**You said:**

import os import google.auth from googleapiclient.discovery import build from googleapiclient.errors import HttpError import time # Set your project ID and zone here PROJECT\_ID = 'manifest-zephyr-457115-v2' # Updated project ID ZONE = 'us-central1-a' # Or another zone you prefer INSTANCE\_NAME = 'private-ubuntu-vm' IMAGE\_PROJECT = 'ubuntu-os-cloud' IMAGE\_FAMILY = 'ubuntu-2004-lts' # Ubuntu 20.04 LTS image # Authenticate and create the compute client def create\_instance(): # Initialize the Compute Engine client credentials, project = google.auth.default() compute = build('compute', 'v1', credentials=credentials) # Define the machine type and image machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1" # You can change the machine type here source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}" # Define the configuration for the VM instance config = { 'name': INSTANCE\_NAME, 'machineType': machine\_type, 'disks': [{ 'boot': True, 'autoDelete': True, 'initializeParams': { 'sourceImage': source\_disk\_image } }], 'networkInterfaces': [{ 'network': 'global/networks/default', # No accessConfigs means no public IP address }], 'tags': { 'items': ['http-server', 'https-server'] }, } try: # Create the VM instance print(f"Creating private VM instance {INSTANCE\_NAME}...") operation = compute.instances().insert( project=PROJECT\_ID, zone=ZONE, body=config ).execute() # Wait for the operation to complete wait\_for\_operation(compute, operation) print(f"Private VM instance {INSTANCE\_NAME} created successfully.") except HttpError as err: print(f"An error occurred: {err}") return def wait\_for\_operation(compute, operation): print("Waiting for operation to complete...") while True: result = compute.zoneOperations().get( project=PROJECT\_ID, zone=ZONE, operation=operation['name'] ).execute() if result['status'] == 'DONE': if 'error' in result: raise Exception(f"Error: {result['error']}") else: print("Operation completed successfully.") break else: print("Operation is still in progress...") time.sleep(10) if \_\_name\_\_ == "\_\_main\_\_": create\_instance() - in this script install nginx on vm instance access it using private ip

**ChatGPT said:**

To **install NGINX on the VM instance** and **access it via its private IP**, you’ll need to do the following:

1. **Add startup script** to install NGINX on the VM.
2. **Allow internal network communication**, so your client can reach the VM using its **private IP**.
3. (Optional) **Enable firewall rules** for HTTP if you're accessing from within the same VPC.

Here’s the **modified script** with a startup script that installs NGINX:

python

CopyEdit

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Set your project ID and zone here

PROJECT\_ID = 'manifest-zephyr-457115-v2'

ZONE = 'us-central1-a'

INSTANCE\_NAME = 'private-ubuntu-vm'

IMAGE\_PROJECT = 'ubuntu-os-cloud'

IMAGE\_FAMILY = 'ubuntu-2004-lts'

def create\_instance():

credentials, project = google.auth.default()

compute = build('compute', 'v1', credentials=credentials)

machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"

source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

# Startup script to install nginx

startup\_script = """#!/bin/bash

sudo apt-get update

sudo apt-get install -y nginx

sudo systemctl enable nginx

sudo systemctl start nginx

"""

config = {

'name': INSTANCE\_NAME,

'machineType': machine\_type,

'disks': [{

'boot': True,

'autoDelete': True,

'initializeParams': {

'sourceImage': source\_disk\_image

}

}],

'networkInterfaces': [{

'network': 'global/networks/default',

# No external IP (private only)

}],

'tags': {

'items': ['http-server']

},

'metadata': {

'items': [{

'key': 'startup-script',

'value': startup\_script

}]

}

}

try:

print(f"Creating private VM instance {INSTANCE\_NAME}...")

operation = compute.instances().insert(

project=PROJECT\_ID,

zone=ZONE,

body=config

).execute()

wait\_for\_operation(compute, operation)

print(f"Private VM instance {INSTANCE\_NAME} created and NGINX installed successfully.")

except HttpError as err:

print(f"An error occurred: {err}")

return

def wait\_for\_operation(compute, operation):

print("Waiting for operation to complete...")

