Craig ELlrod

senior cloud security architect

Deploying Armor Agent with TERRAFORM

Microsoft Windows

Table of Contents

[Introduction 2](#_Toc72931455)

[Prerequisites 2](#_Toc72931456)

[How it works 2](#_Toc72931457)

[Step-by-Step 3](#_Toc72931458)

[Step 1 – Add the curl script to the EC2 instance user data 3](#_Toc72931459)

[Step 2 – Add your AWS Key pair 3](#_Toc72931460)

[Step 3 – Add user data and keypair to the aws\_instance resource block 4](#_Toc72931461)

[Step 4 – Launch the EC2 instance 5](#_Toc72931462)

[Step 5 – EC2 instance shows up in the Armor Management Portal (AMP) 5](#_Toc72931463)

[Step 6 – SSH and validate armor install 6](#_Toc72931464)

[Step 6 – Tear down (optional) 7](#_Toc72931465)

# Introduction

This ARMOR Defense field guide will present a step-by-step method to deploy the Armor Agent within a TERRAFORM script for Linux.

## Prerequisites

An Amazon Web Services account.

<https://aws.amazon.com/free/>

Terraform getting started. Version 15 or higher is preferred.

<https://learn.hashicorp.com/collections/terraform/aws-get-started>

A Linux host – For example Linux Ubuntu – running as a virtual machine in VirtualBox, with Docker installed.

<https://virtualbox.org>

<https://ubuntu.com/download>

<https://learn.hashicorp.com/tutorials/terraform/install-cli>

The github for this tutorial.

<https://github.com/surfd4wg/armor_tutorial_tf_win>

Note: A Terraform step-by-step tutorial is located here. This tutorial is a required prerequisite, as Armor will not attempt to build a complete end-to-end tutorial of Terraform. Armor will simply augment the Terraform tutorial, to add in the specific pieces to get the Armor Agent installed from within a Terraform script.

<https://learn.hashicorp.com/collections/terraform/aws-get-started>

Note2: The last “Store Remote State” step of the tutorial is not required.

## How it works

Installing the Armor Agent for Linux using terraform involves these simple steps.

* Insert the PowerShell script into the EC2 instance user data 🡪 main.tf terraform file.
* Add your AWS keypair 🡪 main.tf terraform file.
* Launch the EC2 instance through terraform automation 🡪 init; plan; apply.
* The EC2 instance shows up in the Armor Management Portal (AMP).
* Tear down (optional).

# Step-by-Step

## Step 1 – Add the PowerShell script to the EC2 instance user data

The preferred method of installing the Armor Agent into a Linux EC2 instance, is by providing the installation ‘curl’ command in the user data of the EC2 instance when it launches.

Add the following to the main.tf terraform file. Replace the license key placeholder XXXXX-XXXXX-XXXXX-XXXXX-XXXXX with your AMP license key.

Note: The PowerShell installation script can be found by logging into the Armor Management Portal (<https://amp.armor.com>) and navigating to Infrastructure 🡪 Virtual Machines 🡪 + sign. Click on the shield. Select the Windows operating system.

Note2: Note the addition of the <powershell> </powershell> tags.

﻿locals {

user\_data = <<EOF

<powershell>

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12; Invoke-WebRequest https://agent.armor.com/latest/armor\_agent.ps1 -outfile armor\_agent.ps1; .\armor\_agent.ps1 -license XXXXX-XXXXX-XXXXX-XXXXX-XXXXX -region us-west-armor -full

</powershell>

EOF

}

## Step 2 – Add your AWS Key pair

In order to ssh into your Windows instance, you need to specify the AWS keypair to the EC2 instance at launch. The public key should be stored in the ~/.ssh directory, along with the private key.

To create an AWS keypair, follow the AWS instructions.

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

Once the private key .pem file has been downloaded and moved to the ~/.ssh directory, you need to create the public key from that file, by following the “Retrieve the public key for your key pair” section in the above link.

At this point you should have a .pub (public) key and a .pem (private) key in the ~/.ssh directory. Add the following terraform resource block to the main.tf terraform file. Replace the “public key name” with the name of your AWS public keypair filename.

﻿ ﻿resource "aws\_key\_pair" "terraform\_pub\_key" {

public\_key = file("~/.ssh/<public key name>.pub")

}

## Step 3 – Add your AWS private key to decrypt the Windows password in the output

The Administrator password is generated automatically; however, you need to decrypt it with the rsadecrypt function in terraform. The first step is to tell terraform where to find your private key. Add the following terraform variable block to the main.tf terraform file. Replace the “private key name” with the name of your AWS private keypair filename.

﻿variable "private\_key\_path" {

description = "Path to the private SSH key, used to access the instance."

default = "~/.ssh/<private key name>.pem"

}

## Step 4 – Decrypt the Administrator password in the output

In the previous step you instructed terraform where to find your private key. Now add some output code blocks to decrypt the Windows password and output it to the terminal. This password will be used later to log into the Windows server using an RDP client.

﻿output "Admin\_Username" {

value = "Administrator"

}

output "Admin\_Password\_Data" {

value = aws\_instance.app\_server.password\_data

}

output "Admin\_Password\_Data\_Decrypted" {

value = rsadecrypt(aws\_instance.app\_server.password\_data, file(var.private\_key\_path))

}

## Step 5 – Add user data and keypair to the aws\_instance resource block

Once the user\_data and keypair resources are added to the main.tf terraform, those two parameters need to be added to the aws\_instance resource block of the main.tf terraform file. Add the following to the aws\_instance resource block in the main.tf terraform file. There is one additional line, get\_password\_data = “true” we need to specify, so that terraform will get the password data from the Windows EC2 instance.

