WIN307 Step-by-step Solution

Below is a step-by-step solution for the workshop. We encourage you to try each challenge on your own before reading this solution. Spend about five minutes on each, then read the step-by-step solution. We suggest that you complete one challenge before moving on to the next.

Challenge 1 - Fix the Application

You are a consultant and you have just joined a new startup that is building an application using ASP.Net. You arrive to find things are in disarray. The latest release of the application is not working. You need to figure out why.

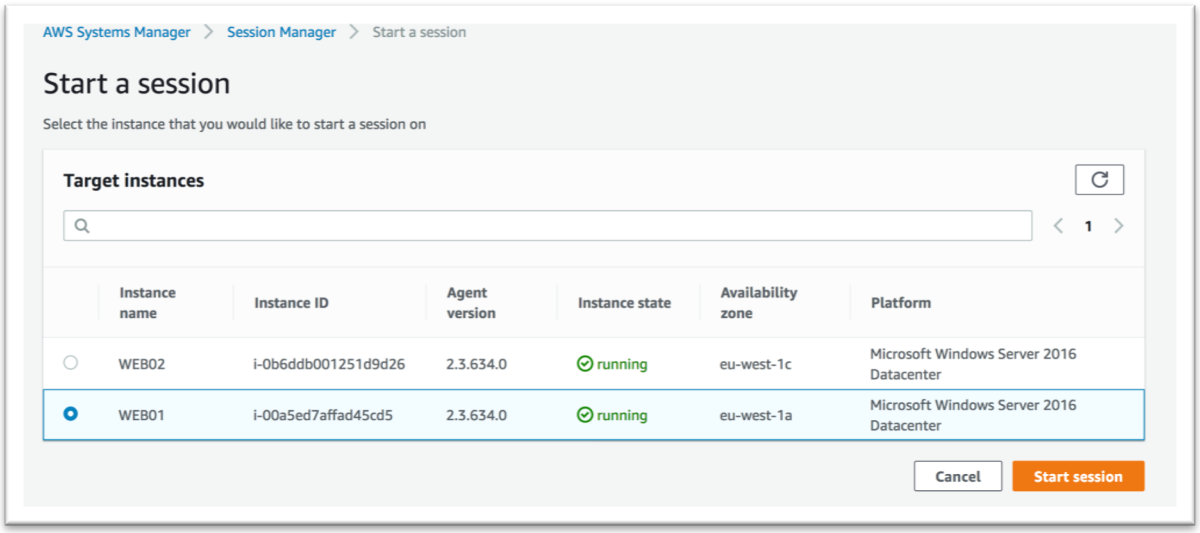
Let's begin by looking at the resources that make up the application.

1. Login to the AWS console and select the **Ireland** region at the top right corner.
2. Navigate to the **EC2** console
3. Choose **Load Balancers** in the left navigation and pick the load balancer who's name starts with "mod-."
4. Copy the **DNS name** and paste it into a new browser tab. *NOTE: This will fail. Chrome will show a 503 error. Firefox shows a blank page.* Keep this tab open as we will return to it later.
5. Go back to the EC2 console and choose **Instances** from the left navigation
6. Select "WEB01" from the list of instances and note that is has no public IP. *How are we going to diagnose this if we cannot RDP to the instance?*
7. Switch to the Tags tab and look at the Tags that have been applied to WEB01. We are going to use these tags to apply policy later in the workshop.
8. Right click on "WEB01"**,** choose **Instance Settings,** and **View/Change User Data**. *Note the PowerShell script that is being user to boot strap the instance. Something must be wrong with it?*

A. Use SSM Session Manager to remotely connect to the instance.

> *Session Manager is a fully managed AWS Systems Manager capability that lets you manage your Amazon EC2 instances through an interactive one-click browser-based shell or through the AWS CLI. Session Manager provides secure and auditable instance management without the need to open inbound ports, maintain bastion hosts, or manage SSH keys. Session Manager also makes it easy to comply with corporate policies that require controlled access to instances, strict security practices, and fully auditable logs with instance access details, while still providing end users with simple one-click cross-platform access to your Amazon EC2 instances.*

1. Navigate to **AWS Systems Manager** console
2. Choose **Session Manager** in the navigation pane and click the **Start Session** button
3. Under Target Instances select “WEB01” and click on **Start Session**.



B. Analyze the instance Userdata execution log to figure out why the application failed to install.

1. Once connected to the session, run a PowerShell command to read the content of the User Data execution log to identify the cause of execution failure.

