Citrix XenApp on AWS: Implementation Guide

citrix.com
# Table of Contents

Abstract .......................................................................................................................... 5
Introduction ....................................................................................................................... 5
Implementing XenApp 6.5 Architecture Scenarios in AWS ............................................. 6
Step 1: Sign up for an AWS Account ............................................................................. 8
Step 2: Launch the virtual Network and Active Directory infrastructure ....................... 9
  Setting Up Amazon VPC for the XenApp Cloud Hosted Farm Model ......................... 9
    Using XenApp RA Template ..................................................................................... 10
    VPC and Subnet Setup ............................................................................................. 11
    Private and Public Routes ....................................................................................... 12
    Setting up Secure Administrative Access using Bastion Hosts ............................... 15
    Securing our Infrastructure with Security Groups ................................................ 15
  Setting Up CloudBridge for the XenApp Cloud Hosted Farm Model .......................... 17
    Configuration Steps ............................................................................................... 19
  Setting Up the NetScaler Appliance in the Datacenter .............................................. 19
  Setting up the CloudBridge Appliance in the Datacenter ......................................... 21
    Configuring the Network at the Datacenter ............................................................ 21
    Configuring the TCP MSS and Virtual Inline Settings on the CloudBridge .......... 21
    Deploying NetScaler and Branch Repeater Appliances in AWS using CloudFormation ..... 23
    Using XenApp RA CloudBridge Template ........................................................... 23
    NetScaler HA Pairs ............................................................................................... 26
    CloudBridge VPX Instances .................................................................................. 30
  Accessing the NetScaler Instance on AWS .............................................................. 30
  Accessing the CloudBridge Instance on AWS ......................................................... 31
  CloudBridge Connector ......................................................................................... 33
    Configuring CloudBridge Connector between Datacenter and AWS Cloud .......... 33
      Redirecting traffic to the CloudBridge VPX Appliances................................. 39
      To configure the load balancing of CloudBridge appliances by using the NetScaler configuration utility .................................................................................. 39
    Configuring Listen Policies for the Virtual Servers Created by the Wizard .......... 43
  Launching the Active Directory in the Right Order .................................................... 44
    AD DS Setup and DNS Configuration .................................................................... 45
  Setting up and configuring Windows Server for AD DS ............................................ 46
    Using XenApp RA Active Directory Template .................................................... 46
Citrix XenApp on AWS: Implementation Guide

DNS and DHCP Setup .......................................................... 51

Step 3: Launch the Database Layer ........................................... 54

Creating a SQL Server 2008 R2 AMI with EC2Config service and AWS CloudFormation
Helper Scripts ................................................................. 54

Using SQL Server Template .................................................. 55

Template Customization ...................................................... 55

Provisioning our SQL Server Instances using AWS CloudFormation and Windows PowerShell .............. 56

Provisioning EBS Volumes ..................................................... 56

Provision and Configure SQL Server for Use with XenApp 6.5 .................................................. 59

Changing the default database location ...................................... 59

Adding XenApp Logins to Database roles .................................. 60

Opening SQL Server Ports in the Windows Firewall ....................... 61

Configuring SQL Server for High Availability ............................ 62

Step 4: Launch the XenApp Worker server layer ...................... 64

Citrix Service Provider Automation Pack scripts ...................... 64

Install Server ........................................................................ 64

CloudFormation Script Method ............................................... 64

Template Customization ...................................................... 65

Building the XenApp Farm with Citrix App Orchestration ............... 69

AWS CloudFormation Template with MSI tools ......................... 72

Template Customization ...................................................... 72

Licensing Configuration ....................................................... 77

Step 5: Launch the StoreFront Layer ....................................... 79

StoreFront Installation and Configuration .................................. 79

Step 6: Launch and Configure the Desktop Layer ....................... 80

Adding and Removing Farm Capacity ....................................... 81

Change Capacity via Service Provider Automation Pack ................ 81

Change Capacity via automated MSI installation of XenApp ................ 82

Template Customization ...................................................... 82

Change Capacity via Golden XenApp Worker AMIs ...................... 84

Automating adding XenApp Workers to a XenApp Farm and Worker Group .................................. 89

Template Customization ...................................................... 89

Provisioning the XenApp Server using AWS CloudFormation and Windows PowerShell .................. 90

Configuring XenApp XML Server and StoreFront Loadbalancing ............... 91
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure NetScaler to Load Balance Storefront</td>
<td>92</td>
</tr>
<tr>
<td>Configuring StoreFront server instances using the StoreFront configuration Wizard</td>
<td>95</td>
</tr>
<tr>
<td>Configuring external access using the NetScaler Gateway Wizard</td>
<td>103</td>
</tr>
<tr>
<td>Dynamic Capacity Management</td>
<td>109</td>
</tr>
<tr>
<td>Auto Scaling</td>
<td>110</td>
</tr>
<tr>
<td>Amazon CloudWatch</td>
<td>110</td>
</tr>
<tr>
<td>Provision new XenApp workers ready for Dynamic Provisioning</td>
<td>111</td>
</tr>
<tr>
<td>Set Up Task Scheduler to Send Metrics Reports to Amazon CloudWatch</td>
<td>113</td>
</tr>
<tr>
<td>Scaling down – Using XenApp PCM</td>
<td>115</td>
</tr>
<tr>
<td>XenApp Farm in AWS Deployment</td>
<td>118</td>
</tr>
<tr>
<td>Simulated environment</td>
<td>118</td>
</tr>
<tr>
<td>XenApp Configuration</td>
<td>120</td>
</tr>
<tr>
<td>Data store</td>
<td>120</td>
</tr>
<tr>
<td> How much storage is needed for the data store?</td>
<td>121</td>
</tr>
<tr>
<td> What type of database should be used to host the data store?</td>
<td>121</td>
</tr>
<tr>
<td> What type of instances are needed to support 1,000 servers?</td>
<td>122</td>
</tr>
<tr>
<td> Optimizing your data store performance with Session-Host mode</td>
<td>122</td>
</tr>
<tr>
<td> Servers running in Session-Host mode</td>
<td>123</td>
</tr>
<tr>
<td> How should the data store be deployed?</td>
<td>124</td>
</tr>
<tr>
<td>Data collector</td>
<td>124</td>
</tr>
<tr>
<td> How many zones are needed to support 1,000 users?</td>
<td>125</td>
</tr>
<tr>
<td> Is a backup data collector needed for this farm?</td>
<td>128</td>
</tr>
<tr>
<td> What type of instance type is recommended for the data collector?</td>
<td>129</td>
</tr>
<tr>
<td>XML server</td>
<td>131</td>
</tr>
<tr>
<td> Is a dedicated instance needed for the XML server?</td>
<td>131</td>
</tr>
<tr>
<td>License server</td>
<td>131</td>
</tr>
<tr>
<td> How many license servers are necessary?</td>
<td>132</td>
</tr>
<tr>
<td> Where in the farm should the license server be located?</td>
<td>132</td>
</tr>
<tr>
<td> What type of instance type is recommended for the license server?</td>
<td>132</td>
</tr>
<tr>
<td> How much bandwidth is used during license consumption?</td>
<td>133</td>
</tr>
<tr>
<td> Is the license server a single point of failure?</td>
<td>133</td>
</tr>
<tr>
<td>Citrix AppCenter Console</td>
<td>135</td>
</tr>
<tr>
<td> How should the farm be managed from remote locations?</td>
<td>135</td>
</tr>
<tr>
<td> Can the AppCenter Console accommodate a hybrid administration model?</td>
<td>136</td>
</tr>
<tr>
<td> How many StoreFront servers are needed to support 5,000 users?</td>
<td>136</td>
</tr>
</tbody>
</table>
Worker group configuration .............................................................................................. 136
Application publishing.................................................................................................. 140
Load balancing based on worker groups .................................................................... 142
Citrix policy filters....................................................................................................... 145
Citrix policy configuration ............................................................................................ 145
Policy refresh interval.................................................................................................. 146
Further Reading............................................................................................................. 148
CloudFormation Templates............................................................................................ 151
Abstract

Amazon Web Services (AWS) provides a comprehensive set of services and tools for deploying Microsoft Windows workloads and Citrix NetScaler VPX technology, making it a perfect fit for deploying or extending a Citrix XenApp farm on a highly reliable and secure cloud infrastructure platform.

Deploying an enterprise-class XenApp solution that involves multiple components can be resource and time-consuming. To help simplify the deployment process, Citrix published a series of blogs - Deploying a XenApp Farm on AWS with CloudFormation - that included AWS CloudFormation sample templates that enable the launch of a fully functional XenApp 6.5 server farm on AWS. This Implementation Guide builds on the blog series and is intended to be a companion piece to the Citrix XenApp on AWS Reference Architecture white paper. This guide targets IT infrastructure administrators and DevOps personnel. It will provide the reader with a good understanding of how to deploy and configure a XenApp 6.5 server farm on AWS repeatedly and reliably through the use of scripting.

Introduction

XenApp is an application delivery solution that enables any Windows © application to be virtualized, centralized and managed in the datacenter and instantly delivered as a service to users anywhere on a device, empowering users with on-demand self-service to enterprise applications. AWS not only provides the on-demand resources (compute, database, network, and storage) needed to run this solution but also provides a way to script the provisioning and configuration steps for easy deployment. AWS CloudFormation enables you to create and manage AWS infrastructure in a predictable and repeatable manner. CloudFormation templates assist in the deployment of AWS services such as Amazon Elastic Compute Cloud (EC2), Amazon Elastic Block Store (EBS), and Auto Scaling groups to build reliable, scalable, and cost-efficient applications. In addition, Windows PowerShell scripts can be used for a more detailed configuration of the Windows-based Amazon EC2 instances. The Windows PowerShell scripts provided with this Implementation Guide provide limited functionality and are not meant to represent a final solution. The scripts are built from samples freely available on the Windows PowerShell community sites and are meant to show how AWS CloudFormation templates and Windows PowerShell scripts can be used together to reach deep into instances at provisioning time and perform the necessary configuration steps. Consider replacing these scripts with your own.

NOTE: The scenario discussed in this guide is that of a XenApp Cloud Hosted Farm deployment. The accompanying templates, scripts, and methods discussed in this guide serve as a starting point that the reader will later modify or extend. This document does not discuss scripted configuration of XenApp 6.5, as there are no two XenApp configurations alike. However, Step 6 provides a set of detailed instructions that will help configure a dynamic scaling XenApp 6.5 scenario with full functionality for either a demonstration or proof of concept (POC).
Implementing XenApp 6.5 Architecture Scenarios in AWS

This advanced Implementation Guide provides a walkthrough of the sample templates and describes the AWS-specific implementation details that can be customized to meet business, IT, and security requirements. This guide follows the outline presented in the Citrix XenApp on AWS: Reference Architecture white paper so that the reader can follow along while launching the sample templates. Chapter 2 and Chapter 3 are useful as a general reference for how to deploy Windows-based infrastructure components such as Microsoft Active Directory and Microsoft SQL Server in the AWS cloud.

This guide discusses the following topics:

- **Step 1: Sign up for an AWS Account**

- **Step 2: Launch the virtual Network and Active Directory infrastructure. This includes:**
  
  - Setting up the virtual network for the multi-layered XenApp 6.5 server farm within AWS, including subnets in two Availability Zones to support logical server groups for different layers and roles within the XenApp reference architecture.
  
  - Setting up the Bridge Layer that will use a NetScaler VPX HA pair per AWS Availability Zone with supporting CloudBridge optimization to create transparent access between the on-premises resources and our XenApp Cloud Hosted Farm.
  
  - Deploying Active Directory to provide authentication and DNS services for the XenApp server farm.
  
  - Configuring Windows Server instances as bastion hosts to enable secure administrative access, and deploying NAT instances to enable secure communication (e.g., to obtain security and general updates from Windows Update).
  
  - Implementing security mechanisms in AWS, including how to configure instance and network security to enable authorized access to the overall XenApp server farm as well as access between layers and instances within the farm.

- **Step 3: Launch the Database layer. This includes:**
  
  - Creating an AWS CloudFormation-enabled SQL Server 2008 R2 Standard Edition Amazon Machine Image (AMI) to enable scripted configuration of the SQL Server components of the farm.
  
  - Joining the SQL Server instance to the domain, enabling Named Pipes and starting the SQL Browser Service.
Citrix XenApp on AWS: Implementation Guide

- Provisioning of XenApp farm administrator logins and dbcreator and securityadmin roles using Windows PowerShell.

- **Step 4: Launch the Control layer.** This includes:
  
  o Creating an extended Windows Server AMI that will function as the install server and also holds the raw XenApp bits and the Service Provider Pack Windows PowerShell modules to reduce deployment time.
  
  o Create the XenApp farm infrastructure installing the XenApp Zone Data Collectors using the XenApp deployment Windows PowerShell scripts.

- **Step 5: Launch the Web and Access layer.** This includes:
  
  o Installing StoreFront servers (one per Availability Zone) to enable load-balanced on-demand self-service to enterprise applications delivered via XenApp.
  
  o Deploying NetScaler Gateway in front of the StoreFront servers.

- **Step 6: Launch and Configure the Desktop Layer.** This includes:
  
  o Creating XenApp golden images.
  
  o Creating XenApp worker servers based on those golden images using one of two methods:
    
    - Using the Cloud Provider Pack Windows PowerShell modules to reduce deployment time.
    
    - Using AWS Cloud Formation scripts.
  
  o Configuring workloads provided by the worker servers using the XenApp AppCenter Wizard.
  
  o Configuring StoreFront server instances using the StoreFront configuration Wizard.
  
  o Configuring external access using the NetScaler Gateway Wizard.
  
  o Testing the XenApp 6.5 deployment and demonstrating the facilities of the virtual desktops and apps.
  
  o Configuration of dynamic scaling and power management of a XenApp 6.5 on AWS farm.
When complete, the XenApp server farm implements the following scenario:

**Step 1: Sign up for an AWS Account**

If you already have an AWS account, skip to the next step. If you don't already have an AWS account, use the following procedure to create one.

To create an AWS account, go to [http://aws.amazon.com](http://aws.amazon.com), and click Sign Up Now. Follow the on-screen instructions. Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.

When you create an AWS account, AWS automatically signs up the account for all AWS services, including Amazon EC2. You are charged only for the services that you use.

As a first step create a Key Pair in the region(s) where you want to deploy your XenApp Infrastructure. This Key Pair will allow you afterwards to connect securely to your launched instances as part of this deployment.
Step 2: Launch the virtual Network and Active Directory infrastructure

Let’s start with the necessary infrastructure and virtual network setup to provide the environment in which you instantiate and configure your servers and database.

The Citrix XenApp on AWS Reference Architecture white paper is organized around a multi-layered (Access, Bridge, Web, Control, Active Directory, Database and Desktop) approach, allowing you to independently scale and configure each layer. Your first task is to define a virtual network environment that supports this type of layered structure and enables you to deploy the various server roles in each layer with suitable security configuration.

Note: The outlined models of the Citrix XenApp on AWS Reference Architecture are deployed into an Amazon Virtual Private Cloud (Amazon VPC). Amazon VPC lets you provision a private, isolated section of the AWS cloud where you can launch AWS resources in a virtual network that you define. With Amazon VPC, you can define a virtual network topology closely resembling a traditional network that you might operate in your own datacenter. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

Setting Up Amazon VPC for the XenApp Cloud Hosted Farm Model

For the XenApp Cloud Hosted Farm model, we are accommodating the following requirements:

We want to launch the Web, Control, Active Directory, Database and Desktop layers in private subnets; users only need to get to the NetScaler Gateways (which are configured in Step 6 after the StoreFront instances are created).

It is advisable for the Cloud Hosted Farm model to enable the essential datacenter security capabilities of the NetScaler Gateway for firewall and threat management. It is out of scope for this article and the accompanying artefacts. However, you might want to consider implementing these features after all the scripts have been run.

We add NAT instances in each Availability Zone to facilitate servers in private subnets communicating out to the Internet (to receive operating system software updates, for example).
Given the preceding requirements, Figure 2 shows the network setup and administrative access for the public website scenario:

Figure 2: Network configuration and administrative access for the XenApp Cloud Hosted Farm Model

Using XenApp RA Template

Open up the sample XenApp RA AWS CloudFormation template file and follow along.

Template Customization

The sample XenApp RA AWS CloudFormation (XA-VPC) Template allows for rich customization of defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. AWS CloudFormation currently supports a maximum of 30 parameters per template. The XA-VPC Template parameters include the following default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>NATInstanceType</td>
<td>m1.small</td>
<td>Amazon EC2 instance type for the NAT instance.</td>
</tr>
<tr>
<td>BastionInstanceType</td>
<td>m1.small</td>
<td>Amazon EC2 instance type for the Bastion Host.</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td>AZ1</td>
<td>us-east-1a</td>
<td>Name of Availability Zone that will contain public and private subnets; select a valid</td>
</tr>
</tbody>
</table>
VPC and Subnet Setup

Creating a VPC using AWS CloudFormation requires only a few lines of code in the Resources section of your template. This launches a resource of the type `AWS::EC2::VPC`.

```
"VPC": {
  "Type": "AWS::EC2::VPC",
  "Properties": {
    "CidrBlock": { "Ref": "VPCCIDR" },
    "Tags": [
      { "Key": "Application", "Value": { "Ref": "AWS::StackName" } },
      { "Key": "Network", "Value": "Public" }
    ]
  }
},
```

As in Figure 2, we want to give the users of our templates control over the definition of the CIDR block for the VPC. To do so, we need to declare a parameter in the Parameters section of our template that we can then reference { "Ref": "VPCCIDR" } when creating the VPC resource itself or resources associated with this VPC. This parameter definition is as follows:

```
"VPCCIDR": {
  "Description": "CIDR Block for the VPC",
  "Type": "String",
  "Default": "10.16.0.0/16",
},
```

Next, we create the six private subnets by following a similar pattern as we used for creating the VPC. First, we declare a resource of the type `AWS::EC2::Subnet`:

```
"Infra1Subnet": {
  "Type": "AWS::EC2::Subnet",
  "Properties": {
    "VpcId": { "Ref": "VPC" },
    "CidrBlock": { "Ref": "Infra1CIDR" },
    "AvailabilityZone": { "Ref": "AZ1" },
    "Tags": [
      { "Key": "Application", "Value": { "Ref": "AWS::StackName" } },
      { "Key": "Network", "Value": "Private" }
    ]
  }
},
```

---

<table>
<thead>
<tr>
<th>AZ2</th>
<th>us-east-1b</th>
<th>Name of Availability Zone that will contain public and private subnets; select a valid zone for your region.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMZ1CIDR</td>
<td>10.16.9.0/24</td>
<td>CIDR Block for the Public DMZ subnet 1 located in AZ1</td>
</tr>
<tr>
<td>DMZ2CIDR</td>
<td>10.16.10.0/24</td>
<td>CIDR Block for the Public DMZ subnet 2 located in AZ2</td>
</tr>
<tr>
<td>NS1CIDR</td>
<td>10.16.7.0/24</td>
<td>CIDR Block for the Public NetScaler subnet 1 located in AZ1</td>
</tr>
<tr>
<td>NS2CIDR</td>
<td>10.16.8.0/24</td>
<td>CIDR Block for the Public NetScaler subnet 2 located in AZ2</td>
</tr>
<tr>
<td>Web1CIDR</td>
<td>10.16.0.0/24</td>
<td>CIDR Block for Web/StoreFront Subnet 1 located in AZ1</td>
</tr>
<tr>
<td>Web2CIDR</td>
<td>10.16.1.0/24</td>
<td>CIDR Block for Web/StoreFront Subnet 2 located in AZ2</td>
</tr>
<tr>
<td>Infra1CIDR</td>
<td>10.16.2.0/24</td>
<td>CIDR Block for Infrastructure Subnet 1 located in AZ1</td>
</tr>
<tr>
<td>Infra2CIDR</td>
<td>10.16.3.0/24</td>
<td>CIDR Block for Infrastructure Subnet 2 located in AZ2</td>
</tr>
<tr>
<td>Desktop1CIDR</td>
<td>10.16.4.0/24</td>
<td>CIDR Block for Desktop Subnet 1 located in AZ1</td>
</tr>
<tr>
<td>Desktop2CIDR</td>
<td>10.16.5.0/24</td>
<td>CIDR Block for Desktop Subnet 2 located in AZ2</td>
</tr>
<tr>
<td>VPCCIDR</td>
<td>10.16.0.0/16</td>
<td>CIDR Block for the VPC</td>
</tr>
<tr>
<td>AD1PrivateIp</td>
<td>10.16.2.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1</td>
</tr>
<tr>
<td>AD2PrivateIp</td>
<td>10.16.3.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>desktop.xencloud.net</td>
<td>Fully qualified domain name (FQDN) to be used for the DHCP scope e.g. desktop.xencloud.net</td>
</tr>
</tbody>
</table>
We are using references to four different types of resources.

{ "Ref" : "VPC" } is a reference to the VPC created in the previous step. Launch all subnets into this VPC.

{ "Ref" : "Infra1CIDR" } is a reference to the CIDR block for this private subnet as we want to give users of the template the ability to define the IP ranges for each subnet to best match what they are used to from their on-premise deployment. This parameter definition is as follows:

```
"Infra1CIDR" : {
  "Description" : "CIDR Block for Infrastructure Subnet 1 located in AZ1",
  "Type" : "String",
  "Default" : "10.16.2.0/24",
  "AllowedPattern" : "[a-zA-Z0-9]+\..+
}
```

{ "Ref" : "AZ1" } is a reference to the Availability Zone in which you want to create the subnet. As we outlined earlier, we want to set up a mirror in two Availability Zones to provide redundancy and failover. This parameter definition for AZ1 is as follows (the definition is similar for AZ2):

```
"AZ1" : {
  "Description" : "Name of Availability Zone that will contain public & private subnets - Select a valid zone for your region",
  "Type" : "String",
  "Default" : "us-east-1a",
  "ConstraintDescription" : "must be a valid EC2 Availability Zone for region being deployed to. Only supports eu-west-1, us-east-1, us-west-1, sa-east-1, ap-northeast-1, ap-southeast-1 & ap-southeast-2 <- You can expand"
}
```

We are using a reference to the StackName property { "Ref" : "AWS::StackName"} to tag our subnet.

Besides the six private subnets, we also want to deploy two public subnets, one in each Availability Zone. Deploying public subnets follows the same pattern as described earlier with the private subnets. The only two things that distinguish public subnets from private are: the route (e.g., the public route channels Internet traffic directly to the Internet gateway while the private route channels Internet traffic to the NAT instance); and that instances in the public subnet actually have an Internet-routable IP address. We discuss how to define and encode public and private routes in the next section.

**Private and Public Routes**

After we create the VPC and the subnets inside the VPC, we need to define how traffic will flow inside the VPC and out of the VPC. We define the routes: one route for defining the traffic flow for all the private subnets, and one route for the two public subnets.
Before we can create those routes, however, we need to define the means by which the VPC communicates with the Internet. We create an Internet gateway resource of the type AWS::EC2::InternetGateway with a few lines of script. This script is as follows:

```
"InternetGateway" : {
    "Type" : "AWS::EC2::InternetGateway",
    "Properties" : {
        "Tags" : [
            {"Key" : "Application", "Value" : { "Ref" : "AWS::StackName" } },
            {"Key" : "Network", "Value" : "Public" }
        ]
    }
},
```

After we create the Internet gateway, we only have to attach the gateway to the VPC. The code for doing this is as follows:

```
"AttachGateway" : {
    "Type" : "AWS::EC2::VPCGatewayAttachment",
    "Properties" : {
        "VpcId" : { "Ref" : "VPC" },
        "InternetGatewayId" : { "Ref" : "InternetGateway" }
    }
},
```

Next, we create the NAT instance in each Availability Zone to facilitate servers in private subnets communicating out to the Internet (to get operating system software updates, for example). The code for doing this is as follows:

```
"NAT1" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArchNatAMI", { "Ref" : "AWS::Region" }, { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "NATInstanceType" }, "Arch" ] } ] },
        "InstanceType" : { "Ref" : "NATInstanceType" },
        "SubnetId" : { "Ref" : "DMZ1Subnet" },
        "Tags" : [ { "Key" : "Name", "Value" : "NAT1" } ],
        "SecurityGroupId" : { "Ref" : "NATSecurityGroup" },
        "KeyName" : { "Ref" : "KeyPairName" },
        "SourceDestCheck" : "false"
    }
},
```

Similar to other VPC and subnet resources, we are extensively using references either to previously-created resources like the DMZ Subnet { "Ref" : "DMZ1Subnet" } that we want to launch this instance into, or to a security group that governs the type of traffic we allow to flow in or out of this instance. (We discuss security group setup in detail later in this document.)

