[Auto Server Provisioning]

[Provision EC2 servers from ServiceNow]

Abstract

[The document provides a step-by-step explanation of setting up an AWS EC2 server provision mechanism from a ServiceNow catalog item

[Anand Atreya]

[anandatreya@yahoo.com]

Table of Contents

1. Conceptual Design

The intention of this project is to enable a simple AWS EC2 server provisioning mechanism from a ServiceNow Technical Catalog item.

The simplest use case for this project would be a developer who would want an EC2 server provisioned as per a standard specification.

A more advanced use case would be to provide developers a choice of multiple server types within AWS (Ubuntu OS, Microsoft, RedHat etc..) and /or server of different specifications (t2.micro. etc..)

A second use case of this project could be to use AWS CLI's rich command structure to provision environments supported by AWS Cli.

Another use case would be to configure multiple ServiceNow catalog items to allow Jenkins to connect to multiple cloud environments such as Microsoft Azure and Google Cloud to provision basic servers in those environments.

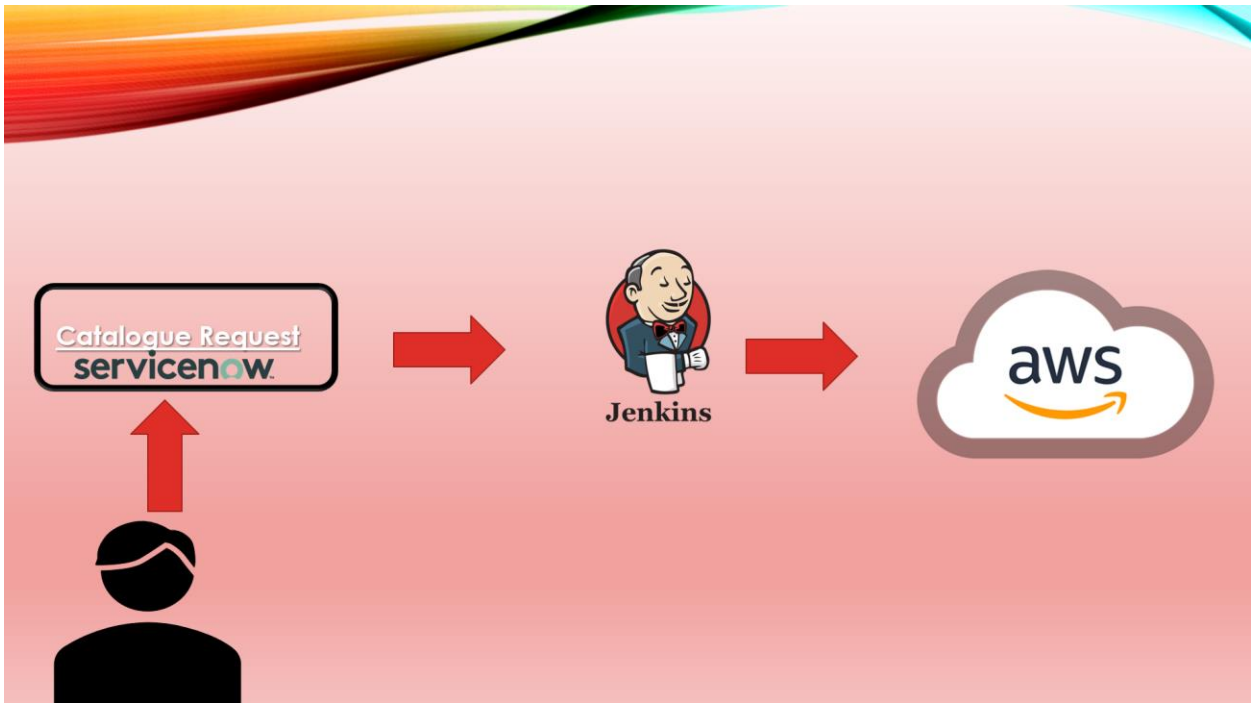
I have used ServiceNow as the ITSM product; this solution could be extended to allow other ITSM products such as BMC Remedy, HP Service Manager to integrate with Jenkins to allow this integration.

The conceptual design is very simple.

At the heart of the architecture is Jenkins which servers as the Continuous Integration / Continuous Delivery engine.

