# Vectra cPacket cVu-V Azure Deployment Guide

Version: October 23, 2023

## Table of Contents

## Overview

For enterprises pursuing a lift-n-shift approach to the Microsoft Azure cloud, a key consideration is ensuring the same level of visibility, detection and response for their Azure footprint, as they have for their on-premise footprint. Vectra is the leader in network detection and response, leveraging artificial intelligence to analyze network traffic and detect advanced attacks in real-time.

Key to this solution is the ability to get packets from the network – which is challenging since the network virtual tap feature is not generally available to Microsoft Azure customers. To this end, Vectra proposes a joint solution with cPacket Networks that offers an agentless, scalable, easy-to-deploy approach to deliver packets to Vectra Sensors in Azure.

## Solution

The Vectra solution for Azure IaaS environments comprises of a footprint of Sensors deployed across the subscription. The Sensors ingest VxLAN encapsulated traffic from cPacket, extract rich metadata about every session and forward the metadata to a Vectra Brain for analytics.

Detections are accessible using the Vectra UI and metadata can be further forwarded either to the Vectra Recall service (Vectra Quadrant UX deployments only) or to the customer's data lake using Vectra Stream. Vectra Respond UX deployments can further analyze metadata using Instant and Advanced Investigation features integrated into the Respond UX.

This allows the Vectra solution to analyze all traffic leaving a subnet in Azure to any other subnet or leaving the cloud and identify attacks in any phase. The question is, how does network traffic reach the Vectra Sensors.

cPacket has products in a variety of areas that are complementary to Vectra deployments. cVu network packet broker products consolidate, monitor and forward packet data to Vectra. cTap provides test access point capability to forward wire data to Vectra. cStor packet capture appliances capture, record, and analyze network traffic for security and performance use cases. The cProbe flow generator generates and exports network flow data for customers requiring flow data in other tools. The cClear analytics engine provides single pane of glass dashboards and consistent workflows across the cPacket product line. Vectra can interoperate with both physical and virtual or cloud deployed cPacket products.

Typically, multiple cPacket cVu-v virtual appliances must be deployed across the Azure footprint. Traffic from Azure VMs gets forwarded in accordance with the Azure route tables for the Virtual Network. User Defined Routes (UDR) assigned to the subnet send all traffic from the Azure VMs in that subnet to the cVu-V virtual appliance.

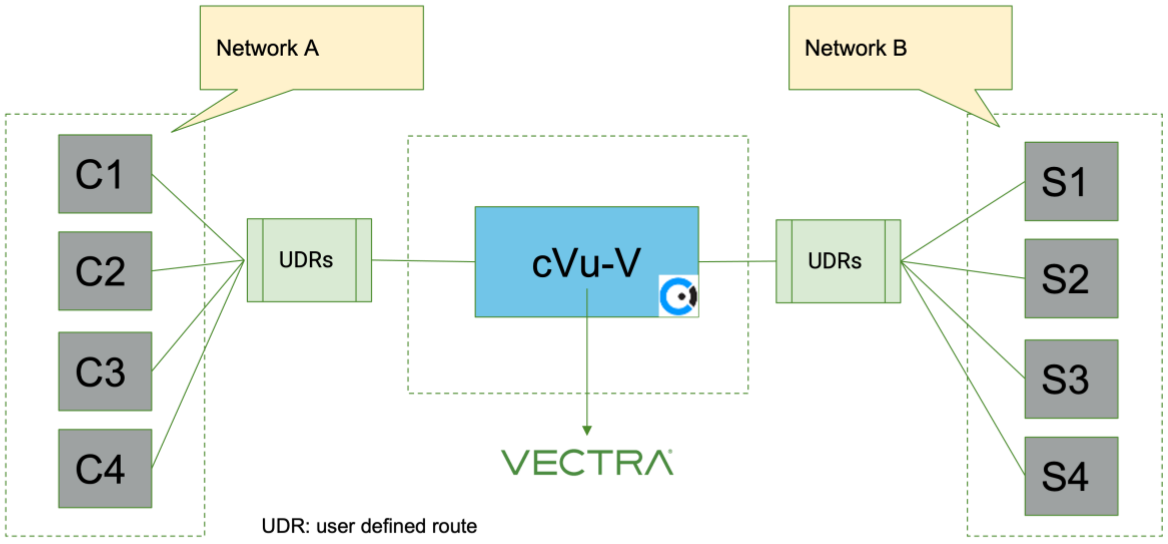
Once deployed, a cVu-V instance receives packets that were forwarded to it from the Azure network. The cVu-V acts as a bump in the wire that will forward all traffic received from the network to the destination.

For each packet received, cVu-V 's internal packet processing engine inspects the packet and compares it to a set of configured Rules and Filters to determine if the replica should then be forwarded to one or more downstream Endpoints (tools like Vectra Sensors or cStor-v packet capture) or dropped. The replica is encapsulated using a VxLAN header and forwarded to the Vectra Sensor.

This document will cover the example use case of capturing traffic in Azure IaaS environments and forwarding it via cVu-V to a Vectra vSensor that will produce and send a metadata stream to the Vectra Cognito Platform. It is not meant to replace deployment guides from cPacket, and Vectra recommends that customers work jointly with your Vectra account team and cPacket to ensure the smoothest deployment possible.

## High Level Deployment Examples

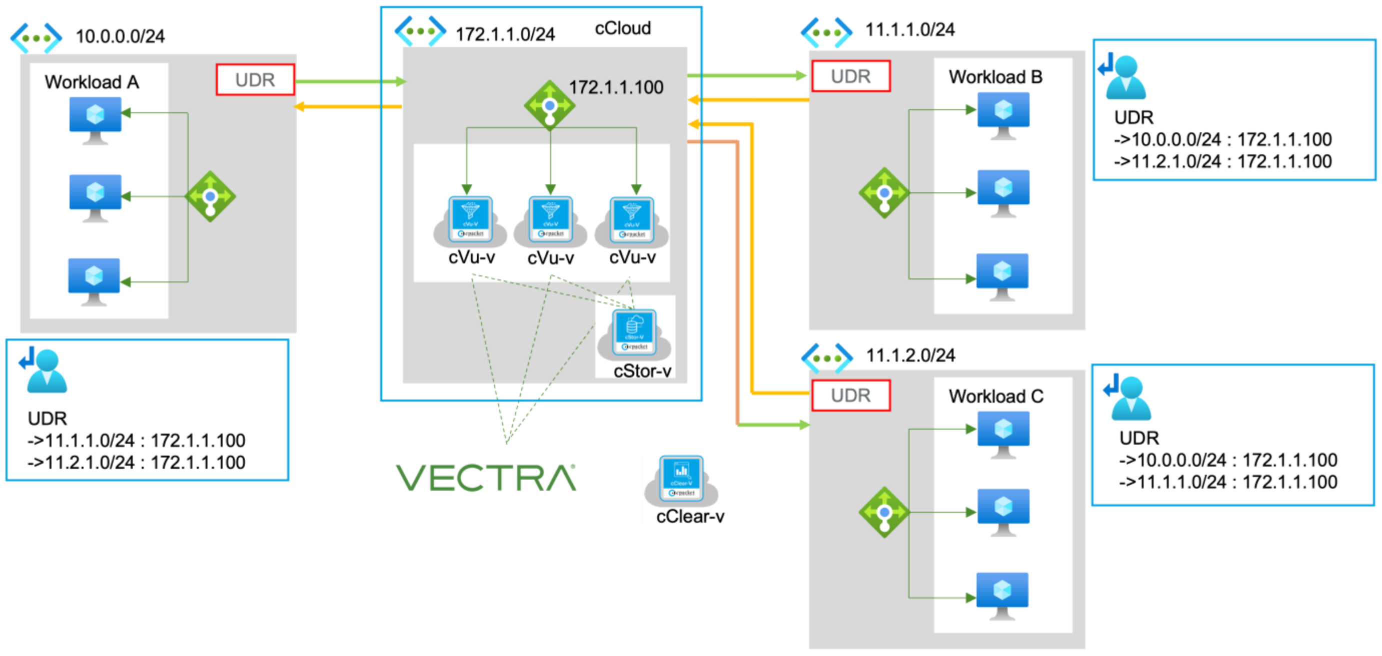
### Basic Concept



Deploying cPacket to feed packets to a Vectra Sensor is very straightforward.  Looking at the example above:

- Clients C1 through C4 on the left in Network A have an Azure route table with UDRs that point all traffic destined for Network B to the IP address assigned to cVu-V.
- When cVu-V receives a packet, it forwards it to a Vectra Sensor using VXLAN encapsulation.
    - It also forwards the traffic on to the destination server.
- Servers S1 through S4 on the right in Network B have an Azure route table with URDs that point all traffic destined for Network A to the IP address assigned to cVu-V

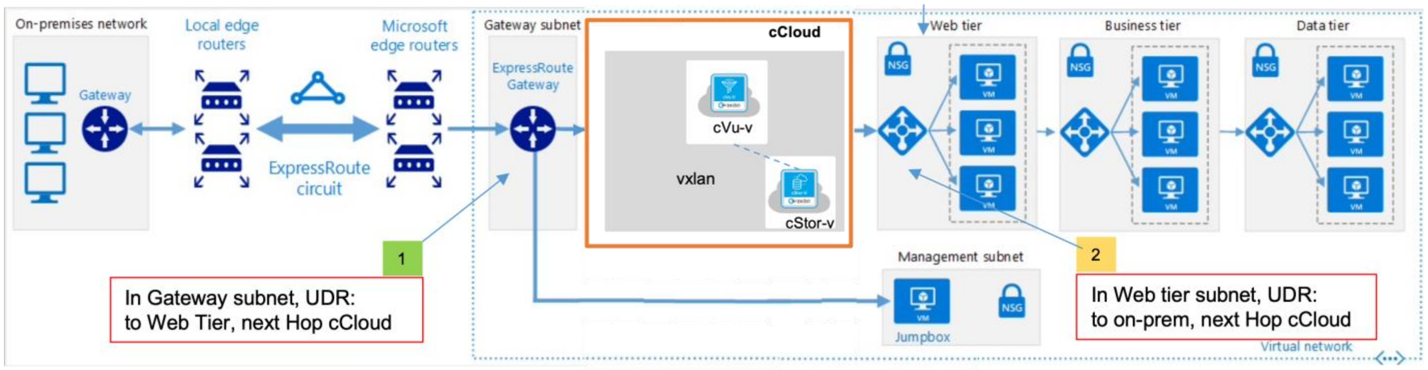
### Additional Detail with Azure Standard Load Balancer



In the example above we have a more detailed scenario including an Azure standard load balancer front ending the cVu-V instances.

- Workload A instances have an Azure route table with UDRs for Workload B and Workload C subnets that points to the IP address assigned to the Azure standard load balancer.
- Workload B instances have an Azure route table with UDRs for Workload A and Workload C subnets that points to the IP address assigned to the Azure standard load balancer.
- Workload C instances have an Azure route table with UDRs for Workload A and Workload B subnets that points to the IP address assigned to the Azure standard load balancer.
- Vectra Sensors would also need a load balancer deployed in front of them but in this diagram all of that is just represented by “Vectra” in the diagram.
- This example also shows cStor-V for packet capture and cClear-V for centralized dashboarding.

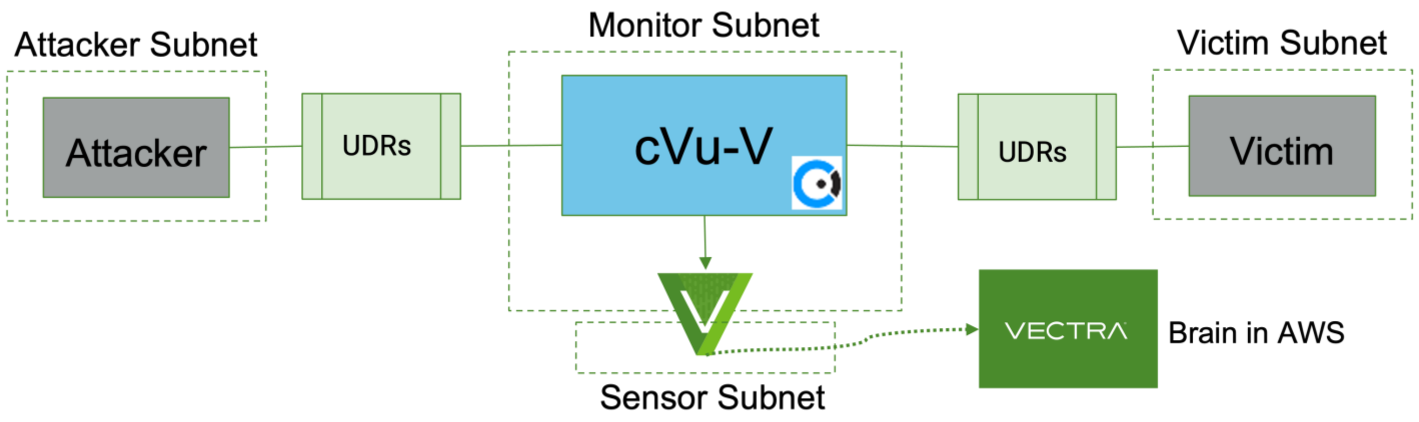
## Example with ExpressRoute



In the example above, we have a scenario with ExpressRoute connectivity between an on-premises network and an Azure deployment.

- In the gateway subnet, an Azure route table with UDRs would point traffic destined for the Web tier to the cCloud (cVu-V) deployment.
- In the Web tier subnet, an Azure route table with UDRs would point traffic destined for the on-premises network to the cCloud (cVu-V) deployment.
- Similar configurations could be applied for the Business tier and Data tiers.

## Example Deployment Discussed in this Deployment Guide



- The Brain used in this example is deployed in AWS but it could be in other supported clouds or a physical Brain appliance. It just needs to be reachable from the vSensor and where you want to log into it from.

- There is one VNET in Azure with 4 subnets.
  - An attacker subnet with a single ubuntu instance deployed in it.
  - A victim subnet with a single ubuntu instance deployed in it.
  - A monitor subnet that houses cVu-V and the traffic interface of the Vectra vSensor
    - The traffic interface is not required to be in the same subnet as cVu-V, just reachable from cVu-V.
  - A Sensor subnet that houses the management interface of the Vectra vSensor.
- User defined routes in Azure that are assigned to the attacker and victim subnets direct traffic to flow to cVu-V which then forwards VXLAN encapsulated copies of the traffic to the traffic capture interface of the Vectra vSensor.
- The Vectra vSensor then forwards metadata to the Vectra Brain

## Scaling and High Availability

cVu-V appliances are deployable in sizes that support 1 Gbps to 30 Gbps per appliances.  Multiple subnets can be pointed to the same appliance.  A single cVu-v virtual appliance can forward replicated traffic to multiple Vectra vSensors.

