FED4FIRE
Federation for FIRE Plus

**Thijs Walcarius**

*imec – Ghent University, Belgium*

# Advanced Networking and Cloud Experiments

*June 2020*

# Table of Contents

## CLOUD

- **ESpec**: The Experiment Specification
- Easy deployment of **OpenStack** with EnOS

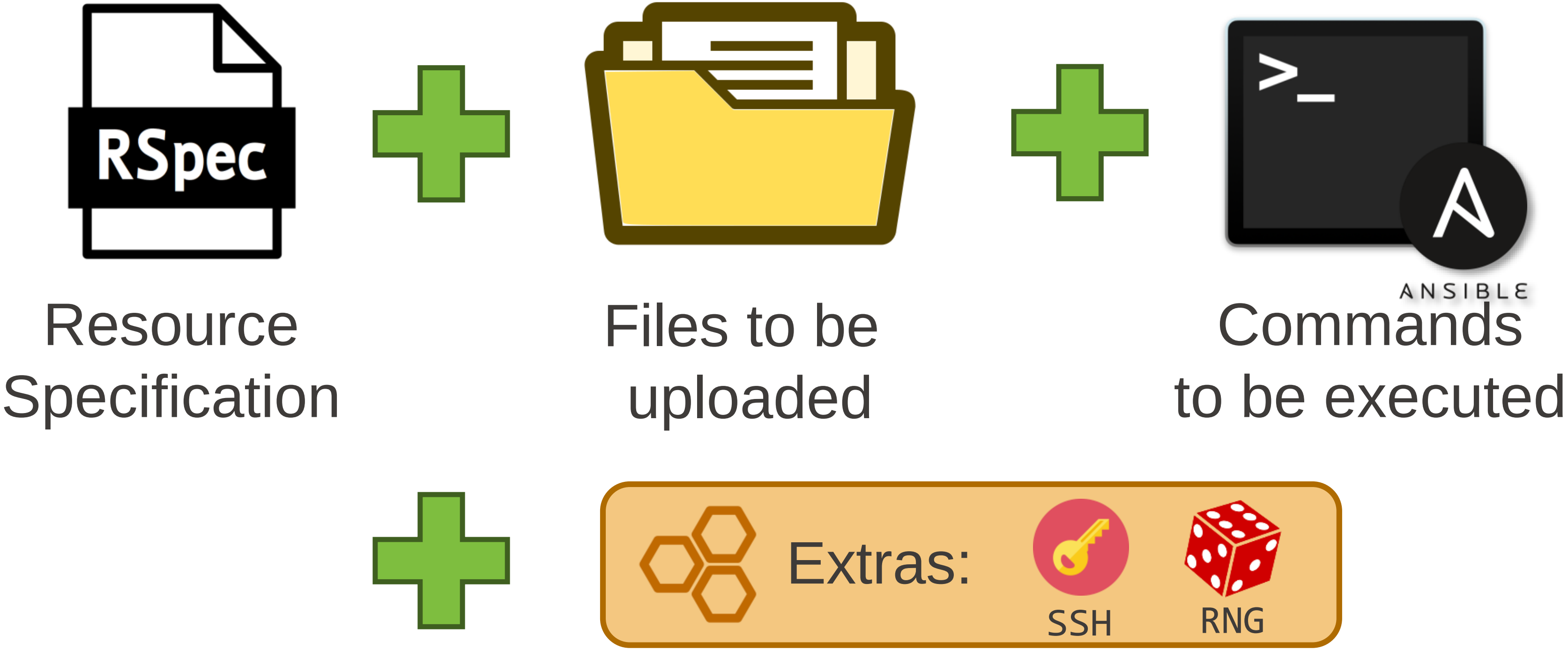
## WIRED NETWORKING

- Workflow for scaling up experiments
- Scaling up networking experiments with **Shared LAN's**
- **Examples** of Advanced Networking Experiments

# The Experiment Specification

# What is an Experiment Specification?

## Espec bundles:



Resource Specification + Files to be uploaded + Commands to be executed

+ Extras:

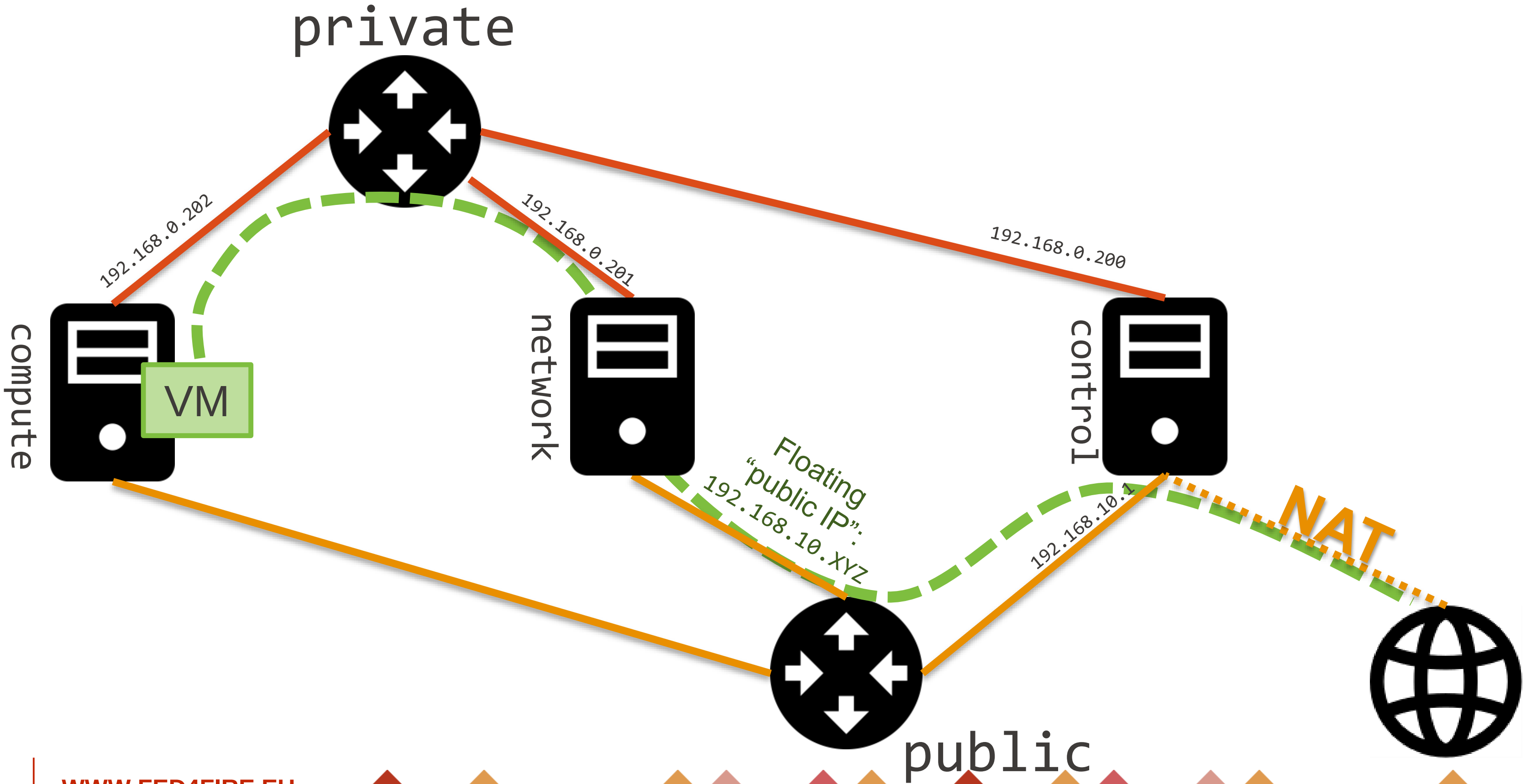
# Deploying OpenStack with EnOS

# Deploying OpenStack with EnOS

EnOS allows you to **Deploy**, **Customize** and **Benchmark** OpenStack

- Developed by Inria
- Wrapper around **Kolla-Ansible**
- Deploys all OS-services as Docker containers
- ESpec generates the EnOS config file for bootstrapping the deployment

