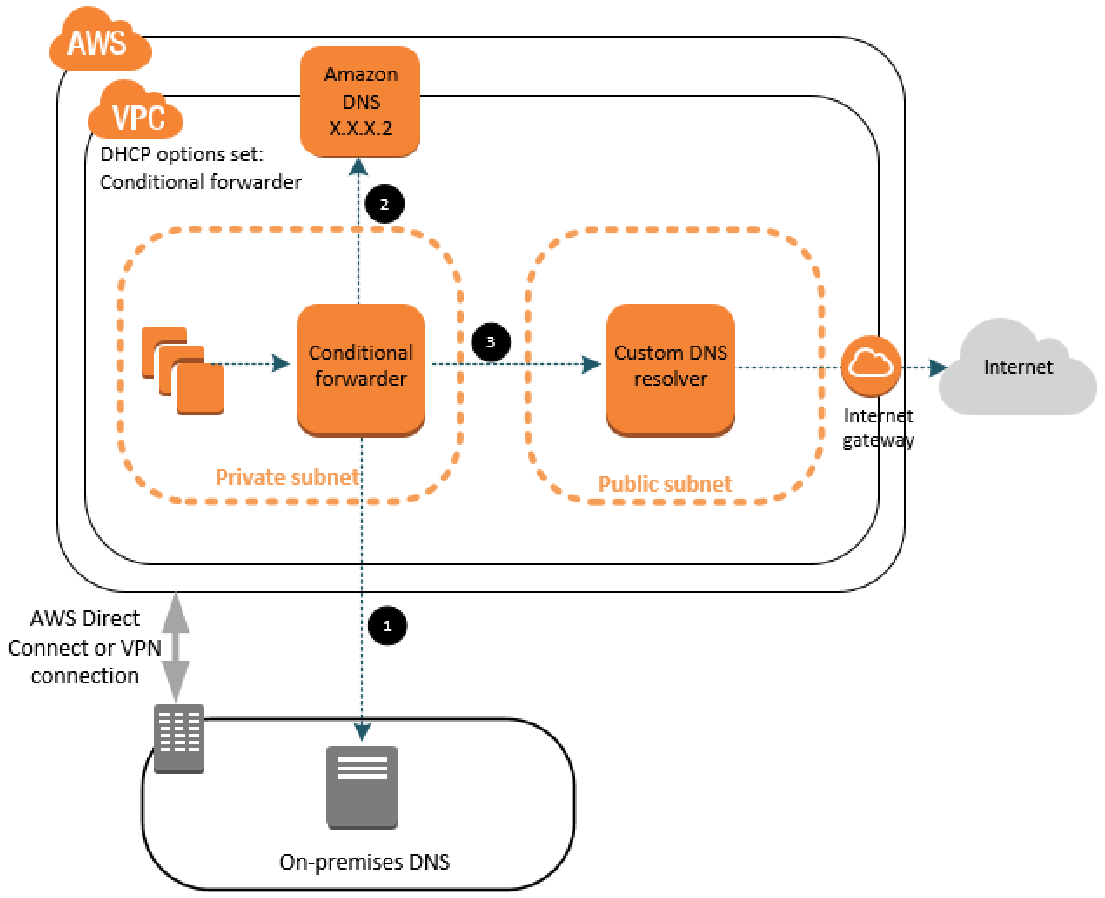

Figure 4: Amazon EC2 DNS instances with segregated resolver and forwarder

	Description
❶	DNS queries for internal EC2 names and Route 53 private hosted zones are forwarded to the Amazon DNS Server
❷	DNS queries bound for on-premises servers are conditionally forwarded to on-premises DNS servers.
❸	DNS queries for public domains are conditionally forwarded to the custom DNS resolver in the public subnet. The resolver then recursively resolves public domains using the latest root hints available from the Internet Assigned Number Authority (IANA).

For security reasons, it is recommended that the Conditional forwarder instance that requires connectivity to on-premises sits separately in a private subnet of the VPC. As the custom DNS resolver needs to be able to query public DNS Servers, it runs in its own public subnet of the VPC.

Ideally, you would have security group rules on the EC2 instance running the custom DNS resolver but if this custom DNS resolver has extremely high rates of querying out to the internet, then there is a possibility that you will hit connection tracking limits as discussed in the connection tracking section. Therefore, to avoid running into such a scenario, connection tracking by itself needs to be avoided and it is possible to do so by opening up all ports, TCP and UDP, to the whole world at the security group level, both inbound and outbound.

As this is granting permissive rules to instance level security group, you will have to handle the security of the instance at a different layer. At the least, it is recommended to control the traffic entering into the entire public subnet by using Network Access Control Lists (NACL), which thereby restricts access to the instance or you could use application level control mechanisms like *access-control* provided by a DNS resolver like unbound.[13]

Custom DNS resolvers might develop a reputation upstream on the internet. If the instance is assigned a dynamic public IP that belonged to another customer and previously earned a bad reputation, requests upstream could be throttled or even blocked. To avoid being throttled or blocked, consider assigning elastic IP addresses to these resolver instances. This provides these IP addresses that talk to the upstream servers with the opportunity to build a good reputation over time that can be owned and maintained.

Microsoft Windows Instances

Typically, Microsoft Windows instances are joined using Active Directory Domain Services (AD DS). In scenarios where you use the Amazon VPC DHCP options set, unlike the Linux resolver, you can set the full set of four DNS servers. You can set the DNS servers independently from the DHCP supplied IP address similar to the supersede option discussed earlier. This can be accomplished using Active Directory Group Policy or via configuration management tools such as Amazon EC2 Run Command[14] or AWS OpsWorks for

Chef Automate[15] mentioned earlier. In addition, the Windows DNS client also enables you to cache recently resolved queries, which reduces the overall demand on the primary DNS server.