while True:

result = compute.zoneOperations().get(

project=PROJECT\_ID,

zone=ZONE,

operation=operation['name']

).execute()

if result['status'] == 'DONE':

if 'error' in result:

raise Exception(f"Error: {result['error']}")

else:

print("Operation completed successfully.")

break

else:

print("Operation is still in progress...")

time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

create\_instance()

**✅ To test the setup:**

1. **Create another VM** in the same VPC with a public IP (e.g., a bastion host or jumpbox).
2. SSH into that public VM.
3. From the SSH session, run:

bash

CopyEdit

curl http://<PRIVATE-IP-OF-YOUR-NGINX-VM>

You should see the NGINX default page HTML.

**🔒 Note:**

If you're accessing via internal IP:

* Ensure **"allow all internal traffic"** is enabled in the default VPC firewall rules.
* No external IP is attached, so no direct access from the internet.

Let me know if you’d like to automate the bastion host setup or SSH access from Cloud Shell as well!

**You said:**

user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $ curl http://10.128.0.9 curl: (28) Failed to connect to 10.128.0.9 port 80 after 21021 ms: Could not connect to server user121@DESKTOP-JFJ21I5 MINGW64 /d/agcp $

**ChatGPT said:**

Thanks for sharing the curl error — the key line here is:

vbnet

CopyEdit

curl: (28) Failed to connect to 10.128.0.9 port 80 ... Could not connect to server

This means the NGINX service on the private VM is **not reachable** from your local machine. Let's troubleshoot step by step.

**🔍 Why This Happened**

Your VM **doesn't have a public IP**, so:

* You're trying to curl from your **local machine**, which is **outside Google Cloud’s private VPC**.
* That **private IP (10.128.0.9) is only accessible within the same VPC or peered VPC**.

**✅ How to Fix or Test It Properly**

You have **2 main options**:

**✅ Option 1: Use a Bastion Host (Recommended for Private VMs)**

1. **Create a second VM** in the same VPC **with a public IP** (call it bastion-host).
2. SSH into the bastion host from your local machine:

bash

CopyEdit

gcloud compute ssh bastion-host --zone=us-central1-a

1. From inside that VM, test the private IP:

bash

CopyEdit

curl http://10.128.0.9

You should get the NGINX welcome page.

**✅ Option 2: Temporarily Add a Public IP to the Private VM (for quick testing only)**

If you just want to test it quickly, you can:

1. Go to GCP Console → Compute Engine → Your Instance.
2. Click **Edit** → under **Network interfaces**, click default.
3. Set **External IP** to “Ephemeral”.
4. Save.
5. Then from your machine:

bash

CopyEdit

curl http://<EXTERNAL\_IP>

1. Afterwards, remove the public IP if you want to make it private again.

**✅ Optional: Confirm NGINX is Running**

If you're able to SSH into the private VM (via bastion or Cloud Shell + internal IP), run:

bash

CopyEdit

sudo systemctl status nginx

You should see something like:

pgsql

CopyEdit

● nginx.service - A high performance web server and a reverse proxy server

Active: active (running)

If it’s not running, you can restart it:

bash

CopyEdit

sudo systemctl restart nginx

**Summary**

| **Goal** | **What to Do** |
| --- | --- |
| Access VM via private IP | Use a **bastion host** or **Cloud Shell** |
| Just test NGINX | Temporarily assign a **public IP** |
| Ensure NGINX runs | SSH → sudo systemctl status nginx |

Let me know which method you want to try, and I’ll help set it up.

**You said:**

create a public VM using the above configuration and install nginx on it and access ngin using public ip only

**ChatGPT said:**

Perfect — let's update your Python script to:

✅ Create a **public VM**  
✅ Install **NGINX** using a **startup script**  
✅ Allow HTTP access (by setting firewall tags)  
✅ Output the **public IP address** so you can test it via browser or curl

**✅ Updated Script (Create Public VM with NGINX)**

python

CopyEdit

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Configuration

PROJECT\_ID = 'manifest-zephyr-457115-v2'

ZONE = 'us-central1-a'

INSTANCE\_NAME = 'public-ubuntu-vm'