Vectra Azure vSensors are available in 1 or 2 Gbps sizes.  You can deploy as many Sensors or cVu-V appliances as needed to cover the required Azure footprint.

When deployed as a Scale Set, Azure automatically replaces virtual appliances in case of failure.  There is no state maintained on the cVu-V appliance or the Vectra vSensor, so there is minimal impact.

When deployed with an Azure Standard Load balancer, Microsoft offers a 99.99% availability SLA for the LB service.

Deploying cVu-V with UDRs to redirect traffic is less complicated than commonplace firewall deployments.  This implementation only needs changes to routes and does not introduce complex rulesets.  In the event of any issues, you can simply dissociate any route table from its associated subnet to return to a prior state before implementing cVu-V.

## Prerequisites

### Vectra

One of the below guides should be the starting point for your overall Vectra deployment:

- Vectra Respond UX Deployment Guide
- Vectra Quadrant UX Deployment Guide

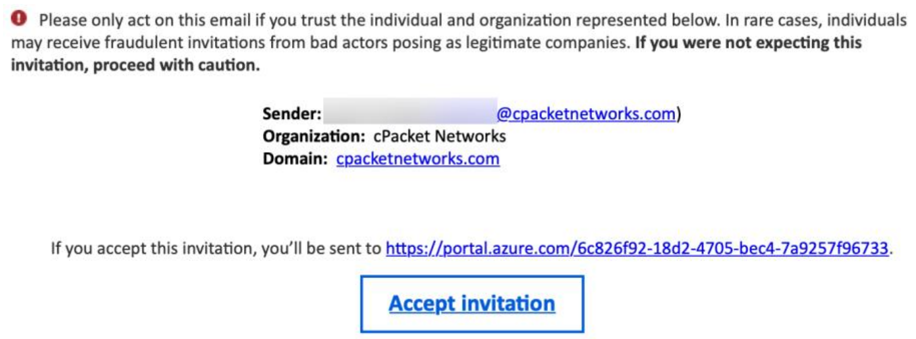
For more detail on Respond UX vs Quadrant UX please see Vectra Analyst User Experiences (Respond vs Quadrant).  Either of the above guides cover basic firewall rules needed for the overall deployment and initial platform settings.  Virtual Sensor (VMware, Hyper-V, KVM, AWS, Amazon, and GCP) configuration and pairing and covered in their respective guides.  Physical appliance pairing is covered in the Vectra Physical Appliance Pairing Guide.  Please see the Vectra Product Documentation Index on the Vectra support site for additional documentation including deployment guides for CDR for M365 / IDR for Azure AD and CDR for AWS.

You will also need a Vectra Sensor deployed in Azure for this solution.  Please refer to the following deployment guide for assistance:

- Azure Sensor Deployment Guide

## cPacket

- The Azure CLI must be installed.
    - Instructions are available here: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli.
    - cPacket also supports PowerShell instead of the Azure CLI
    - Please work with cPacket if this is a requirement for your organization.
- cPacket will require an Azure account login email from you so that they can share the cVu-V image with your Azure subscription via “Shared Image Gallery” functionality.
    - This must the login email that is use for access to Azure and not a separate corporate email address that the user might have.
    - Once cPacket shares the appliance images with you, you will receive an organization invite from Microsoft on behalf of cPacket that looks similar to the below:



Please only act on this email if you trust the individual and organization represented below. In rare cases, individuals may receive fraudulent invitations from bad actors posing as legitimate companies. **If you were not expecting this invitation, proceed with caution.**

**Sender:** [redacted]@cpacketnetworks.com)
**Organization:** cPacket Networks
**Domain:** cpacketnetworks.com

If you accept this invitation, you'll be sent to https://portal.azure.com/6c826f92-18d2-4705-bec4-7a9257f96733.

    - This is the same process that Vectra uses for access to the Azure Brain image.
- Click the “Accept invitation” link and follow the steps presented by Microsoft.
- On a machine with the Azure CLI installed, you must re-login using the Azure CLI by running the “**`az login`**” command.
    - This process will have you authenticate in your browser and then when you return to the shell you should see the Azure subscriptions that you have access to.
    - Please ensure that the login process shows you the following subscription (you can also run “**`az account list`**” to see the list at any time you are logged in)

```
{
    "cloudName": "AzureCloud",
    "homeTenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "id": "93004638-8c6b-4e33-ba58-946afd57efdf",
    "isDefault": false,
    "managedByTenants": [],
    "name": "cPacket Azure Root",
    "state": "Enabled",
    "tenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "user": {
      "name": "tbilen@demolab.vectra.ai",
      "type": "user"
    }
```

- You can list the appliance images that cPacket has shared with as follows:

```
az sig image-definition list -o table \
--gallery-name cpacketccloudpre \
-g cstor-aidsinga-rg1 \
--subscription "93004638-8c6b-4e33-ba58-946afd57efdf"
```

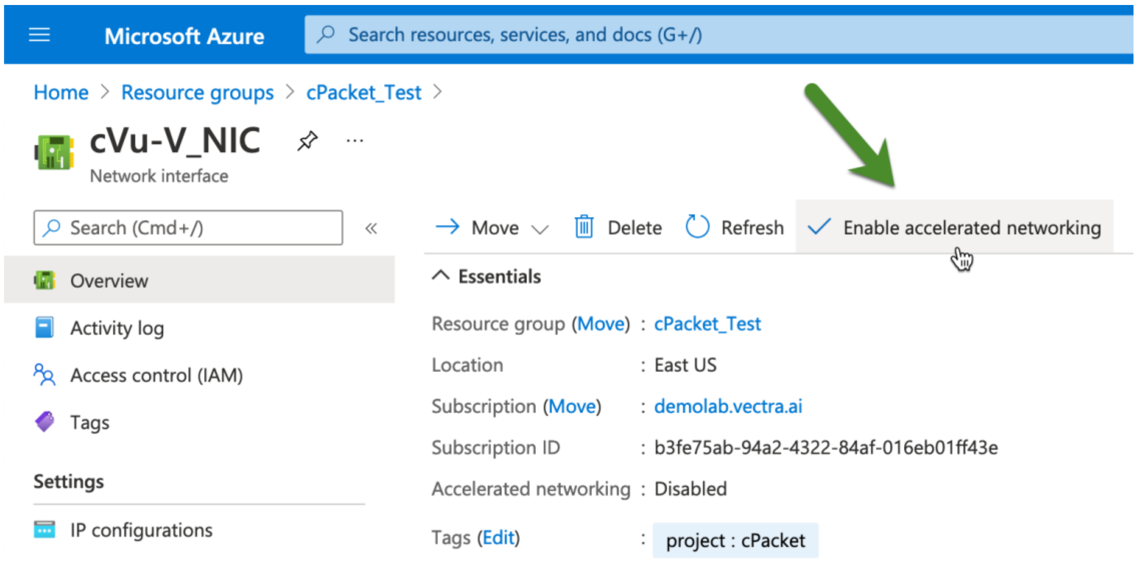
Approximate output:

```
HyperVGeneration    Location    Name          OsState      OsType    ProvisioningState
ResourceGroup
------------------  ----------  ------------- -----------  --------  -------------------
------------------
V1                  eastus2     cclearvpre    Generalized  Linux     Succeeded
cstor-aidsinga-rg1
V1                  eastus2     cstorvpre     Generalized  Linux     Succeeded
cstor-aidsinga-rg1
V1                  eastus2     cvuvinlinepre Generalized  Linux     Succeeded
cstor-aidsinga-rg1
```

- cPacket's deployment requires that NICs be pre-created that you will use with the cCloud family of appliances.
  - They require at least one Azure NIC for each of cVu-V, cStor-V, and cClear-V.
  - You will need the names of these NICs for use in the cPacket deployment scripts.

