cPacket cCloud Azure Quick Start Guide

This document will guide the operator through the basic cCloud virtual appliance setup process. This is usually done with the assistance of cPacket technical staff and can be expected to take 30-60 minutes. More complicated routing setups could take on the order of a day to configure and to verify traffic flows. Pre-deployment discussions with cPacket technical staff are helpful to ensure timely cCloud integration into your network topology and achieve desired visibility.

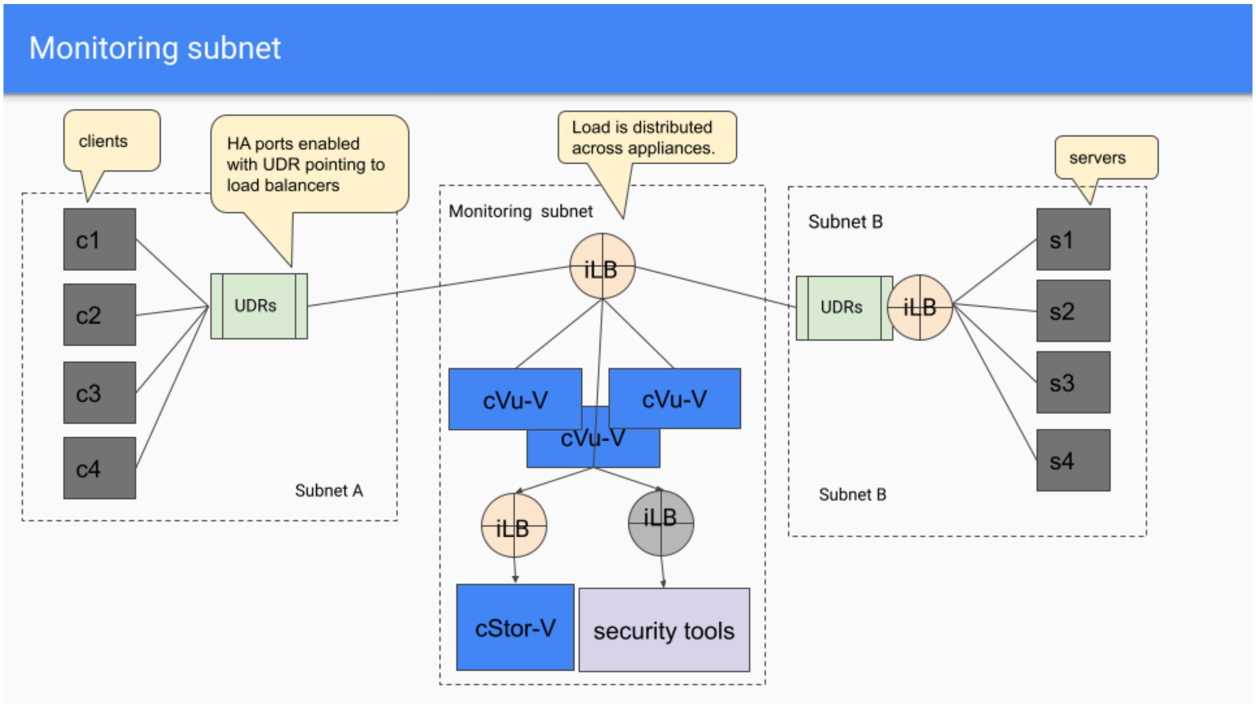
Table of contents

Audience

Deployment to Microsoft Azure requires technical users who:

- have experience with the Azure console, virtual machines, load balancers, and Azure network traffic routing
- are familiar with Bash shell scripting
- are comfortable using SSH

Network topologies

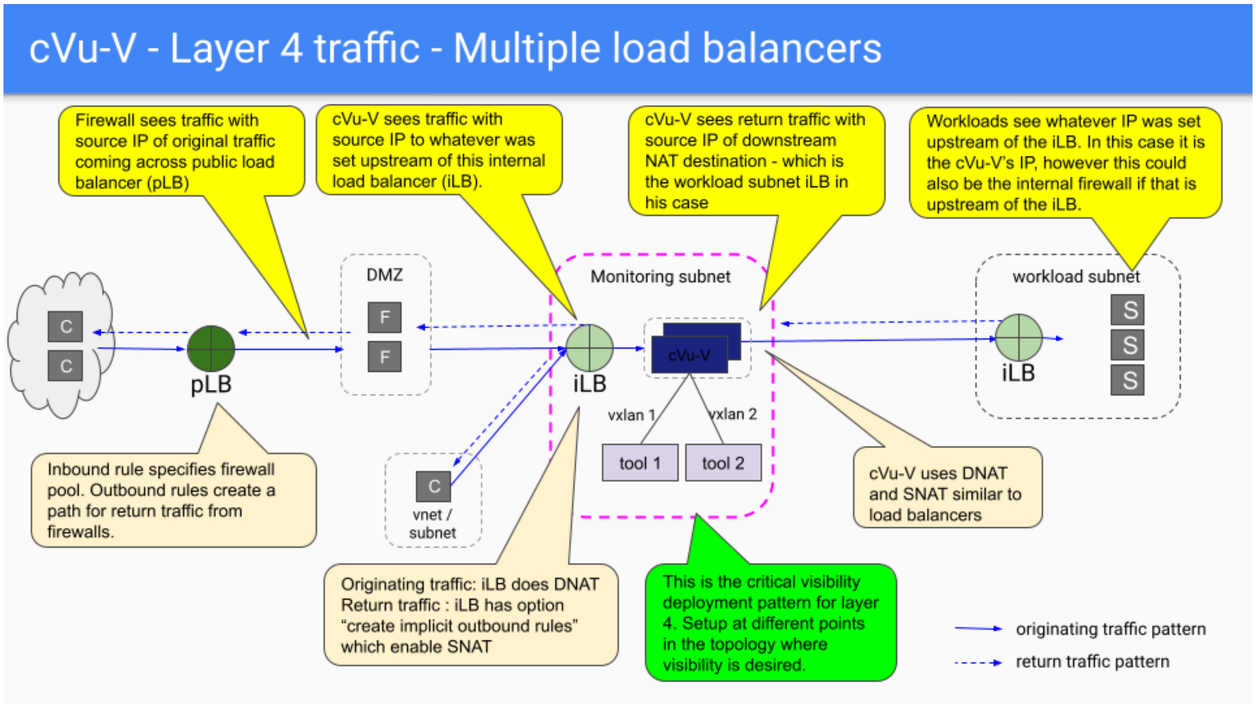


Monitoring subnet

HA ports enabled with UDR pointing to load balancers

Load is distributed across appliances.

Traffic flow for a monitoring subnet configuration:



cVu-V - Layer 4 traffic - Multiple load balancers

Firewall sees traffic with source IP of original traffic coming across public load balancer (pLB)

cVu-V sees traffic with source IP to whatever was set upstream of this internal load balancer (iLB).

cVu-V sees return traffic with source IP of downstream NAT destination - which is the workload subnet iLB in his case

Workloads see whatever IP was set upstream of the iLB. In this case it is the cVu-V's IP, however this could also be the internal firewall if that is upstream of the iLB.

Inbound rule specifies firewall pool. Outbound rules create a path for return traffic from firewalls.

Originating traffic: iLB does DNAT
Return traffic : iLB has option "create implicit outbound rules" which enable SNAT

This is the critical visibility deployment pattern for layer 4. Setup at different points in the topology where visibility is desired.

