Container4NFV

Release Latest

Nov 21, 2018

Contents

CHAPTER 1

Container4NFV Gap Analysis

Project Container4NFV, https://wiki.opnfv.org/display/OpenRetriever/Container4NFV

Editors Xuan Jia (China Mobile), Gergely Csatari (Nokia)

Authors Container4NFV team

Abstract This document provides the users with top-down gap analysis regarding OpenRetriever feature requirements with OPNFV Installer, OpenStack Official Release and Kubernetes Official Release.

1.1 Container4NFV architecture options

Analyzis of the architecture options were moved to the Container4NFV wiki.

1.2 Container4NFV Gap Analysis with OPNFV Installer

This section provides users with Container4NFV gap analysis regarding feature requirement with OPNFV Installer in Danube Official Release. The following table lists the use cases / feature requirements of container integrated functionality, and its gap analysis with OPNFV Installer in Danube Official Release. OPNFV installer should support them.

Use Case / Requirement	Supported in Danube	Notes
Use Openstack Magnum to install container environment	No	Magnum is supported in Openstack Official Re
Use Openstack Ironic to supervise bare metal machine	No	Container could be installed in bare metal mac
Use Openstack Kuryr to provide network for container	No	Container has its own network solution. Contai

1.3 Container4NFV Gap Analysis with OpenStack

This section provides a gap analyzis between the targets of Container4NFV for release Euphrates (E) or later and the features provided by OpenStack in release Ocata. As the OPNFV and OpenStack releases tend to change over time this analyzis is planned to be countinously updated. During the analyzis all OpenStack projects considered.

(**Editors note:** Maybe we should define a scope of OpenStack projects which is considered. All OpenStack projects can mean anything.)

The following table lists the use cases / feature requirements of container integrated functionality, and its gap analysis with OpenStack.

Use Case / Requirement	Related OpenStack project	Notes
Manage container and virtual machine lifecycle with the same NB API	Zun or nova-docker driver	Magnum can deploy a Con
Container private registry to store container images	Swift, Cinder, Glance, Glare	Container images need a st
Kuryr needs to support MACVLAN and IPVLAN	Kuryr	Using MACVLAN or IPVI
Kuryr Kubernetes integration is needed	Kuryr	It is done in the frame of C
HA support for Kuryr	Kuryr	
HA support for Zun	Zun	

1.4 Container4NFV Gap Analysis with Kubernetes v1.5

This section provides users with Container4NFV gap analysis regarding feature requirement with Kubernetes Official Release. The following table lists the use cases / feature requirements of container integrated functionality, and its gap analysis with Kubernetes Official Release.

Use Case / Requirement	Supported in v1.5	Notes
Manage conainter and virtual machine in the same platform.	No	There are some ways how Kubernetes could manage VM-s: 1. Kubevirt 2. Kubernetes can start rkt and with rkt it is possible to start VM-s 3. Virtlet 4. Hypercontainer
Kubernetes support multiple networks.	No	As VNF needs at least three interfaces. Management,control plane, data plane. CNI already supports multiple interfaces in the API definition. 1. Multus 2. CNI-Genie 3. A solution built into Kubernetes
Kubernetes support NAT-less connections to a container	No	SIP/SDP and SCTP are not working with NAT-ed networks
Kubernetes scheduling support CPU bindingNUMA features	No	The kubernetes schedular don't support these features

Continued on next page

Table 3 – continued from previous page

Use Case / Requirement	Supported in v1.5	Notes
DPDK need to support CNI	No	DPDK is the technology to accelerate the data plane. Container need support it, the same with virtual machine.
SR-IOV can support CNI (Optional)	No	SR-IOV could let container get high performance

CHAPTER 2

Container4NFV Release Notes

2.1 Container4NFV E release Notes

1. Gap analysis for openstack,kubernetes,opnfv installer
2. Container architecture options
3. Joid could support Kubernetes
4. Using vagrant tool to setup an env with DPDK enabled.

2.2 Container4NFV F release Notes

1. Enable Multus in Kubernetes
2. Enable SR-IOV in Kubernetes
3. Support ARM platform

CHAPTER 3

Container4NFV User Guide

3.1 Installation

This quickstart shows you how to easily install a Kubernetes cluster on VMs running with Vagrant. You can find the four projects inside *container4nfv/src/vagrant* and their documentation: - kubeadm_basic: weave.rst - kubeadm_multus: multus.rst - kubeadm_ovsdpdk: ovs-dpdk.rst - kubeadm_virtlet: virtlet.rst

Vagrant is installed in Ubuntu 16.04 64bit. vagrant is to create kubernetes cluster using kubeadm. kubernetes installation by kubeadm can be refered to https://kubernetes.io/docs/getting-started-guides/kubeadm.

3.2 e release

3.2.1 Vagrant Setup

sudo apt-get install -y virtualbox wget –no-check-certificate https://releases.hashicorp.com/vagrant/1.8.7/vagrant_1.8.7_x86_64.deb sudo dpkg -i vagrant_1.8.7_x86_64.deb

3.2.2 K8s Setup

git clone http://gerrit.opnfv.org/gerrit/container4nfv -b stable/euphrates cd container4nfv/src/vagrant/k8s_kubeadm/ vagrant up

3.2.3 Run K8s Example

vagrant ssh master -c “kubectl apply -f /vagrant/examples/virtio-user.yaml”

3.2.4 K8s Cleanup

vagrant destroy -f

3.3 f release

3.3.1 Vagrant Setup

1. *setup_vagrant.sh* may install all for you. The project uses vagrant with libvirt as default because of performance.

```
` container4nfv/src/vagrant# ./setup_vagrant.sh `
```

Consequently, we need to reboot to make libvirtd group effective.

