# IoT Static IP Endpoints

## Implementation Guide

# IoT Static IP Endpoints: Implementation Guide

# Table of Contents

# Create a secure VPN connection between IoT devices and AWS services using static IP addresses and a single port number

*February 2021*

This implementation guide describes architectural considerations and configuration steps for deploying IoT Static IP Endpoints in the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, developers, network engineers, and administrators who have practical experience with IoT devices, network routing, and architecting in the AWS Cloud. To ensure security and reliability, you need a deep understanding about how operating systems route traffic through client VPN software.

# Overview

The IoT Static IP Endpoints solution creates a secure virtual private network (VPN) connection with IoT devices by providing static IP addresses using a single port number. Network traffic is routed over the secure connection and then out to AWS service endpoints or other public Internet services. Furthermore, you can provide these static IP addresses to third-party security organizations, and have them added to their firewall rules.

Most AWS service endpoints are fully qualified domain name (FQDN) entries, such as *https://dynamodb.us-west-2.amazonaws.com* for accessing Amazon DynamoDB. To provide resilience and scale, this single FQDN resolves to a different set of IP addresses over time. This makes it difficult for firewalls to allow the outbound connections from devices to Amazon DynamoDB as it would necessitate allow-listing large ranges of IP addresses that change over time requiring continual firewall rule updates. By providing static IP addresses, this solution helps you maintain a secure network posture without the need for multiple IP address ranges to be opened in your firewall. Instead, IoT devices destined for multiple AWS service endpoints can tunnel through the static IP addresses.

For example, AWS IoT Greengrass requires connections to various AWS services for normal operation. This includes services such as Amazon Simple Storage Service (Amazon S3), AWS IoT Core, and Amazon CloudWatch. Each of these services has a service endpoint, and each service endpoint will resolve to continuously changing IP address ranges over time. By using this solution, only a few static IP addresses would need to added to your firewalls, and will not change over time.

OpenVPN is used as the VPN system to create a secure client-to-server connection in a routed configuration mode. This solution deploys the OpenVPN server instances as configuration. Each IoT device client is required to run an OpenVPN client in order to route device traffic to AWS service endpoints.

This guide provides infrastructure, networking, and configuration information for planning and deploying IoT Static IP Endpoints in the AWS Cloud.

# Cost and licenses

You are responsible for the cost of the AWS services used while running this solution. At the date of publication, the cost for running this solution with the default settings in the US East (N. Virginia) Region using two t3.small Amazon Elastic Compute Cloud (Amazon EC2) instances in public subnets is approximately **$54.80 / month**, and **$0.164 per GB of data sent** from the AWS Cloud to your IoT devices. Table 1 shows the breakdown of costs by AWS service excluding data transfer.

Table 1: Monthly cost for AWS services

| AWS Service | Total Cost for 730 Hours (Monthly) |
|---|---|
| Amazon EC2 | $30.66 |
| Amazon Elastic File System (EFS) | $0.30 |
| Network Load Balancing (NLB) | $16.44 |
| CloudWatch Metrics | $4.10 |

| AWS Service | Total Cost for 730 Hours (Monthly) |
|---|---|
| CloudWatch Logs | $0.30 |
| CloudWatch Dashboards | $3.00 |
| **Total:** | **$54.80 / month** |

Additionally, factor in the cost for each gigabyte of data transferred through the Network Load Balancer and outbound data from the AWS Cloud to your IoT devices. Table 2 shows the estimated data processing costs for a *single device* sending 10 GB of data to the AWS Cloud, and the AWS Cloud sending 10 GB of data to the device using this solution. This estimate is based on devices with consistent connections. If your device usage patterns differ, the Network Load Balancer costs may differ. For information about Network Load Balancer pricing, refer to the Network Load Balancer tab on the Elastic Load Balancing pricing page.

Table 2: Estimated data processing costs

| AWS Service | Total Cost for 10 GB in / 10 GB out |
|---|---|
| Network Load Balancer (NLB) | $0.74 |
| Data transfer from the AWS Cloud to one IoT device | $0.90 |
| Data transfer from one IoT device to the AWS Cloud | $0.00 (no cost) |

**Note**
While this solution can be deployed in one of the unavailable Regions for AWS IoT Core, all network traffic will flow between the Regions and incur additional costs. We recommend deploying the solution in a Region where targeted service endpoints are available.

# Costs for optional features and AWS services

Activating the optional features when deploying this solution increases the costs, both as an hourly cost for the feature along with per gigabyte of data of inbound and outbound data processed. Table 3 lists the optional features and their estimated monthly costs.

Table 3: Estimated costs for optional AWS services

| AWS Service | Total Cost for 730 Hours (Monthly) |
|---|---|
| AWS Global Accelerator (excluding data transfer) | $18.00 |
| AWS Global Accelerator Data Transfer-Premium (per GB) | $0.015 to $0.091 / GB |
| NAT Gateway | $65.70 |
| NAT Gateway per GB processed | $0.045 / GB |

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

# Software licenses

During deployment, third-party software packages are installed from the Extra Packages for Enterprise Linux (EPEL) repository. This solution is licensed under Apache 2.0. Table 4 lists the additional packages with their corresponding licenses that are installed.