# OpenStack experiment architecture



# Resources on EnOS

## Tutorial

https://doc.ilabt.imec.be/ilabt/virtualwall/tutorials/openstack.html

## EnOS documentation

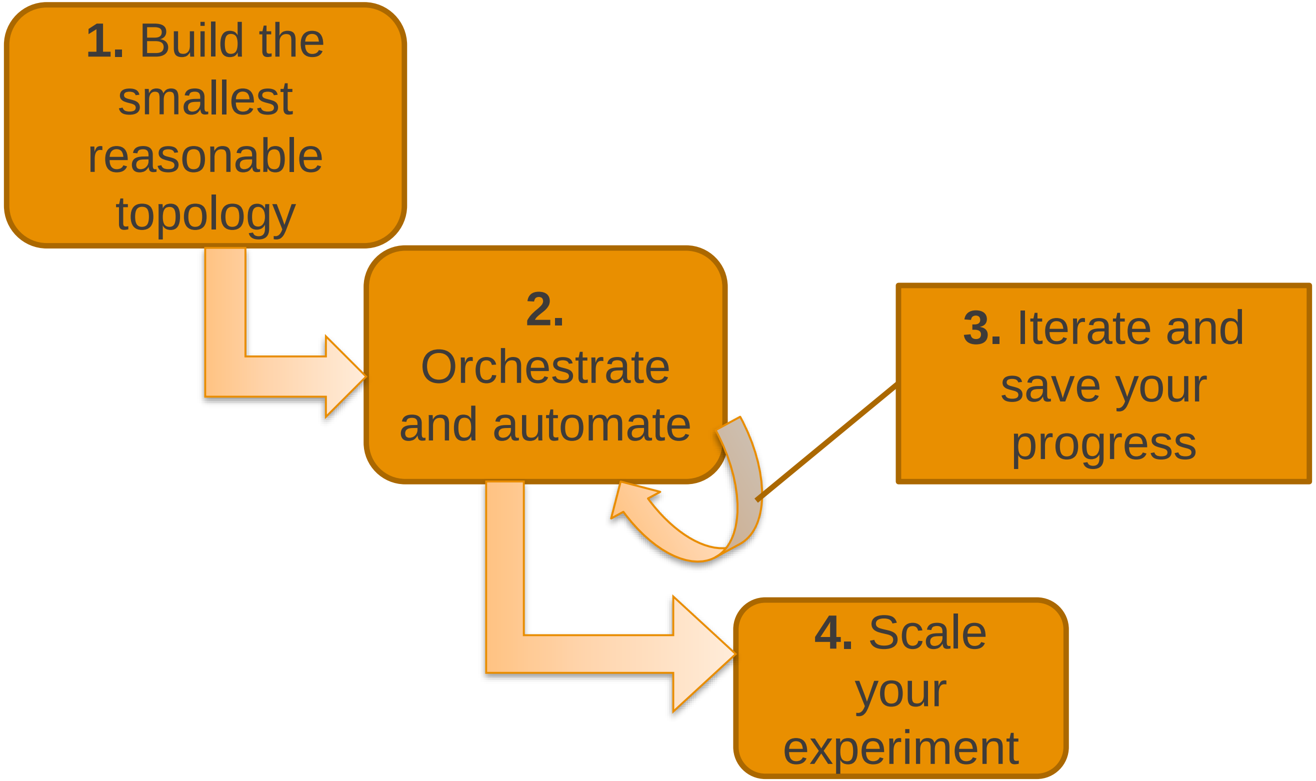
https://enos.readthedocs.io/

## EnOS ESpec repository

https://gitlab.ilabt.imec.be/ilabt/enos-espec/

# Workflow for creating and scaling up experiments

# Recommended workflow



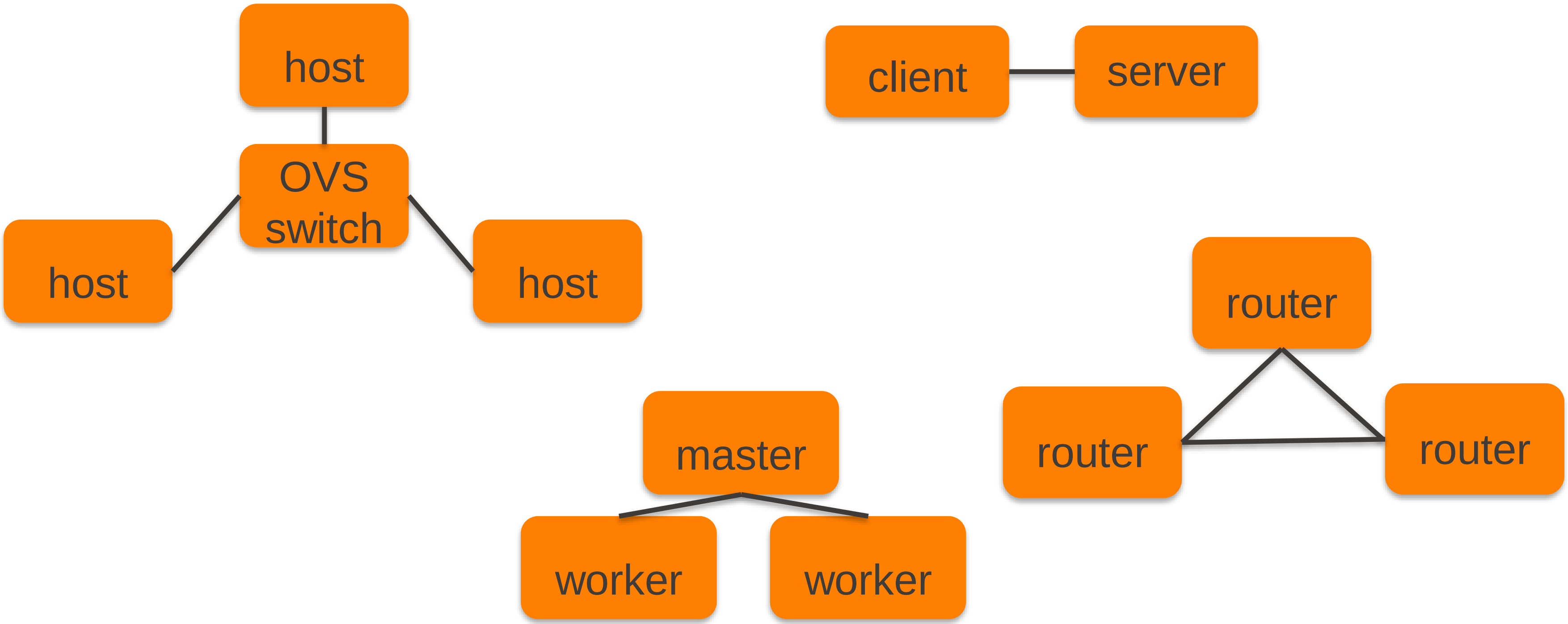
**1.** Build the smallest reasonable topology

**2.** Orchestrate and automate

**3.** Iterate and save your progress

**4.** Scale your experiment

# 1. Build the smallest reasonable topology



host

OVS switch

host

host

client

server

router

router

router

master

worker

worker

# 2. Automate as you go

Use Configuration Management Systems to automate installation and configuration of software

Many tools available for this job: Ansible, Chef, Puppet, …

# 3. Save your progress

Log all of your experimental artifacts for every experiment that works

- RSpec
- image
- install script
- custom software
- measurements
- etc.

Use version control to store your artifacts

Always know the **last configuration that worked**

# 4. Scale your experiment

Only scale up when your smallest reasonable experiment is working

Adapt your request RSpec to add more nodes

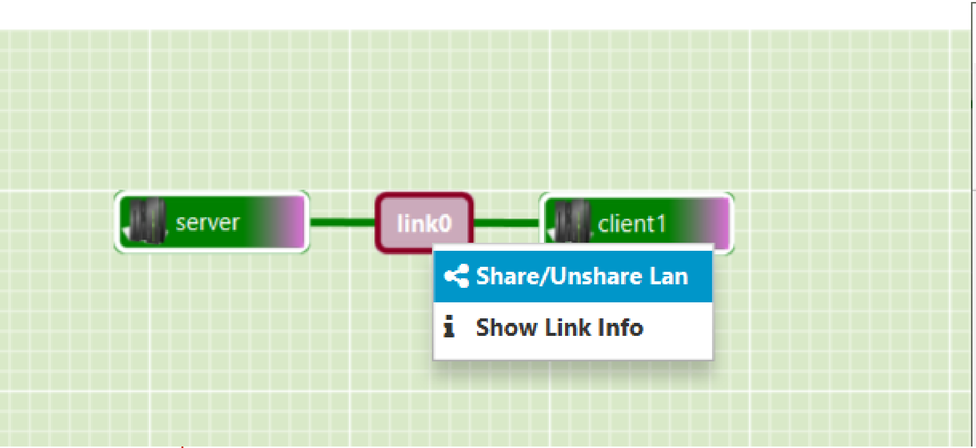
- Roll your own scaling script: mostly copy/pasting with minimal editing required
- Use geni-lib

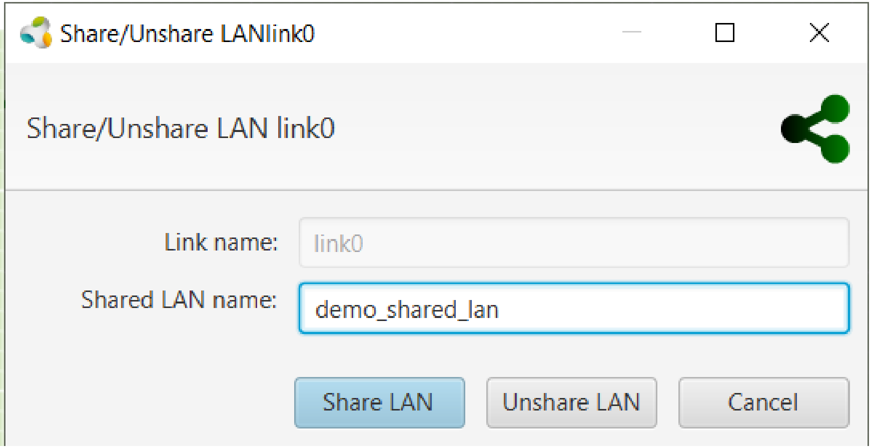
# Scaling up experiments with shared LAN’s

# Shared LAN

Shared LAN’s allow you to add extra servers to an existing network in an experiment

