Managing Your Microsoft Windows Server Fleet with AWS Directory Service

May 2015

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

Whether on-premises or in the cloud, managing a large Windows Server fleet can be challenging. Active Directory addresses many of the challenges by centralizing credentials, enforcing server configurations, and more. With the launch of the AWS Directory Service, you can connect your existing Active Directory domain to the AWS cloud using AD Connector or launch a new standalone domain in AWS using Simple AD directory.

This white paper will describe how AWS Directory Service and Amazon EC2 API Simple Systems Manager (SSM) can be used to manage your Windows Server fleet in Amazon EC2.

Introduction

Using a directory like Microsoft Active Directory simplifies tasks related to credential management and server configuration. It provides a centralized repository to store credentials, which enables single sign-on (SSO) across all servers, as well as intercommunication between servers. With Group Policy objects (GPOs) in Active Directory, you can manage different configuration options for thousands of servers.

This paper provides an overview of the AWS Directory Service and describes how to join Amazon Elastic Compute Cloud (Amazon EC2) instances to an AWS domain.

Challenges of Managing a Server Fleet

Whether on-premises or in the cloud, managing fleets of servers can be challenging. Each server has a unique set of credentials and keeping track of which credentials match which server is cumbersome. Intercommunication between members of the fleet requires storing or entering credentials for servers that need to communicate with each other, and updating credentials means touching every server where the changed credentials are stored. Configuring each server is also a challenge. Although some configuration options might be consistent across all servers, other options may apply only to a server role. Ensuring the correct configuration is applied requires you to log on to each server with unique credentials to validate the settings.

These challenges do not go away when you run your fleet on Amazon EC2. In fact, the AWS pay-for-what-you-use model may make managing the fleet more complicated as instances are dynamically added and removed. Adding a few instances to a group of servers already running could require you to log into each instance to store credentials for server communication. For example, a password change for the credentials used to communicate with SQL Server means downtime until the stored credentials are updated. Even a simple configuration change, like altering the size of the event logs, requires you to manually log in to each instance to make the configuration change. Regardless of the size of the fleet, mistakes from manual configuration changes and downtime can negatively impact user satisfaction. The process to ensure your instances are configured

correctly and your credentials are documented must be rigorously followed because a lapse could mean application failures and frustrated users.

Active Directory helps address some of these challenges. It provides a centralized repository to store credentials, allowing for SSO across all servers in the fleet and simplifying intercommunication between servers by using Kerberos to authenticate requests. With GPOs, you can manage the configuration options of your fleet. Deploying a new Active Directory domain or extending your existing domain to the AWS cloud and running instances on Amazon EC2 provides great benefits. You can have SSO using Kerberos, leverage Group Policy to manage the configuration of the Windows operating system, and easily manage application and user credentials with native Windows tools like Active Directory Users and Computers.

There are concerns to address when deploying an Active Directory forest on Amazon EC2, for example, how to manage the additional domain controller instances, DNS resolution for the Active Directory domain, and how to monitor replication traffic between domain controllers in different Availability Zones. One of the biggest challenges is joining the Windows Server instances to the Active Directory domain, because you must first use the Amazon EC2 key pair to individually decrypt the administrator password for all of the instances. This process is manual, time-consuming, and error-prone. For large-scale server additions, the domain join step can be automated. For example, you can place a PowerShell script in the "User Data" option to join the instance to the domain during launch, but you will have to store the Active Directory credentials where a script running on a newly launched instance can read them and risk exposing powerful credentials. AWS Directory Service and SSM make joining your instances to a domain a low-risk, quick process.

AWS Directory Service Overview

There are two AWS Directory Service products: AD Connector and Simple AD. AD Connector lets you use your existing identities with AWS services without replicating them to AWS. Simple AD lets you create a new Active Directory-compatible directory in AWS with deep integration into AWS services. There is no software to install; AWS handles patching, backups, and upgrades and runs your directory infrastructure across multiple Availability Zones for high availability.

After setup, your end users and IT administrators can use their corporate credentials to log on to AWS applications, such as Amazon Workspaces, Amazon WorkDocs, and Amazon WorkMail, as well as manage AWS resources, such as Amazon EC2 instances or Amazon Simple Storage Service (Amazon S3) buckets, through AWS Identity and Access Management (IAM) role-based access to the AWS Management Console.

AD Connector

AD Connector enables you to connect your Active Directory to the AWS cloud without complex directory synchronization technologies or the cost and complexity of hosting a federation infrastructure. AD Connector is a proxy; it sends requests to domain controllers in your domain, but does not replicate directory data to AWS. Your existing security policies, such as password expiration, password history, and account lockouts, can be enforced consistently whether users or IT administrators are accessing resources in your on-premises infrastructure or the AWS cloud.