ServiceNow is the front end IT Service Management interface to the user (allowing users to request stuff through the Service Catalog).

Jenkins in turn executes jobs on the AWS infrastructure to provision an EC2 instance. Jenkins executes an AWS CLI command to provision the EC2 server.



Below are some of the Operational steps to achieve this integration -

a. Preparatory Tasks

b. Install Jenkins on an AWS EC2 server

c. Install and configure AWS Cli on the Jenkins server to be able to run commands on the AWS instance.

d. Configure Jenkins job and script to invoke the AWS Cli command to create the AWS EC2 instance.

e. Create an **Outbound Rest message** connection between ServiceNow and Jenkins

f. Create a workflow within ServiceNow to invoke the Outbound REST message. Configure the script (Outbound rest call) into the Workflow.

g. Configure a catalog item to invoke the Workflow that will execute the Outbound REST message.

2. Preparatory tasks:

I. Get a ServiceNow Developer instance.

II. Login to https://developer.servicenow.com/ and request your free ServiceNow developer instance.

III. Please choose Paris version.

IV. Get an EC2 instance from AWS to configure and manage Jenkins.

V. Install "awscli" on the Ec2 Jenkins host.

   a. Please refer to aws documentation on how to install awscli for your chosen OS

VI. Configure awscli using:

   a. Awscli config and ensure you input 4 parameters to the awscli configuration:

      i. the Secret Access key

      ii. Access Key ID

      iii. Default region Name

      iv. Default Output format

3. Setup Jenkins connectivity to EC2

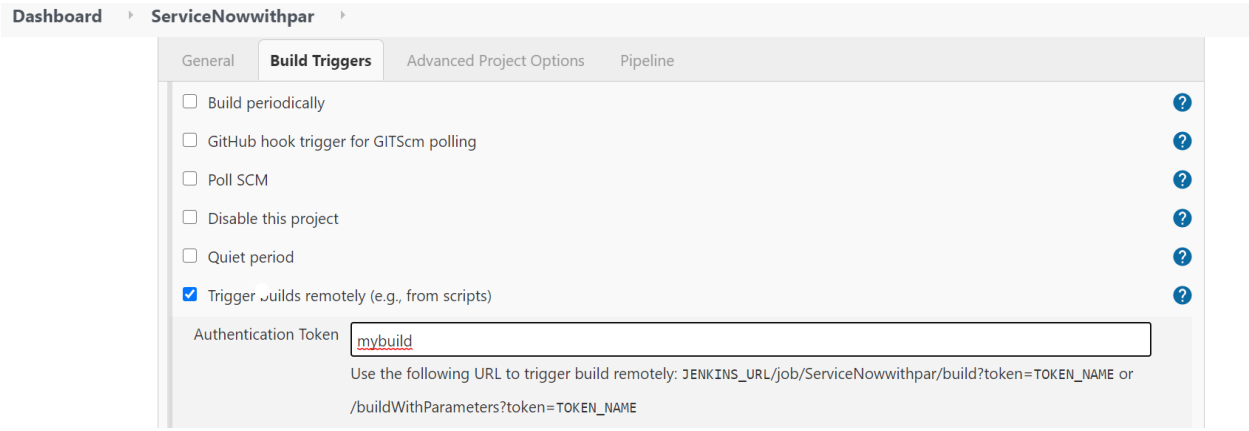
a. Setup Jenkins:

Installation of Jenkins on an EC2 instance. We assume you have already configured Jenkins in your environment. We are deploying Jenkins on an EC2 instance in the cloud (exposed on a Public IP).