IMAGE\_PROJECT = 'ubuntu-os-cloud'

IMAGE\_FAMILY = 'ubuntu-2004-lts'

def create\_instance():

credentials, project = google.auth.default()

compute = build('compute', 'v1', credentials=credentials)

machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"

source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

# Startup script to install NGINX

startup\_script = """#!/bin/bash

sudo apt-get update

sudo apt-get install -y nginx

sudo systemctl enable nginx

sudo systemctl start nginx

"""

config = {

'name': INSTANCE\_NAME,

'machineType': machine\_type,

'disks': [{

'boot': True,

'autoDelete': True,

'initializeParams': {

'sourceImage': source\_disk\_image

}

}],

'networkInterfaces': [{

'network': 'global/networks/default',

'accessConfigs': [{

'type': 'ONE\_TO\_ONE\_NAT',

'name': 'External NAT'

}]

}],

'tags': {

'items': ['http-server']

},

'metadata': {

'items': [{

'key': 'startup-script',

'value': startup\_script

}]

}

}

try:

print(f"Creating public VM instance {INSTANCE\_NAME}...")

operation = compute.instances().insert(

project=PROJECT\_ID,

zone=ZONE,

body=config

).execute()

wait\_for\_operation(compute, operation)

print(f"Public VM instance {INSTANCE\_NAME} created and NGINX installed successfully.")

# Get instance info to print public IP

result = compute.instances().get(project=PROJECT\_ID, zone=ZONE, instance=INSTANCE\_NAME).execute()

public\_ip = result['networkInterfaces'][0]['accessConfigs'][0]['natIP']

print(f"Access NGINX at: http://{public\_ip}")

except HttpError as err:

print(f"An error occurred: {err}")

return

def wait\_for\_operation(compute, operation):

print("Waiting for operation to complete...")

while True:

result = compute.zoneOperations().get(

project=PROJECT\_ID,

zone=ZONE,

operation=operation['name']

).execute()

if result['status'] == 'DONE':

if 'error' in result:

raise Exception(f"Error: {result['error']}")

else:

print("Operation completed successfully.")

break

else:

print("Operation is still in progress...")

time.sleep(10)

if \_\_name\_\_ == "\_\_main\_\_":

create\_instance()

**🔥 After Running**

You’ll see a line like this in the output:

nginx

CopyEdit

Access NGINX at: http://34.123.45.67

Just copy that IP into your browser or run:

bash

CopyEdit

curl http://<PUBLIC-IP>

You’ll see the default NGINX welcome page.

**📌 Important Note**

Ensure this firewall rule exists (usually does by default):

bash

CopyEdit

gcloud compute firewall-rules create default-allow-http \

--allow tcp:80 \

--target-tags http-server \

--description "Allow HTTP traffic" \

--direction INGRESS

Want me to add that firewall creation directly into the script too?