2. Deploy:

To test all the projects inside *vagrant/* just run the next script:

```
` container4nfv/ci# ./deploy.sh `
```

3.4 Senario:

3.4.1 k8-nosdn-nofeature-noha

Using Joid to deploy Kubernetes in bare metal machine https://build.opnfv.org/ci/job/joid-k8-nosdn-nofeature-noha-baremetal-daily-euphrates/lastBuild/

3.4.2 k8-nosdn-lb-noha

Using Joid to deploy Kubernetes in bare metal machine with load balance enabled https://build.opnfv.org/ci/job/joid-k8-nosdn-lb-noha-baremetal-daily-euphrates/

3.5 YardStick test Cases

3.5.1 opnfv_yardstick_tc080

measure network latency between containers in k8s using ping https://git.opnfv.org/yardstick/tree/tests/opnfv/test_cases/opnfv_yardstick_tc080.yaml

3.5.2 opnfv_yardstick_tc081

measure network latency between container and VM using ping https://git.opnfv.org/yardstick/tree/tests/opnfv/test_cases/opnfv_yardstick_tc081.yaml

3.6 Multus implementation for OPNFV

This quickstart shows you how to easily install a Kubernetes cluster on VMs running with Vagrant. The installation uses a tool called kubeadm which is part of Kubernetes.

kubeadm assumes you have a set of machines (virtual or bare metal) that are up and running. In this way we can get a cluster with one master node and 2 workers (default). If you want to increase the number of workers nodes, please check the Vagrantfile inside the project.

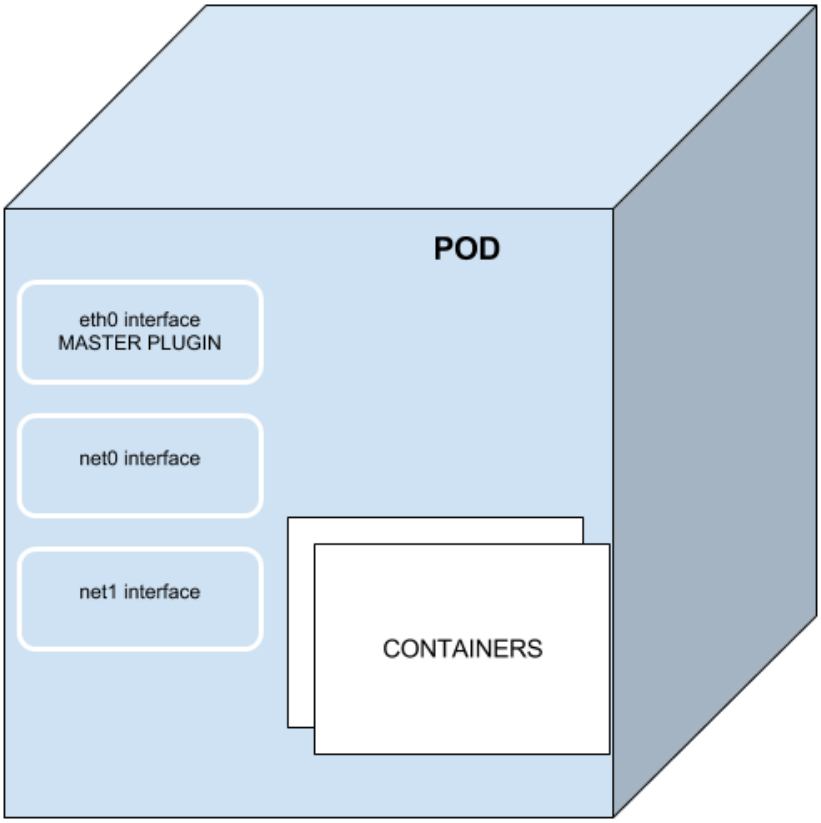
3.6.1 About Multus

[Multus](https://github.com/Intel-Corp/multus-cni) is a CNI proxy and arbiter of other CNI plugins.

With the help of Multus CNI plugin, multiple interfaces can be added at the same time when deploying a pod. Notably, Virtual Network Functions (VNFs) are typically requiring connectivity to multiple network interfaces.

The Multus CNI has the following features: - It is a contact between the container runtime and other plugins, and it doesn't have any of its own net configuration, it calls other plugins like flannel/calico to do the real net conf. job. - Multus reuses the concept of invoking the delegates in flannel, it groups the multi plugins into delegates and invoke each other in sequential order, according to the JSON scheme in the cni configuration. - No. of plugins supported is dependent upon the number of delegates in the conf file. - Master plugin invokes "eth0" interface in the pod, rest of plugins(Mininon plugins eg: sriov,ipam) invoke interfaces as "net0", "net1".. "netn". - The "masterplugin" is the only net conf option of Multus cni, it identifies the primary network. The default route will point to the primary network.

3.6.2 Multus example



3.7 Nginx implementation for OPNFV

This quickstart shows you how to easily install a Kubernetes cluster on VMs running with Vagrant. The installation uses a tool called kubeadm which is part of Kubernetes.

kubeadm assumes you have a set of machines (virtual or bare metal) that are up and running. In this way we can get a cluster with one master node and 2 workers (default). If you want to increase the number of workers nodes, please check the Vagrantfile inside the *kubeadm_basic/*.

3.7.1 About Nginx

Nginx is a web server which can also be used as a reverse proxy, load balancer and HTTP cache.

3.8 Ovsdpdk implementation for OPNFV

This quickstart shows you how to easily install a Kubernetes cluster on VMs running with Vagrant. The installation uses a tool called kubeadm which is part of Kubernetes.

kubeadm assumes you have a set of machines (virtual or bare metal) that are up and running. In this way we can get a cluster with one master node and 2 workers (default). If you want to increase the number of workers nodes, please check the Vagrantfile inside the project.

3.8.1 About OvS-dpdk

Open vSwitch* with the Data Plane Development Kit [OvS-DPDK](http://openvswitch.org/) is a high performance, open source virtual switch.

Using DPDK with OVS gives us tremendous performance benefits. Similar to other DPDK-based applications, we see a huge increase in network packet throughput and much lower latencies.

3.9 Clearwater implementation for OPNFV

CONTAINER4NFV setup a Kubernetes cluster on VMs running with Vagrant and kubeadm.

kubeadm assumes you have a set of machines (virtual or bare metal) that are up and running. In this way we can get a cluster with one master node and 2 workers (default). If you want to increase the number of workers nodes, please check the Vagrantfile inside the project.

Is Clearwater suitable for Network Functions Virtualization?

