Networking Project 6 — VPC Peering: Clean Step-by-Step Guide

Goal: Create **two VPCs**, peer them, launch one EC2 instance in each, verify connectivity (ping private IPs), then delete everything.

> 🔔**Important:** Delete all resources by the end of the day to avoid charges.

Prerequisites

- AWS account with **IAM Admin** access (or permissions to create VPCs, EC2, EIP, and modify security groups/route tables).
- Region selected in the top-right of the AWS Console (use the same region for both VPCs).
- Recommended: a note of your current public IP (for safer SSH rules) — you can find it with `whatismyip` websites.

Quick checklist (one-line version)

1. Create **NextWork-1** VPC — CIDR `10.1.0.0/16` (1 public subnet)
2. Create **NextWork-2** VPC — CIDR `10.2.0.0/16` (1 public subnet)
3. Create & **accept** VPC peering `VPC 1 <> VPC 2`
4. Add route entries in both route tables (point remote CIDR to peering connection)
5. Launch EC2 instance in each VPC (Amazon Linux 2023, `t2.micro`)
6. Allocate & associate Elastic IP (to VPC1 instance) so Instance Connect works
7. Update security groups: allow SSH (to VPC1) and allow ICMP from other VPC's CIDR (to VPC2)
8. Connect to Instance-NextWork VPC1 via EC2 Instance Connect and `ping <private-ip-of-vpc2>`
9. Verify ping replies
10. Clean up (terminate instances, release EIP, delete peering, delete VPCs)

Detailed step-by-step

1) Login & open consoles

- Sign in to the AWS Console as your Admin user.
- Open **VPC** service (search `VPC` in the top search bar).

2) Create VPC 1 (NextWork-1)

- VPC Console → **Your VPCs** → **Create VPC** → choose **VPC and more**.
- Settings:
- **Name tag auto-generation:** `NextWork-1`
- **IPv4 CIDR block:** `10.1.0.0/16`

- **IPv6 CIDR block:** `No IPv6 CIDR block`
- **Tenancy:** `Default`
- **Number of AZs:** `1`
- **Number of public subnets:** `1`
- **Number of private subnets:** `0`
- **NAT gateways:** `None`
- **VPC endpoints:** `None`
- Leave DNS options checked
- Click **Create VPC** → **View VPC** → (optional) open **Resource map** to confirm resources created.

  Tip: note the VPC ID (vpc-xxxxxxxx) for later filtering.

3) Create VPC 2 (NextWork-2)

- Repeat the steps above but with:
- **Name:** `NextWork-2`
- **IPv4 CIDR block:** `10.2.0.0/16` (must be unique)
- Create and view the VPC. Note its VPC ID.

4) Create VPC Peering (VPC 1 ⇄ VPC 2)

- In VPC Console → **Peering connections** → **Create peering connection**.
- Fill in:
- **Peering connection name:** `VPC 1 <> VPC 2`
- **Requester VPC:** `NextWork-1-vpc` (your VPC 1)
- **Accepter VPC:** `NextWork-2-vpc` (same account, same region)
- Click **Create peering connection**.
- The status will be **Requested**. Select the peering connection → **Actions** → **Accept request** → **Accept**.
- Optional step shown by console: click **Modify my route tables now** to jump to route updates.

5) Update route tables — allow cross-VPC routing

Important: When adding routes, use the *remote* CIDR (not your own!).

- For **VPC 1**'s public route table (likely named `NextWork-1-rtb-public`):
- Select that route table in **Route tables**.
- **Routes** tab → **Edit routes** → **Add route**.
- **Destination:** `10.2.0.0/16`
  Target: `Peering Connection` → select `VPC 1 <> VPC 2`.

- Save changes.

- For **VPC 2**'s public route table (`NextWork-2-rtb-public`):

- Edit routes → Add route.
- **Destination:** `10.1.0.0/16`
  Target: `Peering Connection` → select the same peering.
- Save changes.

  If you mistakenly add your own CIDR as a destination the console will error — correct the typo to the remote CIDR.

6) Launch EC2 instance in VPC 1 (Instance - NextWork VPC 1)

- Open **EC2** console → **Instances** → **Launch instances**.
- Settings:
- **Name:** `Instance - NextWork VPC 1`
- **AMI:** Amazon Linux 2023
- **Instance type:** `t2.micro`
- **Key pair:** *Proceed without a key pair* (project choice) — note: less secure.
- **Network settings → Edit**:
  - **VPC:** `NextWork-1-vpc`
  - **Subnet:** the public subnet created earlier
  - **Auto-assign public IP: Disable** (we will allocate an Elastic IP later)
- **Firewall (security groups):** Select existing security group (default for this VPC) — we will edit it.
- Launch instance.

7) Launch EC2 instance in VPC 2 (Instance - NextWork VPC 2)

- Repeat above with:
- **Name:** `Instance - NextWork VPC 2`
- **VPC:** `NextWork-2-vpc`
- **Subnet:** VPC2 public subnet
- **Auto-assign public IP:** Disable
- Launch instance.

> Alternative: If you want to skip Elastic IP steps, enable **Auto-assign public IPv4** during launch — then Instance Connect can work without allocating an EIP. This guide follows the project flow (disable then allocate EIP).