**You said:**

import os import google.auth from googleapiclient.discovery import build from googleapiclient.errors import HttpError import time # Set your project ID, VPC, subnet, and zone here PROJECT\_ID = 'manifest-zephyr-457115-v2' # Your Google Cloud project ID ZONE = 'us-central1-a' # The zone for the VM (make sure it's within 'us-central1') INSTANCE\_NAME = 'private-ubuntu-vm-2' # New name for the VM to avoid conflict IMAGE\_PROJECT = 'ubuntu-os-cloud' # The project where the Ubuntu image resides IMAGE\_FAMILY = 'ubuntu-2004-lts' # The image family (Ubuntu 20.04 LTS) VPC\_NAME = 'hu-devops-25' # Your VPC network name SUBNET\_NAME = 'private-subnet' # Your private subnet name # Authenticate and create the compute client def create\_instance(): # Initialize the Compute Engine client credentials, project = google.auth.default() compute = build('compute', 'v1', credentials=credentials) # Define the machine type and image machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1" # You can change the machine type here source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}" # Define the network and subnet configuration network\_interface = { 'network': f'global/networks/{VPC\_NAME}', # Specify the VPC network 'subnetwork': f'regions/us-central1/subnetworks/{SUBNET\_NAME}', # Specify the correct region 'us-central1' and subnet # No accessConfigs means no public IP address (this creates a private VM) } # Define the configuration for the VM instance config = { 'name': INSTANCE\_NAME, 'machineType': machine\_type, 'disks': [{ 'boot': True, 'autoDelete': True, 'initializeParams': { 'sourceImage': source\_disk\_image } }], 'networkInterfaces': [network\_interface], # Attach the VM to the private network 'tags': { 'items': ['http-server', 'https-server'] # Tags for access control (e.g., firewall rules) }, } try: # Create the VM instance print(f"Creating private VM instance {INSTANCE\_NAME}...") operation = compute.instances().insert( project=PROJECT\_ID, zone=ZONE, body=config ).execute() # Wait for the operation to complete wait\_for\_operation(compute, operation) print(f"Private VM instance {INSTANCE\_NAME} created successfully.") except HttpError as err: print(f"An error occurred: {err}") return def wait\_for\_operation(compute, operation): print("Waiting for operation to complete...") while True: result = compute.zoneOperations().get( project=PROJECT\_ID, zone=ZONE, operation=operation['name'] ).execute() if result['status'] == 'DONE': if 'error' in result: raise Exception(f"Error: {result['error']}") else: print("Operation completed successfully.") break else: print("Operation is still in progress...") time.sleep(10) if \_\_name\_\_ == "\_\_main\_\_": create\_instance() - in this script install nginx and tell how to access nginx using private ip in gcp

**ChatGPT said:**

Great — you’re creating a **private VM** inside a **custom VPC subnet**. Let's:

1. ✅ **Install NGINX** on the private VM using a startup script
2. 📥 Show you **how to access NGINX** using the **private IP** (through a **bastion host** or IAP tunnel)

**✅ Modified Script — Install NGINX on Private VM**

Just add a **startup-script** in the metadata section to install and start NGINX:

python

CopyEdit

import os

import google.auth

from googleapiclient.discovery import build

from googleapiclient.errors import HttpError

import time

# Configuration

PROJECT\_ID = 'manifest-zephyr-457115-v2'

ZONE = 'us-central1-a'

INSTANCE\_NAME = 'private-ubuntu-vm-2'

IMAGE\_PROJECT = 'ubuntu-os-cloud'

IMAGE\_FAMILY = 'ubuntu-2004-lts'

VPC\_NAME = 'hu-devops-25'

SUBNET\_NAME = 'private-subnet'

def create\_instance():

credentials, project = google.auth.default()

compute = build('compute', 'v1', credentials=credentials)

machine\_type = f"zones/{ZONE}/machineTypes/n1-standard-1"

source\_disk\_image = f"projects/{IMAGE\_PROJECT}/global/images/family/{IMAGE\_FAMILY}"

startup\_script = """#!/bin/bash

apt-get update

apt-get install -y nginx

systemctl enable nginx

systemctl start nginx

"""

network\_interface = {

'network': f'global/networks/{VPC\_NAME}',

'subnetwork': f'regions/us-central1/subnetworks/{SUBNET\_NAME}',

}

config = {

'name': INSTANCE\_NAME,

'machineType': machine\_type,

'disks': [{

'boot': True,

'autoDelete': True,

'initializeParams': {

'sourceImage': source\_disk\_image

}

}],

'networkInterfaces': [network\_interface],

'tags': {

'items': ['http-server']

},

'metadata': {

'items': [{

'key': 'startup-script',

'value': startup\_script

}]

}

}

try:

print(f"Creating private VM instance {INSTANCE\_NAME}...")

operation = compute.instances().insert(

project=PROJECT\_ID,

zone=ZONE,

body=config

).execute()

wait\_for\_operation(compute, operation)

print(f"✅ Private VM '{INSTANCE\_NAME}' with NGINX created successfully.")

except HttpError as err:

print(f"❌ An error occurred: {err}")

def wait\_for\_operation(compute, operation):

print("⏳ Waiting for operation to complete...")

while True:

result = compute.zoneOperations().get(

project=PROJECT\_ID,

zone=ZONE,

operation=operation['name']

).execute()

if result['status'] == 'DONE':

if 'error' in result:

raise Exception(f"❌ Error: {result['error']}")

print("✅ Operation completed successfully.")

