When developing projects on Compute Engine, there are a variety of scenarios in which you want to keep the instances from being reached from the public internet:

- Web services are still under development and not ready to be exposed to external users because they are feature incomplete or have not yet been configured with HTTPS.
- Instance might be providing services designed to be consumed only by other instances in the project.
- Instances should only be reached through dedicated interconnect options from company offices or data centers.

Even when a service is intentionally internet-facing, it is important that communication with the service be restricted to the target user groups, and occur over secure channels, such as SSH or HTTPS, to protect sensitive information.

This article demonstrates several methods for securing communications with Compute Engine instances with or without external IP addresses.

- Firewalls (#firewalls)
- HTTPS and SSL (#https-and-ssl)
- Port forwarding over SSH (#port-forwarding-over-ssh)
- SOCKS proxy over SSH (#socks-proxy-over-ssh)

- Bastion hosts and SSH forwarding (#bastion)
- IAP for TCP forwarding (#cloud_iap)
- VPN (#vpn)
- NAT gateway for egress (#natgateway)
- Interactive serial console access (#interactiveserialconsole)
- HTTPS and SSL proxy load balancers (#https-ssl-lb)

When instances have a public IP address, it is important that only the services and traffic you intend to be exposed are reachable, and for those that are exposed, any sensitive information is secured in transit.

Your first line of defense is to restrict who can reach the instance using firewalls (/compute/docs/vpc/firewalls). By creating firewall rules, you can restrict all traffic to a network or target machines on a given set of ports to specific source IP addresses.

Firewalls are not a standalone solution. Restricting traffic to specific source IPs does not protect sensitive information, such as login credentials, commands that create or destroy resources or files, or logs. When running a web service on a publicly-accessible machine, such as a Compute Engine instance with an external IP, you must encrypt all communication between your host and the deployed instance to ensure proper security.

In addition, firewalls aren't always the appropriate solution. For example, firewalls are not ideal for development environments that do not have static IP addresses, such as roaming laptops.

For production web systems, you should configure HTTPS/SSL. HTTPS/SSL can be set up either by setting up an instance to terminate HTTPS or by configuring HTTPS load balancing. HTTPS/SSL does involve some initial complexity, requiring you to perform the following tasks:

- Register a domain name.
- Acquire an SSL certificate from a certificate authority.
- Register the certificate with your HTTPS load balancer (/load-balancing/docs/ssl-certificates) and its connected instances, or configure an SSL-terminated web server or proxy on one or more Compute Engine instances.

If you have set up SSL-serving domains before, it should be straightforward to do the same with Compute Engine. If not, you might find it easier to use a different security method, such as port forwarding (#port-forwarding-over-ssh) or SOCKS proxy (#socks-proxy-over-ssh).

You can use the `gcloud` command-line tool (/sdk/docs/) to start a server on a given local port that forwards all traffic to a remote host over an SSH connection.

First, take note of the instance and port that are providing the service to which you would like to establish a secure connection. Next, run the following command:

In the preceding command, the parameters are defined as follows:

- `example-instance` is the name of the instance to which you'd like to connect.
- `my-project` is your Google Cloud project ID (/resource-manager/docs/creating-managing-projects#identifying_projects).
- `us-central1-a` is the zone in which your instance is running.
- `2222` is the local port you're listening on.
- `8888` is the remote port you're connecting to.

With these example settings, if you open http://localhost:2222/ (http://localhost:2222/) in your browser, the HTTP connection uses the SSH tunnel that you created to your remote host to connect to the specified instance using SSH. The HTTP connection will then use the SSH tunnel to connect to port `8888` on the same machine, but over an encrypted, secure SSH connection.

The `gcloud` command creates and maintains an SSH connection, and this approach only works while the SSH session is active. As soon as you exit, the SSH session that `gcloud` creates, port forwarding using http://localhost:2222/ stops working.

To create more than one port forwarding rule, you can specify multiple rules on a single command line by repeating the flags:

Alternatively, you can run a new `gcloud` command each time to create a separate tunnel. Note that you cannot add or remove port forwarding from an existing connection without exiting and re-establishing the connection from scratch.