cVu-V uses DNAT and SNAT similar to load balancers

Prerequisites and setup

- Azure CLI
- cCloud CLI

- Network security group
- cClear Managed Identity and Role Assignment

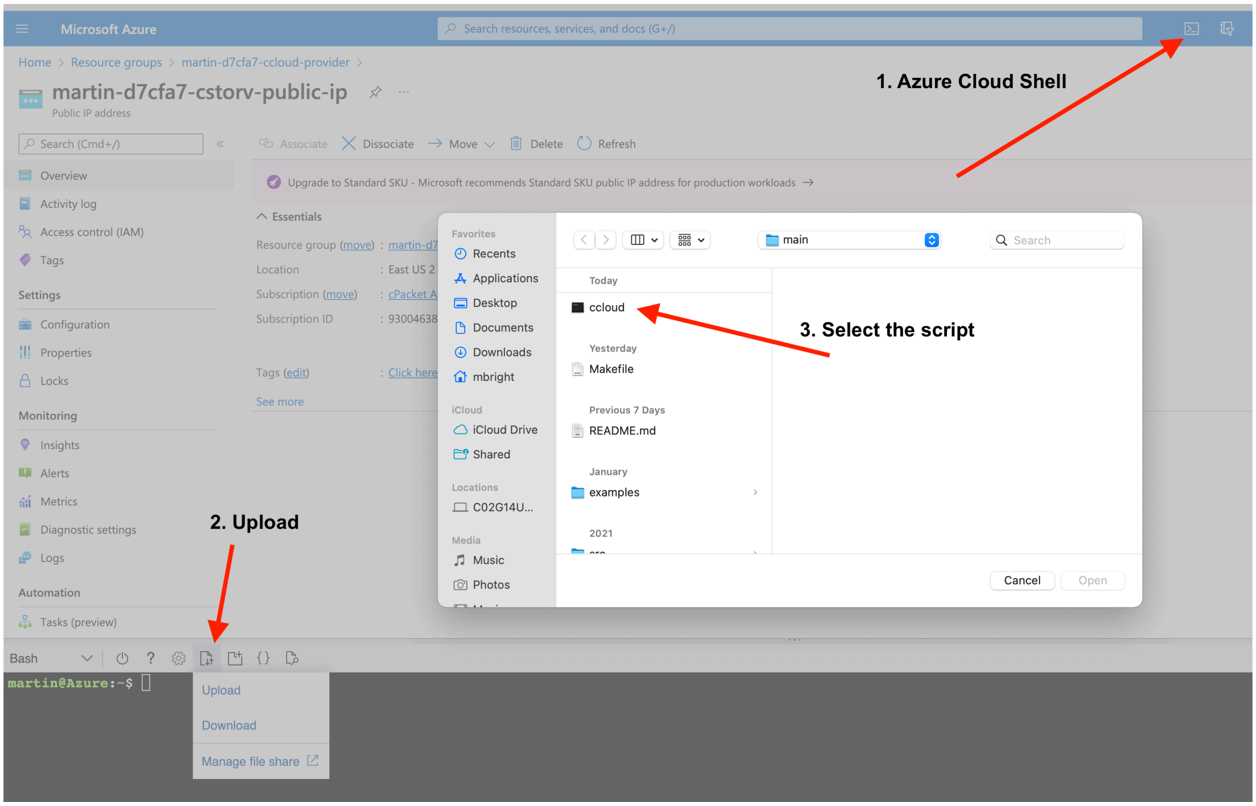
Azure CLI

The Azure CLI must be installed on a host running a modern version of Bash, namely Ubuntu LTS. Alternatively, you may use the Azure Cloud Shell, in which the Azure CLI is already installed.

See the Appendix for assistance logging into Microsoft Azure.

cCloud CLI

The cCloud CLI, while not strictly necessary to deploy cCloud, is handy and saves time with common deployment tasks. The script was written in Bash, with a modern Linux distribution as a target platform. As a baseline environment, we assume the software and utilities available in the Microsoft Azure Cloud Shell. Upload the script to the Azure Cloud Shell and *ensure the script is executable.*



```
martin@Azure:~$
```

```
chmod 755 ccloud
# Examine all Azure subcommands and configuration options with:
./ccloud az --help
```

The current production version of the cCloud CLI is 0.42.3.

```
./ccloud --version
0.42.3
```

Network security group

It is strongly recommended to use a network security group to limit access to the cCloud deployment, especially from the Internet.

The `ccloud az inline` sub-command will create a network security group for you if you supply the `--nsg-cidrs` repeatable flag with CIDR ranges that are allowed access into the network.

```
./ccloud az inline \
    --nsg-cidr="65.132.110.218/32" \
    --nsg-cidr="$another_cidr"
    --resource-group "$resource_group"
```

However, network security groups are often highly customized per installation. They may be created by another organization and supplied to the team that performs cCloud deployment prior to installation.

If deploying cCloud to pre-existing subnets within a vnet, ensure they are correctly protected via a network security group. The `ccloud` CLI will not create a network security group if cCloud is deployed to an existing subnet: it assumes the subnet has been protected appropriately.

If you are manually configuring a network security group, the additional inbound rules should be as follows:

port (service)	source	destination	notes
tcp 22 (ssh)	source networks	VirtualNetwork	shell access to appliances
tcp 443 (https)	source networks	VirtualNetwork	cCloud appliance web UIs
tcp 3389 (rdp)	source networks	VirtualNetwork	RDP to Windows hosts
udp 4789 (vxlan)	source networks	VirtualNetwork	Traffic from cVu-V to cStor-V and 3rd party tools
icmp 22,443,3389,4789 (ping)	source networks	Any	Debugging and diagnostics

cClear-V Managed Identity and Reader Role

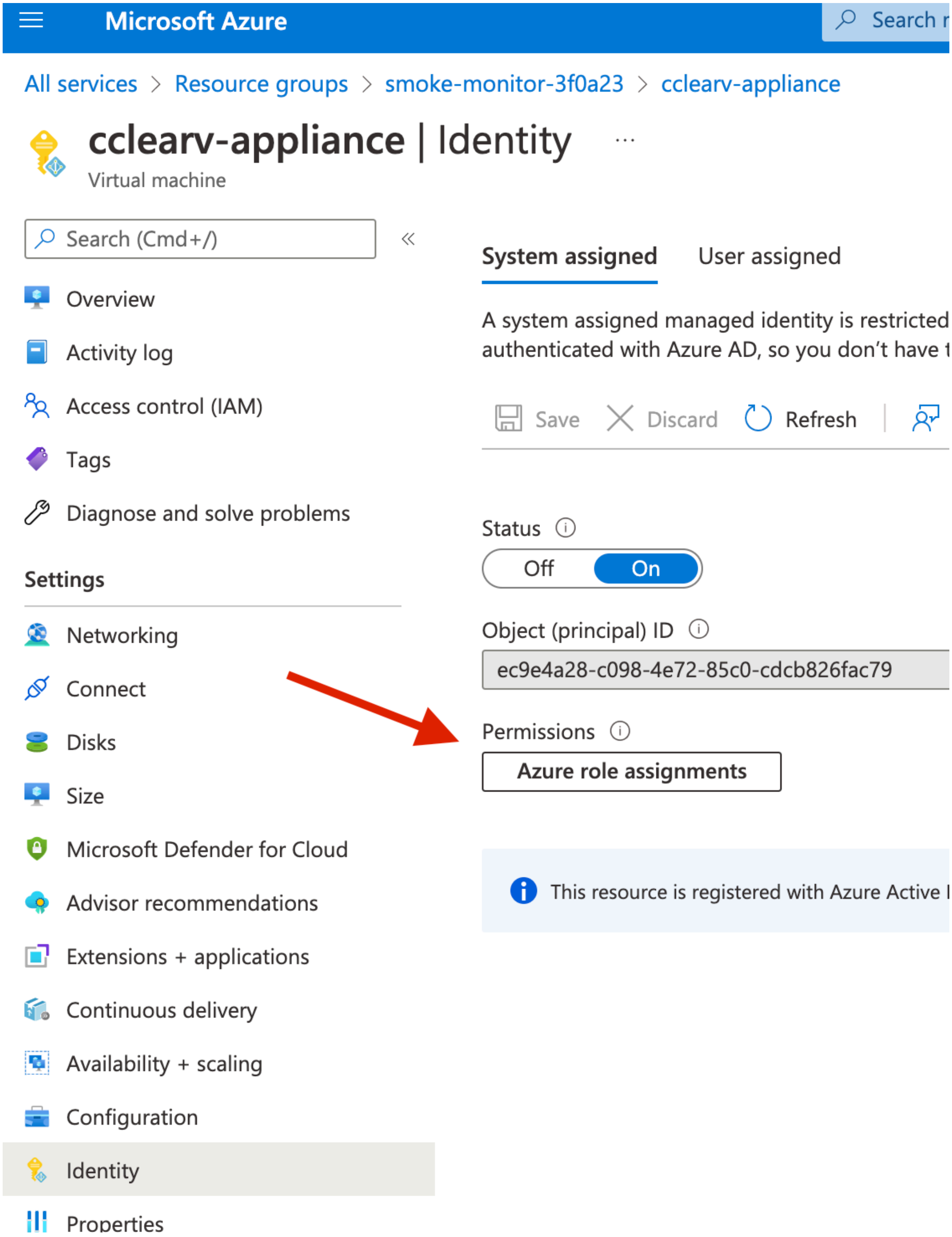
cClear-V actively queries information about deployed cCloud resources in its subscription. In order to achieve this:

1. It must have a managed identity.
2. The managed identity must be assigned the Reader builtin role.
3. The role assignment must be scoped to the subscription.

The cCloud CLI will request a managed identity (1.) when it provisions cClear-V and attempt to assign the Reader role (2.). It will output a warning if it does not succeed.

Note: cClear-V will be operational even if the role assignment fails.