As the NAT instance resides in a public subnet, it also needs a publicly routable IP address. We achieve this by creating an EIP resource { "Type" : "AWS::EC2::EIP" } and associating it with the instance. The code for doing this is as follows in AWS CloudFormation:

```
"NAT1EIP" : {
    "Type" : "AWS::EC2::EIP",
    "Properties" : {
        "Domain" : "vpc",
        "InstanceId" : { "Ref" : "NAT1" }
    }
}
```
Now that we have both our Internet gateway and NAT instance deployed, we can construct our routes and associate the routes with the proper subnet.

First, we create the private route table. This looks as follows:

```
"PrivateRouteTable" : {
   "Type" : "AWS::EC2::RouteTable",
   "Properties" : {
      "VpcId" : {"Ref" : "VPC"},
      "Tags" : [
         {"Key" : "Application", "Value" : { "Ref" : "AWS::StackName"} },
         {"Key" : "Network", "Value" : "AZ1 Private" }]
   }
},
```

Then we construct the private route that is associated to the route table.

```
"PrivateRoute" : {
   "Type" : "AWS::EC2::Route",
   "Properties" : {
      "RouteTableId" : { "Ref" : "PrivateRouteTable" },
      "DestinationCidrBlock" : "0.0.0.0/0",
      "InstanceId" : { "Ref" : "NAT1" }
   }
},
```

What this route defines is that all traffic destined for the Internet ( "DestinationCidrBlock" : "0.0.0.0/0" ) has to go through the NAT instance we created earlier. Now we need to associate all six of our private subnets with this route, and the code for doing this is as follows:

```
"Infra1SubnetRouteTableAssociation1" : {
   "Type" : "AWS::EC2::SubnetRouteTableAssociation",
   "Properties" : {
      "SubnetId" : { "Ref" : "InfraSubnet" },
      "RouteTableId" : { "Ref" : "PrivateRouteTable" }
   }
},
```

This takes care of all outbound traffic from our private subnets to the Internet. What about traffic from the Internet to other types of instances we will deploy at a later stage into the DMZ, like a Bastion Host for securely logging into your instances? This traffic is routed via our public route and it follows the same pattern as established with the private route. First, we create the DMZ (public) route table as follows:

```
"DMZRouteTable" : {
   "Type" : "AWS::EC2::RouteTable",
   "Properties" : {
      "VpcId" : {"Ref" : "VPC"},
      "Tags" : [
         {"Key" : "Application", "Value" : { "Ref" : "AWS::StackName"} },
         {"Key" : "Network", "Value" : "DMZ" }
   }
},
```

Then we construct the DMZ (public) route and associate it with the DMZ route table. This is why we first had to create the Internet gateway resource (and the NAT instance earlier).

```
"DMZRoute" : {
   "Type" : "AWS::EC2::Route",
   "Properties" : {
      "RouteTableId" : { "Ref" : "DMZRouteTable" },
This route defines that all traffic destined for the Internet ("DestinationCidrBlock" : "0.0.0.0/0") has to go through the Internet gateway we created earlier. Now we need to associate the two public (DMZ) subnets with this route and the code for doing this is as follows:

"DMZSubnetRouteTableAssociation" : {  
"Type" : "AWS::EC2::SubnetRouteTableAssociation",  
"Properties" : {  
"SubnetId" : { "Ref" : "DMZSubnet" },  
"RouteTableId" : { "Ref" : "DMZRouteTable" }  
}  
},

Setting up Secure Administrative Access using Bastion Hosts

Before performing the final step in scripting the deployment of our infrastructure, setting up a set of security groups designed to isolate the different tiers of our Server Farm, and thereby significantly reducing any attack surface, we want to deploy our Bastion Host. As we designed our architecture for high availability, we deploy two Bastion Hosts, one in each Availability Zone. This way, we allow access to the resources that may have failed over to the other Availability Zone in case of an Availability Zone outage.

Deploying the Bastion Hosts using AWS CloudFormation in your scripted deployment is straightforward. All concepts that apply for this type of instance deployment were discussed earlier in this article. We launch the instance, but won’t join the instance to a domain.

Securing our Infrastructure with Security Groups

The Securing the Microsoft Platform on Amazon Web Services whitepaper discusses in detail the different methods for securing your AWS infrastructure. In this section, we examine how we actually encode our security groups so we repeatedly and reliably deploy an environment that is secure without having to perform any manual steps using the AWS Management Console.

A security group is just another resource of the type AWS::EC2::SecurityGroup that we declare and that has a set of properties. The two key properties that define how the security group behaves and what kind of filtering it performs are the properties SecurityGroupIngress and SecurityGroupEgress. For example, the security group we construct in our script to enable all domain member servers to communicate with the domain controllers for authentication and DNS lookup are as follows.

"DomainMemberSG" : {  
"Type" : "AWS::EC2::SecurityGroup",  
"Properties" : {  
"GroupDescription" : "Domain Members",  
"VpcId" : { "Ref" : "VPC" },  
"SecurityGroupIngress" : [  
{"IpProtocol" : "tcp", "FromPort" : "49152", "ToPort" : "65535", "CidrIp" : { "Ref" : "Infra1CIDR" } },  
{"IpProtocol" : "tcp", "FromPort" : "3389", "ToPort" : "3389", "CidrIp" : { "Ref" : "DMZ2CIDR" } }  
]  
}  
}
In our particular implementation, we only define ingress rules. If we define no egress rule, the default egress rule of allowing all outbound traffic is applied. In our VPC setting, this is fine, as it removes complexity in managing the security groups. Also, it provides a sufficient level of security, as individual instances (or instances of the same type that we deploy into separate subnets) listen only on the defined set of ports from specified IP ranges or members of other security groups. Our domain members listen to the typical high port range and port 53 (DNS) from our two domain controllers, plus they allow 3389 (RDP) traffic from our DMZs which host the Bastion Hosts.

With the setup of our security groups that govern access between subnets and instances, we will now deploy the remaining pieces of our foundational infrastructure, our CloudBridge to interconnect the On-Premises datacenter to our newly created AWS VPC, as well as the Active Directory components needed for our public XenApp server farm. We will configure the last piece of our infrastructure, NetScaler Gateway, in the final step 6 after we have set up the StoreFront servers.
Setting Up CloudBridge for the XenApp Cloud Hosted Farm Model

For the XenApp Cloud Hosted Farm model, we are accommodating the following requirements:

- We recommend at least three IP addresses and ENIs for a single NetScaler instance, and for a HA pair a single IP address and ENI per NetScaler instance for NetScaler management (NSIP) and at least two floating public and private subnet IP addresses and ENIs for the NetScaler HA pair.

- It is advisable for the Cloud Hosted Farm model to enable the essential datacenter security capabilities of the NetScaler Gateway for firewall and threat management. It is out of scope for this article and the accompanying artefacts. However, you might want to consider implementing these features after all the scripts have been run.

The CloudBridge deployment consists of three key components - the datacenter, CloudBridge, and AWS. The following infographic illustrates the topology of a seamless network with a secure tunnel and WAN optimization.

![Topology of network with a secure tunnel and WAN optimization](image)

**Figure 3:** Topology of network with a secure tunnel and WAN optimization

Following is the traffic flow in this topology:

1. The NetScaler in the datacenter sends the data to the CloudBridge/Branch Repeater appliance in the datacenter.

2. CloudBridge/Branch Repeater compresses the data and sends it back to the NetScaler.

3. The NetScaler uses the CloudBridge Connector tunnel and sends the compressed data over the bridge to its peer in the AWS cloud.

4. The NetScaler in the AWS cloud sends the compressed data to CloudBridge VPX/Branch Repeater in the AWS cloud for decompression.
5. This CloudBridge VPX/Branch Repeater decompresses the data and sends it back to the NetScaler in AWS.

6. The NetScaler then forwards the data.

The following infographic illustrates an example setting for the CloudBridge Connector tunnel and CloudBridge VPX (previously named Branch Repeater) setup.
Setting Up the NetScaler Appliance in the Datacenter

Before you set up a NetScaler MPX or VPX in the datacenter, you need to rack mount the MPX appliance, or provision the VPX appliance. For instructions

- To rack mount a NetScaler MPX appliance, see [http://support.citrix.com/proddocs/topic/netscaler-10-1/ns-gen-getting-started-wrapper-10-con.html](http://support.citrix.com/proddocs/topic/netscaler-10-1/ns-gen-getting-started-wrapper-10-con.html)

- To provision a NetScaler VPX virtual appliance, see [http://support.citrix.com/proddocs/topic/netscaler-10-1/ns-gen-nsvpx-wrapper-con-10.html](http://support.citrix.com/proddocs/topic/netscaler-10-1/ns-gen-nsvpx-wrapper-con-10.html)
A NetScaler appliance has both a command line interface (CLI) and a graphical user interface (GUI). The GUI includes a configuration utility for configuring the appliance. For initial access, all NetScaler appliances ship with the default NetScaler IP address (NSIP) of 192.168.100.1 and default subnet mask of 255.255.0.0. You can assign a new NSIP and an associated subnet mask during initial configuration. On AWS the NetScaler NSIP is determined by the AWS assigned IP address.
Setting up the CloudBridge Appliance in the Datacenter

This guide assumes that the CloudBridge in the datacenter is deployed in the one-arm mode. References to an appliance apply to virtual appliances (VPX) as well as physical appliances. After you have installed and completed the initial configuration on the CloudBridge appliance, configure the interfaces on the NetScaler and CloudBridge appliances so that they can communicate with each other. Also, configure the virtual inline and TCP Maximum Segment Size (MSS) settings on the CloudBridge appliance.

Configuring the Network at the Datacenter

By default, the first and second interfaces on the CloudBridge appliance are bridged. Configure one network interface on the NetScaler appliance and the first network interface on the CloudBridge appliance to be part of the client subnet. For the second interface on the CloudBridge appliance, do the following:

- If the CloudBridge appliance is a physical appliance, do not connect the interface to the network.
- If the CloudBridge appliance is a virtual appliance, configure a dummy internal network. Make sure that the second of the bridged ports is the only interface in the dummy internal network.

Configuring the TCP MSS and Virtual Inline Settings on the CloudBridge

The configuration on the CloudBridge appliance (either a physical appliance or a virtual appliance) in the datacenter must match the configuration on the CloudBridge instance in the Amazon VPC. CloudBridge VPX/Branch Repeater AMI instances on AWS have a preconfigured value of 1300 for both the default and maximum values of TCP MSS (maximum segment size). Additionally, the Virtual Inline option should be set to Return to Ethernet Sender, so that packets are returned to the NetScaler appliance. Therefore, you must configure these settings on the datacenter CloudBridge appliance.
To configure the TCP MSS and Virtual Inline settings on a CloudBridge appliance in the datacenter:

1. Log on to the CloudBridge appliance in the datacenter using the GUI or CLI.

2. In the Configuration menu, click Tuning.

3. On the Tuning page, do the following:

   • In Virtual Inline, click Return to Ethernet Sender, and then click Update.

   • In TCP Maximum Segment Size (MSS), in the Default MSS and Maximum MSS boxes, enter 1300, and then click Update.
Deploying NetScaler and Branch Repeater Appliances in AWS using CloudFormation

The NetScaler VPX Amazon Machine Image (AMI) is packaged as an EC2 instance that is launched within an AWS Virtual Private Cloud (VPC). The VPX AMI instance requires a minimum of 2 virtual CPUs and 2 GB of memory, which are available through various EC2 instance types. An EC2 instance launched within an AWS VPC can also provide the multiple interfaces, multiple IP addresses per interface, and public and private IP addresses needed for configuring CloudBridge VPX.

Currently, on Amazon AWS, NetScaler VPX can be launched only within a VPC, because each VPX requires at least three IP addresses per NetScaler instance. The Multi-IP functionality is available only to instances running within an AWS VPC. A VPX instance in a VPC can be used to load balance servers running in EC2. An Amazon VPC allows you to create and control a virtual networking environment, including your own IP address range, subnets, route tables and network gateways. Note: By default, you can create up to 5 VPC instances per AWS region for each AWS account; you can request for higher VPC limits using Amazon’s request form (http://aws.amazon.com/contact-us/vpc-request/). Although NetScaler on AWS can be implemented with one or two network elastic interfaces, Citrix recommends three network interfaces for a standard installation. Each AWS EC2 instance provides one default network interface. Amazon provides the functionality to attach additional network interfaces to EC2 instances running within an AWS VPC through the use of Elastic Network Interfaces (ENIs). You can attach up to eight ENIs to an EC2 instance.

Using XenApp RA CloudBridge Template

Open up the sample XenApp RA CloudBridge AWS CloudFormation template file and follow along. This template creates a NetScaler HA pair per each of the two Availability Zones, and a supporting CloudBridge VPX for traffic optimization, including all associated resources, such as ENIs, NetScaler and CloudBridge instances and security mappings.

Template Customization

The sample XenApp RA CloudBridge AWS CloudFormation (XA_VPC_CB) Template allows for rich customization of defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. AWS CloudFormation currently supports a maximum of 30 parameters per template.
The **Citrix XenApp on AWS: Implementation Guide**

The [Citrix XenApp RA CloudBridge](https://aws.amazon.com/cloudformation/aws-cloudformation/examples/itrexaan/labs中信/itexan) AWS CloudFormation template contains a Mappings section that maps the AMIs for a given region to a given NetScaler or CloudBridge instance on AWS Marketplace. Change these AMI references accordingly to your needs. The defaults in the sample template are for the **NetScaler VPX Platinum Edition - 10 Mbps** with the 10.1-119.7 firmware.

```json
"Mappings" : {
  "AWSInstanceType2Arch" : {
    "m1.small" : { "Arch" : "64" },
    "m1.medium" : { "Arch" : "64" },
    "m1.large" : { "Arch" : "64" },
    "m1.xlarge" : { "Arch" : "64" },
    "m2.xlarge" : { "Arch" : "64" },
    "m2.2xlarge" : { "Arch" : "64" },
    "m2.4xlarge" : { "Arch" : "64" },
    "m3.xlarge" : { "Arch" : "64" },
    "m3.2xlarge" : { "Arch" : "64" },
    "c1.medium" : { "Arch" : "64" },
    "c1.xlarge" : { "Arch" : "64" },
    "cc2.8xlarge" : { "Arch" : "64" }
  },
  "AWSRegionArchCBAMI" : {
    "us-east-1" : {"64":"ami-a57cf3cc"},
    "us-west-2" : {"64":"ami-ae0323eb"},
    "us-west-1" : {"64":"ami-eae166da"},
    "eu-west-1" : {"64":"ami-f044d9c0"},
    "ap-southeast-1" : {"64":"ami-955012c7"},
    "ap-southeast-2" : {"64":"ami-084dd32"},
    "ap-northeast-1" : {"64":"ami-c10ab0d"},
    "sa-east-1" : {"64":"ami-9c0bd381"}
  },
  "AWSRegionArchNSAMI" : {
    "us-east-1" : {"64":"ami-0d014164"},
    "us-west-2" : {"64":"ami-f044d9c0"},
    "us-west-1" : {"64":"ami-c26e4487"},
    "eu-west-1" : {"64":"ami-99f0e8ed"},
    "ap-southeast-1" : {"64":"ami-8ccba83e"},
    "ap-southeast-2" : {"64":"ami-256f61f"},
    "ap-northeast-1" : {"64":"ami-4928b548"},
    "sa-east-1" : {"64":"ami-f547e3e8"}
  }
},

To change this to use, for instance, the [NetScaler VPX - Customer Licensed](https://aws.amazon.com/cloudformation/aws-cloudformation/examples/itrexaan/labs中信/itexan) edition we will need to change the AWSRegionArchNSAMI block to map to the correct AMI references as found on AWS Marketplace. See the below example with the AMI references adapted to the NetScaler VPX - Customer Licensed edition:

```json
"AWSRegionArchNSAMI" : {
  "us-east-1" : {"64":"ami-991252f0"},
  "us-west-2" : {"64":"ami-1043e2c"},
  "us-west-1" : {"64":"ami-b26248f7"},
  "eu-west-1" : {"64":"ami-a3bf3c7"},
  "ap-southeast-1" : {"64":"ami-02ce8650"},
  "ap-southeast-2" : {"64":"ami-256f71f"},
  "ap-northeast-1" : {"64":"ami-fb2ebfa"},
  "sa-east-1" : {"64":"ami-9b41e586"}
}
```
The XA_VPC_CB AWS CloudFormation Template parameters include the following default values, the IDs of the various security groups and subnets are provided as output by the XA_VPC AWS CloudFormation Template.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>VPCID</td>
<td>10.16.0.0/16</td>
<td>VPC ID of the existing Virtual Private Cloud</td>
</tr>
<tr>
<td>NSVPXInstanceType</td>
<td>m1.large</td>
<td>Amazon EC2 instance type for the NetScaler VPX instances.</td>
</tr>
<tr>
<td>CBVPXInstanceType</td>
<td>m1.large</td>
<td>Amazon EC2 instance type for the CloudBridge Accelerator instances</td>
</tr>
<tr>
<td>NS1VPXNameAZ1</td>
<td>NS1VPXAZ1</td>
<td>Host name of the Primary NetScaler of the NetScaler pair located in AZ1</td>
</tr>
<tr>
<td>NS2VPXNameAZ1</td>
<td>NS2VPXAZ1</td>
<td>Host name of the Secondary NetScaler of the NetScaler pair located in AZ1</td>
</tr>
<tr>
<td>NS1VPXNameAZ2</td>
<td>NS1VPXAZ2</td>
<td>Host name of the Primary NetScaler of the NetScaler pair located in AZ2</td>
</tr>
<tr>
<td>NS2VPXNameAZ2</td>
<td>NS2VPXAZ2</td>
<td>Host name of the Secondary NetScaler of the NetScaler pair located in AZ2</td>
</tr>
<tr>
<td>CBVPXNameAZ1</td>
<td>CBVPXAZ1</td>
<td>Host name of the CloudBridge VPX located in AZ1</td>
</tr>
<tr>
<td>CBVPXNameAZ2</td>
<td>CBVPXAZ2</td>
<td>Host name of the CloudBridge VPX located in AZ2</td>
</tr>
<tr>
<td>DMZ1IP</td>
<td>10.16.9.10</td>
<td>Private IP for the NetScaler ENI located in the DMZ in AZ1</td>
</tr>
<tr>
<td>DMZ2IP</td>
<td>10.16.10.10</td>
<td>Private IP for the NetScaler ENI located in the DMZ in AZ2</td>
</tr>
<tr>
<td>Infra1MIP</td>
<td>10.16.2.10</td>
<td>Private IP (MIP) for the NetScaler ENI located in the Infrastructure subnet in AZ1</td>
</tr>
<tr>
<td>Infra2MIP</td>
<td>10.16.3.10</td>
<td>Private IP (MIP) for the NetScaler ENI located in the Infrastructure subnet in AZ2</td>
</tr>
<tr>
<td>Infra1CBIP</td>
<td>10.16.2.11</td>
<td>Private IP for the CloudBridge VPX located in the Infrastructure subnet in AZ1</td>
</tr>
<tr>
<td>Infra2CBIP</td>
<td>10.16.3.11</td>
<td>Private IP for the CloudBridge VPX located in the Infrastructure subnet in AZ2</td>
</tr>
<tr>
<td>NS1AZ1NSIP</td>
<td>10.16.7.10</td>
<td>Private NetScaler IP (NSIP) for the ENI of the primary NetScaler located in the NetScaler subnet in AZ1</td>
</tr>
<tr>
<td>NS2AZ1NSIP</td>
<td>10.16.8.10</td>
<td>Private NetScaler IP (NSIP) for the ENI of the secondary NetScaler located in the NetScaler subnet in AZ1</td>
</tr>
<tr>
<td>NS1AZ2NSIP</td>
<td>10.16.7.11</td>
<td>Private NetScaler IP (NSIP) for the ENI of the secondary NetScaler located in the NetScaler subnet in AZ1</td>
</tr>
<tr>
<td>NS2AZ2NSIP</td>
<td>10.16.8.10</td>
<td>Private NetScaler IP (NSIP) for the ENI of the primary NetScaler located in the NetScaler subnet in AZ1</td>
</tr>
<tr>
<td>DMZ1Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the DMZ Subnet located in AZ1</td>
</tr>
<tr>
<td>DMZ2Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the DMZ Subnet located in AZ2</td>
</tr>
<tr>
<td>Infra1Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ1</td>
</tr>
<tr>
<td>Infra2Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ2</td>
</tr>
</tbody>
</table>

AllOpenSecurityGroup       | <User Provides>             | ID of the All Open Security Group                                           |
NSSecurityGroup             | <User Provides>             | ID of the NetScaler Security Group                                           |
NS1Subnet                   | <User Provides>             | ID of the NetScaler Subnet located in AZ1                                    |
NS2Subnet                   | <User Provides>             | ID of the NetScaler Subnet located in AZ2                                    |
DMZSecurityGroup            | <User Provides>             | ID of the DMZ Security Group                                                 |
DMZ1Subnet                  | <User Provides>             | ID of the DMZ Subnet located in AZ1                                           |
DMZ2Subnet                  | <User Provides>             | ID of the DMZ Subnet located in AZ2                                           |
Infra1Subnet                | <User Provides>             | ID of the Infrastructure Subnet located in AZ1                              |
Infra2Subnet                | <User Provides>             | ID of the Infrastructure Subnet located in AZ2                              |
NetScaler HA Pairs

After we create the VPC and the subnets inside the VPC, we need to define how traffic will flow inside the VPC and out of the VPC. We define the routes: one route for defining the traffic flow for all the private subnets, and one route for the two public subnets.