If you want to connect to a number of different hosts in your cloud deployment, the easiest way to do so is to change your browser to do the lookups directly from your network (/compute/docs/vpc). This approach lets you use the short name of the hosts instead of looking up each host's IP address, opening up ports for each service, or creating an SSH tunnel for each host/port pair.

The approach that you use here is as follows:

1. Set up a single SSH tunnel to one of the hosts on the network, and create a SOCKS proxy on that host.
2. Change the browser configuration to do all the lookups using that SOCKS proxy host.

Note that because you are tunneling *all* traffic using that host, avoid using that browser or that specific profile to browse the web because you need to dedicate that bandwidth to your cloud service. In general, you might want to use a separate browser profile and switch to it when necessary.

To start your SOCKS proxy, run the following command:

Replace the following:

- ***`example-instance`***: The name of the instance to which you would like to connect.
- ***`my-project`***: Your Google Cloud project ID (/storage/docs/projects#projectid-using).
- ***`zone`***: The zone in which your instance is running, for example, `us-central1-a`.
- ***`NNNN`***: The local port you're listening on, for example, `1080`.

Note that, in this case, you don't need to specify a remote port. Because a SOCKS proxy does not bind to any specific remote port, any connection you make using the SOCKS proxy will be resolved relative to the host you connect to.

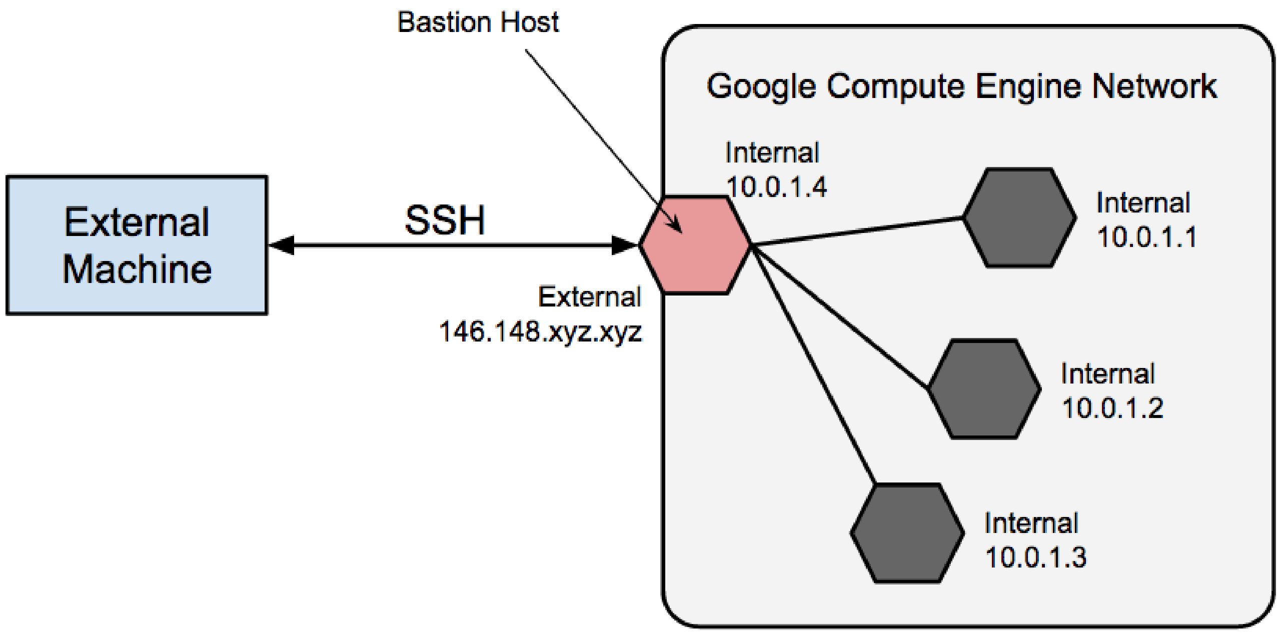
By using a SOCKS proxy, you can connect to any instance that shares a Compute Engine network with your proxy instance by using the instance's short name. In addition, you can connect to any port on a given instance.