If it is necessary to assign the Reader role manually, edit the Identity settings for the cClear-V virtual machine.



cclearv-appliance | Identity

Virtual machine

A system assigned managed identity is restricted
authenticated with Azure AD, so you don't have t

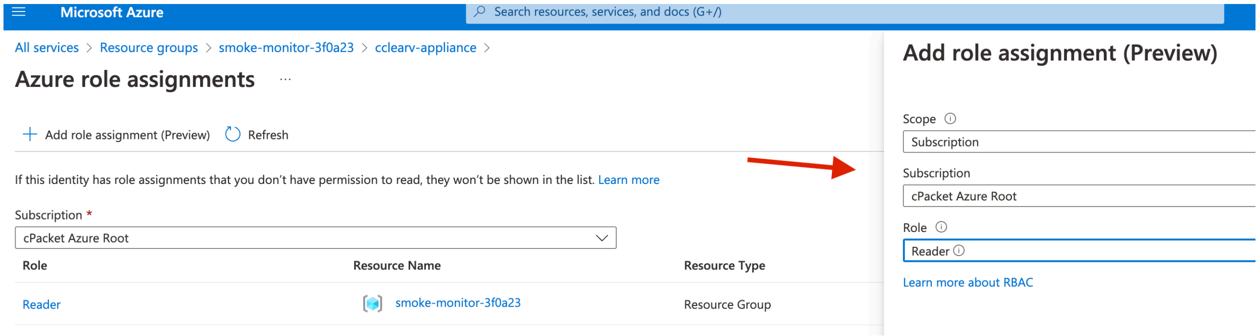
Status: ○ Off ● On

Object (principal) ID: ec9e4a28-c098-4e72-85c0-cdcb826fac79

Permissions

This resource is registered with Azure Active I

Then assign the Reader role scoped to the subscription.


Microsoft Azure
Search resources, services, and docs (G+/)
All services > Resource groups > smoke-monitor-3f0a23 > cclearv-appliance >
+ Add role assignment (Preview)
Refresh
Learn more
Learn more about RBAC

Azure role assignments

If this identity has role assignments that you don't have permission to read, they won't be shown in the list. Learn more

Subscription * cPacket Azure Root

Role	Resource Name	Resource Type
Reader	smoke-monitor-3f0a23	Resource Group

Add role assignment (Preview)

Scope: Subscription

Subscription: cPacket Azure Root

Role: Reader

cCloud images

cCloud is distributed as URLs with Shared Access Secrets, which are then used to generate images.

Accessing shared .vhd images

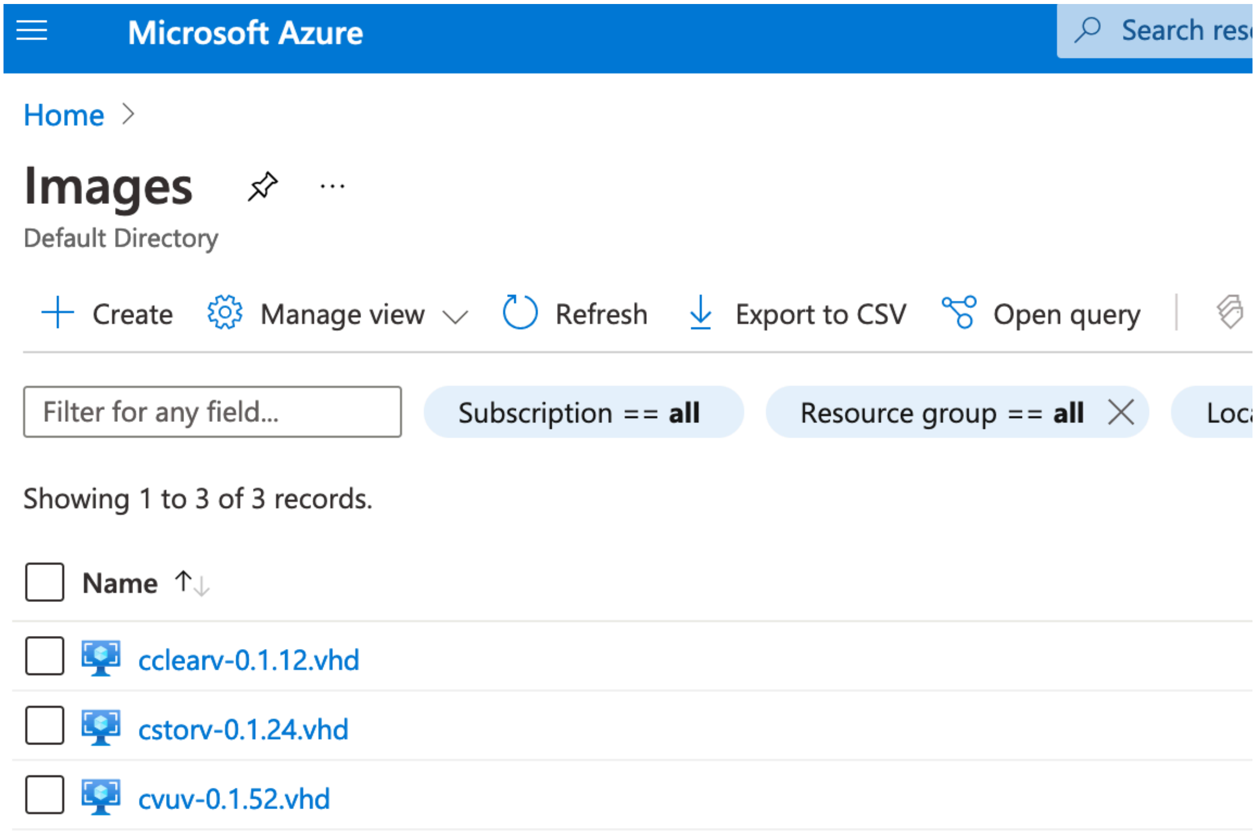
This method requires cPacket to share URLs with Shared Access Signatures (SAS) for each of the three cCloud appliances.

1. cPacket provides the SAS URLs to cCloud `.vhd` files. They will look similar to the following:
2. `cclearv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cclearv-21.1.1.vhd?se=2021-11-13T01%3A05%3A24Z&sp=r&sv=2018-11-09&sr=b&sig=o3HgynXf9GEk2lSYLS1D790vwkVlLFjVO3e5r03xQPA%3D"`
3. `cstorv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cstorv-21.1.1.vhd?se=2021-11-13T01%3A05%3A27Z&sp=r&sv=2018-11-09&sr=b&sig=G9CP%2BLD3TcP7ejwFLy2Rpm8d0CFY7tyPnFeWE3x2KqA%3D"`
4. `cvuv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cvuv-21.1.1.vhd?se=2021-11-13T01%3A05%3A28Z&sp=r&sv=2018-11-09&sr=b&sig=HNt1BR%2BG5s2wzL1QEKph%2Bi4Mmu4ZVrfUfbIuGZpXXpk%3D"`
5. **Running the following commands, and using the URLs supplied by cPacket from step 1., customers or solution engineers create images in the appropriate Azure subscription.**
6. `subscription_id="<your subscription ID>"`
7. `resource_group="<your resource group>"`
8.
9. `# Note: a storage account will be created if it doesn't exist.`
10. `storage_account="ccloudimages"`
11.
12. `cclearv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cclearv-21.1.1.vhd?se=2022-01-18T17%3A41%3A27Z&sp=r&sv=2018-11-09&sr=b&sig=ggG1VcpC%2BTvAmLKmcQlaAHV%2Fsd4TVvnrDwa1KzOxyCE%3D"`
13. `cstorv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cstorv-21.1.1.vhd?se=2022-01-18T17%3A41%3A38Z&sp=r&sv=2018-11-09&sr=b&sig=GLEGWvrxSe6UJmYxYJNeCJ7%2FrRrnGWdH4i02v1nHiVg%3D"`
14. `cvuv_sas_url="https://ccloudvhds.blob.core.windows.net/vhds/cvuv-21.1.1.vhd?se=2022-01-18T17%3A41%3A43Z&sp=r&sv=2018-11-09&sr=b&sig=IIadhU0Xe486AP%2FxD2JLibqosKg7RPvzWs1%2BLOwjBns%3D"`
15.

```
./ccloud az image create -g "$resource_group" -a "$storage_account" --subscription "$subscription_id" "$cclearv_sas_url"
./ccloud az image create -g "$resource_group" -a "$storage_account" --subscription "$subscription_id" "$cstorv_sas_url"
./ccloud az image create -g "$resource_group" -a "$storage_account" --subscription "$subscription_id" "$cvuv_sas_url"

```

After step 2., cCloud images should be available in the subscription.



Images

Default Directory

Filter for any field...

Subscription == **all**

Resource group == **all**

Showing 1 to 3 of 3 records.

Name ↑↓
cclearv-0.1.12.vhd
cstorv-0.1.24.vhd
cvuv-0.1.52.vhd

Installation notes

Load balancers

An Azure standard load balancer is typical for use with cVu-V. We recommend that at least 3 instances of cVu-V be included in the backend pool for a high availability configuration. This should be an “internal standard” load balancer SKU. See the Azure load balancer overview document for more details.

By default, `ccloud az inline` command creates a load balancer and 3 cVu-Vs as part of its deployment. The number of cVu-Vs can be customized.

```
./ccloud az inline \
    --cvuv-count 5 \
    --resource-group "$resource_group"
```

Deploying cCloud

Provisioning

Test/Development

We recommend the DSv5 series of VMs as they provide a variety of network bandwidth options suitable for high throughput packet processing. *This is the default VM type used by the* `ccloud` CLI.

It's very useful to test different network traffic and storage capacity scenarios in a development environment. Varying the number of cVu-V instances or disks attached to cStor-V instances can enable increased throughput.

Production

A sizing/provisioning exercise must be performed to select the correct VMs depending on bandwidth and storage requirements.