You can also use AD Connector to enable multi-factor authentication by integrating it with your existing RADIUS-based infrastructure. This provides an additional security when users access AWS applications.

For more information, see Connecting to Your Existing Directory with AD Connector.

Simple AD

Simple AD is a managed, Samba-based directory in the AWS cloud. It supports commonly used Active Directory features, such as user accounts, group memberships, domain joining (of Amazon EC2 instances running Windows Server), Kerberos-based SSO, and group policies. This makes it even easier to manage instances and deploy Windows-based applications in the AWS cloud. Many of your existing applications and tools that require Active Directory support can be used with Simple AD. By default, Simple AD also provides daily, automated snapshots to enable point-in-time recovery.

For more information, see Creating a Directory with Simple AD.

Join Amazon EC2 for Windows Instances to an AWS Domain

SSM lets you configure, manage, and deploy server configurations to Amazon EC2 instances running Windows Server-based applications and workloads. The domain join feature of SSM reduces the number of steps required to join Amazon EC2 instances running Windows Server to a Simple AD directory during launch. You can use SSM to create a JSON-based configuration document with configuration tasks and then associate the document to one or more Windows instances. The EC2Config service is installed on AWS Windows AMIs. When you launch a new Amazon EC2 instance running Windows, the EC2Config service calls SSM to obtain and apply configurations that are associated with your instance. The EC2Config service will also periodically check for configuration updates to apply. SSM can be used to automate the installation or removal of MSI packages, run PowerShell scripts, configure and export Windows Server event log data to Amazon CloudWatch logs. The SSM extension requires EC2Config version 3.0 or later. If you did not launch your instance from a current Windows AMI, then you can install the latest version of EC2Config by following the steps here.

Domain join leverages SSM and the AWS Management Console to create a JSON-based configuration document that SSM uses to seamlessly join Windows Server instances to AWS Directory Service domains. You also eliminate the need to log in to each Amazon EC2 instance running Windows before joining it to the AWS Directory Service domain or the risk of placing domain credentials in a PowerShell script.

Amazon EC2 for Windows Example

The following figure shows a common architecture for a Windows application: a VPC with four subnets (two private and two public) split across two Availability Zones. The application and database tiers are in the private subnets; the web front-end servers are in the public subnets.

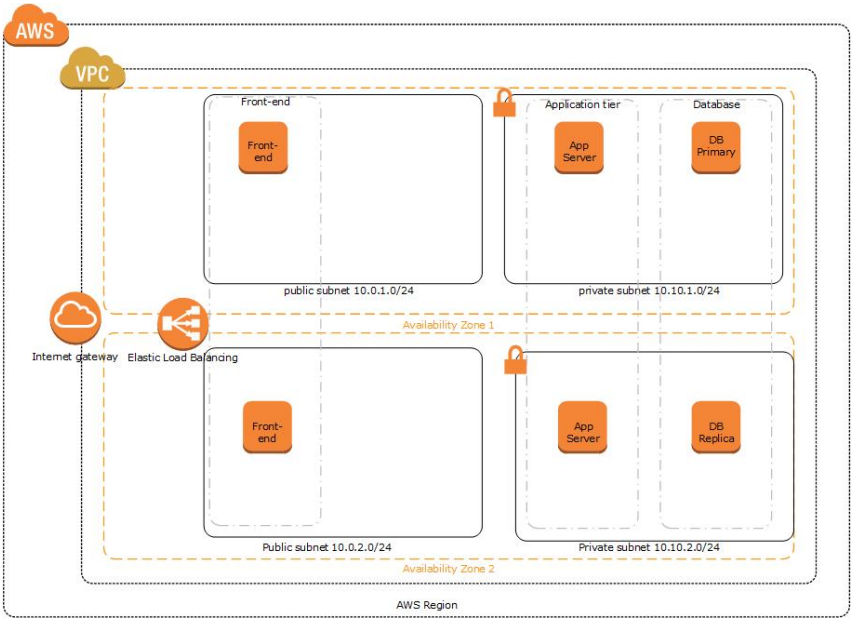


Figure 1: VPC with Four Subnets

SQL Server is running in the database tier. Kerberos authentication is required to enable the use of a business intelligence tool. You need to ensure the instances are configured to meet your company's security requirements, which require IPsec to secure communication between the application and database servers and BitLocker to encrypt the data volumes of the database servers. To address these requirements, you create a GPO with the appropriate settings and local group management to give permissions to the Active Directory group, DBA Admins.