**Step 1:** Right click on network and choose “Share/Unshare Lan”




Share/Unshare LANlink0

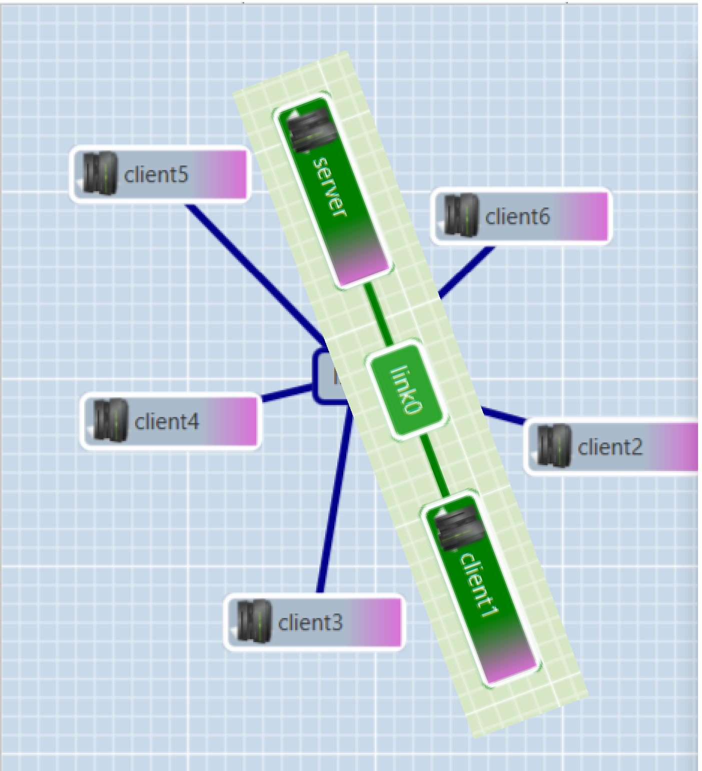
Share/Unshare LAN link0

Link name: link0

Shared LAN name: demo_shared_lan

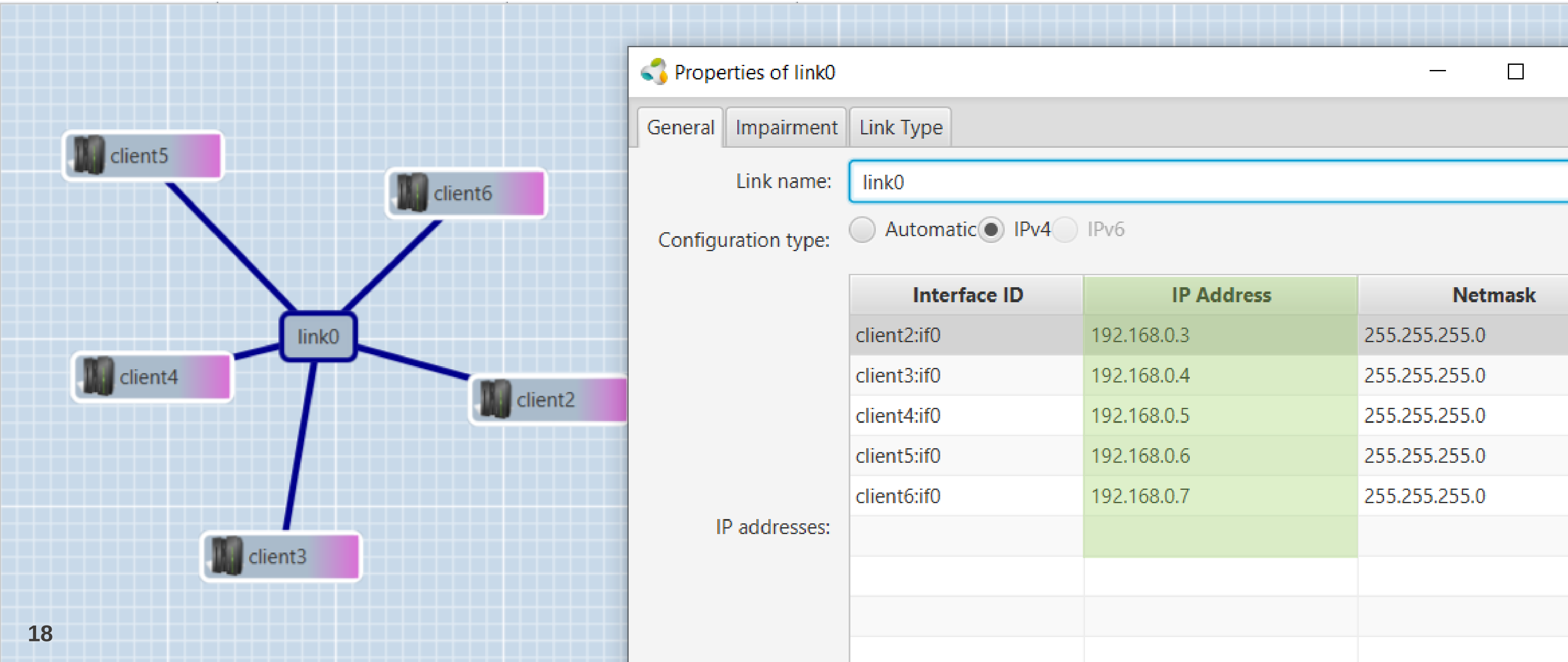
# Shared LAN

**Step 2**: Design a new experiment with extra servers



# Shared LAN

## **Step 3**: Fix duplicate IP-addresses



Properties of link0

General | Impairment | Link Type

Link name: link0

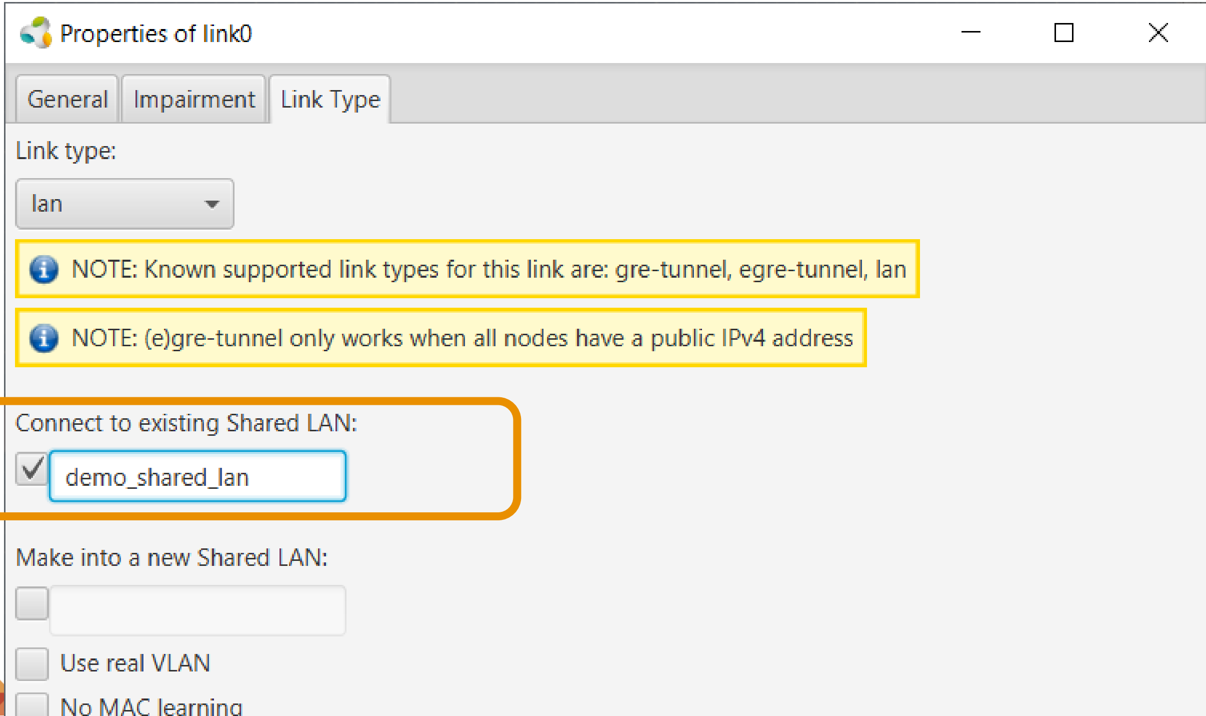
Configuration type: ○ Automatic ● IPv4 ○ IPv6