Note: Ensure that accelerated networking is used for all network interfaces.

To enable the desired capture rates, the number and type of disks and virtual machines must be selected appropriately.

Please consult the following Microsoft documents to assist in your sizing exercise:

- DSv5 VM type
- Expected throughput

Inline configuration

It is typical to deploy cStor-V, cClear-V, and cVu-V to a monitoring subnet and make traffic flow through the appliances via User Defined Routes (UDR). You can create this configuration via the following command, first creating a 'ccloud-settings.ini' that will contain the resource IDs from the images generated above:

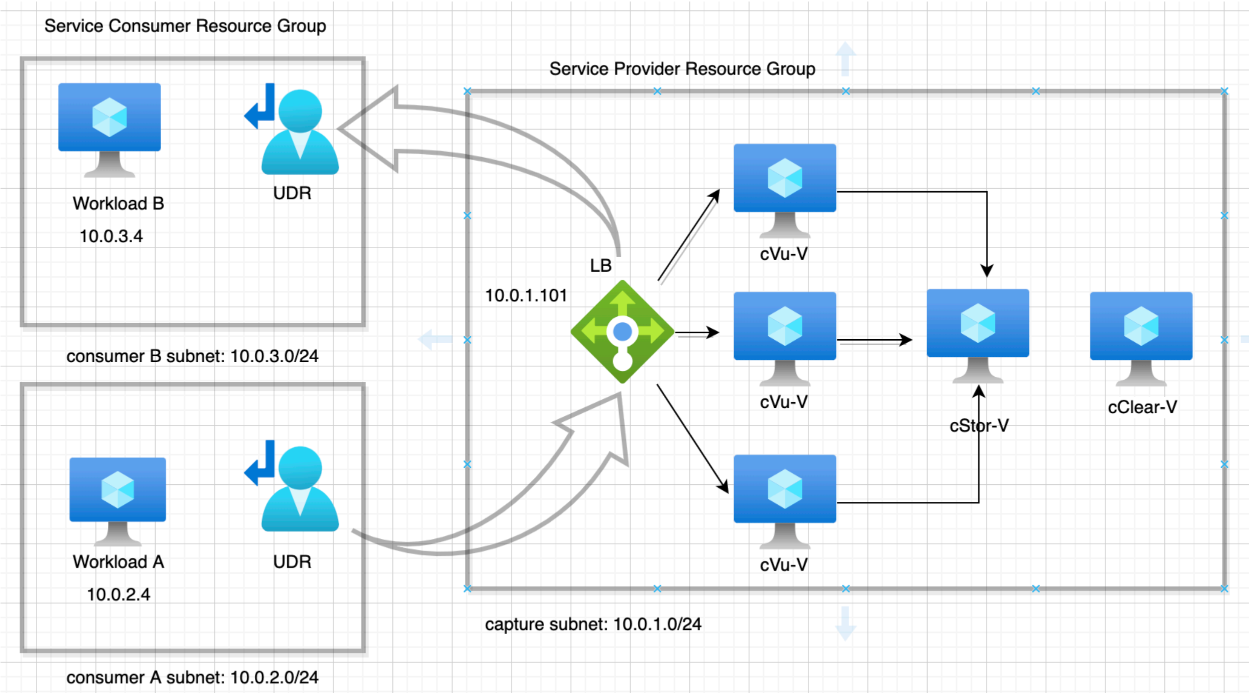
```
#!/bin/bash
set -e

cat >ccloud-settings.ini <<EOF
cclearv_image_rid = /subscriptions/a2fb1277-0845-49e9-8af7-
b7f6d59a70a5/resourceGroups/CLOUD-
BUILDS/providers/Microsoft.Compute/galleries/qa_builds/images/cclear-
v/versions/21.4.29
```

```
cstorv_image_rid = /subscriptions/a2fb1277-0845-49e9-8af7-
b7f6d59a70a5/resourceGroups/CLOUD-
BUILDS/providers/Microsoft.Compute/galleries/qa_builds/images/cstor-
v/versions/21.4.74
cvuv_image_rid = /subscriptions/a2fb1277-0845-49e9-8af7-
b7f6d59a70a5/resourceGroups/CLOUD-
BUILDS/providers/Microsoft.Compute/galleries/qa_builds/images/cvu-
v/versions/21.4.89
EOF

./ccloud az inline --ssh-public-key="$HOME/.ssh/ccloud.pub"
```

This will create the monitoring network ("capture" in the diagram below). The operator must then create the appropriate User Defined Routes.


Service Consumer Resource Group
Workload B
10.0.3.4
UDR
consumer B subnet: 10.0.3.0/24
Workload A
10.0.2.4
UDR
consumer A subnet: 10.0.2.0/24
Service Provider Resource Group
LB
10.0.1.101
cVu-V
cVu-V
cVu-V
cStor-V
cClear-V
capture subnet: 10.0.1.0/24

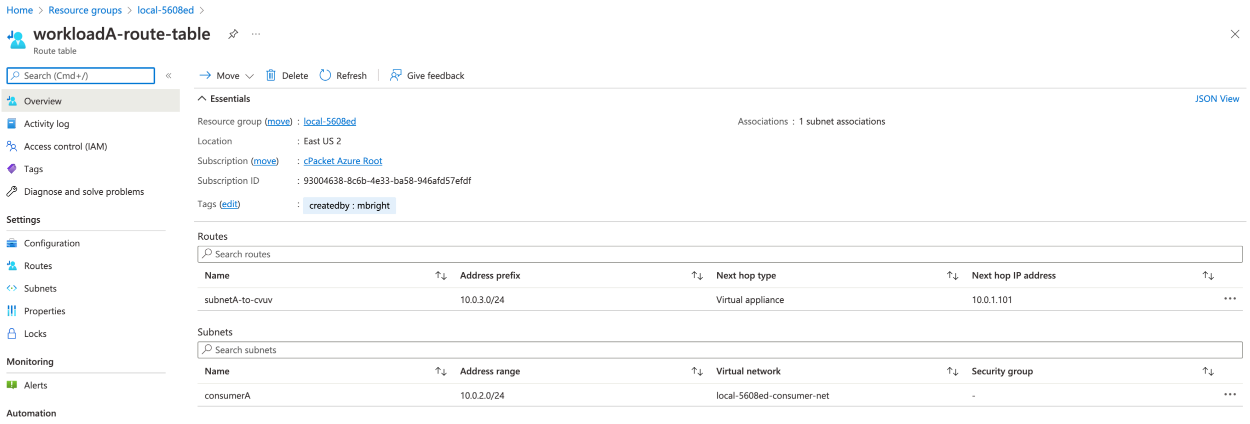
Traffic coming from the 10.0.2.0/24 subnet on the bottom left destined for the 10.0.3.0/24 subnet on the top left is going through a User Defined Route (UDR). That route sets the "Next Hop" to be the load balancer 10.0.1.101.

(Similarly, traffic coming from the 10.0.3.0/24 subnet destined for the 10.0.2.0/24 subnet will flow through a UDR that *also* sets the Next Hop to 10.0.1.101.)

The monitoring subnet (10.0.1.0/24) does not have any user defined routes associated with it.

User Defined Routes

A user defined route configuration from the Azure console will look similar to the following.



workloadA-route-table

Route table

Resource group (move) : local-5608ed

Location : East US 2

Subscription (move) : cPacket Azure Root

Subscription ID : 93004638-8c6b-4e33-ba58-946afd57efdf

Tags (edit) : createdby : mbright

Associations : 1 subnet associations

Routes

Name	Address prefix	Next hop type	Next hop IP address
subnetA-to-cvuv	10.0.3.0/24	Virtual appliance	10.0.1.101

Subnets

Name	Address range	Virtual network	Security group
consumerA	10.0.2.0/24	local-5608ed-consumer-net	-

Downstream Tools

Traffic is tapped/mirrored from cVu-V on a VXLAN interface to downstream tools that capture/store/analyze packet data. With the `ccloud az inline` command, the downstream tool is cStor-V. However, the downstream tool VM can be any IP address that is accessible from the cVu-V. The downstream tool VM and related network security groups must be configured to allow VXLAN traffic (UDP port 4789).

If the IP addresses of the downstream tools are known at deploy time, they can be configured in cVu-V on the command line via the cCloud CLI:

```
./ccloud az inline \
    --resource-group "$resource_group" \
    --additional-tool "10.15.20.3" \
    --additional-tool "10.15.20.4"
```

To learn more about security tools that cPacket works with, please contact your cPacket sales representative.