With the GPO created and stored in the AWS Directory Service, here are some steps to ensure new and existing Windows instances launched in the private subnet will meet your requirements:

1. Use Windows Management Instrumentation (WMI) filters to configure GPOs to be applied to instances running Windows.
2. Seamlessly join these instances to the AWS Directory Service domain.
3. Verify the instances are configured correctly.

Use WMI Filters to Apply GPOs

The ability to automate the addition and removal of servers from your Amazon EC2 instances can make it difficult to apply configuration settings consistently.

Although there are many tools to dynamically detect the addition of new servers (for example, System Center Configuration Manager), this paper will focus on tools and methods included with Windows Server, primarily GPOs.

WMI is the infrastructure for management data and operations on Windows-based operating systems. You can write WMI scripts or applications to automate administrative tasks on remote computers, but WMI also supplies management data to other parts of the operating system and to products such as System Center Operations Manager and Windows Remote Management (WinRM).

WMI filters allow you to dynamically determine the scope of GPOs based on the attributes of the target computer. When a GPO linked to a WMI filter is applied on the target computer, the filter is evaluated on the target computer. If the WMI filter evaluates to false, the GPO is not applied. If the WMI filter evaluates to true, the GPO is applied. Using WMI filters, new instances that are launched dynamically will have the GPO applied based on information about the instance (for example, the subnet of the instance).

With WMI filters, you can ensure that GPOs linked to the AWS Directory Service domain will be applied based on the subnet in which you launched the instance. In the sample architecture shown in Figure 1, the private subnet for Availability Zone 1 is 10.10.1.0/24. Using the following WMI filter, GPOs will be applied only to instances launched in the private subnet for Availability Zone 1:

```
Select * FROM Win32_IP4RouteTable
 WHERE ((Mask='255.255.255.255' AND NextHop='0.0.0.0')
 AND (Destination Like '10.10.1.255'))
```

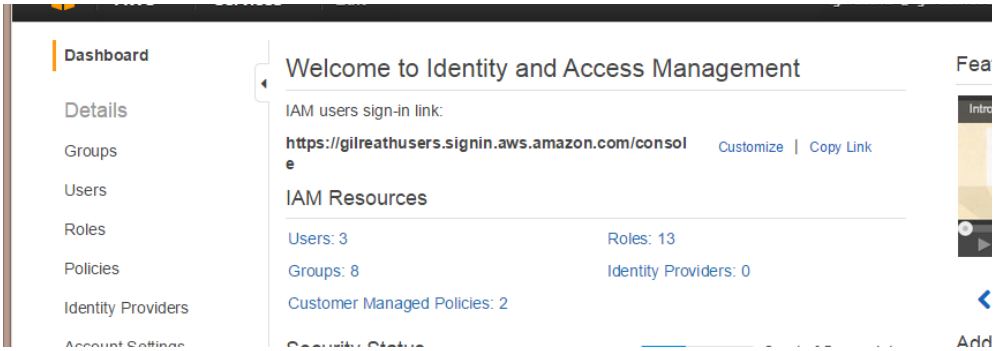
To determine the subnet of the instance, the WMI filter looks at the routing table, finds the route with a 32-bit subnet mask, a next hop of 0.0.0.0 (which indicates the IP address of the instance), and a destination that matches the desired subnet.

As new instances are launched manually or through automation like Auto Scaling, the WMI filter will be evaluated. If the instance is launched in the 10.10.1.0/24 subnet, then the GPO will be applied.

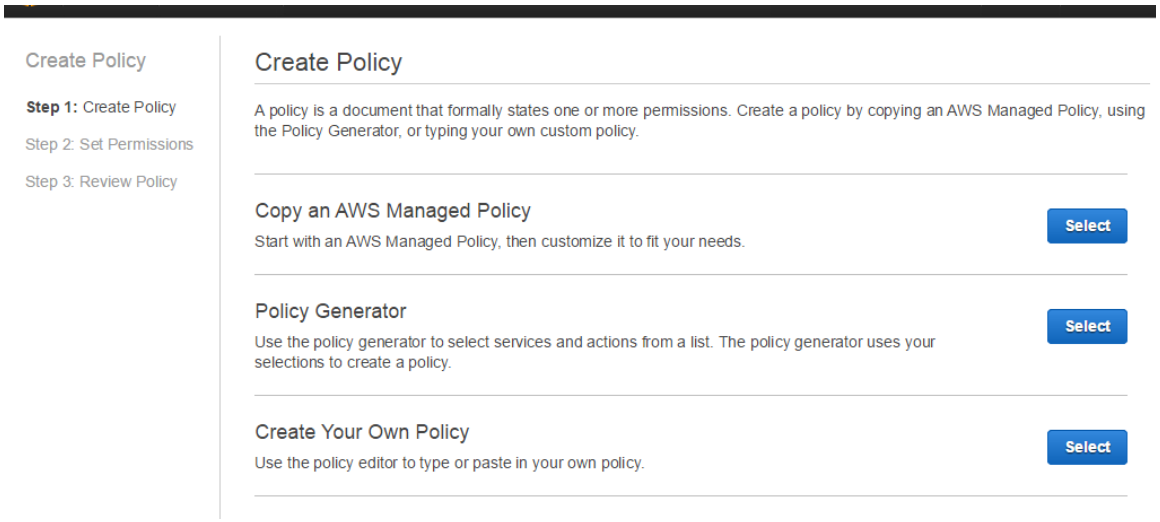
Join Instances to the AWS Directory Service Domain

