

[AWS Compute Blog](#)

Secure Connectivity from Public to Private: Introducing EC2 Instance Connect Endpoint

by [Sheila Busser](#) | on 14 JUN 2023 | in [Amazon EC2](#), [Amazon VPC](#), [Announcements](#), [Compute](#), [Customer Solutions](#), [Security, Identity, & Compliance](#), [Technical How-To](#) | [Permalink](#) | [Share](#)

This blog post is written by Ariana Rahgozar, Solutions Architect, and Kenneth Kitts, Sr. Technical Account Manager, AWS.

Imagine trying to connect to an [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) instance within your [Amazon Virtual Private Cloud \(Amazon VPC\)](#) over the Internet. Typically, you'd first have to connect to a bastion host with a public IP address that your administrator set up over an [Internet Gateway \(IGW\)](#) in your VPC, and then use port forwarding to reach your destination.

Today we launched [Amazon EC2 Instance Connect \(EIC\) Endpoint](#), a new feature that allows you to connect securely to your instances from the Internet. With EIC Endpoint, you no longer need an IGW in your VPC, a public IP address on your resource, a bastion host, or any agent to connect to your resources. EIC Endpoint combines identity-based and network-based access controls, providing the isolation, control, and logging needed to meet your organization's security requirements. As a bonus, your organization administrator is also relieved of the operational overhead of maintaining and patching bastion hosts for connectivity. EIC Endpoint works with the [AWS Management Console](#) and [AWS Command Line Interface \(AWS CLI\)](#). Furthermore, it gives you the flexibility to continue using your favorite tools, such as PuTTY and OpenSSH.

In this post, we provide an overview of how the EIC Endpoint works and its security controls, guide you through your first EIC Endpoint creation, and demonstrate how to SSH to an instance from the Internet over the EIC Endpoint.

EIC Endpoint product overview

EIC Endpoint is an identity-aware TCP proxy. It has two modes: first, AWS CLI client is used to create a secure, WebSocket tunnel from your workstation to the endpoint with your [AWS Identity and Access Management \(IAM\)](#) credentials. Once you've established a tunnel, you point your preferred client at your loopback address (`127.0.0.1` or `localhost`) and connect as usual. Second, when not using the AWS CLI, the Console gives you secure and seamless access to resources inside your VPC. Authentication and authorization is evaluated before traffic reaches the VPC. The following figure shows an illustration of a user connecting via an EIC Endpoint:

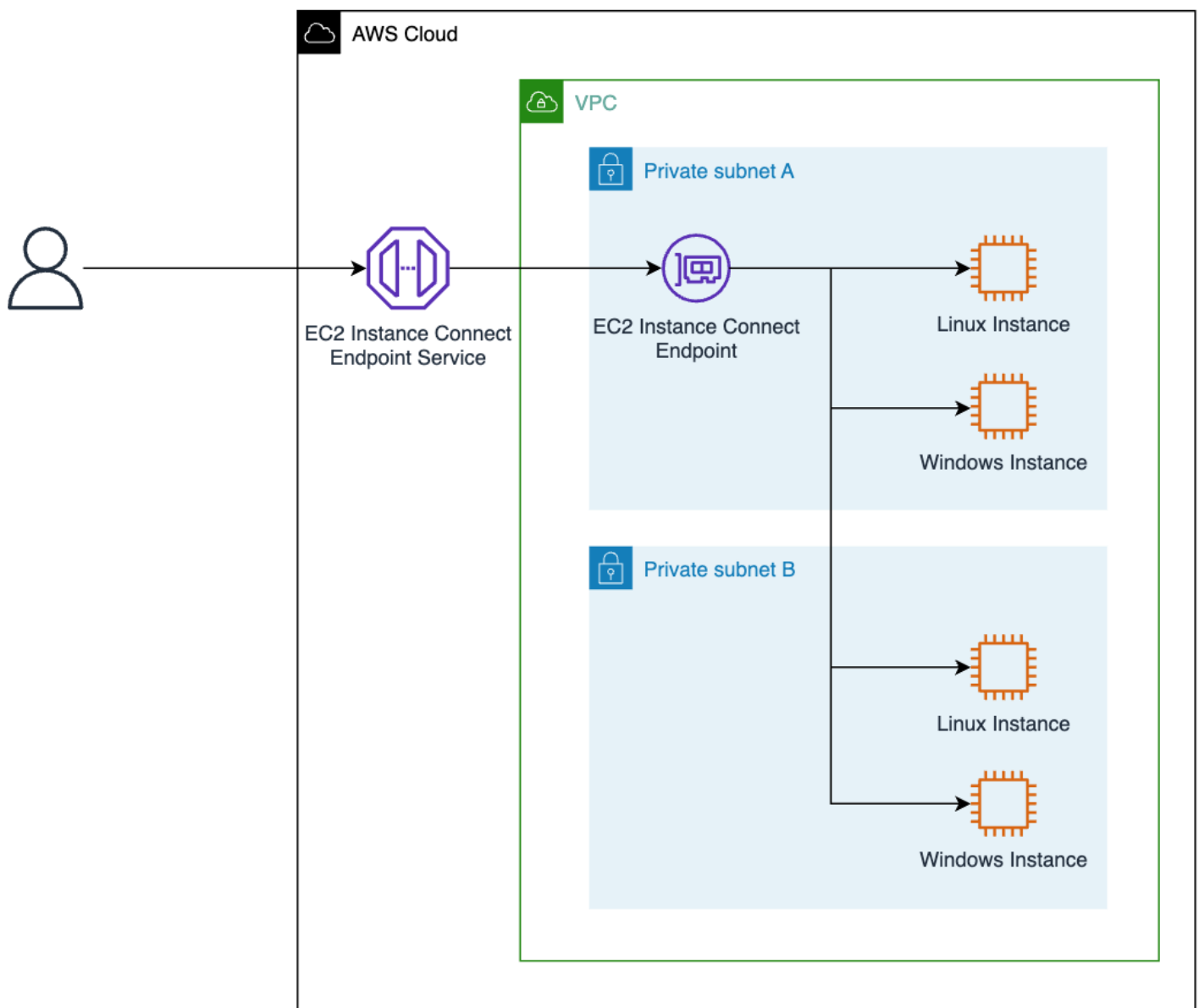


Figure 1. User connecting to private EC2 instances through an EIC Endpoint

EIC Endpoints provide a high degree of flexibility. First, they don't require your VPC to have direct Internet connectivity using an IGW or NAT Gateway. Second, no agent is needed on the resource you wish to connect to, allowing for easy remote administration of resources which may not support agents, like third-party appliances. Third, they preserve existing workflows, enabling you to continue using your preferred client software on your local workstation to connect and manage your resources. And finally, IAM and [Security Groups](#) can be used to control access, which we discuss in more detail in the next section.

Prior to the launch of EIC Endpoints, AWS offered two key services to help manage access from public address space into a VPC more carefully. First is [EC2 Instance Connect](#), which provides a mechanism that uses IAM credentials to push ephemeral SSH keys to an instance, making long-lived keys unnecessary. However, until now EC2 Instance Connect required a public IP address on your instance when connecting over the Internet.

With this launch, you can use EC2 Instance Connect with EIC Endpoints, combining the two capabilities to give you ephemeral-key-based SSH to your instances without exposure to the public Internet. As an alternative to EC2 Instance Connect and EIC Endpoint based connectivity, AWS also offers [Systems Manager Session Manager \(SSM\)](#), which provides agent-based connectivity to instances. SSM uses IAM for authentication and authorization, and is ideal for environments where an agent can be configured to run.

Given that EIC Endpoint enables access to private resources from public IP space, let's review the security controls and capabilities in more detail before discussing creating your first EIC Endpoint.

Security capabilities and controls

Many AWS customers remotely managing resources inside their VPCs from the Internet still use either public IP addresses on the relevant resources, or at best a bastion host approach combined with long-lived SSH keys. Using public IPs can be locked down somewhat using IGW routes and/or security groups. However, in a dynamic environment those controls can be hard to manage. As a result, careful management of long-lived SSH keys remains the only layer of defense, which isn't great since we all know that these controls sometimes fail, and so defense-in-depth is important. Although bastion hosts can help, they increase the operational overhead of managing, patching, and maintaining infrastructure significantly.