Table 4: Software packages and licenses

| Package name | License |
|---|---|
| OpenVPN | GNU GPLv2 |
| EasyRSA | GNU GPLv2 |
| Socat | GNU GPLv2 |

# Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

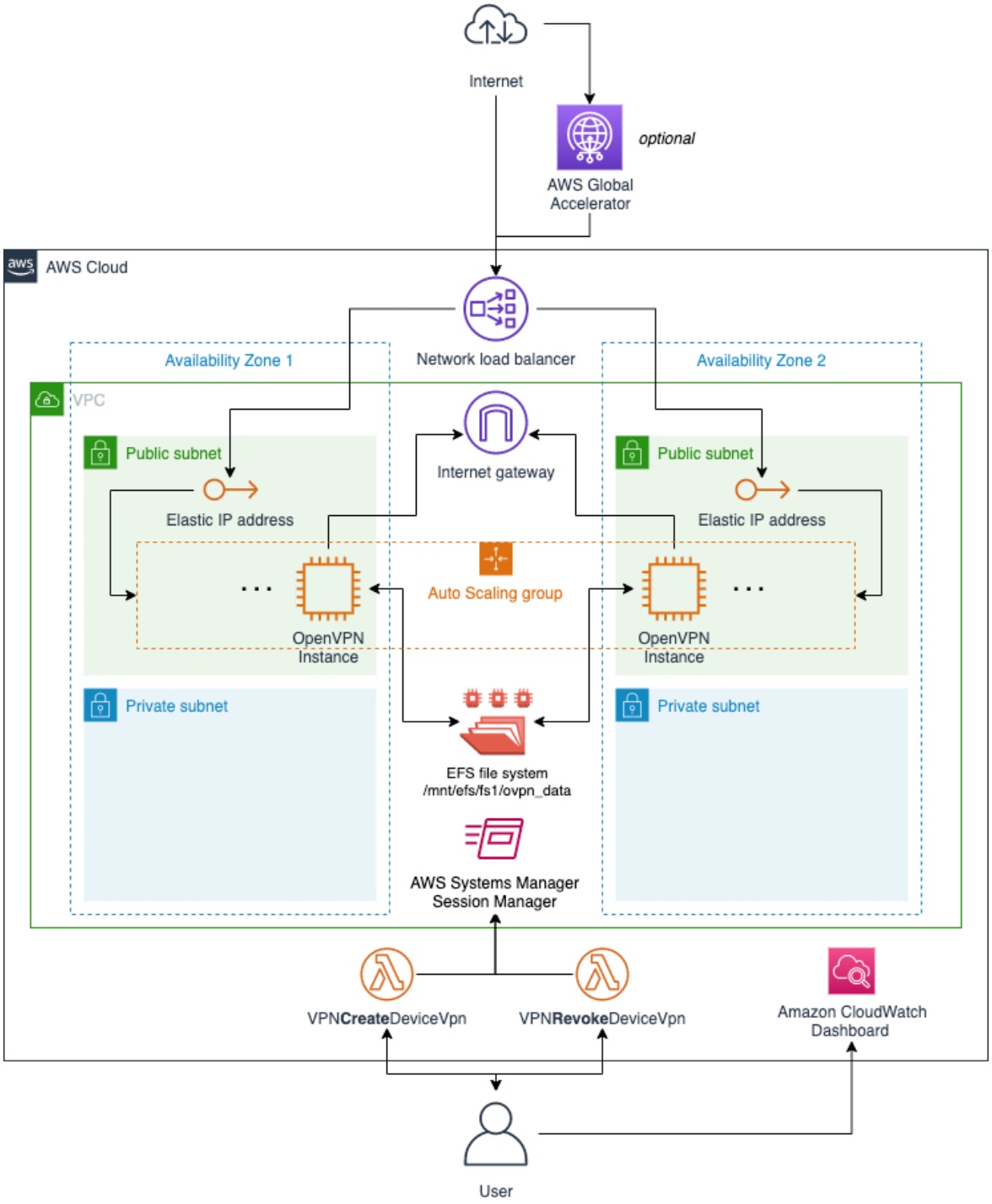


**Figure 1: IoT Static IP Endpoints without NAT Gateways architecture on AWS**

The AWS CloudFormation template deploys an Amazon Virtual Private Cloud (Amazon VPC) with a public and a private subnet in two Availability Zones (AZs). Within the Amazon VPC, an Auto Scaling Group (ASG) deploys a range of instances that run the OpenVPN server software. An Elastic File System (EFS) share is created and mounted as `/mnt/efs/fs1/ovpn_data`, and used as the common location for all OpenVPN software configurations.

A Network Load Balancer (NLB) is set up with the appropriate protocol, either UDP or TCP, and a port number on which it listens. It also allocates an Elastic IP (EIP) address for each AZ, which serves as the static IP address for incoming connections.

# Client interaction with this solution

Once a client configuration is created and deployed to a client, the interaction takes place as shown in Figure 2.

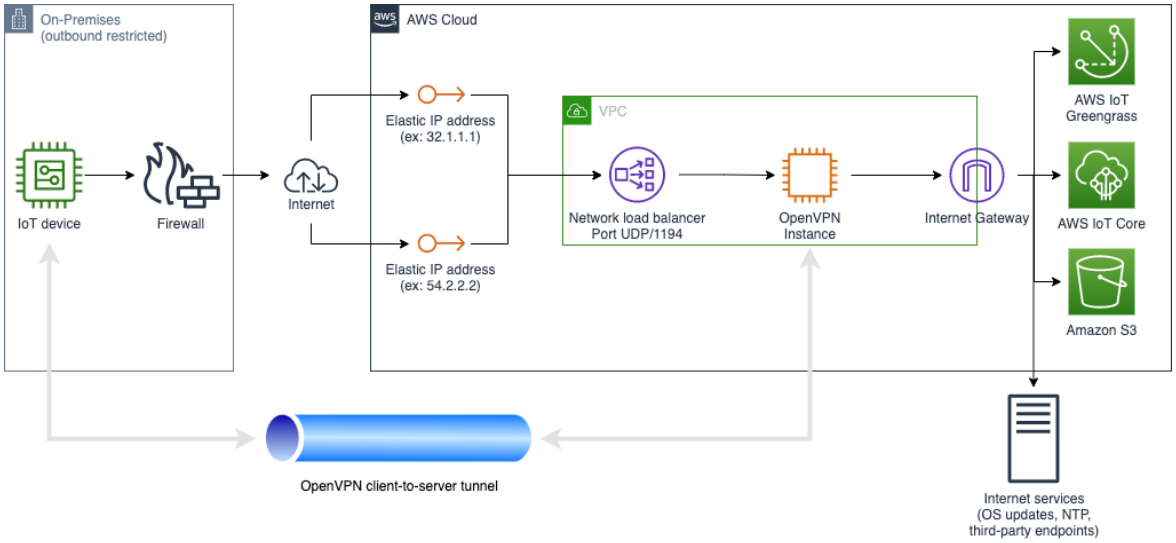


**Figure 2: Client interaction**

The IoT device starts the OpenVPN client software locally, and connects through a local on-premises firewall to one of the two static IP addresses listed in the configuration file. The protocol and port (default is UDP/1194) are also set in the configuration file. The connection is established on an OpenVPN instance. The instance and the IoT device perform X.509 mutual authentication and then completes the VPN tunnel. Network traffic flows through the tunnel on UDP/1194 in an encrypted form, then it is unencrypted and forwarded to other public AWS service endpoints or other Internet services.

