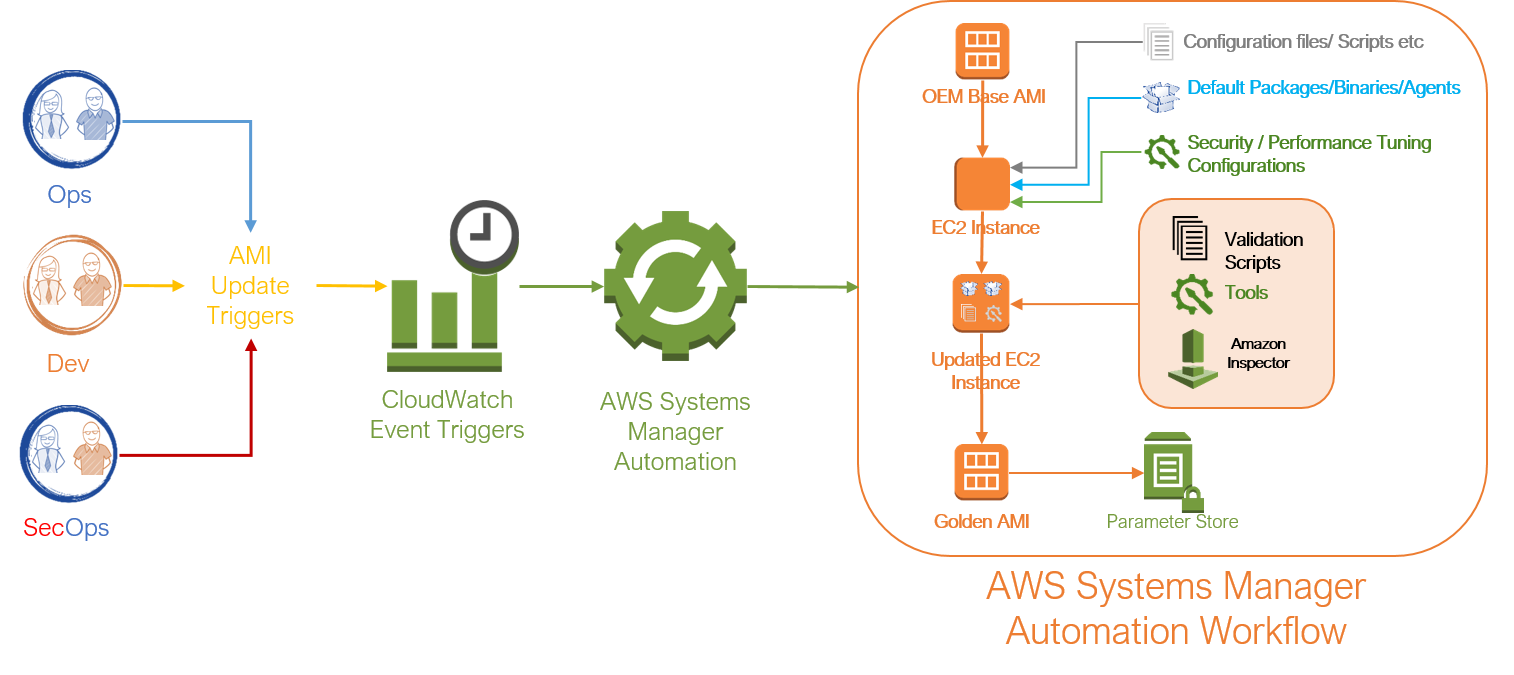
**AMI LifeCycle Management Using EC2 Systems Manager | Automation**

This walkthrough shows you how to create a custom Golden AMI using AWS SSM Automation document. The high level steps are shown below,



* Launch an instance from Source AMI mentioned in [SSM Parameter Store](https://github.com/miztiik/AWS-Demos/blob/master/How-To/setup-ami-lifecycle-management-using-ssm)
  + *Default:*/GoldenAMI/Linux/RedHat-7/latest
* Executes SSM Run Command that applies the vendor updates to the instance
* Stops the instance
* Creates a new AMI
* Encrypt the AMI
* Tag the AMI
* Update the parameter store using Lambda
* Delete unencrypted AMI
* Terminates the original instance