break

time.sleep(5)

if \_\_name\_\_ == "\_\_main\_\_":

create\_instance()

**🔓 How to Access NGINX on a Private VM**

Since the VM has **no public IP**, you **cannot access it directly**. But you can still access NGINX by these 2 methods:

**🔁 Option 1: Use IAP Tunnel (Recommended for GCP Secure Access)**

1. Make sure **IAP is enabled** and you have the **roles/iap.tunnelResourceAccessor** permission.
2. Run this from your terminal:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--project=manifest-zephyr-457115-v2 \

--zone=us-central1-a \

--tunnel-through-iap

1. Once you're inside the VM:

bash

CopyEdit

curl http://127.0.0.1

You’ll see the NGINX welcome page.

**🔁 Option 2: Use a Bastion Host (Public VM in Same VPC)**

1. Create a **bastion VM** in the same VPC with a **public IP**.
2. SSH into the bastion:

bash

CopyEdit

gcloud compute ssh bastion-vm --zone=us-central1-a

1. From the bastion, access the private VM's internal IP:

bash

CopyEdit

curl http://<PRIVATE-IP>

You can get the private IP by running:

bash

CopyEdit

gcloud compute instances list --filter="name=private-ubuntu-vm-2" --format="get(networkInterfaces[0].networkIP)"

Want me to include bastion creation in your script too?

**You said:**

Option 1: Use IAP Tunnel (Recommended for GCP Secure Access) - how to enable this

**ChatGPT said:**

To enable **IAP Tunnel** (Identity-Aware Proxy Tunnel) for accessing your private VM securely, follow these steps:

**✅ Step-by-Step: Enable IAP Tunnel in GCP**

**🔐 1. Enable Required APIs**

Go to the following URL and enable:

* **IAP API**
* **Compute Engine API**

Direct links:

* Enable IAP API
* Enable Compute Engine API

**👤 2. Grant IAP Access Permissions**

You must give your user/service account **IAP SSH access**.

Go to **IAM & Admin > IAM**:  
https://console.cloud.google.com/iam-admin/iam

Then:

* Click **"Edit principal"** next to your account
* Add the following role:

🟢 **Role**: IAP-Secured Tunnel User (roles/iap.tunnelResourceAccessor)  
🔧 Optional (for SSH): Compute OS Login (roles/compute.osLogin)  
🔧 If using keys: Compute Instance Admin (roles/compute.instanceAdmin.v1)

**📍 3. Enable OS Login (Recommended)**

To use your Google credentials for SSH:

1. Go to:  
   https://console.cloud.google.com/compute/settings
2. Under **Metadata**, ensure the following key is present:
   * enable-oslogin = TRUE

If not, add it manually in the **project metadata**.

**🚀 4. Use IAP Tunnel to SSH into Private VM**

Run this in your terminal:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--project=manifest-zephyr-457115-v2 \

--zone=us-central1-a \

--tunnel-through-iap

If successful, you’ll enter the private VM directly — no public IP required!

**🧪 5. Test NGINX from Within the VM**

Inside the VM:

bash

CopyEdit

curl http://127.0.0.1

Or use the internal IP (if needed):

bash

CopyEdit

curl http://10.128.X.X

Want me to generate the required gcloud commands to set all of this up automatically?

**You said:**

 i am here where to go now

**ChatGPT said:**

Perfect — you're on the right screen in the IAM section of your GCP project. Now, let's grant the necessary role to **your Google account** so you can use the **IAP Tunnel**. Here’s what to do next:

**✅ Step-by-Step From Here:**

**🔹 1. Click “Grant Access”**

You’ll see a pop-up to add a principal and select roles.

**🔹 2. Under “New principals”, enter your Gmail address**

This is the account you're using to access GCP — likely the same as shown with the role **Owner** in your screenshot.

**🔹 3. Under “Role”, click “Select a role” and search for:**

* **IAP-Secured Tunnel User**
  + Path: Project > IAP-Secured Tunnel User
* ✅ Add this one for tunneling through IAP

🔧 *(Optional but helpful for SSH access)*:

* **Compute OS Login**
  + Path: Compute Engine > Compute OS Login
* Or: **Compute Viewer**
  + Path: Compute Engine > Viewer

**🔹 4. Click Save**

Now your account will have permissions to use **--tunnel-through-iap**.