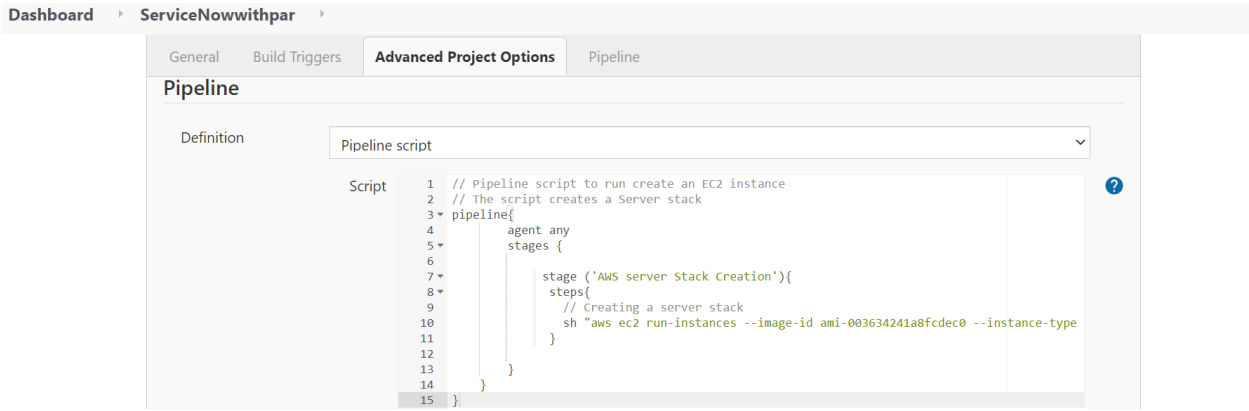
b. Create a new Jenkins job.

Two important configurations:

1. *Use an authentication token → mybuild (example, you can choose any text)*
2. *Check the box "Trigger builds remotely" as shown below.*



c. Configure the Jenkins script:



d. **Test the Jenkins AWS connectivity** by executing the Jenkins script to see if you are able to provision a new EC2 instance.

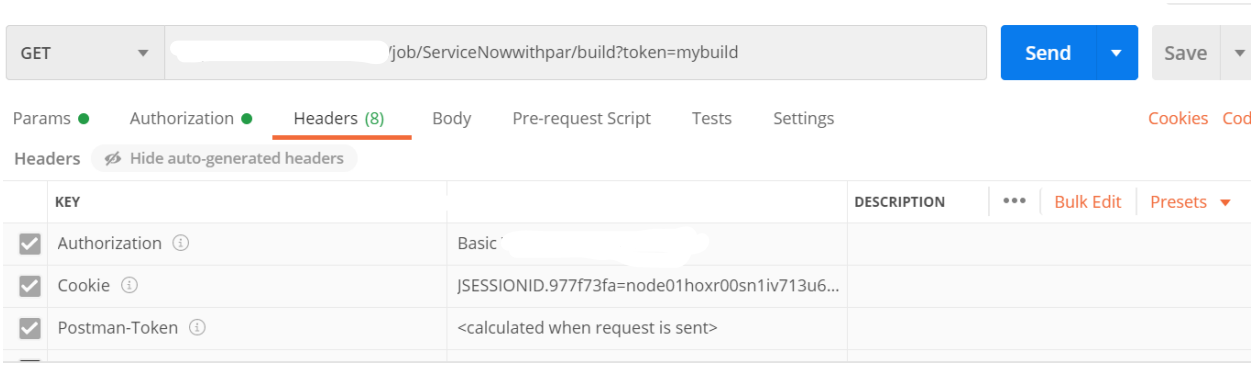
4. Configure Outbound REST Message between ServiceNow and Jenkins

a. ServiceNow side configuration

I have made use of Postman to configure a REST Script to receive the following:

1. The Get EndPoint
2. Authorization key and value
3. Token

On Postman:



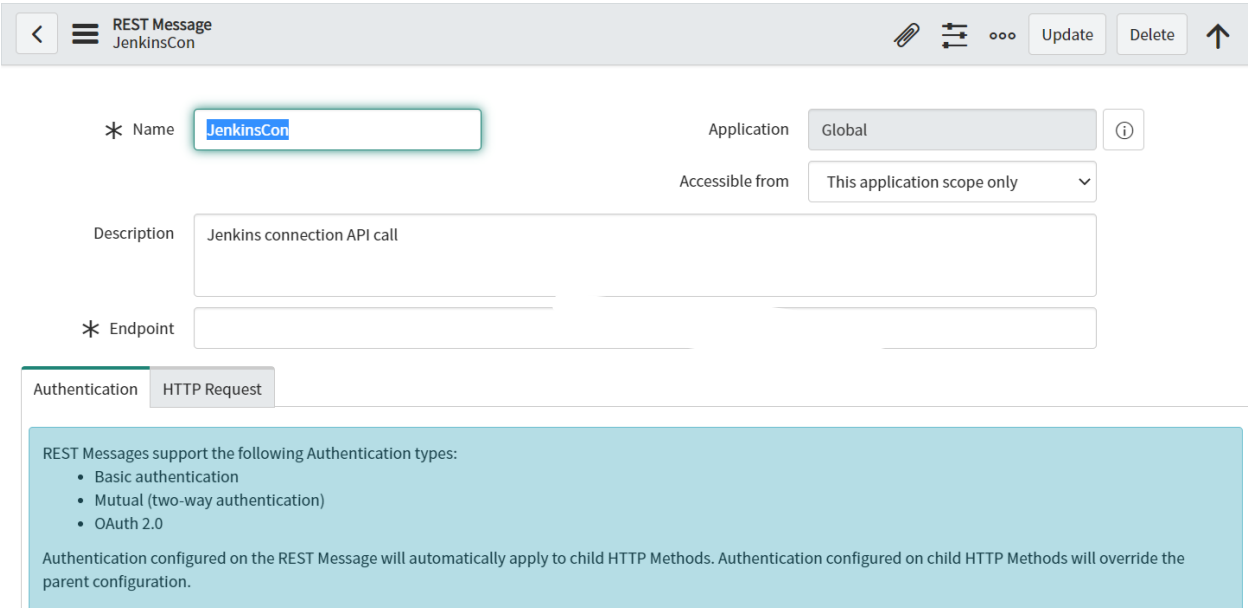
b. Create a new REST Message in ServiceNow

Go to Outbound REST interface.

Name of Rest Message: Jenkins Con

Reference:

Setup a Connection string using Postman.



Properties:

Authentication – No Authentication

HTTP Request tab – Authorization (value from postman)

Default Get tab –

Jenkins Job Name – ServiceNowWithPar

Endpoint:http://<EC2 server IP address>:8080/job/ServiceNowwithpar/build

HTTP Query Parameters –

Name = Token

Value = mybuild

5. Workflow within ServiceNow

a. Create a workflow within ServiceNow to invoke the Outbound REST message.

Configure the script (Outbound rest call) into the Workflow.

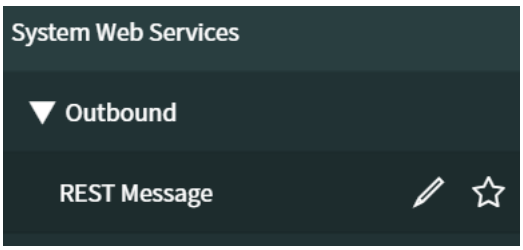
Before you create the workflow, please extract the script code for the REST message that will be invoked by ServiceNow workflow to connect to the jenkins server.

Since Orlando, Serviecenow does not allow REST API messages via the workflow. Since Orlando, ServiceNow strongly recommends using Flow Editor or specific serviceNow connector to Jenkins (available from Hi Support – this is available only if you are an enterprise customer of ServiceNow).

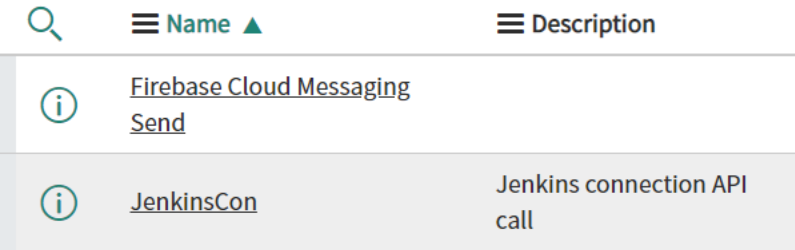
In our example, I have used the script generated automatically by the Outbound Rest Message within the workflow.