- Additional NIC configuration steps
  - On any cCloud appliance VM, the NICs should be configured to enable network acceleration. Also, on the cVu-V appliance, IP forwarding must be enabled for its NIC.
  - Note that network acceleration, while not strictly required, will allow the cCloud appliances to achieve close to the maximum performance available to the VM.

  - Setting NIC Acceleration:
    - To enable this, Microsoft recommends that any VM that the NIC is associated with must 1st be stopped and deallocated.
    - This can be enabled in the Azure portal by finding your NIC and then clicking on "Enable accelerated networking".

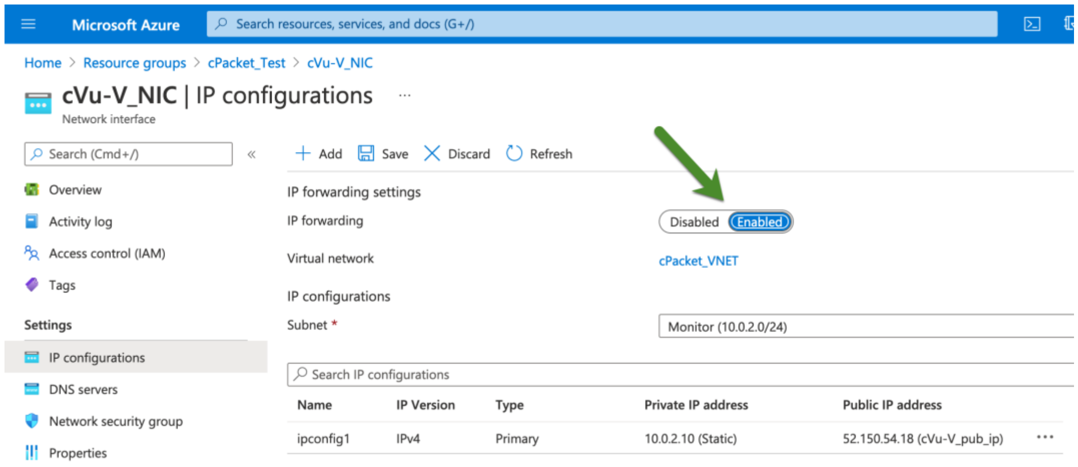


Resource group (Move) : cPacket_Test
Location : East US
Subscription (Move) : demolab.vectra.ai
Subscription ID : b3fe75ab-94a2-4322-84af-016eb01ff43e
Accelerated networking : Disabled
Tags (Edit) : project : cPacket

        - This can also be enabled from the Azure CLI.  Example syntax below:

```
az network nic update \
--name "YOUR_NIC_NAME" \
--resource-group "YOUR_RESOURCE_GROUP" \
--accelerated-networking true
```

    - Enabling IP forwarding
        - Some additional guidance from Microsoft is available here.
        - This can be enabled in the Azure portal if you change the setting to "Enabled" under IP forwarding settings in the IP configurations section of the NIC:



IP forwarding settings
IP forwarding: Disabled [✓ Enabled]
Virtual network: cPacket_VNET
IP configurations
Subnet *: Monitor (10.0.2.0/24)
Search IP configurations

| Name | IP Version | Type | Private IP address | Public IP address |
|---|---|---|---|---|
| ipconfig1 | IPv4 | Primary | 10.0.2.10 (Static) | 52.150.54.18 (cVu-V_pub_ip) |

        - This can also be enabled using the Azure CLI:

```
az network nic update \
--name "YOUR_NIC_NAME" \
--resource-group "YOUR_RESOURCE_GROUP" \
--ip-forwarding true
```

- Subnets
    - For Layer 3 setup (forcing IP packets to cVu-V using User Defined Routes):
        - Within the VNET, a subnet must be created for use by cPacket's appliances.
            - This is referred to as a the "Monitoring Network".
    - For Layer 4 setup (cVu-V behind an Azure Standard Load Balancer):
        - cPacket also recommends that cCloud appliances be on their own monitoring network.
            - This allows flexibility in terms of using both layer 3 and layer 4 traffic flows.
            - That said, for strictly layer 4 configurations with load balancers, a separate monitoring subnet is not required.
- An Azure "Standard Load Balancer" must be created when designing a fault tolerant solution.
    - Even when a single cVu-V could handle the load, it is a best practice to use multiple appliances for fault tolerance.
    - This can be a "public" load balancer or an "internal" load balancer that front ends multiple cVu-V appliances.
        - When configuring an Azure Standard Load Balancer you have a choice of NIC or IP address for the Backend Pool Configuration.  This should be set to IP address.
        - See https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview for additional detail.
- Create a storage account for cPacket virtual machine diagnostics:
    - Type: Storage (general purpose v1)

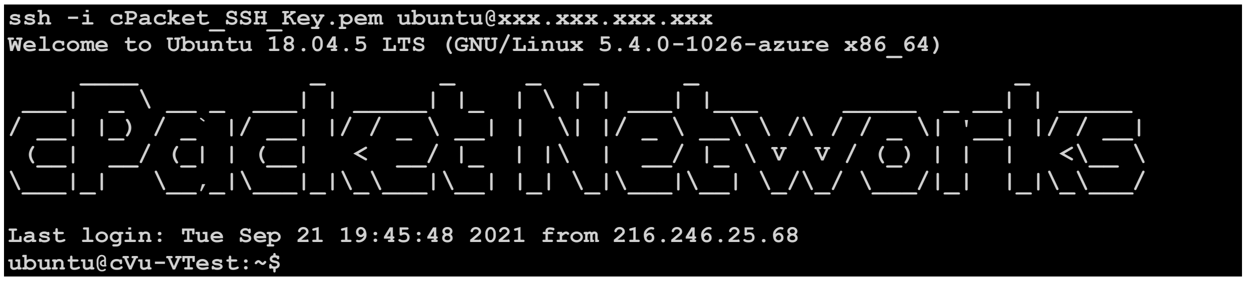
- Network Security Groups
  - Azure NSGs should be used to lock down access to the cCloud appliances to only those networks and machines that require access.
  - Inbound configuration:

| Port/Protocol | Source | Destination | Notes |
|---|---|---|---|
| 443 / HTTPS | Specific IPs or networks for management staff | VirtualNetwork | Web management UI |
| 22 / SSH | Specific IPs or networks for management staff | VirtualNetwork | Only required during tech support with cPacket technical staff |
| ICMP | VirtualNetwork, other management networks | VirtualNetwork | |
| Any/Any | UDRs or VirtualNetwork | UDRs or VirtualNetwork | Limit to sources that have been established with UDRs, or if desired, the broader rule to allow all on the current VNET with “VirtualNetwork” |