Network Functions Virtualization or NFV is, without any doubt, the hottest topic in the telco network space right now. It's an approach to building telco networks that moves away from proprietary boxes wherever possible to use software components running on industry-standard virtualized IT infrastructures. Over time, many telcos expect to run all their network functions operating at Layer 2 and above in an NFV environment, including IMS. Since Clearwater was designed from the ground up to run in virtualized environments and take full advantage of the flexibility of the Cloud, it is extremely well suited for NFV. Almost all of the ongoing trials of Clearwater with major network operators are closely associated with NFV-related initiatives.

3.9.1 About Clearwater

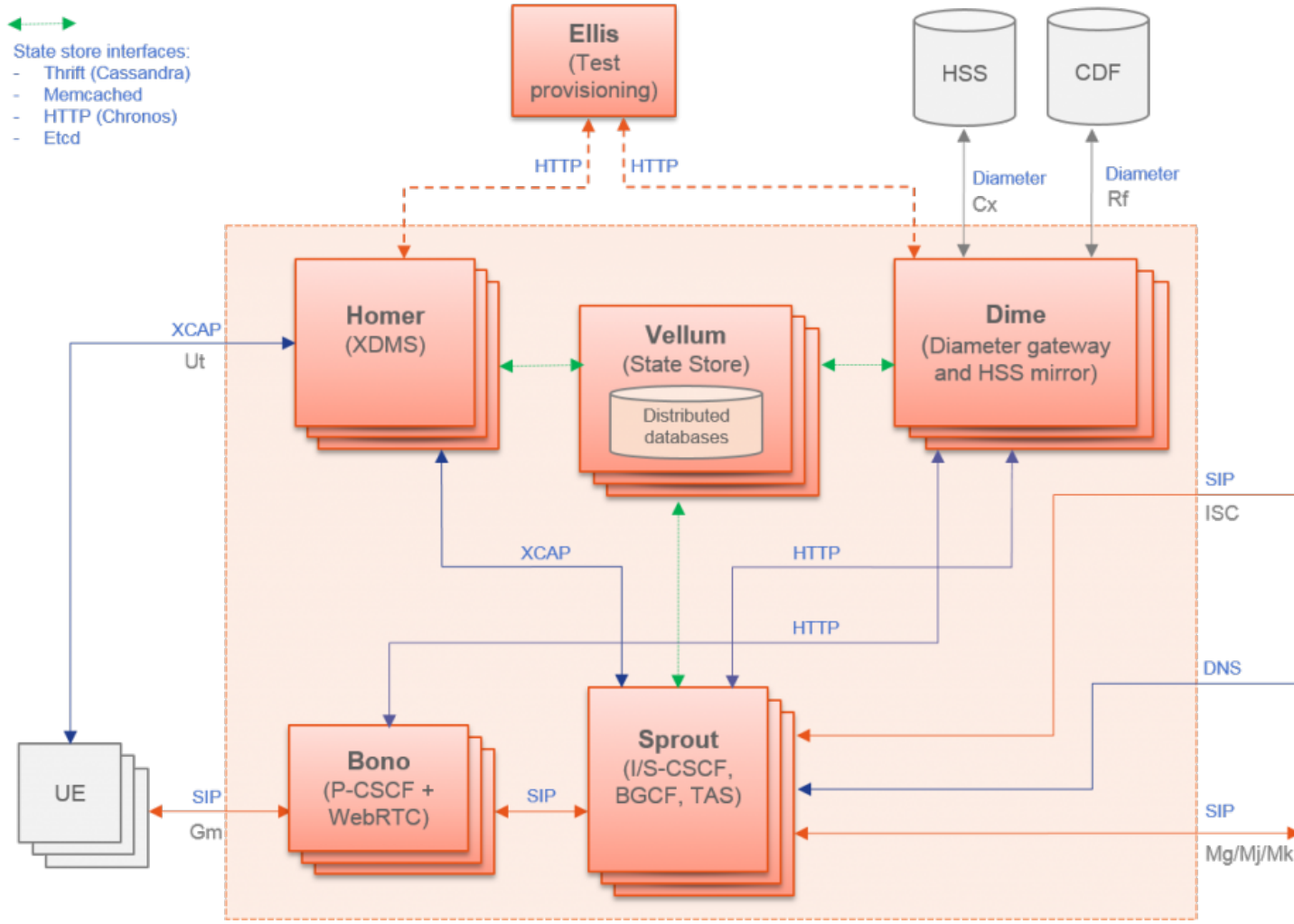
Clearwater follows IMS architectural principles and supports all of the key standardized interfaces expected of an IMS core network. But unlike traditional implementations of IMS, Clearwater was designed from the ground up for the Cloud. By incorporating design patterns and open source software components that have been proven in many global Web applications, Clearwater achieves an unprecedented combination of massive scalability and exceptional cost-effectiveness.

Clearwater provides SIP-based call control for voice and video communications and for SIP-based messaging applications. You can use Clearwater as a standalone solution for mass-market VoIP services, relying on its built-in set of basic calling features and standalone susbscriber database, or you can deploy Clearwater as an IMS core in conjunction with other elements such as Telephony Application Servers and a Home Subscriber Server.

Clearwater was designed from the ground up to be optimized for deployment in virtualized and cloud environments. It leans heavily on established design patterns for building and deploying massively scalable web applications, adapting these design patterns to fit the constraints of SIP and IMS. The Clearwater architecture therefore has some similarities to the traditional IMS architecture but is not identical.

- All components are horizontally scalable using simple, stateless load-balancing.
- All long lived state is stored on dedicated "Vellum" nodes which make use of cloud-optimized storage technologies such as Cassandra. No long lived state is stored on other production nodes, making it quick and easy to dynamically scale the clusters and minimizing the impact if a node is lost.
- Interfaces between the front-end SIP components and the back-end services use RESTful web services interfaces.
- Interfaces between the various components use connection pooling with statistical recycling of connections to ensure load is spread evenly as nodes are added and removed from each layer.

3.9.2 Clearwater Architecture



3.10 Quickstart

This repository contains instructions and resources for deploying Metaswitch's Clearwater project with Kubernetes.

If you need more information about Clearwater project please checkout our [documentation](https://github.com/opnfv/container4nfv/blob/master/docs/release/userguide/clearwater-project.rst) or the official repository.

3.10.1 Exposed Services

The deployment exposes:

- the Ellis web UI on port 30080 for self-provisioning.
- STUN/TURN on port 3478 for media relay.
- SIP on port 5060 for service.
- SIP/WebSocket on port 5062 for service.