Before we can create the NetScaler HA pair we need to define the ENIs for each of the NetScaler VPXes. This script defines the ENIs for the Primary VPX in each Availability Zone and assigns these ENIs to their respective Security Groups:

```json
"AZ1NS1ENI": {
    "Type": "AWS::EC2::NetworkInterface",
    "Properties": {
        "Description": "ENI connected to primary Netscaler in subnet in AZ1",
        "SubnetId": {
            "Ref": "NS1Subnet"
        },
        "GroupSet": [ {
            "Ref": "NSSecurityGroup"
        } ],
        "PrivateIpAddress": {
            "Ref": "NS1AZ1NSIP"
        }
    }
},

"Infra1ENI": {
    "Type": "AWS::EC2::NetworkInterface",
    "Properties": {
        "Description": "ENI connected to Infrastructure subnet in AZ1",
        "SubnetId": {
            "Ref": "Infra1Subnet"
        },
        "GroupSet": [ {
            "Ref": "AllOpenSecurityGroup"
        } ],
        "PrivateIpAddress": {
            "Ref": "Infra1MIP"
        }
    }
},

"DMZ1ENI": {
    "Type": "AWS::EC2::NetworkInterface",
    "Properties": {
        "Description": "ENI connected to DMZ subnet in AZ1",
        "SubnetId": {
            "Ref": "DMZ1Subnet"
        },
        "GroupSet": [ {
            "Ref": "DMZSecurityGroup"
        } ],
        "PrivateIpAddress": {
            "Ref": "DMZ1IP"
        }
    }
}
```
This part of the script defines the single ENI for the Secondary VPX in each Availability Zone and assigns this ENI to its Security Groups:

```
"AZ1NS2ENI": {
  "Type": "AWS::EC2::NetworkInterface",
  "Properties": {
    "Description": "ENI connected to secondary Netscaler in subnet in AZ1",
    "SubnetId": {
      "Ref": "NS1Subnet"
    },
    "GroupSet": [
      {
        "Ref": "NSSecurityGroup"
      }
    ],
    "PrivateIpAddress": {
      "Ref": "NS2AZ1NSIP"
    }
  }
},
```

Now we can instantiate the Primary VPX in each Availability Zone including the mapping of the ENIs and the passing of the IAM access keys. These IAM access keys will be used by the NetScaler instances in an HA setup to initiate failover in case of a failure of one of the NetScaler nodes. Note also that these IAM access keys have to be provided to the instance via UserData upon instantiation of the instances. Note that NetScaler requires that the Source and Destination Check be set to disabled for its ENIs to function correctly. The code for doing this is as follows:

```
"NS1AZ1VPXInstance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
    "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArchNSAMI", { "Ref" : "AWS::Region" }, { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "NSVPXInstanceType" }, "Arch" ] } ] },
    "InstanceType": {
      "Ref": "NSVPXInstanceType"
    },
    "NetworkInterfaces": [ 
      {
        "NetworkInterfaceId": {
          "Ref": "AZ1NS1ENI"
        },
        "DeviceIndex": "0"
      }, 
      {
        "NetworkInterfaceId": {
          "Ref": "DMZ1ENI"
        },
        "DeviceIndex": "1"
      }, 
      {
        "NetworkInterfaceId": {
          "Ref": "Infra1ENI"
        },
        "DeviceIndex": "2"
      }
    ],
    "UserData": {
      "Fn::Base64": {
        "Fn::Join": [ "", [
```

Citrix XenApp on AWS: Implementation Guide

"ACCESS_KEY=",
"\n",
{ "Ref": "IAMUserAccessKey"
},
"\n",
"SECRET_KEY=",
"\n",
{ "Fn::GetAtt": [
    "IAMUserAccessKey",
    "SecretAccessKey"
] }
},
"\n"

"Tags": [ 
{ 
  "Key": "Name",
  "Value": { "Ref" : "NS1VPXNameAZ1" }
}
],
"KeyName": {
  "Ref": "KeyPairName"
},
"SourceDestCheck" : "false"
}

Instantiating the secondary VPX is similar besides that there’s only one ENI (the ENI that will be connected to the NetScaler management subnet and will host the NSIP) that needs to be mapped. The code for doing this is as follows:

"NS2AZ1VPXInstance": {
  "Type": "AWS::EC2::Instance",
  "Properties": {
   "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArchNSAMI", { "Ref" : "AWS::Region" }, 
   { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "NSVPXInstanceType" }, "Arch" ] } ] },
   "InstanceType": { 
    "Ref": "NSVPXInstanceType"
   },
   "NetworkInterfaces": [ 
    { "NetworkInterfaceId": { 
      "Ref": "AZ1NS2ENI"
    },
    "DeviceIndex": "0"
  }
],
  "UserData": { 
   "Fn::Base64": { 
    "Fn::Join": [ 
      
      "ACCESS_KEY=",
      "\n",
      { "Ref": "IAMUserAccessKey"
    },
    "\n",
    "SECRET_KEY=",
    "\n"}
Similar to other VPC and subnet resources, we are extensively using references either to previously-created resources like the NetScaler NSIP ENI in AZ1 ["Ref": "AZ1NS2ENI"] with which we want to launch this instance, or to a security group that governs the type of traffic we allow to flow in or out of this ENI for this instance. (We discuss security group setup in detail later in this document.)

As the NetScaler instances have ENIs that reside in a public subnet, it also needs a publicly routable IP address. We achieve this by creating an EIP resource ["Type": "AWS::EC2::EIP"] and associating it with the respective ENIs for this instance. The code for allocating the EIP is as follows in AWS CloudFormation:

```json
"AZ1NS1EipNsip": {
  "Type": "AWS::EC2::EIP",
  "Properties": {
    "Domain": "vpc"
  }
},
```

The code for associating the allocated EIP to the right ENI is as follows in AWS CloudFormation:

```json
"AssociateAZ1NS1EipNsip": {
  "Type": "AWS::EC2::EIPAssociation",
  "Properties": {
    "AllocationId": {
      "Fn::GetAtt": [
        "AZ1NS1EipNsip",
        "AllocationId"
      ]
    },
    "NetworkInterfaceId": {
      "Ref": "AZ1NS1ENI"
    }
  }
},
```
CloudBridge VPX Instances

After we have created the NetScaler HA pairs we now need to define the CloudBridge instances in each Availability Zone. This script defines the CloudBridge VPX in each Availability Zone and assigns them to their respective Security Groups and places them in the respective Infrastructure Subnets:

```
"NSAZ1CBInstance": { 
  "Type": "AWS::EC2::Instance", 
  "Properties": { 
    "ImageId" : { "Fn::FindInMap" : [ "AWSRegionArchCBAMI", { "Ref" : "AWS::Region" }, { "Fn::FindInMap" : [ "AWSInstanceType2Arch", { "Ref" : "CBVPXInstanceType" }, "Arch" ] } ] }, 
    "InstanceType": {"Ref": "CBVPXInstanceType"}, 
    "SubnetId" : { "Ref" : "Infra1Subnet" }, 
    "Tags" : [ { 
      "Key" : "Name", 
      "Value" : { "Ref" : "CBVPXNameAZ1" } 
    } ], 
    "SecurityGroupIds" : [ {"Ref": "AllOpenSecurityGroup"} ], 
    "PrivateIpAddress" : { "Ref" : "Infra1CBIP" } 
  } 
},
```

Accessing the NetScaler Instance on AWS

When the NetScaler instances are running, you can access the instances through the NetScaler GUI or the NetScaler CLI by connecting to the EIP associated with management ENI (NSIP). For example, use the following addressing notation in a web browser: http://<Elastic_IP>:80 (unsecured access) or https://<Elastic_IP>:8080 (secured access) 

Notes:

- The default username and password of the NetScaler instance are nsroot and nsroot.
- To access a NetScaler instance via SSH, provide the .pem file.
- You can use the AWS GUI console to manually add the private IP addresses for MIPs or SNIPs on server subnets and VIPs on client subnets.
  a. MIPs must be used for
     i. Reach subnets in same VPC but a different AZ
     ii. Reaching IPs outside of the VPC
  b. IP connectivity (SNIP/MIP)
     i. SNIPs can be used for connectivity to resources in same AZ/VPC
     ii. MIPs must be used for connectivity outside of AZ or VPC
If you want to access the NSIP from the Internet, you must assign an EIP to the NSIP address of each NetScaler instance. Take care to adapt accordingly the Security Group or Network ACL to which this ENI that hosts the EIP belongs.

If you want VIP addresses to be accessible through the Internet you must associate an EIP with each VIP address that is defined in the configuration.

**Accessing the CloudBridge Instance on AWS**

When the CloudBridge instances are up and running, you can access the instances through the CloudBridge/Branch Repeater administration GUI by connecting to the IP associated with the CloudBridge VPX instance. For example, use the following addressing notation in a web browser: http://<IP>:80 (unsecured access) or https://<IP>:443 (secured access) **Notes:**

- A few manual configuration steps need to be completed:
  
  a. [Disabling the Source/Destination Check Feature](#)
  
  b. [Configuring SNMP Monitoring on the CloudBridge VPX AMI on AWS](#)

- Limitations and Usage Guidelines for CloudBridge VPX/Branch Repeater AMI Instances on AWS

  a. High Availability setup for CloudBridge VPX AMI instances is not supported.

  b. CloudBridge VPX AMI instance in Group Mode is not supported.

  c. Branch Repeater/CloudBridge plug-ins are not supported.

  d. Tagged VLAN is not supported because of the inherent limitation of AWS.

  e. Traffic shaping is not supported.

  f. You may create only an m1.large CloudBridge VPX AMI instance on AWS.

  g. IP address/gateway/subnet assignment using the CloudBridge VPX/Branch Repeater management user interface is not supported.

  h. Console access is not available for CloudBridge VPX/Branch Repeater AMI instances on AWS.

  i. It is not possible to install licenses on a Branch Repeater AMI instance on AWS. The Branch Repeater AMI instance is automatically licensed and rate limiting is controlled by the CloudBridge instance, which is paired with the Branch Repeater AMI instance.
j. While configuring the Branch Repeater instance, you may not change the disk size, which has a default value of 250 GB. A higher capacity disk does not increase the available Disk Based Compression (DBC) cache size.
CloudBridge Connector

The CloudBridge Connector feature of the Citrix NetScaler appliances connects enterprise datacenters to public clouds such as AWS, making the cloud a secure extension of the enterprise network and suited for XenApp deployment models which use On-Premises resources such as Active Directory or user data. Cloud hosted applications such as the XenApp deployment in this implementation appear as though they are running on one contiguous routed enterprise network.

Configuring CloudBridge Connector between Datacenter and AWS Cloud

To setup the CloudBridge Connector tunnel between the previously instantiated NetScaler HA pairs that reside on the AWS cloud and the NetScaler appliance that resides in the datacenter, we will use the configuration utility of the On-Premises NetScaler appliance.

When you use the configuration utility, the CloudBridge Connector tunnel configuration created on the NetScaler appliance, is automatically pushed to the other endpoint or peer (the NetScaler VPX on AWS) of the CloudBridge Connector tunnel. Therefore, you do not have to access the configuration utility (GUI) of the NetScaler VPX on AWS to create the corresponding CloudBridge Connector tunnel configuration on it.

The CloudBridge Connector tunnel configuration on both peers (the NetScaler appliance that resides in the datacenter and the NetScaler virtual appliance (VPX) that resides on the AWS cloud) consists of the following entities:

- **IPSec profile.** An IPSec profile entity specifies the IPSec protocol parameters, such as IKE version, encryption algorithm, hash algorithm, and PSK, to be used by the IPSec protocol in both the peers of the CloudBridge Connector tunnel.

- **GRE tunnel.** An IP tunnel specifies a local IP address (a public SNIP address configured on the local peer), remote IP address (a public SNIP address configured on the remote peer), protocol (GRE) used to set up the CloudBridge Connector tunnel, and an IPSec profile entity.

- **Netbridge.** A logical container that holds or represents the CloudBridge Connector tunnel configuration on each of the peers. A GRE tunnel entity is associated with the netbridge. A particular CloudBridge Connector tunnel configuration on a peer is identified by the name of the netbridge entity.
To configure a CloudBridge Connector tunnel in a NetScaler appliance by using the configuration utility

1. Type the NSIP address of a NetScaler appliance in the address line of a web browser.

2. Log on to the configuration utility of the NetScaler appliance by using your account credentials for the appliance.


4. In the right pane, under Getting Started, click Create/Monitor CloudBridge.

5. Click Get Started.
Note: If you already have any CloudBridge Connector tunnel configured on the NetScaler appliance, this screen does not appear, and you are taken to the CloudBridge Connector Setup pane.

6. In the CloudBridge Connector Setup pane, click amazon web services.
7. In the Amazon pane, enter the values for the access keys in the AWS Access Key ID and AWS Secret Access Key text boxes, and then click Continue. You can obtain these access keys from the AWS GUI console. Click Continue.

8. In the NetScaler pane, select the NSIP address or enter the EIP associated with the NSIP address of the NetScaler VPX running in AWS. Then, provide your account credentials for the NetScaler VPX. Click Continue.

9. In the CloudBridge Connector Setting pane, set the following parameter:

   - **CloudBridge Connector Name** - Name for the CloudBridge Connector configuration on the local appliance. Must begin with an ASCII alphabetic or underscore (_) character, and must contain only ASCII alphanumeric, hash (#), period (.), space, colon (:), at (@), equals (=), and hyphen (-) characters. Cannot be changed after the CloudBridge Connector configuration is created.

10. Under Local Setting, set the following parameter:

    - **Subnet IP** - IP address of the local endpoint of the CloudBridge Connector tunnel. Must be a public IP address of type SNIP.

11. Under Remote Setting, set the following parameters:

    - **Subnet IP** - IP address of the CloudBridge Connector tunnel endpoint on the AWS side. Must be an IP address of type SNIP on the NetScaler VPX instance on the AWS.
    
    - **NAT** - Public IP address (EIP) in AWS that is mapped to the SNIP configured on the NetScaler VPX instance on AWS.

12. (Optional) Under Security Settings, set the following IPSec protocol parameters to be used by the IPSec protocol in the CloudBridge Connector tunnel:

    - **Encryption Algorithm** - Encryption algorithm to be used by IPSec protocol in the CloudBridge tunnel.
    
    - **Hash Algorithm** - Hash algorithm to be used by the IPSec protocol in the CloudBridge Connector tunnel.
    
    - **Key** - Select one of the following IPSec authentication methods to be used by the two peers to mutually authenticate.

       - **Auto Generate Key** - Authentication based on a text string, called a pre-shared key (PSK), generated automatically by the local appliance. The PSKs keys of the peers are matched against each other for authentication.
13. Click **Done**.

The new CloudBridge Connector tunnel configuration on the NetScaler appliance in the datacenter appears on the **Home** tab of the configuration utility. The corresponding new CloudBridge Connector Tunnel configuration on the NetScaler VPX appliance in the AWS VPC appears on the configuration utility.

The current status of the CloudBridge Connector tunnel is indicated in the **Configured Cloud Bridges** pane. A green dot indicates that the tunnel is up. A red dot indicates that the tunnel is down.
Note: The CloudFormation API does not allow associating AWS elastic IP with an ENI. To associate one or more EIPs with an ENI in the AWS Console in the Navigation pane, in the NETWORK & SECURITY area, click Elastic IPs and associate EIPs with all VIPs (Private IPs) defined on this AWS/VPX instance.
Redirecting traffic to the CloudBridge VPX Appliances

On the NetScaler appliances in the datacenter and in the Amazon VPC, specify the CloudBridge appliance as a service by using the Dynamic Load Balancing wizard for Citrix CloudBridge. The wizard automatically creates a service and two virtual servers on the NetScaler appliance. The following table describes the service and virtual servers created by the wizard.

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>Entity Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR_LB_SVC_DYN_&lt;IP address of the CloudBridge/Branch Repeater appliance&gt; For example, BR_LB_SVC_DYN_10.17.0.10</td>
<td>Service</td>
<td>A service representing the CloudBridge VPX/Branch Repeater appliance.</td>
</tr>
<tr>
<td>BR_LB_VS_DYN_1 and BR_LB_VS_DYN_2</td>
<td>Virtual Server</td>
<td>Virtual servers to entertain different types of traffic for the CloudBridge VPX appliance. The service (for example BR_LB_SVC_DYN_10.17.0.10) representing the CloudBridge VPX appliance is bound to both the virtual servers.</td>
</tr>
</tbody>
</table>

To configure the load balancing of CloudBridge appliances by using the NetScaler configuration utility

In the left pane, click Load Balancing, and then, in the right pane, click Dynamic Load Wizard for Citrix CloudBridge.
In the wizard, on the Specify Branch Repeaters page, click Add.

In the Specify Branch Repeater IP Address dialog box, enter the IP address of a CloudBridge/Branch Repeater appliance, click Create, and then click Close.

Click Next.
On the Specify Data Center Server Subnets page, click Add.

In the Create Server Subnet dialog box, specify the following parameters:

- **Subnet IP** - The IP address of the subnet in which the servers reside.
- **Mask** - The subnet mask.

Repeat these steps until you have added all the server subnets, and then click Close.

Click Next and click Finish.
The Summary Page displays additional settings that are added automatically to complete the Branch Repeater load balancing configuration on the NetScaler appliance.

Click Close.
**Configuring Listen Policies for the Virtual Servers Created by the Wizard**

When setting up the Branch Repeater appliance as a service by using the Dynamic Load Balancing wizard for Citrix CloudBridge, the *Prefer Direct Route* parameter is set to **NO** to enable for the NetScaler to forward traffic to the CloudBridge through the wildcard virtual server/service.

To ensure that even when the CloudBridge appliance is down, and consequently the virtual server goes down, the NetScaler looks up the correct route and sends traffic, you need to set the following listen policies for the virtual servers BR_LB_VS_DYN_1 and BR_LB_VS_DYN_2. These virtual servers were created on the NetScaler appliance by the Dynamic Load Balancing wizard for Citrix CloudBridge.

<table>
<thead>
<tr>
<th>Virtual servers</th>
<th>Parameters to be modified</th>
<th>Value to be set for the parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR_LB_VS_DYN_1</td>
<td>Listen Policy</td>
<td>sys.vserver(&quot;BR_LB_VS_DYN_1&quot;).state.eq(up)&amp;client.tcp.repeater_option.exists&amp;client.ip.src.eq(&lt;IP address of the Branch Repeater appliance&gt;).not For example, sys.vserver(&quot;BR_LB_VS_DYN_1&quot;).state.eq(up)&amp;client.tcp.repeater_option.exists&amp;client.ip.src.eq(192.0.2.30).not</td>
</tr>
<tr>
<td>BR_LB_VS_DYN_2</td>
<td>Listen Policy</td>
<td>sys.vserver(&quot;BR_LB_VS_DYN_2&quot;).state.eq(up)&amp;client.ip.src.eq(&lt;IP address of the Branch Repeater appliance&gt;).not For example, sys.vserver(&quot;BR_LB_VS_DYN_2&quot;).state.eq(up)&amp;client.ip.src.eq(192.0.2.30).not</td>
</tr>
</tbody>
</table>

To set listen policies for the virtual servers to which the CloudBridge service is bound by using the NetScaler command line At the NetScaler command prompt, type:

```plaintext
set lbvserver <name> -Listenpolicy <expression>
```

Example:

```plaintext
set lbvserver BR_LB_VS_DYN_1 -listenpolicy sys.vserver
(“BR_LB_VS_DYN_1”).state.eq(up)&client.tcp.repeater_option.exists&client.ip.src.eq(<BR_IP>).not
```

Done

```plaintext
>set lbvserverBR_LB_VS_DYN_2 -
listenpolicysys.vserver("BR_LB_VS_DYN_2").state.eq(up)&client.ip.src.eq(<IP of BR>).not
```

Done
Launching the Active Directory in the Right Order

Now we have our connection up between the On-Premises datacenter and the AWS regions, optimized and accelerated by the CloudBridges on both end.

After this step, we are ready to deploy the Active Directory instances into the VPC. If we try to deploy an Active Directory instance in a private subnet inside a VPC without the proper public and private routing setup, the instance itself is created but none of the subsequent configuration scripts are executed. AWS CloudFormation requires access to the AWS CloudFormation Endpoints where the state of Wait Conditions is stored so we can orchestrate the right launch order of our resources.

Here’s a list of the AWS CloudFormation Endpoints whose DNS names must be resolvable and reachable from our instances.

Table 1 The AWS CloudFormation endpoints are:

<table>
<thead>
<tr>
<th>Region</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Northern Virginia) Region</td>
<td>cloudformation.us-east-1.amazonaws.com</td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>cloudformation.us-west-2.amazonaws.com</td>
</tr>
<tr>
<td>US West (Northern California) Region</td>
<td>cloudformation.us-west-1.amazonaws.com</td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>cloudformation.eu-west-1.amazonaws.com</td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>cloudformation.ap-southeast-1.amazonaws.com</td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>cloudformation.ap-southeast-2.amazonaws.com</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>cloudformation.ap-northeast-1.amazonaws.com</td>
</tr>
<tr>
<td>South America (Sao Paulo) Region</td>
<td>cloudformation.sa-east-1.amazonaws.com</td>
</tr>
</tbody>
</table>

Note: All AWS CloudFormation endpoints use the HTTPS protocol for access.
AD DS Setup and DNS Configuration

XenApp 6.5 requires Active Directory Domain Services (AD DS) for user authentication and availability of the complete feature set. However, you can also leverage AD DS to provide Domain Name Server (DNS) functionality within the VPC among the various server instances.

For the XenApp server farm to operate, you need connectivity to one or more domain controllers to facilitate user authentication and DNS resolution across servers within the farm.

In the hybrid XenApp scenarios, the XenApp server farm uses CloudBridge to connect to the corporate infrastructure. It is advised to maintain AD DS to be instantiated within the AWS environment to facilitate user registration and authentication for the XenApp instances running there. For more information on detailed setup and configuration steps, see the “Windows Server Setup and Configuration” section. It is suggested that you host domain controllers in multiple Availability Zones to provide redundancy and high availability, as illustrated in Figure 7.

Figure 7: Active Directory Setup
Setting up and configuring Windows Server for AD DS

First, let’s take a closer look at how we configure and provision the Windows Server instance with the AD DS and DNS server role.

In the previous section, we introduced one of the AWS CloudFormation helper scripts: cfn-signal.exe. The real workhorse of the AWS CloudFormation helper scripts—cfn-init—provide the ability to execute a number of detailed configuration tasks on Windows-based instances. The cfn-init helper script reads template metadata from the AWS::CloudFormation::Init key and acts accordingly to perform the following tasks:

- Fetch and parse metadata from AWS CloudFormation.
- Install packages.
- Write files to disk.
- Enable/disable and start/stop services.

For more information about the template metadata that cfn-init uses, see AWS::CloudFormation::Init.

So let’s understand how to accomplish this in our code.

Using XenApp RA Active Directory Template

We have created two sample AWS CloudFormation templates. The first covers the scenario of a standalone AD infrastructure on AWS useful for POC requirements (XA_VPC_DC_S). The second AWS CloudFormation Template (XA_VP_DC) covers the hybrid scenario, with an existing AD DS infrastructure in place on-premises, where we will need to place additional domain controllers for at least the resource domain (in our example desktop.xencloud.net) in the AWS VPC. Open up the sample XenApp RA AD hybrid scenario AWS CloudFormation template file or the sample XenApp RA AD standalone AD scenario and follow along. These templates create an AD Domain Controller per Availability Zone. The hybrid scenario will require access to an existing domain controller of the AD infrastructure to be able to join the domain.