  - Outbound configuration:

| Port/Protocol | Source | Destination | Notes |
|---|---|---|---|
| Any/Any | UDRs or VirtualNetwork | UDRs or VirtualNetwork | Limit to sources that have been established with UDRs, or if desired, the broader rule to allow all on the current VNET with “VirtualNetwork” |

- SSH keys
  - An RSA SSH key pair will need to be created, or reuse an existing pair, for the cVu-v to allow an administrator to login to the CLI as the “**`ubuntu`**” user.
  - These can be generated using any standard tool.
  - The public key will need to be saved to a file so that it can be passed as an argument to the deployment command for cVu-V.
  - After cVu-V is deployed, you can login to the CLI via SSH using the private key that corresponds to the public key.
  - You may need to make the private key readable to you using a command such as:
    - **`chmod 400 cPacket_SSH_Key.pem`**
  - Example login command:



```
ssh -i cPacket_SSH_Key.pem ubuntu@xxx.xxx.xxx.xxx
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1026-azure x86_64)

      ____            _        _   _   _      _                      _
  ___|  _ \ __ _  ___| | _____| |_| \ | | ___| |___      _____  _ __| | _____
 / __| |_) / _` |/ __| |/ / _ \ __|  \| |/ _ \ __\ \ /\ / / _ \| '__| |/ / __|
| (__|  __/ (_| | (__|   <  __/ |_| |\  |  __/ |_ \ V  V / (_) | |  |   <\__ \
 \___|_|   \__,_|\___|_|\_\___|\__|_| \_|\___|\__| \_/\_/ \___/|_|  |_|\_\___/

Last login: Tue Sep 21 19:45:48 2021 from 216.246.25.68
ubuntu@cVu-VTest:~$
```

- cPacket recommends the Dsv3-series of VMs for deployment. They provide a variety of network bandwidth options suitable for high throughput packet processing. Sizing is beyond the scope of this document and Vectra recommends that you work with your cPacket account team for sizing of their solution.
  - The Dsv3-series of VMs supports 1 to 30 Gbps of throughput
  - Other Microsoft VM types may be feasible, but you should work with you cPacket technical rep to discuss requirements.
  - Our example deployment did not require high throughput so we will use the Standard_D2s_v3 type.

## cPacket Deployment

- **`cvuv-azure-launcher.sh`**
  - This is an optional script that can be used to add the full "**`az vm create`**" command to bash history.
  - You can then up arrow to edit and/or run the command.
  - For more information about using this command, please see cPacket's documentation.
- **`userdata-cvuv.txt`**
  - This file is passed as one of the arguments to the "**`az vm create`**" command.
  - It contains values to be used during the deployment of the cPacket cVu-V VM.
  - The key items in this file as it relates to a joint Vectra / cPacket deployment are:
  - "YOUR_CVUV_IP" – IP address of the cVu-V VM
  - "REMOTE_TOOL_IP" – IP address assigned in the Frontend IP configuration of the load balancer that is deployed by default with the Vectra vSensor
  - Here is an unconfigured example:

```
#!/bin/bash

# cVu-V-k inline mode config example

# cvuv_nat_xxx values define NAT passthroughs - up to 4 allowed
#   (suffix _0,_1,_2, _3)
#    NOTE : local RESERVED ports 443,80,22,161,162
# cvuv_nat_loc_ip, cvuv_nat_dst_ip : emptry strings ('') will disable that nat port

# for cvuv_vxlan_srcip, cvuv_vxlan_remoteip : empty strings ('') will disable
# the vxlan output port.

# make writable so that next boot can overwrite if need be
chmod ug+w /home/cpacket/boot_config.txt

# cVu-V-k inline
cat <<EOF_BOOTCFG >/home/cpacket/boot_config.txt
{
'vm_mode'                : 'microsoft',
'capture_mode'           : 'cvuv',
'cvuv_mode'              : 'inline',
'cvuv_inline_mode'       : 'tctap',

'cvuv_mirror_eth_0'      : 'eth0',

'cvuv_vxlan_id_0'        : '1337',
'cvuv_vxlan_srcip_0'     : 'YOUR_CVUV_IP',
'cvuv_vxlan_remoteip_0'  : 'REMOTE_TOOL_IP',

'cvuv_vxlan_id_1'        : '1338',
'cvuv_vxlan_srcip_1'     : 'YOUR_CVUV_IP',
'cvuv_vxlan_remoteip_1'  : 'REMOTE_TOOL_IP',

'cvuv_nat_loc_proto_0'   : 'tcp',
'cvuv_nat_loc_ip_0'      : 'YOUR_CVUV_IP',
'cvuv_nat_loc_port_0'    : 'LOCAL_PORT',
'cvuv_nat_dst_ip_0'      : 'NEXT_HOP_IP',
'cvuv_nat_dst_port_0'    : 'NEXT_HOP_PORT',

'burnside_mode'          : False,
'cstor_lite_mode'        : False,
'ssh'                    : {'enabled': True},
}
EOF_BOOTCFG

#

echo "cloud-init ran user-data at: " $(date) >>/home/cpacket/prebootmsg.txt
```

- We save a new version of this file to use for deployment and enter the IPs for cVu-V and the Sensor LB.
- In our example deployment, we only have a single Vectra Sensor running in Azure that our cVu-V needs to send data to, so we removed the 2nd section of 3 lines relating to cvu-v_vxlan from this example before running the "**`az vm create`**" command.

- "**`az vm create`**" command syntax

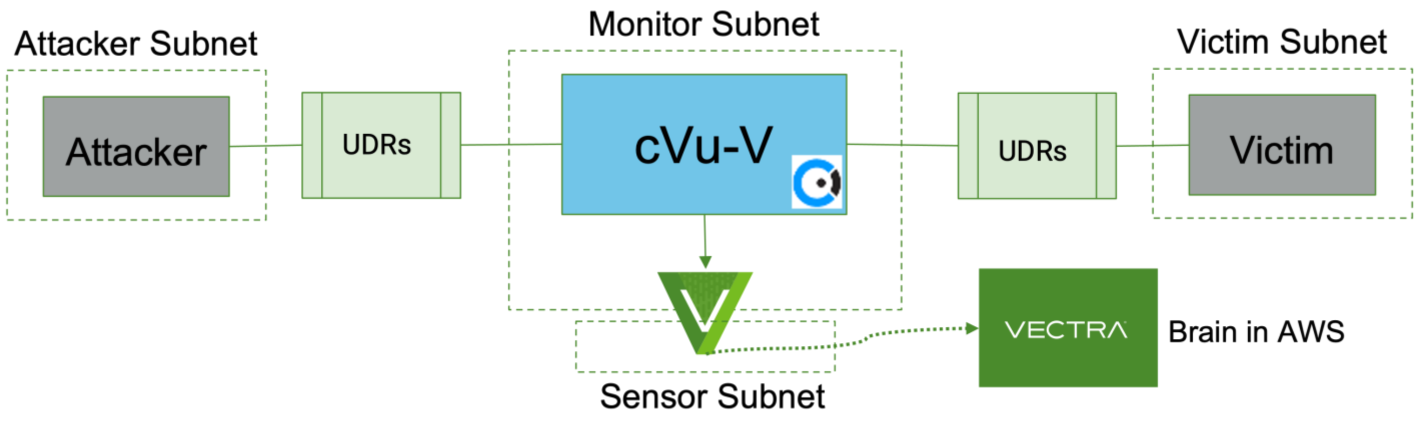
```
az vm create \
--image "$IMAGE" \
--resource-group "$RES_GROUP" \
--name "$NEW_VM_NAME" \
--size "$VM_SIZE" \
--admin-username "$ADMIN_USER" \
--authentication-type ssh \
--ssh-key-values @"$KEYS_FILE" \
--custom-data @"$USERDATA_FILE" \
--boot-diagnostics-storage "$BOOT_DIAGS_STOR" \
--nics "$NIC_LIST" \
--specialized false \
--validate
```

