Craig ELlrod

senior cloud security architect

Deploying Armor Agent with TERRAFORM

Linux

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_ubu>

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 curl 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 curl 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 curl installation script can be found by logging into the Armor Management Portal (<https://amp.armor.com>) and navigating to Infrastructure 🡪 Virtual Machines 🡪 + sign. Select the Linux operating system.

﻿locals {

user\_data = <<EOF

#!/bin/bash

sudo curl -sSL https://agent.armor.com/latest/armor\_agent.sh | sudo bash /dev/stdin -l XXXXX-XXXXX-XXXXX-XXXXX-XXXXX -r us-west-armor -f

EOF

}

## Step 2 – Add your AWS Key pair

In order to ssh into your Linux instance, you need to specify the AWS keypair to the EC2 instance at launch. 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.

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 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.

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

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

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 name.

﻿terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "~> 3.42"

}

}

required\_version = ">= 0.15"

}

provider "aws" {

profile = "default"

region = "us-east-1"

}

locals {

user\_data = <<EOF

#!/bin/bash

sudo curl -sSL https://agent.armor.com/latest/armor\_agent.sh | sudo bash /dev/stdin -l XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-r us-west-armor -f

EOF

}

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

ami = "ami-0db6c6238a40c0681"

instance\_type = "t2.micro"

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

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

tags = {

Name = var.instance\_name

}

}

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

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

}

variable "instance\_name" {

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

type = string

default = "ExampleAppServerInstance"

}

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

}

## 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.

## 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, Teams

Description automatically generated

## Step 6 – SSH and validate armor install

If you want to ssh into the EC2 instance and check the armor install logs, you can perform the ssh commands as listed in <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html>.

For example, for an Ubuntu 20.04 EC2 instance:

ssh -i ~/.ssh/<private key name>.pem ubuntu@<public ip address>

Note: The public ip address of the EC2 instance can be found in the AWS console and was also output as a part of the terraform automation.

Note2: Each distribution of Linux has a different username in AWS. In the above example, for Ubuntu, the username is ubuntu. For Amazon Linux 2, the username is ec2-user. Usernames for the different Linux versions can be found here: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/managing-users.html>.

If the installation was successful, you will see the armor executable in the /opt/armor directory.

The log file for the armor agent installation can be found in /var/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 Linux terminal.

Example input / output:

/opt/armor/armor show db

﻿ ACCOUNTID COREINSTANCEID

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

XXXX 5f9588b7-7a68-49b3-b6ea-f8592518fc6d

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 – Tear down (optional)

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

﻿$ terraform destroy