To do this,

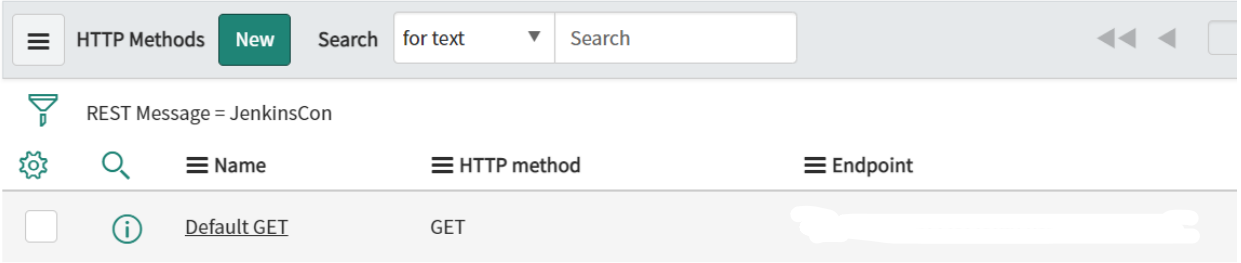
Go to the Outbound Rest Message option:



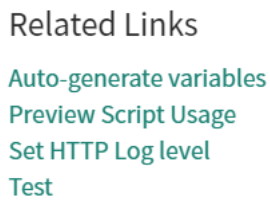
Choose your Jenkins Outbound rest message



Within the Rest message, go to Default Get HTTP Methods



b. **Within the Default Get screen, go to related links and click on the "Preview Script Usage"**



c. **View the code generated by ServiceNow as below:**

Preview REST Message script usage

Script

```
 try {
 var r = new sn_ws.RESTMessageV2('JenkinsCon', 'Default GET');

//override authentication profile
//authentication type ='basic'/ 'oauth2'
//r.setAuthenticationProfile(authentication type, profile name);

//set a MID server name if one wants to run the message on MID
//r.setMIDServer('MY_MID_SERVER');

//if the message is configured to communicate through ECC queue, either
//by setting a MID server or calling executeAsync, one needs to set skip_sensor
//to true. Otherwise, one may get an intermittent error that the response body is n
//r.setEccParameter('skip_sensor', true);

 var response = r.execute();
 var responseBody = response.getBody();
 var httpStatus = response.getStatusCode();
}
catch(ex) {
 var message = ex.message;
}
```

Copy this script. We will use this script into the ServiceNow workflow in the next step.

Go to workflow editor

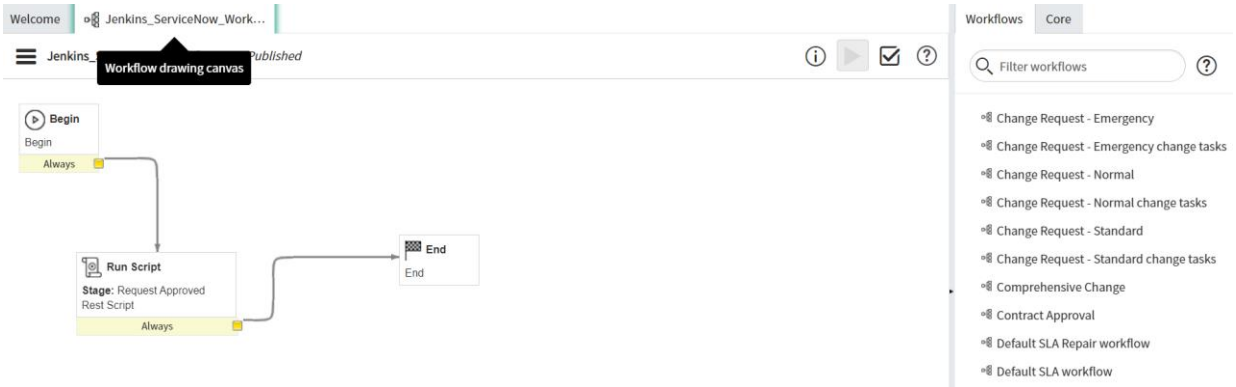
d. Create a new workflow

Jenkins_ServiceNow_Workflow_v1	Requested Item [sc_req_item]	admin	2021-01-26 06:19:37	true

Below are the stage of the workflow:

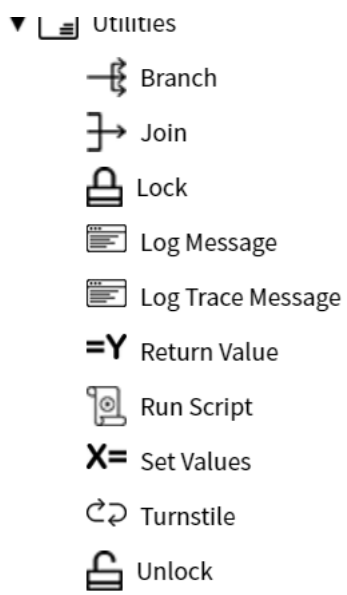
Stages of the workflow:

Begin → Run Script → End

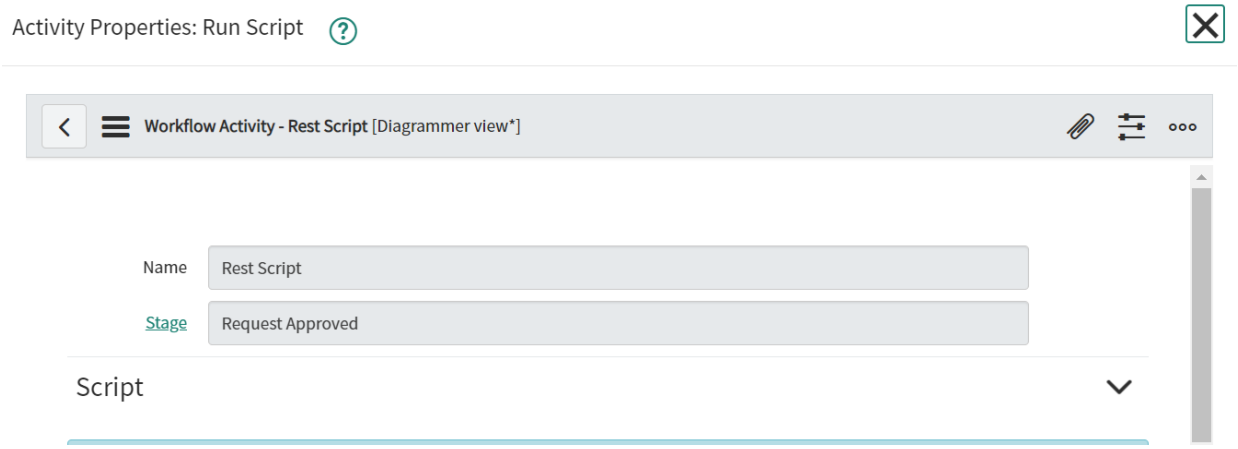


e. Run Stage description:

Run script is to be chosen from the workflow editor menu on the right had side (core option)



Run Script Configuration:



Past the actual script you extract from the Rest Message earlier into the script area.

Please remove the try and catch constructs since somehow this does not allow the execution of the script.

Also, do not touch other parameters unless you are using MID server. This example does not use the ServiceNow MID server.

Script

```
var r = new sn_ws.RESTMessageV2('JenkinsCon', 'Default GET');

//override authentication profile
//authentication type ='basic'/ 'oauth2'
//r.setAuthenticationProfile(authentication type, profile name);

//set a MID server name if one wants to run the message on MID
//r.setMIDServer('MY_MID_SERVER');

//if the message is configured to communicate through ECC queue, either
//by setting a MID server or calling executeAsync, one needs to set skip_sensor
//to true. Otherwise, one may get an intermittent error that the response body is null
//r.setEccParameter('skip_sensor', true);

var response = r.execute();
var responseBody = response.getBody();
var httpStatus = response.getStatusCode();
```

With this step, your workflow “Run script” is ready.

Ensure that you “Publish” the workflow.