IAM authorization is required to create the EIC Endpoint and also to establish a connection via the endpoint's secure tunneling technology. Along with identity-based access controls governing who, how, when, and how long users can connect, more traditional network access controls like security groups can also be used. Security groups associated with your VPC resources can be used to grant/deny access. Whether it's IAM policies or security groups, the default behavior is to deny traffic unless it is explicitly allowed.

EIC Endpoint meets important security requirements in terms of separation of privileges for the control plane and data plane. An administrator with full EC2 IAM privileges can create and control EIC Endpoints (the control plane). However, they cannot use those endpoints without also having EC2 Instance Connect IAM privileges (the data plane). Conversely, DevOps engineers who may need to use EIC Endpoint to tunnel into VPC resources do not require control-plane privileges to do so. In all cases, IAM principals using an EIC Endpoint must be part of the same AWS account (either directly or by cross-account role assumption). Security administrators and auditors have a centralized view of endpoint activity as all API calls for configuring and connecting via the EIC Endpoint API are recorded in [AWS CloudTrail](#). Records of data-plane connections include the IAM principal making the request, their source IP address, the requested destination IP address, and the destination port. See the following figure for an example CloudTrail entry.

```
"userIdentity": {
  "type": "IAMUser",
  "principalId": "AROAI DPPEZS35WEXAMPLE",
  "arn": "arn:aws:iam::123456789012:user/Alice",
  "accountId": "123456789012",
  "accessKeyId": "ACCESSKEYEXAMPLE",
  "userName": "Alice"}
"eventTime": "2023-04-06T23:11:53Z",
"eventSource": "ec2-instance-connect.amazonaws.com",
"eventName": "OpenTunnel",
"awsRegion": "us-east-1",
"sourceIPAddress": "xx.xx.xx.xx",
"requestParameters": {
  "instanceConnectEndpointId": "eice-exampleendpoint",
  "maxTunnelDuration": "3600",
  "remotePort": "22",
  "privateIpAddress": "10.10.10.10"}
```

Figure 2. Partial CloudTrail entry for an SSH data-plane connection

EIC Endpoint supports the optional use of Client IP Preservation (a.k.a Source IP Preservation), which is an important security consideration for certain organizations. For example, suppose the resource you are connecting to has network access controls that are scoped to your specific public IP address, or your instance access logs must contain the client's "true" IP address. Although you may choose to enable this feature when you create an endpoint, the default setting is *off*. When off, connections proxied through the endpoint use the endpoint's private IP address in the network packets' source IP field. This default behavior allows connections proxied through the endpoint to reach as far as your route tables permit. Remember, no matter how you configure this setting, CloudTrail records the client's true IP address.

EIC Endpoints strengthen security by combining identity-based authentication and authorization with traditional network-perimeter controls and provides for fine-grained access control, logging, monitoring, and more defense in depth. Moreover, it does all this without requiring Internet-enabling infrastructure in your VPC, minimizing the possibility of unintended access to private VPC resources.

Getting started

Creating your EIC Endpoint

Only one endpoint is required per VPC. To create or modify an endpoint and connect to a resource, a user must have the required IAM permissions, and any security groups associated with your VPC resources must have a rule to allow connectivity. Refer to the following resources for more details on [configuring security groups](#) and sample [IAM permissions](#).

The AWS CLI or Console can be used to create an EIC Endpoint, and we demonstrate the AWS CLI in the following. To create an EIC Endpoint using the Console, refer to the [documentation](#).

Creating an EIC Endpoint with the AWS CLI

To create an EIC Endpoint with the AWS CLI, run the following command, replacing **[SUBNET]** with your subnet ID and **[SG-ID]** with your security group ID:

```
aws ec2 create-instance-connect-endpoint \
  --subnet-id [SUBNET] \
  --security-group-id [SG-ID]
```

After creating an EIC Endpoint using the AWS CLI or Console, and granting the user IAM permission to create a tunnel, a connection can be established. Now we discuss how to connect to Linux instances using SSH. However, note that you can also use the OpenTunnel API to connect to instances via [RDP](#).