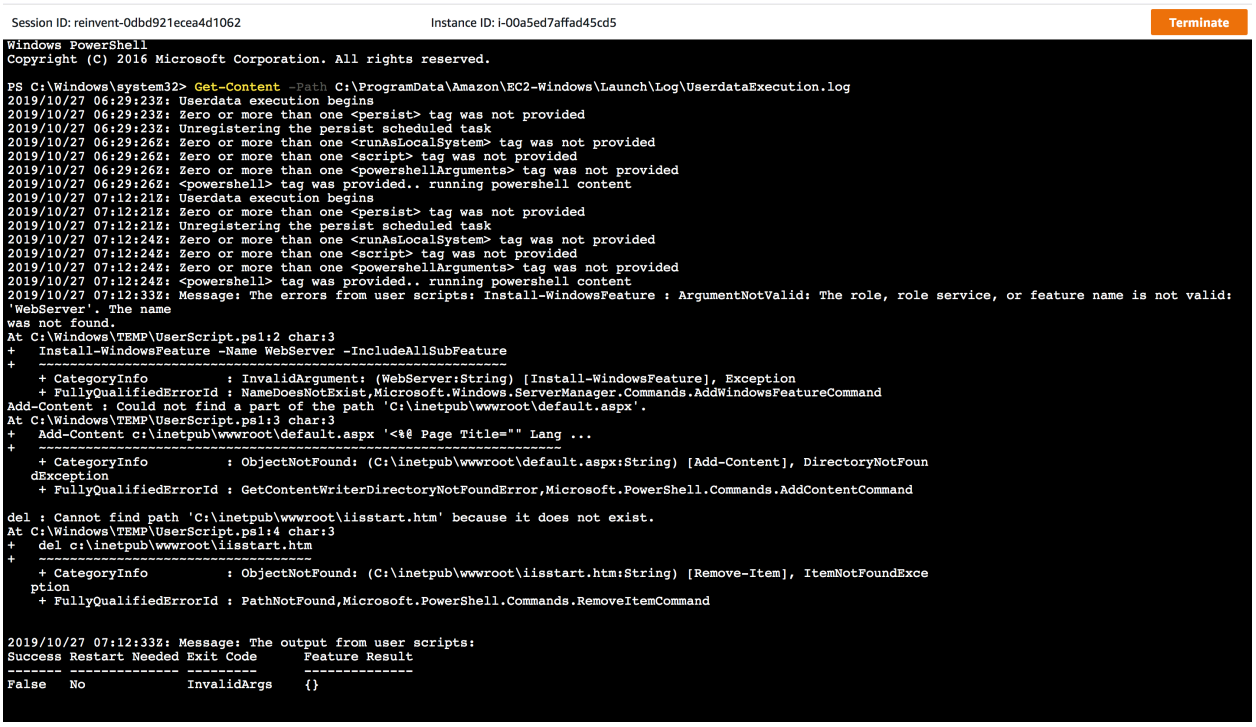
```
Get-Content -Path C:\ProgramData\Amazon\EC2-Windows\Launch\log\UserdataExecution.log
```

2. Make a note of the error message:

```
The errors from user scripts: Install-WindowsFeature : ArgumentNotValid: The role, role
service, or feature name is not valid:'WebServer'. The name was not found.
```

3. The above error states that the Windows Server Role name “WebServer” defined in the User Data script is invalid. It should have been “Web-Server”. Oops!

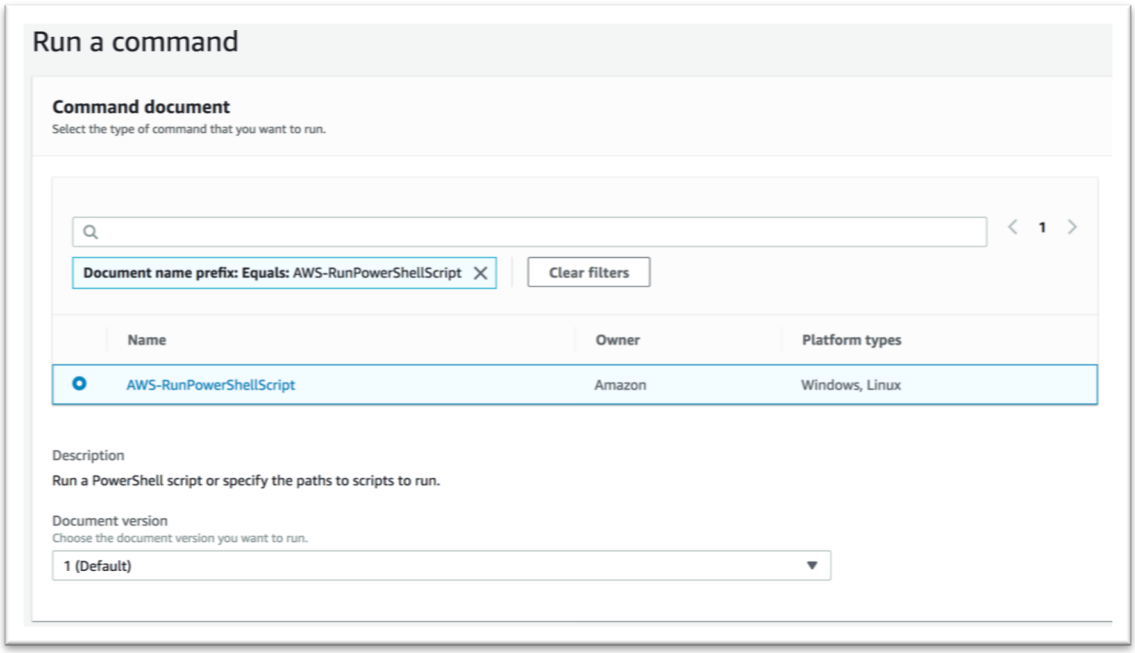
   While we could fix this in the instance, we would have to log into each instance to fix them individually. There is a better way. Let’s use **Run Command** to fix all the instances at once.



C. Use AWS Systems Manager Run Command to fix the IIS installation

AWS Systems Manager Run Command lets you remotely and securely manage the configuration of your managed instances. A managed instance is any Amazon EC2 instance or on-premises machine in your hybrid environment that has been configured for Systems Manager. Run Command enables you to automate common administrative tasks and perform ad hoc configuration changes at scale. You can use Run Command from the AWS console, the AWS Command Line Interface, AWS Tools for Windows PowerShell, or the AWS SDKs. Run Command is offered at no additional cost.