IP addresses:

| Interface ID | IP Address | Netmask |
|---|---|---|
| client2:if0 | 192.168.0.3 | 255.255.255.0 |
| client3:if0 | 192.168.0.4 | 255.255.255.0 |
| client4:if0 | 192.168.0.5 | 255.255.255.0 |
| client5:if0 | 192.168.0.6 | 255.255.255.0 |
| client6:if0 | 192.168.0.7 | 255.255.255.0 |

# Shared LAN

**Step 4**: Configure link in new experiment to connect to existing Shared LAN



Properties of link0

Link type:

lan

NOTE: Known supported link types for this link are: gre-tunnel, egre-tunnel, lan

NOTE: (e)gre-tunnel only works when all nodes have a public IPv4 address

Connect to existing Shared LAN:

☑ demo_shared_lan

Make into a new Shared LAN:

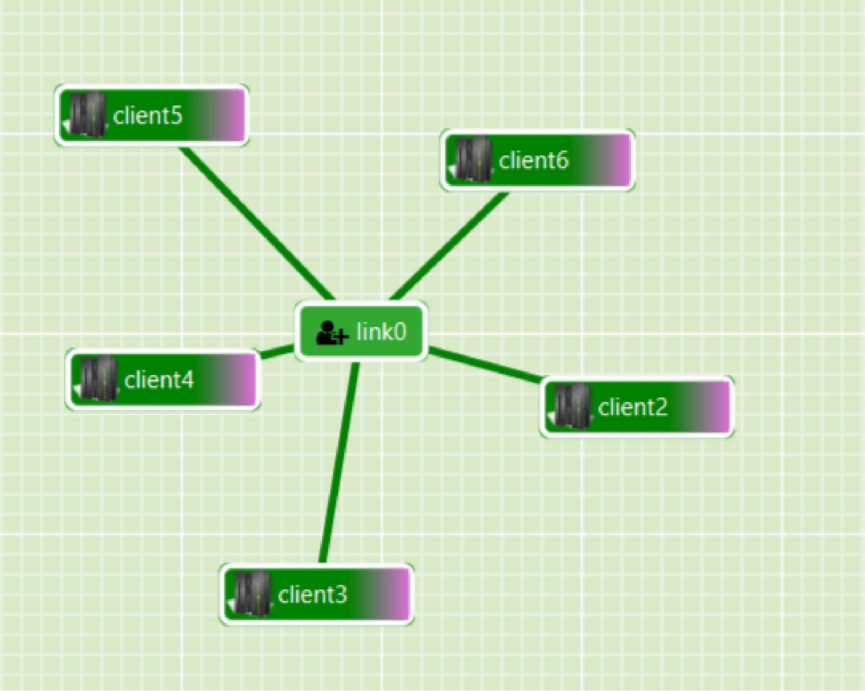
☐

☐ Use real VLAN

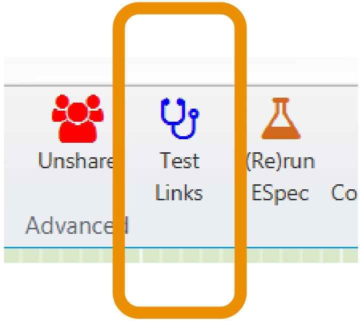
☐ No MAC learning

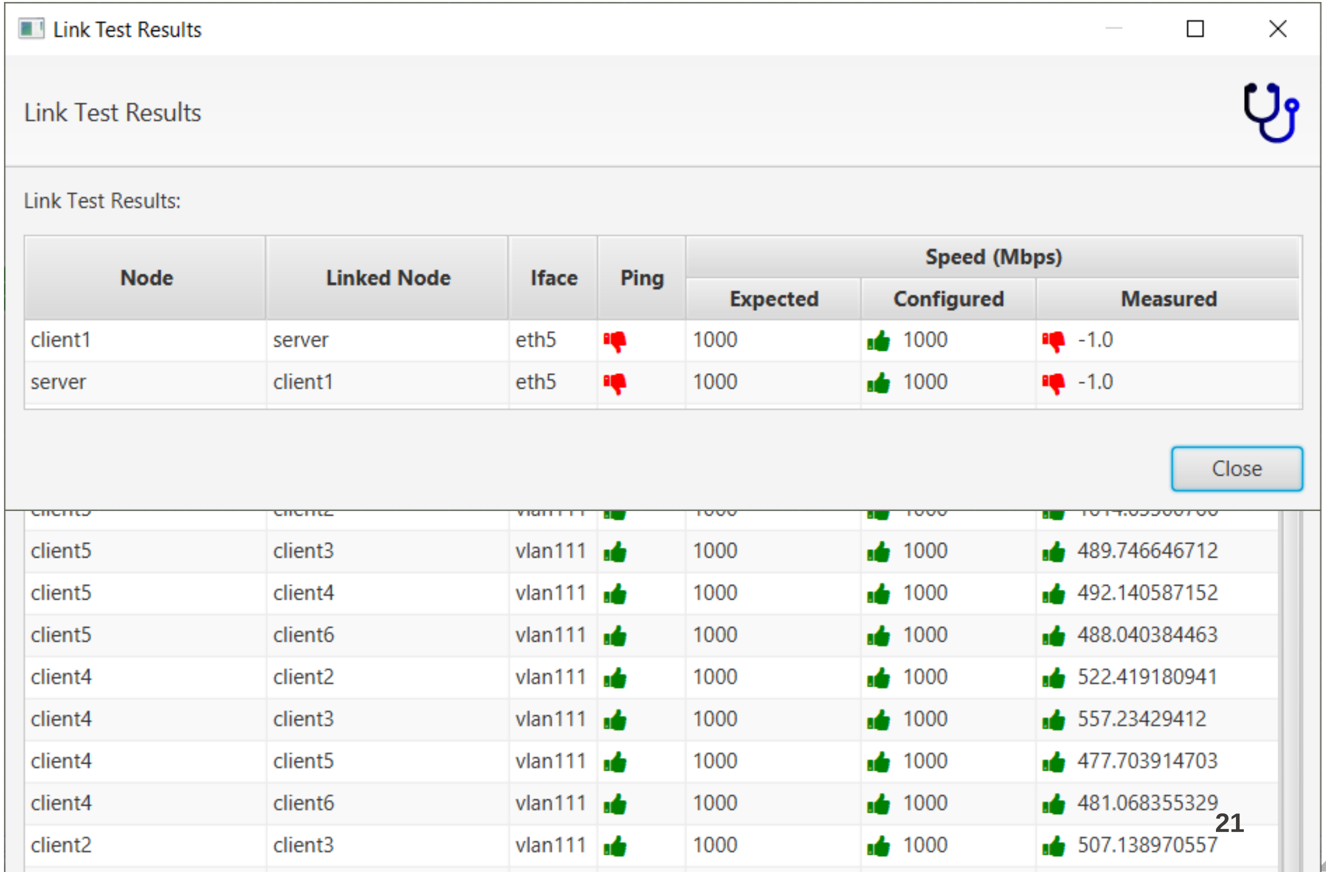
# Shared LAN

**Step 5**: Start the new experiment



# Test your links!





Link Test Results

Link Test Results:

| Node | Linked Node | Iface | Ping | Speed (Mbps) | | |
|---|---|---|---|---|---|---|
| | | | | Expected | Configured | Measured |
| client1 | server | eth5 | 👎 | 1000 | 👍 1000 | 👎 -1.0 |
| server | client1 | eth5 | 👎 | 1000 | 👍 1000 | 👎 -1.0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| client5 | client3 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 489.746646712 |
| client5 | client4 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 492.140587152 |
| client5 | client6 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 488.040384463 |
| client4 | client2 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 522.419180941 |
| client4 | client3 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 557.23429412 |
| client4 | client5 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 477.703914703 |
| client4 | client6 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 481.068355329 |
| client2 | client3 | vlan111 | 👍 | 1000 | 👍 1000 | 👍 507.138970557 |

# Examples of advanced networking experiments

# Using multiple testbeds in an experiment

## PUBLIC INTERNET

+ Available between all testbeds

– No guarantee on latency, throughput, packet loss, …

– Sometimes only NAT-ted access to public internet

## PRIVATE CONNECTIONS

~ Available between certain testbeds (Fed4FIRE+, but also GENI (=US testbeds))

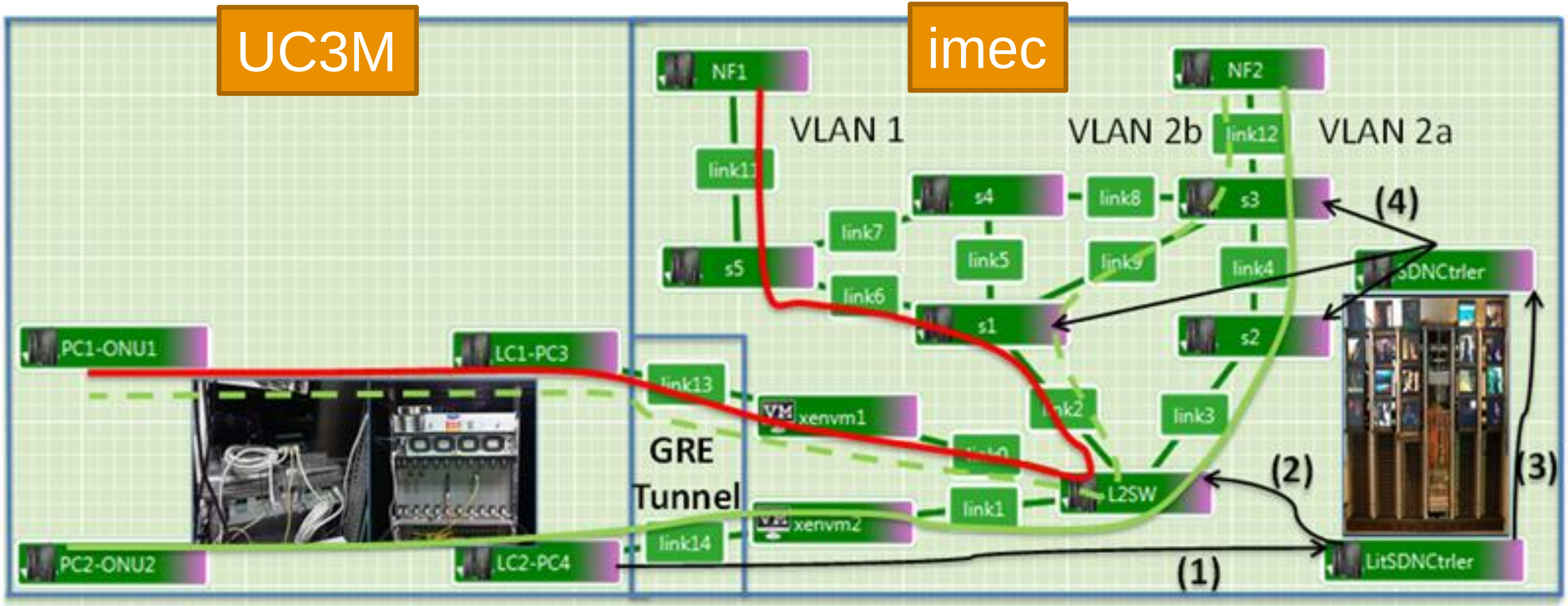
+ Better guarantees on latency throughput, packet loss

~ Not well documented due to volatile nature
➔ Contact us!