For example, an IoT device connecting to AWS IoT Core resolves the IP address for the service endpoint (such as, `a2l4icd999aaab-ats.iot.us-east-1.amazonaws.com`) and sends the connection request through the VPN tunnel. The instance forwards traffic though the Internet gateway to AWS IoT Core.

> **Note**
> An OpenVPN session between an IoT device and an OpenVPN instance is encrypted end-to-end. All traffic from the source IoT device to any public Internet destination is unmodified, with the exception that a destination server will identify traffic originating from the OpenVPN instances public IP address as the source IP address instead of from the IoT device itself. This source IP address will either be one given at the time the instance starts up, or if using NAT Gateway, the public IP address assigned to the NAT Gateway will be used.

# Solution components

## Solution monitoring

The IoT Static IP Endpoints solution includes a set of Amazon CloudWatch components to monitor and report on the state of the solution and provide email notifications for important events, such as replacement or scaling out of instances. Deploying this solution creates a set of Amazon CloudWatch metrics and an Amazon CloudWatch dashboard that monitors the health and status of the solution. You can configure this solution to send instance data to Amazon CloudWatch Logs for use in the AWS Cloud.

## OpenVPN client configuration

This solution includes two AWS Lambda functions that request either the creation or revocation of an OpenVPN client configuration.

- **VPNCreateDeviceVpn**—This Lambda function creates a device VPN configuration including endpoints, connection options, X.509 certificate, and a private key for the unique device.
- **VPNRevokeDeviceVpn**—This Lambda function immediately revokes a previously created device configuration and updates the certificate revocation list for all instances, permanently revoking the unique certificate for that device.

You can automate the method to configure clients by using the AWS Command Line Interface (AWS CLI) to issue commands directly from the instances.

# Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

## IAM roles

AWS Identity and Access Management (IAM) roles allows customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to create Regional resources. For more information, refer to IAM roles in the *AWS IAM User Guide*.

## Security groups

The security groups created in this solution are designed to control and isolate network traffic between the IoT devices (Peer CIDR ranges) and the OpenVPN instances. We recommend that you review the security groups and further restrict access as needed once the deployment is up and running.

# Implementation considerations

## Optional services

This solution offers optional services during deployment of the AWS CloudFormation template such as NAT Gateways and AWS Global Accelerator. Depending on your requirements, the use of these additional services may either change the architecture for the deployed resources or change the method in which data is transferred between an IoT device and the static IP addresses.

### NAT Gateways

When activated, a NAT Gateway is deployed in each Availability Zone. All traffic between the IoT device and OpenVPN server instance is routed through the static IP address deployed by this solution. When activated, traffic from the OpenVPN service instance to public Internet services is processed through the NAT Gateway instead of from the random public IP address assigned to the OpenVPN instance. The NAT Gateway feature is not enabled by default as using this feature will incur additional costs (p. 2).

The benefits of using a NAT Gateway include:

- OpenVPN instances are in private IP address spaces
- Connections initiated outbound to the public Internet uses a non-changing public IP address

This option can be used to track and allow traffic from the non-changing public IP addresses provided by this solution. By default, the **Use NAT Gateways** parameter is set to `No`.

### AWS Global Accelerator

When activated, AWS Global Accelerator reduces packet loss, jitter, and latency by using the AWS global network infrastructure. This is beneficial for devices that are located in areas where normal Internet routing affects packet loss, jitter, or latency.

By default, the **Activate Global Accelerator** parameter is set to `No`. In this configuration, traffic to and from this solution uses standard Internet routing to the Elastic IP addresses that are either provisioned or provided. For most use cases which allow two or more static IP endpoints, this configuration is preferred as it does not incur any additional costs. If you need a singular, highly available static IP address, then AWS Global Accelerator can support this need.

## Optional template architecture: AWS Global Accelerator

An optional architecture using AWS Global Accelerator and NAT Gateways can also be deployed to provide static IP addresses for inbound connections using AWS Global Accelerator addresses, and for outbound connections using the NAT Gateway static IP addresses. This optional architecture is useful for those deployments where static IP addresses in both directions are required.