SIP devices can register with bono.:5060 and the Ellis provisioning interface can be accessed at port 30080.

3.10.2 Prerequirement

Install Docker and Vagrant

CONTAINER4NFV uses `setup_vagrant.sh` to install all resource used by this repository.

```
container4nfv/src/vagrant# ./setup_vagrant.sh -b libvirt
```

3.10.3 Instalation

Deploy Clearwater with kubeadm

Check `clearwater/clearwater_setup.sh` for details about k8s deployment.

```
container4nfv/src/vagrant/kubeadm_clearwater# ./deploy.sh
```

3.10.4 Destroy

```
container4nfv/src/vagrant# ./cleanup.sh
```

3.10.5 Making calls through Clearwater

Connect to Ellis service

It's important to connect to Ellis to generate the SIP username, password and domain we will use with the SIP client. Use your <master ip addres> + port 30080 (k8s default port). If you are not which Ellis's url is, please check inside your master node.

```
kubeadm_clearwater# vagrant ssh master
master@vagrant# ifconfig eth0 | grep "inet addr" | cut -d ':' -f 2 | cut -d ' ' -f 1
192.168.121.3
```

In your browser connect to *<master_ip>:30080* (ex. 192.168.121.3:30080).

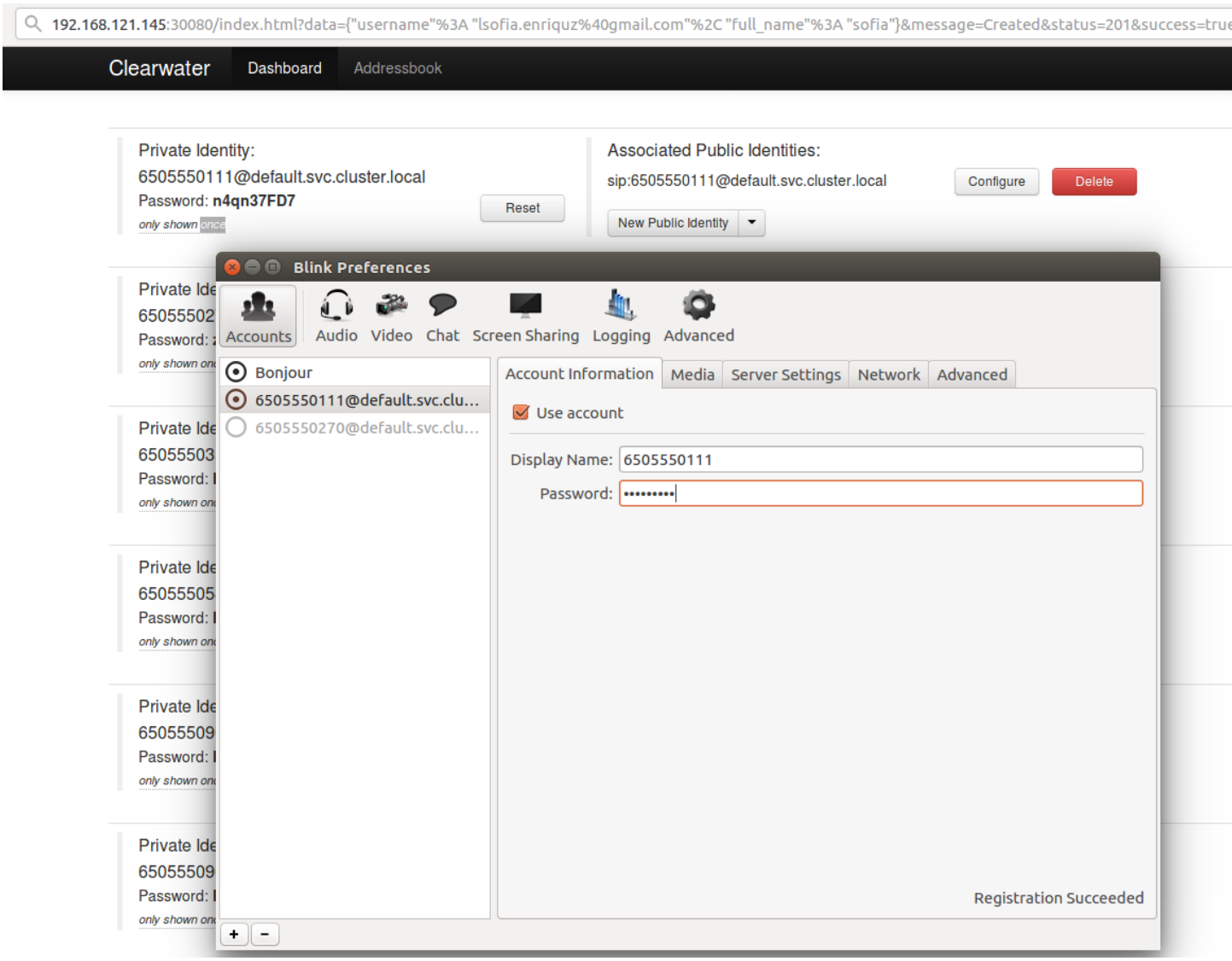
After that, signup and generate two users. The signup key is **secret**. Ellis will automatically allocate you a new number and display its password to you. Remember this password as it will only be displayed once. From now on, we will use <username> to refer to the SIP username (e.g. 6505551234) and <password> to refer to the password.