**✅ Final Step: Try the IAP SSH Command**

Once the role is added, run this in your terminal:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--zone=us-central1-a \

--project=manifest-zephyr-457115-v2 \

--tunnel-through-iap

Let me know if you hit any prompt or error — I’ll help you out live.

**You said:**

Click “Grant Access” You’ll see a pop-up to add a principal and select roles. - i cant see grant access

**ChatGPT said:**

Thanks for the update! If you **can’t see the "Grant access" button**, it's usually because of one of these issues:

**🔍 Troubleshooting Why "Grant Access" Is Missing:**

**✅ 1. Do you have Owner or Editor role?**

From your screenshot:

* You are listed as Owner under the name "RUSEV" ✅ — so that should give full permissions.

But still, GCP sometimes hides the button due to:

* UI lag
* Collapsed window
* Incorrect view selected

**✅ Try This First:**

**🔹 Step 1: Maximize your browser or zoom out (Ctrl + -)**

The "Grant access" button is **sometimes hidden off-screen** on small resolutions.

**🔹 Step 2: Scroll up slightly on the IAM panel**

Make sure you are on the **“Allow”** tab (which you are) and **“View by principals”** (you are).

**🔄 Alternative Way to Grant Access (Always Works)**

Let’s directly grant via **IAM Role binding manually**:

**🔹 Go to this direct link:**

https://console.cloud.google.com/iam-admin/iam

**🔹 Click the blue "➕ Add" button near the top (top-center of the IAM list panel)**

If it still doesn't appear, you can also go to:

**🔁 Alternate Path:**

1. **Go to the left nav menu** → IAM & Admin → **IAM**
2. Look to the top for a blue button: **“+ Add”**
3. Add your Gmail (same one you're logged in with)
4. Choose the role:
   * IAP-Secured Tunnel User under "Project"
   * Compute OS Login (optional, for SSH access)

Let me know what you see, or feel free to share another screenshot if you're stuck again — I’ll walk you through it step by step!

**You said:**

**ChatGPT said:**

Perfect! You're exactly where you need to be — and now you're **one click away** from success.

**✅ Here's What to Do Now:**

Just click the **blue "Save" button** at the bottom of the dialog (bottom-left corner of the popup).

This will:

* Grant dk1078451@gmail.com the **IAP-Secured Tunnel User** role ✅
* And the **Compute OS Login** role ✅

**🔄 After Saving:**

Once saved, wait a few seconds for the roles to propagate. Then you can:

* Try **SSHing into your VM again** (using the Google Cloud Console or terminal)
* Or, if you're using Cloud Shell / IAP Tunnel — it should now work.

Let me know once you’ve hit save, or if the SSH still gives an error — I’ll guide you right through that next too 👨‍💻