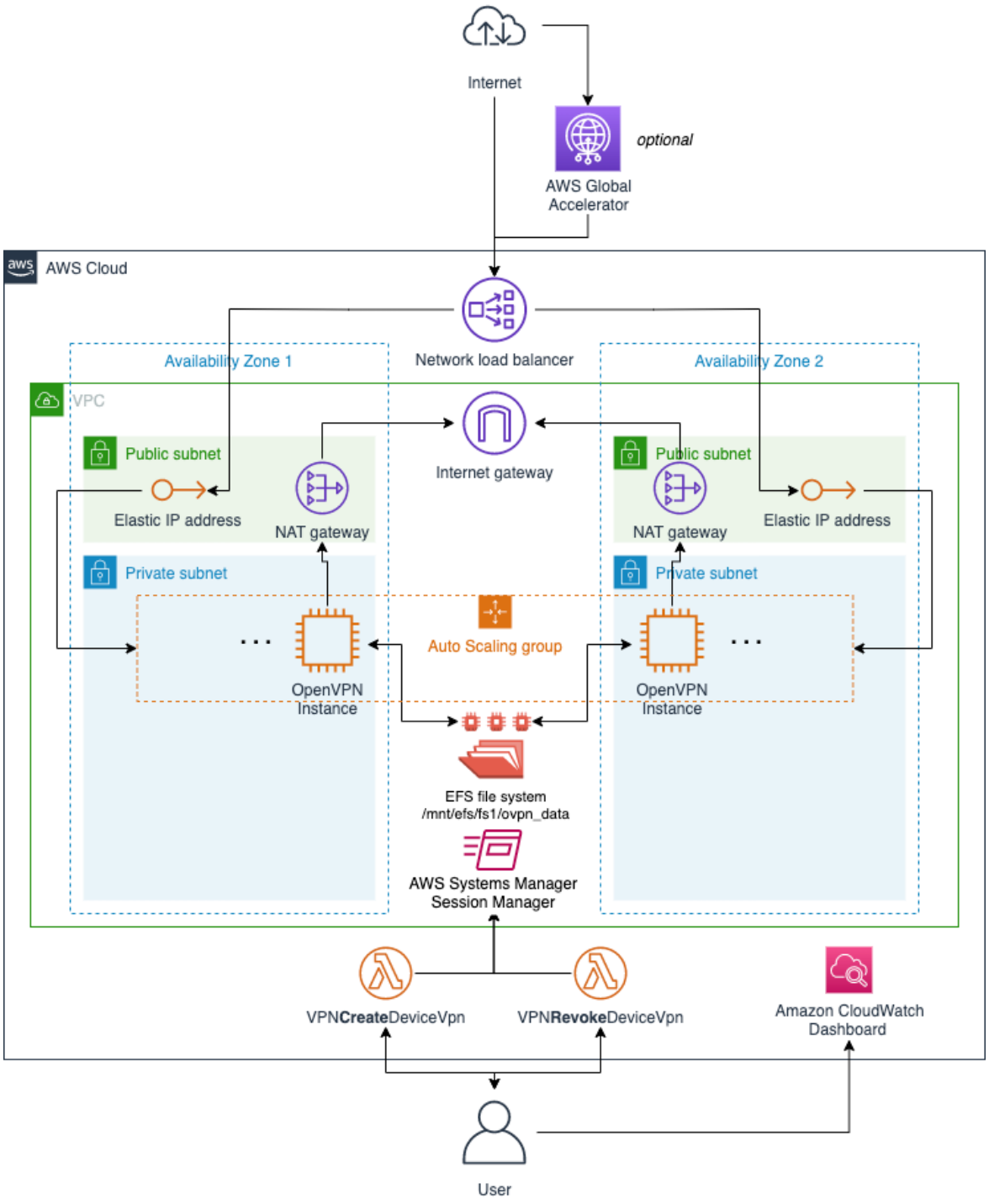


**Figure 3: IoT Static IP Endpoints with NAT Gateways architecture on AWS**

This optional architecture, depicted in Figure 3, introduces NAT Gateways that provide non-changing IP addresses for outbound connections, or if a corporate policy requires the OpenVPN instances to reside within a private subnet. To activate this functionality, change the **Use NAT Gateways** parameter to `Yes` when deploying the AWS CloudFormation template.

# Data retention

This solution retains the OpenVPN data stored with the EFS file system, the log files created by this solution's components, and the log files for the EC2 instances. By default, data is retained for one year (365 days), but you can change the length of time by editing the **Log Retention Days** parameter. Changing this parameter to either a shorter or longer retention duration affects the costs for the Amazon CloudWatch Logs service. We recommend setting the retention period to meet your operational or compliance needs.

**Note**
Uninstalling this solution does not delete Amazon CouldWatch Logs, even if the **CloudWatch Logs Retention Policy** parameter is set to `No`. Refer to Uninstall the solution (p. 27) to remove these logs and other unwanted resources.

# Regional deployments

This solution uses AWS services that are available in all AWS Regions. However, targeted endpoints such as AWS IoT Core are not currently available in all Regions. While this solution can be deployed in one of the unavailable Regions for AWS IoT Core, all network traffic will flow between the Regions and incur additional costs (p. 2). We recommend deploying the solution in a Region where targeted service endpoints are available.

# AWS CloudFormation template

This solution uses AWS CloudFormation to automate the deployment of the IoT Static IP Endpoints solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment:


**iot-static-ip-endpoints.template:** Use this template to launch the solution and all associated components. The default configuration deploys:

- An Amazon Virtual Private Cloud network topology
- Amazon Amazon Elastic Compute Cloud t3.small instances
- Elastic Load Balancing Network Load Balancer
- Elastic IP addresses
- AWS Systems Manager agents
- AWS Identity and Access Management roles
- Amazon Simple Notification Service topic and email
- Amazon CloudWatch metrics
- Amazon CloudWatch Logs
- Amazon CloudWatch dashboards
- An Elastic File System shared file system for instances
- An OpenVPN package installed from the EPEL repository

You can customize this template to meet your specific needs.

# Automated deployment

Before you launch the solution, review the architecture, configuration, network security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

**Time to deploy:** Approximately 10 minutes

## Prerequisites

Before deploying this solution, determine the following:

- Whether the OpenVPN instances needs to reside in public or private subnets for your use case. If there are services where a source static IP address would be useful, or there are third-party services that only filter on IP addresses, activate the NAT gateway feature. When activated, traffic originates from static IP addresses assigned to the NAT gateway. If not activated, a random IP address is assigned and used by each Amazon EC2 instance.
- Whether you are able to install and configure the OpenVPN client software on your IoT devices and create the client configuration file generated by this solution. Refer to Implementation considerations (p. 9) for guidance.

Also, ensure that you have an AWS account or a role with sufficient permissions to deploy the AWS resources. Refer to Security (p. 8) for more information.

## Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Step 1. Launch the stack (p. 14)

- Launch the AWS CloudFormation template into your AWS account, in your desired Region.
- Enter a value for the required parameter: **Stack name**.
- Review the other template parameters, and adjust if necessary. We recommend reviewing the following parameters and ensuring that the default values are suitable for your deployment needs: **CA Valid Days**, **Peer CIDR**, **Notifications Email**, **Log Retention Days**, and **Activate VPC FlowLogs Delivery to CloudWatch**.

