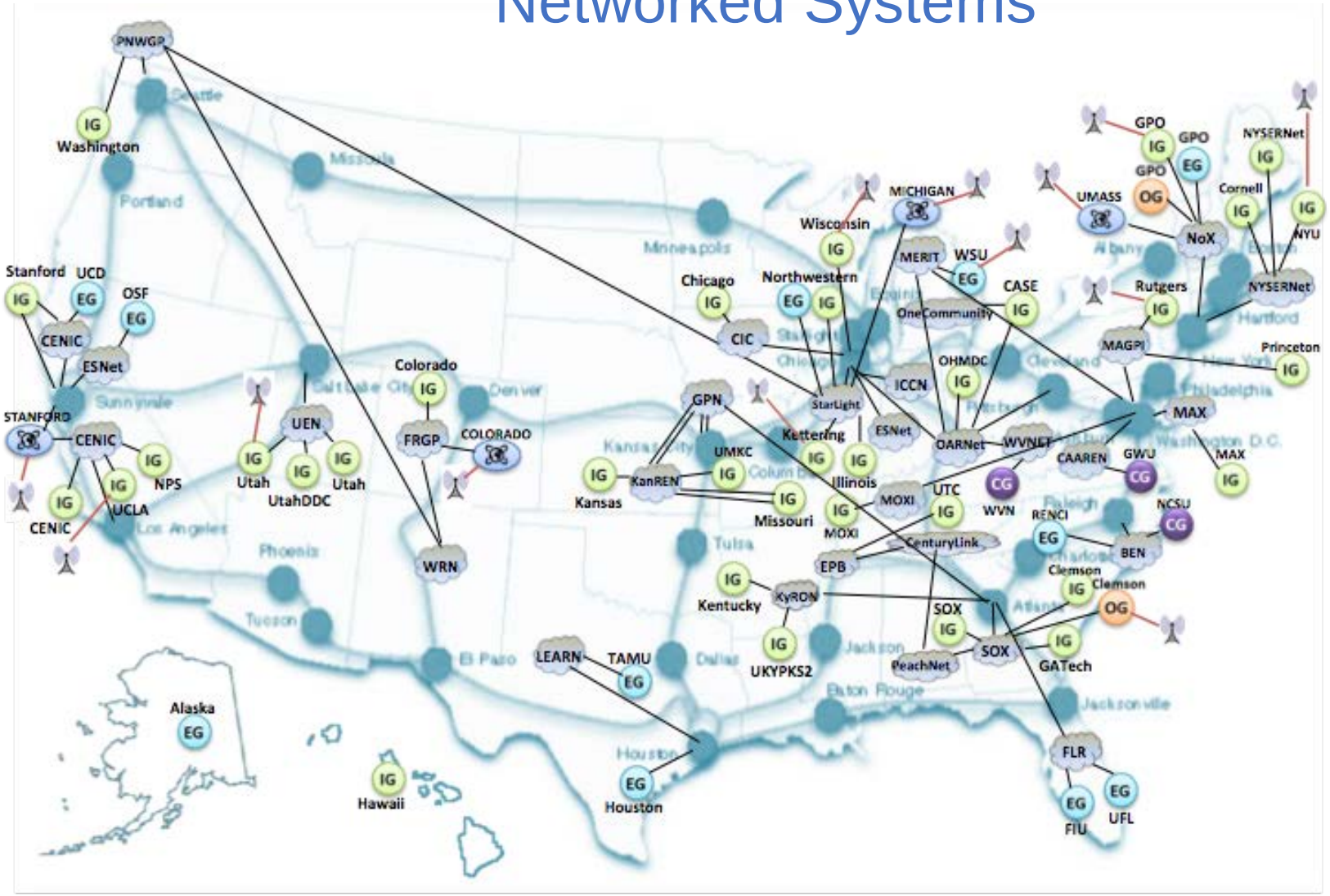
Running Workflows on Educational Clouds
Part 1: ExoGENI

George Papadimitriou

September 10th, 2020

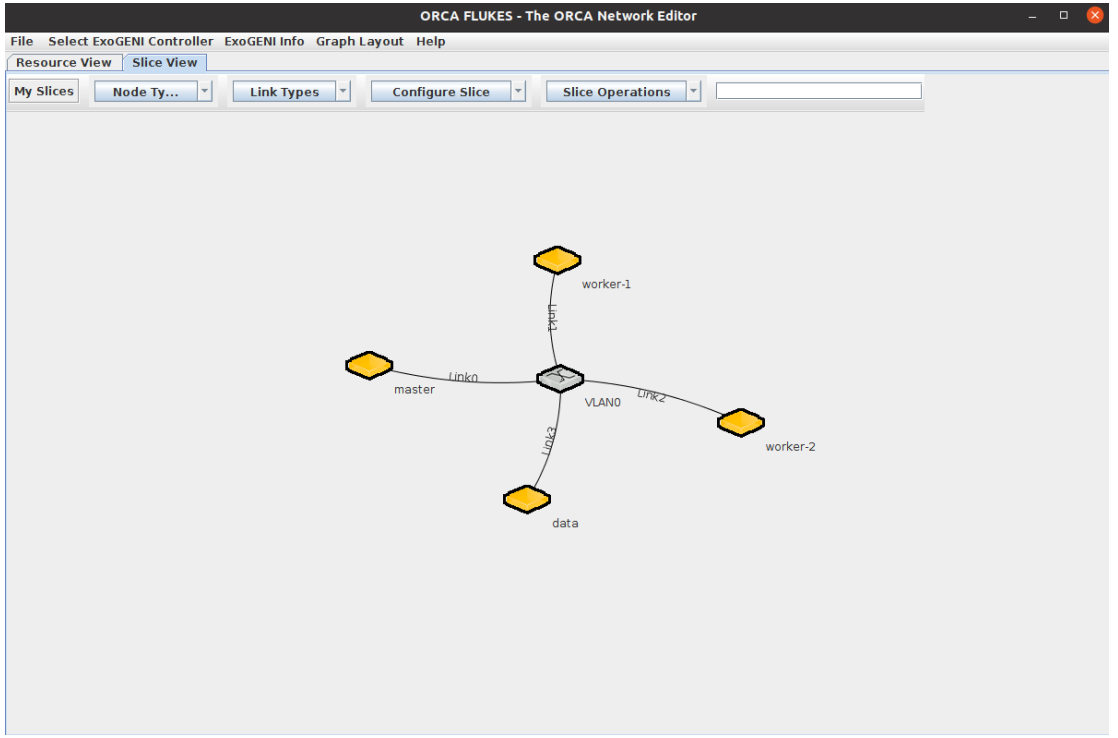
Background – ExoGENI: National Infrastructure for Next-Gen Networked Systems