Verifying Deployment

Once your cVu-V appliance is up and running, you can access the management interface through a web browser:

```
https://YOUR_VM_APPLIANCE_IP
```

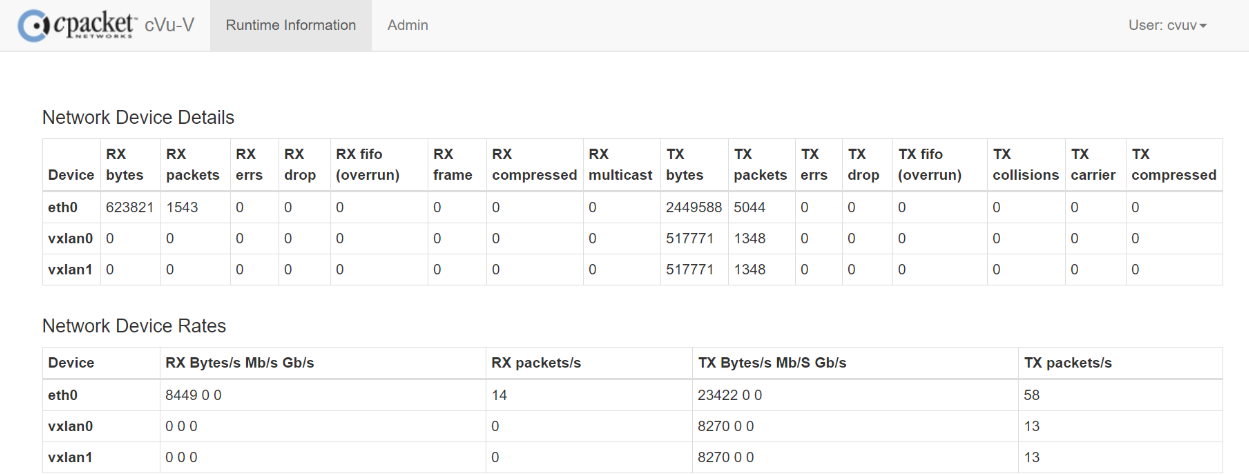
Note that a security notice will appear in the browser. This is due to the default TLS certificate packaged in the appliance being self-signed by cPacket.

The login screen will prompt you for a user name and password which is provided to you by your cPacket technical representative. **It is strongly advised to change the default password after first login.**

The cVu-V virtual machine 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

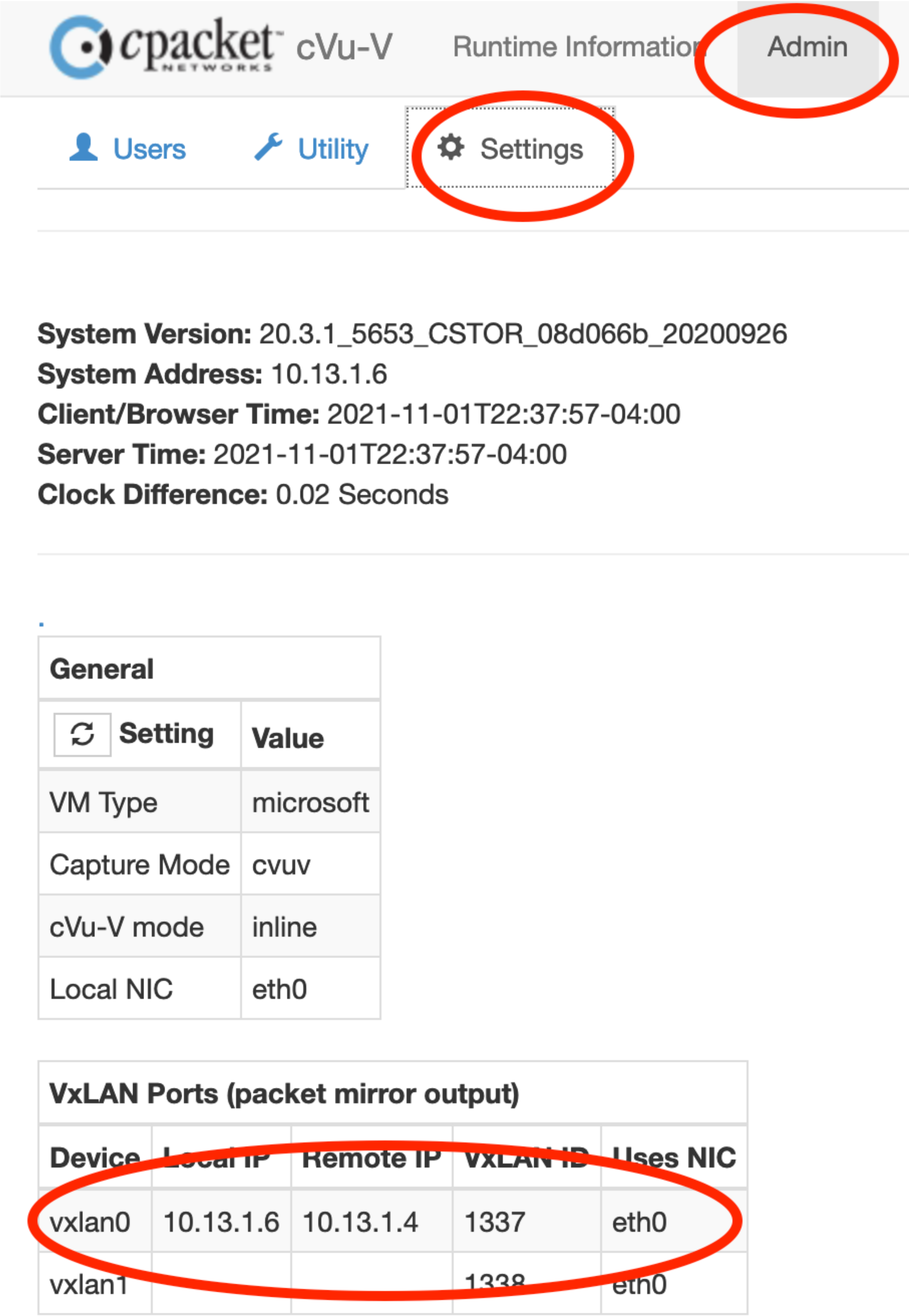
Device	RX bytes	RX packets	RX errs	RX drop	RX fifo (overrun)	RX frame	RX compressed	RX multicast	TX bytes	TX packets	TX errs	TX drop	TX fifo (overrun)	TX collisions	TX carrier	TX compressed
eth0	623821	1543	0	0	0	0	0	0	2449588	5044	0	0	0	0	0	0
vxlan0	0	0	0	0	0	0	0	0	517771	1348	0	0	0	0	0	0
vxlan1	0	0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



System Version: 20.3.1_5653_CSTOR_08d066b_20200926
System Address: 10.13.1.6
Client/Browser Time: 2021-11-01T22:37:57-04:00
Server Time: 2021-11-01T22:37:57-04:00
Clock Difference: 0.02 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	10.13.1.6	10.13.1.4	1337	eth0
vxlan1			1338	eth0

Verifying traffic is received at the downstream tool

The simplest way to verify that traffic is being received at the downstream tool is to run a packet sniffer on that machine, looking at UDP port 4789. The UDP packets being received on port 4789 are VxLan encapsulated and are arriving from the cVu-V appliance.

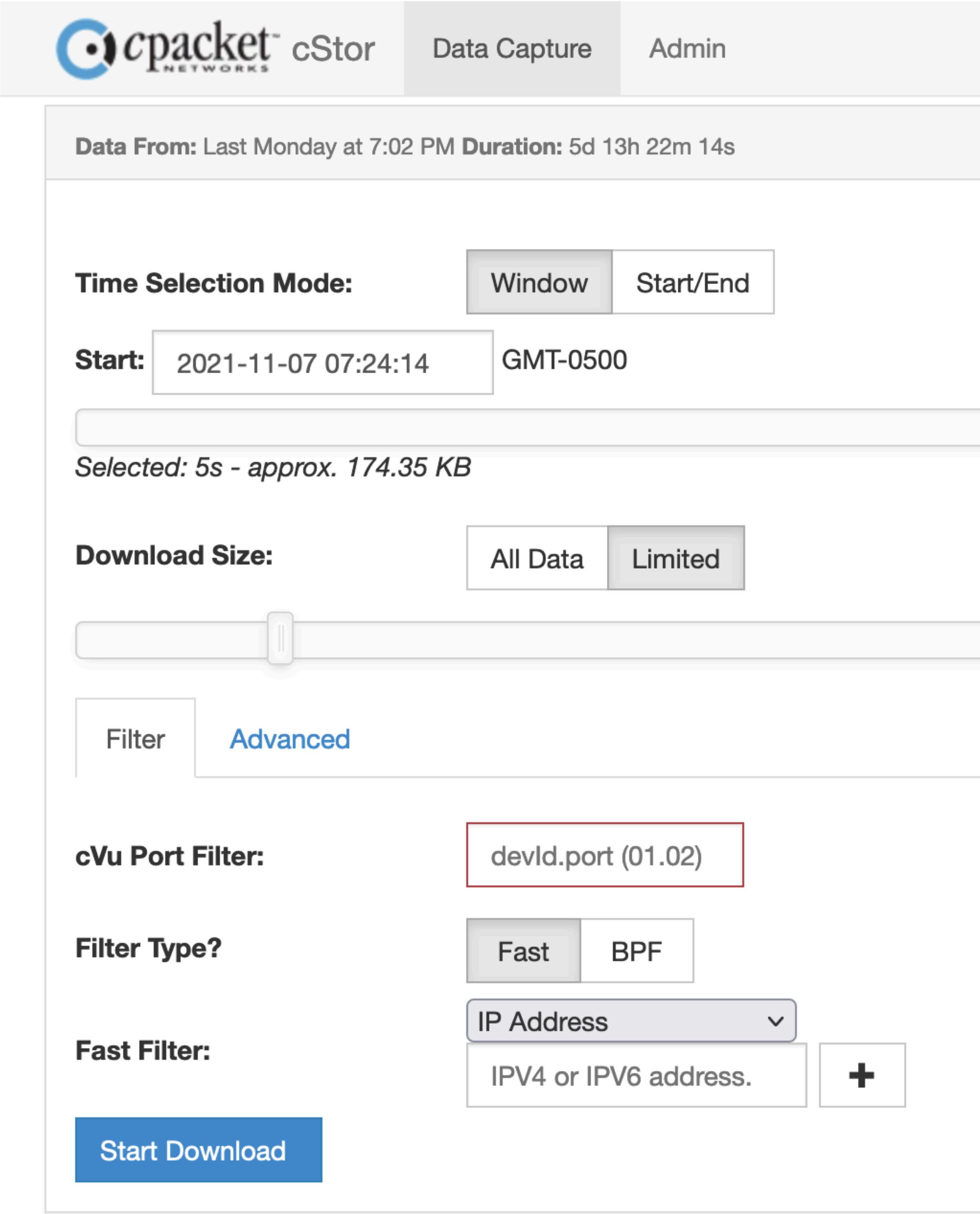
Note that downstream tools must understand how to strip away the VxLan part of the packet to get to the original packet data that was directed to the cVu-V.