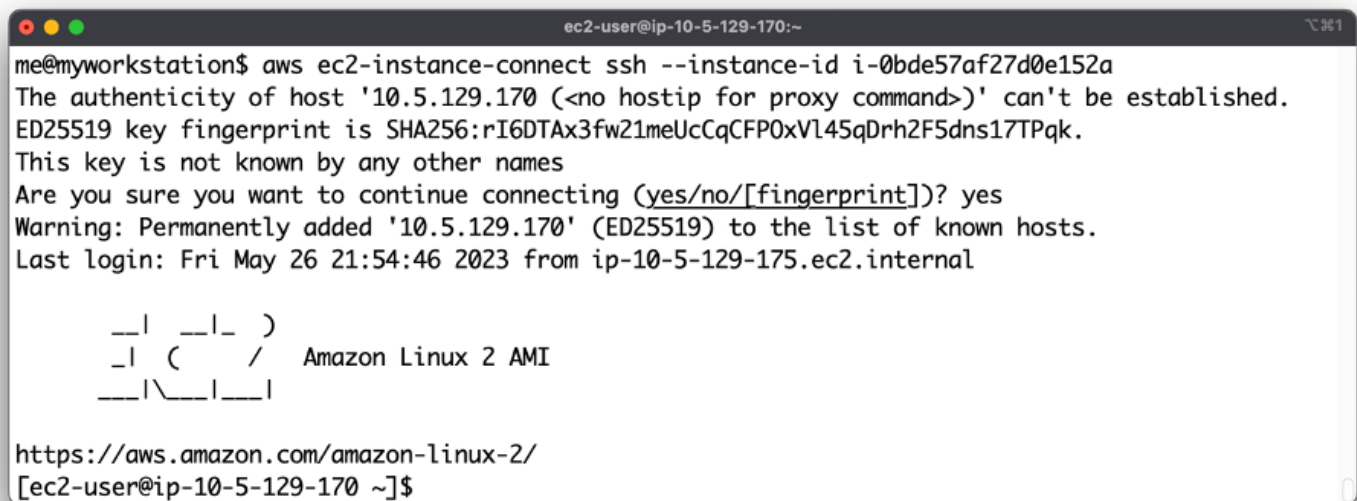
Connecting to your Linux Instance using SSH

With your EIC Endpoint set up in your VPC subnet, you can connect using SSH. Traditionally, access to an EC2 instance using SSH was controlled by key pairs and network access controls. With EIC Endpoint, an additional layer of control is enabled through IAM policy, leading to an enhanced security posture for remote access. We describe two methods to connect via SSH in the following.

One-click command

To further reduce the operational burden of creating and rotating SSH keys, you can use the new `ec2-instance-connect ssh` command from the AWS CLI. With this [new command](#), we generate ephemeral keys for you to connect to your instance. Note that this command requires use of the OpenSSH client and the latest version of the AWS CLI. To use this command and connect, you need IAM permissions as detailed [here](#).

Once configured, you can connect using the new AWS CLI command, shown in the following figure:



```
ec2-user@ip-10-5-129-170:~$ aws ec2-instance-connect ssh --instance-id i-0bde57af27d0e152a
The authenticity of host '10.5.129.170 (<no hostip for proxy command>)' can't be established.
ED25519 key fingerprint is SHA256:rI6DTAx3fw21meUcCqCFP0xVL45qDrh2F5dns17TPqk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.5.129.170' (ED25519) to the list of known hosts.
Last login: Fri May 26 21:54:46 2023 from ip-10-5-129-175.ec2.internal

  _ | _ | _ )
  _ | (   /   Amazon Linux 2 AMI
  _ | \_ | _ |

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-5-129-170 ~]$
```

Figure 3. AWS CLI view upon successful SSH connection to your instance

To test connecting to your instance from the AWS CLI, you can run the following command where **[INSTANCE]** is the instance ID of your EC2 instance:

```
aws ec2-instance-connect ssh --instance-id [INSTANCE]
```

Note that you can still use long-lived SSH credentials to connect if you must maintain existing workflows, which we will show in the following. However, note that dynamic, frequently rotated credentials are generally safer.