- GENI Racks at 50+ campuses around the US
- OpenFlow switches for software defined networking

Background – ExoGENI

- ExoGENI is and NSF funded project that provides open infrastructure for networking, distributed systems research and education.
- It allows you to create “slices” that provide network isolation to your transfers
- A Slice is an entity that encapsulates the information of your topology.
    - It contains the compute nodes, OS and node configuration (usually virtual machines)
    - The network links between them and network configuration



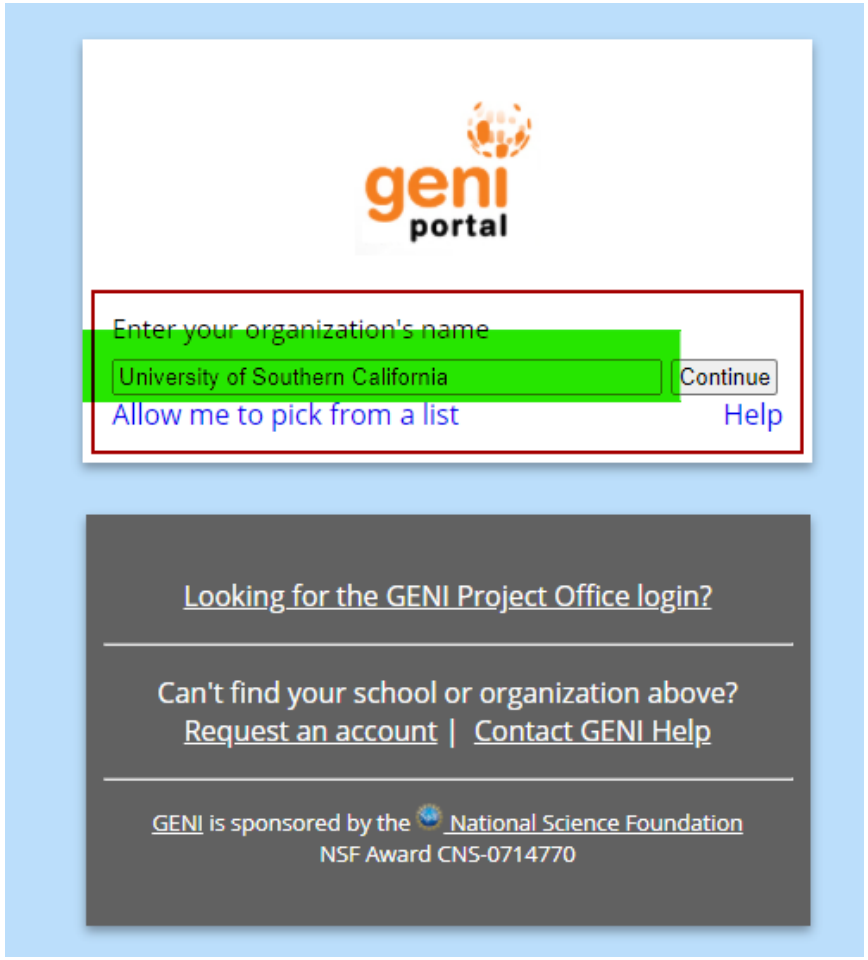
Reference: https://www.geni.net/

ExoGENI – Getting Started (Login)