6. Configure ServiceNow Catalog item

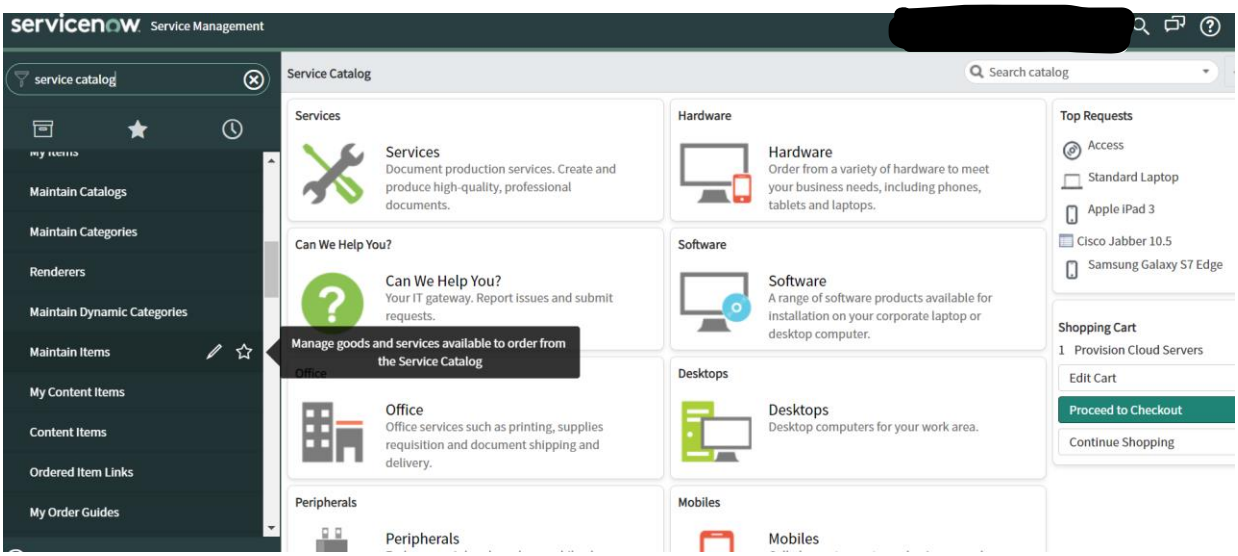
Type of catalog item – Technical Catalog