Config and install two SIP clients

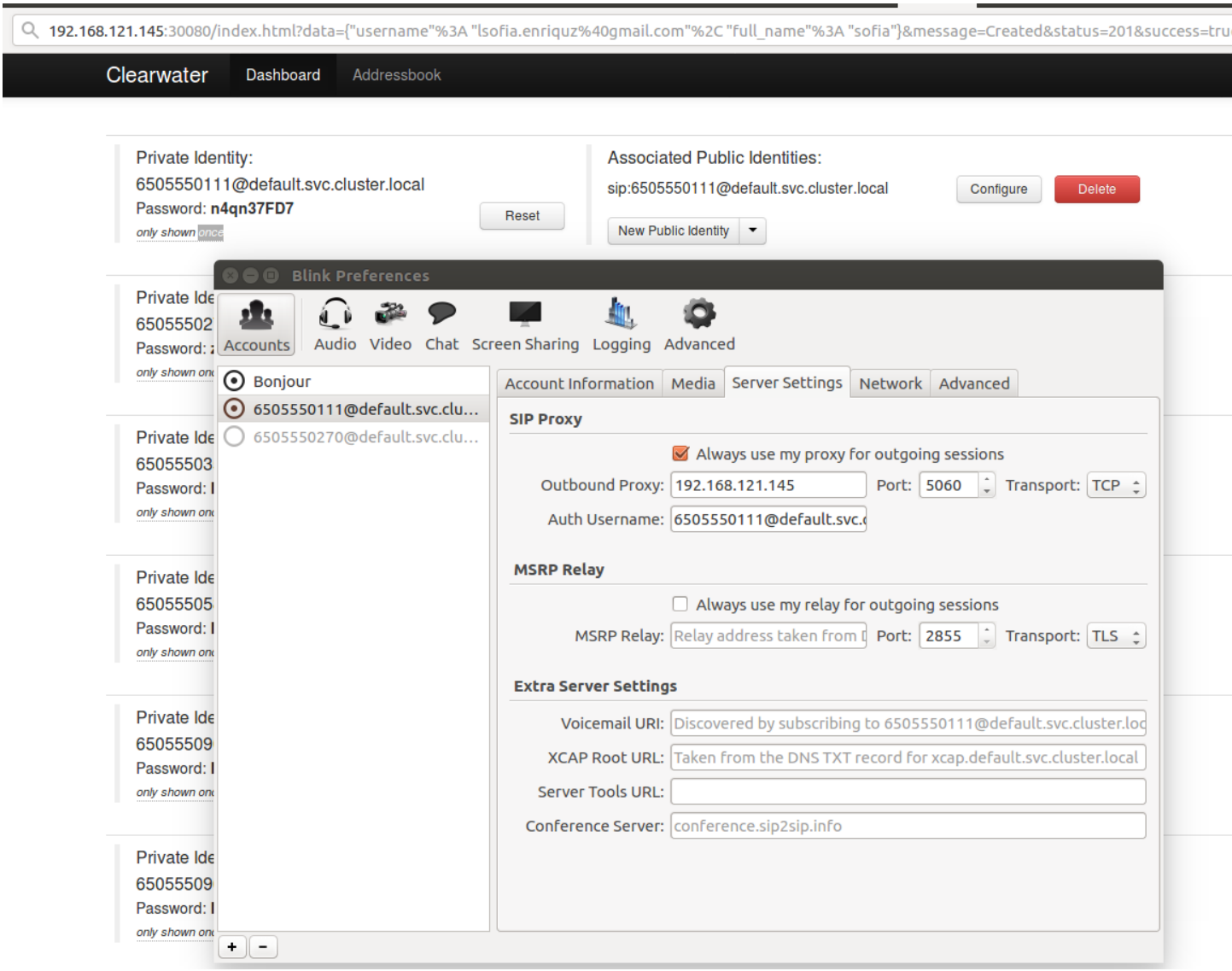
We'll use both Twinkle and Blink SIP client. , since we are going to try this out inside a LAN network. This is, of course, only a local test inside a LAN network. Configure the clients may be a little bit trickie, so we add some screenshots:

Blink setup

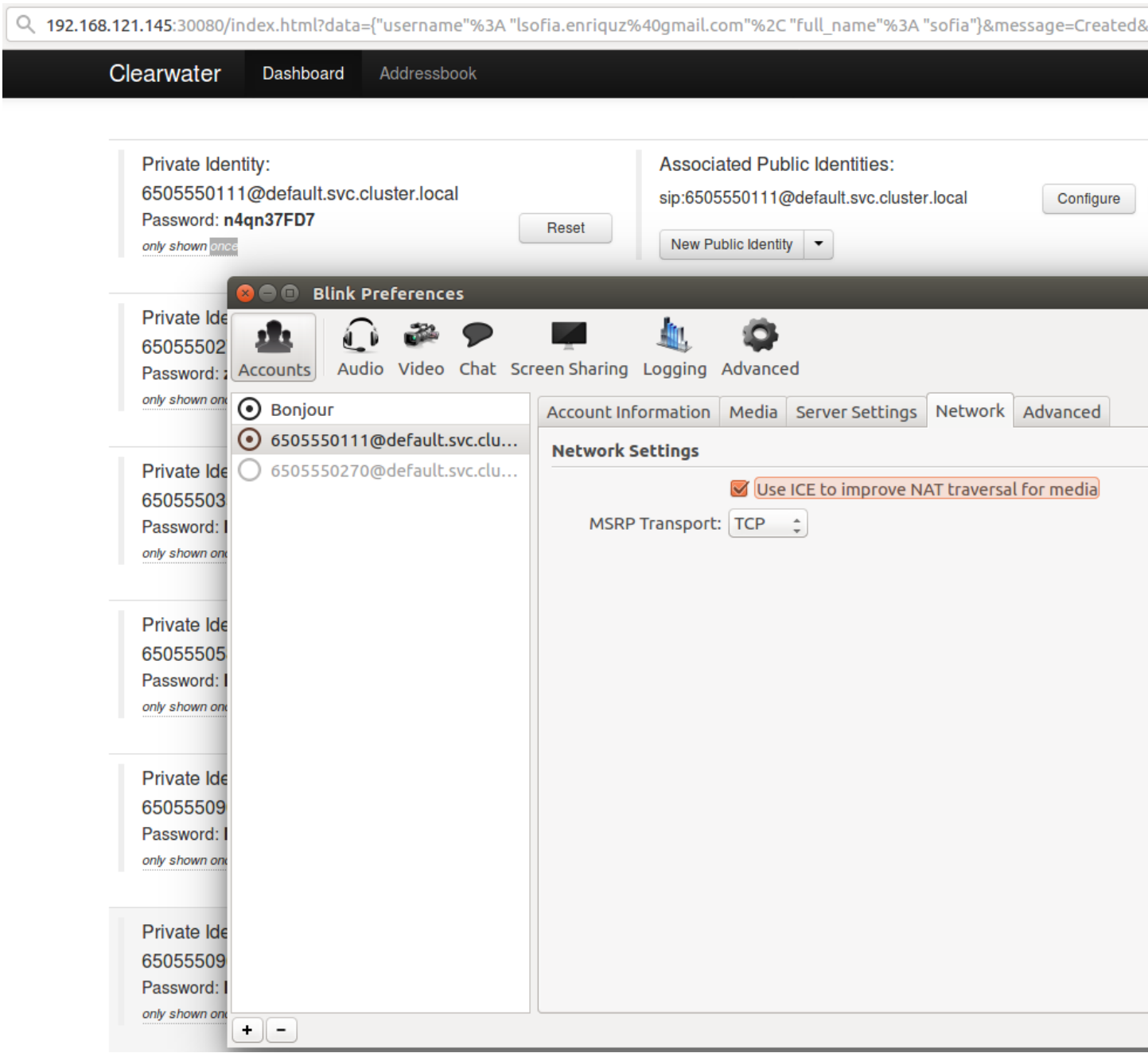
1. Add <username> and <password>.