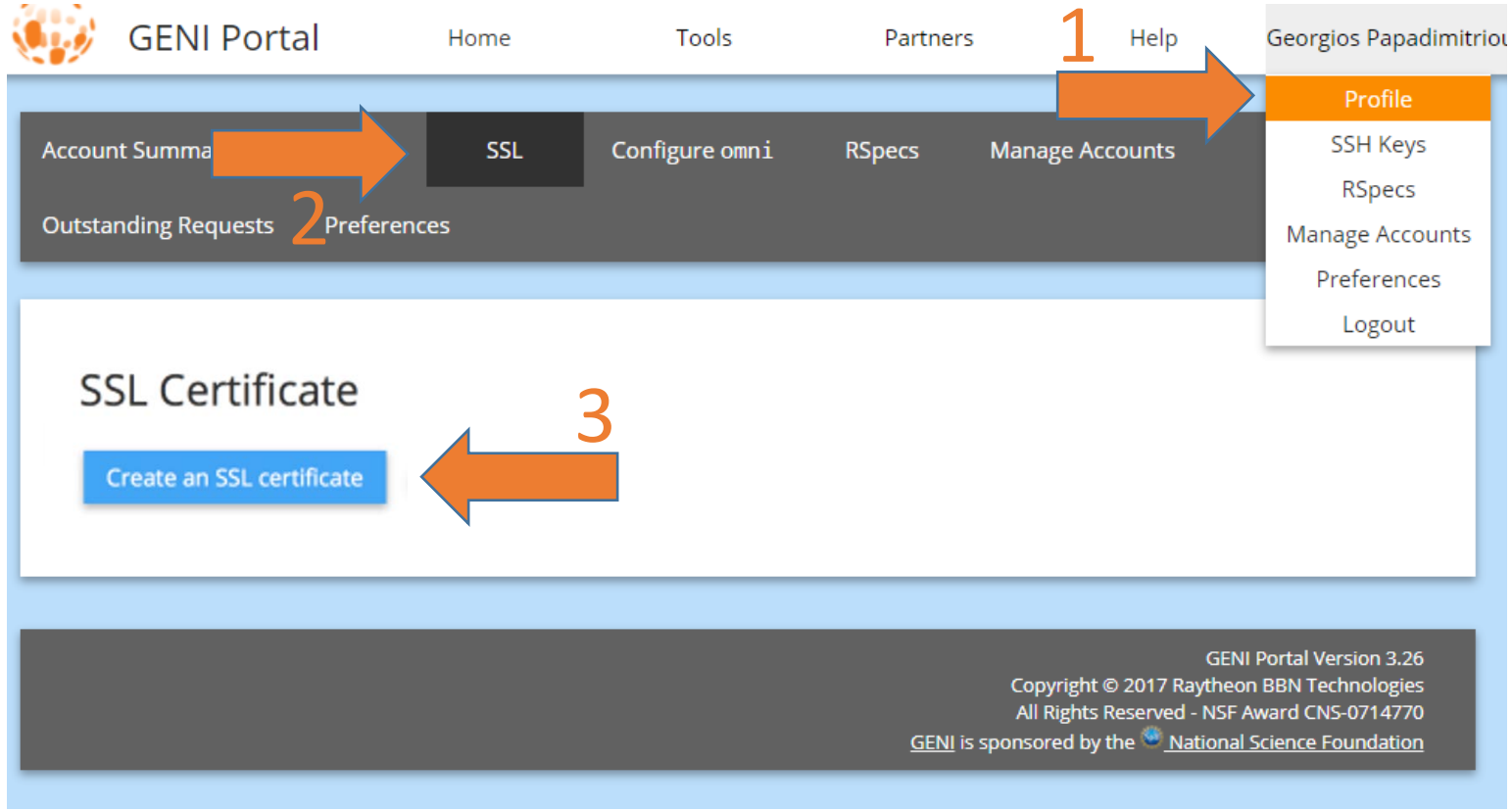
Visit the GENI portal and login using your USC account:

https://portal.geni.net/

These are some of the many resources being used in GENI experiments across the world.