- **image** – cVu-V image location that was shared by cPacket
  - /subscriptions/93004638-8c6b-4e33-ba58-946afd57efdf/resourceGroups/cstor-aidsinga-rg1/providers/Microsoft.Compute/galleries/cpacketccloudpre/images/cvuvinlinepre
- **resource-group** – Azure resource group to deploy into.
- **name** – Name you want the VM to have.
- **size** – VM type (cPacket recommends you choose from Dsv3-series
- **admin-username** – "ubuntu" should be used
- **authentication-type** – ssh should be chosen
- **ssh-key-values** – File where you stored the public key
- **custom-data** – This should be your userdata-cvuv.txt file
- **boot-diagnostics-storage** – Name of the storage account created to store diagnostics
- **nics** – Nics created for the cVu-V VM
- **specialized** – This should be left as false unless directed otherwise by cPacket
- **validate** – Optional parameter to validate that the deployment will be successful before executing

- Example output from a successful deployment:

```
az vm create \
--image "/subscriptions/93004638-8c6b-4e33-ba58-946afd57efdf/resourceGroups/cstor-aidsinga-
rg1/providers/Microsoft.Compute/galleries/cpacketccloudpre/images/cvuvinlinepre" \
--resource-group "cPacket_Test" \
--name "cVu-V_Test" \
--size "Standard_D2s_v3" \
--admin-username "ubuntu" \
--authentication-type ssh \
--ssh-key-values @"./cpacketsshkey.txt" \
--custom-data @"./userdata-cvuv.txt" \
--boot-diagnostics-storage "cpacketstorage" \
--nics "cVu-V_NIC" \
--specialized false
It is recommended to use parameter "--public-ip-sku Standard" to create new VM with
Standard public IP. Please note that the default public IP used for VM creation will be
changed from Basic to Standard in the future.
{
  "fqdns": "",
  "id": "/subscriptions/b3fe75ab-94a2-4322-84af-
016eb01ff43e/resourceGroups/cPacket_Test/providers/Microsoft.Compute/virtualMachines/cVu-
V_Test",
  "location": "eastus",
  "macAddress": "00-0D-3A-15-50-0F",
  "powerState": "VM running",
  "privateIpAddress": "10.0.2.10",
  "publicIpAddress": "",
  "resourceGroup": "cPacket_Test",
  "zones": ""
}
```

## Post Deployment Steps



As a reminder, the above diagram is the example deployment we are discussing in this guide.  We are primarily detailing the setup of cVu-V and the User Defined Routes (UDRs) required to route traffic through the cVu-V so that the Vectra Sensor can monitor the network traffic between the Attacker and Victim subnets.

This guide does not detail the setup of the subnets, the attacker and victim machines, or the Vectra Brain and vSensor.  Please refer to the guides referenced in the Prerequisites section for more detail on deploying the Vectra Cognito Brain and any needed Sensors.

## Configuring User Defined Routes (UDRs)

Microsoft provides some background information and instructions related to UDRs here:

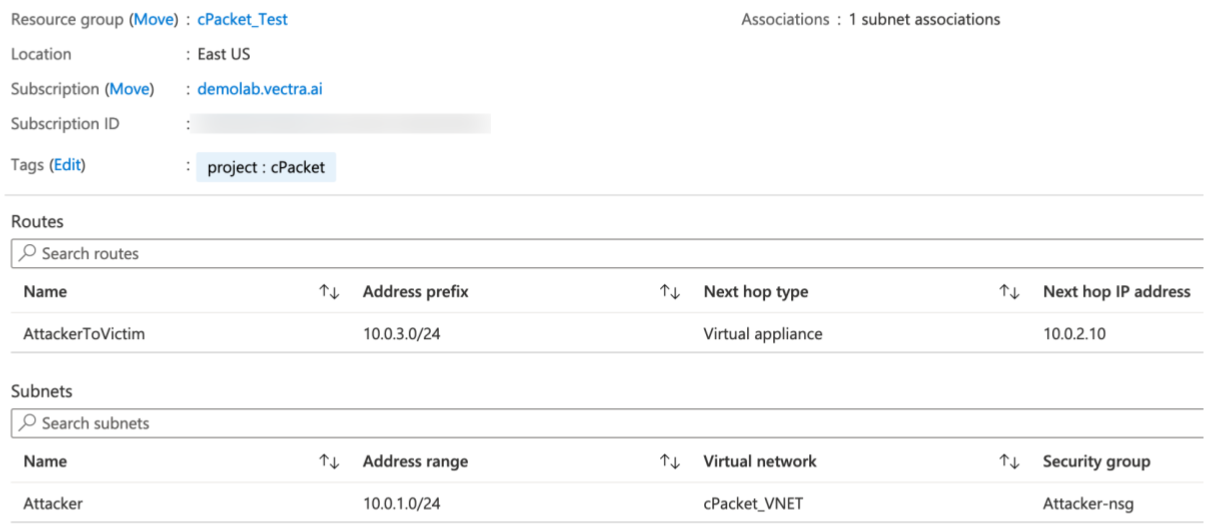
- Virtual network traffic routing: User-defined
- Tutorial: Route network traffic with a route table using the Azure portal

## Subnets in our example

- Vectra_Sensor – 10.0.0.0/24
    - Used for the management interface of the Vectra vSensor.
- Attacker – 10.0.1.0/24
    - Used for the interface of an “Attacker” ubuntu instance.
- Monitor – 10.0.2.0/24
    - Used for the interface of the cVu-V VM.
    - Used for the capture interface of the Vectra vSensor.
- Victim – 10.0.3.0/24
    - Used for the interface of a “Victim” ubuntu instance.

## Steps in our example configuration

- Create a route table and add UDR’s for each of the subnets that we want to monitor traffic to/from (Attacker and Victim subnets) so that traffic will route through the cVu-V VM.
    - Microsoft has a tutorial available here. Example route table is below:



| | |
|---|---|
| Resource group (Move) | : cPacket_Test |
| Location | : East US |
| Subscription (Move) | : demolab.vectra.ai |
| Subscription ID | : |
| Tags (Edit) | : project : cPacket |

Associations : 1 subnet associations

Routes

| Name | Address prefix | Next hop type | Next hop IP address |
|---|---|---|---|
| AttackerToVictim | 10.0.3.0/24 | Virtual appliance | 10.0.2.10 |

Subnets

| Name | Address range | Virtual network | Security group |
|---|---|---|---|
| Attacker | 10.0.1.0/24 | cPacket_VNET | Attacker-nsg |

    - This route table is associated to the “Attacker” subnet and routes all traffic destined for the 10.0.3.0/24 “Victim” subnet to the cVu-V whose NIC has the 10.0.2.10 IP address.
    - A similar route table that is associated to the “Victim” subnet should be constructed that routes all traffic destined for the 10.0.1.0/24 “Attacker” subnet also to the cVu-V VM
    - When adding a route to the route table, the next hop type should be set to “Virtual appliance” for both layer 3 (forcing IP packets to cVu-V using User Defined Routing) and layer 4 (cVu-V behind and Azure Standard Load Balancer) setups.
    - When creating a route table, there is an option called “Propagate gateway routes”.
        - Microsoft has some guidance available here.
        - Of particular note is the section on Border gateway protocol (BGP).
        - It refers to another Microsoft routing article that states:
            - “If you plan to associate the route table to a subnet in a virtual network that's connected to your on-premises network through a VPN gateway, and you don't want to propagate your on-premises routes to the network interfaces in the subnet, set Virtual network gateway route propagation to Disabled.”
        - Some customers may need “Propagate gateway routes” to “Yes” and others may require “No”.
            - Please work with your networking team if you are usure of which option to choose.