On Linux, `tcpdump` can be used on the downstream tool to verify VxLan packets are being received from the cVu-V appliance:

```
sudo tcpdump -n -i eth0 port 4789
```

Verifying traffic to cStor-V

cStor-V is configured downstream from cVu-V. It starts to process traffic on startup, and indicates the rate of capture at the bottom of each page in its web management interface.



Data From: Last Monday at 7:02 PM **Duration:** 5d 13h 22m 14s

Time Selection Mode: [✓ Window] Start/End

Start: 2021-11-07 07:24:14 GMT-0500

Selected: 5s - approx. 174.35 KB

Download Size: All Data [✓ Limited]

cVu Port Filter: devId.port (01.02)

Filter Type? [✓ Fast] BPF

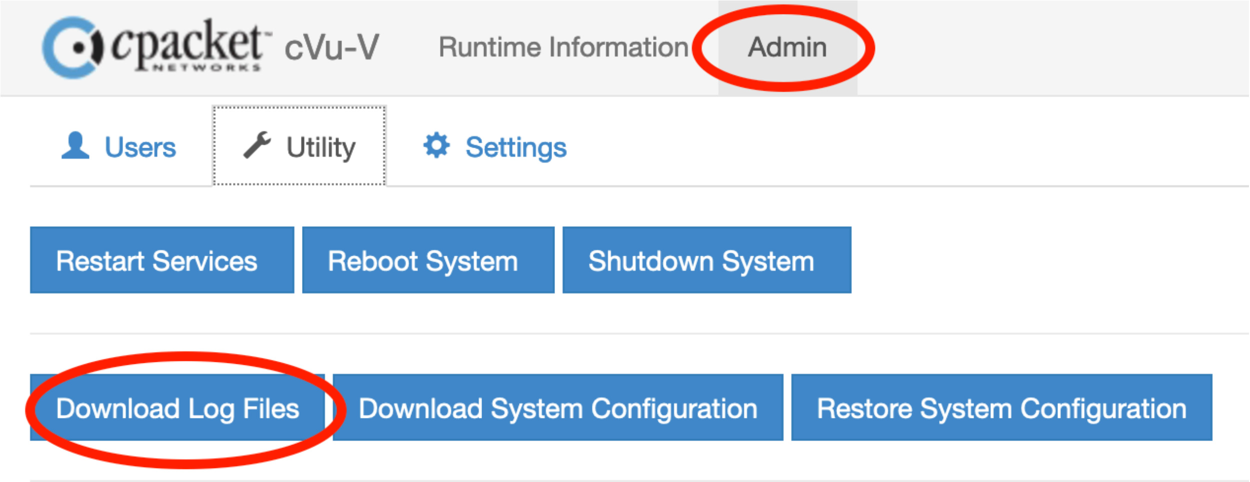
Fast Filter: IP Address

IPV4 or IPV6 address.

Accessing logs for troubleshooting

If technical problems arise, cPacket technical staff will request the logs be downloaded from the cVu-V.

The Utility subtab on the Admin screen contains provides the operator with the ability to download log files.



Synthetic traffic and Demo mode

It is possible to deploy a cCloud monitoring network that accepts synthetic traffic via User Defined Routes. The synthetic traffic is generated in a virtual network created by the following command:

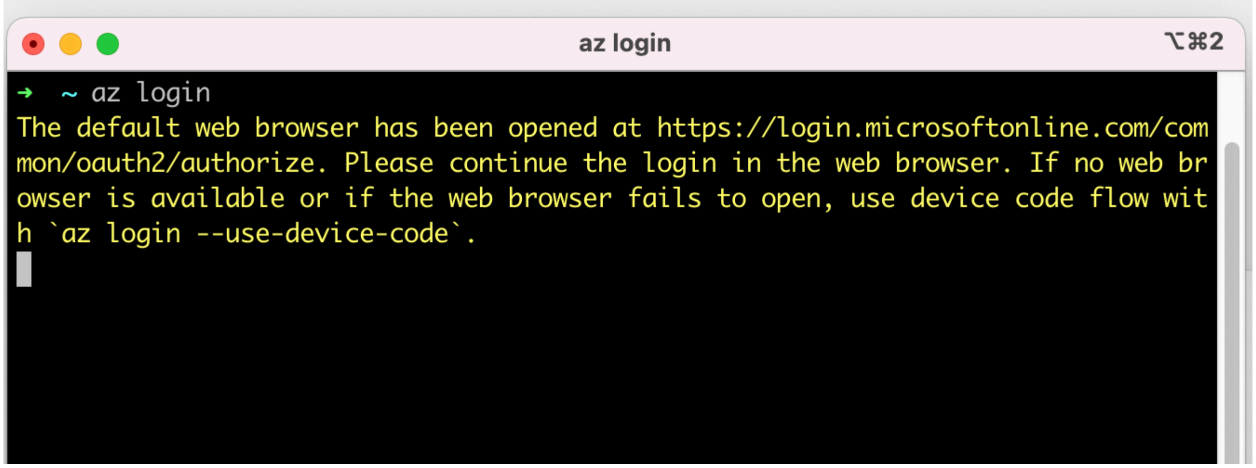
```
./ccloud az inline \
    --deployment-id "$deployment_id" \
    --resource-group "$resource_group" \
    --public-ip-addresses \
    --synthetic-traffic
```

Caution: This deployment creates public IP addresses for the machines that generate synthetic traffic to be reachable via SSH.