Step 2. Document the static IP addresses and the OpenVPN protocol/port (p. 20)

- Provide the static IP addresses, protocol, and port information to those responsible for configuring the firewall or security devices between the IoT device and the Internet.

Step 3. Create an OpenVPN client device configuration file (p. 20)

- Use the AWS Command Line Interface (AWS CLI) to generate an OpenVPN client configuration file for a test device.

Step 4. Test connecting an OpenVPN client to this solution (p. 22)

- Use the configuration file from Step 3 to verify communication with this solution using static IP addresses.

Step 5. Revoke the OpenVPN client configuration (p. 22)

- Use AWS CLI to revoke the test client configuration.
- Use the configuration file from Step 4 to verify that the revoked certificate no longer allows communication with this solution.

# Step 1. Launch the stack

This automated AWS CloudFormation template deploys the IoT Static IP Endpoints solution in the AWS Cloud. Review the prerequisites before launching the stack.

**Note**
You are responsible for the cost of the AWS services used while running this solution. For more details, visit the Cost (p. 2) section in this guide, and refer to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and select the button to launch the `iot-static-ip-endpoints` AWS CloudFormation template.


   Alternatively, you can download the template as a starting point for your own implementation.
2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

   **Note**
   This solution uses AWS services that are available in all commercial regions. However, targeted endpoints such as AWS IoT Core are not currently available in all AWS Regions. While the solution can be deployed in one of the unavailable Regions for AWS IoT Core, all network traffic will flow between the Regions and incur additional costs (p. 2). We recommend deploying the solution in a Region where targeted service endpoints are available.
3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, refer to IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

| Parameter | Default | Description |
|---|---|---|
| **Amazon VPC configuration** | | |
| Zone 1 | *<Requires input>* | Choose the Availability Zone for Zone 1. Zone 1 and Zone 2 must be different values. Ensure that the instance type |

| Parameter | Default | Description |
|---|---|---|
| | | selected is available in the selected zone. |
| Zone 2 | *<Requires input>* | Choose the Availability Zone for Zone 2. Zone 1 and Zone 2 must be different values. Ensure that the instance type selected is available in the selected zone. |
| VPC CIDR | `10.249.0.0/24` | The private IP address range used for all subnets. Change to another range if you intend to peer with any of the deployed VPC subnets. |
| Use NAT Gateways | `No` | Choose whether to activate the NAT Gateways. The default is `No`, which means the instances deploy in the public subnets. When set to `Yes`, the instances deploy in the private subnets and the NAT Gateways are deployed for outbound Internet traffic. |
| NAT Gateway Zone 1 - EIP Allocation ID | <Optional input> | This parameter is used only when the **Use NAT Gateways** parameter is set to `Yes`.<br><br>Enter an Allocation ID (for example, `eipalloc-64d5890a`) of an existing, unused Elastic IP address to be used for the NAT Gateway in Zone 1. If left blank, an Elastic IP is automatically provisioned during deployment. The ID is deleted when this solution is uninstalled. |

| Parameter | Default | Description |
|---|---|---|
| NAT Gateway Zone 2 – EIP Allocation ID | <Optional input> | This parameter is used only when the **Use NAT Gateways** parameter is set to `Yes`.<br><br>Enter an Allocation ID (for example, `eipalloc-64d5890a`) of an existing unused Elastic IP address to be used for the NAT Gateway in Zone 2. If left blank, an Elastic IP is automatically provisioned during deployment. The ID is deleted when this solution is uninstalled. |
| **Load balancer configuration** | | |
| Port | `1194` | The port used by the Network Load Balancer for incoming OpenVPN connections. |
| NLB Zone 1 – EIP Allocation ID | <Optional input> | Enter an Allocation ID (for example, `eipalloc-64d5890a`) of an existing unused Elastic IP address to be used for the Network Load Balancer in Zone 1. If left blank, an Elastic IP is automatically provisioned during deployment, and deleted when this solution is uninstalled. |
| NLB Zone 2 – EIP Allocation ID | <Optional input> | Enter an Allocation ID (for example, `eipalloc-64d5890a`) of an existing unused Elastic IP address to be used for the Network Load Balancer in Zone 2. If left blank, an Elastic IP is automatically provisioned during deployment, and deleted when this solution is uninstalled. |
| **AWS Global Accelerator configuration** | | |
| Activate Global Accelerator | `No` | Choose whether to activate AWS Global Accelerator. By default, this service is not activated. If set to `Yes`, a Global Accelerator is deployed, providing IP addresses for IoT devices. |

| Parameter | Default | Description |
|---|---|---|
| Global Accelerator IP 1 - Bring Your Own IP Address | <Optional input> | If Global Accelerator is activated, this parameter provides the IP address (for example, `1.2.3.4`) to use from an available pool of Bring Your Own IP Addresses. If left blank, AWS Global Accelerator automatically provisions an IP address during deployment. The address is deleted when this solution is uninstalled. |
| Global Accelerator IP 2 - Bring Your Own IP Address | <Optional input> | If Global Accelerator is activated, this parameter provides the IP address (for example, `1.2.3.4`) to use from an available pool of Bring Your Own IP Addresses. If left blank, AWS Global Accelerator automatically provisions an IP address during deployment, and deletes the address when this solution is uninstalled. |
| **Amazon VPN configuration** | | |
| VPN Tunnel Protocol | `UDP` | The TCP/IP protocol used by OpenVPN clients to communicate with the OpenVPN instances. Either `UDP` or `TCP` can be selected. The default setting is `UDP`, which is strongly recommended to avoid TCP Meltdown. |
| Auto Scaling Group - Min Capacity | `2` | The minimum number of EC2 instances to provision for OpenVPN. The range limit depends on the CIDR range that is used and the IP limits in your subnet. Addresses below 50 is recommended for configurations and CIDR ranges. |