After you have used Simple AD or AD Connector to define a domain in AWS Directory Service, you can configure Windows instances to join the domain at launch. The Windows instances must be able to make calls to the SSM API, so put them in a role with access to the Amazon EC2 SSM API. A role is essentially a set of permissions that grants access to actions and resources in AWS. These permissions are attached to the role, not to an IAM user or group. You define the permissions for a role in an IAM policy, which is a JSON document written in the IAM policy language. When you create the role, you create two separate policies for it: the trust policy, which specifies who is allowed to assume the role (the trusted entity, or principal) and the permissions policy, which defines which actions and resources the principal is allowed to use. Roles can be used by an AWS service, such as Amazon EC2. A role is assigned to an Amazon EC2 instance when you launch it; a role cannot be assigned to an instance that is already running. For more information, see Using IAM Roles to Delegate Permissions to Applications that Run on Amazon EC2.

You must first create the policy to attach to a role. In the AWS Management Console, in the navigation pane, click **Policies**.



Choose **Create Policy**, and then choose **Select** to create your own policy.



Type a name and description for your policy, paste the following text into the policy document, and then choose **Create Policy**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowaccesstoSSM",
            "Effect": "Allow",
            "Action": [
                "ssm:DescribeAssociation",
                "ssm:ListAssociations",
                "ssm:GetDocument",
                "ssm:UpdateAssociationStatus",
                "ds:CreateComputer",
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

The details in the policy document will help you understand the permissions granted by the policy:

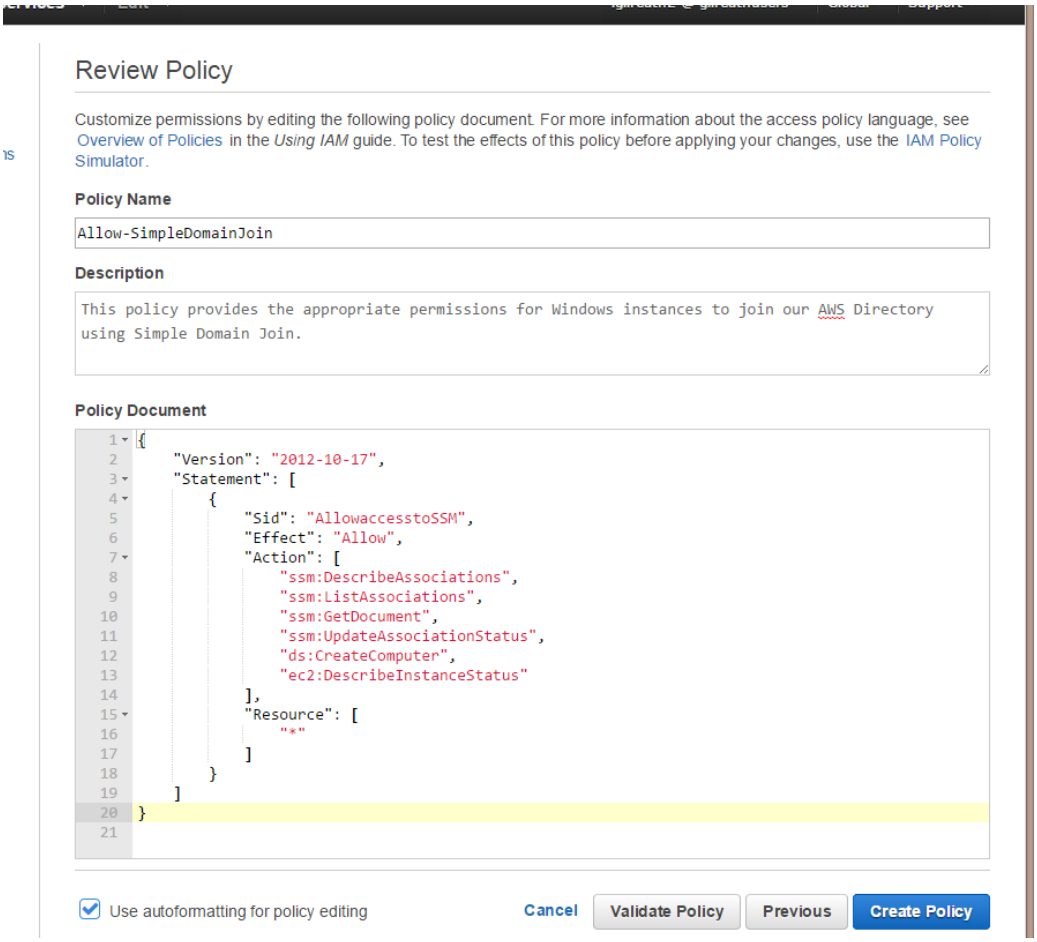
ssm:DescribeAssociation: API call that describes the associations for the specified configuration document or instance.