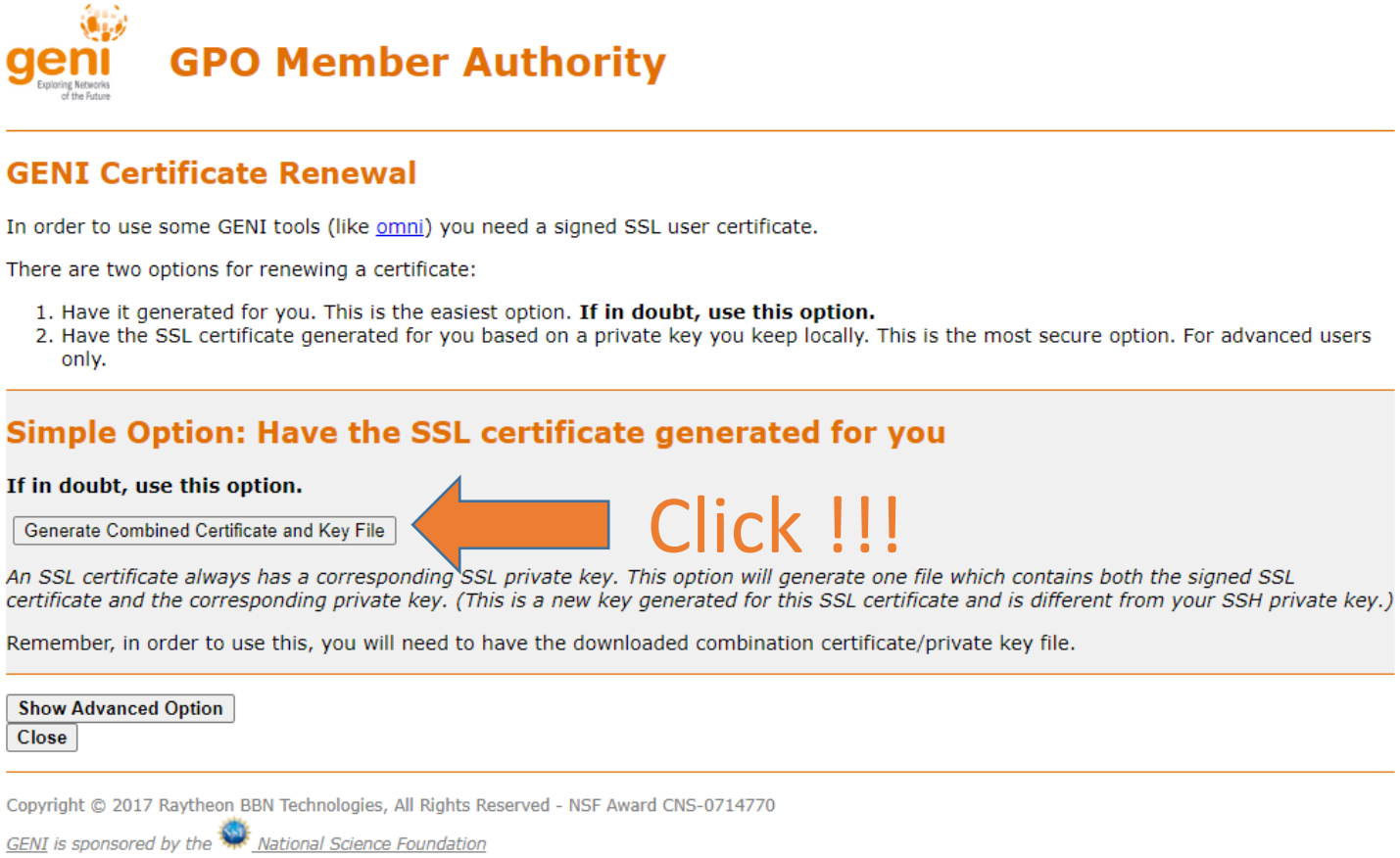


ExoGENI – Getting Started (Generate SSL Keys)



After you login, go to your profile, click on the SSL tab and hit the “Create an SSL certificate” button.

ExoGENI – Getting Started (Generate SSL Keys)

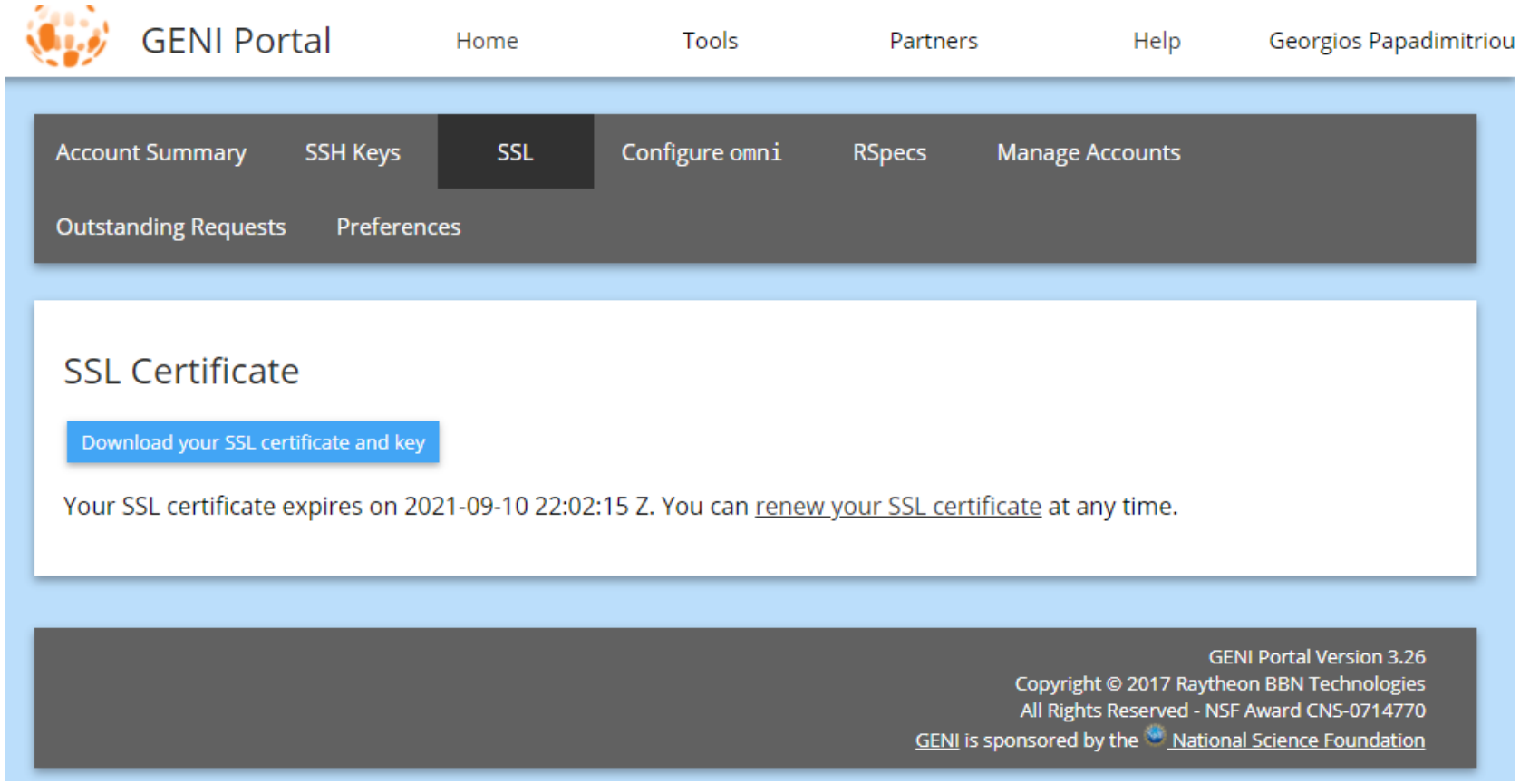


Click on the “Generate Combined Certificate and Key” button.