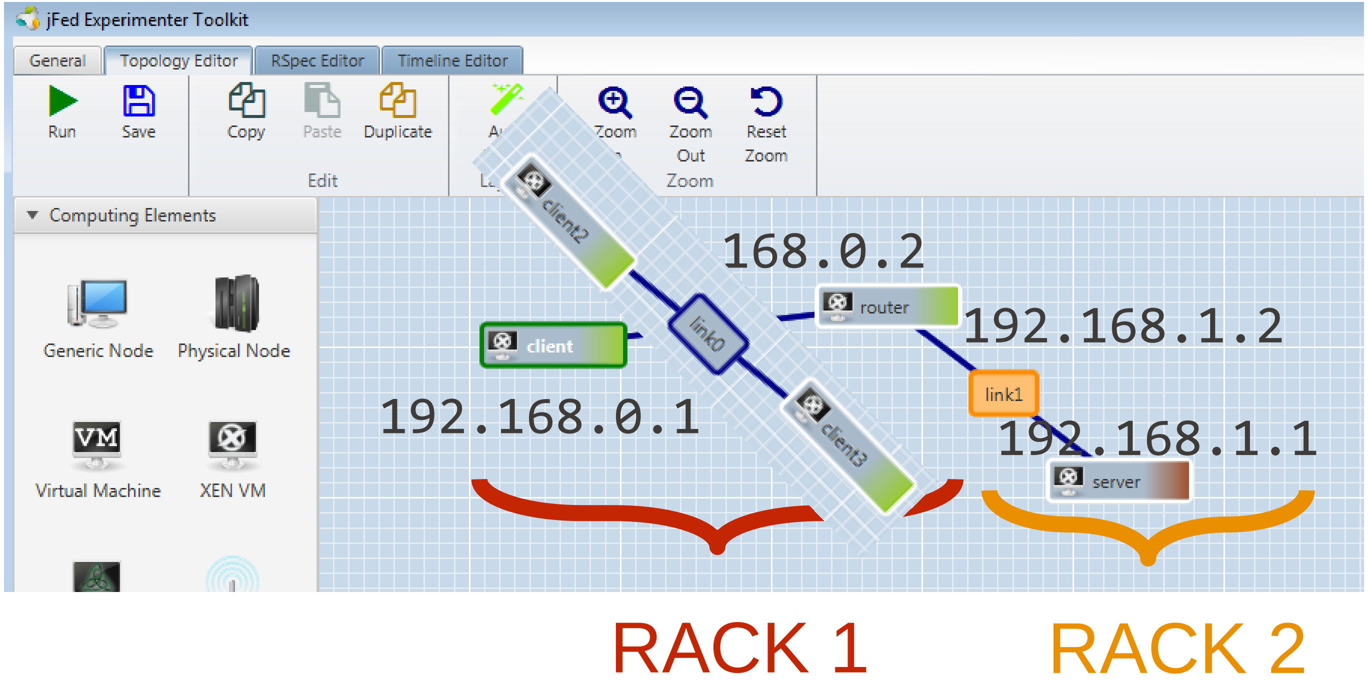
# Creating multi-testbed experiments

**Recommended workflow:**

1. Start with creating a multi-testbed backbone
2. Scale up with shared LAN's on each site

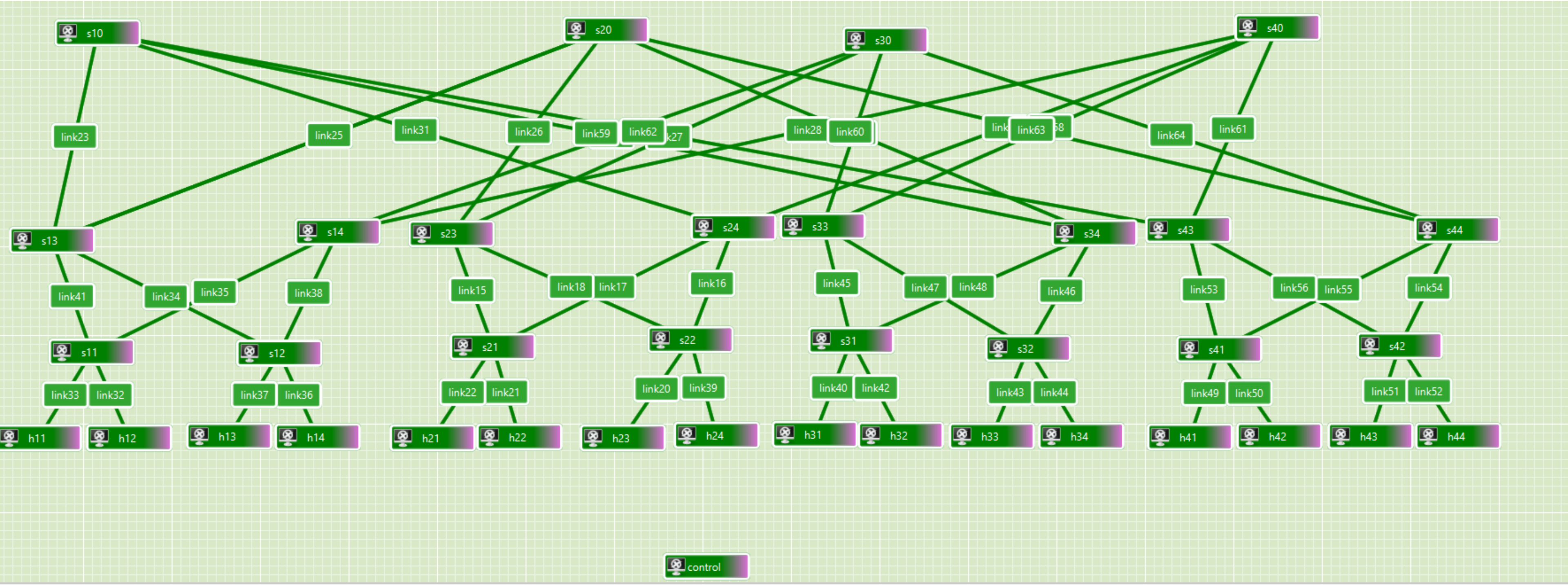


# Topology



```
route add -net 192.168.1.0/24 gw 192.168.0.2
```

# Advanced SDN experiment



# Documentation

FED4FIRE+

## Testbeds Overview

https://www.fed4fire.eu/testbeds/

## Technical Documentation

https://doc.fed4fire.eu/

# Scaling up manually

# Scaling manually: nodes

```
<rspec type="request" xmlns=…>
  <node client_id="node0" exclusive="true"
component_manager_id="urn:publicid:IDN+wall2.ilabt.iminds.be
+authority+cm">
    <sliver_type name="raw-pc"/>
  </node>
</rspec>
```

Copy paste <node>-element with all its child elements

Change client_id of node to be unique

# Scaling manually: nodes

```
<rspec type="request" xmlns=…>
 <node client_id="node0" exclusive="true"
component_manager_id="urn:publicid:IDN+wall2.ilabt.iminds.be
+authority+cm">
    <sliver_type name="raw-pc"/>
  </node>
  <node client_id="node1" exclusive="true"
component_manager_id="urn:publicid:IDN+wall2.ilabt.iminds.be
+authority+cm">
    <sliver_type name="raw-pc"/>
  </node>
</rspec>
```

# Scaling manually: nodes with links

```
<rspec type="request" xmlns=…>
  <node client_id="node0" exclusive="false" component_manager_id="urn:publicid:IDN+utahddc.geniracks.net+authority+cm">
    <sliver_type name="default-vm"/>
    <interface client_id="node0:if0">
      <ip address="192.168.0.1" netmask="255.255.255.0" type="ipv4"/>
    </interface>
  </node>
  <node client_id="node1" exclusive="false" component_manager_id="urn:publicid:IDN+utahddc.geniracks.net+authority+cm">
    <sliver_type name="default-vm"/>
    <interface client_id="node1:if0">
```

# Scaling manually

- Copy paste `<node>`-element with all its child elements
- Change `client_id` of node and interface to be unique
- Change IP-addresses of interface to prevent duplicates
- Add extra interface-reference in `<link>`

Pay attention to the details!

