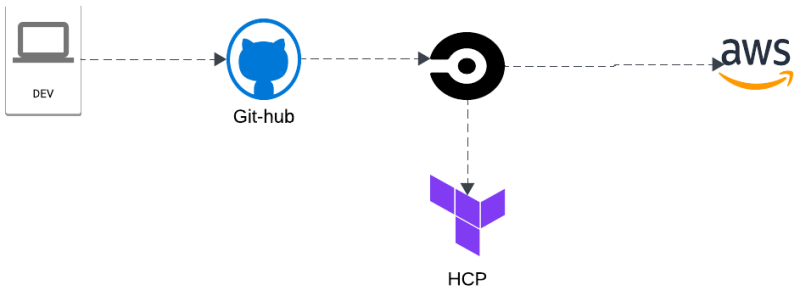
Automate Transit Gateway Deployment

. Project Overview

- **Summary**:

  This project implements a multi-VPC AWS architecture that provides secure cross-VPC communication and centralized internet access through a private subnet landing zone in VPC3. It leverages Terraform for Infrastructure-as-Code (IaC) with Terraform Cloud as the remote backend for collaborative state management, and CircleCI for automated deployments, ensuring a streamlined, consistent, and scalable setup for networking and CI/CD pipelines.

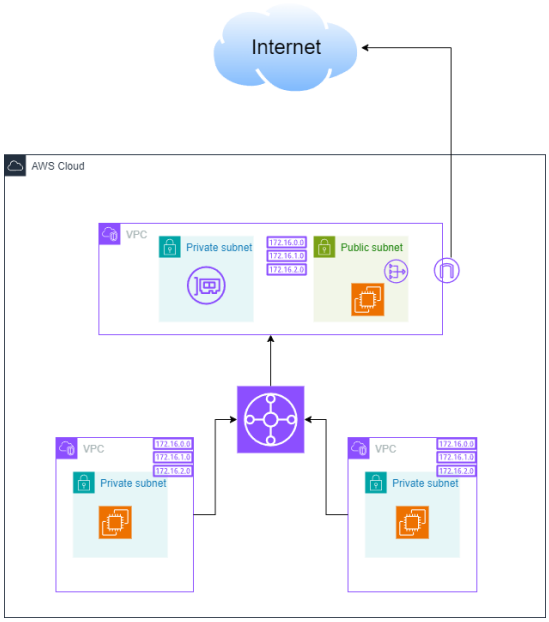
- **Technologies Used**:
  - **Terraform**: For defining and provisioning AWS resources.
  - **Terraform Cloud**: Used as the remote backend for managing state and enabling team collaboration.
  - **AWS Services**: Transit Gateway, NAT Gateway, VPCs, subnets, EC2 instances, and route tables.
  - **CircleCI**: To automate Terraform workflows and enforce deployment best practices.
  - **GitHub/GitLab**: For version control and pipeline triggers.



. Architecture and Workflow

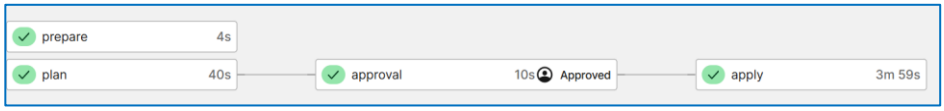
- **AWS Architecture Overview**:
  - **VPC1 & VPC2**:
    - Each contains private subnets for application workloads.
  - **VPC3**:
    - **Public Subnet**: Hosts the NAT Gateway.
    - **Private Subnet (Landing Zone)**: Serves as the initial hop for private subnets in VPC1 and VPC2 to reach the internet.
  - **Transit Gateway**:
    - Facilitates cross-VPC communication and routing of internet-bound traffic from VPC1 and VPC2 to VPC3.

- **Workflow**:
  - **Terraform Infrastructure Setup**:
    - Provisions the three VPCs, subnets, NAT Gateway, Transit Gateway, route tables, and required connectivity.
    - An **AWS Infrastructure Diagram** showing:
      - Transit Gateway connecting VPC1, VPC2, and VPC3.
      - Routing paths from private subnets in VPC1 and VPC2 through the landing zone in VPC3 to the internet.