**Pre-Requisites**

1. SSM Parameter Store:
   * Parameter Name: /GoldenAMI/Linux/RedHat-7/latest
   * Parameter Value: *Any linux based AMI of your choice*
2. *Optional*: An S3 bucket with the file Post-Update-Scripts
   * Make sure this bucket is publicly accessible or having VPC Endpoint
3. RoleName: **lambda-ssm-role**:
   * Permissions: Managed Policies AWSLambdaExecute and AmazonSSMFullAccess
4. RoleName: **ManagedInstanceRole**: - An EC2 Role to allow SSM to start instances, create images etc,
   * Permissions: Managed Policies AmazonEC2RoleforSSM
5. RoleName: **AutomationServiceRole**: - An EC2 Role to Allow SSM to run documents and allow it to assume *ManagedInstanceRole* we just created.
   * Permissions: Managed Policies AmazonSSMAutomationRole
   * Inline Policy to attach the iam:PassRole
6. {
7. "Version": "2012-10-17",
8. "Statement": [
9. {
10. "Sid": "VisualEditor0",
11. "Effect": "Allow",
12. "Action": "iam:PassRole",
13. "Resource": "arn:aws:iam::YOUR-ACCOUNT-NUMBER:role/YOUR-ManagedInstanceRole-name"
14. }
15. ]
16. }
17. **Trust Relationships**: Make the below trust relationship to both the roles

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": [

"ec2.amazonaws.com",

"ssm.amazonaws.com"

]

},

"Action": "sts:AssumeRole"

}

]

}

If this role creation and attaching trust policies, that part alone can be done using this [cloudformation template](https://console.aws.amazon.com/cloudformation/home?region=us-east-1" \l "/stacks/new?stackName=Systems-Manager-AMI-Automation-Setup&templateURL=https://github.com/miztiik/AWS-Demos/blob/master/How-To/setup-ami-lifecycle-management-using-ssm/build-scripts/create-iam-roles-for-build.yaml). It doesn't create the lambda roles though. [W.I.P](https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-permissions.html)

**Create Parameter for Golden AMI**

Use the AWS CLI or GUI to create a parameter store entry, here is the CLI command.

* OEM Source AMI - /GoldenAMI/Linux/RedHat-7/source
* Updated Golden AMI - /GoldenAMI/Linux/RedHat-7/latest

# For example

aws ssm put-parameter --name /GoldenAMI/Linux/RedHat-7/source --value "ami-c86c3f23" --type String

**Create**[**Lambda**](https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-simpatch.html#automation-pet1)**to update SSM Parameter Store**