ExoGENI – Getting Started (Generate SSL Keys)

Click on the “Download” button.
Save the file to a known location (E.g., ~/.ssl/geni-papadimi.pem)

ExoGENI – Getting Started (Generate SSL Keys)



You can find the generated SSL certificate and key if you go back to your profile, under the SSL tab !

ExoGENI – Getting Started (Generate SSH Keys)

```
ssh-keygen -b 4096 -f ~/.ssh/exogeni_rsa
```

If you don’t have already, generate an SSH key (with or without password protection) !

ExoGENI – Getting Started (Install Oracle's Java)

1) Download latest Java from Oracle
   https://www.java.com/en/download/linux_manual.jsp

2) Untar the file into your preferred location
   E.g., tar -xzvf jre-8u261-linux-x64.tar.gz -C ~/Software/Java/

3) Add Java bin to your PATH env variable
   E.g., echo "export PATH=/home/papajim/Software/Java/jre1.8.0_261/bin:\$PATH" >> ~/.bashrc

ExoGENI – Getting Started (Download Flukes)

1) Download latest Flukes (gui to interact with ExoGENI)
   http://geni-images.renci.org/webstart/flukes-0.7.4.zip

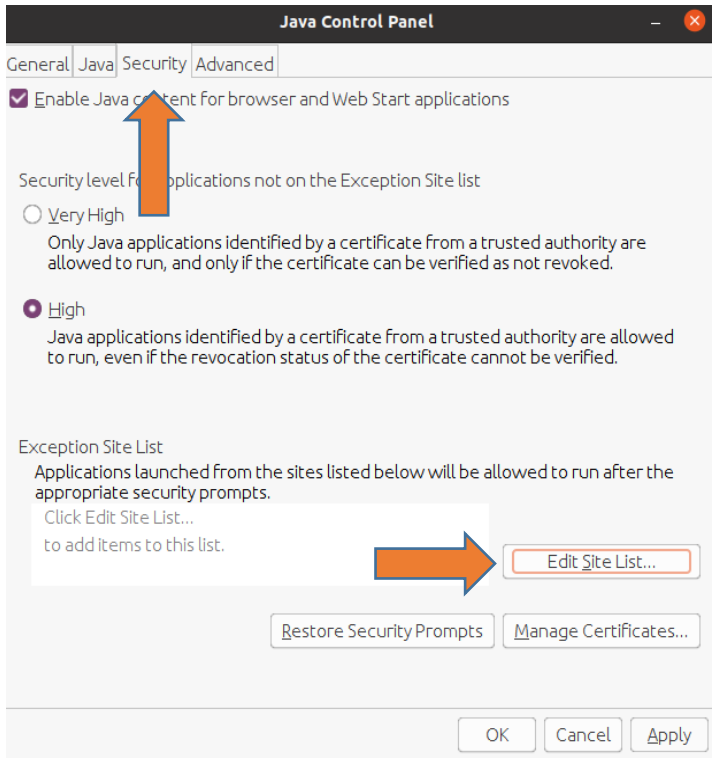
2) Unzip the file into your preferred location
   E.g., cd ~/Software/flukes && unzip flukes-0.7.4.zip

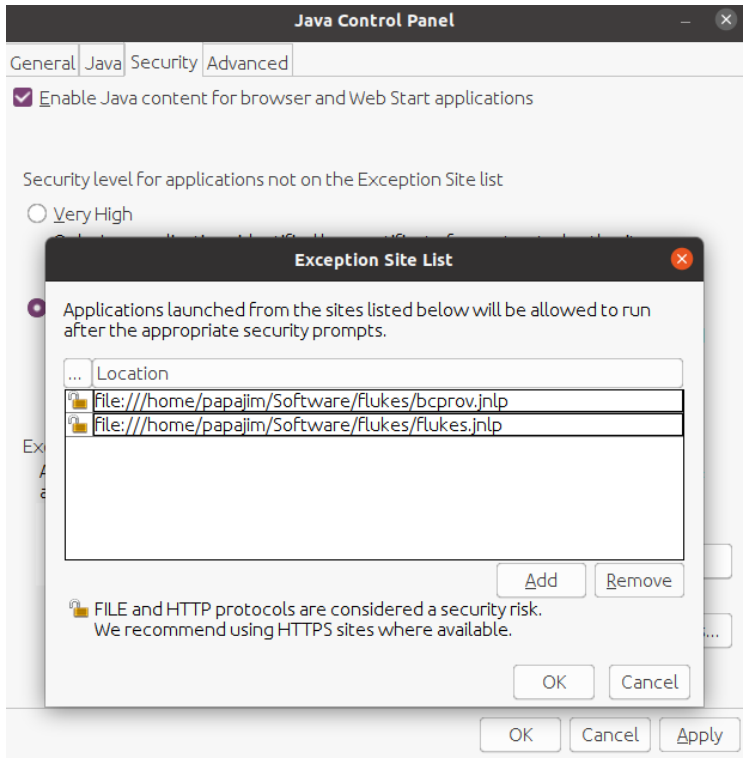
3) Update flukes.jnlp codebase location
   E.g., sed –i 's/.*codebase.*/ codebase="file:\/\/\/home\/papajim\/Software\/flukes"/g' flukes.jnlp