8) Allocate & associate an Elastic IP (so Instance Connect can reach the instance)

- EC2 Console → **Elastic IPs** → **Allocate Elastic IP addresses** → leave defaults → **Allocate**.
- Select the newly allocated EIP → **Actions** → **Associate Elastic IP address**.
- **Instance:** choose `Instance - NextWork VPC 1` → **Associate**.
- Verify the instance now shows a **Public IPv4 address** in the Instances table.

9) Fix default security group inbound rule for SSH (VPC1)

- VPC Console → **Security groups** → filter by VPC ID for VPC1 to find the default SG.
- Select it → **Inbound rules** → **Edit inbound rules** → **Add rule**:
- **Type:** SSH
- **Source type:** `Anywhere-IPv4` (0.0.0.0/0) — **project choice**
- **Save rules**

> **Security note:** `Anywhere-IPv4` is convenient but open. If you can, restrict SSH to your IP (e.g., `x.x.x.x/32`) or use Session Manager (SSM) in production.

10) Connect to Instance - NextWork VPC 1 using EC2 Instance Connect

- EC2 Console → select `Instance - NextWork VPC 1` → **Connect** → **EC2 Instance Connect** → **Connect**.
- You get a web terminal session.

11) From Instance 1, ping Instance 2's private IP

- In EC2 Instances page, copy **Private IPv4** of `Instance - NextWork VPC 2` (it will be in the `10.2.x.x` range).
- In the Instance Connect terminal (for VPC1 instance), run:

```
ping -c 4 <PRIVATE_IPV4_OF_INSTANCE2>
```

  Example: `ping -c 4 10.2.1.123`

- Expected: multiple replies showing `time=` and `ttl=` values.

12) If ping fails: quick troubleshooting checklist

1. **Peering status** — VPC Peering should show **Active** (VPC Console → Peering connections).
2. **Route tables** — verify route entries exist in both route tables (destination: remote CIDR → target: peering).
3. **Network ACLs** — open the subnet's NACL and verify it allows the traffic (default NACL allows all; if modified, ensure ICMP allowed).
4. **Security Groups (VPC2)** — make sure `Instance - NextWork VPC 2`'s security group allows ICMP from VPC1 CIDR:
5. Security groups → filter by VPC2 ID → select SG → **Inbound rules** → **Edit inbound rules** → Add:
   - **Type:** `All ICMP - IPv4`
   - **Source:** `10.1.0.0/16`
6. Save rules.
7. **Confirm subnet association** — ensure instances are in the public subnet that has the internet gateway route (for EIP access) and the route table we edited.

13) Confirm success

- From Instance 1 terminal, `ping` should return replies. Celebrate.

Teardown (delete everything) — DO THIS when finished

1. **Terminate EC2 instances**
2. EC2 → Instances → select both `Instance - NextWork VPC 1` and `Instance - NextWork VPC 2` → **Instance state** → **Terminate instance** → Confirm.
3. **Release Elastic IP**
4. EC2 → Elastic IPs → select allocated EIP → **Actions** → **Release Elastic IP addresses** → Confirm.
5. **Delete VPC peering connection**
6. VPC → Peering connections → select `VPC 1 <> VPC 2` → **Actions** → **Delete peering connection** → Check **Delete related route table entries** → Type `delete` → Delete.
7. **Delete VPCs**
8. VPC → Your VPCs → select `NextWork-1-vpc` → **Actions** → **Delete VPC** → Type `delete` → Delete.
9. Repeat for `NextWork-2-vpc`.
10. If deletion is blocked, check for lingering network interfaces (ENIs) — delete those first.

11. Refresh pages to confirm nothing remains.

    Final sanity check: EC2 → Instances should be empty; VPCs list should not show NextWork-1/2.

Screenshots to capture (if asked by the project)

- VPC list after creating both VPCs (show names and CIDRs)
- Resource map view for each VPC
- Peering connection **Requested** and then **Active** (after accept)
- Route table entries showing the new route to remote CIDR
- EC2 Instances page showing instances and the public IP on Instance1
- EC2 Instance Connect terminal showing successful `ping` replies
- Final confirmation of deleted resources (optional)

Quick CLI cheatsheet (optional)

These are **examples** — adapt region/account as needed.

```
# Create a VPC (example)
aws ec2 create-vpc --cidr-block 10.1.0.0/16 --tag-specifications
'ResourceType=vpc,Tags=[{Key=Name,Value=NextWork-1}]'

# Allocate an Elastic IP
aws ec2 allocate-address --domain vpc

# Terminate an instance
aws ec2 terminate-instances --instance-ids i-0123456789abcdef0
```

Security & best-practice notes

- Don't leave SSH open to `0.0.0.0/0` in production. Limit SSH to your IP or use Session Manager (SSM).
- VPC peering does not support transitive routing — if you later add a 3rd VPC, you must peer each pair explicitly or use a transit gateway.
- Deleting unused resources stops charges — do it.

Done — next steps

- Want this as a printable **checklist** (one-line tasks with checkboxes)? I can convert it.
- Want the **CLI-only** version for automation (CloudFormation, Terraform, or `aws` CLI scripts)? I can produce that too.

Good luck — go make clouds connect and then delete them like a responsible adult. 🔮