ssm:ListAssociations: Lists the associations for the specified configuration document or instance.

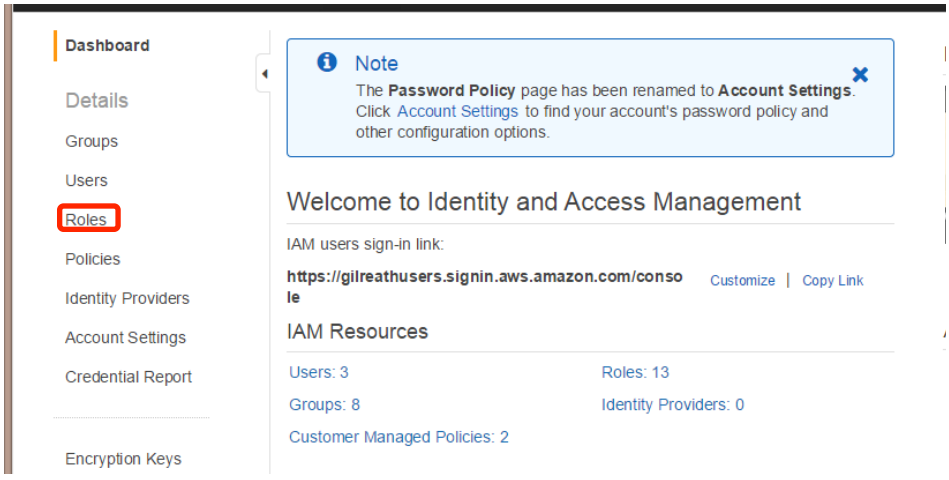
ssm:GetDocument: Gets the contents of the specified configuration document.

ssm:UpdateAssociationStatus: Updates the status of the configuration document associated with the specified instance.

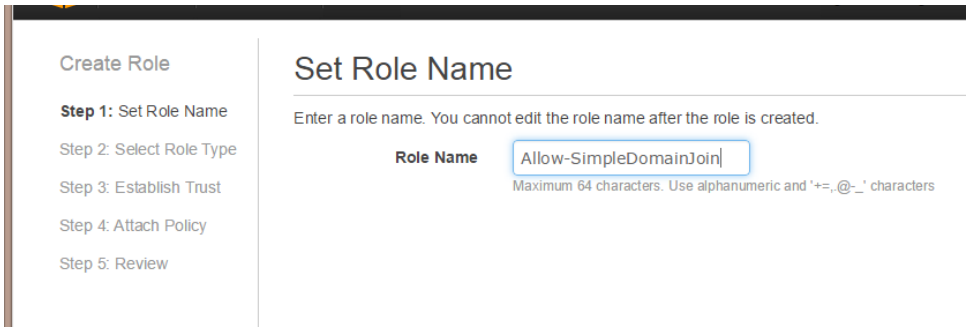
ds:CreateComputer: Allows creation of a computer object in the AWS Directory Service domain.