**You said:**

Welcome to Cloud Shell! Type "help" to get started. Your Cloud Platform project in this session is set to manifest-zephyr-457115-v2. Use gcloud config set project [PROJECT\_ID] to change to a different project. dk1078451@cloudshell:~ (manifest-zephyr-457115-v2)$ gcloud compute ssh private-ubuntu-vm-2 \ --zone=us-central1-a \ --project=manifest-zephyr-457115-v2 \ --tunnel-through-iap WARNING: To increase the performance of the tunnel, consider installing NumPy. For instructions, please see https://cloud.google.com/iap/docs/using-tcp-forwarding#increasing\_the\_tcp\_upload\_bandwidth ERROR: [0] Error during local connection to [stdin]: Error while connecting [4003: 'failed to connect to backend']. (Failed to connect to port 22) Traceback (most recent call last): File "/usr/bin/../lib/google-cloud-sdk/lib/googlecloudsdk/command\_lib/compute/iap\_tunnel.py", line 775, in RunReceiveLocalData websocket\_conn = self.\_InitiateConnection( ^^^^^^^^^^^^^^^^^^^^^^^^^ File "/usr/bin/../lib/google-cloud-sdk/lib/googlecloudsdk/command\_lib/compute/iap\_tunnel.py", line 744, in \_InitiateConnection new\_websocket.InitiateConnection() File "/usr/bin/../lib/google-cloud-sdk/lib/googlecloudsdk/api\_lib/compute/iap\_tunnel\_websocket.py", line 152, in InitiateConnection self.\_WaitForOpenOrRaiseError() File "/usr/bin/../lib/google-cloud-sdk/lib/googlecloudsdk/api\_lib/compute/iap\_tunnel\_websocket.py", line 444, in \_WaitForOpenOrRaiseError raise ConnectionCreationError(error\_msg) googlecloudsdk.api\_lib.compute.iap\_tunnel\_websocket.ConnectionCreationError: Error while connecting [4003: 'failed to connect to backend']. (Failed to connect to port 22) Connection closed by UNKNOWN port 65535 Recommendation: To check for possible causes of SSH connectivity issues and get recommendations, rerun the ssh command with the --troubleshoot option. gcloud compute ssh private-ubuntu-vm-2 --project=manifest-zephyr-457115-v2 --zone=us-central1-a --troubleshoot Or, to investigate an IAP tunneling issue: gcloud compute ssh private-ubuntu-vm-2 --project=manifest-zephyr-457115-v2 --zone=us-central1-a --troubleshoot --tunnel-through-iap ERROR: (gcloud.compute.ssh) [/usr/bin/ssh] exited with return code [255]. dk1078451@cloudshell:~ (manifest-zephyr-457115-v2)$

**ChatGPT said:**

Thanks for sharing the full output — this tells us a **lot**. You're very close — here's a breakdown and the **exact fix steps**.

**❌ Problem Summary:**

The error:

vbnet

CopyEdit

Error while connecting [4003: 'failed to connect to backend']. (Failed to connect to port 22)

means that:

* IAP **can’t connect to port 22** on your VM.
* Your **VM is private** (no external IP) and IAP is being used correctly.
* But... **port 22 is blocked** in the VM's **Firewall Rules**. 🔥🚫

**✅ Solution: Open Port 22 via Firewall Rule**

1. Go to **VPC network > Firewall** in GCP Console  
   Or click here directly
2. Click **“CREATE FIREWALL RULE”** at the top.
3. Fill in the details:
   * **Name**: allow-ssh-iap
   * **Network**: Choose the same network your VM is using (probably default)
   * **Targets**: All instances in the network (or choose specific tags if you prefer)
   * **Source IP ranges**:

CopyEdit

35.235.240.0/20

*(This is the official IP range Google IAP uses)*

* + **Protocols and ports**:
    - Check Specified protocols and ports
    - Tick tcp and write: 22

1. Click **Create**.

**🔄 After Firewall is Created:**

Wait ~1–2 minutes for it to take effect. Then re-run:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--zone=us-central1-a \

--project=manifest-zephyr-457115-v2 \

--tunnel-through-iap

**🧠 Why This Works:**

IAP needs to reach **port 22** (SSH) on your VM from its own IP range. If the firewall doesn’t allow that, it will always fail, even if roles/permissions are perfect.

Let me know once you’ve created the rule — or if you want me to generate the equivalent gcloud CLI command instead of using the UI.

**You said:**

how to open port 22 in VM

**ChatGPT said:**

To open **port 22 (SSH)** for your VM in Google Cloud, follow these steps:

**✅ Step 1: Go to VPC network > Firewall rules**

1. In the Google Cloud Console, search for **"VPC network"** in the top search bar.
2. Under "VPC network", click **Firewall rules**.

**✅ Step 2: Create a New Firewall Rule**

Click on **“Create Firewall Rule”** and fill the form as follows:

* **Name**: allow-ssh
* **Network**: Select the VPC network of your VM (usually default)
* **Priority**: 1000
* **Direction of traffic**: Ingress
* **Action on match**: Allow
* **Targets**: Choose either:
  + All instances in the network **OR**
  + Specified target tags (and make sure to **add the same tag to your VM**, like ssh-access)
* **Source IP ranges**: 0.0.0.0/0 (for public access) or restrict to your IP for security
* **Protocols and ports**: Select **“Specified protocols and ports”** → check tcp and enter 22

Click **Create**.