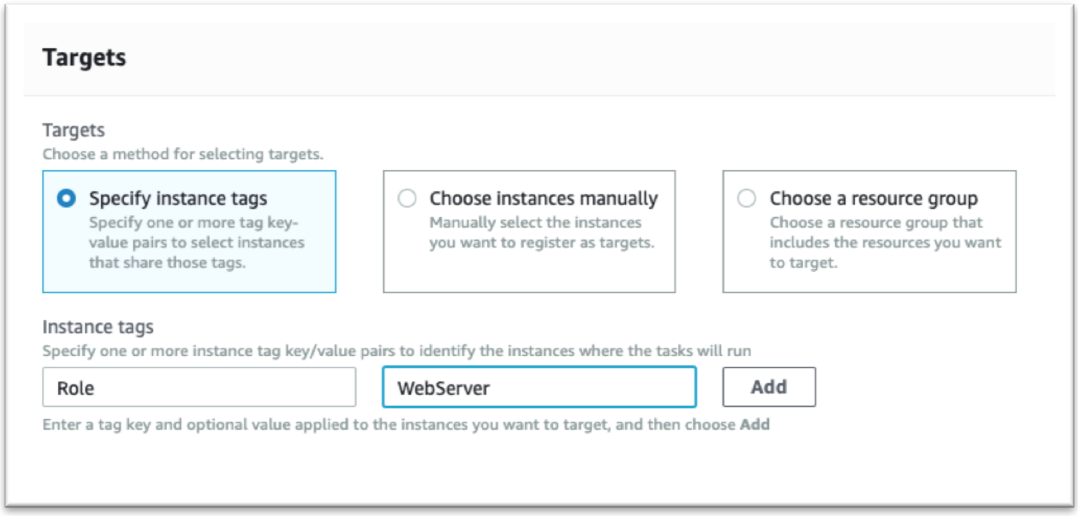
1. Return to the **Systems Manager** Console and choose **Run Command** from the left navigation.
2. Click the **Run a Command** button and search for **AWS-RunPowerShellScript**. *Note that the search box is case sensitive.*
3. Click the radio button to the left of **AWS-RunPowerShellScript**



4. Scroll down to **Commands** and enter the following. *Note that the second line is wrapping and may need to be fixed after you copy and paste it.*

```
Install-WindowsFeature -Name Web-Server -IncludeAllSubFeature
Add-Content c:\inetpub\wwwroot\default.aspx '<%@ Page Title="" Language="C#"
Trace="true"%>'
del c:\inetpub\wwwroot\iisstart.htm
```

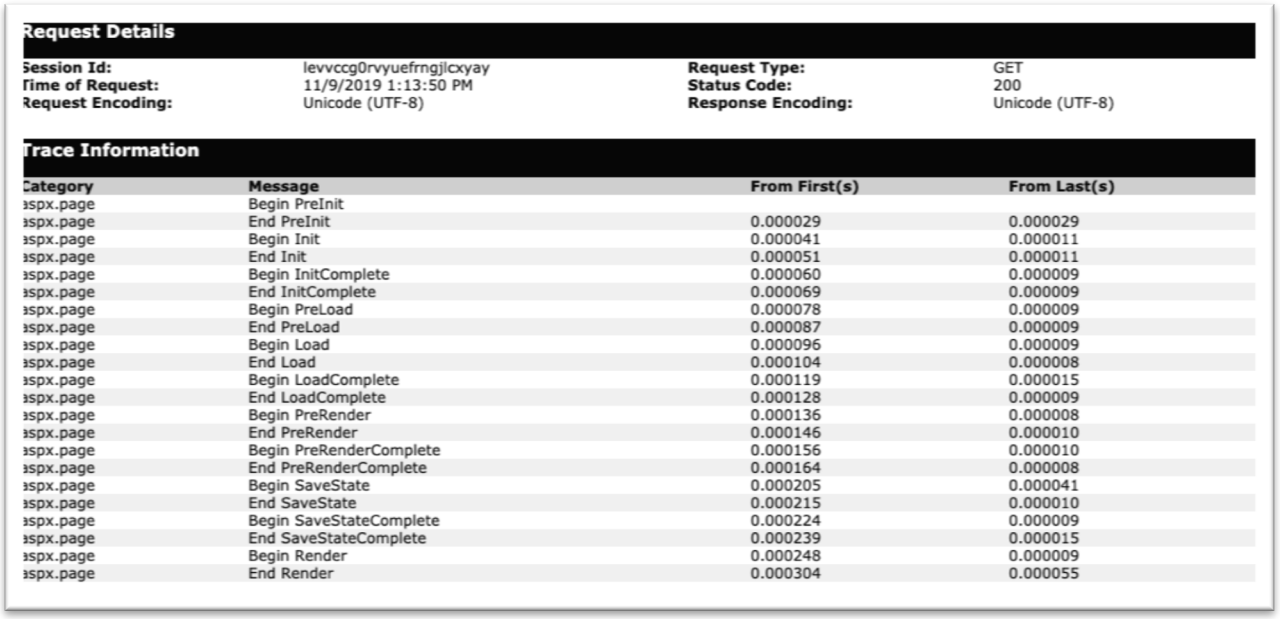
5. Scroll down to **Targets** and enter "Role" for the Tag Key and "WebServer" for TagValue. *(NOTE – Key/Value pair is cAsEsEnSiTivE)*



6. Press the **Add** button, then scroll down to the bottom and click the **Run** button.
7. Wait for the action to complete on both instances. *It will take a 3-5 minutes.*
8. When it completes, click on the instance Ids to see the output.

D. Confirm that the application is working

1. Return to the bowser tab that has the application open. *Remember that you pasted the load balancer URL in this tab earlier but got an error. Note that it may take 60 seconds or so after RunCommand completes for the instances to pass health checks and the page to load.*
2. Reload the page to ensure the application is working. *It's just an ASP trace page that looks like this:*