| Parameter | Default | Description |
|---|---|---|
| Auto Scaling Group - Max Capacity | `10` | The maximum number of EC2 instances provisioned for OpenVPN. Capacity is increased or decreased based on instance CPU utilization. The range limit depends on the CIDR range that is used and the IP limits in your subnet. Addresses below 50 is recommended for configurations and CIDR ranges. |
| Instance AMI | `/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2` | The Amazon Machine Image (AMI) serves as the base operating system. The default is Amazon Linux 2. Changing to another operating system type requires changing the user data and scripts to deploy OpenVPN. For information about the AMI format, refer to Query for the latest Amazon Linux AMI IDs in the *AWS Compute Blog*. |
| Instance Type | `t3.small` | The instance family (t3) and size (small) to use. The default is suitable for supporting hundreds of OpenVPN clients and megabytes of throughput. |
| CA Valid Days | `3653` | The OpenVPN Certificate Authority validity. The default is 10 years, which is specified in days. Once the CA certificate expires, a new CA and all client configurations will need to be regenerated. |
| OpenVPN Keepalive Seconds | `10` | The interval in seconds that a keepalive message will be sent from the client to the server. The default value is to send a message every 10 seconds, and can be adjusted between 1 and 60 seconds. The 60 second maximum is to ensure that the Network Load Balancer setting of 120 seconds for UDP connections is not exceeded (this ensures a timeout is detected on the client side before the service side drops the connection). |

| Parameter | Default | Description |
|---|---|---|
| **Security and monitoring** | | |
| Peer CIDR | `0.0.0.0/0` | The remote CIDR range to permit ingress traffic to the OpenVPN service. If a discrete amount of ranges are required, you will need to manually modify the `VPNEC2SecurityGroup` EC2 security group created during deployment. For information about Amazon EC2 security groups, refer to Security groups for your VPC in the *Amazon VPC User Guide*. |
| Notifications Email | <Optional input> | Enter an email address to receive notifications about events, such as auto scaling. You must confirm the subscription request in order to receive these notifications. |
| **Logging configuration** | | |
| Log Retention Days | `7` | Sets the number of days to retain CloudWatch Logs log entries. By default, logs are retained for one week. |
| Activate VPC FlowLogs Delivery to CloudWatch | `No` | Choose whether to deliver the Amazon VPC flow logs to the CloudWatch Logs log group. This capability is deactivated by default. |
| **Data retention policies** | | |
| EFS Retention Policy | `Retain` | Choose whether to `retain` or `delete` the EFS file share when you uninstall the solution. If retained, you can save the OpenVPN configurations for future use. |
| CloudWatch Logs Retention Policy | `Retain` | Choose whether to `retain` or `delete` the CloudWatch Logs log group when you uninstall the solution. If retained, you can save the logging entries for audit or other purposes. |

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.

9. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should receive a **CREATE_COMPLETE** status in approximately 10 minutes.

If there is an error during the deployment of the stack, you will receive a **CREATE_FAILED** status in the Events tab, where all of the resources will be deleted. Review the **Status** reason, delete the stack from the **ROLLBACK_COMPLETE** status, correct the error, and launch the stack again.

**Note**
In addition to the AWS Lambda functions that make up the IoT Static IP Endpoints services, this solution deploys additional Lambda functions, which runs only during initial configuration or when resources are updated or deleted.
When you run this solution, all Lambda functions are listed in the AWS console, including ones that display *<your-stackname>-<FunctionName>*. Do not delete these Lambda functions, as they are necessary to manage associated resources.

# Step 2. Document the static IP addresses and protocol/port

Use this procedure to document the static IP addresses and protocol information that allows IoT or on-premises devices to connect to this solution. This information may be needed by your organization's information security group responsible for configuring firewalls.

1. From the AWS CloudFormation console, left menu pane, select **Stacks**.
2. On the **Stacks** page, select this solution's stack name.
3. From the solution's stack page, select the **Outputs** tab.
4. Record the IP addresses that are used as the *destination* in security devices:
   - If using AWS Global Accelerator, record the values for **AcceleratorGaIp1** and **AcceleratorGaIp2**.
   - If you are not using AWS Global Accelerator, record the values for **NLBServiceNlbEip1** and **NLBServiceNlbEip2**.
5. Select the **Parameters** tab and record the values for **VPNProtocol** and **Port**. These are used as the *destination protocol* and *port* in firewall security devices.
6. Provide these details to the information security group responsible for configuring firewalls.

**Note**
We recommend saving the destination IP addresses, VPNProtocol, Port, and their corresponding values for future use.

# Step 3. Create an OpenVPN client device configuration file

Use this procedure to create an OpenVPN client configuration file. These steps create a test configuration that you can use and delete later. You can use these steps to create your OpenVPN configuration files for IoT devices. All configurations and deletions are managed through AWS Lambda functions that connect to an OpenVPN instance. The response of the Lambda functions contain the complete OpenVPN

configuration for a specific device name, including the private key. At no time is the private key stored or logged within the solution, nor is it retrievable if lost.

**Note**
The command lines provided in the following steps are used to either create or revoke a certificate using AWS CLI and Linux commands. Optionally, you can use AWS CloudShell to enter these commands. However, if you are using a different computing environment, some of the commands may differ.

1. From your terminal, enter the following command to set the Region where the stack was deployed and the client name for issuing a configuration file:

```
export AWS_REGION=<your-aws-region>
export CLIENT_NAME=<your-client-name>
```

2. Enter the following command to query and set the name of the AWS Lambda function:

```
export LAMBDA_FUNCTION=$(aws lambda list-functions --region $AWS_REGION --query
 'Functions[?contains(FunctionName, `VPNCreateDeviceVpn`) == `true`].FunctionName' --
output text)
```

3. Enter one of the following commands to invoke the Lambda function. Extra characters will be removed from the name, for example **your-client-name** is recorded as **yourclientname**.