4) Update bcprov.jnlp codebase location
   E.g., sed –i 's/.*codebase.*/ codebase="file:\/\/\/home\/papajim\/Software\/flukes"/g' bcprov.jnlp

ExoGENI – Getting Started (Add Flukes Exception to Java Secutiry)

1) Execute the command “javaws –viewer”
2) Click on the Security Tab
3) Click on Edit Site List
4) Add exceptions for both
   - flukes.jnlp
   - bcprov.jnlp





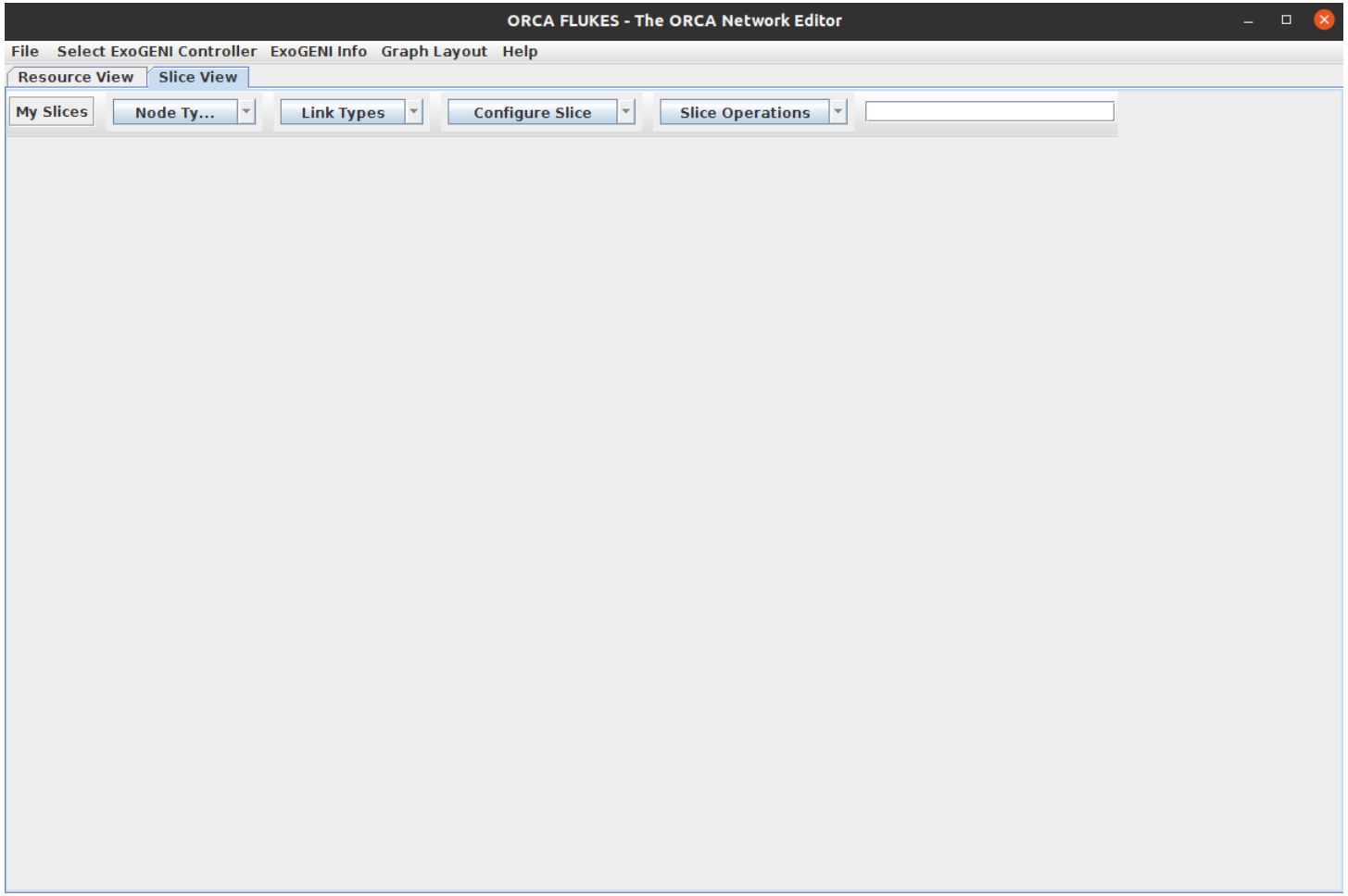
ExoGENI – Getting Started (Add Flukes Properties)

In your home directory (E.g., /home/georgpap) create a file ".flukes.properties" with the following content.

```
orca.xmlrpc.url=https://geni.renci.org:11443/orca/xmlrpc
user.certfile=/home/papajim/.ssl/geni-papadimi.pem
user.certkeyfile=/home/papajim/.ssl/geni-papadimi.pem
enable.modify=true
ssh.key=~/.ssh/exogeni_rsa
# SSH Public key to install into VM instances
ssh.pubkey=~/.ssh/exogeni_rsa.pub
# Path to XTerm executable on your system
xterm.path=/opt/X11/bin/xterm
```

ExoGENI – Getting Started (Run Flukes)

Execute: “javaws flukes.jnlp”