This approach is much more flexible than the simple port-forwarding method, but will also require you to change the settings in your web browser to utilize the proxy.

Next, configure either Chrome or Firefox to use the proxy.

When instances do not have external IP addresses they can only be reached by other instances on the network, Identity-Aware Proxy's TCP forwarding feature, or by using managed VPN gateway. You can provision instances in your network to act as trusted relays for inbound connections (bastion hosts) or network egress (NAT Gateways). For more transparent connectivity without setting up such connections, you can use a managed VPN gateway resource.

Bastion hosts (https://wikipedia.org/wiki/Bastion_host) provide an external facing point of entry into a network containing private network instances, as illustrated in the following diagram.



This host can provide a single point of fortification or audit and can be started and stopped to enable or disable inbound SSH. By using a bastion host, you can connect to an instance that does not have an external IP address. This approach allows you to connect to a development environment or manage the database instance for your external application, for example, without configuring additional firewall rules.

A complete hardening of a bastion host is outside the scope of this article, but some initial steps taken can include:

- Limit the CIDR range of source IPs that can communicate with the bastion.
- Configure firewall rules to allow SSH traffic to private instances from only the bastion host.

By default, SSH on instances is configured to use private keys for authentication. When using a bastion host, you log into the bastion host first, and then into your target private instance. Because of this two-step login, which is why bastion hosts are sometimes called "jump servers," you should use `ssh` forwarding instead of storing the target machine's private key on the bastion host as a way of reaching the target machine. You need to do this even if using the same key pair for both bastion and target instances because the bastion has direct access to only the public half of the key pair.

To learn how to use a bastion host instance to connect to other instances on your Google Cloud network, see Connecting through a bastion host (/compute/docs/instances/connecting-advanced#bastion_host).

To learn how to use `ssh` forwarding and other methods to connect to instances that do not have an external IP address, see see Connecting to instances that do not have external IP addresses (/compute/docs/instances/connecting-advanced#sshbetweeninstances).

Using SSH with IAP's TCP forwarding feature wraps an SSH connection inside HTTPS. IAP's TCP forwarding feature then sends it to the remote instance.

To learn how to connect to a remote instance with IAP, see Using IAP for TCP forwarding (/iap/docs/using-tcp-forwarding#tunneling_with_ssh).

Cloud VPN lets you connect your existing network to your Google Cloud network by using an IPsec connection to a VPN gateway device. This allows direct routing of traffic from your premises to the private IP interfaces of Compute Engine instances. Traffic is encrypted as it transits over public links to Google.

For details on setting up, configuring, and using VPN with Compute Engine, see the Cloud VPN documentation (/vpn/docs/).

To learn how to connect to instances on your Google Cloud network through an existing VPN rather than through external IP addresses of instances, read Connecting to instances that do not have external IP addresses (/compute/docs/instances/connecting-advanced#sshbetweeninstances).

When an instance does not have an external IP address assigned it cannot make direct connections to external services, including other Google Cloud services. To allow these instances to reach services on the public internet, you can set up and configure a NAT gateway (/compute/docs/vpc/special-configurations#natgateway) machine, which can route traffic on behalf of any instance on the network. Do not consider a single instance to be highly available or able to support high traffic throughput for multiple instances.

When an instance doesn't have an external IP address, you might still need to interact with the instance for troubleshooting or maintenance purposes. Setting up a Bastion host (#bastion), as discussed earlier, is one option but might require more setup than worthwhile for your needs. If you want to troubleshoot an instance without an external IP address, consider enabling interactive access on the serial console (/compute/docs/instances/interacting-with-serial-console), which allows you to interact with an instance's serial console using SSH and run commands against the serial console.

To learn more, read Interacting with the Serial Console (/compute/docs/instances/interacting-with-serial-console).

Instances that are backends for HTTPS and SSL proxy load balancers do not have to have external IP addresses to be accessed through the load balancer. To access these resources directly requires the use of methods listed in the section Connecting to instances without external IP addresses (#external).

To learn more, read the load balancing documentation (/load-balancing/docs) for those load balancers.

Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).