Appendix

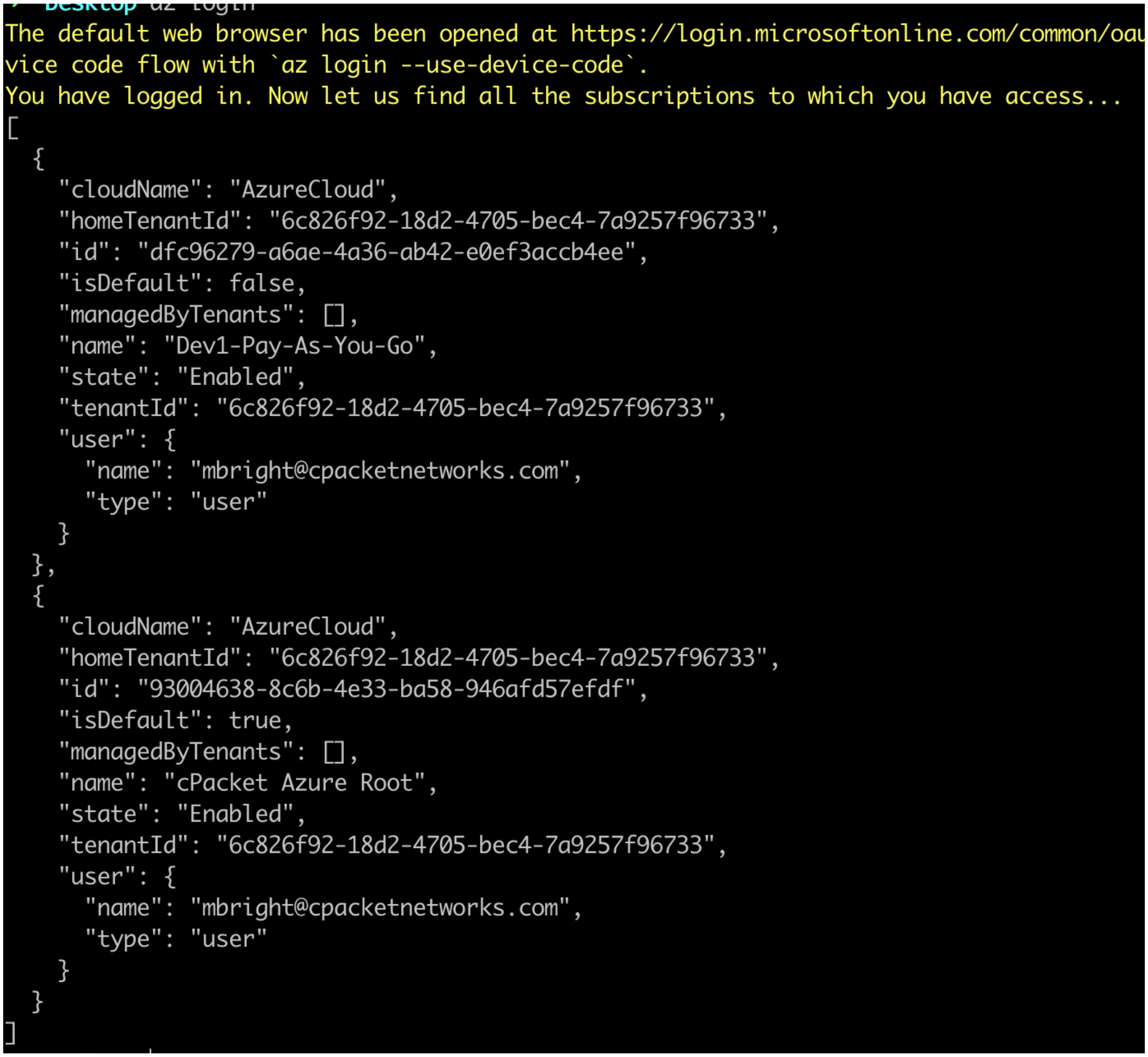
Microsoft Azure sign-in

On a machine with the Azure CLI installed, you can sign in using the Azure CLI. This may force you to verify your credentials by going to a URL in a web browser with a code that the CLI will provide you. Once you've verified your credentials, the CLI will return and you can continue with the next step.



```
➜  ~ az login
The default web browser has been opened at https://login.microsoftonline.com/com
mon/oauth2/authorize. Please continue the login in the web browser. If no web br
owser is available or if the web browser fails to open, use device code flow wit
h `az login --use-device-code`.
```

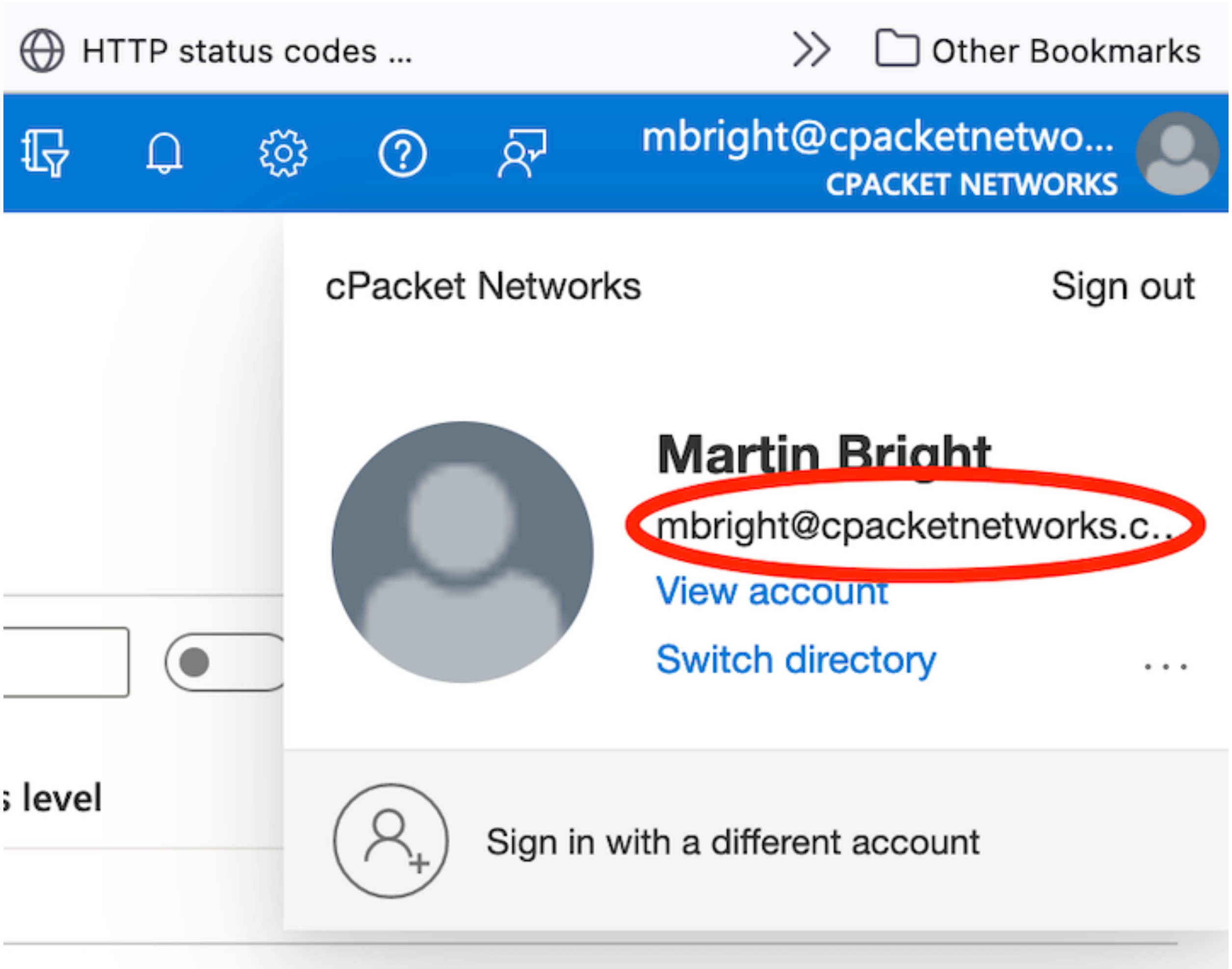
When you return to the shell, the ‘az login’ command will return and list the Azure Subscriptions you are connected to. You may see several subscriptions, including your own, and also the cPacket subscription.



```
➜  Desktop az login
The default web browser has been opened at https://login.microsoftonline.com/common/oau
vice code flow with `az login --use-device-code`.
You have logged in. Now let us find all the subscriptions to which you have access...
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "id": "dfc96279-a6ae-4a36-ab42-e0ef3accb4ee",
    "isDefault": false,
    "managedByTenants": [],
    "name": "Dev1-Pay-As-You-Go",
    "state": "Enabled",
    "tenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "user": {
      "name": "mbright@cpacketnetworks.com",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "id": "93004638-8c6b-4e33-ba58-946afd57efdf",
    "isDefault": true,
    "managedByTenants": [],
    "name": "cPacket Azure Root",
    "state": "Enabled",
    "tenantId": "6c826f92-18d2-4705-bec4-7a9257f96733",
    "user": {
      "name": "mbright@cpacketnetworks.com",
      "type": "user"
    }
  }
]
```

Account email

When logged into the Azure console, you can find your account email as indicated in the screen shot below:



Updating cVu-V appliance TLS certificates

Currently, updating the TLS certificates can not be performed from the web management interface: it must be done via the command line.

In the procedure below, 10.1.2.3 is the private IP address of the cVu-V appliance.

1. Copy the trusted CA signed certificates to the cVu-V instance.

   ```
   rsync -e "ssh -i $HOME/.ssh/ccloud-key" TrustedCASigned.{crt,key}
   ubuntu@10.1.2.3
   ```

2. SSH to the cVu-V instance.

   ```
   ssh -i $HOME/.ssh/ccloud-key ubuntu@10.1.2.3
   ```

3. Become the root user.

   ```
   sudo su -l
   ```

4. Locate the `nginx` configuration and self-signed certificates.

```
cd /home/cpacket/packages/cstordep/conf/nginx
```

5. Backup the self-signed certificate and key.

```
6. cp spifee_ssl.crt spifee_ssl.crt.1
   cp spifee_ssl.key spifee_ssl.key.1
```

7. Copy the new key and cert on top of the old ones.

```
8. cp /home/ubuntu/TrustedCASigned.key spifee_ssl.key
9. cp /home/ubuntu/TrustedCASigned.crt spifee_ssl.crt
   chown root:root spifee_ssl.*
```