# Scaling an experiment with geni-lib

# What is geni-lib?

- **Python library** for querying the GENI Aggregate Manager API

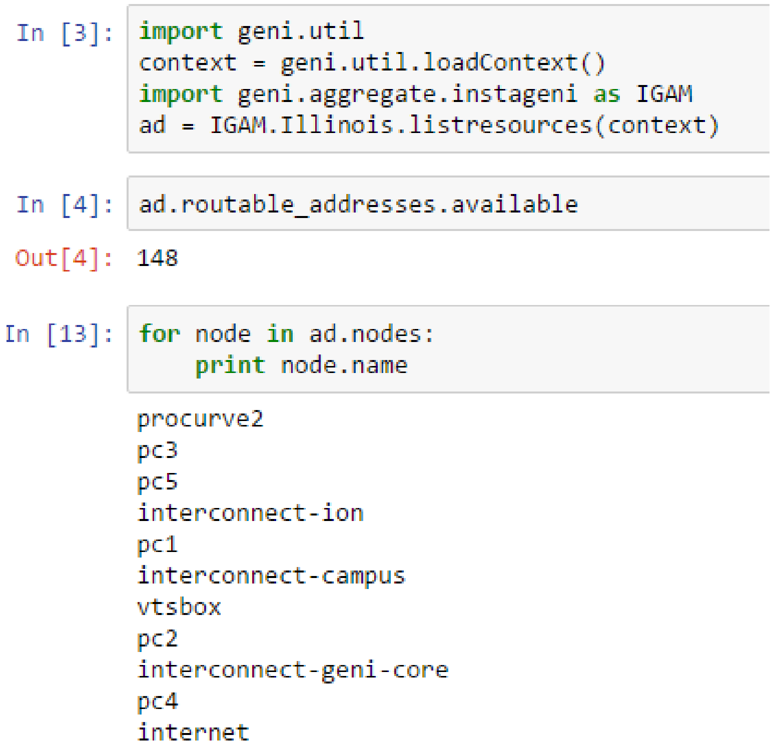
- Allows you to:
  - List resources;
  - Create experiments;
  - Change experiments;
  - Query experiments;
  - …

# Advantages / Disadvantages

- + Very powerful
- + Gives low-level access to the various resources

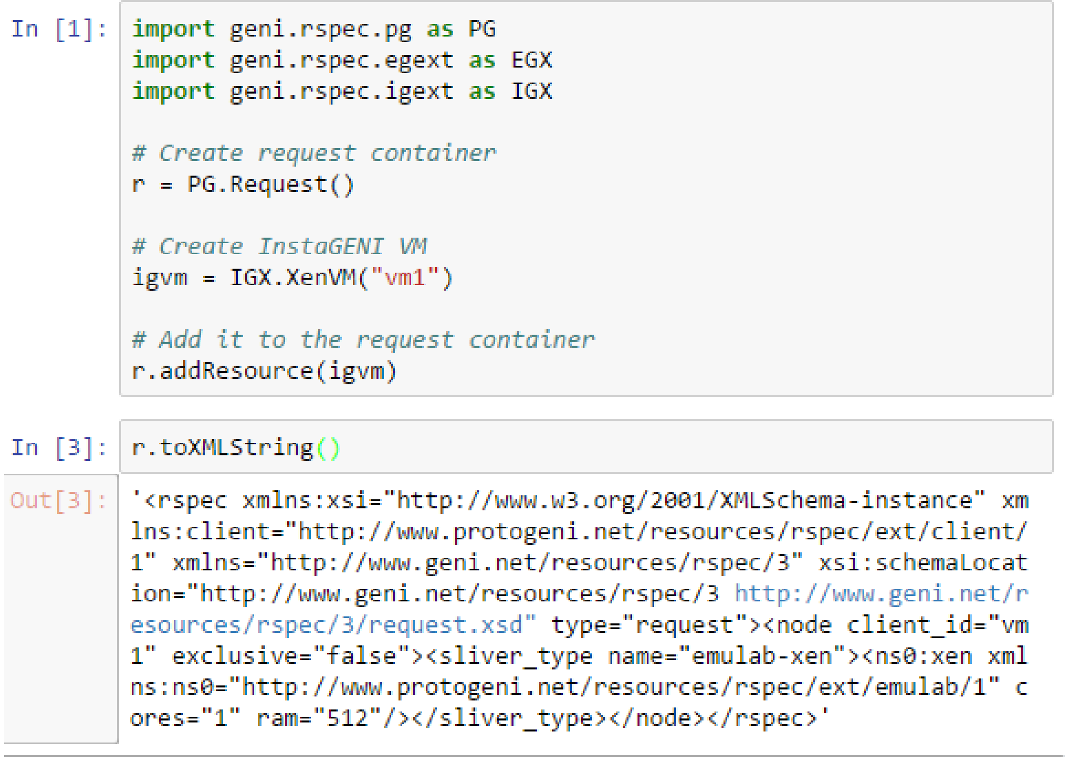
- - Requires a deep understanding of how the GENI AM API works
- - Focused on support of GENI-resources: more manual work needed for using Fed4FIRE resources

# Example: Query available resources



```
In [3]:  import geni.util
         context = geni.util.loadContext()
         import geni.aggregate.instageni as IGAM
         ad = IGAM.Illinois.listresources(context)

In [4]:  ad.routable_addresses.available

Out[4]:  148

In [13]: for node in ad.nodes:
             print node.name

         procurve2
         pc3
         pc5
         interconnect-ion
         pc1
         interconnect-campus
         vtsbox
         pc2
         interconnect-geni-core
         pc4
         internet
```

# Example: Create an request



In [1]:

```
import geni.rspec.pg as PG
import geni.rspec.egext as EGX
import geni.rspec.igext as IGX

# Create request container
r = PG.Request()

# Create InstaGENI VM
igvm = IGX.XenVM("vm1")

# Add it to the request container
r.addResource(igvm)
```

In [3]:

```
r.toXMLString()
```

Out[3]:

```
'<rspec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xm
lns:client="http://www.protogeni.net/resources/rspec/ext/client/
1" xmlns="http://www.geni.net/resources/rspec/3" xsi:schemaLocat
ion="http://www.geni.net/resources/rspec/3 http://www.geni.net/r
esources/rspec/3/request.xsd" type="request"><node client_id="vm
1" exclusive="false"><sliver_type name="emulab-xen"><ns0:xen xml
ns:ns0="http://www.protogeni.net/resources/rspec/ext/emulab/1" c
ores="1" ram="512"/></sliver_type></node></rspec>'
```

# Additional resources on geni-lib

**Documentation**:

https://geni-lib.readthedocs.io

**Tutorial on scaling**:

http://groups.geni.net/geni/wiki/GEC21Agenda/ScalingUp

FED4FIRE

Co-funded by the
European Union

Co-funded by the
Swiss Confederation

This project has received funding from the European Union’s Horizon 2020 research and innovation programme, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation, under grant agreement No 732638.

WWW.FED4FIRE.EU