**✅ Step 3: Add Network Tag (if required)**

From your screenshot, if you're using **Network tags**, ensure your VM instance has the same tag (e.g., ssh-access) used in the firewall rule.

To add a tag:

1. Go to **Compute Engine > VM Instances**.
2. Click your VM name → **Edit**.
3. Scroll to **Network tags** → Add the tag used in your firewall rule (e.g., ssh-access).
4. Click **Save**.

**✅ Step 4: Try Connecting Again**

Once the firewall rule is in place and propagated:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--zone=us-central1-a \

--project=manifest-zephyr-457115-v2 \

--tunnel-through-iap

If you’re still having issues, let me know if this VM has an external IP or if it's fully private (internal IP only). SSH over IAP needs specific IAM permissions and proper IAP configuration.

Want me to help check that too?

**You said:**

**ChatGPT said:**

Thanks for the screenshot! You're currently in **Cloud Armor’s Network Firewall Policy**, which is different from the **VPC Firewall rules** section we need to modify to allow SSH (tcp:22).

Let me guide you to the correct place:

**✅ Follow These Steps to Add a VPC Firewall Rule for SSH**

1. Go to **Navigation Menu (☰)** → **VPC network** → Click on **Firewall**.

OR just visit this direct link to the correct section: 👉 https://console.cloud.google.com/networking/firewalls

1. Click **“Create Firewall Rule”**.
2. Fill the form like this:
   * **Name**: allow-ssh
   * **Network**: default (or whichever VPC your VM is in)
   * **Priority**: 1000 (default is fine)
   * **Direction of traffic**: Ingress
   * **Action on match**: Allow
   * **Targets**:
     + If unsure, use All instances in the network
     + Or use Specified target tags (e.g., ssh-access) if you want finer control.
   * **Source filter**:
     + **IP ranges**: 0.0.0.0/0 (allows access from anywhere)  
       ⚠️ You can restrict to your public IP for more security.
   * **Protocols and ports**:
     + Select **“Specified protocols and ports”**
     + Check tcp and enter 22
3. Click **Create**.

**✅ (Optional) If You Used Target Tags**

Add the tag to your VM:

* Go to **Compute Engine > VM Instances** → Click your VM → **Edit**
* Under **Network tags**, add the tag you used (e.g., ssh-access)
* Save

Once done, try your SSH command again:

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--zone=us-central1-a \

--project=manifest-zephyr-457115-v2 \

--tunnel-through-iap

Let me know once you've created the rule, or if you'd like me to stay with you while you do it.

**You said:**

 - what to enter in destination

**ChatGPT said:**

Thanks for the screenshot again! You're still in **Cloud Armor > Network Firewall Policies**, but what you need to configure is in the **VPC firewall section**, not Cloud Armor.

**🔁 Let's Correct the Path First:**

Please follow these exact steps to get to the **right place**:

1. In the **left side menu**, scroll down to **“VPC network”** → Click on **“Firewall”** (NOT “Firewall policies” under Cloud Armor).

Direct link for convenience:  
👉 https://console.cloud.google.com/networking/firewalls

**✅ Once You're in the Correct Firewall Section:**

1. Click on **“Create Firewall Rule”**.
2. Fill it like this:

* **Name**: allow-ssh
* **Network**: default (or whichever your VM uses)
* **Priority**: 1000 (default is fine)
* **Direction of traffic**: Ingress
* **Action on match**: Allow
* **Targets**: All instances in the network (for now)
* **Source filter**:
  + **IP ranges**: 0.0.0.0/0 (for open access from all IPs)
* **Protocols and ports**:
  + Select **"Specified protocols and ports"**
  + Check tcp and type 22 in the box

1. Click **Create**

**✅ Then Retry Your SSH Command:**

bash

CopyEdit

gcloud compute ssh private-ubuntu-vm-2 \

--zone=us-central1-a \

--project=manifest-zephyr-457115-v2 \

--tunnel-through-iap

Would you like me to walk you through the VPC firewall screen directly with you right now?