Request Details

Session Id:	levvccg0rvyuefrngjlcxyay	**Request Type:**	GET
Time of Request:	11/9/2019 1:13:50 PM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000029	0.000029
aspx.page	Begin Init	0.000041	0.000011
aspx.page	End Init	0.000051	0.000011
aspx.page	Begin InitComplete	0.000060	0.000009
aspx.page	End InitComplete	0.000069	0.000009
aspx.page	Begin PreLoad	0.000078	0.000009
aspx.page	End PreLoad	0.000087	0.000009
aspx.page	Begin Load	0.000096	0.000009
aspx.page	End Load	0.000104	0.000008
aspx.page	Begin LoadComplete	0.000119	0.000015
aspx.page	End LoadComplete	0.000128	0.000009
aspx.page	Begin PreRender	0.000136	0.000008
aspx.page	End PreRender	0.000146	0.000010
aspx.page	Begin PreRenderComplete	0.000156	0.000010
aspx.page	End PreRenderComplete	0.000164	0.000008
aspx.page	Begin SaveState	0.000205	0.000041
aspx.page	End SaveState	0.000215	0.000010
aspx.page	Begin SaveStateComplete	0.000224	0.000009
aspx.page	End SaveStateComplete	0.000239	0.000015
aspx.page	Begin Render	0.000248	0.000009
aspx.page	End Render	0.000304	0.000055

3. Great everything is working

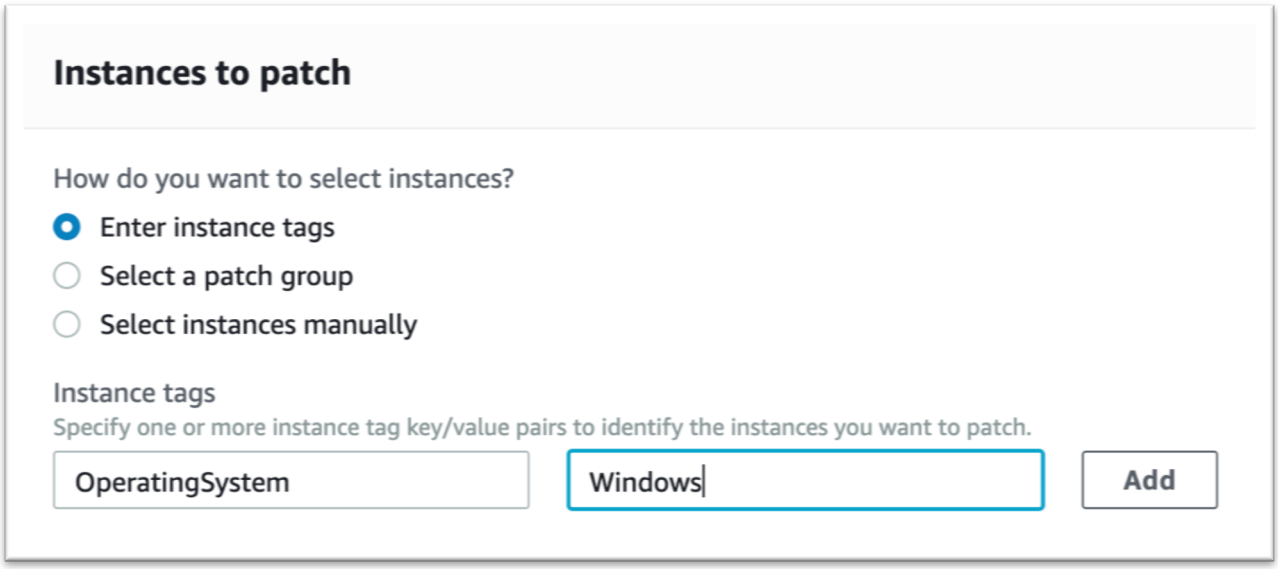
Challenge 2 - Automate Inventory and Patching

During your investigation you notice that the web server was missing a few critical patches. You want to get these machines under management. You would also like to collect inventory.

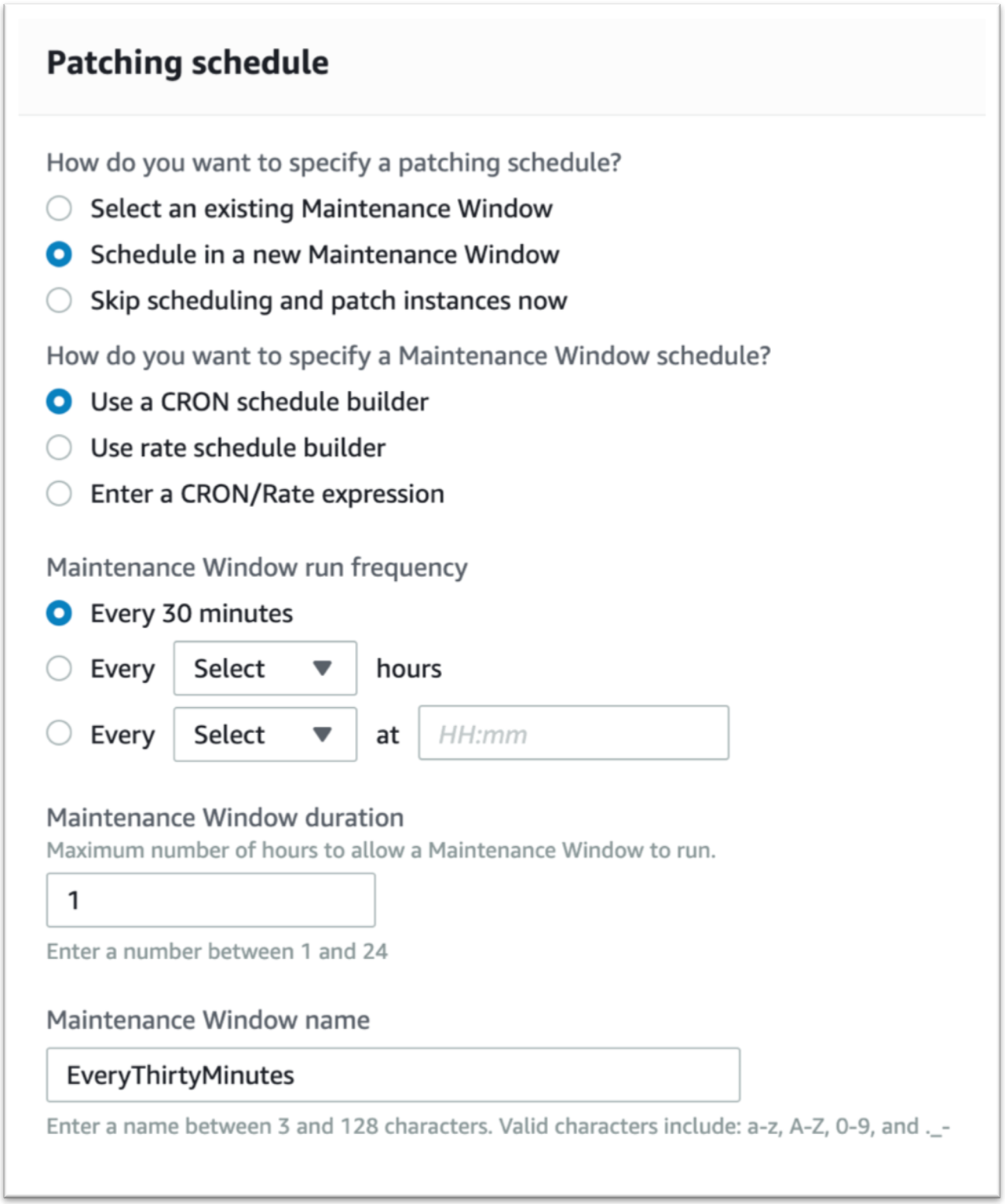
A. Use Systems Manager Patch Manger to configure patching