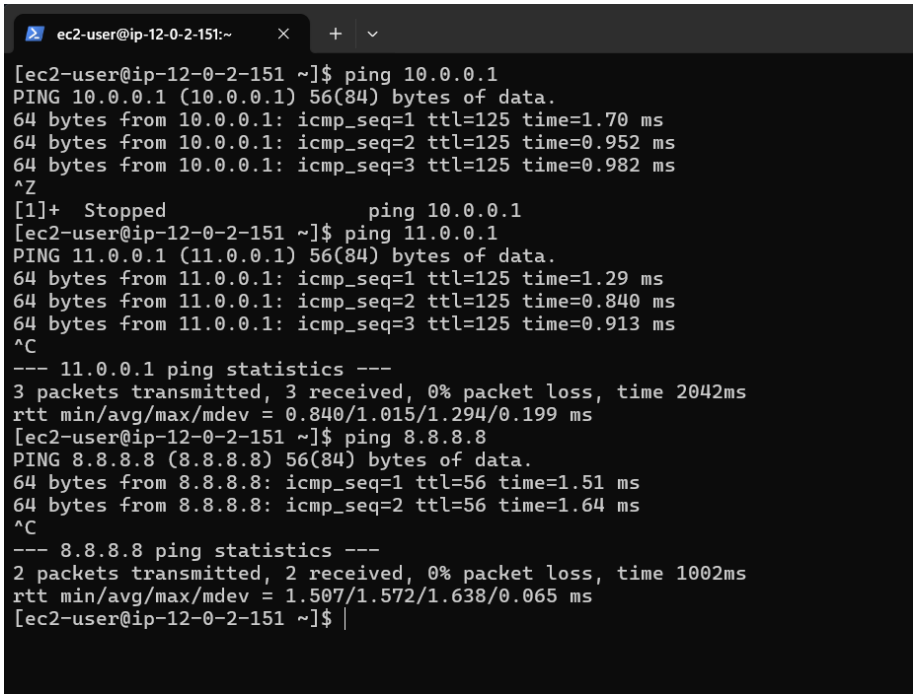
  - **Routing Logic**:
    - Private subnets in VPC1 and VPC2 route internet-bound traffic via the Transit Gateway to the landing zone (private subnet in VPC3).
    - Landing zone routes traffic to the NAT Gateway, which connects to the Internet Gateway.

- **CircleCI Integration**:
  - **Pipeline Stages**:
    - `terraform fmt`: Ensures Terraform code is formatted correctly.
    - `terraform validate`: Verifies Terraform configuration.
    - `terraform plan`: Previews changes to infrastructure.
    - `terraform apply`: Deploys the infrastructure to AWS after approval.



- **Testing and Validation**:
  - Use tools like `traceroute` and `ping` to validate:
    - Cross-VPC connectivity via Transit Gateway.
    - Internet access for private subnets in VPC1 and VPC2 via NAT Gateway in VPC3.

Public Subnet:



VPC 1 Private Subnet:

```
[ec2-user@ip-10-0-0-214 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=2.59 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=1.90 ms
10.0.0.21464 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=2.09 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=54 time=1.85 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=54 time=1.81 ms
```

```
[ec2-user@ip-10-0-0-214 ~]$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  * * *
 2  12.0.2.28 (12.0.2.28)  18.434 ms  18.414 ms  18.400 ms
 3  216.182.239.243 (216.182.239.243)  56.112 ms * 216.182.229.28 (216.182.229.28)  18.372 ms
 4  100.66.9.228 (100.66.9.228)  18.330 ms 100.66.9.204 (100.66.9.204)  18.314 ms 100.66.9.154 (100.66.9.154
 5  100.66.15.210 (100.66.15.210)  18.446 ms 100.66.10.198 (100.66.10.198)  18.272 ms 100.66.10.124 (100.66.
211 ms
 6  241.0.11.138 (241.0.11.138)  18.243 ms 241.0.11.147 (241.0.11.147)  1.604 ms 241.0.11.146 (241.0.11.146)
 7  240.3.180.13 (240.3.180.13)  2.530 ms  5.208 ms 241.0.11.144 (241.0.11.144)  1.831 ms
 8  99.82.181.22 (99.82.181.22)  2.391 ms  2.405 ms 240.3.180.13 (240.3.180.13)  2.970 ms
 9  72.14.203.158 (72.14.203.158)  3.156 ms * 99.82.181.23 (99.82.181.23)  2.355 ms
10  72.14.203.158 (72.14.203.158)  3.107 ms 192.178.105.111 (192.178.105.111)  4.867 ms 72.14.203.158 (72.14
.070 ms
11  * * 142.251.77.63 (142.251.77.63)  2.980 ms
12  dns.google (8.8.8.8)  2.250 ms 142.251.70.85 (142.251.70.85)  2.608 ms dns.google (8.8.8.8)  2.233 ms
```