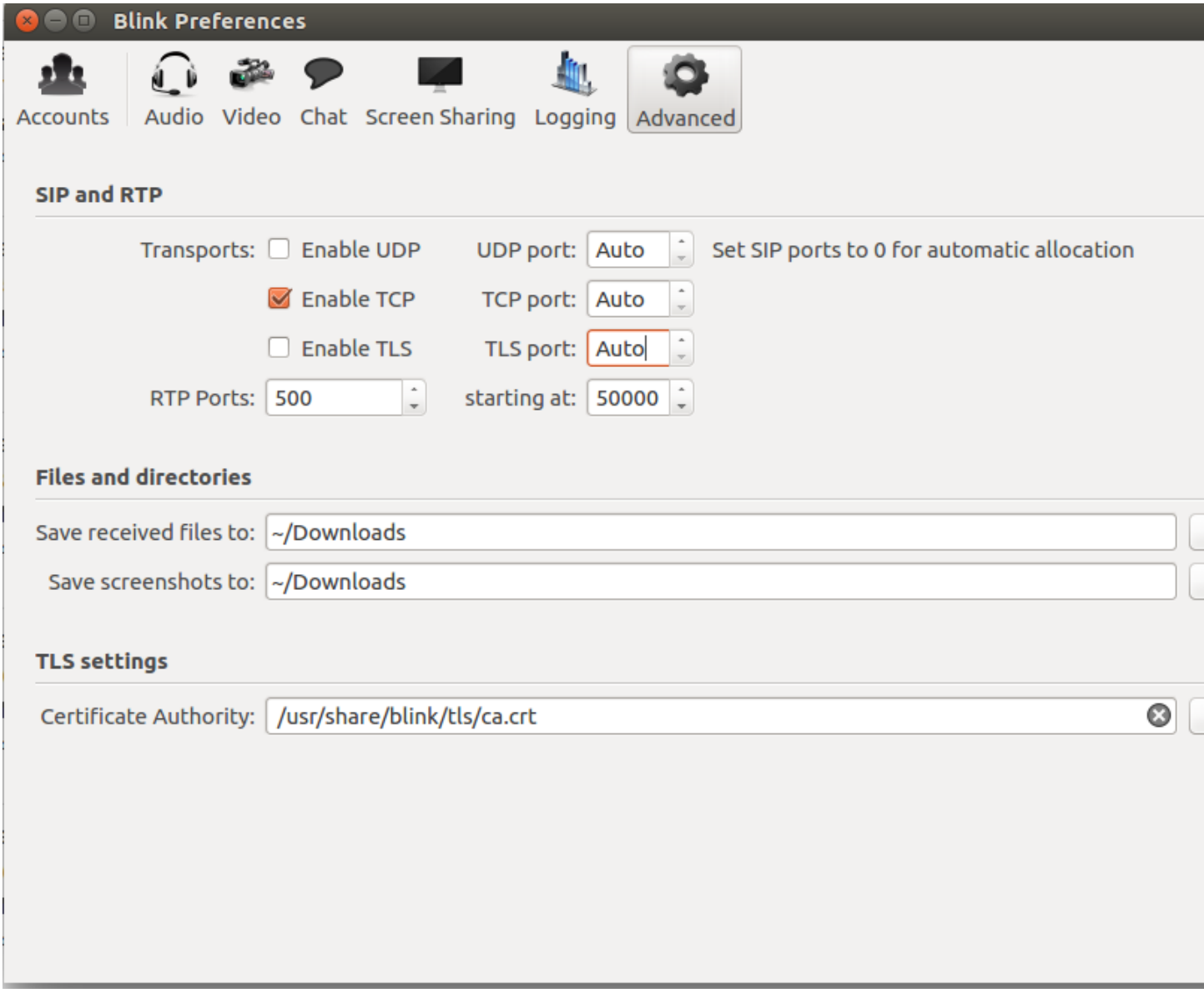


2. Configure a proxy to k8s.



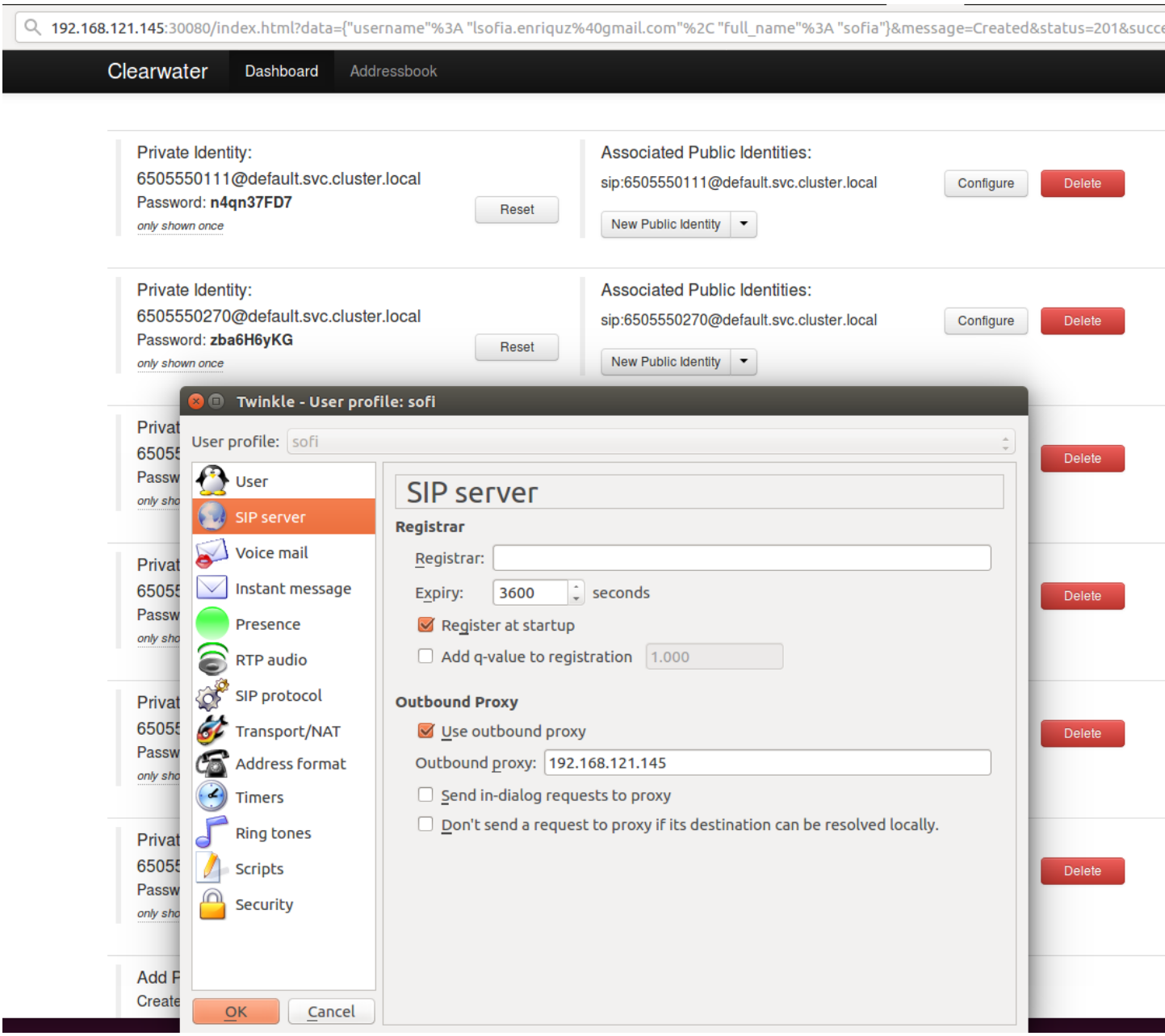
3. Configure the network to use TCP only.


192.168.121.145:30080/index.html?data={"username"%3A "lsofia.enriquz%40gmail.com"%2C "full_name"%3A "sofia"}&message=Created&
Clearwater
Dashboard
Addressbook
Private Identity:
6505550111@default.svc.cluster.local
Password: n4qn37FD7
only shown once
Reset
Associated Public Identities:
sip:6505550111@default.svc.cluster.local
Configure
New Public Identity
Blink Preferences
Accounts
Audio
Video
Chat
Screen Sharing
Logging
Advanced
Bonjour
6505550111@default.svc.clu...
6505550270@default.svc.clu...
Account Information
Media
Server Settings
Network
Advanced
Network Settings
Use ICE to improve NAT traversal for media
MSRP Transport:
TCP