## Vectra Supplied UDR Script

Vectra has created a shell script that automates the creation of user defined routes to aid in deployments involving cPacket and Vectra. If you would like a copy of this script, please request it from your Vectra account team.

This script was created by the Vectra Sales Engineering team, and it is not supported by Vectra's support team.

## Lateral Visibility in Subnets

If lateral visibility inside certain subnets is required, it is possible to add individual /32 routes for IPs. This adds complexity to UDR provisioning and upkeep though and many customers choose to keep visibility at a subnet-to-subnet level in the Cloud. You should also keep in mind the service limits Microsoft has for route table entries if pursuing this:

- Networking limits - Azure Resource Manager

## Validating Traffic Flow

### Accessing the cVu-V web management interface

Once you're cVu-V appliance is up and running, you can access the management interface at:

- https://YOUR_VM_IP

Note that a security notice may appear in the browser. This is due to the default SSL cert packaged in the appliance being self-signed.

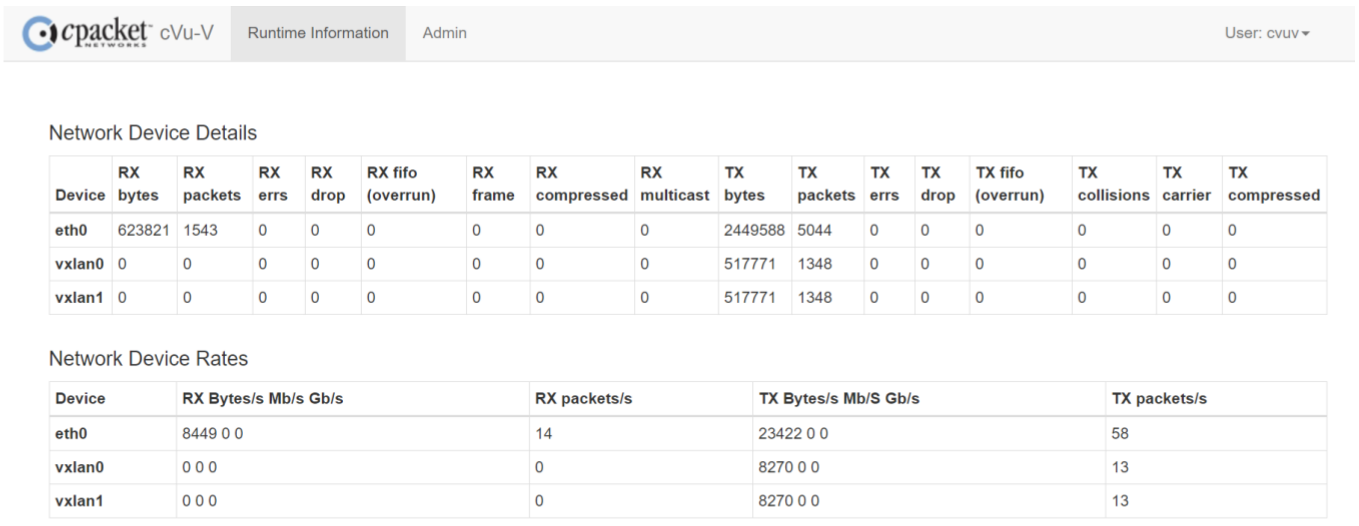
The login screen will prompt you for a username and password which is provided to you by your cPacket technical rep. We advise that the default password be changed after first login for improved security.

- Default username / password as of this documents publishing are: cvuv / cvuvpw

Note that since you started the cVu-V virtual machine with configuration data (userdata.txt above), it should already be handling traffic and tapping/mirroring packets to downstream tools.

### Verifying traffic through cVu-V

To get a quick sense of activity flowing through the cVu-V, review the "Runtime Information" screen:



Network Device Details

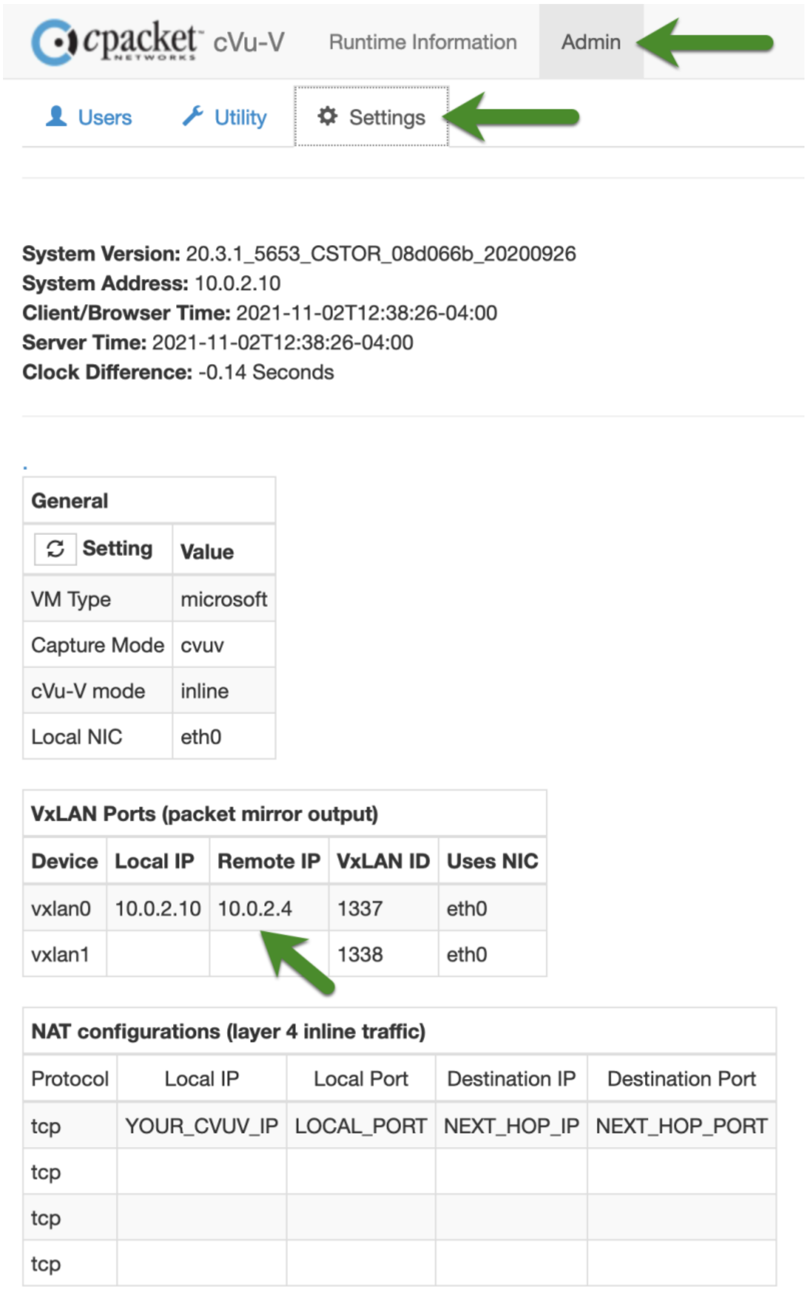
| Device | RX bytes | RX packets | RX errs | RX drop | RX fifo (overrun) | RX frame | RX compressed | RX multicast | TX bytes | TX packets | TX errs | TX drop | TX fifo (overrun) | TX collisions | TX carrier | TX compressed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| eth0 | 623821 | 1543 | 0 | 0 | 0 | 0 | 0 | 0 | 2449588 | 5044 | 0 | 0 | 0 | 0 | 0 | 0 |
| vxlan0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 517771 | 1348 | 0 | 0 | 0 | 0 | 0 | 0 |
| vxlan1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 517771 | 1348 | 0 | 0 | 0 | 0 | 0 | 0 |