VPC 2 Private Subnet:

```
ec2-user@ip-11-0-0-146:~
[ec2-user@ip-11-0-0-146 ~]$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=54 time=3.62 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=54 time=2.13 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=54 time=2.15 ms
^C
```

```
[ec2-user@ip-11-0-0-146 ~]$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  * * *
 2  12.0.2.28 (12.0.2.28)  11.936 ms  11.911 ms  11.888 ms
 3  216.182.229.30 (216.182.229.30)  27.521 ms * 216.182.229.0 (216.182.229.0)  22.951 ms
 4  100.65.24.224 (100.65.24.224)  19.406 ms 100.66.9.84 (100.66.9.84)  21.016 ms *
 5  100.66.8.102 (100.66.8.102)  20.300 ms 100.66.8.90 (100.66.8.90)  15.557 ms 100.66.15.64 (100.66.15.64)
 6  241.0.11.143 (241.0.11.143)  11.677 ms 241.0.11.158 (241.0.11.158)  2.073 ms 100.66.15.158 (100.66.15.158
 7  240.3.180.13 (240.3.180.13)  2.958 ms 240.3.180.15 (240.3.180.15)  2.941 ms  3.067 ms
 8  * 240.3.180.13 (240.3.180.13)  2.427 ms *
 9  99.82.181.23 (99.82.181.23)  2.551 ms 72.14.203.158 (72.14.203.158)  3.193 ms 99.82.181.22 (99.82.181.22)
10  216.239.62.111 (216.239.62.111)  3.390 ms *  3.557 ms
11  142.251.255.179 (142.251.255.179)  2.499 ms dns.google (8.8.8.8)  3.012 ms 142.251.77.61 (142.251.77.61)
[ec2-user@ip-11-0-0-146 ~]$
```

. Outcomes and Benefits

- **Outcomes**:
  - Centralized NAT Gateway significantly reduces cost by eliminating multiple NAT Gateways and saves ~$32.4/month or ~$388.8/year
  - Seamless cross-VPC communication enabled by Transit Gateway.
  - Fully automated CI/CD pipeline simplifies infrastructure deployment and updates.

- **Benefits**:
  - **Cost Savings**:

- By using a centralized NAT Gateway in VPC3 instead of individual NAT Gateways for each VPC, you save ~$32.4/month or ~$388.8/year.
- If scaled to multiple environments or regions, the savings increase proportionally.

  - **Deployment Time Efficiency**:

Deploying and testing this multi-VPC architecture manually could take 4-6 hours per environment (including troubleshooting and validation).

- Using Terraform and CircleCI reduces this to less than **30 minutes**, improving agility and consistency.
- For teams managing multiple environments (e.g., dev, staging, prod), this adds up significantly.

- **Automation Coverage**:
  - Terraform and CircleCI cover ~90% of the infrastructure lifecycle, including provisioning, updates, and version control.
  - Minimal manual intervention needed, reducing human error and ensuring repeatability.

Conclusion

This project demonstrates the power of Infrastructure as Code (IaC) and automation in creating a scalable, secure, and cost-effective multi-VPC architecture on AWS. By integrating Terraform and CircleCI, the process of deploying and managing infrastructure becomes faster, more reliable, and repeatable, reducing potential errors associated with manual setups.

The centralized NAT Gateway solution significantly optimizes costs while the Transit Gateway simplifies cross-VPC communication. This architecture also enables seamless scalability for future environments, such as additional VPCs or regions. The successful implementation highlights the following key outcomes:

1. **Improved Efficiency**: Automation reduced deployment time from hours to under 30 minutes, ensuring faster rollouts and updates.
2. **Cost Optimization**: Centralized NAT Gateway saved approximately ~$388.8/year per additional NAT Gateway, enhancing budget efficiency.
3. **Enhanced Security and Consistency**: Isolated private subnets, consistent routing configurations, and automated pipelines ensure a reliable and secure infrastructure.

This project is a step forward in adopting modern cloud infrastructure practices, laying a strong foundation for future scalability and agility in cloud operations.