AWS Systems Manager Patch Manager automates the process of patching managed instances with both security related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications. (On Windows Server, application support is limited to updates for Microsoft applications.) You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Ubuntu Server, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), CentOS, Amazon Linux, and Amazon Linux 2. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.

1. Open **Systems Manager** and choose **Patch Manager** from the left navigation.
2. Click **Configure patching** button
3. Enter “OperatingSystem” as the tag key and “Windows” as the tag value



4. Click the **Add** button
5. Scroll to the **Patching schedule** and choose **Schedule in a new Maintenance Window**



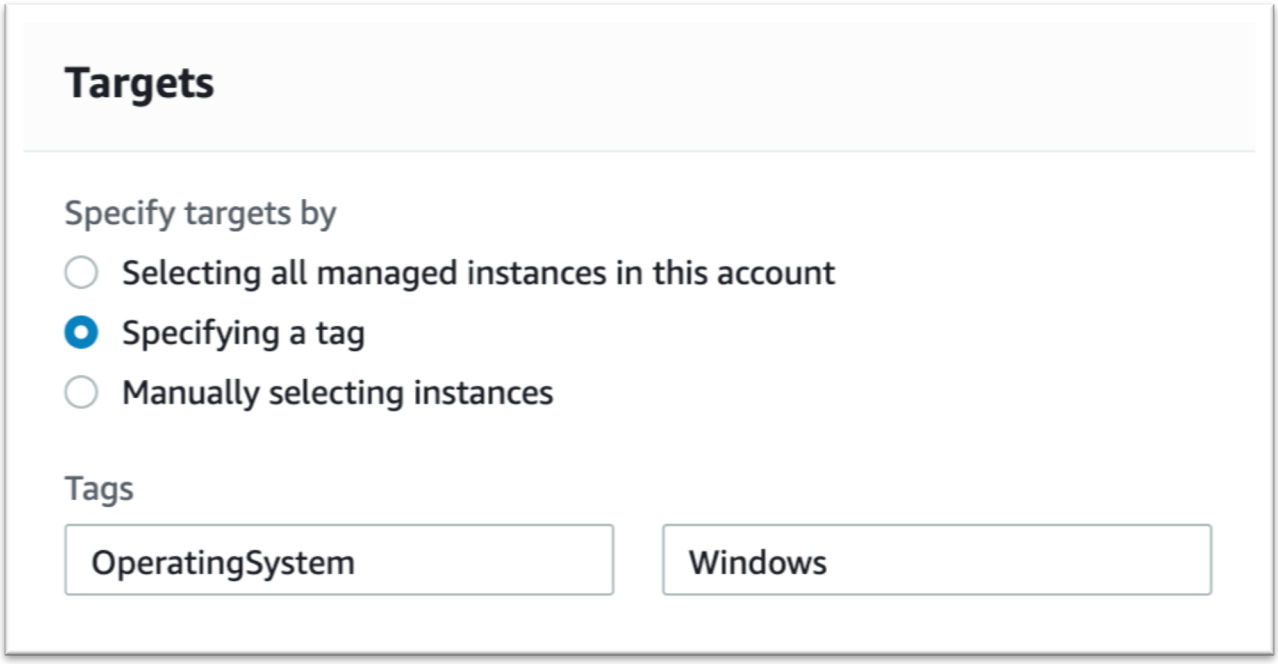
6. Leave the defaults and name the maintenance window "EveryThirtyMinutes"
7. Scroll to the bottom and click the **Configure Patching.**

NOTE: in a real-world application you would likely not patch every 30 minutes. For example, you could schedule patching for 2AM on the weekend. This schedule will causing patching to happen right away, which is good for a workshop.

B. Use Systems Manager Inventory to configure inventory

AWS Systems Manager Inventory provides visibility into your Amazon EC2 and on-premises computing environment. You can use Inventory to collect metadata from your managed instances. You can store this metadata in a central Amazon Simple Storage Service (Amazon S3) bucket, and then use built-in tools to query the data and quickly determine which instances are running the software and configurations required by your software policy, and which instances need to be updated. You can configure Inventory on all of your managed instances by using a one-click procedure. You can also configure and view inventory data from multiple AWS Regions and accounts.