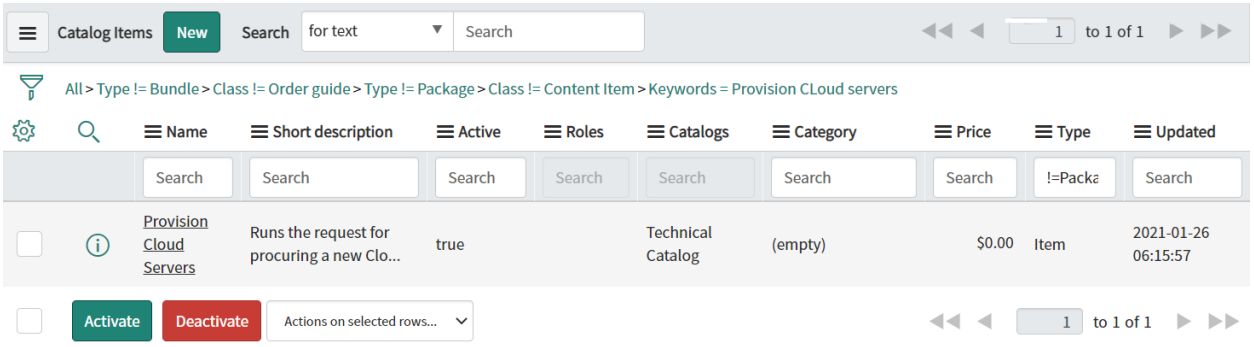
Application Scope – Global

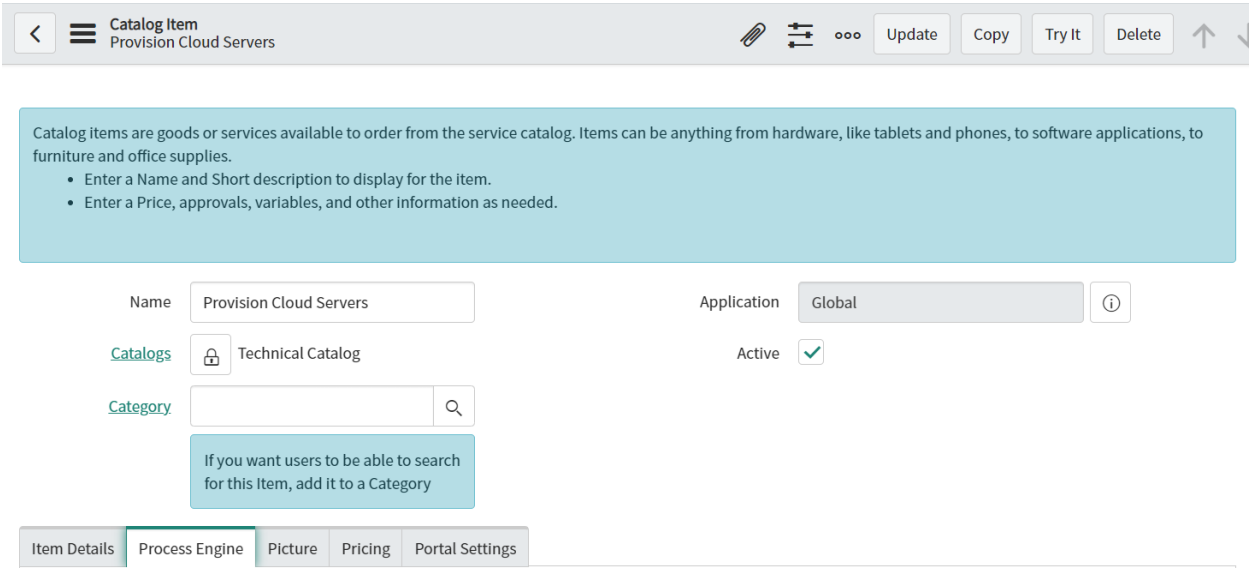
Create a service catalog item in ServiceNow

Create a new Technical Catalog item called “Provision Cloud Servers” (you could choose any name for the catalog)

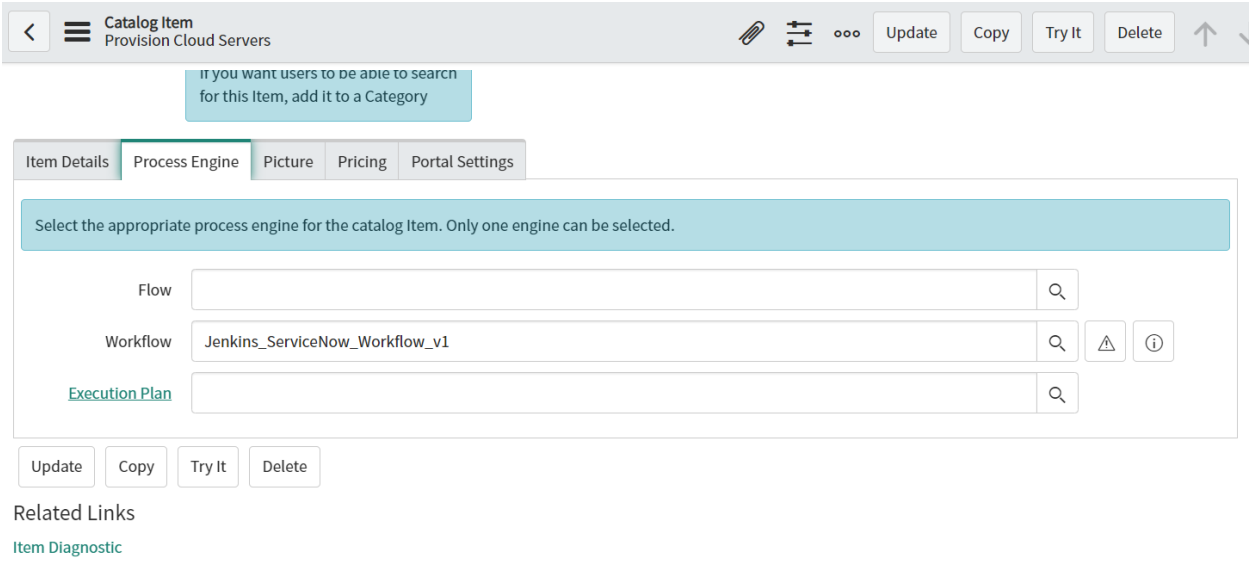
Go to Maintain items option to create a new Catalog item







Process Workflow:



To try out the workflow, please “TRY IT” and this will execute the catalog item.

The catalog item will in turn invoke the workflow, which will execute the REST Script and invoke the target Jenkins job. The Jenkins job in turn executes the AWS CLI command and provisions an EC2 server in AWS.