Template Customization

The sample XenApp RA AD AWS CloudFormation (XA_VPC_DC or XA_VPC_DC_S) Templates allow for rich customization of defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. AWS CloudFormation currently supports a maximum of 30 parameters per template.
### Table 2: XenApp on AWS RA AD Hybrid Infrastructure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>ADInstanceType</td>
<td>m1.xlarge</td>
<td>Amazon EC2 instance type for the AD Instances</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>desktop.xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the domain; e.g., desktop.xencloud.net.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>DESKTOP</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., DESKTOP.</td>
</tr>
<tr>
<td>RestoreModePassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for a separate Administrator account when the domain controller is in Restore Mode. Must be at least 8 characters containing letters, numbers and symbols.</td>
</tr>
<tr>
<td>Server1NetBIOSName</td>
<td>AWSDSKDC1</td>
<td>NetBIOS name of the first AD Server (up to 15 characters) located in AZ1.</td>
</tr>
<tr>
<td>Server2NetBIOSName</td>
<td>AWSDSKDC2</td>
<td>NetBIOS name of the second AD Server (up to 15 characters) located in AZ2.</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This is separate from the default administrator account.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td>XAAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp Admin account. This Account is a Domain User and will also be added later to the SQL DB as a member of the dbcreator role.</td>
</tr>
<tr>
<td>XAAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the XenApp Admin account. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td>DomainControllerSG</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Controller Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>AZ1</td>
<td>us-east-1a</td>
<td>Name of Availability Zone that will contain public &amp; private subnets. Select a valid Zone for your region.</td>
</tr>
<tr>
<td>AZ2</td>
<td>us-east-1b</td>
<td>Name of Availability Zone that will contain public &amp; private subnets. Select a valid Zone for your region.</td>
</tr>
<tr>
<td>Infra1Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ1 you want to provision the AD Server into.</td>
</tr>
<tr>
<td>Infra2Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ2 you want to provision the AD Server into.</td>
</tr>
<tr>
<td>AD1PrivateIp</td>
<td>10.16.3.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1.</td>
</tr>
<tr>
<td>AD2PrivateIp</td>
<td>10.16.5.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2.</td>
</tr>
<tr>
<td>AD1OnPremIp</td>
<td>10.17.0.10</td>
<td>IP address for the first existing Active Directory server for the resource domain located On-Premises.</td>
</tr>
<tr>
<td>AD2OnPremIp</td>
<td>10.17.0.11</td>
<td>IP address for the second existing Active Directory server for the resource domain located On-Premises.</td>
</tr>
</tbody>
</table>
## XenApp on AWS RA AD Standalone Infrastructure Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KeyPairName</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td><strong>ADInstanceType</strong></td>
<td><em>m1.xlarge</em></td>
<td>Amazon EC2 instance type for the SQL instance</td>
</tr>
<tr>
<td><strong>DomainDNSName</strong></td>
<td><em>xencloud.net</em></td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td><strong>DomainNetBIOSName</strong></td>
<td><em>xencloud</em></td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td><strong>RestoreModePassword</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>Password for a separate Administrator account when the domain controller is in Restore Mode. Must be at least 8 characters containing letters, numbers and symbols.</td>
</tr>
<tr>
<td><strong>Server1NetBIOSName</strong></td>
<td><em>AWSDC1</em></td>
<td>NetBIOS name of the first AD Server (up to 15 characters) located in AZ1</td>
</tr>
<tr>
<td><strong>Server2NetBIOSName</strong></td>
<td><em>AWSDC2</em></td>
<td>NetBIOS name of the second AD Server (up to 15 characters) located in AZ2</td>
</tr>
<tr>
<td><strong>DomainAdminUser</strong></td>
<td><em>XenAdmin</em></td>
<td>User name for the account that is added as domain administrator. This is separate from the default &quot;administrator&quot; account.</td>
</tr>
<tr>
<td><strong>DomainAdminPassword</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>Password for the domain admin user. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td><strong>XAAdminUser</strong></td>
<td><em>XAFarmAdmin</em></td>
<td>User name for the XenApp Admin Account. This Account is a Domain User and will also be added to the SQL DB as a member of the dbcreator role.</td>
</tr>
<tr>
<td><strong>XAAdminPassword</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>Password for the XA admin user. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td><strong>AZ1</strong></td>
<td><em>us-east-1a</em></td>
<td>Name of primary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td><strong>AZ12</strong></td>
<td><em>us-east-1b</em></td>
<td>Name of secondary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td><strong>DomainControllerSGID</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td><strong>VPC</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>ID of the VPC</td>
</tr>
<tr>
<td><strong>Infra1Subnet</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>ID of the Infrastructure Subnet located in AZ1 you want to provision the SQL Server into</td>
</tr>
<tr>
<td><strong>Infra2Subnet</strong></td>
<td><em>&lt;User Provides&gt;</em></td>
<td>ID of the Infrastructure Subnet located in AZ2 you want to provision the SQL Server into</td>
</tr>
<tr>
<td><strong>Web1CIDR</strong></td>
<td><em>10.16.0.0/24</em></td>
<td>CIDR Block for Web/StoreFront Subnet 1 located in AZ1</td>
</tr>
<tr>
<td><strong>Web2CIDR</strong></td>
<td><em>10.16.1.0/24</em></td>
<td>CIDR Block for Web/StoreFront Subnet 2 located in AZ2</td>
</tr>
<tr>
<td><strong>Infra1CIDR</strong></td>
<td><em>10.16.2.0/24</em></td>
<td>CIDR Block for Infrastructure Subnet 1 located in AZ1</td>
</tr>
<tr>
<td><strong>Infra2CIDR</strong></td>
<td><em>10.16.3.0/24</em></td>
<td>CIDR Block for Infrastructure Subnet 2 located in AZ2</td>
</tr>
<tr>
<td><strong>Desktop1CIDR</strong></td>
<td><em>10.16.4.0/24</em></td>
<td>CIDR Block for Desktop Subnet 1 located in AZ1</td>
</tr>
<tr>
<td><strong>Desktop2CIDR</strong></td>
<td><em>10.16.5.0/24</em></td>
<td>CIDR Block for Desktop Subnet 2 located in AZ2</td>
</tr>
</tbody>
</table>

We invoke the `cfn-init` helper script in the user data section of the properties of the instance; this code is as follows:

```
"UserData" : { "Fn::Base64" : { "Fn::Join" : ["", [     
        "<script">
        "cfn-init.exe -v -c config -s ", { "Ref" : "AWS::StackName" },
        " -r DomainController1 ",
        " --access-key ", { "Ref" : "IAMUserAccessKey" },
        " --secret-key ", "{Fn:GetAtt": ["IAMUserAccessKey", "SecretAccessKey"]},
    
```
Let's look at what is going on here so we understand what the cfn-init requires, as that determines our next steps. The cfn-init script looks for a configsets option (-c) with the name config and the logical resource ID of our domain controller. At this point, you might recall the earlier discussion about why we need to deploy the NAT instances first and have the public and private routes set up to successfully launch an instance that uses AWS CloudFormation and the CloudFormation helper scripts to configure the instance. The cfn-init helper script needs access to the Internet and the Amazon Simple Storage Service (S3) to derive, amongst other things, the AWS CloudFormation URL. The -access-key, -secret-key, and -region provided in the user data section are necessary for cfn-init to obtain the necessary information securely.

As we just defined, all the action really happens in the configset called “config” which lives in the metadata section on the instance under the AWS::CloudFormation::Init key. Let’s move on and create building blocks.

For the standalone scenario our configset contains several building blocks that we named setup, rename, dcpromo, createsite, and finalize.

For the hybrid scenario our configset contains the building blocks named setup, rename, join, dcpromo and finalize.

You will find these blocks either partially or completely reused in many of the subsequently discussed scripts.

We are using the “Setup” section to assemble any Windows PowerShell and other file types where we want to use parameters provided from the AWS CloudFormation template in our Windows PowerShell script or other small scripts.

In subsequent sections, such as the rename section, we run commands that either execute the Windows PowerShell script blocks created in the setup section or execute a single command, as in the reboot command. Executing a Windows PowerShell script is as follows:

```powershell
rename" : {
"commands" : {
"1-execute-powershell-script-RenameComputer" : {
}}
```
Now that we understand how to create files and execute commands on our Windows-based instance, it becomes fairly straightforward to understand how the next configuration steps are encoded and executed.

For the hybrid scenario in the dcpromo block of our configset, we promote the Windows Server instance to be an additional domain controller for the resource domain desktop.xencloud.net and it also serves as a DNS server that holds the global catalog.

For the standalone scenario in the dcpromo block of our configset, we will promote the Windows Server instance to be a domain controller for a new forest and it also serves as a DNS server that holds the global catalog. Through the AWS CloudFormation template, the user can choose the domain DNS name (FQDN), the domain NetBIOS name, and the safe mode admin password in both scenarios.

After that, we create our domain admin and XenApp administrator accounts.

You will find at the end of each command in our script a line that either says "waitAfterCompletion" : "forever" or "waitAfterCompletion" : "0". Why do we add these commands? AWS CloudFormation, by design, pauses for 60 seconds after execution of a command. This allows any reboots to happen before continuing, among other reasons. This should take care of the scenario where we force a reboot after the dcpromo block. However, sometimes Windows takes longer to actually perform the reboot. For this scenario, when we know a reboot has to happen, we insert "waitAfterCompletion" : "forever". For all other cases, we can speed up the execution of each command by telling AWS CloudFormation to move right on to the next step. We do this by inserting "waitAfterCompletion" : "0".

To highlight further the fine control an administrator or DevOps person has in fully configuring an Active Directory environment, we can execute a longer and more complex Windows PowerShell script that would create the physical layout of our Active Directory infrastructure as deployed in the AWS VPC by creating AD sites, subnets, and proper site links. This approach is applied in the standalone scenario, and can be customized.

Note: This script, like all the other scripts provided, is not meant to be the ultimate solution but instead is meant to provide an example for what you can do and how far you can take this when you code your scripts for your deployment. Before rolling this out into a production environment, you might want to consider, for example, providing scripts that configure group policies appropriate for your organization.

Lastly, we do a little bit of cleanup in the finalize block of our configset and add the PowerShell-ISE feature to our Windows server to have that ready for any future scripting needs when the deployed instance is in production.
Installing and configuring the second domain controller, which we deploy into a different Availability Zone, follows much the same blueprint as we established with the first domain controller. For the standalone scenario the only difference is that, for the first time, we need to provide a script to join the server to the domain first before promoting it to a replica domain controller. In the hybrid scenario with the existing On-Premises AD infrastructure in place, the second domain controller will get joined to the domain in the same way as the first domain controller. Joining a domain is done as follows (this little building block is used repeatedly in the other scripts):

1. We have to configure the network interface (NIC) on the newly launched server to point the first domain controller (AWSDSKDC01) static IP address as the DNS server. Windows provides several ways of achieving this in a scripted manner. We choose to work with the WMI object Win32_NetworkAdapterConfiguration. Our Windows PowerShell script is as follows:

   ```powershell
   "c:\cfn\scripts\SetDNSOnNIC.ps1" : {
     "content" : { "Fn::Join" : [ "\n", [
       "param($Gateway1)",
       "$NICs = Get-WMIObject Win32_NetworkAdapterConfiguration | where{$_.IPEnabled -eq "TRUE"}; "$Gateway1",
       "$NIC.SetDNSServerSearchOrder($DNSServers); "$Gateway1",
       "$NIC.SetDynamicDNSRegistration("TRUE")"
     ] ]
   },
   2. When we execute this script in our join section of the configset, we provide the private IP address of an existing domain controller as a reference to the template parameter. This code is as follows:

   ```powershell
   "join" : {
     "commands" : {
       "-set-dns-on-nic" : {
         "command" : { "Fn::Join" : [ "\n", [
           "powershell.exe " , "-ExecutionPolicy", "Unrestricted", " c:\cfn\scripts\SetDNSOnNIC.ps1 ", {
             "Ref" : "AD1PrivateIp" }
         ] ]
       },
     },
   },
   3. After this is set, we can join the domain.

   ```powershell
   "2-join-domain" : {
     "command" : {"Fn::Join" : [ "","NETDOM join localhost /Domain:", { "Ref" : "DomainDNSName" }, " /userd:",
       { "Ref" : "DomainAdminUser" },
       " /password:"],
     },
     "waitAfterCompletion" : "forever"
   },
   Now we can move on to promote the server to domain controller.

**DNS and DHCP Setup**

Before we finalize our setup, we want to spend a little time looking at the options we have available to provide DNS and DHCP services within a VPC.
Citrix XenApp on AWS: Implementation Guide

For domain name services (DNS), we already made the choice and deployed the DNS Server role on our two Active Directory servers. Those DNS servers will hold the “A” records of all the instances we have deployed in our VPC, where the On-Premise Active Directory servers hold the “A” records for our On-Premise resources. What about DHCP for dynamically configuring our devices in the VPC? We could configure the DHCP server role on our Active Directory servers but why add load to these servers when we can use a service that is available at no additional charge with every VPC deployed on AWS? For this deployment, we take advantage of the DHCP Option Set and configure it accordingly. Configuration is, like with many other resource types, a two-step process.

1. First, we define our DHCP option set and the code is as follows:

   "XenCloudDhcpOptions" : {
     "Type" : "AWS::EC2::DHCPOptions",
     "Properties" : {
       "DomainName" : { "Ref" : "DomainDNSName"},
       "DomainNameServers" : ["AmazonProvidedDNS"],
       "NtpServers" : [{ "Ref" : "AD1PrivateIp"}],
       "NetbiosNameServers" : [{ "Ref" : "AD1PrivateIp"}, { "Ref" : "AD2PrivateIp"}],
       "NetbiosNodeType" : "2",
       "Tags" : [ {
         "Key" : "Domain", "Value" : { "Ref" : "DomainDNSName"}
       }]
     }
   },

2. Then we associate our DHCP option set with the VPC.

   "XenCloudVPCDHCPOptionsAssociation" : {
     "Type" : "AWS::EC2::VPCDHCPOptionsAssociation",
     "Properties" : {
       "VpcId" : {"Ref" : "VPC"},
       "DhcpOptionsId" : {"Ref" : "XenCloudDhcpOptions"}
     }
   },

This is all we have to do to achieve dynamic host configuration and proper NetBIOS name resolution in our VPC.
At the end of this step, the following resources of the architecture should be launched:

Figure 8: VPC with AD Infrastructure
Step 3: Launch the Database Layer

This step of the advanced implementation guide describes how you can use the AWS CloudFormation service and Windows PowerShell to perform the necessary provisioning and configuration steps to deploy SQL Server 2008 R2 Standard Edition as the database for the database layer, following the recommendations from the XenApp on AWS: Reference Architecture whitepaper.

NOTE: As with other configuration steps highlighted in this article, this step is meant to be viewed as an example that shows how an administrator might want to script a deployment. While the configuration steps performed in our sample template result in a SQL Server configuration that is capable of performing well in a medium-size deployment of a XenApp server farm, adjustments may be necessary to reflect individual deployment circumstances.

Let's dive into the individual steps it takes to script the deployment of the SQL Server resources.

Creating a SQL Server 2008 R2 AMI with EC2Config service and AWS CloudFormation Helper Scripts

While the current AWS-published SQL Server 2008 R2 Standard Edition AMIs are AWS CloudFormation enabled, you can use your own AMIs. To use your Windows AMI with SQL Server 2008 R2 installed in our scripted infrastructure deployment, we have to extend that AMI so that it can use the EC2Config Service and the AWS CloudFormation helper scripts. The steps to create such an AMI are as follows:

1. Start with your fully patched and updated AMI for your region.
2. Bring up the instance as a standalone instance (anything larger than T1.Micro works).
3. Install the latest IronPython AWS Cloud Formation tools MSI package.
4. Manually reset the UserData flag in C:\Program Files\Amazon\Ec2ConfigService\Settings\Config.xml:
   a. <Name>Ec2HandleUserData</Name>
   b. <State>Enabled</State>
5. Verify that the PATH variable in Computer/Properties/Advanced System Settings/Advanced/Environment Variables has been modified to include the CloudFormation Tools directory/.. Under System variables (*NOT* User Variables), verify:

Path=C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Program Files (x86)\Amazon\cfn-bootstrap
6. Use Sysprep from within the EC2Config service to bundle everything up. Wait until the instance shows up as "STOPPED" in the AWS Management Console.

7. Create the CFN-enabled AMI image using the AWS Management Console.

Using SQL Server Template

Open up the sample XA_VPC_SQL AWS CloudFormation template file and follow along. This template creates two SQL Server Instances in our two Availability Zones (AZ1 and AZ2) ready to be used as SQL Database Mirror for the XenApp databases.

Template Customization

The sample XenApp on AWS SQL Server template allows for customization of several defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. The XenApp on AWS SQL Server parameters include the following default values.

Table 4 XenApp on AWS RA SQL Server Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>SQLInstanceType</td>
<td>c1.xlarge</td>
<td>Amazon EC2 instance type for the SQL Instance</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>SQLServer1NetBIOSName</td>
<td>AWSSQL01</td>
<td>NetBIOS name of the first SQL Server (up to 15 characters) located in AZ1</td>
</tr>
<tr>
<td>SQLServer2NetBIOSName</td>
<td>AWSSQL012</td>
<td>NetBIOS name of the second SQL Server (up to 15 characters) located in AZ2</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This is separate from the default &quot;administrator&quot; account.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td>XAAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp Admin Account. This Account is a Domain User and will also be added to the SQL DB as a member of the dbcreator role.</td>
</tr>
<tr>
<td>AZ1</td>
<td>us-east-1a</td>
<td>Name of primary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>AZ2</td>
<td>us-east-1b</td>
<td>Name of secondary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>DomainMemberSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td>SqlServerSecurityGroupID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the SQL Server Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>Infra1Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ1 you want to provision the SQL Server into</td>
</tr>
<tr>
<td>Infra2Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ2 you want to provision the SQL Server into</td>
</tr>
</tbody>
</table>
Citrix XenApp on AWS: Implementation Guide

<table>
<thead>
<tr>
<th><strong>DataDirectory</strong></th>
<th>D:\XenApp_Data</th>
<th>Location and Directory of the SharePoint Content and Configuration Databases</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LogDirectory</strong></td>
<td>E:\XenApp_Log</td>
<td>Location and Directory of the SharePoint Database Log Files</td>
</tr>
<tr>
<td><strong>SQLServerStripeVolumeSize</strong></td>
<td>10</td>
<td>Size in GB of the 6 individual stripes for the SQL Server Striped Volumes</td>
</tr>
<tr>
<td><strong>SQLServerStripeVolumeIops</strong></td>
<td>100</td>
<td>IOPS of the 6 individual stripes for the SQL Server Striped Volumes</td>
</tr>
<tr>
<td><strong>SQLAMIID</strong></td>
<td>&lt;User Provides&gt;</td>
<td>ID of the 64-bit CloudFormation enabled SQL AMI available for launch in your region</td>
</tr>
<tr>
<td><strong>AD1Privatelp</strong></td>
<td>10.16.3.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1</td>
</tr>
<tr>
<td><strong>AD2Privatelp</strong></td>
<td>10.16.5.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2</td>
</tr>
</tbody>
</table>

Provisioning our SQL Server Instances using AWS CloudFormation and Windows PowerShell

After AWS CloudFormation is enabled on the SQL Server 2008 R2 AMI, you can use the same methods and some of the building blocks already highlighted in the Step-2 scripting.

Provisioning EBS Volumes

We looked in the XenApp RA Active Directory template sample at ways to assemble Windows PowerShell script on the fly in the AWS CloudFormation template. Let’s look at ways for creating batch files in the context of the SQL Server deployment.

Our SQL Server instance has, besides the OS EBS volumes, multiple EBS data volumes attached: one for the XenApp database files and one for the log files. This follows, in part, Microsoft’s recommendations for mitigating any I/O contention. EBS volumes are just another resource, in this case of the type AWS::EC2::Volume. We want to ensure consistent SQL Server I/O performance that is in line with the I/O profile of our application. We therefore create volumes provisioned with the specific number of I/O operations per second (IOPS) that they support. You can attach these provisioned IOPS volumes to special “EBS-Optimized” instance types. For more information, see the complete list of all Amazon EC2 instance types. Currently, you can launch the following instance types as EBS-Optimized instances:

- Standard Large (m1.large)
- Standard Extra Large (m1.xlarge)
- Second Generation Extra Large (m3.xlarge)
- Second Generation Double Extra Large (m3.2xlarge)
- High-CPU Extra Large (c1.xlarge)
- High-Memory Double Extra Large (m2.2xlarge)
Citrix XenApp on AWS: Implementation Guide

- High-Memory Quadruple Extra Large (m2.4xlarge)

For the sample deployment, we have chosen a c1.xlarge instance as it is a high CPU instance type (20 EC2 Compute Units) that lends itself well to a medium-performing database server and we attach six additional EBS volumes, each provisioned with 100 IOPS. Out of these six volumes, we create two Raid 0 stripe sets: one with four disks to hold the XenApp databases, and one with two disks to hold the log files. You can expand on the concepts highlighted in this article and attach additional disks provisioned with higher IOPS, for example, to hold the tempdbs database.

To begin, we script the provisioning of the additional volumes. First, we declare our SQL Server Stripe Volume Member size and IOPS in the Parameters section:

```
"SQLServerStripeVolumeSize" : {
   "Description" : "Size in GB of the 6 individual stripes for the SQL Server Striped Volumes",
   "Type" : "String",
   "Default" : "10"
},
"SQLServerStripeVolumeIops" : {
   "Description" : "IOPS of the 6 individual stripes for the SQL Server Striped Volumes",
   "Type" : "String",
   "Default" : "100"
},
```

This allows us to adapt our IOPS and Volumesize to your respective requirements.

Second we declare SQL Server Strip Volume resources as follows:

```
"SQLServerStripeVolume1" : {
   "Type" : "AWS::EC2::Volume",
   "Properties" : {
      "Size" : { "Ref" : "SQLServerStripeVolumeSize" },
      "Iops" : { "Ref" : "SQLServerStripeVolumeIops" },
      "VolumeType" : "io1",
      "AvailabilityZone" : { "Ref" : "AZ1" }
   }
},
"SQLServerStripeVolume2" : {
   "Type" : "AWS::EC2::Volume",
   "Properties" : {
      "Size" : { "Ref" : "SQLServerStripeVolumeSize" },
      "Iops" : { "Ref" : "SQLServerStripeVolumeIops" },
      "VolumeType" : "io1",
      "AvailabilityZone" : { "Ref" : "AZ1" }
   }
},
```
Next, we attach these volumes to our instance. You can do this using the Volumes property in the Properties section of our instance and the code is as follows:

```
"Properties": {  
  "ImageId" : { "Ref" : "SQLAMIID" },  
  "InstanceType" : { "Ref" : "SQLInstanceType" },  
  "SubnetId" : { "Ref" : "Infra1Subnet" },  
  "EbsOptimized" : "true",  
  "Tags" : [ {  
    "Key" : "Name",  
    "Value" : { "Ref" : "SQLServer1NetBIOSName" }  
  } ],  
  "Volumes" : [  
    { "VolumeId" : { "Ref" : "SQLServer1StripeVolume1" },  
      "Device" : "/dev/xvdf" },  
    { "VolumeId" : { "Ref" : "SQLServer1StripeVolume2" },  
      "Device" : "/dev/xvdg" }  
  ]},
```

Then we create the batch file and the associated text file to run the DISKPART utility and this is as follows:

```
"C:\\cfn\\scripts\\StripeDisk1.txt" : {  
  "content" : { "Fn::Join" : ["", [  
    "select disk=1 ", \\
    "select partition 1 ", \\
    "delete partition ", \\
    "select disk=2 ", \\
    "select partition 1 ", \\
    "delete partition ", \\
    "select disk=3 ", \\
    "select partition 1 ", \\
    "delete partition ", \\
    "select disk=4 ", \\
    "select partition 1 ", \\
    "delete partition ", \\
    "select disk=1 ", \\
    "convert dynamic " \\
    "select disk=2 ", \\
    "convert dynamic " \\
    "select disk=3 ", \\
    "convert dynamic " \\
    "select disk=4 ", \\
    "convert dynamic " \\
    "create volume stripe disk=1,2,3,4 ", \\
    "assign letter=D ", \\
    "format fs=ntfs quick ", \\
    "exit"  
    ]  
  ]},
```

```
"C:\\cfn\\scripts\\StripeDisk1.bat" : {  
  "content" : { "Fn::Join" : ["", [  
    "diskpart /s C:\\cfn\\scripts\\StripeDisk1.txt", \\
    "exit"  
    ]  
  ]}  
}
```
This leaves us with subsequently executing the command to run the batch file and is as follows:

```
"createraid" : {
  "commands" : {
    "1-perform-StripeDisk1" : {
      "command" : { "Fn::Join" : [ "", [ "C:\cfn\scripts\StripeDisk1.bat" ] ] },
      "waitAfterCompletion" : "0"
    },
    "2-perform-StripeDisk2" : {
      "command" : { "Fn::Join" : [ "", [ "C:\cfn\scripts\StripeDisk2.bat" ] ] },
      "waitAfterCompletion" : "0"
    }
  },
},
```

And that’s all there is to automatically provisioning your SQL Server 2008 R2 Standard Edition instance with two Raid 0 stripe sets on an EBS-Optimized instance.

**Provision and Configure SQL Server for Use with XenApp 6.5**

After having provisioned the necessary volumes and joined the SQL Server instance to the domain (which follows the same steps and re-uses the code highlighted in STEP-2), we want to provision the SQL Server instance so that it:

1. Makes use of the provisioned volumes.
2. Allows for a seamless XenApp configuration experience by adding the XenApp farm administrator account to specific roles.

You use Windows PowerShell for both of these tasks.

***Changing the default database location***

As discussed in the previous section, we have provisioned two striped EBS volumes, one to hold the XenApp databases and one to hold the log files.

SQL Server Management Objects (SMO) helps us with accomplishing the tasks of changing the default database location, enabling Named Pipes for communication and auto-starting the SQL Browser service for named pipes. We also want to give the users of our scripts some control over naming of the directories that hold the databases and log files. Therefore, we allow the passing of parameters from the AWS CloudFormation template to the Windows PowerShell script. The Windows PowerShell script is as follows:

```
param (  
[string]$datalocation,
[string]$loglocation
```
You invoke this script and pass the parameters $datalocation, $loglocation, and $SQLServerName as follows, in our AWS CloudFormation script:

"4-execute-powershell-script-ChangeDefaultDBandLog" : {
    "command" : { "Fn::Join" : [ "", [ "powershell.exe \"-ExecutionPolicy\", "Unrestricted\", "C:\\cfn\\scripts\\ChangeDefaultDBandLog.ps1 -datalocation \"{\"Ref\":\\"DataDirectory\\"},\"-loglocation \"{\"Ref\":\\"LogDirectory\\"},\"-SQLServerName \"{\"Ref\":\\"SQLServer1NetBIOSName\\"}
    ]]],
    "waitAfterCompletion" : "0"
},

Adding XenApp Logins to Database roles

To enable the XenApp farm administrator to perform the farm configuration tasks seamlessly without having to do any additional SQL Server configuration, we also need to add the farm administrator to SQL Server with a login, and associate that login to the role of dbcreator and securityadmin. The methods we use for downloading the script from our Amazon S3 bucket and running the script is the same as we have used on several other locations; we skip examining these steps in detail.
Opening SQL Server Ports in the Windows Firewall

Besides opening port 1433 and port 1434 in our SQL Server security group, we also need to open those ports on the Windows firewall on the Windows instance itself. For that purpose, we construct a small batch file that calls the `netsh advfirewall` command and performs the desired action. This batch file is as follows:

```batch
@echo ======== SQL Server Ports ============
@echo Enabling SQL Server default instance port 1433
netsh advfirewall firewall add rule name="SQL Server" dir=in action=allow protocol=TCP localport=1433
@echo Enabling Dedicated Admin Connection port 1434
netsh advfirewall firewall add rule name="SQL Admin Connection" dir=in action=allow protocol=TCP localport=1434
@echo Enabling conventional SQL Server Service Broker port 4022
netsh advfirewall firewall add rule name="SQL Service Broker" dir=in action=allow protocol=TCP localport=4022
@echo Enabling Transact-SQL Debugger/RPC port 135
netsh advfirewall firewall add rule name="SQL Debugger/RPC" dir=in action=allow protocol=TCP localport=135
```

This wraps up the provisioning of our SQL Server instance so it performs well, and it enables the XenApp farm administrator to perform any XenApp configuration task without any additional SQL Server configuration.
Configuring SQL Server for High Availability

As discussed in the XenApp on AWS: Reference Architecture whitepaper, the SQL Server instance should be provisioned for high availability. Running the script we constructed in Step 3, deploying a replica of the primary SQL Server instance as secondary, and mirroring into a different subnet that resides in another Availability Zone takes care of half the steps necessary to set up a SQL Server mirroring solution. The following Citrix KnowledgeBase Article CTX111311 (Using SQL Database Mirroring to Improve Citrix XenApp Server Farm Disaster Recovery Capabilities) outlines the remaining steps to complete our setup for constructing the mirrored databases ready for use by our XenApp Farm.

Figure 9: Configuring SQL Server for High Availability
At the end of this step, the following resources of the architecture should be launched:

Figure 10: Architecture implemented at the completion of STEP-3
Step 4: Launch the XenApp Worker server layer

XenApp 6.5 is not pre-installed in any publicly available AMI. There are several ways to automate the construction of the XenApp farm and XenApp Workers.

- Citrix App Orchestration PowerShell scripts.
- AWS CloudFormation Template with MSI tools.

Citrix Service Provider Automation Pack scripts

We can make use of the so-called Service Provider Automation Pack which contains PowerShell scripts for several automated capabilities including: the set-up and configuration of a Windows 7 Desktop Experience for session-based desktops, desktop lockdown, and hands-free deployment of a complete XenApp farm.

In this approach we will use an AWS CloudFormation Template to provision an Install Server which already contains these PowerShell scripts and the needed AWS instances. The deployment of our XenApp Farm is handled by a workflow that automates the provisioning and set up of the following XenApp components: Data Collector (and backup DC), Licensing Server, and Worker Servers.

Install Server

The install server’s purpose is to host the installation media for XenApp 6.5 and to provide single location from which scripts and other utilities can be run. Generally, the Install Server is a member of the domain so before creating the Install Server, a domain controller should be available. As with the bastion host, there are two different methods to create the Install Server-- the CloudFormation script and the AWS Console. We will only describe the CloudFormation script method.

CloudFormation Script Method

To use the CloudFormation script method to build an Install Server, you will need to create an Install Server instance, a volume and a volume attachment resource. Citrix has already created an Install Server AMI in each of the regions which can be leveraged along with a snapshot which includes the XenApp 6.5 media. Before creating an instance, you will need the following information.

Open up the sample XA_VPC_ZDC AWS CloudFormation template file and follow along. This template creates the Install Server instance in Availability Zone AZ1 and two Windows Server Instances in our two Availability Zones (AZ1 and AZ2) ready to be used from our Install Server to deploy as Zone Datacollectors for our XenApp Farm.
Template Customization

The sample XenApp on AWS ZDC template allows for customization of several defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. The XenApp on AWS ZDC parameters include the following default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>XAZDCInstanceType</td>
<td>m1.xlarge</td>
<td>Amazon EC2 instance type for the XenApp Zone Data Collector Instances</td>
</tr>
<tr>
<td>XAINSInstanceType</td>
<td>m1.medium</td>
<td>Amazon EC2 instance type for the XenApp Install Server Instance</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>ServerNetBIOSNameAZ1</td>
<td>XAZDC1</td>
<td>NetBIOS name of the XenApp ZDC (up to 15 characters) located in AZ1</td>
</tr>
<tr>
<td>ServerNetBIOSNameAZ2</td>
<td>XAZDC2</td>
<td>NetBIOS name of the XenApp ZDC (up to 15 characters) located in AZ2</td>
</tr>
<tr>
<td>XAINSNetBIOSName</td>
<td>STARBURST</td>
<td>NetBIOS name of the XenApp Install Server</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This is separate from the default &quot;administrator&quot; account.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user. Must be at least 8 characters containing letters, numbers, and symbols.</td>
</tr>
<tr>
<td>XAAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp Admin Account. This Account is a Domain User and will also be added to the SQL DB as a member of the dbcreator role.</td>
</tr>
<tr>
<td>AZ1</td>
<td>us-east-1a</td>
<td>Name of primary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>AZ2</td>
<td>us-east-1b</td>
<td>Name of secondary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>DomainMembersGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td>DesktopWorkersGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Desktop Worker Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>XAZDCStripeVolumeSize</td>
<td>10</td>
<td>Size in GB of the 6 individual stripes for the SQL Server Striped Volumes</td>
</tr>
<tr>
<td>AD1PrivateIp</td>
<td>10.16.2.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1</td>
</tr>
<tr>
<td>AD2PrivateIp</td>
<td>10.16.3.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2</td>
</tr>
<tr>
<td>INSPrivateIp</td>
<td>10.16.2.9</td>
<td>Fixed private IP for the Install server located in AZ1</td>
</tr>
<tr>
<td>Infra1Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ1 you want to provision the XenApp ZDC into</td>
</tr>
<tr>
<td>Infra2Subnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Infrastructure Subnet located in AZ2 you want to provision the XenApp ZDC into</td>
</tr>
</tbody>
</table>
Citrix XenApp on AWS: Implementation Guide

We’ll be using the same approach as in our other templates to construct instances, but for the Install Server a specific AMI will be used which already contains the Citrix App Orchestrator PowerShell scripts. This Install Server will also need access to the XenApp 6.5 media.

As of this writing these are the Citrix’s Install Server AMI image ID’s (named W2KB CF Base v3.2.3):

```
"us-east-1" : "ami-deb22cb7"
"us-west-1" : "ami-885875cd"
"us-west-2" : "ami-9e0d98ae"
"eu-west-1" : "ami-3a9f944e"
"ap-southeast-1" : "ami-820945d0"
"ap-southeast-2" : "ami-3a79e800"
"ap-northeast-1" : "ami-5f01805e"
"sa-east-1" : "ami-5b29f246"
```

The AMI image ID’s of the XenApp 6.5 Media Snapshot that will be used for the XenApp 6.5 Media for the Install Server for each region is:

```
"us-east-1" : "snap-f277618d"
"us-west-1" : "snap-f96769d6"
"us-west-2" : "snap-866c68ed"
"eu-west-1" : "snap-e3ece188"
"ap-southeast-1" : "snap-e7ff5388"
"ap-southeast-2" : "snap-de4e2fee"
"ap-northeast-1" : "snap-93a202fd"
"sa-east-1" : "snap-cf7a00a7"
```

Note that you can also create your own volume based on the snapshot and use the volume to install XenApp 6.5 manually whenever required by simply mounting the resulting volume to your own server.

The following sections to the CloudFormation script deal with the creation of the Install Server, the XenApp 6.5 volume, and mount the volume on the Install Server.

```
"InstallServer":{
    "Type":"AWS::EC2::Instance",
    "Properties":{
        "PrivateIpAddress":{
            "Ref":"INSPrivateIp"
        },
        "SecurityGroupIds":{
            "Ref":"DomainMemberSGID"
        },
        "InstanceType":{
            "Ref":"XAINSInstanceType"
        }
    }
},
```
Mounting the XenApp media on the Install server can be accomplished by first creating the volume from the AWS snapshot and then attaching the volume to the Install server. First create a volume from the AWS snapshot by adding this Volume resource to the CloudFormation script:

```
"XenAppDVD":{
   "Type":"AWS::EC2::Volume",
   "Properties":{
      "Tags":[
         {
            "Key":"Name",
            "Value":"XA6.5 Installation Media"
         }
      ],
      "AvailabilityZone":{
         "Fn::GetAtt":{
            "InstallServer",
            "AvailabilityZone"
         }
      },
      "SnapshotId":{
         "Fn::FindInMap":{
            "RegionMap",
            {
               "Ref":"AWS::Region"
            },
            "XenAppSnapShot"
         }
      }
   }
}
```

Then attach the volume to the Install Server with this VolumeAttachment resource in the CloudFormation script.

```
"DVDAttachment":{
   "Type":"AWS::EC2::VolumeAttachment",
   "Properties":{
      "VolumeId":{
         "Ref":"XenAppDVD"
      },
      "InstanceId":{
         "Ref":"InstallServer"
      },
      "Device":"xvdh"
   }
}
```
As already applied in our previous scripts we will automatically join the Install Server to the domain during the instantiation process.

The template will also instantiate two Windows Server instances for each Availability Zone ready for usage by the Install Server to create as the Zone Data Collectors of the XenApp Farm.
Building the XenApp Farm with Citrix App Orchestration

The Citrix App Orchestration scripts will allow an automated build of the XenApp Farm if you choose not to build the servers manually. You will need to know the following information to build out the XenApp farm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>DomainDNSServerIP</td>
<td>&lt;User Provides&gt;</td>
<td>IP address of the domains DNS server</td>
</tr>
<tr>
<td>InstanceType</td>
<td>m1.large</td>
<td>Type of instance to create for the farm controllers; m1.large is recommended for farms with less than 20 servers not hosting user connections.</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>InstallServerPrivateIp</td>
<td>10.16.1.9</td>
<td>Fixed private IP for the install server</td>
</tr>
<tr>
<td>XenAppDVDNetworkShare</td>
<td>&lt;User Provides&gt;</td>
<td>Location of the share which contains the XenApp DVD media. This should be a UNC path with the FQDN, ie \installserver.domain.com\xa65</td>
</tr>
<tr>
<td>XenAppEdition</td>
<td>&lt;User Provides&gt;</td>
<td>The XenApp edition that corresponds to the XenApp licenses that will be installed</td>
</tr>
<tr>
<td>XenAppFarmName</td>
<td>&lt;User Provides&gt;</td>
<td>The name to be used for the XenApp farm</td>
</tr>
<tr>
<td>XenAppSQLServer</td>
<td>&lt;User Provides&gt;</td>
<td>The FQDN of the server hosting the SQL database</td>
</tr>
<tr>
<td>XenAppDBAuthMethod</td>
<td>Windows</td>
<td>The authentication method (Windows or SQL) for the XenApp server communication with the SQL server</td>
</tr>
<tr>
<td>XenAppDBCreate</td>
<td>Yes</td>
<td>Whether you want the script to automatically create the XenApp database. In case you deployed the SQL Mirror setup you will already have a database, in case this value should be set to No</td>
</tr>
<tr>
<td>XenAppDBCredentials</td>
<td>&lt;User Provides&gt;</td>
<td>The SQL connection credentials for IMA service in the format DOMAIN\Username, use here the XAAdminUser created earlier which will have the necessary rights on the SQL Server</td>
</tr>
<tr>
<td>XenAppPrimaryCollector</td>
<td>&lt;User Provides&gt;</td>
<td>The FQDN of the primary XenApp farm DataCollector located in AZ1</td>
</tr>
<tr>
<td>XenAppSecondaryCollector</td>
<td>&lt;User Provides&gt;</td>
<td>The FQDN of the secondary XenApp farm DataCollector located in AZ2</td>
</tr>
<tr>
<td>WebInterfaceInstall</td>
<td>No</td>
<td>If StoreFront will be used, this value should be No. If Yes, you will need to specify the FQDN of the Web Interface instance</td>
</tr>
<tr>
<td>SeparateXMLServer</td>
<td>No</td>
<td>Do not select separate XML servers</td>
</tr>
<tr>
<td>CitrixLicenseServer</td>
<td>&lt;User Provides&gt;</td>
<td>The FQDN of the Citrix License Server</td>
</tr>
<tr>
<td>SecureACLTools</td>
<td>No</td>
<td>Do not allow the script to secure the XenApp Tools, unless you are in a true multi-tenant infrastructure.</td>
</tr>
</tbody>
</table>

Before we can start our XenApp Farm creation process from these PowerShell scripts we need to have the following in place:

- Active Directory and DNS configured and functioning as part of Step-2.
- SQL Server setup and configured as part of Step-3. It is important to have the XenApp Datastore database created (preferably mirrored for HA purposes) and accessible.
- From the bastion host, RDP to these instances, verifying that they are configured correctly for DNS and have all joined the domain.
- From the bastion host, RDP to the Install Server.
Verify name resolution is working to the XenApp farm collectors to prevent script failure.

Once name resolution has been verified, from the Install Server, launch PowerShell and navigate to “C:\Program Files (x86)\Citrix\App Delivery Setup Tools”:

a) Save-setupconfiguration.ps1 to create the setupconfig.xml file that will be used throughout the process. This file is located in by default in %AppData%\Citrix\InfrastructureSetupConfig.xml but can be specified on the command line with “-SetupConfigFile “c:\mypath\myfile.xml””.

b) Use the values from the Parameter table to set the configuration details requested.

c) As part of the CloudFormation Template we’ve already mapped the XenApp 6.5 DVD and shared it as XA6.5. A separate PowerShell script is available in the same App Delivery Setup Tools named Install-CtxFarm-Wrapper which will map the XenApp DVD Media as well.

d) Install-CtxFarm script using information from the InfrastructureSetupConfig.xml file generated earlier. This PowerShell script will also prompt you for the correct domain name and password credentials to be used during the installation.
Citrix XenApp on AWS: Implementation Guide
AWS CloudFormation Template with MSI tools

This approach does not use an InstallServer but deploys XenApp using an unattended install with the XenApp MSI tools.

Open up the sample XA_VPC_ZDC_MSI AWS CloudFormation template file and follow along. This template creates two Windows Server Instances in two Availability Zones (AZ1 and AZ2), deploys required windows roles and features, deploys XenApp and creates the XenApp Farm.

Template Customization

The sample XenApp on AWS ZDC MSI template allows for customization of several defined parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. The XenApp on AWS ZDC MSI parameters include the following default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>XAZDCInstance1Type</td>
<td>m1.xlarge</td>
<td>Amazon EC2 instance type for the primary XenApp Zone Data Collector Instance</td>
</tr>
<tr>
<td>XAZDCInstance2Type</td>
<td>m1.xlarge</td>
<td>Amazon EC2 instance type for the XenApp backup Zone Data Collector Instance</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., xencloud.net.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>ServerNetBIOSNameAZ1</td>
<td>XAZDC1</td>
<td>NetBIOS name of the XenApp ZDC (up to 15 characters) located in AZ1</td>
</tr>
<tr>
<td>ServerNetBIOSNameAZ2</td>
<td>XAZDC2</td>
<td>NetBIOS name of the XenApp ZDC (up to 15 characters) located in AZ2</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This account is also used for the SQL database connectivity.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user.</td>
</tr>
<tr>
<td>XAFarmAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp Admin Account. This Account is a Domain User and will also be added to the SQL DB as a member of the dbcreator role.</td>
</tr>
<tr>
<td>AZ1</td>
<td>us-east-1a</td>
<td>Name of primary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>AZ2</td>
<td>us-east-1b</td>
<td>Name of secondary Availability Zone that will contain public and private subnets; select a valid zone for your region.</td>
</tr>
<tr>
<td>DomainMemberSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td>DesktopWorkersSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Desktop Worker Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>XAZDCStripeVolumeSize</td>
<td>10</td>
<td>Size in GB of the 6 individual stripes for the SQL Server Striped Volumes</td>
</tr>
</tbody>
</table>
We’ll be using the same approach as in our other templates to construct instances. This AWS CloudFormation template will also need access to the XenApp 6.5 media. See the previous Install Server chapter for the mapping of the XenApp 6.5 Media

The AMI image ID’s of the XenApp 6.5 Media Snapshot that will be used for the XenApp 6.5 Media that will be mapped to our XenApp Servers for each region are:

```
"us-east-1" : "snap-f277618d"
"us-west-1" : "snap-f96769d6"
"us-west-2" : "snap-866c68ed"
"eu-west-1" : "snap-e3ec1888"
"ap-southeast-1" : "snap-e7ff5388"
"ap-southeast-2" : "snap-de4e2fee"
"ap-northeast-1" : "snap-99a202fd"
"za-east-1" : "snap-cf7a00a7"
```

Note: you can also create your own volume based on the snapshot and use this volume to install XenApp 6.5 manually whenever required by simply mounting the resulting volume to your own server.

This CloudFormation Template follows the same approach as our other instance creations, but now also installs XenApp. But before installing XenApp it will enable the necessary Windows Features such as Remote Desktop Services and .NET Framework. See the setup section of this AWS CloudFormation Template:

```
"C:\\cfn\\scripts\\AddWindowsFeatures.ps1"{
    "content":{
        "Fn::Join":{
            """,
            ["import-module servermanager",
            "\n",
            "add-windowsfeature RDS-RD-Server, Desktop-Experience",
            "\n",
            ""]
        }
    },
}
```

The next step is to deploy the Citrix Application Delivery Tools located on the XenApp 6.5 media located in the FP2 directory.
Citrix XenApp on AWS: Implementation Guide

"C:\\cfn\\scripts\\InstallAppSetupTools.ps1":{
  "content":{
    "Fn::Join":{
      "Fn::Join":
      ["D:\FP2\CitrixAppDeliverySetupTools.exe ", "\\"
      ]
    }
  }
},

Next, configure the mf20.dsn file to point to the SQL Server database.

"C:\\cfn\\mf20.dsn":{
  "content":{
    "Fn::Join":{
      "Fn::Join":
      ["[ODBC]", "\n", "DRIVER=SQL SERVER", "\n", "DATABASE=", "\n", "DATABASE=", "\n", "SERVER=", "\n", "SERVER=", "\n", "Trusted_Connection=Yes", "\n"
      ],
      "Ref":"XenAppDatabaseName"
    },
    "\n", "UID=",
    {"Ref":"DomainAdminUser"
    },
    "\n", "SERVER=",
    {"Ref":"SQLDBNameAZ1"
    },
    "\n", "\n", "\n", "\n",
    {"Ref":"DomainDNSName"
    },
    "\n", "Trusted_Connection=Yes",
    "\n"
    }
  }
},

The next step is to launch the XenApp Console. We will log any console errors or events in c:\cflog\xainstall.txt:

"C:\\cfn\\scripts\\InstallXenApp.ps1":{
  "content":{
    "write-output "Install Script Started."
    , "D: ", "\n"
  }
},
For the first XenApp Datacollector which will be deployed in AZ1 we will have to instruct the XenApp Setup Wizard to actually create the XenApp farm, where the second XenApp Datacollector to be deployed in AZ2 will be added to the previously created farm.

See here the code setting up our first XenApp Datacollector in AZ1 using an unattended install, creating the database on the SQL Server and the XenApp farm using the previously populated mf20.dsn file.