1. Choose **Inventory** from the left navigation. *NOTE: you will see a red error at the top of this page. This will not impact your ability to configure inventory.*
2. Click **Setup Inventory** button.
3. Scroll down to **Targets** and choose **Specifying a tag**
4. Enter “OperatingSystem” as the tag key and “Windows” as the tag value



5. Scroll to the bottom and click the **Setup Inventory**

C. Explore the Results

NOTE: It will take a few minutes to collect Inventory information from the instances. Patching can take up to 30 minutes (the window we defined). You might want to move onto the next step and come back to confirm patching has complete.

1. Choose **Managed Instances** from the left navigation.
2. Click the link in the **Instance ID** column of either instance
3. Choose the **Inventory** tab at the top of the page to view inventory
4. Use the **Inventory Type** dropdown to explore the information collected
5. Choose the **Patch** tab at the top of the page to view patching status
6. Explore the patches that have been applied to the instance
7. Choose the **Configuration Compliance** tab at the top of the page
8. From here you track the compliance with your patching and other policies
9. Optionally return to the **Inventory** page to see an aggregate view across many instances

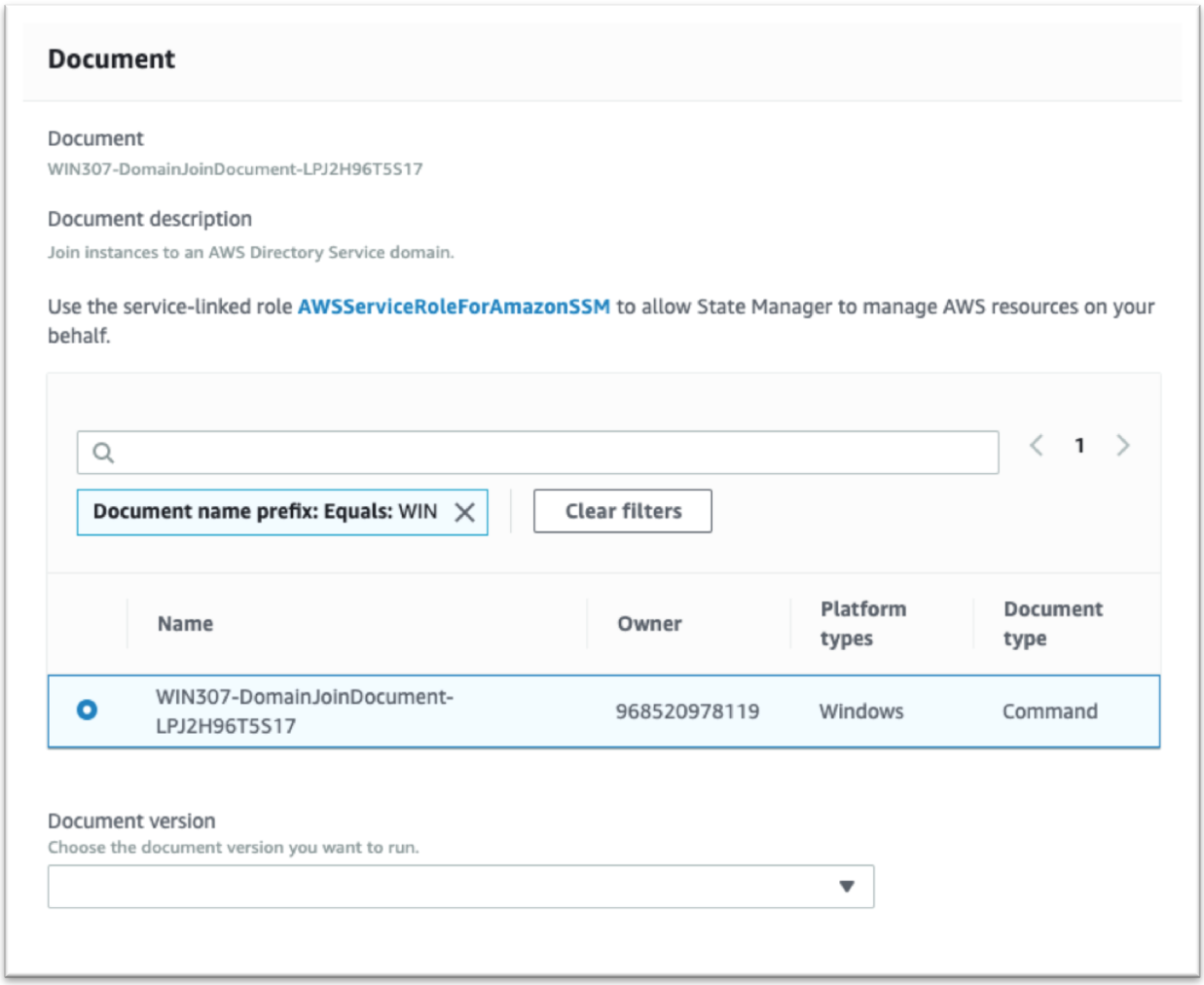
Challenge 3 - Automate Domain Join

The security team requires that the servers be joined to the corporate domain. In the past, the team has been doing this manually. You want to automate it ensure it happens quickly and consistently.