   - Use the following command if you are using AWS CLI version 2:

```
aws lambda invoke \
  --region $AWS_REGION \
  --function-name $LAMBDA_FUNCTION \
  --cli-binary-format raw-in-base64-out \
  --payload '{"ClientName": "'"$CLIENT_NAME"'"}' \
  $CLIENT_NAME.ovpn && \
awk  '{gsub("\\\\n","\n")};1' < $CLIENT_NAME.ovpn | \
awk 'NR>2 {print last} {last=$0}' > $CLIENT_NAME.$$ && \
mv $CLIENT_NAME.$$ $CLIENT_NAME.ovpn
```

   - Use the following command if you are using AWS CLI version 1:

```
aws lambda invoke \
  --region $AWS_REGION \
  --function-name $LAMBDA_FUNCTION \
  --payload '{"ClientName": "'"$CLIENT_NAME"'"}' \
  $CLIENT_NAME.ovpn && \
awk  '{gsub("\\\\n","\n")};1' < $CLIENT_NAME.ovpn | \
awk 'NR>2 {print last} {last=$0}' > $CLIENT_NAME.$$ && \
mv $CLIENT_NAME.$$ $CLIENT_NAME.ovpn
```

The command that invokes the Lambda function passes the client name and returns the configuration as a string. The chained set of commands formats the Lambda response and saves it as a local file in AWS CloudShell. If you deployed this solution on a local system and the commands do not run, use AWS CloudShell, AWS Cloud9, or any instance running Amazon Linux 2.

You can also generate a certificate signing request (CSR) from a local private key and pass that as a parameter to the Lambda function. The CSR must be a PEM-formatted serialized string. Use the following command, but replace the payload parameter for the CSR:

```
--payload '{"ClientName": "testclient", "CSR": "===Begin===\n..."}'
```

**Warning**
The generated OpenVPN configuration file contains the private key used to authenticate. Do not reuse or share this private key and ensure this data is kept confidential. This private key is not stored in your AWS account and cannot be retrieved again.

**Note**
OpenVPN manages client names using alphanumeric characters only. For example, if you enter the client name `test-client`, OpenVPN converts this entry to `testclient`. In this scenario, attempting to create another client named `test_client` (using an underscore) or `testclient` returns a **Certificate file already exists** error. We recommend using a unique naming convention for your IoT devices.

# Step 4. Test connecting an OpenVPN client to the solution

Use this procedure to test the configuration file on a macOS or Microsoft Windows operating system. The goal is to ensure that a unique IPv4 address is created after the OpenVPN connection has run.

1. Download and install and OpenVPN client software for your operating system.
   - For macOS, use either the OpenVPN Connect Client or the alternative Tunnelblick client.
   - For Microsoft Windows, use the OpenVPN Connect for Windows client.
   - For Linux environments, use the OpenVPN Connect for Linux client.
2. Copy the OpenVPN client configuration file created in Step 3 (p. 20), `test-client.ovpn` to your target environment as defined by the OpenVPN documentation.
3. Launch the OpenVPN client and follow the OpenVPN Client instructions to import the `test-client.ovpn` configuration file.
4. Deactivate other VPN clients and then record your current public IP from ifconfig.me.
5. From the OpenVPN client, connect with the test client configuration and monitor the log files during the connection attempt.
6. Once the connection has been established, refresh the ifconfig.me web page and verify that a different IPv4 address is shown.
7. When completed, disconnect the OpenVPN connection.

At this point, Internet traffic is routing through the IoT Static IP Endpoints solution. To monitor the OpenVPN connections, access the Amazon CloudWatch console, then access **Dashboards**.

# Step 5. Revoke the OpenVPN client configuration

Use the following step to manually revoke the test OpenVPN client configuration, and verify the test client from Step 4 can no longer connect.

From your terminal, revoke the client certificate by running the `RevokeDeviceVpnCertificate` Lambda function.

```
export AWS_REGION=<your-aws-region>
export CLIENT_NAME=<your-client-name>

export LAMBDA_FUNCTION=$(aws lambda list-functions \
  --region $AWS_REGION \
```

```
--query 'Functions[?contains(FunctionName, 'VPNRevokeDeviceVpn') == 'true'].FunctionName'
--output text)
```

- Use the following command if you are using AWS CLI version 2:

```
aws lambda invoke \
  --region $AWS_REGION \
  --function-name $LAMBDA_FUNCTION \
  --cli-binary-format raw-in-base64-out \
  --payload '{"ClientName": "'"$CLIENT_NAME"'"}' /dev/stdout
```

- Use the following command if you are using AWS CLI version 1:

```
aws lambda invoke \
  --region $AWS_REGION \
  --function-name $LAMBDA_FUNCTION \
  --payload '{"ClientName": "'"$CLIENT_NAME"'"}' /dev/stdout
```

After the client configuration and certificate are revoked, you will receive a confirmation message from the Lambda function. Additionally, each certificate that is revoked is entered into the certificate revocation list referenced by all instances. The revoked certificate will no longer be accepted by an instance deployed by this solution.

# Additional resources

**AWS services**

- AWS CloudFormation
- Amazon DynamoDB
- AWS IoT Greengrass
- Amazon Simple Storage Service (Amazon S3)
- AWS IoT Core
- Amazon CloudWatch
- Elastic Load Balancing—Network Load Balancer
- Amazon Elastic File System (Amazon EFS)
- Amazon Virtual Private Cloud (Amazon VPC)
- AWS Lambda
- AWS Command Line Interface
- AWS Global Accelerator
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon EC2 Auto Scaling
- AWS Systems Manager
- AWS Identity and Access Management (IAM)
- Amazon Simple Notification Service (Amazon SNS)

**Other resources**

- OpenVPN package from the EPEL repository
- Raspberry Pi OS
- Ubuntu

# Common OpenVPN operating system configurations

The IoT Static IP Endpoints solution is designed to be used in unattended environments. However, different types of environments can be supported, including Raspberry Pi OS and Ubuntu. The following configurations can be used as the basis for running this solution in different environments. Each configuration demonstrates how to install, configure, and automatically start the OpenVPN client to connect to this solution.

## Raspberry Pi OS

Previously known as Raspbian, Raspberry Pi OS is the recommended operation for use on a Raspberry Pi single-board computer.