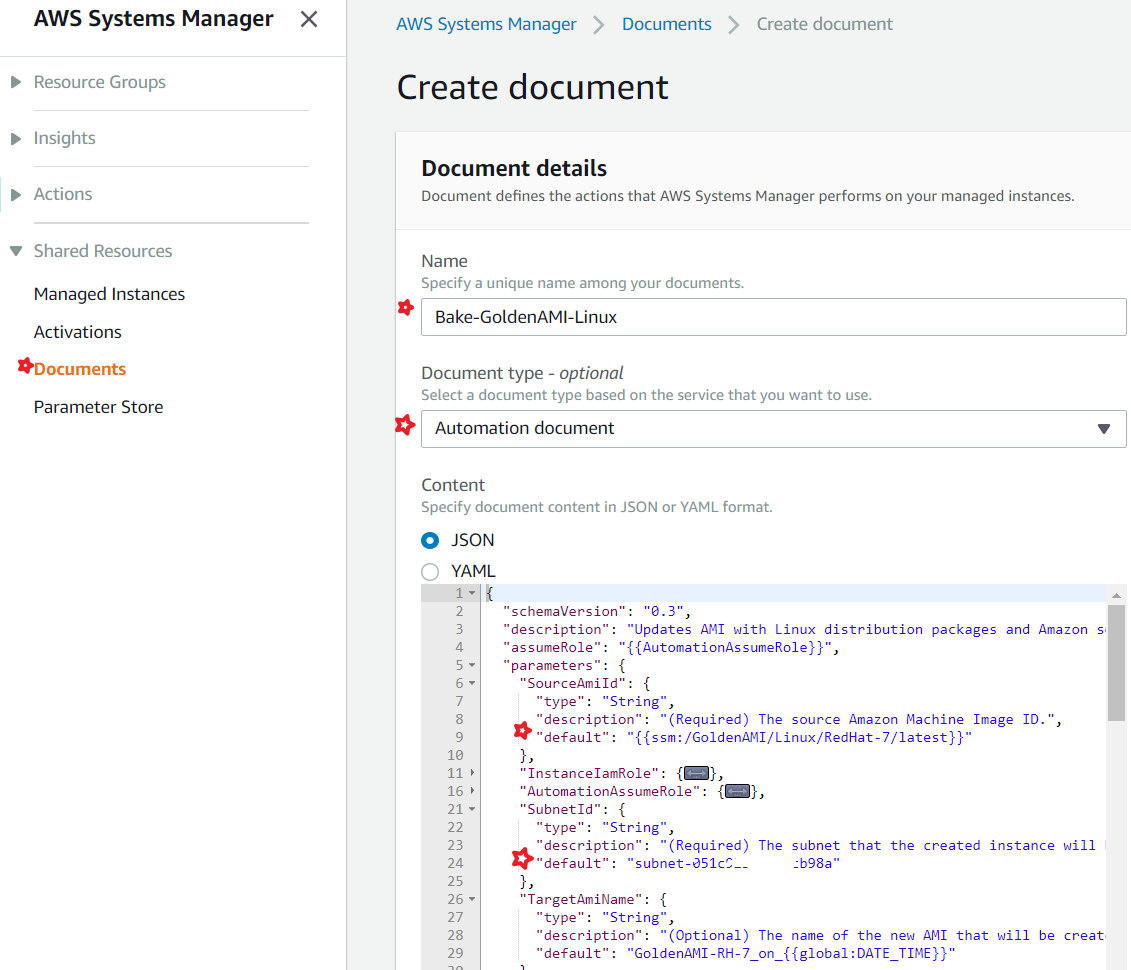
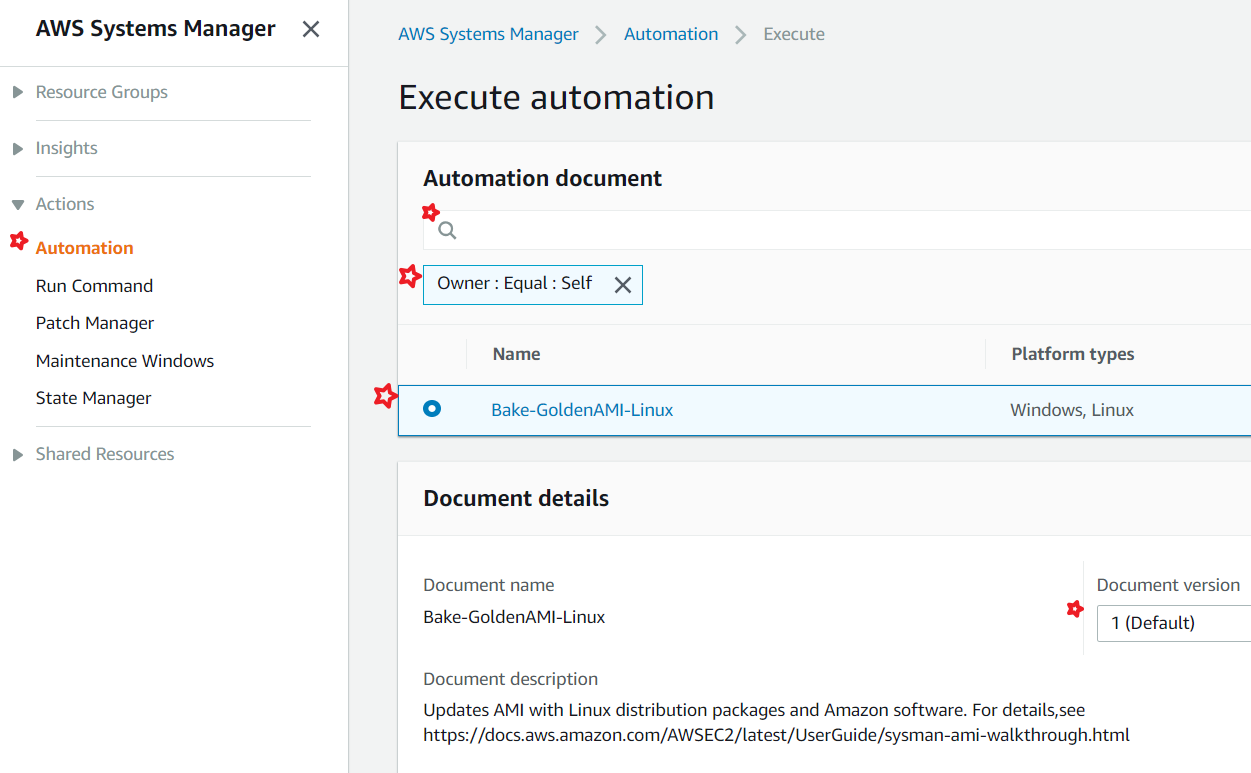
This function will helps us to automatically update the parameter store with the latest AMI when the *Automation Document* successfully create a new image.