```
"C:\cfn\scripts\ConfigureXenApp.ps1":{
  "content":{
    "Fn::Join":[
      "",
      ["C: ",
       "\n",
       "Start-process -FilePath \"C:\\Program Files (x86)\\Citrix\\XenApp\\ServerConfig\\XenAppConfigConsole.exe\\", 
       "-WorkingDirectory \"C:\\Program Files (x86)\\Citrix\\XenApp\\ServerConfig\\", 
       "-ArgumentList \"/AddLocalAdmin /AddAnonymousUsersToRemoteDesktopUserGroup /AddUsersGroupToRemoteDesktopUserGroup ", 
       "/ExecutionMode:Create /AuthenticationType:Windows 
       /logfile:c:\\cfn\\log\\xaconfig.txt /DSNFile:c:\\cfn\\mf20.dsn /FarmName:", 
       
       "Ref":"XenAppFarmName"
    ],
    "CitrixAdministratorAccount":",
    "Ref":"DomainNetBIOSName"
    ],
    "\n",
    "Ref":"XAAdminUser"
    ],
   ="/ODBCUsername":",
    "Ref":"DomainNetBIOSName"
    ],
    "\n",
    "Ref":"DomainAdminUser"
    ],
   ="/ODBCPassword":",
    "Ref":"DomainAdminPassword"
    ],
    "\n"
    ],
    "\n"
    ]
  }
},
```

For our second XenApp Datacollector which will be deployed in AZ2 we will need to join the XenApp farm using the /ExecutionMode:Join option to perform the installation as shown below. Note that we’re also specifying the /ImaWorkerMode:False option to instruct the installation procedure to create a Datacollector:
"C:\\cfn\\scripts\\ConfigureXenApp.ps1":{
  "content":{
    "Fn::Join":[
      ",
      ["c:\",
      "Start-process -FilePath "C:\Program Files (x86)\Citrix\XenApp\ServerConfig\XenAppConfigConsole.exe" ",",
      "-WorkingDirectory "C:\Program Files (x86)\Citrix\XenApp\ServerConfig" ",",
      "-ArgumentList "/AddAnonymousUsersToRemoteDesktopUserGroup",
      ",/AddUsersGroupToRemoteDesktopUserGroup",
      ",/ExecutionMode:Join /ImaWorkerMode:False /logfilename:c:\cfn\log\xaconfig.txt
      
      /AuthenticationType:Windows /DSNFile:c:\cfn\mf20.dsn",
      
      /ODBCUsername:
      
      
      /DBPassword:
      
      /ODBCPassword:
      
      /DomainAdminPassword"
    ]
  }
}
Licensing Configuration

Once the farm creation is complete, you must configure Citrix Farm licensing as follows.

1. Connect to the XenApp Datacollector in AZ1 via an RDP client as the Domain Administrator. The XenApp Server Role Manager should start automatically.

2. Select the Configure link under the License Server heading.

3. Complete the licensing configuration wizard to setup the license service. If you get a “port already in use” error, stop the Citrix licensing service (which probably started during a reboot) and finish the wizard to restart it.

4. Upload your license file for hostname XENAPP to the license server and verify that it can see the licenses.

5. Start Citrix AppCenter.
6. Configure the Unfiltered Computer policy to set the license server, license port, product edition, and product model.

At the end of this step, the following resources of the architecture should be launched:

Figure 11: Architecture after Step 4
Step 5: Launch the StoreFront Layer

- Installing StoreFront servers (one per Availability Zone) to enable load-balanced on-demand self-service to enterprise applications delivered via XenApp.
- Deploying NetScaler Gateway in front of the StoreFront servers.

The final step is deploying and provisioning the StoreFront servers and providing—using Elastic Load Balancing—the only ingress into the VPC from the Internet (besides ingress on RDP port 3389 using the Bastion Host for administrative purposes).

By now, we have executed every module that has been scripted for this deployment several times, so you should be familiar with the accompanying script. We are using the same private AMI we have created for the application server instances; we provision a volume, join the domain, and execute the installation Windows PowerShell script with a user-provided license. In the sample script, we deploy two StoreFront servers so we can demonstrate the effectiveness of the NetScaler Loadbalancing.

StoreFront Installation and Configuration

The process to install a StoreFront server, which must be done manually at this point, is detailed below.

1. Use EC2 to create an m1.large instance.
2. From the Bastion host, launch the RDP client to the StoreFront server.
3. Install StoreFrontServicesx64.msi.
Step 6: Launch and Configure the Desktop Layer.

This includes:

- Creating XenApp golden images.

- Creating XenApp worker servers based on those golden images using one of two methods:
  
  o Using the Service Provider Pack Cloud Provider Pack Windows PowerShell modules to reduce deployment time.

  o Using AWS Cloud Formation scripts.

- Configuring workloads provided by our worker servers using the XenApp AppCenter Wizard.

- Configuring StoreFront server instances using the StoreFront configuration Wizard.

- Configuring external access using the NetScaler Gateway Wizard.

- Testing your XenApp 6.5 deployment and demonstrating the facilities of the virtual desktops and apps.

- Configuring dynamic scaling and power management of a XenApp 6.5 on AWS farm.
Adding and Removing Farm Capacity

Farm capacity is defined as the number of XenApp servers that are available for a certain workload in a XenApp Farm. There are several ways to perform capacity changes. We can use the Install Server approach, use an automated MSI approach or use a Golden XenApp AMI.

Change Capacity via Service Provider Automation Pack

Let’s start with the Install Server or Citrix Service Provider Automation approach. To perform capacity changes, you will need to provision a new Windows instance that is joined to the domain with the following requirements:

- Windows Server 2008 R2 operating system must be installed.
- .NET Framework 3.5 SP1 must be installed.
- PowerShell execution policy must be set to AllSigned.
- PowerShell remoting must be enabled. For more information, see the Microsoft TechNet article “about_Remote_Requirements.”
- The servers must be joined to the same domain as the Install Server.

Now use the Add-CtxFarmCapacity or Remove-CtxFarmCapacity scripts. Before you can add capacity for a tenant, the tenant must be registered and the Active Directory and farm objects must be created.

You can create the AD structure required for the tenant by running the Register-Tenant script. This script creates the objects for the tenant if they do not already exist and adds the tenant’s worker group to the farm. The script offers some flexibility in creating the AD structure, though the easiest approach may be to specify the Tenant parameter only. The AD structure is then created directly under the domain root.

To run the script with advanced options such as User OU and Computer OU, refer to the Help. To access the Help, use a PowerShell command prompt to open the App Delivery Setup Tools folder and enter GET-HELP .\REGISTER-TENANT.PS1.

The Add-CtxFarmCapacity script uses a list of servers and the tenant’s AD information as parameters and installs and configures XenApp on these servers. After the servers are configured and joined to the farm, they are moved into the tenant’s Computer OU so that they are automatically included in the tenant’s worker group that was created during registration. The servers may not be listed immediately in the worker group and Active Directory synchronization must occur before the servers are recognized as being part of the OU.
If you need to reduce the number of servers allocated for a tenant, you can run the RemoveCtxFarmCapacity script with a list of the XenApp servers to remove. To reduce the capacity, the script removes the servers from the farm while leaving XenApp installed and moves the servers back to the Computers folder in Active Directory. After removing the server from the tenant’s farm, the server can be reallocated for other purposes or removed from the domain as well as terminated from the AWS console. See the Citrix Service Provider Automation Pack for more details.

**Change Capacity via automated MSI installation of XenApp**

As we’ve seen in the chapter “AWS CloudFormation Template with MSI tools” we can also install XenApp workers via an unattended MSI installation. And with the help of a CloudFormation template we can automate process of creating a new Windows instance that will automatically become an active XenApp Worker and member of the XenApp Farm upon launch of the instance.

**Template Customization**

The sample XA_VPC_XA_WRK_MSI AWS CloudFormation Template allows for customization of parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. The XenApp Worker Provisioning template parameters include the following default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>XAWorkerInstanceType</td>
<td>m1.xlarge</td>
<td>Type of EC2 instance to launch or XenApp Worker</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., corp.example.com.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>ServerNetBIOSName</td>
<td>XAWORKER</td>
<td>NetBIOS name of the XenApp Worker (up to 15 characters)</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This is separate from the default “administrator” account.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user</td>
</tr>
<tr>
<td>XAAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp server admin account. This account is a domain user and is be added to the SQL Server database as a member of the dbcreator role.</td>
</tr>
<tr>
<td>AZ</td>
<td>us-east-1a</td>
<td>Name of Availability Zone that will be used by your XenApp Worker - Select a valid Zone for your region</td>
</tr>
<tr>
<td>DomainMemberSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td>WorkerSecurityGroupOID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Worker Server Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>XenAppDatabaseName</td>
<td>XenApp65</td>
<td>Name for the XenApp Database to join</td>
</tr>
<tr>
<td>SQLDBNameAZ</td>
<td>SQLDBAZ1</td>
<td>Name of the SQL Database Server where the database is located</td>
</tr>
<tr>
<td>InfraSubnet1</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the XenApp Worker Server Subnet</td>
</tr>
<tr>
<td>AD1Privatelp</td>
<td>10.16.1.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1</td>
</tr>
<tr>
<td>AD2Privatelp</td>
<td>10.16.5.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2</td>
</tr>
</tbody>
</table>
Citrix XenApp on AWS: Implementation Guide

We’ll be using the same approach as in our other templates to construct instances. This AWS CloudFormation template will also need access to the XenApp 6.5 media.

The AMI image ID’s of the XenApp 6.5 Media Snapshot that will be used for the XenApp 6.5 Media that will be mapped to our XenApp Servers for each region are:

- “us-east-1” : “snap-f277618d”
- “us-west-1” : “snap-f96769d6”
- “us-west-2” : “snap-866c68ed”
- “ap-southeast-1” : “snap-e3ece188”
- “ap-southeast-2” : “snap-de4e2fee”
- “ap-northeast-1” : “snap-9a202fd”
- “sa-east-1” : “snap-cf7a00a7”

Note: you can also create your own volume based on the snapshot and use this volume to install XenApp 6.5 manually whenever required by simply mounting the resulting volume to your own server.

This CloudFormation Template follows the same approach as the creation of the XenApp Farm using MSI Tools as described in chapter “AWS CloudFormation Template with MSI tools”, but now after installation of XenApp adds the XenApp server as a Worker Server to an already existing XenApp farm.

For our XenApp Worker Server to be able to join the XenApp farm we still need to use the /ExecutionMode:Join option as for an additional XenApp ZDC, but we now need to use the /ImaWorkerMode:True option to perform the installation of a XenApp Worker Server:

```
"C:\\cfn\\scripts\\ConfigureXenApp.ps1":{
  "content":{
    "Fn::Join":[
      "",
      ["c:\nStart-process -FilePath "C:\Program Files (x86)\\Citrix\\XenApp\\ServerConfig\\XenAppConfigConsole.exe" ",
       "WorkingDirectory "C:\Program Files (x86)\\Citrix\\XenApp\\ServerConfig" ",
       "ArgumentList "/AddAnonymousUsersToRemoteDesktopUserGroup ",
       "/ExecutionMode:Join /ImaWorkerMode:True /logfilename:c:\cfn\\log\\xaconfig.txt
       "/AuthenticationType:Windows /DSNFile:c:\cfn\\mf20.dsn",
       "/ODBCUsername:",
       ","/ODBCPassword:",
       ",
       "/Ref":"DomainNetBIOSName",
       ",",
       ",
       "/Ref":"DomainAdminUser",
       ","/ODBCPassword:",
       ",
       "/Ref":"DomainAdminPassword"
    ]
  }
}
```
Another approach to creating a XA Worker server is to build the server with all the necessary applications and then seal it ready for cloning with a new identity. From there you create an AMI and instantiate instances which then are placed in the farm. The steps to do this are as follows:

a) Create the instance of Windows Server 2008 R2 through the AWS console.
   I. A minimum of an m1.large instance type is recommended, depending on the load you are expecting to have.
   II. Add to the DesktopWorker and DomainMember security group.

b) Join the Worker Server to the domain.

c) Setup XenApp.
   I. Install the XenApp Server Role (mapping a volume based on the XenApp 6.5 Media Snapshot as listed in the Farm Creation section).
   II. Configure XenApp to join a farm, and then restart the instance.
III. Install your line of business applications and configure the settings you want in your AMI.

IV. Edit your XenApp configuration and select the task **Prepare this server for imaging and provisioning**. For a command-line configuration, specify the `/ExecutionMode:ImagePrep` option.

- If you are working with an image template that you do not want to keep in the current farm, select the **Remove this current server instance from the farm** checkbox. (For a command-line configuration, use the `/RemoveCurrentServer:True` option.)

- If you are provisioning the XenApp server with SmartAuditor or other features that depend on MSMQ, selecting the **Prepare Microsoft Messaging Queuing provisioning** checkbox ensures a new unique machine identifier when the server image boots. (For a command-line configuration, use the `/PrepMsmq:True` option.)

- If you select the **Clear database location settings from this server** checkbox, the default database information is removed from local settings (server, database, and failover partner entries are removed from the DSN file; LGPO is set to NotConfigured). This ensures that cloned servers can join only a XenApp farm that is specified with inherited group policy settings. (For a command-line configuration, use the `/ClearLocalDatabaseInformation:True` option.)

**IMPORTANT:** IF YOU SELECT THIS CHECKBOX, **XENAPP ASSUMES AN ACTIVE DIRECTORY POLICY WILL PROVIDE DATABASE SETTINGS. IF A POLICY IS NOT APPLIED, THE IMA SERVICE WILL NOT START.**
d) Run the generalization tools you normally run.

e) Run the EC2ConfigService Settings program

I. Enable the Set Computer Name.

II. Enable the Initialize Drives checkbox to format the ephemeral storage upon instance instantiation.

III. Disable both the Set Password checkboxes.

IV. Select the Bundle tab and locate the Sysprep file by clicking the Details button.
V. Open the Sysprep file and edit the Microsoft-Windows-Deployment section to not scramble the password and manually set the password value
VI. After editing the answer file, click the Run Sysprep and Shutdown Now button.

f) After the image shuts down, return to the AWS Console and choose Create Image (EBS AMI) for the instance.

g) Once complete, launch new instances from the AMI.

h) Join the new instances to the domain.

i) Move the instances to the correct Worker Group OU and reboot.
Automating adding XenApp Workers to a XenApp Farm and Worker Group

We have created a XenApp Worker Provisioning AWS CloudFormation Template (XA_VPC_WRK) which will use our previously create Golden XenApp AMI to dynamically provision new instances.

**Template Customization**

The sample XA_VPC_WRK AWS CloudFormation Template allows for customization of parameters at template launch. You can modify those parameters, change the default values, or create an entirely new set of parameters based on your specific deployment scenario. The XenApp Worker Provisioning template parameters include the following default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyPairName</td>
<td>&lt;User Provides&gt;</td>
<td>Public/private key pairs allow you to connect securely to your instance after it launches.</td>
</tr>
<tr>
<td>XAWSRKInstanceType</td>
<td>m1.xlarge</td>
<td>Type of EC2 instance to launch or XenApp Worker</td>
</tr>
<tr>
<td>XAWSRkWorkerOU</td>
<td>OU=Workers,OU=Daas, DC=xencloud,DC=net</td>
<td>The Organization Unit to place our XenApp Workers into</td>
</tr>
<tr>
<td>DomainDNSName</td>
<td>xencloud.net</td>
<td>Fully qualified domain name (FQDN) of the forest root domain; e.g., corp.example.com.</td>
</tr>
<tr>
<td>DomainNetBIOSName</td>
<td>xencloud</td>
<td>NetBIOS name of the domain (up to 15 characters) for users of earlier versions of Windows; e.g., CORP.</td>
</tr>
<tr>
<td>XAWSRkWorkerNetBIOSName</td>
<td>AWSXAWSRk01</td>
<td>NetBIOS name of the XenApp Worker (up to 15 characters)</td>
</tr>
<tr>
<td>DomainAdminUser</td>
<td>XenAdmin</td>
<td>User name for the account that is added as domain administrator. This is separate from the default “administrator” account.</td>
</tr>
<tr>
<td>DomainAdminPassword</td>
<td>&lt;User Provides&gt;</td>
<td>Password for the domain admin user</td>
</tr>
<tr>
<td>XAAdminUser</td>
<td>XAFarmAdmin</td>
<td>User name for the XenApp server admin account. This account is a domain user and is be added to the SQL Server database as a member of the dbcreator role.</td>
</tr>
<tr>
<td>DomainMemberSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Domain Member Security Group</td>
</tr>
<tr>
<td>DesktopWorkerSGID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the Desktop Worker Security Group</td>
</tr>
<tr>
<td>VPC</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the VPC</td>
</tr>
<tr>
<td>DesktopSubnet</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the XenApp Worker Subnet</td>
</tr>
<tr>
<td>AD1Privatelp</td>
<td>10.16.1.10</td>
<td>Fixed private IP for the first Active Directory server located in AZ1</td>
</tr>
<tr>
<td>AD2Privatelp</td>
<td>10.16.5.10</td>
<td>Fixed private IP for the second Active Directory server located in AZ2</td>
</tr>
<tr>
<td>XAWSRkAMIID</td>
<td>&lt;User Provides&gt;</td>
<td>ID of the 64-bit CloudFormation enabled XenApp Golden AMI available for launch in your region</td>
</tr>
</tbody>
</table>
Provisioning the XenApp Server using AWS CloudFormation and Windows PowerShell

Open up the sample `XA_VPC_WRK` AWS CloudFormation template file and follow along. This template creates a XenApp Worker, based on a Golden XenApp AMI, joins this instance to the domain and places it in a specific OU of our AD to allow it to become automatically part of a Worker Group and publish applications or desktops.

This CloudFormation template is very similar to the SQL Server template used in Step-3. It however places this Windows instance in a specific OU of our AD in the following way:

```
"2-join-domain" : {
    "command" : "{"Fn::Join" : [ ", [ "NETDOM join localhost /Domain:" , { "Ref" : "DomainDNSName" }, ", /userd:", { "Ref" : "DomainAdminUser" }, ", /passwordd:", { "Ref" : "DomainAdminPassword" }, ", /OU:" , { "Ref" : "XAWorkerOU" } , ", /reboot" ]]
    },
    "waitAfterCompletion" : "forever"
},
```

Make sure that the Golden XenApp AMI being used is based on a CloudFormation enabled AMI.
Configuring XenApp XML Server and StoreFront Loadbalancing

In the GUI of the primary NetScaler, use the Load Balancing wizard for XenApp to create the vServers for our XenApp Zone Datacollectors XML Servers XAZDC1 and ZAZDC2.

Add the following XenApp objects...

- Load Balancing > Load Balancing Wizard for Citrix XenApp
  - Add XenApp1 XML server (10.17.0.25).
- Add XenApp2 XML server (10.17.0.26).
- Add Monitor (type: Citrix-xml-service, special parameter: Notepad).
- Add Services (port 443) and bind monitor.
- Note successful probe (Success - TicketTag found in the response).
- Add LB vServer (port 443, 10.17.0.48, Least Connection, no persistence) and bind XA services.

Configure NetScaler to Load Balance Storefront

In the GUI of the primary NetScaler, use the Load Balancing wizard vServers for our StoreFront Servers XAZDC1 and ZAZDC2.

First we want to create a StoreFront type monitor for each of our StoreFront servers.

1) In the navigation pane, expand Traffic Management, expand Load Balancing, and then click Monitors.

2) In the Create Monitor dialog box, in the details pane, click Add, and then, in the Create Monitor dialog box, in the Type list, select STOREFRONT.

3) On the Special Parameters tab, set the following parameters:
Citrix XenApp on AWS: Implementation Guide

- Host Name
- Store Name

Second we want to create the vServer that will loadbalance our StoreFront servers.

1) Select the Traffic Management \ Load Balancing \ Virtual Servers node on the NetScaler, and then select Add...

2) Enter StoreFront_SSL_VS for the vServer name.

3) Create a Service Group named StoreFront_SSL which will contain our two StoreFront Servers using the previously created monitors.
Citrix XenApp on AWS: Implementation Guide
Configuring StoreFront server instances using the StoreFront configuration Wizard

1. Launch the configuration console from the Start Menu: All Programs >> Citrix >> Citrix StoreFront

2. Click the “Deploy a new deployment” link.

Accept as base address the FQDN for your Loadbalanced StoreFront setup in the previous chapter as https://storefront.xencloud.net

3. Enter CorporateStore as the Store Name. Click Next.
4. Provide information on the delivery controllers.

5. Name the farm and provide the FQDN of the XenApp farm controller servers. Set the Transport type and the Port to match your environment. Click OK to finish the delivery controller configuration.
6. Once the delivery controllers are configured, click **Next**.
7. On the **Remote Access** dialog, since we will be using NetScaler Gateway on the NetScaler (either On-Premises or within AWS, we set the **Remote Access** to No VPN tunnel and click **Add** to configure the gateway.

8. Provide a **Display Name** (Go-Red Gateway), **Gateway URL** (ex: https://red.awsxencloud.net), select **Version** “10.0 (Build 69.4) or later” NetScaler Gateway Enterprise Edition, and provide the MIP of the internal subnet. Then click **Next**.

9. Provide the Callback URL and click **Next**.
10. The final step in setting up the NetScaler Gateway is to provide the FQDN of the Secure Ticket Authorities (usually the XenApp farm controllers). When finished, click **Create**.
11. Then click **Create** one more time to finish off the Store creation.
12. At this point the store is configured. Click Finish.

Now we can add the second StoreFront server to our deployment.

1. Launch the configuration console from the Start Menu: All Programs >> Citrix >> Citrix StoreFront.

2. Select the second option “Join existing server group”.

3. Enter the authorization server as CorpSF1
   Switch over to CorpSF1 server. Launch the Storefront Console. Click on the Server Group node and click Add Server under the Actions menu.

4. In the Add Server window, make note of the Authorization code. Switch back to CorpSF2 and enter the code provided, and click the Join button. This process can take up to 5 minutes.

5. Once join process is complete, acknowledge message that server is now part of multi-server group.

6. Click on the Server Group node. Verify the Synchronization Status.
Configuring external access using the NetScaler Gateway Wizard

Create a NetScaler Gateway vServer.


2. Enter the following details:

   - **Name**: RemoteAccess
   - **IP Address**: 192.168.10.70
   - **Port**: 443
   - Redirect requests from port 80 to secure port: Enabled

   ![NetScaler Gateway Settings](image)

   - **Gateway FQDN**: <YourPublicIPFQDN>
3. From the Certificate drop-down menu, select Your Certificate and click Continue.
4. Select LDAP as **Primary Authentication**, click the button for **Configure New** and enter the following details:

- **IP Address:** 10.17.0.10
- **Base DN:** `cn=Users, dc=xencloud, dc=net`
- **Admin Base DN:** `cn=Administrator, cn=Users, dc=xencloud, dc=net`
- **Password/Confirm Password:** <User Provided>
5. Enter the following details for our Windows StoreFront deployment type:

- **StoreFront FQDN**: storefront.xencloud.net
- **Receiver for Web Path**: /Citrix/CorporateStoreWeb
- **PNAgent Path**: /Citrix/PNAgent/config.xml
- **Single Sign-on Domain**: xencloud

- **STA URL**: https://corpxadc01.desktop.xencloud.net

6. Click **Done**.

7. Under **Configuration**, go to **NetScaler Gateway > Virtual Servers**.

8. Double-click the **RemoteAccess** entry.

9. Under the **Published Applications** tab, click **Add** under the **Secure Ticket Authority**.

10. Type in **https://corpxadc02.desktop.xencloud.net/Scripts/ctxsta.dll** and click **Create**.
11. Open Internet Explorer or any other browser and navigate to your public FQDN to validate the functionality of your infrastructure.
At the end of this step, all the resources of the architecture should be launched:

Figure 12: Architecture after Step 6.
Dynamic Capacity Management

This section discusses how dynamic capacity management would work within the Amazon cloud with XenApp. Items such as CloudWatch and Auto Scaling triggers can be used to Scale Up, while either integration with XenApp PCM or using AWS’s SNS can be used to shutdown unused and idle servers.

Power and Capacity Management architecture

Citrix XenApp Power and Capacity Management is designed to manage XenApp server capacity by dynamically scaling up or scaling down the number of online XenApp servers. Consolidating sessions onto fewer online servers improves server utilization, while providing sufficient capacity to handle load while minimizing consumption of unnecessary AWS resources. As users log off and idle capacity increases, idle instances on AWS will be shut down. Although XenApp Power and Capacity Management also has the capability to start additional serves when the workload increases, it does not have the capability to do this using AWS. It does have the capability to start new workload servers on the main hypervisors (Microsoft Hyper-V, Citrix XenServer or VMware ESX) as well as physical servers using Wake on LAN technology.

Scheduling provides an automated approach. An administrator defines specific times for powering on and powering off workloads. For example, a schedule powers on servers at 8 a.m. and powers servers down at 7 p.m. from Monday through Friday. XenApp PCM while working with AWS can only power servers down; hence we need to resort to another mechanism to start our instances at 8 a.m. An easy approach would be to use the Windows Task Scheduler from the PCM server that would kick off a script that would call the AWS CLI to
start our required instances. Or use AWS Autoscaling to start the instance on demand as described in the chapter CloudWatch and AutoScaling ScaleUp of XenApp.

PCM Load consolidation and power management operate in unison; load consolidation ensures sessions are not spread across online servers, which provides a better opportunity to power off excess servers later, using the power management feature of PCM

The administrator can manually override capacity and schedule settings to accommodate unexpected changes in demand.

We will use a combination of XenApp Power and Capacity Management and AWS’s AutoScaling technology. XenApp PCM (Power and Capacity Management) will be used to consolidate sessions and gracefully shut down servers, while draining user sessions from active XenApp Worker Servers.

We will use AWS’s AutoScaling and CloudWatch technology to scale up servers when workloads increase.

This combination helps optimize capacity and resource consumption for XenApp workloads on AWS.

**Auto Scaling**

Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define. With Auto Scaling, you can ensure that the number of Amazon EC2 instances you’re using increases seamlessly during demand spikes to maintain performance, and decreases automatically during demand lulls to minimize costs. Auto Scaling is particularly well suited for applications, such as XenApp, that experience hourly, daily, or weekly variability in usage. AWS Auto Scaling is however aimed at stateless short-lived sessions and as such is not able to identify and shutdown the appropriate instance without impacting active XenApp sessions (Scale Down). It can however identify with the help of Amazon CloudWatch the need for additional resources to respond to workload requests (Scale Up). Auto Scaling is enabled by Amazon CloudWatch and available at no additional charge beyond Amazon CloudWatch fees.

**Amazon CloudWatch**

Amazon CloudWatch provides monitoring for AWS cloud resources and the applications customers run on AWS. Developers and system administrators can use it to collect and track metrics, gain insight, and react immediately to keep their applications and businesses running smoothly. Amazon CloudWatch monitors AWS resources such as Amazon EC2 and Amazon RDS DB instances, and can also monitor custom metrics generated by a customer’s applications and services. With Amazon CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

Amazon CloudWatch provides a reliable, scalable, and flexible monitoring solution that you can start using within minutes. You no longer need to set up, manage, or scale your own monitoring systems and infrastructure. Using Amazon CloudWatch, you can easily monitor as much or as little metric data as you need. Amazon CloudWatch lets you programmatically retrieve your monitoring data, view graphs, and set
alarms to help you troubleshoot, spot trends, and take automated action based on the state of your cloud environment.

Combining XenApp PCM, AWS Auto Scaling and AWS CloudWatch allows us to dynamically scale up and scale down XenApp Workers.

**Provision new XenApp workers ready for Dynamic Provisioning**

The approach we’re going to take to scale up and down our XenApp workers is to use the XenApp Sealing feature as described on page 86.

We will use a CloudFormation template to place our “golden images” into the correct OU, passing the OU as a parameter in Userdata to the instances. Open up the sample **XA_WRK_AS_DYN** CloudFormation template file and follow along.

Our objective is that the new instance created or started by AutoScaling, already preloaded with XenApp including XenApp PCM agent and other monitoring tools such as EdgeSight or a Microsoft System Center and all required applications or the appropriate application virtualization components, will be placed directly into the correct OU in AD. This will allow the worker group feature of XenApp to automatically insert the new XenApp worker into the worker group and start hosting user sessions.

For this purpose we need to make a small change to our join-domain statement in the CloudFormation template, namely to include the /OU parameter as follows:

```json
"2-join-domain" : {
    "command" : {"Fn::Join" : [ "", [ "NETDOM join localhost /Domain:" , 
    " /user:" ,
    { "Ref" : "DomainAdminUser" },
    " /password:" ,
    { "Ref" : "DomainAdminPassword" },
    " /OU:"OU=Workers,OU=DaaS,DC=xencloud,DC=net,
    " /reboot" ]]
    },
    "waitAfterCompletion" : "forever"
}
```

This would place every XenApp worker into this specific OU, hence would not be very flexible to be able to deal with multiple Worker Groups. Passing the parameter via Userdata would allow use to place the instances in a specific OU. For this we need to define a parameter in the Parameter section of our CloudFormation Template which we can then use during the Resource creation.