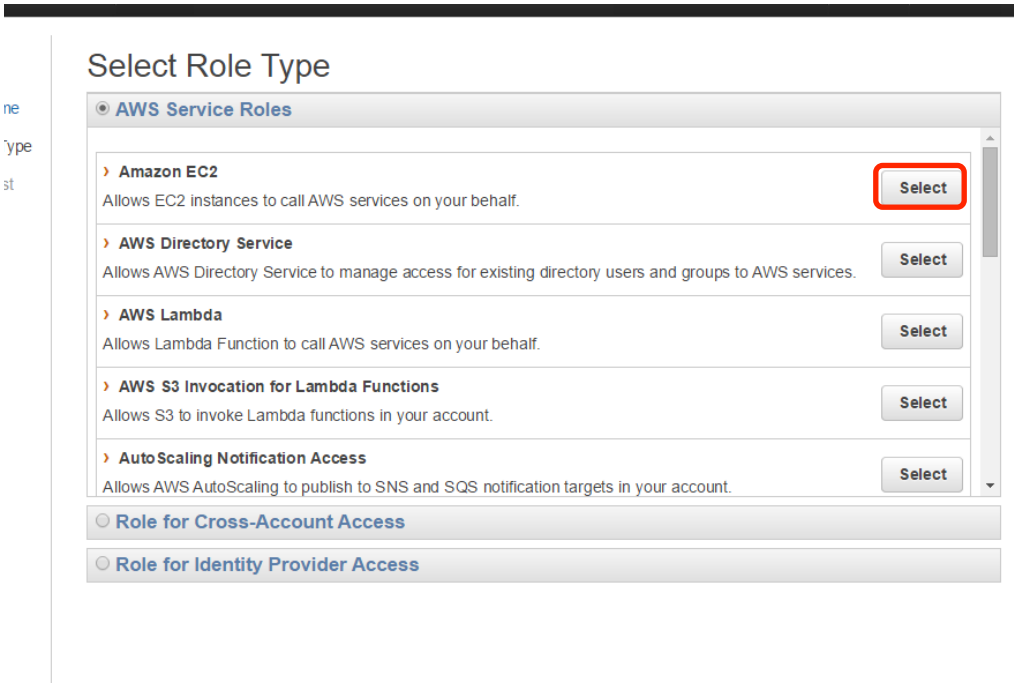
In the navigation pane, under **Dashboard**, choose **Roles**.



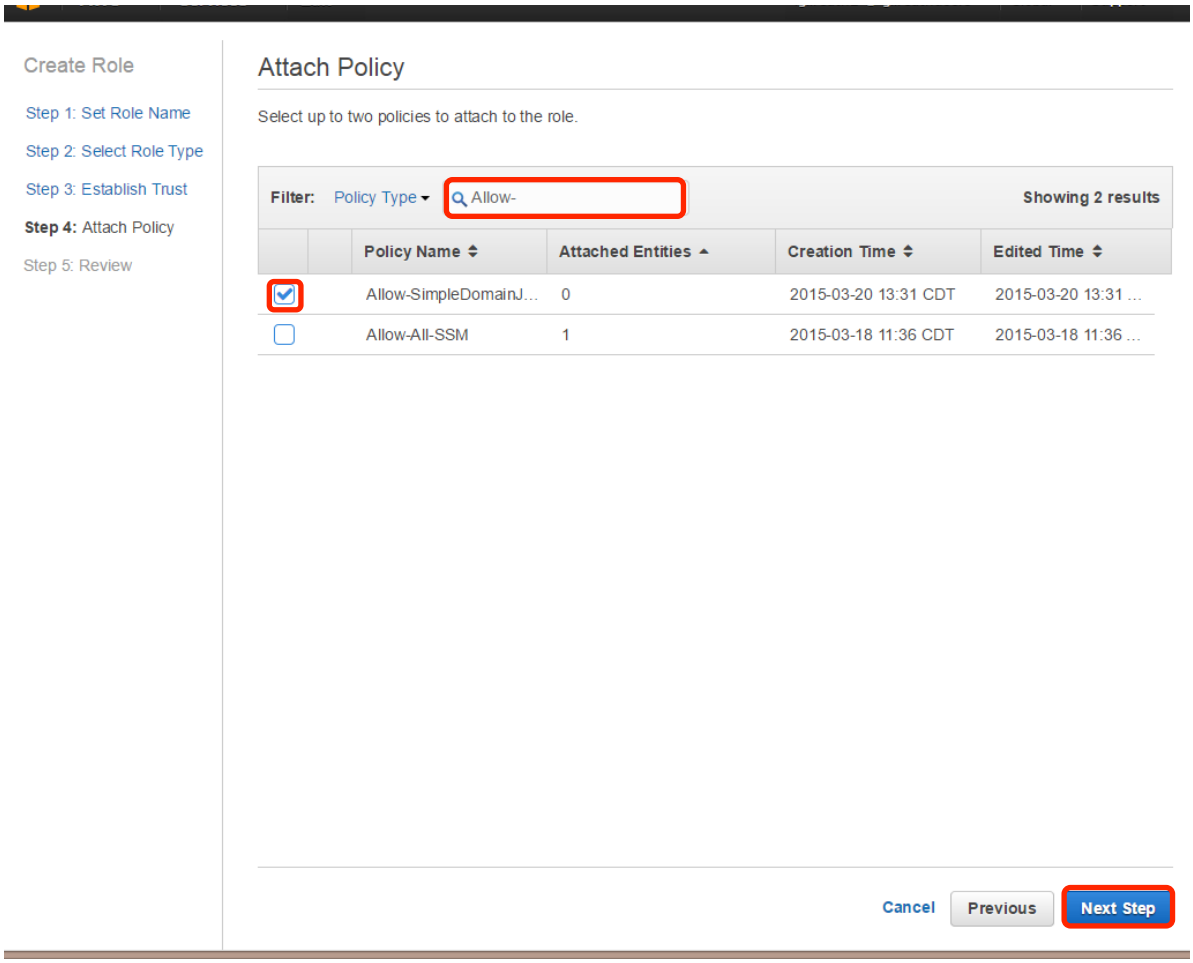
Choose **Create Role,** type a name for your role, and then choose **Next Step.**