ExoGENI – Getting Started (Creating a Slice)

Demo

Scripts: https://github.com/papajim/exogeni

ExoGENI – Limit Network Speeds

```
#!/bin/bash

interface="ens6"
bandwidth="1000"

tc qdisc add dev $interface root handle 1: htb default 11
tc class add dev $interface parent 1: classid 1:1 htb rate ${bandwidth}mbit
tc class add dev $interface parent 1:1 classid 1:11 htb rate ${bandwidth}mbit
tc qdisc add dev $interface parent 1:11 handle 20: sfq perturb 10
timestamp=$(date '+%d/%m/%Y %H:%M:%S');
echo "[$timestamp] Upload bandwidth limited to $bandwidth Mbits (IF: $interface)"

#tc qdisc del dev $interface root
```

https://github.com/papajim/exogeni/blob/master/limit_network.sh

Network Monitoring - TSTAT

To execute TSTAT use the following:

```
sudo tstat -s /home/panorama/tstat-logs/ -i ens6 -l
```

ExoGENI – Adding Anomalies

Interference	Command
CPU	stress -c THREAD_NUM e.g., stress -c 2
HDD	stress -d THREAD_NUM --hdd-bytes BYTES e.g., stress -d THREAD_NUM --hdd-bytes 50M
Network Loss	tc qdisc add dev DEV root netem loss LOSS e.g., tc qdisc add dev ens6 root netem loss 1%
Network Reordering	tc qdisc add dev DEV root netem delay DELAY reorder REORDER CORRELATION e.g., tc qdisc add dev ens6 root netem delay 10ms reorder 25% 50%

Panorama360 Monitoring - Prerequisites

- HTCondor 8.6+:
  - https://research.cs.wisc.edu/htcondor/downloads/
- Pegasus Panorama:
  - Compile from source: https://github.com/pegasus-isi/pegasus/tree/panorama
  - Pre-compiled binaries: http://download.pegasus.isi.edu/pegasus/4.9.3panorama/
- Docker 17.02+:
  - https://docs.docker.com/install/
- Docker Compose:
  - https://docs.docker.com/compose/

How to Deploy: Monitoring Backend (RabbitMQ, ELK Stack)

- On a host that has Docker and Docker Compose installed, clone https://github.com/Panorama360/data-collection-arch
- Change to the cloned directory and execute the following command:

docker-compose up –d

- **Example:**

```
georgpap@iris:~/GitHub/panorama360/data-collection-arch$ docker-compose up -d
Creating network "panorama_net" with driver "bridge"
Creating panorama-rabbitmq      ... done
Creating panorama-elasticsearch ... done
Creating panorama-kibana        ... done
Creating panorama-logstash      ... done
georgpap@iris:~/GitHub/panorama360/data-collection-arch$
```

How to Deploy: Checking Services (RabbitMQ, ELK Stack)

Now the host should have RabbitMQ, Elasticsearch, Logstash, and Kibana running as Docker containers with their service ports exposed. Try to access them…

- RabbitMQ: http://<hostname or ip>:15672
- Elasticsearch: http://<hostname or ip>:9200
- Logstash: http://<hostname or ip>:9600
- Kibana: http://<hostname or ip>:5601

How to Deploy: Enabling Stampede Events

- In order to get pegasus-monitord to publish **all** of its events to an AMQP endpoint in JSON format, 3 properties must be specified in the workflow's properties file (eg. "pegasus.properties").
  - pegasus.monitord.encoding = json
  - pegasus.catalog.workflow.amqp.url = amqp://[username:password]@<hostname>[:port]/<exchange_name>
  - pegasus.catalog.workflow.amqp.events = stampede.*

- **Example:**

```
19  # Monitord Events
20  pegasus.monitord.encoding=json
21  pegasus.catalog.workflow.amqp.url=amqp://panorama:panorama@amqp.isi.edu:5672/panorama/monitoring
22  pegasus.catalog.workflow.amqp.events = stampede.*
```

- More about stampede events: https://pegasus.isi.edu/documentation/stampede_wf_events.php

How to Deploy: Enabling Transfer Events

- In order to get pegasus-transfer to publish transfer statistics from the Globus Transfer Service to an AMQP endpoint in JSON format, 2 profiles must be specified in the workflow's sites catalog (eg. "sites.xml"), under the site where pegasus-transfer is going to be invoked (eg. "local").
  - env.PEGASUS_TRANSFER_PUBLISH = 1
  - env.PEGASUS_AMQP_URL = amqp://[username:password]@<hostname>[:port]/<exchange_name>
- **Example:**

```
5   <!-- The local site contains information about the submit host -->
6   <site handle="local">
7      <directory type="shared-scratch" path="${PWD}/work/scratch">
8         <file-server operation="all" url="go://georgepap#exo-master/${PWD}/work/scratch"/>
9      </directory>
10     <directory type="local-storage" path="${PWD}/work/outputs">
11        <file-server operation="all" url="go://georgepap#exo-master/${PWD}/work/outputs"/>
12     </directory>
13     <!-- These profiles tell pegasus-transfer to publish stats to an AMQP endpoint -->
14     <profile namespace="env" key="PEGASUS_TRANSFER_PUBLISH">1</profile>
15     <profile namespace="env" key="PEGASUS_AMQP_URL">amqp://panorama:panorama@amqp.isi.edu:5672/panorama/monitoring</profile>
16  </site>
```