Open-tunnel command

You can also connect using SSH with standard tooling or using the proxy command. To establish a private tunnel (TCP proxy) to the instance, you must run one AWS CLI command, which you can see in the following figure:

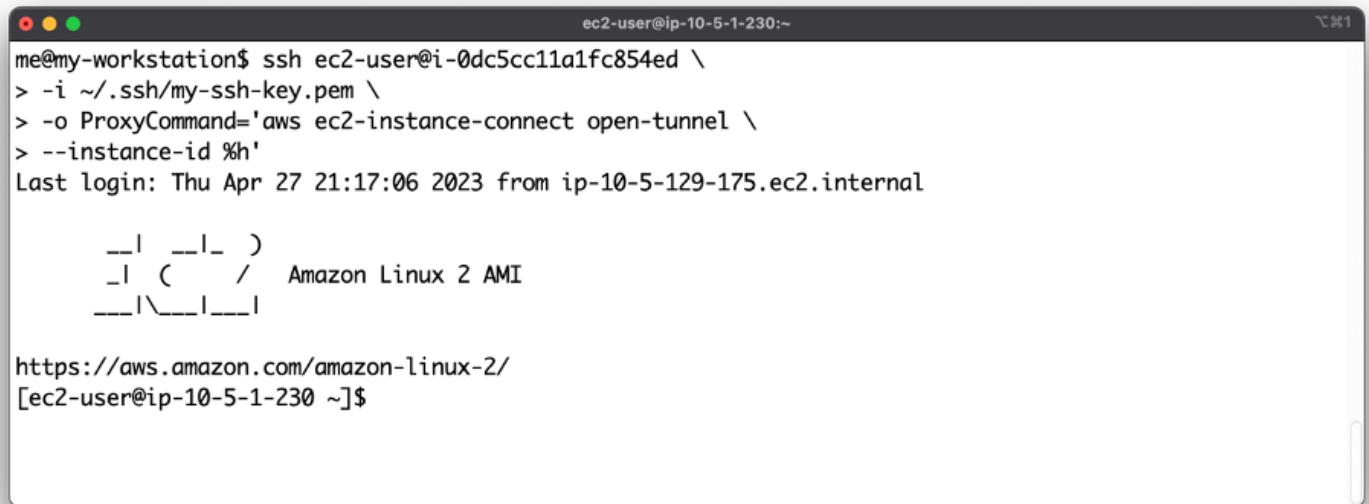
A terminal window titled 'ec2-user@ip-10-5-1-230:~' showing the execution of an SSH command. The command is 'ssh ec2-user@i-0dc5cc11a1fc854ed \ -i ~/.ssh/my-ssh-key.pem \ -o ProxyCommand='aws ec2-instance-connect open-tunnel \ --instance-id %h''. The terminal output shows the last login time as 'Thu Apr 27 21:17:06 2023 from ip-10-5-129-175.ec2.internal', followed by the Amazon Linux 2 AMI logo and the URL 'https://aws.amazon.com/amazon-linux-2/'. The prompt changes to '[ec2-user@ip-10-5-1-230 ~]\$'.

Figure 4. AWS CLI view after running new SSH open-tunnel command, creating a private tunnel to connect to our EC2 instance

You can run the following command to test connectivity, where **[INSTANCE]** is the instance ID of your EC2 instance and **[SSH-KEY]** is the location and name of your SSH key. For guidance on the use of SSH keys, refer to our documentation on [Amazon EC2 key pairs and Linux instances](#).

```
ssh ec2-user@[INSTANCE] \  
-i [SSH-KEY] \  
-o ProxyCommand='aws ec2-instance-connect open-tunnel \  
--instance-id %h'
```

Once we have our EIC Endpoint configured, we can SSH into our EC2 instances without a public IP or IGW using the AWS CLI.

Conclusion

EIC Endpoint provides a secure solution to connect to your instances via SSH or RDP in private subnets without IGWs, public IPs, agents, and bastion hosts. By configuring an EIC Endpoint for your VPC, you can securely connect using your existing client tools or the Console/AWS CLI. To learn more, visit the [EIC Endpoint documentation](#).

TAGS: [EC2 instances](#)