Network Device Rates

| Device | RX Bytes/s Mb/s Gb/s | RX packets/s | TX Bytes/s Mb/S Gb/s | TX packets/s |
|---|---|---|---|---|
| eth0 | 8449 0 0 | 14 | 23422 0 0 | 58 |
| vxlan0 | 0 0 0 | 0 | 8270 0 0 | 13 |
| vxlan1 | 0 0 0 | 0 | 8270 0 0 | 13 |

## Verifying network configuration

If you don't see any TX traffic on the vxlan0 interface, check the Admin screen to verify configuration. The Local IP should be the IP of the cVu-V appliance and the Remote IP should be IP address assigned in the Frontend IP configuration of the load balancer that is deployed by default with the Vectra vSensor.



**System Version:** 20.3.1_5653_CSTOR_08d066b_20200926
**System Address:** 10.0.2.10
**Client/Browser Time:** 2021-11-02T12:38:26-04:00
**Server Time:** 2021-11-02T12:38:26-04:00
**Clock Difference:** -0.14 Seconds

**General**

| Setting | Value |
|---|---|
| VM Type | microsoft |
| Capture Mode | cvuv |
| cVu-V mode | inline |
| Local NIC | eth0 |

**VxLAN Ports (packet mirror output)**

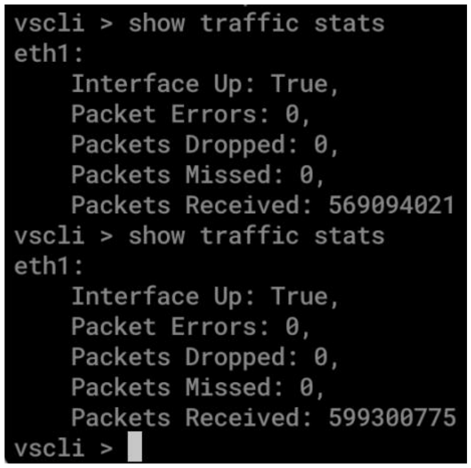
| Device | Local IP | Remote IP | VxLAN ID | Uses NIC |
|---|---|---|---|---|
| vxlan0 | 10.0.2.10 | 10.0.2.4 | 1337 | eth0 |
| vxlan1 | | | 1338 | eth0 |

**NAT configurations (layer 4 inline traffic)**

| Protocol | Local IP | Local Port | Destination IP | Destination Port |
|---|---|---|---|---|
| tcp | YOUR_CVUV_IP | LOCAL_PORT | NEXT_HOP_IP | NEXT_HOP_PORT |
| tcp | | | | |
| tcp | | | | |
| tcp | | | | |

## Validating traffic flow in Vectra

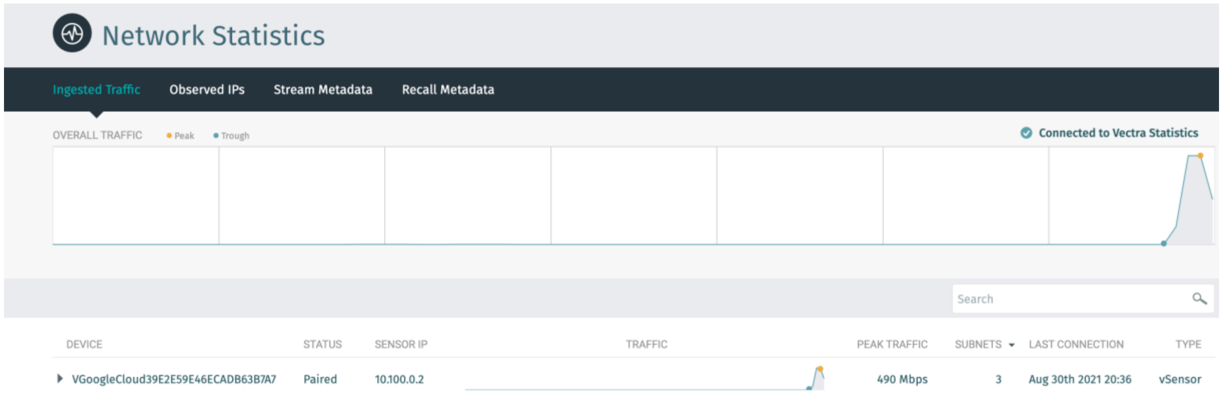
- To see that packets are being receive by the capture interface, use ssh to login to the CLI of the Sensor as the "**`vectra`**" user and use the "**`show traffic stats`**" command.



```
vscli > show traffic stats
eth1:
    Interface Up: True,
    Packet Errors: 0,
    Packets Dropped: 0,
    Packets Missed: 0,
    Packets Received: 569094021
vscli > show traffic stats
eth1:
    Interface Up: True,
    Packet Errors: 0,
    Packets Dropped: 0,
    Packets Missed: 0,
    Packets Received: 599300775
vscli >
```

  - Run the "**`show traffic stats`**" several times to see that packet counts are increasing.

- In the Vectra GUI, if you navigate to *Network Stats > Ingested Traffic*, you can see the traffic graph for your Sensor.
  - For this graph to display, there must be at least 1 Mbps of traffic being captured.
  - Once traffic capture begins, it will take a few minutes for this graph to be populated. Use the CLI of the Sensor as shown above to validate that packets are flowing 1st.



| DEVICE | STATUS | SENSOR IP | TRAFFIC | PEAK TRAFFIC | SUBNETS | LAST CONNECTION | TYPE |
|---|---|---|---|---|---|---|---|
| VGoogleCloud39E2E59E46ECADB63B7A7 | Paired | 10.100.0.2 | | 490 Mbps | 3 | Aug 30th 2021 20:36 | vSensor |

- Please see the following Vectra support article for recommendations on network traffic that should be examined and excluded from analysis:
  - Vectra Platform Network Traffic Recommendations
- After sending traffic to your Sensors, it is a best practice to validate that the traffic observed meets quality standards required for accurate detection and processing. Vectra's Enhanced Network Traffic Validation feature provides alarms and metrics that can be used to validate the quality of your traffic. Please see the following Vectra support article for details:
  - Enhanced Network Traffic Validation (CLI)

## Worldwide Support Contact Information

- Support portal: https://support.vectra.ai
- Email: support@vectra.ai (preferred contact method)
- Additional information: https://www.vectra.ai/support