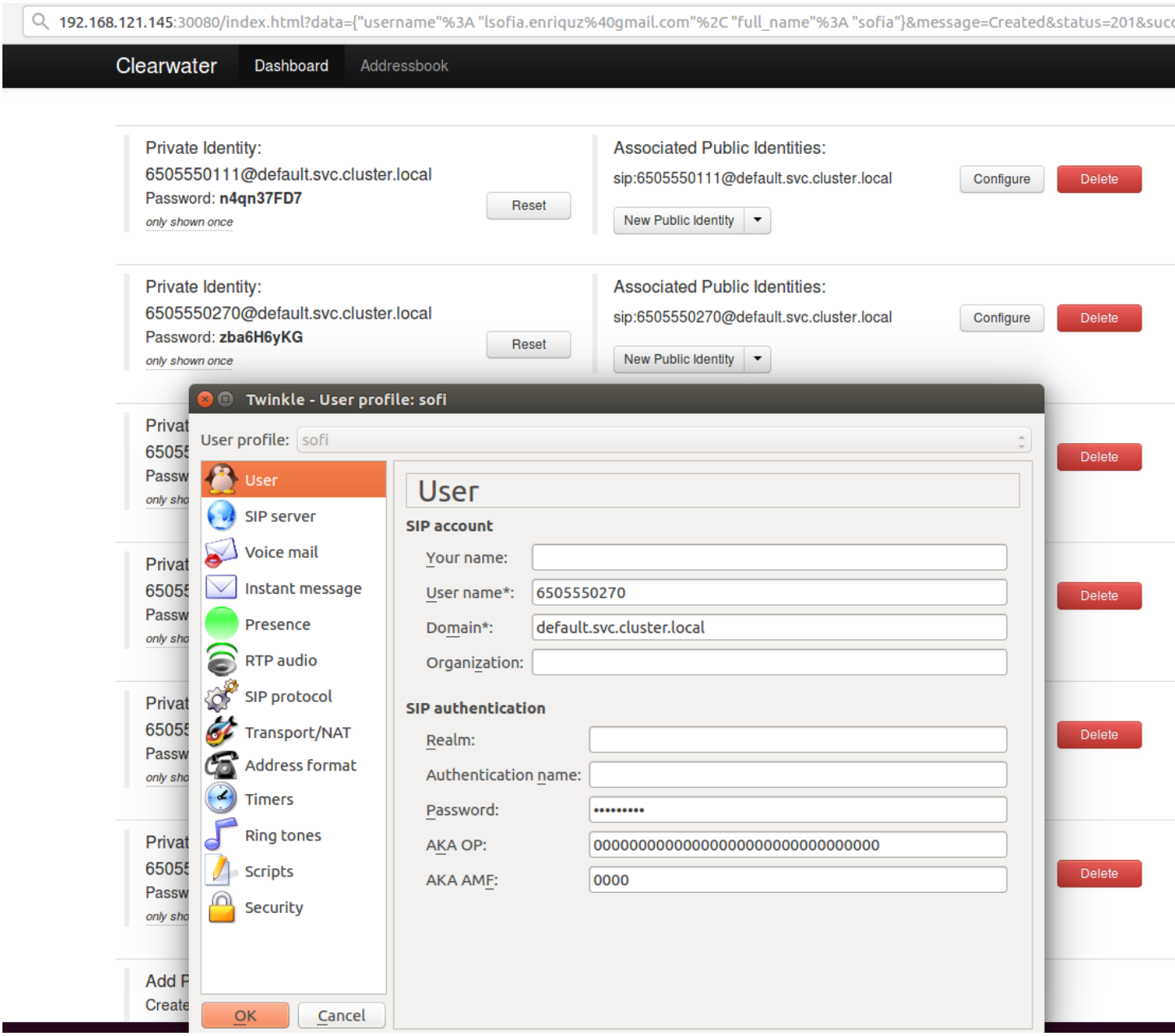


Twinkle setup

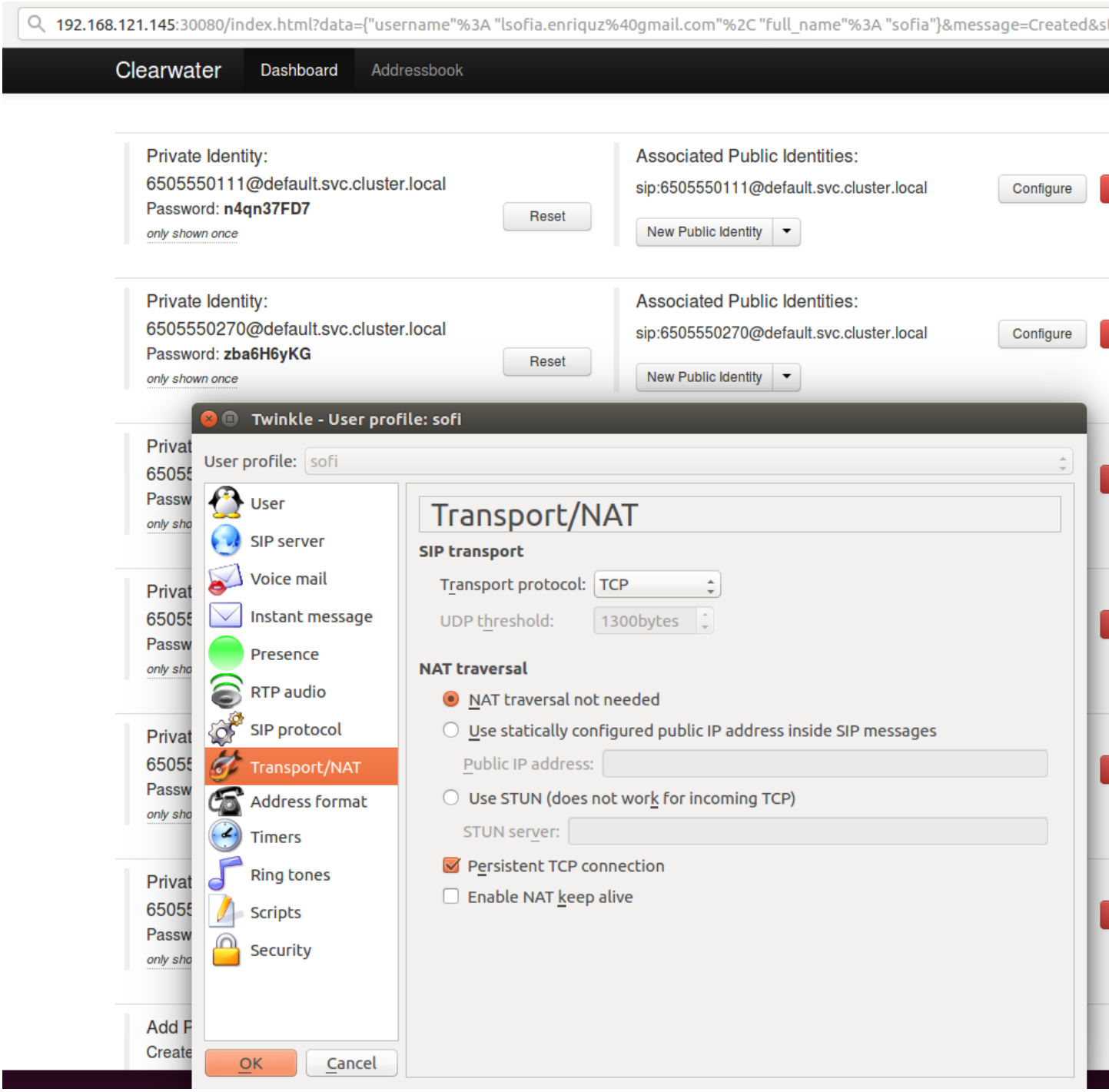
1. Configure a proxy to k8s.



2. Add <username> and <password>.



3. Configure the network to use TCP only.


192.168.121.145:30080/index.html?data={"username"%3A "lsofia.enriquz%40gmail.com"%2C "full_name"%3A "sofia"}&message=Created&s
Clearwater
Dashboard
Addressbook
Private Identity:
6505550111@default.svc.cluster.local
Password: n4qn37FD7
only shown once
Reset
Associated Public Identities:
sip:6505550111@default.svc.cluster.local
Configure
New Public Identity
Private Identity:
6505550270@default.svc.cluster.local
Password: zba6H6yKG
only shown once
Reset
Associated Public Identities:
sip:6505550270@default.svc.cluster.local
Configure
New Public Identity
Twinkle - User profile: sofi
User profile:
sofi
User
SIP server
Voice mail
Instant message
Presence
RTP audio
SIP protocol
Transport/NAT
Address format
Timers
Ring tones
Scripts
Security
Transport/NAT
SIP transport
Transport protocol:
TCP
UDP threshold:
1300bytes
NAT traversal
NAT traversal not needed
Use statically configured public IP address inside SIP messages
Public IP address:
Use STUN (does not work for incoming TCP)
STUN server:
Persistent TCP connection
Enable NAT keep alive
OK
Cancel

Make the call