On the **Select Role Type** page, select **Amazon EC2**.



On the **Attach Policy** page, search for the policy you created in the previous step, select the policy, and then choose **Next Step**.

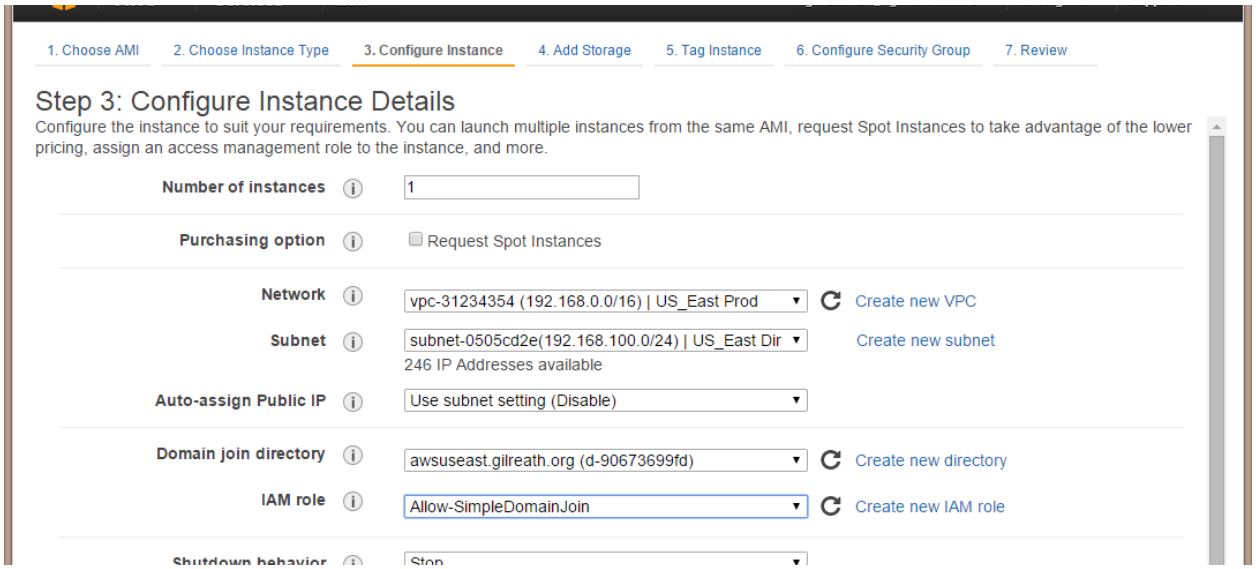


On the **Review** page, choose **Create Role**.

You can use the **Launch More Like This** wizard option in the Amazon EC2 console to seamlessly join a new instance to a domain that you specify.

To join a domain using the wizard

1. Open the Amazon EC2 console at https://console.aws.Amazon.com/.
2. On the Amazon EC2 console, click **Launch Instance**.
3. On the first page of the wizard, select a Windows AMI, and then click **Next**. On the next page, select an instance type, and then click **Next**.
4. From the **Network** drop-down list, select a VPC. Make sure you select a VPC located in your AWS Directory Service domain. From the **Subnet** drop-down list, select a subnet.

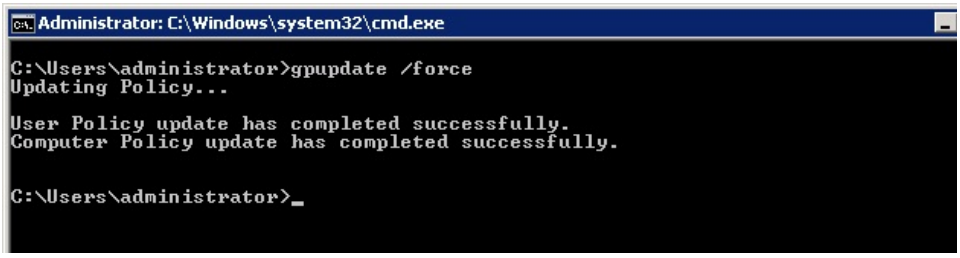


5. From the **Domain join directory** drop-down list, select your domain. From the **IAM role** drop-down list, select the IAM role to associate with the instance.
6. Complete the rest of the configuration steps as required, and then click **Next**.
7. When you reach Step 6 of the wizard, make sure you select or create a security group with a rule that allows RDP access from your IP address, or from a range of IP addresses, in your network. For more information about security group rules, see Authorizing Inbound Traffic for Your Windows Instances.
8. Click **Review and Launch** to launch your instance.
9. Check the status of the domain join. For more information, see Getting the Domain Join Status.