The Windows DNS client service is designed to prompt a dynamic update from the DNS server if a change is made to its IP address information. When prompted, the DNS server updates the host record IP address for that computer (according to RFC 2136). Microsoft DNS provides support for dynamic updates and this is enabled by default in any Active Directory integrated DNS zone. When you use a lightweight forwarder like unbound for Windows instances, note that there isn't any support for these dynamic updates and it can't support RFC 2126. If you want to do this, you should use the Microsoft DNS server as a primary for these instances.

Unbound – Additional Options

Unbound caches the results for subsequent queries until the time to live (TTL) expires, after which it forwards the request. By enabling the *prefetch* option in unbound, you can ensure that frequently used records are pre-fetched before they expire to keep the cache up to date. Also, if the on-premises DNS server is not available when the cache expires, unbound returns SERVFAIL. To protect yourself against such a situation, you can enable the serve-expired option to serve old responses from the cache with a TTL of zero in the response without waiting for the actual resolution to finish. After the resolution is completed the response is cached for subsequent use.

DNS Forwarder – Forward First

Some DNS servers (notably BIND) includes a *forward first* option enabled by default, which causes the server to query the forwarder first and, if there is no response, to recursively retry the internet DNS servers. For private DNS domains in this scenario, the internet DNS servers return an authoritative NXDOMAIN, which is a non-existent Internet or Intranet domain name, or they return the public address if you're using split horizon DNS for public zones, which is used to provided different answers for private vs public IP addresses. Therefore, it is critical to specify the forward only option which specifies that retries are made against the forwarders, which means that you avoid ever seeing the response from public nameservers. The unbound DNS server has the forward first option disabled by default.

DNS Server Resiliency

The solutions in this whitepaper are intended to provide high availability in the event that there is an issue with your primary DNS server. However, there are factors can prevent or delay this failover from occurring. These factors include, but are not limited to, the timeout value in resolv.conf, configuration issues with the superseded DNS, or incorrect DHCP options set settings. In some cases, these factors could impact the availability of applications that are dependent on name resolution. There are a few simple approaches to ensure the resilience of your forwarders in case there is an issue with the underlying hardware or instance software. While these approaches don't eliminate the need for well-architected design, they can help you increase the overall resiliency of your solution.

EC2 Instance Recovery

In the case of an underlying hardware failure of a DNS forwarder instance, you can use EC2 instance recovery to start the instance on a new host. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata. To do this, you can create a CloudWatch alarm that monitors an EC2 instance and automatically recovers the instance if it becomes impaired. You can use the CloudWatch alarm to monitor issues like loss of network connectivity, loss of system power, software issues on the physical host, or hardware issues on the physical host that affect network reachability. For more information about instance recovery, see Recover Your Instance in the *Amazon EC2 User Guide for Linux Instances*.[16] For step-by-step instructions on using CloudWatch alarms to recover an instance, see Create Alarms That Stop, Terminate, Reboot or Recover an instance in the *Amazon EC2 User Guide for Linux Instances*.[17]

Secondary IP Address

In an Amazon VPC, instances can be assigned secondary IPs which are transferrable. If an instance fails, the secondary IP can be transferred to a standby instance and this avoids the need for every instance to reconfigure their resolver IPs. This approach redirects traffic to the healthy instance so that it can respond to DNS queries. This approach is appropriate for scenarios where EC2 instance recovery might not provide fast enough recovery or might not be appropriate (for example, an operating system fault or software issue). For more information about working with multiple IP addresses, see Multiple IP Addresses in the *Amazon EC2 User Guide for Linux Instances*.[18]

Conclusion

For organizations with on-premises resources, operating in a hybrid architecture is a necessary part of the cloud adoption process. As such architecture patterns that streamline this transition are essential for success. We discussed concepts as well as constraints to help you obtain a better understanding of the fundamental building blocks of the solutions provided here, as well as the limitations that help to create the most optimal solution for your workload. The solutions that were provided, included how to setup Secondary DNS in the Amazon VPC with Amazon Lambda and Route 53 Private hosted zones, and solutions leveraging Decentralized forwarders using the Unbound DNS server. We also provided guidance on how to select the appropriate solution for your intended workload. Finally, we examined some additional considerations to help you to better tailor your solution for different workload requirements, faster failover and better DNS server resiliency. By using the architectures provided you can achieve the most ideal private DNS interoperability between your on-premises environments and your Amazon VPC.

Contributors

The following individuals contributed to this document:

- Gavin McCullagh, Systems Development Engineer
- Sohamn Chaterjee, Cloud Infrastructure Architect
- Harsha Warrdhan Sharma, Technical Account Manager
- Anthony Galleno, Senior Technical Account Manager

Document Revisions

Date	Description
October 2017	First publication

Notes

1 http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_DHCP_Options.html#DHCPOptionSets

2 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html

3 http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zones-private.html

4 http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_DHCP_Options.html#AmazonDNS

5 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html#security-group-connection-tracking

6 https://aws.amazon.com/blogs/security/how-to-set-up-dns-resolution-between-on-premises-networks-and-aws-by-using-unbound/

7 https://aws.amazon.com/blogs/compute/powering-secondary-dns-in-a-vpc-using-aws-lambda-and-amazon-route-53-private-hosted-zones/

8 http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/DNSLimitations.html#limits-api-requests

9 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

10 https://aws.amazon.com/ec2/run-command/

11 https://aws.amazon.com/opsworks/chefautomate/

12 https://aws.amazon.com/ec2/dedicated-hosts/

13 http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html

14 https://aws.amazon.com/ec2/run-command/

15 https://aws.amazon.com/opsworks/chefautomate/

16 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-recover.html

[17] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/UsingAlarmActions.html

[18] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/MultipleIP.html