**Customizations:** *Modify as required for your environment*

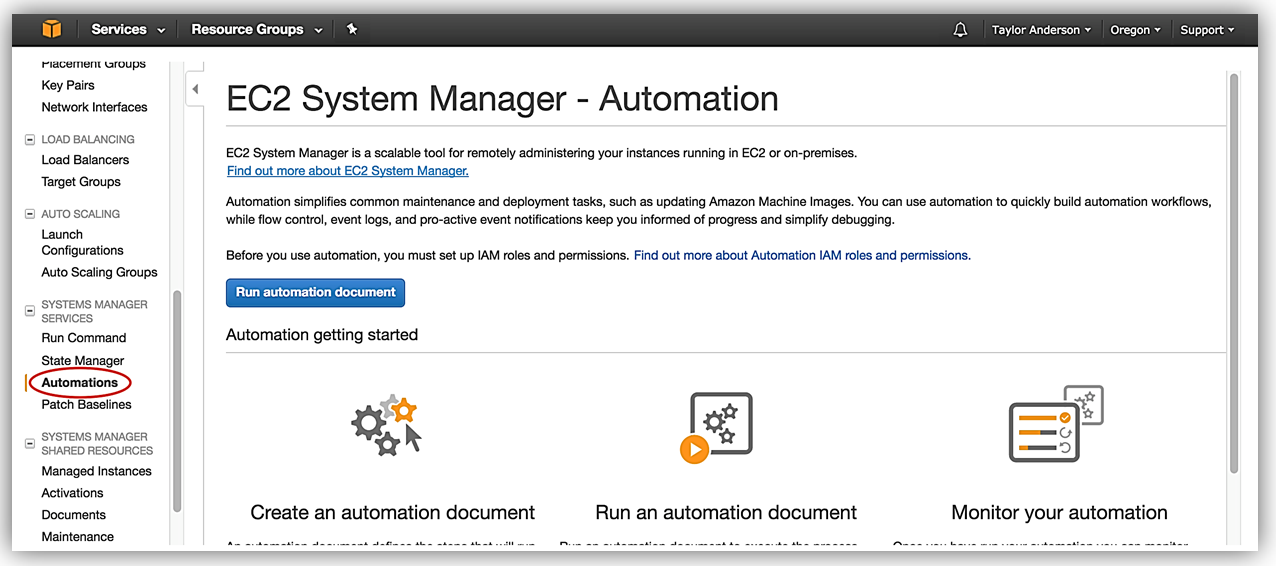
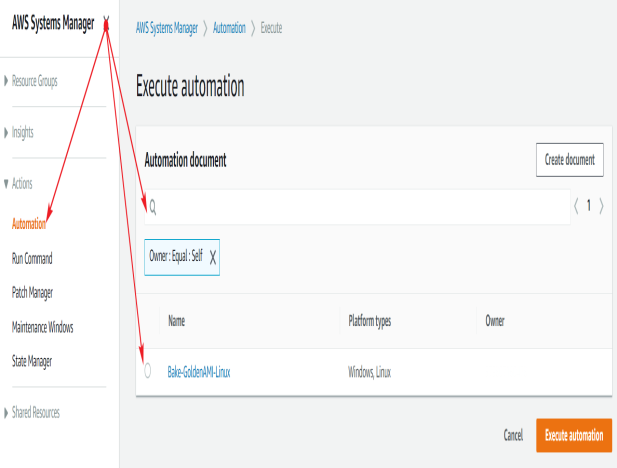
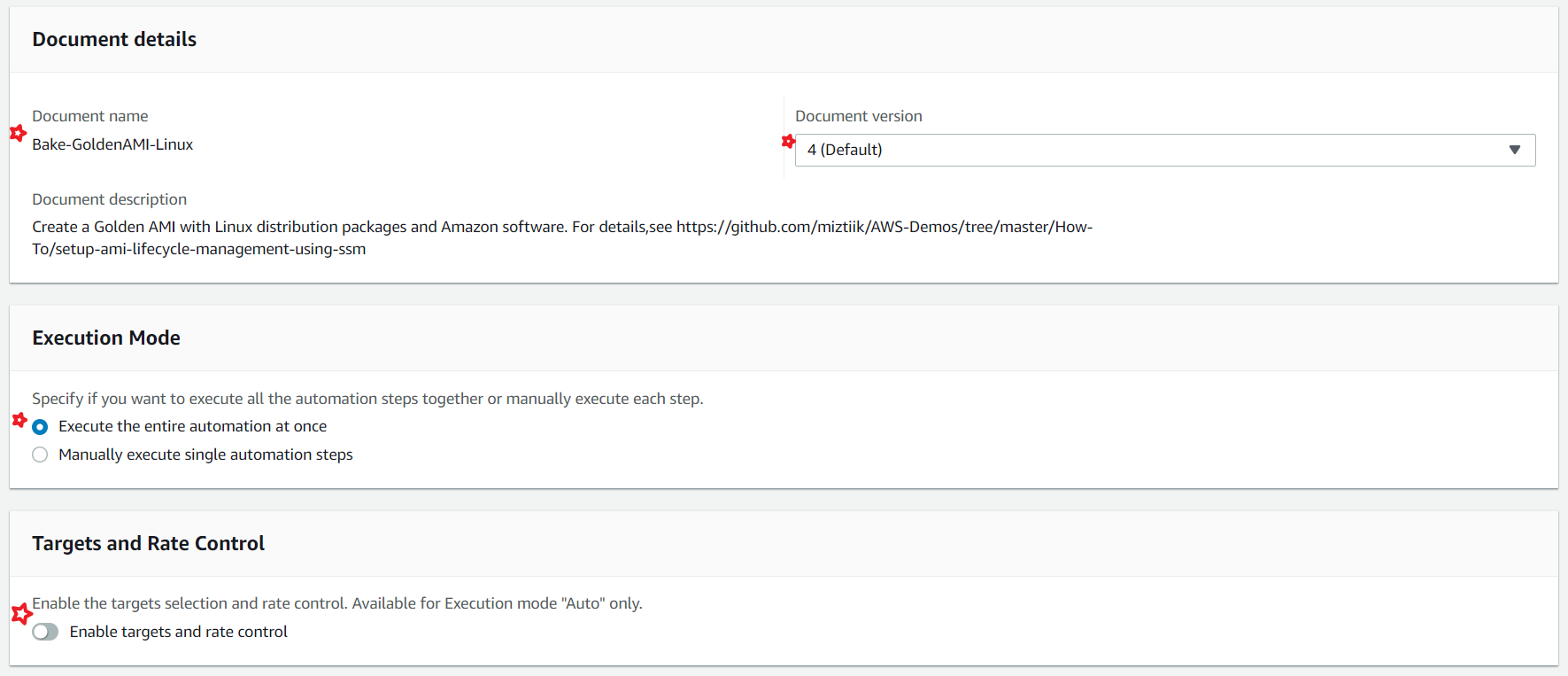
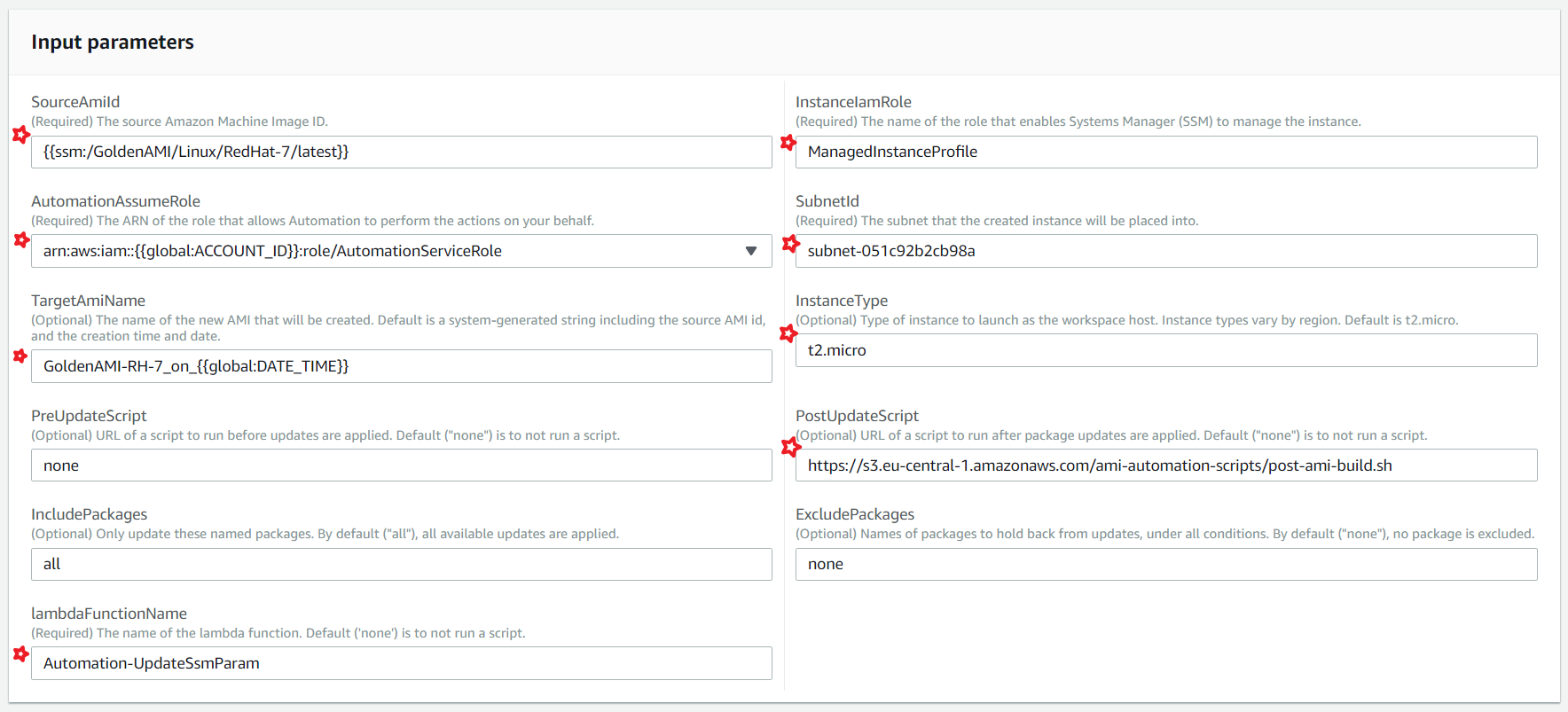
1. Lambda Function Name: Choose it as Automation-UpdateSsmParam. If you change here, update the *Automation Document* also with the same name.
   * Runtime should be Python 2.7
2. Choose the lambda-ssm-role you created earlier.
3. The code for lambda function is provided in this file Automation-UpdateSsmParam.py