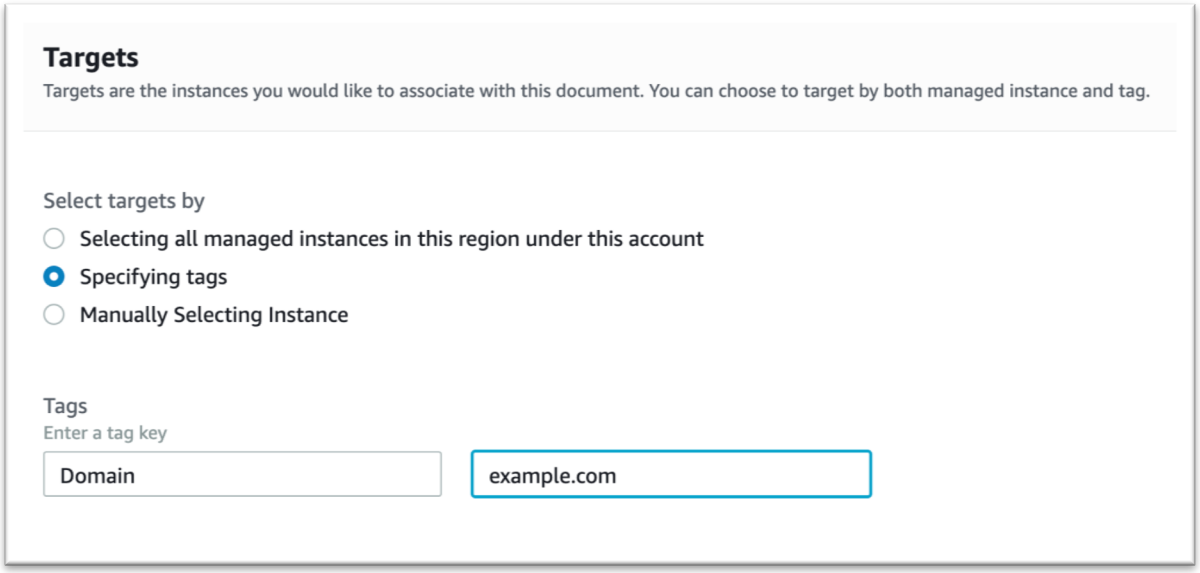
A. Use Systems Manager State Manager to configure domain join

AWS Systems Manager provides configuration management, which helps you maintain consistent configuration of your Amazon EC2 or on-premises instances. With Systems Manager, you can control configuration details such as server configurations, anti-virus definitions, firewall settings, and more. You can define configuration policies for your servers through the AWS Management Console or use existing scripts, PowerShell modules, or Ansible playbooks directly from GitHub or Amazon S3 buckets. Systems Manager automatically applies your configurations across your instances at a time and frequency that you define. You can query Systems Manager at any time to view the status of your instance configurations, giving you on-demand visibility into your compliance status.

1. Open **Systems Manager** and choose **State Manager** from the left navigation
2. Click the **Create Association** button
3. Name the association “JoinExampleDomain”
4. Scroll down to the Document section and search for document name prefix “mod-”
5. Choose the document that starts with **mod-############-DomainJoinDocument**



6. Scroll down to the Targets section and choose **Specifying tags**
7. Enter “Domain” as the tag key and “example.com” as the tag value



8. Scroll to the bottom and click the **Create Association**

B. Confirm that our servers have been joined to the example.com domain

NOTE: It will take a few minutes for the instance to join the domain.

1. Choose **State Manager** from the left navigation
2. Click the link in the **Association Id** for the **JoinExampleDomain** row
3. Choose the **Execution History** tab
4. Click link in the **Execution Id**
5. Click the **Output** link on either row
6. Click **Step 1 – Output** to review the logs
7. Note that the logs say **Domain Join Succeeded**

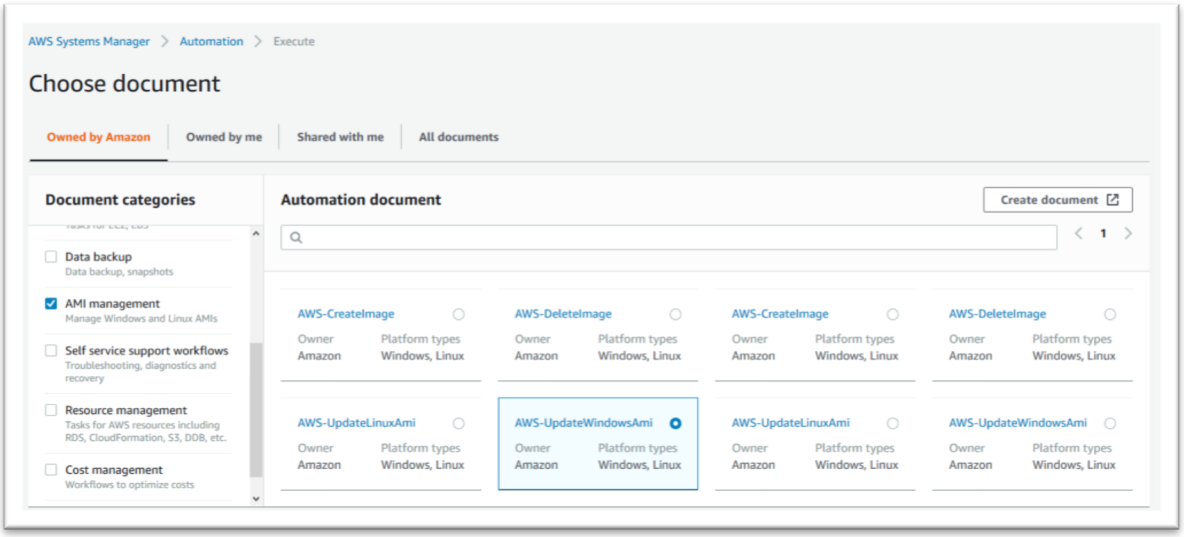
Challenge 4 - Create a Custom Image

During the last sprint review they mentioned that server launch times are longer than they would like. You suggest using SSM Automation to create a custom AMI. The custom AMI should be based on the latest public Windows Server 2019 AMI, and it should have IIS and our application pre-installed.

A. Use Systems Manager Automation to create a new custom AMI

The Automation feature simplifies common maintenance and deployment tasks, such as updating Amazon Machine Images (AMI). With the Automation feature in Systems Manager, you can apply patches, update drivers and agents, or bake applications in to your AMI using a streamlined, repeatable, and auditable process.