```json
"2-join-domain" : {
    "command" : {"Fn::Join" : [ "", [ "NETDOM join localhost /Domain:" ,
    " /user:" ,
    { "Ref" : "DomainAdminUser" },
    " /password:" ,
    { "Ref" : "DomainAdminPassword" },
    " /OU:"OU=xaworkerOU",
    " /reboot" ]]
    },
    "waitAfterCompletion" : "forever"
}
```
Dealing with multiple Workergroups means that we probably have different golden images. For example, there could be a Finance Golden Image and an Office Golden Image that are the base for different worker groups.

We would therefore need to provide a facility to allow for selecting the right AMI (Golden Image). This requires setting up the mapping in the CloudFormation template as follows.

Office Worker AMI:

```
"AWSRegionArchXAOffAMI" : {
    "us-east-1" : {"64" : "ami-deb22cb7"},
    "us-west-2" : {"64" : "ami-9e0d98ae"},
    "us-west-1" : {"64" : "ami-885875cd"},
    "eu-west-1" : {"64" : "ami-3a9f944e"},
    "ap-southeast-1" : {"64" : "ami-820945d0"},
    "ap-southeast-2" : {"64" : "ami-3a79e800"},
    "ap-northeast-1" : {"64" : "ami-5f01805e"},
    "sa-east-1" : {"64" : "ami-5b29f246"}
},
```

Finance Worker AMI:

```
"AWSRegionArchXAFinAMI" : {
    "us-east-1" : {"64" : "ami-deb22cb7"},
    "us-west-2" : {"64" : "ami-9e0d98ae"},
    "us-west-1" : {"64" : "ami-885875cd"},
    "eu-west-1" : {"64" : "ami-3a9f944e"},
    "ap-southeast-1" : {"64" : "ami-820945d0"},
    "ap-southeast-2" : {"64" : "ami-3a79e800"},
    "ap-northeast-1" : {"64" : "ami-5f01805e"},
    "sa-east-1" : {"64" : "ami-5b29f246"}
}
```

We will use this selection during the instance creation.

Now let’s create an AutoScaling group that will start with 1 single Office Worker Instance using the CloudFormation CLI.

We will use the `cfn-create-stack` command with the `-template-file` option to point at our CloudFormation template. The `-parameters` option will be used to supply the required parameters.

```
PROMPT> cfn-create-stack XARef-stack --template-file XA_WRK_AG_DYN.template --parameters
"InstanceType=m1.xlarge;XAWorkerOU=OU=Workers,OU=DaaS,DC=xencloud,DC=net;KeyName=XenCloudKey;DomainDNSName=desktop.xencloud.net;DomainNetBIOSName=desktop;AD1PrivateIP=10.16.3.10;AD2PrivateIP=10.16.6.10;DomainMemberSGID=sg-66479909;DesktopWorkerSGID=sg-7a479915;DesktopSubnet=subnet-6d4bf903;StartingXAWorkers=1"
```
Our next step is to add a CloudWatch metric that will monitor the amount of ICA sessions for a given instance, and trigger the XAWorkerScaleUp1Policy when this limit is reached.

```
"XAWorkerScaleUp1Policy":{
  "Type":"AWS::AutoScaling::ScalingPolicy",
  "Properties":{
    "AdjustmentType":"ChangeInCapacity",
    "AutoScalingGroupName":{
      "Ref":"XAWorkerGroup"
    },
    "Cooldown":60,
    "ScalingAdjustment":1
  }
},