**Create custom Automation Document**

***Note:*** By default SSM Automation runs in the default VPC, If you are going to run in a different VPC, you **must** provide the subnet id, which has internet access(directly or through NAT).

1. [Create Document](https://eu-central-1.console.aws.amazon.com/systems-manager/documents/create-document?region=eu-central-1), Give a **Name**, like "Bake-GoldenAMI-Linux.
2. Add the contents of the Bake-GoldenAMI-Linux.json file in the field.
3. After succesful creation of the document, you should be able view,modify versions,

**Creation of AMI**

1. In the EC2 console, choose **Systems Manager Services,** **Automations**.
2. Choose **Execution automation** and pick the one we just created. Easiest way is to search by "*Owned by Me"*.
3. Expand **Document Details** and choose **Latest version at runtime**.
4. Choose the latest document version.
5. For **Input Parameters**
   1. **SourceAmiId**, enter the ID of the Linux AMI to update.
   2. For **InstanceProfileName**, enter the name of the managed instance. i.e ManagedInstanceRole
   3. For **AutomationAssumeRole**, enter the [**ARN**](http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#identifiers-arns) of the service role you created for Automation.
6. Choose **Execute Automation**.
7. Monitor progress in the **Automation Steps** tab, and view step-level outputs.

**Pre-Update Script**

Any packages/libraries/scripts that needs to be copied to the server before the update process begins.

**Post-Update Script**

**Optional**: I have written a crude post-ami-build.sh script to install httpd and update the *index.html* to test everything is working fine. Store the file in S3 and ensure the user who run SSM has access to the s3 bucket.

For production usage, you might want to consider writing some more robust validation scripts

**Boot New Instance,**

if you boot the new instance, and login, you should get a welcome message like this, 

**ToDo**

* Optionally, CloudWatch Events to trigger Automation