﻿ user\_data\_base64 = base64encode(local.user\_data)

key\_name = aws\_key\_pair.terraform\_pub\_key.key\_name

get\_password\_data = “true”

For example, the final code block for the main.tf terraform should look similar to the following. The “new” pieces to the main.tf terraform file are highlighted in orange.

Note: Theoretically, you could copy and paste the below code block into the main.tf terraform file, and run it, of course replacing your license key and keypair file names.

Note: Windows EC2 instances work best with a minimum of t3.medium instance type.

﻿terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.42"

}

}

required\_version = ">= 0.15"

}

provider "aws" {

profile = "default"

region = "us-east-1"

}

﻿resource "aws\_key\_pair" "terraform\_pub\_key" {

public\_key = file("~/.ssh/<public key file>.pub")

}

variable "private\_key\_path" {

description = "Path to the private SSH key, used to access the instance."

default = "~/.ssh/<private key file>.pem"

}

variable "instance\_name" {

description = "Value of the Name tag for the EC2 instance"

type = string

default = "ExampleAppServerInstance"

}

﻿locals {

user\_data = <<EOF

<powershell>

#Armor Agent

mkdir c:\armorinstall

cd c:\armorinstall

Invoke-WebRequest https://agent.armor.com/latest/armor\_agent.ps1 -outfile c:\armorinstall\armor\_agent.ps1

New-Item -Path . -Name "armorinstall.ps1" -ItemType "file" -Value ".\armor\_agent.ps1 -license XXXXX-XXXXX-XXXXX-XXXXX-XXXXX -region us-west-armor -full

.\armorinstall.ps1

</powershell>

EOF

}

resource "aws\_instance" "app\_server" {

ami = " ami-0fa60543f60171fe3"

instance\_type = "t3.medium"

user\_data\_base64 = base64encode(local.user\_data)

key\_name = aws\_key\_pair.terraform\_pub\_key.key\_name

get\_password\_data = “true”

tags = {

Name = var.instance\_name

}

}

output "instance\_id" {

description = "ID of the EC2 instance"

value = aws\_instance.app\_server.id

}

﻿

output "instance\_public\_ip" {

description = "Public IP address of the EC2 instance"

value = aws\_instance.app\_server.public\_ip

}

﻿output "Admin\_Username" {

value = "Administrator"

}

output "Admin\_Password\_Data" {

value = aws\_instance.app\_server.password\_data

}

output "Admin\_Password\_Data\_Decrypted" {

value = rsadecrypt(aws\_instance.app\_server.password\_data, file(var.private\_key\_path))

}

## Step 4 – Launch the EC2 instance

As you learned in the terraform tutorial, to launch the EC2 instance with the Armor Agent installed, enter the following commands in the terminal or CLI.

$terraform init

$terraform plan

$terraform apply

Note: If you log in to the AWS console, (<https://console.aws.amazon.com/console/home>) and navigate to EC2 instances you will see the instance spinning up.

Sample output from the terraform apply:

﻿Outputs:

Admin\_Password\_Data = "<base 64 encoded data>"

Admin\_Password\_Data\_Decrypted = "xXxXxXxXxXxXxXxXxXxXxXxXxXxXxXxX"

Admin\_Username = "Administrator"

instance\_id = "i-xxxxxxxxxxxxxxxxx"

instance\_public\_ip = "xxx.xxx.xxx.xxx"

## Step 5 – EC2 instance shows up in the Armor Management Portal (AMP)

Log in to the Armor Management Portal and navigate to Infrastructure 🡪 Virtual Machines.

Graphical user interface, application, table, Teams

Description automatically generated

## Step 6 – Connect using RDP and validate armor install

If you want to log into into the EC2 instance and check the armor install logs, you will need to use an RDP client.

Windows and Mac OS RDP clients can be downloaded here:

<https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-clients>

Linux RDP clients can be found here:

<https://remmina.org/how-to-install-remmina/>

Launch the RDP Client, and specify the EC2 instance public ip address. You will be prompted for the Windows password, that was output from the terraform automation script.

Graphical user interface, application

Description automatically generated

If the installation was successful, you will see the armor executable in the c:\.armor\opt directory.

The log file for the armor agent installation can be found in c:\.armor\log\armor.log.

You can cross check the EC2 instance Armor ‘CoreInstanceId’ with the machine in the Armor Management Portal, issue the following command in the PowerShell terminal inside the Windows instance.

Example input / output:

C:\.armor\opt\armor.exe show db

﻿ ACCOUNTID COREINSTANCEID

----------- --------------------------------------

XXXX 1d78d570-1aba-4314-8d32-1efd64fa0409

The CoreInstanceId from the CLI command should match the CoreInstanceId listed in the Armor Management Portal.

Graphical user interface, application, Teams

Description automatically generated

## Step 6 – Reboot (required)

When the Armor Agent is installed, it starts to pull down and install “sub-agents”. One of those sub-agents requires a Windows reboot to become fully operational.

There are a couple of different ways to reboot the Windows instance.

1. Issue the “Restart-Computer” command in a PowerShell terminal.

PS C:\.armor> Restart-Computer

1. Issue the Reboot command from the AWS console.

Graphical user interface, text, application, chat or text message

Description automatically generated

## Step 7 – Tear down (optional)

If you want to reverse the automation that terraform performed, simply enter the following commands.

﻿$ terraform destroy