1. Open **Systems Manager** and choose **Automation** from the left navigation
2. Click on **Execute Automation**
3. Select **AMI Management** from list of the Document categories
4. Then select radio button for **AWS-UpdateWindowsAMI** and click **Next**



5. For **SourceAmiId** we are going to use an SSM Public Parameter to look up the latest Windows 2019 AMI in the current region. Copy and paste the following

```
{{ssm:/aws/service/ami-windows-latest/Windows_Server-2019-English-Full-Base}}
```

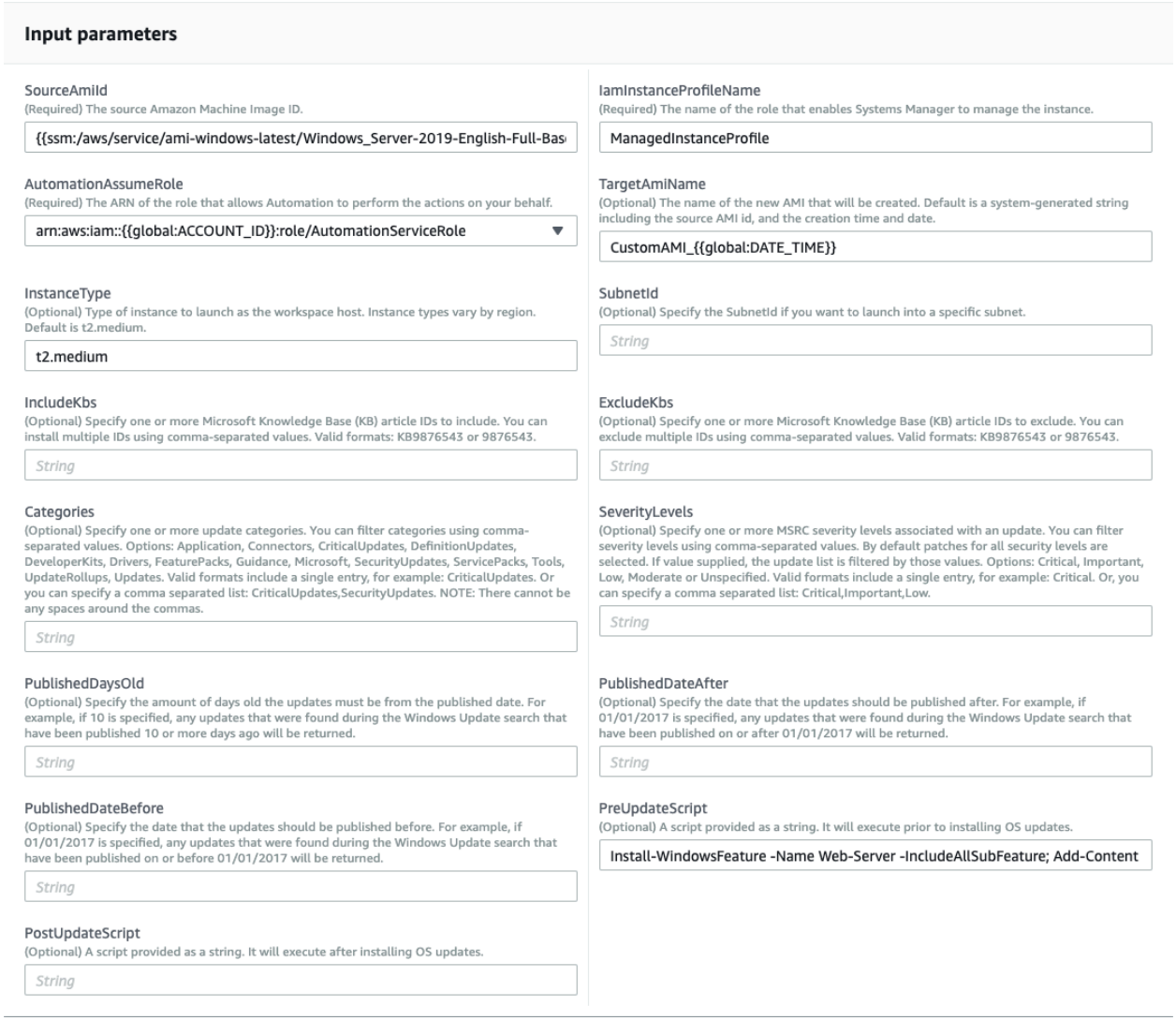
6. For **TargetAmiName** we will use the DATE_TIME parameter to generate a unique name. Copy and paste the following

```
CustomAMI_{{global:DATE_TIME}}
```

7. For **PreUpdateScript** copy and paste the following. *Note that we have fixed the typo from the original user data script. Also note the semicolons added to each line. When you paste this into the browser it will strip the new line characters and the semicolons are needed to separate the three lines.*

```
Install-WindowsFeature -Name Web-Server -IncludeAllSubFeature;
Add-Content c:\inetpub\wwwroot\default.aspx '<%@ Page Title="" Language="C#" Trace="true"%>';
del c:\inetpub\wwwroot\iisstart.htm
```

8. You can leave the default values for **AutomationAssumeRole** and **IamInstanceProfileName**. We have created these IAM roles for you



9. Click on **Execute**

B. Monitor the Automation as it Runs

NOTE: It will take about 30 minutes for the automation to complete.

1. Watch it as it runs each of the 14 steps.
2. Some steps will execute AWS APIs such as **RunInstances**, **CreateImage** and **ChangeInstanceState**. You can watch these happen on the EC2 console.
3. Other actions will use **RunCommand** to execute a Systems Manager Document. For these steps take a moment to drill in and see the logs from each step.
4. Once the automation completes, launch an instance into a public subnet to ensure that it is working as you expect.

*NOTE: You did **NOT** schedule this to run on a regular basis. You could create a Maintenance Window and have it execute this Automation every week. This would be similar to the patching windows we defined earlier.*