1. Install the latest version of Raspberry Pi OS, configure the networking options, and install the required packages.
2. Run the following command to install the OpenVPN package.

```
sudo apt-get install openvpn -y
```

3. Create an OpenVPN configuration file named `/etc/openvpn/<AWS-Endpoint>.conf` (change *<AWS-Endpoint>* to a description of your choice).
4. Modify the configuration file and add the following lines to the end of the file to activate using the solution's DNS server within Amazon VPC.

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

5. Activate and start the configuration.

```
systemctl enable openvpn@AWS_ENDPOINT
systemctl start openvpn@AWS_ENDPOINT
```

6. Test Amazon VPN connectivity and the DNS resolution, then restart your system and verify OpenVPN starts automatically.

## Ubuntu 20.04

Ubuntu is a common Linux distribution used for deployment of AWS IoT Greengrass. The configuration automatically starts the OpenVPN client during system startup and sets the DNS servers to those within Amazon VPC.

1. Install the Ubuntu 20.04.1 package, configure networking, and install the required packages.
2. Run the following command to install the OpenVPN and supporting packages.

```
sudo apt-get install openvpn resolvconf -y
```

3. Create an OpenVPN configuration file named `/etc/openvpn/<AWS-Endpoint>.conf` (change *<AWS-Endpoint>* to a description of your choice).
4. Modify the configuration file and add the following lines to the end of the file to activate using the solution's DNS server within Amazon VPC.

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

5. Activate and start the configuration.

```
systemctl enable openvpn@AWS_ENDPOINT
systemctl start openvpn@AWS_ENDPOINT
```

6. Test Amazon VPN connectivity and the DNS resolution, then restart your system and verify OpenVPN starts automatically.

# Uninstall the solution

You can uninstall the IoT Static Ip Endpoints solution from the AWS Management Console or using the AWS Command Line Interface. However, you must manually delete the Amazon CloudWatch Logs created by this solution.

## Using the AWS Management Console

1. Sign in to the AWS CloudFormation console.
2. Select this solution's installation stack.
3. Choose **Delete**.

## Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

## Deleting resources

During the creation of the stack, you have the option to retain or delete certain resources. However, even when selecting to delete all resources, the processes described above to uninstall the stack may leave behind a few resources in CloudWatch Logs. To complete the removal of these resources, follow these steps:

1. Sign in to the AWS CloudWatch console.
2. Navigate to **Logs**, then **Log groups** and search for Log groups that contain the name of your stack.
3. Select each of the Log groups and then under **Actions** select **Delete log group(s)**.

# Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When activated, the following information is collected and sent to AWS:

- **Solution ID (Solution):** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each IoT Static IP Endpoints deployment
- **TimeStamp:** Data-collection timestamp

**Stack creation or usage data (Create or Usage):** Once at stack creation and then daily, the following deployment parameter values are sent:

- D00_Type: Usage or Create
- D01_Version: Version of solution
- D03_UseNatGateways: **true** or **false**
- D04_UseNatBYOIP: **true** or **false**
- D05_Port: Port number for incoming connections from Internet to NLB
- D06_UseNlbBYOIP: **true** or **false**
- D07_UseGA: **true** or **false**
- D08_UseGABYOIP: **true** or **false**
- D09_Protocol: Incoming protocol, **UDP** or **TCP**
- D10_AutoScalingMinCapacity: Minimum healthy instances
- D11_AutoScalingMaxCapacity: Maximum healthy instances
- D12_InstanceType: EC2 instance family and size
- D13_ActivateFlowLogsToCloudWatch: **true** or **false**

**Daily usage data:** After stack creation, the following operational values are sent daily:

- O03_CPUUtilizationAvg: Percentage of CPU utilization for all instances
- O04_NetworkInAvg: Incoming averages bytes
- O05_NetworkOutAvg: Outgoing average bytes
- O06_ActiveFlowsAvg: Average OpenVPN flows
- O07_NewFlowsAvg: Total OpenVPN flow created
- O08_HealthyHostsAvg: Average healthy instances
- O09_UnhealthyHostsAvg: Average unhealthy instances

Example daily data:

```
{
  "D00_Type": "Usage",
  "D01_Version": "0.1.0",
  "D03_UseNatGateways": false,
  "D04_UseNatBYOIP": false,
  "D05_Port": 1194,
  "D06_UseNlbBYOIP": false,
  "D07_UseGA": false,
  "D08_UseGABYOIP": false,
```

```
  "D09_Protocol": "UDP",
  "D10_AutoScalingMinCapacity": 2,
  "D11_AutoScalingMaxCapacity": 10,
  "D12_InstanceType": "t3.small",
  "D13_ActivateFlowLogsToCloudWatch": true,
  "O03_CPUUtilizationAvg": 2.6,
  "O04_NetworkInAvg": 2544611.625,
  "O05_NetworkOutAvg": 64163.375,
  "O06_ActiveFlowsAvg": 0,
  "O07_NewFlowsAvg": 0,
  "O08_HealthyHostsAvg": 2.192982456140351,
  "O09_UnhealthyHostsAvg": 0.12280701754385964,
  "Solution": "SO0139",
  "TimeStamp": "2020-09-29 19:10:19",
  "UUID": "00008a6e-0689-4915-9fdc-6e952dfbbd3c"
}
```


AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Policy. To opt out of this feature, modify the AWS CloudFormation template mapping section as follows:

```
"SendAnonymousData": {
  "Fn::Equals": [
    {
      "Fn::FindInMap": [
        "Send",
        "AnonymousUsage",
        "Data"
      ]
    },
    "Yes"
  ]
}
```

to

```
"SendAnonymousData": {
  "Fn::Equals": [
    {
      "Fn::FindInMap": [
        "Send",
        "AnonymousUsage",
        "Data"
      ]
    },
    "No"
  ]
}
```

# Source code

Visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

# Contributors

The following individuals contributed to this document:

- Gavin Adams
- Matt MacLean

# Revisions

| Date | Change |
| --- | --- |
| February 2021 | Initial release |

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

IoT Static IP Endpoints is licensed under the terms of the of the Apache License Version 2.0 available at The Apache Software Foundation.