"ICAAlarmHigh":{
  "Type": "AWS::CloudWatch::Alarm",
  "Properties": {
    "AlarmDescription": "Scale-up if ICA Session > 90 for 10 minutes",
    "MetricName": "TerminalServices\Active Sessions",
    "Namespace": "AWS/EC2",
    "Statistic": "Average",
    "Period": "300",
    "EvaluationPeriods": "2",
    "Threshold": "90",
    "AlarmActions": [{ "Ref": "XAWorkerScaleUp1Policy" }],
    "Dimensions": [{
      "Name": "AutoScalingGroupName",
      "Value": { "Ref": "XAWorkerGroup" }
    }],
    "ComparisonOperator": "GreaterThanThreshold"
  }
},
```

Set Up Task Scheduler to Send Metrics Reports to Amazon CloudWatch

You can use Windows Task Scheduler to send metrics reports periodically to Amazon CloudWatch.

To set up task scheduler to send metrics reports to Amazon CloudWatch

1. On your Windows Server instance, click **Start**, click **Administrative Tools**, and then click **Task Scheduler**.
2. On the **Action** menu, click **Create Task**.
3. In the **Create Task** dialog box, on the **General** tab, in the **Name** box, type a name for the task, and then select **Run whether user is logged on or not**.
4. On the **Triggers** tab, click **New**.
5. In the **New Trigger** dialog box, under **Settings**, select **One time**.
6. Under **Advanced settings**, select **Repeat task every** and select **5 minutes** from the drop-down menu.

7. In the **for a duration of** drop-down menu, select **Indefinitely**, and then click **OK**.

8. **Note**: These settings create a trigger that will launch the script every 5 minutes indefinitely. To modify this task to run for set number of days using the **Expire** check box.

9. On the **Actions** tab, click **New**.

10. In the **Action** drop-down menu, select **Start a program**.

11. Under **Settings**, in the **Program/script** box, type **Powershell.exe**.

12. In the **Add arguments (optional)** box, type `-command "C:\scripts\mon-put-metrics-perfmon.ps1 -aws_credential_file c:\awscreds.conf -from_scheduler -logfile C:\mylogfile.log"`, and then click **OK**.

13. On the **Create Task** dialog box, click **OK**.

14. If you selected a user account to run this task, Task Scheduler will prompt you for user credentials. Enter the user name and password for the account that will run the task, and then click **OK**.

15. **Note**: If the PerfMon counters you are using don't require administrator privileges, you can run this task using a system account instead of an administrator account. In the **Create Task** dialog box, on the **General** tab, click **Change User or Group**, and then select a system account.
Scaling down – Using XenApp PCM

We can use two approaches to scale down our instances both provided by XenApp Power and Capacity Manager.

Complete the following Initial configuration tasks:

- Install the XenApp Power and Capacity Management agent on all the XenApp Servers. The best way to do this is to include the agent into the Golden Images and set a Workload per configuration. For example:
  - AWS US-West-1 - Business Essential Workload
  - AWS US-West-1 - Controllers
  - CORP RED - Controllers
  - COPR RED - Financial Apps

Connect to a XenApp Power and Capacity Management farm to manage (required only if you have more than one Power and Capacity Management farm)

- Configure server profile properties for the various workloads.
- Configure server properties such as capacities.
- Specify global configuration settings.
- Add machine managers, for the XenApp resources that are located On-Premises and are hosted on a Virtual Machine Manager such as XenServer.
• Optionally add sites, if the Power and Capacity Management farm includes XenApp servers hosted On-Premises and multiple AWS VPCs. Servers are power controlled and load consolidated according to the preference of their associated site.

After the initial setup, observe management console displays and generate reports. Using the collected information, you can then:

• Create a schedule

• Enable power management and load consolidation
Use XenApp PCM fixed schedule to drain and stop instances at for instance 9pm.

Or use XenApp PCM dynamic Load Balancing to drain servers and react dynamically.
XenApp Farm in AWS Deployment

For the purposes of this document, a fictitious corporation will be created, named XenCloud Corporation. The XenCloud Corporation is faced with all of the challenges that have been mentioned in this report. Citrix XenApp 6.5 provides the means to satisfy all of the business requirements for XenCloud Corporation. XenApp server addresses remote office connectivity, application deployment, workforce mobility, and business continuity requirements leading to a seamless, end-to-end solution.

In order to satisfy the requirements for the company, XenApp server was architected and deployed with the following core XenApp components:

- XenApp 6.5FP2
- StoreFront 2.0

Simulated environment

XenCloud Corporation is in the process of identifying a corporate solution that will satisfy the requirements and challenges of their company. Citrix products will be introduced to maximize the efforts of the company to achieve all of its corporate IT initiatives. The data compiled in this report will consist of real-time simulations and provide a valuable retrospective satisfying the infrastructure requirements of XenCloud Corporation with scalable, secure and reliable Citrix products.

The corporate infrastructure for the XenCloud Corporation is based on Microsoft products and solutions. They have implemented an Active Directory based infrastructure. XenCloud Corporation employs a hybrid administration model. The architecture group is based out of the Fort Lauderdale office. This group is responsible for company-wide IT infrastructure and the Citrix XenApp farm. They implement Active Directory group policies to meet the existing business needs and want the ability to configure and maintain the Citrix XenApp policies from the same Active Directory group policy console.

XenCloud Corporation has distributed operations with an on-premise data center located on the west coast of the United States (Seattle) and an additional IaaS AWS infrastructure US-West-1. Two data centers have been created for business continuity purposes. Fort Lauderdale, Florida hosts one, while Redmond, Washington hosts the other data center. XYZ Corporation uses an Active-Active Disaster Recovery model. Each site is fully redundant and has the capacity to service all the users for the entire organization. The data centers are responsible for serving the following business functions:

- access to mission critical applications for 10,000 corporate users (5,000 at each site)
- each site must have the capacity to service all of the users for the entire company, equating to a farm capable of servicing 10,000 users
Citrix XenApp on AWS: Implementation Guide

- access to applications for 1,000 remote users and traveling users
- access to partner applications for business partners

It is important to note that XenCloud Corporation is a fast growing company and has plans to expand their capacity at their existing site and to add additional AWS sites in the future. They want to ensure that their farm design is scalable and allows them to rapidly add new XenApp servers. The XenApp farm for the XenCloud Corporation contains three types of server images: one server image hosts business essential applications delivered to all employees, the second server image contains applications delivered to engineering, and the third server image contains applications that are reserved for the finance department.

XenCloud.net Forest Design

Figure 14: XenCloud Corporation - Active Directory Model

The Redmond and AWS US-West-1 Datacenters have separate XenApp administrators responsible for maintaining the servers in their respective zones. The local XenApp administrators are responsible for tasks such as publishing applications, rebooting servers, and monitoring servers in their zones. These XenApp administrators only have rights to administer servers in their zones.

The helpdesk is tasked with the responsibility of managing user sessions and being able to shadow in order to better support their employees.
**XenApp Configuration**

The following section outlines the architectural design for the XenApp farm. A review of the primary components of the XenApp farm has been outlined with details to its influence on the solution architected.

**Data store**

The data store is a central repository for all of the static information for the XenApp farm. This includes items like configurations, published applications and desktops, worker groups and load evaluators. Each XenApp server maintains a constant connection to the database server hosting the data store. During server startup, the IMA Service queries the data store for initialization. This is the most CPU-intensive action for the data store, as the IMA Service initialization process ensures the local host cache (LHC) is consistent with the data store. When multiple servers are booted, multiple requests for initialization information are made to the data store at the same time.

During normal farm operation, the data store is accessed every 30 minutes by each server to ensure their local host cache is current. The data store is also accessed if the farm configuration is modified or static information is requested by tools such as the Citrix AppCenter Console or other Citrix query-based utilities. However, the data store is not accessed when a user logs in, disconnects, or reconnects to the farm. All the information needed for a client to establish a connection to a XenApp server is stored in the LHC.
**How much storage is needed for the data store?**

In order to estimate the storage requirements to satisfy our configuration, it is important to have a clear understanding as to how much disk space is consumed for some common XenApp objects. This will aid in the determination of the initial size of the IMA data store to reduce the frequency of database file size increases once the farm is configured. This estimate is crucial because by reducing the frequency of database file size increases, the performance of the IMA data store will be stabilized.

<table>
<thead>
<tr>
<th>XenApp Object</th>
<th>Size Estimate (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial farm creation</td>
<td>1184</td>
</tr>
<tr>
<td>Add one server to the farm</td>
<td>88</td>
</tr>
<tr>
<td>Application or Desktop publication</td>
<td>296</td>
</tr>
<tr>
<td>- 10 hosting servers</td>
<td></td>
</tr>
<tr>
<td>- 500 users</td>
<td></td>
</tr>
<tr>
<td>- Default icon</td>
<td></td>
</tr>
<tr>
<td>Application or Desktop publication</td>
<td>40</td>
</tr>
<tr>
<td>- 10 hosting servers</td>
<td></td>
</tr>
<tr>
<td>- 5 user groups</td>
<td></td>
</tr>
<tr>
<td>- Default icon</td>
<td></td>
</tr>
<tr>
<td>Application or Desktop publication</td>
<td>24</td>
</tr>
<tr>
<td>- 32-bit color icon</td>
<td></td>
</tr>
<tr>
<td>Application or Desktop publication</td>
<td>48</td>
</tr>
<tr>
<td>- 256-bit color icon</td>
<td></td>
</tr>
<tr>
<td>Create one worker group</td>
<td>2</td>
</tr>
<tr>
<td>Create one load evaluator</td>
<td>8</td>
</tr>
<tr>
<td>Apply the load evaluator</td>
<td>16</td>
</tr>
<tr>
<td>- 10 servers</td>
<td></td>
</tr>
</tbody>
</table>

Table 5- XenApp common object sized

Our solution will create a farm with 1000 servers and 1000 applications and desktops, 100 worker groups to serve as application and desktop silos, and 10 load evaluators. To satisfy this design, the IMA data store size requirement will be approximately 500MB.

**What type of database should be used to host the data store?**

To accommodate a farm of this scale, an enterprise-capable database server should be selected. The database selection should be made based on administrative expertise among the existing farm administrators. Our solution will deploy the XenApp environment utilizing Microsoft SQL Server 2008 R2. Ideally, the database should match the expertise provided by the IT organization. For example, if a company implementing the solution has more Oracle database expertise, Oracle could have been used for the deployment.
What type of instances are needed to support 1,000 servers?

For 1,000 servers in a single farm, it is recommended that an Intel Pentium 4 class or better quad processor database server be used to host the data store. The database server should be dedicated to the data store, and should not be used for any other applications. The processing power of the database server determines the speed of administrative activities such as:

- starting IMA Service
- enumerating servers via the Citrix AppCenter Console
- adding a published application

XenApp 6.5 can efficiently handle up to 1,000 servers using Intel Pentium 4 class dual processor database servers.

The database server for XenCloud Corp has a Quad Core 1.6GHZ processor with 4GB of memory. The SQL subscriber database server is of identical specs.

Optimizing your data store performance with Session-Host mode

XenApp 6.5 introduces a new model for XenApp servers, referred to as Session-Host mode to help improve on IMA and data store performance during farm joins and LHC synchronization. Session-Host mode is targeted to assist in fast server provisioning and cloud-burst scenarios where servers need to be brought online quickly for expanding farm capacity. While not concerned with the runtime data of the entire farm, servers running in Session-Host mode act as ‘workers’ and are strictly dedicated for hosting XenApp sessions only. With new optimizations, during a join farm, IMA creates fewer objects to the data store, resulting in fewer database transactions. The downloaded LHC is limited to only their instance and does not contain the data of the rest of the farm. In return, these save on roundtrips to the data store resulting in faster IMA start times and reduced bandwidth. These savings are crucial for servers joining over a WAN or synchronizing the LHC for a single or simultaneous set of XenApp servers.
Citrix XenApp on AWS: Implementation Guide

**Servers running in Session-Host mode:**

- Primarily host XenApp sessions
- Allows member servers to quickly join a farm or synchronize the LHC from the data store
- Cannot be used as data collectors or back-up data collectors
- Cannot participate in the data collector election process
- Do not run the XML service, therefore cannot be used by Web Interface for application enumeration

By default, Session-Host mode is disabled. Configuration of a XenApp server for Session-Host mode is done at the time of joining the farm. Once the server is joined to the farm, the mode cannot be changed. The only way to change the mode of a server is to leave and re-join the farm.

For the XenCloud Corporation, each zone will have at least 2 dedicated servers reserved for the data collector and back-up data collector, while the remaining member servers will be configured in Session-host mode to speed up the IMA start times and to reduce the amount of bandwidth when communicating with the data store.
**How should the data store be deployed?**

In previous releases of XenApp, database replication was often deployed in large, enterprise deployments to decrease network traffic across the WAN and improve on IMA performance. However, by designing your farms to leverage session-host mode, the bandwidth savings and performance improvements eliminate the need for a dedicated SQL replica at each site. Instead, the SQL data store need only be deployed at the on-premises site with the other AWS remote sites connecting to it over the WAN.

The XenCloud Corporation is looking to deploy their data store at the Redmond site and have the AWS US-West-1 member servers connect to it over the WAN. **Figure 15** below illustrates the recommended data store deployment.

![Figure 15: Data store infrastructure](image)

### Data collector

The data collector is responsible for managing all of the dynamic information in the farm. Dynamic information consists of items that change often such as connected sessions, disconnected sessions, and server loads. Each zone contains one data collector. Data collectors are responsible for knowing the global state of the farm, by maintaining the list of connected sessions, not only for their own zone, but also for the entire farm. Since the global state of the farm is stored in the data collector, all changes, except for load information, that are sent to a data collector in one zone are forwarded to all other data collectors in the farm.

The other main responsibility of the data collector is to perform resolutions. A resolution is the process where the data collector determines the least-loaded server that is hosting the requested load-balanced
Citrix XenApp on AWS: Implementation Guide

Published application. If the application spans multiple zones, and therefore multiple locations, the data collector asks the other data collectors to resolve the application as well. When all responses are returned, it selects the least-loaded server, and directs the client to connect to that server.

In order to reduce the impact of performing resolutions across zones, Server group preference and failover should be configured for the users. Server group preference and failover sets an affinity based on username, client name, or client IP address to determine the zone that is optimal for the user to connect to as defined by the administrator. During resolution time, the data collector filters the list of available servers hosting the published application based on the client’s preference setting, and only performs the resolution in the primary zone. If the primary zone is not available, the client fails over to the next preferred zone.

![Data collector infrastructure](image)

**Figure 16: Data collector infrastructure**

*How many zones are needed to support 1,000 users?*

The number of zones required for a farm is dependent on the topology of the site in which the farm is being deployed.

A quick analysis of the number of servers at each site will help in the determination of the optimal configuration for combining sites into zones. Having sufficient network support to maintain the data collector replication is pertinent to the performance of the farm. If bandwidth and network reliability are a concern, a hub and spoke architecture can be used.

For optimal performance, it is best to keep the number of zones in a large enterprise farm below 5 zones. This can best be accomplished using the hub and spoke design as shown in **Figure 16**. Otherwise multi-zone farm should be considered.
Figure 17 outlines key decision points when architecting the zone design based on the topology of the infrastructure.

In the case of XenCloud Corporation, two zones of 500 servers are optimal due to the environment consisting of two distinct data centers which host an equal number of XenApp servers. The number of zones in the farm should be kept to a minimum. The fewer zones in a farm, the more scalable the farm. The reason is that every time a dynamic event occurs, such as a logon, logoff, or disconnect, an update is sent to the data collector. The data collector must then forward the update to all other data collectors in the farm, which consumes bandwidth and CPU. This is an important consideration because data collectors must keep up with the events in other zones as well as their own. Also, zones should not be based on subnets, and zones can scale beyond 500 servers.
In the near future, XenCloud Corporation plans to expand operations to a small, remote site in New York which would house 50 XenApp servers in AWS US-East-1 in the same farm with approximately 500 users. **Would the same strategy of one zone per site apply?** The following analysis will demonstrate why in this case, it would be optimal for the servers in New York to join the Corp Red Zone.

![Figure 18: Proposed corporate expansion to another zone](image)

*Let’s do the math:*

\[
(30,000 \text{ logons in the US-WEST-1 zone} + 30,000 \text{ in Redmond} \\
+ 500 \text{ logons in the US-EAST-1 zone}) = 60,500 \\
60,500 \times 0.5 \text{KB } 2 \text{ zone replication} = 60,500 \text{ of DC updates}
\]
Assume the New York location needs to be added to the current topology. In this proposed design, the bulk of the users are in Redmond and San Francisco with 5,000 users at each location. Meanwhile, the site in New York has 500 users. In this scenario, because all data collectors have to replicate all session information to all other data collectors in the farm, the US-West-1 data collector will receive 10,000 updates although it only generates 1,000 replication updates, resulting in 60.5 MB of bandwidth utilization.

Let's do the math again:

\[(30,500 \text{ users in the AWS Zone} + 50,000 \text{ in Redmond}) \times 0.5 \times 1024 \times 8\]

60,500 * 0.5KB * 1 zone replication = 30.25MB of DC updates

Figure 19: Expansion of CORP RED zone with multiple sites

If the US-West-1 and US-East-1 servers are combined in the AWS Zone, member server requests will still traverse the WAN between US-East-1 and US-West-1 for resolutions, load updates, etc., however, Redmond replication traffic will not be sent across the wire to AWS US-East. Since IMA traffic does not need to be replicated from Redmond and AWS US-West-1 to AWS US-East-1, the bandwidth consumption is cut in half.

In the event the WAN link between AWS US-West-1 and AWS US-East-1 goes down, a data collector will be elected in AWS US-East-1 and users will still be able to connect to resources both within the AWS US-East-1 and AWS US-West-1 sites. For this reason, it is important to configure at least one server in US-East-1 in non-session-host mode so it could participate in elections and designate itself as the data collector for the AWS US-East-1 site. When the WAN link is restored the zone will consolidate back down to a single data collector.

*Is a backup data collector needed for this farm?*

In order to satisfy the business requirements for XenCloud Corporation it is important to have a dedicated backup data collector for each zone and at each site. In the event the primary data collector goes offline, a new dedicated server is available to assume the data collector role. If the data collector role is assumed by a server that is not dedicated, resource contention between application users and the data collector
operations can result in data collector events getting queued. This occurrence can have a ripple effect to the rest of the data collectors in the farm as updates are not sent and received properly.

What type of instance type is recommended for the data collector?

The data collectors store all dynamic information in memory; therefore, it is important that the data collector has enough RAM to store all of the records. Memory usage will vary based on the number of published applications, number of servers and number of user sessions in the farm. The CPU plays an important role in determining the number of resolutions the data collector can process in conjunction with managing dynamic information. The data collectors for the XenCloud Corporation are built using a C1 High-CPU Extra large instance with 8 Cores with 7GB of memory.

![IMA Memory Consumption (Apps Delta)](image)

Figure 20: IMA memory consumption for number of applications
It is important that all data collectors in the farm are sized to accommodate the largest zone. Since data collectors must manage the global state of the farm, they require the same processing capability of the other data collectors in the farm regardless of the size of their particular zone. Likewise if the data collector needs to be dedicated for one zone, all data collectors in the farm should be dedicated.
XML server

The XML service is a component of XenApp that runs on all XenApp servers not configured in session-host mode. The XML service communicates published application information to the StoreFront servers. When a user connects and logs on to StoreFront, they will be presented with a list of published applications in the web browser or their Citrix Receiver as provided by the XML service. When the user selects one of the published applications, the XML service will then respond with the address of a server on which the application is published.

*Is a dedicated instance needed for the XML server?*

The XML Service must contact the data collectors when enumerating and resolving published applications. To reduce this network communication, the XML service should be configured on the data collector. In the event the primary data collector, and thus XML server go offline, a backup XML server should also be configured. For XenCloud Corporation, the XML service was placed on each of the data collectors, and the backup XML service was placed on the backup data collectors for each zone.

Additionally, the StoreFront site can be configured to allow for XML load balancing in which multiple XML servers will be used for enumeration to offset any load being put on the XML service.

License server

The license server is responsible for storing and managing Citrix licenses. The first time a client device connects to a XenApp server, the XenApp server will check out a license from the license server on behalf of the client device. Subsequent connections from the same client device share the same license.
How many license servers are necessary?

A single license server can adequately handle the load placed on it by a thousand servers and tens of thousands of users. Multiple license servers can also be deployed for a single farm. However, the drawback to having multiple license servers is that licenses are not shared between license servers.

Where in the farm should the license server be located?

The license server should be placed at the site that hosts the most users. Whenever a client connects to a XenApp server, a connection license is retrieved from the license server. Figure 23 shows that a single license server has been placed in the Redmond zone, the Citrix XenApp Servers in the AWS US-West-1 region need to request licenses across the WAN.

What type of instance type is recommended for the license server?

One of the most important considerations in determining license server requirements is the processor speed of the hardware running the license server. Although CPU usage is not usually high, CPU time increases as license check-out requests are made and License Management Console activity increases. The time it takes to execute these transactions is dependent on the speed of the CPU. In general, the size of the farm and the number of simultaneous client connections dictate the power of the server needed for the licensing feature.
To appropriately size the license server it is important to determine the number of client logins per second in the farm deployment. This can be accomplished using the Performance Monitor counters available within XenApp and the load evaluator logging feature. This analysis will determine the processor speed needed for optimal license server performance.

Another consideration when sizing the license server is that multiple processors do not provide performance increases because the license server process is single threaded. CPU speed is the most important aspect to consider for sizing the license server. The license server uses approximately 4.5KB of memory for every session license and 39KB of memory for every start-up license that is in-use. The license server is capable of processing 248 license check-out requests per second. In a given scenario with all users logging in over the course of 30 minutes, a single license server would be able to handle 446,400 users.

**How much bandwidth is used during license consumption?**

The following describes the communication paths between the XenApp server and the license server and the associated bandwidth utilization.

When a XenApp server is brought online, it establishes a static connection to the license server and checks out a Citrix startup license. This action consumes 1.68 KB of bandwidth and occurs for every server in the farm. Once a startup license is checked out, the server holds this license until it is taken offline or the license server location is changed.

When a client logs in, the XenApp server requests a license from the license server on behalf of the client. The amount of bandwidth consumed for a license check-out request or check-in request is 1.04 KB.

Every 5 minutes, each XenApp server exchanges a heartbeat with the license server to determine if the license server is still available. The amount of bandwidth in this transaction is 366 bytes for each server. The timing of the heartbeat is based on the start time of the IMA Service.

Therefore, when deploying a license server, it is important to know and understand the communication paths and bandwidth costs associated with licensing, especially when communication is over a WAN.

**Is the license server a single point of failure?**

If a XenApp server cannot contact the license server, the servers enter into a grace period and will allow user connections for 720 hours (30 days). After the 720 hour grace period expires, and the XenApp server still has been unable to contact the license server, connections are denied. For XenCloud Corporation, the 720 hour grace period provided more than enough time for recovery in the event of a failure. However, for some environments, this is not adequate. A hot or cold standby of the license server can be built into the farm. If the license server goes offline, the administrator can bring up a backup license server. In the cases where failover with no administrative interaction is required, Microsoft Clustering Services can be used On-Premises in order to deliver hardware-based fault tolerance for the license server. For more information on setting up the license server in a clustered environment, please refer to the [Citrix Knowledge Base](https://support.citrix.com) or the Citrix Production Documentation Library [eDocs](https://docs.citrix.com).
Citrix XenApp on AWS: Implementation Guide
Citrix AppCenter Console

The Citrix AppCenter Console, as shown in Figure 24, provides XenApp administrators the ability to manage users, published applications, create worker groups, and perform a variety of other tasks associated to the XenApp farm from a common user interface. The console gathers farm information from two main sources:

- the data store is used to collect static information
- the data collector is queried to assemble dynamic information, such as user sessions

![Citrix AppCenter Console](image)

**Figure 24: Citrix AppCenter Console**

*How should the farm be managed from remote locations?*

The Citrix AppCenter console can be run from a XenApp server in the farm or from a standalone computer. If an administrator would like to administer the farm from a remote location, it is a best practice to configure the Citrix AppCenter Console as a published application or remotely to a server running local to the location of the data store. By connecting to the console through an ICA session, the static and dynamic information is
queried from the data store locally, dramatically increasing the performance of the console. This is particularly useful in larger server farms where the data store contains larger amounts of objects.

In the XenCloud Corporation, the AppCenter console should be published on one of the XenApp servers in the Redmond zone and remote administrators from the San Francisco offices should access the console over a published ICA session.

*Can the AppCenter Console accommodate a hybrid administration model?*

The XenCloud Corporation uses a hybrid administration model. Basically, various XenApp administrators have specific delegated authority on the XenApp farm. Administrators in Redmond manage farm infrastructure settings for all servers. The administrators local to the site handle the day-to-day server operations and maintenance. Delegated administration is then used to set granular permissions at the folder level in the Citrix AppCenter Console. For the purposes of the XenCloud Corporation, servers are placed in folders based on the site in which they reside (i.e., all Redmond servers are placed in the “Redmond” folder and all AWS US-West-1 servers are placed in the “AWS\US-West-1” folder). San Francisco administrators are allowed custom permissions to administer all servers in the San Francisco folder with the exception of the infrastructure settings. Helpdesk administrators only need the ability to manage sessions for the farm, so they are granted view-only permissions for all nodes except the ones allowing session management.

The XenCloud Corporation plans to deploy a dedicated StoreFront instance to each of the zones and sites in the farm. In the case the WAN link goes down, users in the remote sites will still be able to access their StoreFront sites locally and access their applications.

*How many StoreFront servers are needed to support 5,000 users?*

The StoreFront instance used in this simulation is a C1 High-CPU Extra Large (c1.xlarge) with 7GB of RAM. As shown in Table 6, this instance was able to handle 9.2 user requests per second. A StoreFront server was placed at each site to service the 5,000 internal users at each location.

<table>
<thead>
<tr>
<th>Platform</th>
<th>requests / second</th>
<th>User scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2008 R2 / IIS7</td>
<td>9.2</td>
<td>33,700 users / hour</td>
</tr>
</tbody>
</table>

Table 6 - StoreFront user scalability

In general the number of users that a single StoreFront server can support is dependent on the processor speed rather than the number of processors in the system.

**Worker group configuration**

The use of worker groups adds powerful administration capabilities for XenApp administrators and can easily integrate with Active Directory (AD). All user and server settings can be managed through AD policies, while applications and load balancing can be managed through a container known as a worker group.
A worker group is simply a collection of XenApp servers in the same farm. Worker groups allow a set of similar servers to be grouped together and managed as one. Worker groups are closely related to the concept of application silos however, they streamline the creation of application silos by providing a way to synchronize the published applications and server settings across a set of XenApp servers.

Workers groups and the integration with Active Directory greatly simplify farm management by streamlining application publishing, providing granular control of load balancing, and allow management of server settings across different groups of servers in the farm.

The worker group container offers the following benefits:

1. A single server may belong to multiple worker groups. Unlike server folders, where a server can only belong to a single folder. Now servers can be grouped into worker groups for multiple reasons. For instance, servers may be grouped into worker groups both by their geographic region and by the applications they host.

2. Worker groups are more granular than zones. Worker groups can be created to control load balancing within a single site. A worker group may even consist of a single server.

3. Worker groups can be dynamic. For example, when AD containers are added to a worker group, changes in the AD container are automatically reflected in the server's worker group memberships.

There are two ways to add servers to a worker group:

1. Servers may be explicitly added to a worker group by name. This allows administrators to add specific servers to a worker group and is the only option in non-AD environments.
Citrix XenApp on AWS: Implementation Guide

Servers may be added by AD Organizational Units or Server Groups. This allows worker groups to be dynamically updated based on the servers AD memberships. That is, as servers are added or removed from the AD containers, they will be automatically added or removed from the respective worker groups.

For this simulation, the XenCloud Corporation administrators have chosen to manage their XenApp farm through Active Directory. Figure 26 shows the Active Directory Organizational Unit (OU) for the XenApp farm and grouping structure of the servers.

In preparation for future expansion, the XenCloud Corporation has created two sets of worker groups: one set to group servers by application and one set to group servers by geographic location. Given this approach, when the company adds a new site, they do not need to modify the servers list of all published applications. Instead, they simply add the site's OU to the appropriate worker groups for the applications. Figure 27 and Table 7 illustrate XenCloud Corporation's worker group structure.
Worker Groups

Apps
- Business Essential Apps
- Engineering Apps
- Financial Apps
- Office Apps

Controllers
- AWS US Controllers
- CORP RED - Controllers

Sites
- AWS US-East-1
  - AWS US-East-1 - Business Essential
- AWS US-West-1
  - AWS US-West-1 - Business Essential
  - AWS US-West-1 - Engineering
  - AWS US-West-1 - Finance
- Redmond
  - CORP RED - Business Essential
  - CORP RED - Engineering
  - CORP RED - Financial

Figure 4 - Worker group layout
The following sections will show these worker groups are integrated with three XenApp features: application publishing, load balancing policies, and Citrix policy filters.

**Application publishing**

Each published application in XenApp contains a list of servers hosting that application. XenApp 6.5 supports adding worker groups to the application server list, which greatly simplifies management of application silos and capacity management.

Without using worker groups, managing a silo of servers requires ensuring each application in the silo is published to all servers for that silo. For example, Figure 28 illustrates the application/server mappings of a 5-server silo hosting Microsoft Office applications.
In this case, each of the five servers has to be added to the servers list of each of the Microsoft Office applications. If a new server is brought online, the application’s server list would need to be updated manually to account for the new server. However, this deployment can be simplified using worker groups. Instead of publishing each application to each server, a worker group is created containing the servers hosting the Microsoft Office applications. Instead of adding individual servers, the worker group is added to the servers list of each of the applications.

In the future, to increase capacity in the application silo, a new instance is added to the worker group. This eliminates the need to manually modify the properties of each published application hosted by the server. With dynamic provisioning, this step can even be automated using AD, by creating a base image for each application silo containing XenApp and all of the applications installed. To add capacity, simply create a new instance of the base image and add it to the desired OU. The server will receive its server settings from AD, join the appropriate worker groups, and begin hosting published applications.
**Load balancing based on worker groups**

Users are no longer required to have access to all servers that host the application and may be restricted to a subset of servers using load balancing policies. In many circumstances, this change eliminates the need to publish multiple copies of the same application.

At XenCloud Corporation, the administrators created three worker groups specifically for publishing applications: the Business Essential Apps, Engineering Apps and Financial Apps worker groups. The administrators configured the Business Essential Apps worker group to contain all the servers from the Business Essential, engineering and financial silos. With previous releases of XenApp, the administrators would have had to publish separate copies of all Business Essential applications, one for the engineering users, one for financial user and another one for all others, because the users use different servers. However, with XenApp 6.5, the administrators can publish the Business Essential apps to all servers and use load balancing policies to direct users appropriately.

<table>
<thead>
<tr>
<th>Application</th>
<th>Server</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Essential Applications</td>
<td>Worker Groups\Apps\Business Essentials Apps</td>
<td>XenCloud\All Employees</td>
</tr>
<tr>
<td>Engineering Applications</td>
<td>Worker Groups\Apps\Engineering Apps</td>
<td>XenCloud\Engineers</td>
</tr>
<tr>
<td>Financial Applications</td>
<td>Worker Groups\Apps\Financial Apps</td>
<td>XenCloud\Finance</td>
</tr>
</tbody>
</table>

Table 8 - Worker groups

Creating separate worker groups for application publishing gives XenCloud Corporation the flexibility to expand their farm in the future. To add additional capacity to the existing sites, XenCloud Corporation can simply add new servers to the appropriate OUs. When they expand their farm to another site, like extending to an additional AWS Region they can create OUs for the new site, and add these to the two worker groups above. There is no need to change individual application settings.
When the administrator selects the checkbox “Configure application connection preference based on worker group” in a load balancing policy, they can configure a prioritized list of worker groups. When a user defined by the policy launches a published application, load balancing will return servers in the order of the priorities configured. Servers at a lower priority level will only be returned if all servers at a higher priority level are offline or fully-loaded (10,000 load).

This feature replaces the Zone Preference and Failover feature in previous releases of XenApp with the following major differences:

1. This feature is not tied to zones. While worker groups may be created based upon sites and contain the same servers as a zone, worker groups may also be more granular than zones.

2. Unlike the Zone Preference and Failover feature in previous releases, users are not directed to servers in worker groups that are not included in the worker group preference list, even if all servers in the preference list are unavailable.
Citrix XenApp on AWS: Implementation Guide

Load balancing policies are evaluated when a user logs in to StoreFront or refreshes applications in the Citrix Receiver. For performance, the resultant settings are then cached on the StoreFront server or with the user’s Citrix Receiver and used during each application launch.

In the case where multiple load balancing policies apply to a single user, the worker group preference list from the highest-priority policy will be used. Only servers in this preference list will be returned by load balancing. XenApp will not consider preference lists from lower-priority policies in the load balancing calculations.

At XenCloud Corporation, the administrators must ensure that engineering and financial users are always load balanced to one of the Engineering or Financial servers close to their offices and that all users are load balanced to servers at the nearest site and fail over to another site if the nearest site becomes unavailable.

For this example, the site is selected by IP range:

- 10.17.0.0/16: REDMOND
- 10.16.0.0/16: AWS US-WEST-1
- 10.18.0.0/16: AWS US-EAST-1

The administrators then configure the Load Balancing Policies for engineering users:

<table>
<thead>
<tr>
<th>Policy Name</th>
<th>Priority</th>
<th>Policy Filter</th>
<th>Worker Group Preference</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORP-RED - ENG</td>
<td>1</td>
<td>Client IP: 10.17.0.0/16</td>
<td>1. Corp-RED - Engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User: XenCloud\Engineer</td>
<td>2. AWS-US-West-1 - Engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS-US-West-1 - ENG</td>
<td>2</td>
<td>Client IP: 10.16.0.0/16</td>
<td>1. AWS-US-West-1 - Engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User: XenCloud\Engineer</td>
<td>2. Corp-RED - Engineering</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORP-RED - FINANCE</td>
<td>3</td>
<td>Client IP: 10.8.0.0/16</td>
<td>1. Corp-RED - Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User: XenCloud\Finance</td>
<td>2. AWS-US-West-1 - Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS-US-West-1 - FINANCE</td>
<td>4</td>
<td>Client IP: 10.16.0.0/16</td>
<td>1. AWS-US-West-1 - Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User: XenCloud\Finance</td>
<td>2. Corp-RED - Finance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CORP-RED - BUSINESS</td>
<td>5</td>
<td>Client IP: 10.17.0.0/16</td>
<td>1. Corp-RED - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. AWS-US-West-1 - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. AWS-US-East-1 - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. AWS-US-East-1 - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Corp-RED - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. AWS-US-East-1 - Business Essential</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Corp-RED - Business Essential</td>
</tr>
</tbody>
</table>

Table 9 - Load balancing policies and associated priorities

As shown in Table 9, users will receive the worker group preference list from the highest-priority policy with matching filters. Therefore, engineering users from the 10.17.0.0/16 IP address range will receive the CORP-RED - ENG policy while non-engineering users will receive CORP-RED - BUSINESS policy.
Citrix XenApp on AWS: Implementation Guide

With this configuration, XenCloud Corporation can deliver separate sets of applications to the two different groups of users within the company and also ensure proper failover if a failure at one site occurs.

Citrix policy filters

All Citrix server policies can be filtered by worker groups, which allow administrators to restrict GPOs to a specific set of servers in the farm. For policies configured in the AppCenter Console, this is the only way to assign different settings to different groups of servers, since all policies are replicated to all servers, completely independent of AD.

Since XenCloud Corporation administrators have control over their XenApp OU, they use AD GPOs to manage the settings in the XenApp farm. For user and per-site settings, they can link the GPO to the appropriate site OUs without any filters. However, if they wish to deploy a setting specifically to the engineering servers or productivity servers, they can add a Worker Group filter to the policy to limit it to the appropriate server type.

Citrix policy configuration

Nearly all server, farm, and user settings are governed by Citrix group policies, which can be configured in three different ways:

1. Local Machine Policy (gpedit)
2. Active Directory Group Policy (gpmc)
3. Policies node of the Citrix AppCenter Console

The local machine policy can be used for managing small farms, but large farms will use either AD or the AppCenter Console to manage settings across multiple servers. AD offers the most powerful solution for administrators and supports managing settings across multiple XenApp and XenDesktop farms. Administrators create a GPO containing the desired Citrix policy settings and link the GPO to the appropriate OUs. However, for Citrix administrators who do not have control over their AD environment, XenApp 6.5 provides the policies node in the management console. Policies configured here are written to the XenApp data store and propagated to all servers in the farm.

It is recommended to use smaller policies to allow for incremental updates. If Farm group policy is used, makes sure there is a replicated IMA data store on the remote site.

If multiple types of policy are created, the priority of policy enforcement (from low to high) is as follows

- LOCAL GPO
- FARM GPO
- DOMAIN GPO
XenCloud Corporation has decided to take full advantage of the AD group policies. The administrators create Citrix policies at different domain and OU structure levels. In this case, the priority of policies enforcement is as follows

1. Policy created at the Default Domain Policy
2. Policy created at the top OU level
3. Policy created at the middle level OU
4. Policy created at lowest level OU

As displayed in Figure 31, the administrators at XenCloud Corporation create the Citrix group policies at different XenApp OU structure levels. The “XenApp General” is created to apply to the XenApp farm as a whole. The Redmond Site GPO is created to apply to XenApp servers at the Redmond location. The Business App GPO is created to apply to the servers that host the business critical applications. The resultant Citrix policies applied to the Business Essential Apps OU will be the merged settings from all three GPOs. If there is a policy settings conflict among these three GPOs, the settings in Business App GPO has the highest priority and will overwrite the settings in Redmond Site GPO and XenApp General GPO.

**Policy refresh interval**

It is very important to understand how the configured policies are refreshed and applied to the XenApp server. This can help the Citrix administrator to troubleshoot the policy related issues.
When Citrix policies are managed from the AD domain group policy, the sequence of policy refresh and update is as follows:

1. change is made on the GPMC
2. within 1½ to 2 hours, member servers pull and apply updates
3. every 3 hours, AD replication occurs between domain controllers
4. within 1½ to 2 hours, remote member servers pull and apply updates

![Figure 32: Group Policy Management refresh intervals as managed by Active Directory](image)

When Citrix policies are managed from the XenApp management console (e.g., AppCenter Console), the sequence of policy refresh and update is as follows:

1. change is made in AppCenter Console
2. member server writes the policy change to the DS and updates its LHC
3. all servers pull policy information from the DS and updates their LHCs
4. within 1½ to 2 hours, member servers apply updates to the registry
Figure 33: Group Policy Management refresh intervals as managed by the DSC

If needed, IMA service can be restarted to refresh machine policies immediately. For user policies, logons and reconnects refresh them immediately. The `gpupdate /force` command can be executed to force the policy synchronization and update.

Further Reading

- Microsoft on AWS:

- Amazon EC2 Windows Guide:

- Microsoft AMIs for Windows and SQL Server:
  - [http://aws.amazon.com/windows](http://aws.amazon.com/windows)

- AWS Windows and .NET Developer Center:
  - [http://aws.amazon.com/net](http://aws.amazon.com/net)

- Whitepapers:
• Citrix XenApp on AWS Reference Architecture

• Scalability and Economics of XenApp on AWS -

• StoreFront Planning Guide -
  http://support.citrix.com/article/CTX136547

• Using SQL Database Mirroring to Improve Citrix XenApp Server Farm Disaster Recovery Capabilities -
  http://support.citrix.com/article/CTX111311

• XenApp 6.5 Enterprise Scalable XenApp Deployments -
  http://support.citrix.com/article/CTX131102

• High Availability for Citrix XenDesktop and Citrix XenApp - Planning Guide -
  http://support.citrix.com/article/CTX134979

• XenDesktop and XenApp Best Practices -
  http://support.citrix.com/article/CTX132799

• Advanced Farm Administration with XenApp Worker Groups -
  http://support.citrix.com/article/CTX124481

• Planning Guide - Citrix XenApp and XenDesktop Policies -
  http://support.citrix.com/article/CTX134081

• High Availability for Citrix XenApp - Reference Architecture -
  http://support.citrix.com/article/CTX131762

• Logon Optimization Guide - XenApp/XenDesktop -
  http://support.citrix.com/article/CTX128277

• Operations Guide - Monitoring Citrix Desktop and Datacenter -
  http://support.citrix.com/article/CTX133540

• Operations Guide - Support and Maintenance Citrix Desktop and Datacenter -
  http://support.citrix.com/article/CTX133786

• XenApp 6.x (Windows 2008 R2) - Optimization Guide -
  http://support.citrix.com/article/CTX131577
• Security Guidelines for Virtual Desktops -
  http://support.citrix.com/article/CTX134780

• Installing CloudBridge VPX on AWS -
  http://support.citrix.com/article/CTX136046

• Providing SSO to Amazon EC2 Apps from an On-premises Windows Domain -

• Secure Microsoft Applications on AWS-
# CloudFormation Templates

Several CloudFormation Templates have been used throughout this document. They can be found at the following locations:

<table>
<thead>
<tr>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://s3.amazonaws.com/cf-XenApp/RA/XA_VPC_DC.template">https://s3.amazonaws.com/cf-XenApp/RA/XA_VPC_DC.template</a></td>
<td>Creates Active Directory Infrastructure for the hybrid scenario referring to an existing on-premise AD infrastructure. Requires two AZs</td>
</tr>
</tbody>
</table>
About Citrix

Citrix (NASDAQ:CTXS) is the cloud company that enables mobile workstyles—empowering people to work and collaborate from anywhere, easily and securely. With market-leading solutions for mobility, desktop virtualization, cloud networking, cloud platforms, collaboration and data sharing, Citrix helps organizations achieve the speed and agility necessary to succeed in a mobile and dynamic world. Citrix products are in use at more than 260,000 organizations and by over 100 million users globally. Annual revenue in 2012 was $2.59 billion. Learn more at www.citrix.com.

Copyright © 2013 Citrix Systems, Inc. All rights reserved. Citrix, XenApp, NetScaler, CloudBridge and Citrix Branch Repeater are trademarks of Citrix Systems, Inc. and/or one of its subsidiaries, and may be registered in the U.S. and other countries. Other product and company names mentioned herein may be trademarks of their respective companies.