10. Reload nginx.

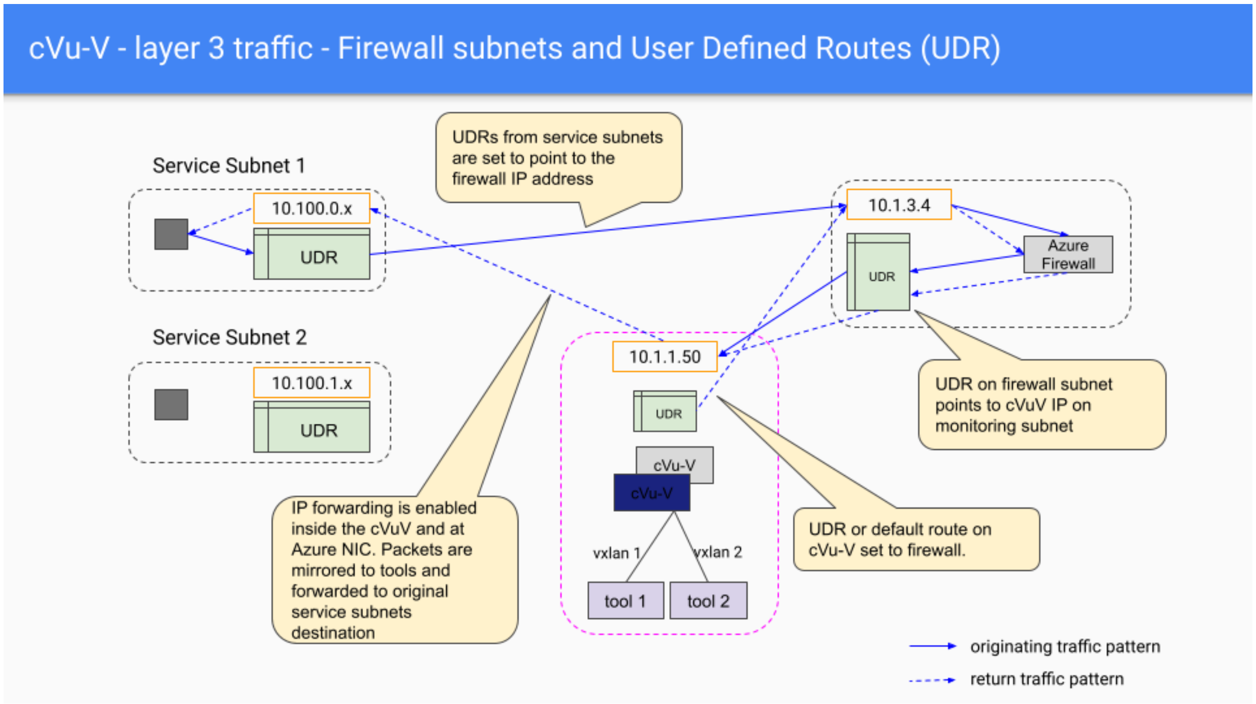
```
systemctl reload nginx
```

11. Verify that the new certificate is being served. This can be done from another machine.

```
12.  curl -L https://10.1.2.3 # This should return a web page, not an error
   echo | openssl s_client -showcerts -servername 10.12.5.6 -connect
   10.12.5.6:443 2>/dev/null | openssl x509 -inform pem -noout -text | tee
   /tmp/new-cert-details.txt
```

Integrating 3rd party firewalls

Some configurations with user defined routes might include firewall subnets.



cVu-V - layer 3 traffic - Firewall subnets and User Defined Routes (UDR)

UDRs from service subnets are set to point to the firewall IP address

IP forwarding is enabled inside the cVuV and at Azure NIC. Packets are mirrored to tools and forwarded to original service subnets destination

UDR on firewall subnet points to cVuV IP on monitoring subnet

UDR or default route on cVu-V set to firewall.

Deploying individual appliances

The following `ccloud` CLI examples deploy the appliances individually into a specified vnet and subnet.

It is practical to deploy cStor-V and any security and/or 3rd party tools before cVu-V so that the cStor-V private IP can be supplied to the cVu-V configuration. (Traffic flows from cVu-V into cStor-V).

cStor-V

The following snippet deploys cStor-V to a specified subnet within an existing vnet. If the subnet doesn't exist, an attempt is made to create it.

As a result of the command, a timestamped `userdata-cstorv.sh` file is created in the current directory. This is supplied to the VM as a custom init script file.

```
#!/usr/bin/env bash
set -e
set -o pipefail

cstorv_name="cstorv-standalone"
cstorv_image_rid="/subscriptions/93004638-8c6b-4e33-ba58-
946afd57efdf/resourceGroups/cloud-
builds/providers/Microsoft.Compute/galleries/dev_builds/images/cstor-
v/versions/21.4.74"
resource_group="smoke-monitor-3f0a23"
nic="cstorv-standalone"

./ccloud az cstorv \
    --name "$cstorv_name" \
    --image "$cstorv_image_rid" \
    --resource-group "$resource_group" \
    --nic "$nic" \
    -k "smoke-monitor-3f0a23.pub"
```

It is possible to supply a hand crafted `userdata-cstorv.sh` file with `--user-data-file` flag.

Security appliances

Similarly to cStor-V, if you're using a third party security appliance or other downstream tool, you should install it before cVu-V, in order to supply its IP address to cVu-V.

Additional tools are specified via the --additional-tools flag to the ccloud az cvuv command.

cVu-V

The following snippet deploys cVu-V.

In this example, the private IP of the cStor-V appliance is supplied so that traffic from cVu-V is directed to cStor-V. Additional downstream tools are also specified by IP address.

```
#!/usr/bin/env bash
set -e

# Deploy a single cVu-V appliance to an existing vnet and subnet
cvuv_image_rid="/subscriptions/93004638-8c6b-4e33-ba58-
946afd57efdf/resourceGroups/cloud-
builds/providers/Microsoft.Compute/galleries/dev_builds/images/cvu-
v/versions/21.4.137"

./ccloud az cvuv \
    --name 'cvuv-standalone' \
    --vnet consumer-net \
    --subnet capture \
    --image "$cvuv_image_rid" \
    --vm-type "Standard_D8s_v3" \
    --resource-group "smoke-monitor-3f0a23" \
    -k "smoke-monitor-3f0a23.pub"
```

cClear-V

The following snippet deploys cClear-V using a specific NIC.

```
#!/bin/bash

azure_cclearv_resource_id="/subscriptions/93004638-8c6b-4e33-ba58-
946afd57efdf/resourceGroups/cloud-
builds/providers/Microsoft.Compute/galleries/dev_builds/images/cclear-
v/versions/21.4.37"
nic="cclearv-standalone"
resource_group="smoke-monitor-3f0a23"
cclear_name="cclearv-standalone"

./ccloud az cclearv \
    --nic "$nic" \
    --ssh-public-key "smoke-monitor-3f0a23.pub" \
    --image "$azure_cclearv_resource_id" \
    --name "$cclear_name" \
    -g "$resource_group"
```

User data files

It’s possible to provide custom initialization scripts to the cCloud appliance. By default, these are created by the cCloud CLI and supplied to the Azure vm creation commands. It’s possible to override them with customized versions which can be saved to version control.

Default cStor-V user data file

```
#!/bin/bash

chmod ug+w /home/cpacket/boot_config.txt

# cVu-V-k inline boot config settings
cat >/home/cpacket/boot_config.txt <<EOF_BOOTCFG
```

```
{
    'vm_type': 'azure',
    'capture_mode': 'libpcap',
    'decap_mode': 'vxlan',
    'num_pcap_bufs': 2,
    'capture_nic_index': 0,
    'pci_whitelist': '0001:00:02.0',
    'eth_dev' : 'eth0',
    'core_mask': '0x3',
    'burnside_mode': False,
    'cstor_lite_mode': False,
    'ssh': {'enabled': True},
    'cleanup_threshold' : 50,
    'use_compression' : False,
    'tiered_stor_en': False
}
EOF_BOOTCFG

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

Example cVu-V user data file

```
#!/bin/bash

# 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

cat >/home/cpacket/boot_config.txt <<EOF_BOOTCFG
{
    'vm_type': 'azure',
    'capture_mode': 'cvuv',
    'cvuv_mode': 'inline',
    'cvuv_inline_mode': 'tctap',
    'cvuv_mirror_eth_0': 'eth0',
    'cvuv_vxlan_id_0': '1337',
    'cvuv_vxlan_srcip_0': '10.0.1.6',
    'cvuv_vxlan_remoteip_0': '10.0.1.4',
    'cvuv_nat_loc_proto_0': 'tcp',
    'cvuv_nat_loc_ip_0': '',
    'cvuv_nat_loc_port_0': '',
    'cvuv_nat_dst_ip_0': '',
    'cvuv_nat_dst_port_0': '',
    'burnside_mode': False,
    'cstor_lite_mode': False,
    'ssh': {'enabled': True},
    'capture_nic_eth': 'eth0',
    'capture_nic_ip': '',
    'management_nic_eth': '',
    'management_nic_ip': '',
    'management_nic_gw': '',
    'management_dest': '',
    'management_dest_netmask': ''
}
EOF_BOOTCFG

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

cCloud CLI on MacOS

Using the cCloud CLI on MacOS requires installing some dependencies. These are conveniently retrieved with Homebrew.

- An updated version of Bash.
- The GNU version of the `date` utility.

```
brew install bash
brew install coreutils # This installs GNU date
```