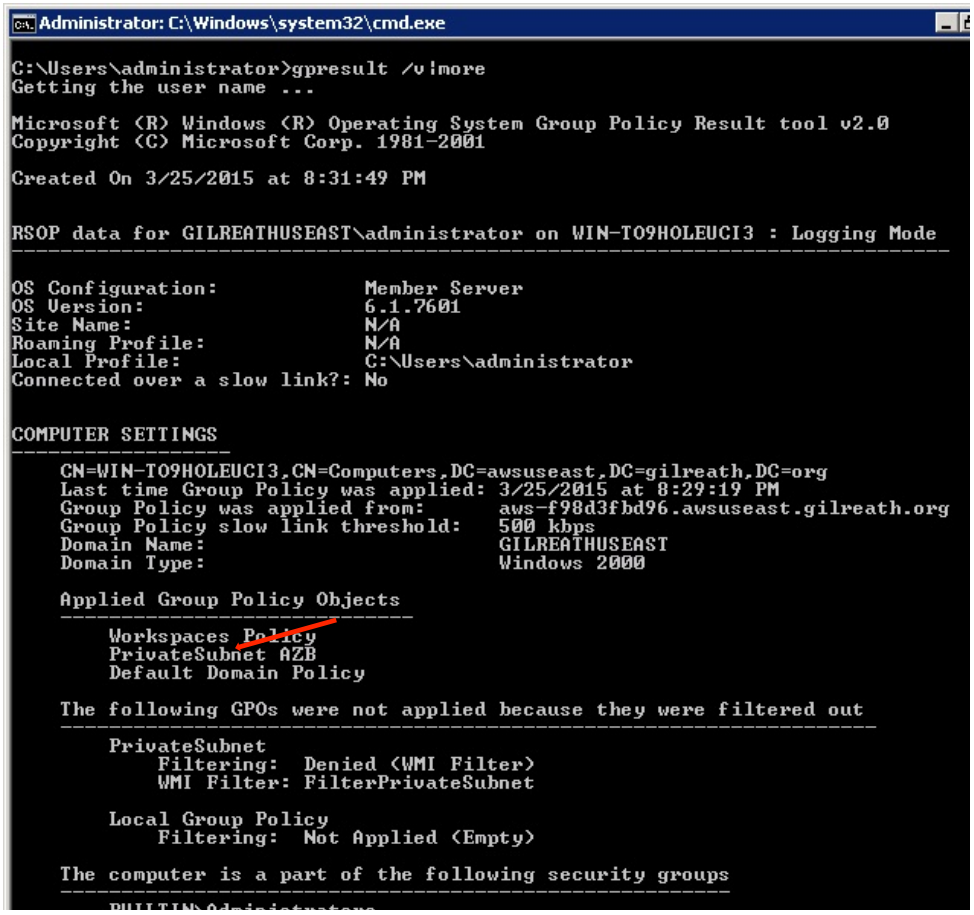
Verify the Instances Are Configured Correctly

After the instance has been launched and joined the domain successfully, you can connect to your instance using domain credentials that you've defined in AWS Directory Service. Follow these steps for connecting to your Windows instance using RDP.

From a command line on the Windows instance, run gpupdate /force to refresh the Group Policy settings on the instance.



Run gpresult /v to view the results.



In the Applied Group Policy Objects section, you will see your GPO. If you don't see your GPO, check the event logs or follow these troubleshooting recommendations.

Conclusion

AWS Directory Service and the domain join feature in SSM make managing your Windows fleet on Amazon EC2 easier. AWS Directory Service removes the burden of managing a separate domain and domain controllers, at a fraction of the cost of running multiple self-managed domain controllers on Amazon EC2. The domain join feature in SSM makes adding Windows instances to an AWS Directory Service domain a process that can be included in automation. In addition, being able to use native tools for Windows allows for management and configuration options that mirror your on-premises standards. Running Windows on Amazon EC2 has always provided flexibility with well-defined security. With AWS Directory Service and the domain join feature in SSM, AWS continues to be the best place to run Windows workloads.