How to Deploy: Enabling Kickstart Online Traces

- In order to get pegasus-kickstart to publish traces of resource usage statistics to an AMQP endpoint in JSON format, 2 profiles must be specified in the workflow's sites catalog (eg. "sites.xml") under the compute site.
  - pegasus.gridstart.arguments = -m <interval in seconds>
  - env.KICKSTART_MON_URL = rabbitmq://[USERNAME:PASSWORD]@<hostname>[:port]/api/exchanges/<exchange_name>/publish
- **Example:**

```
23  <site handle="condorpool" arch="x86_64" os="LINUX">
24    <!-- These profiles tell Pegasus that the site is a plain Condor pool -->
25    <profile namespace="pegasus" key="style">condor</profile>
26    <profile namespace="condor" key="universe">vanilla</profile>
27    <!-- These profiles tell pegasus-kickstart to publish stats to an AMQP endpoint -->
28    <profile namespace="pegasus" key="gridstart.arguments">-m 10</profile>
29    <profile namespace="env" key="KICKSTART_MON_URL">rabbitmq://panorama:panorama@amqp.isi.edu:15672/api/exchanges/panorama/monitoring/publish</profile>
30  </site>
```

- Alternatively if we want to customize the monitoring interval per computational task we can specify the profile in the workflow's transformation catalog (eg. "tx.txt")

How to Deploy: Enabling Kickstart Online Traces (MPI Jobs)

- Usually MPI jobs are not launched by Pegasus-Kickstart. Thus, adding the gridstart.arguments profile doesn't have any effect.
- We can work around this by using a wrapper script for the MPI job, that invokes directly pegasus-monitor.
- We still need to specify **KICKSTART_MON_URL** in the sites catalog.

- **Example:**

```
1  #!/usr/bin/env bash
2
3  cd $PEGASUS_SCRATCH_DIR
4
5  mpirun pegasus-monitor -i 10 /shared/software/NAMD_2.12_Linux-x86_64-MPI/namd2 $@
6
```

How to Deploy: Enabling Darshan Logs (MPI Jobs)

- In case your MPI application wasn't compiled and statically linked with Darshan's library, we need to set a profile in the transformation catalog, adding the path of the library to LD_PRELOAD.

- We launch the application using a **wrapper script**, and as post job steps:
  - Build the darshan log path from the environmental variables
  - Invoke **pegasus-darshan** with the files as input

```
tr namd {
    site local-slurm {
        pfn "/shared/software/wrappers/namd_wrapper_slurm.sh"
        arch "x86_64"
        os "LINUX"
        type "INSTALLED"
        profile pegasus "exitcode.successmsg" "End of program"
        profile pegasus "memory" "2500M"
        profile globus "jobtype" "single"
        profile env "LD_PRELOAD" "/shared/software/darshan-3.1.6/lib/libdarshan.so"
    }
}
```

Transformation Catalog

```
#!/usr/bin/env bash

cd $PEGASUS_SCRATCH_DIR

mpirun pegasus-monitor -i 10 /shared/software/NAMD_2.12_Linux-x86_64-MPI/namd2 $@

#post job parse darshan output
DAY=$(date '+%d')
DAY=${DAY##0}
MONTH=$(date '+%m')
MONTH=${MONTH##0}
YEAR=$(date '+%Y')

darshan_base=/shared/darshan-logs/${YEAR}/${MONTH}/${DAY}
darshan_file=${darshan_base}/${SLURM_JOB_USER}_namd2_id${SLURM_JOB_ID}_*.darshan

for f in $darshan_file; do
   $PEGASUS_HOME/bin/pegasus-darshan -f "$f"
done
```

Wrapper Shell Script

How to Deploy: Enabling Darshan Logs (MPI Jobs)

- pegasus-darshan will output in stdout a monitoring payload, that will be picked by pegasus-monitord, which in its turn will publish it to the AMQP endpoint.
- *This can also be used as a generic way of adding new tools to this architecture*.


```
@@@MONITORING_PAYLOAD - START@@@
{
  &quot;monitoring_event&quot;: &quot;darshan.perf&quot;,
  &quot;payload&quot;: [
    {
      &quot;POSIX_module_data&quot;: {
        &quot;agg_perf_by_cumul&quot;: 14.667417,
        &quot;agg_perf_by_open&quot;: 14.667417,
        &quot;agg_perf_by_open_lastio&quot;: 14.667417,
        &quot;agg_perf_by_slowest&quot;: 14.667417,
        &quot;shared_files&quot;: {
          &quot;time_by_cumul_io_only&quot;: 0.0,
          &quot;time_by_cumul_meta_only&quot;: 0.0,
          &quot;time_by_open&quot;: 0.0,
          &quot;time_by_open_lastio&quot;: 0.0,
          &quot;time_by_slowest&quot;: 0.0
        },
        &quot;total_bytes&quot;: 403761,
        &quot;unique_files&quot;: {
          &quot;slowest_rank&quot;: 0.0,
          &quot;slowest_rank_io_time&quot;: 0.026253,
          &quot;slowest_rank_meta_only_time&quot;: 0.023997
        }
      }
    }
  ],
  &quot;ts&quot;: 1552878285
}
@@@MONITORING_PAYLOAD - END@@@
```

panorama360

USC Viterbi
School of Engineering
Department of Computer Science

Thank you!

George Papadimitriou
Computer Science PhD Student
University of Southern California

email: georgpap@isi.edu