Lab 9: TritonTube - Fault Tolerance with etcd, AWS

Due: June 6, 2025 (Friday) 11:59 PM

Lab 9 design document

Updates / FAQ

- June 2: Initial version of write-up
- June 2: Added a few statements to assist in the lab.

Gradescope form to submit your work

https://www.gradescope.com/courses/1001714/assignments/6302539

Overview

In Lab 9 you are going to gain first-hand experience (1) setting up virtual Linux machines on the AWS cloud platform, (2) setting up a three-node ‘etcd’ cluster, and (3) observing the behavior of etcd under a simulated failure condition.

Note that this lab is a “scaled down” version of the original Lab 9 idea. At the end of this write-up is information on how you could optionally extend the lab to support fault-tolerant

metadata storage in TritonTube. This might be useful if you're showing the work to a recruiter or similar to demonstrate some of the skills/experiences you picked up this term.

The main website for etcd is https://etcd.io/ and an overview of etcd is located at https://github.com/etcd-io/etcd

Setting up your environment - VMs

Begin by visiting https://awsed.ucsd.edu

You are going to create 3 (three) VMs running **Debian Linux** using the **t2.micro** instance type.

UCSD has put together some instructions on how to do this:

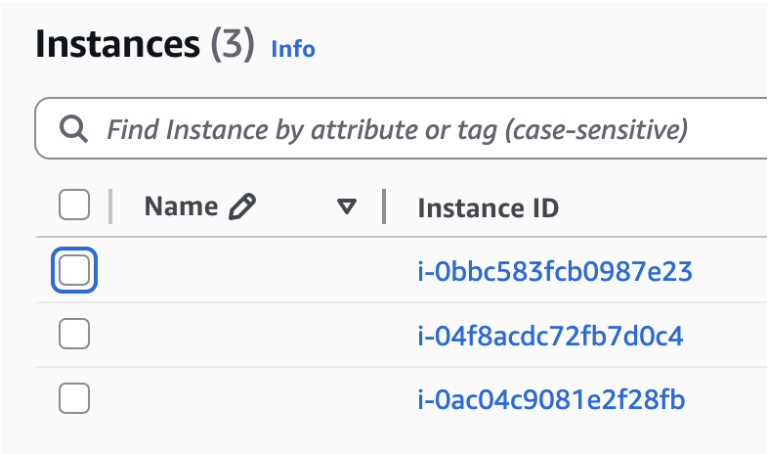
https://support.ucsd.edu/services?id=kb_article_view&sysparm_article=KB0032513

You will be using the *real* EC2 interface, which is the same one that major companies and large-scale sites use. So you'll get experience with a real system, but because it isn't designed for students or learners, it can be a bit confusing and intimidating. So do reach out with any questions you have.

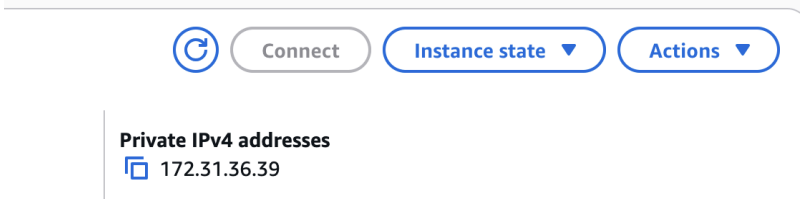
The main thing is to remember to stop/terminate your instances after you're done using them, because there is a small charge while they execute. If you create a "t2.micro" instance, the cost is pretty small (just a penny or two per hour) but if you forget to turn it off it can cause some issues later.

Finally, at NO TIME should you ever put a credit card number anywhere. No credit cards! Campus has provided you with $10 to use. Don't associate your personal information with this system. That way, there is zero chance you could personally be on the hook for any charges.

Once you create your three instances, you'll see them in the "instances" tab on the left side of the screen. Each instance is given an ID, similar to this:

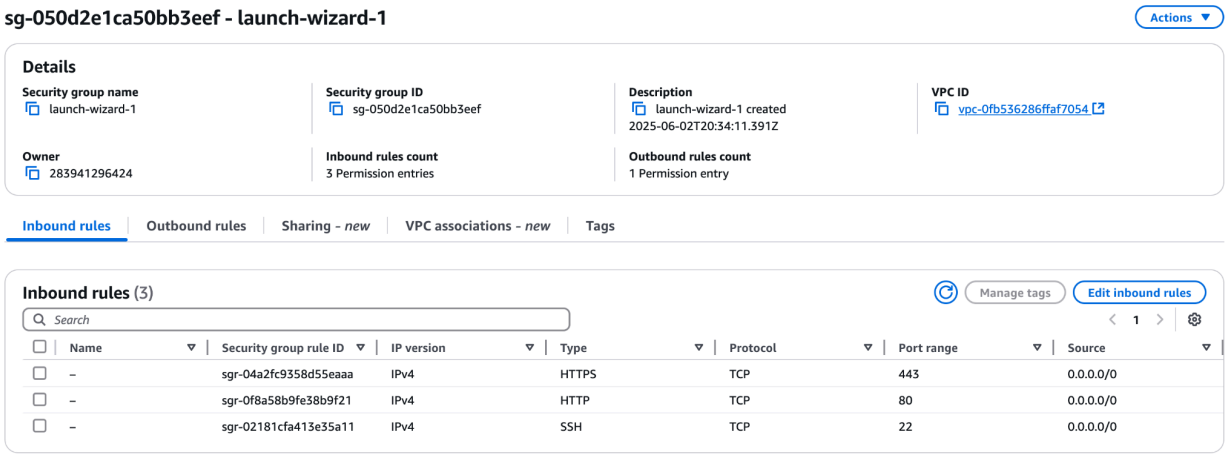


When you click on the Instance ID, it shows you quite a bit of information about the VM, including its two main IP addresses. Each VM you create has an “internal” address and an “external” or “public” address. Internal addresses are useful for communicating between the VMs in the cloud, whereas the public addresses are used to communicate with endpoints elsewhere on the Internet. When you click the instance ID, towards the top right you’ll see the “Private IPv4 address”, similar to this:

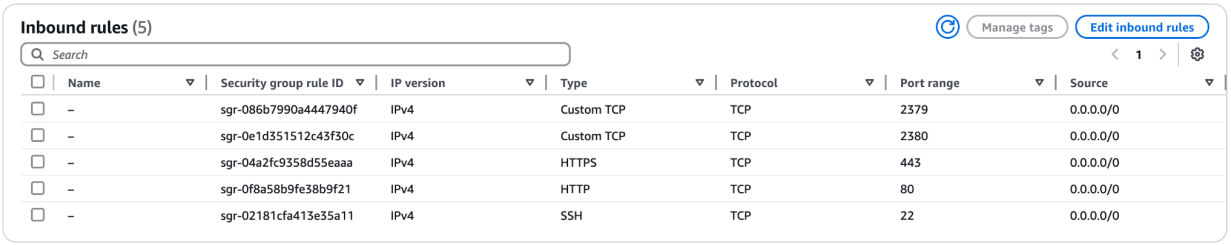


Setting up your environment - Security group

You are going to setup and run etcd, which requires two TCP ports to operate correctly: 2379 and 2380. However, by default, when you launch a VM it has a *security group* which defines firewall rules that block certain ports. We will need to expose those ports for your VMs to communicate with each other. To do that, click the “Network & Security” tab, then click the link for the security group, and you should see a page similar to this:



To add the two rules you'll need, click the "Edit inbound rules" button, then "Add rule", then as a Custom TCP rule, put in the relevant port and choose the CIDR address 0.0.0.0/0 (which allows TCP connections on the given port from any IP address). Once you've added both ports, your display should look like this:



If all of your three VMs use the same security group, all three of them should have these ports available for use to communicate to the other VMs.

Setting up your environment - Installing etcd

From the etcd website, there is a tutorial on installing the software on three nodes. We are going to follow this to get our installation set up:

https://etcd.io/docs/v3.5/tutorials/how-to-setup-cluster/

To download and install etcd, visit https://github.com/etcd-io/etcd/releases/tag/v3.5.21 and download the etcd-v3.5.21-linux-amd64.tar.gz program to your VM. Unpack it by running "tar -xzvf etcd-v3.5.21-linux-amd64.tar.gz". Change into that directory, and make sure the etcd and etcdctl programs are in your path. You can do that by running "export PATH=$PATH:`pwd`" from the directory that contains those binaries.

On each machine, you'll want to create a script called setupenv.sh that contains this information:

```
export TOKEN=token-01
```

```
export CLUSTER_STATE=new
export NAME_1=machine-1
export NAME_2=machine-2
export NAME_3=machine-3
export HOST_1=10.240.0.17 (CHANGE THIS)
export HOST_2=10.240.0.18 (CHANGE THIS)
export HOST_3=10.240.0.19 (CHANGE THIS)
export
CLUSTER=${NAME_1}=http://${HOST_1}:2380,${NAME_2}=http://${HOST_2}:2380,${NAME_3
}=http://${HOST_3}:2380
```

You'll need to set HOST_1, HOST_2, and HOST_3 to be the internal IPv4 addresses assigned to your VMs (that you saw above). You can then run:

$ source setupenv.sh

On each machine, run the appropriate commands to start up your three-node etcd cluster:

```
# For machine 1
THIS_NAME=${NAME_1}
THIS_IP=${HOST_1}
etcd --data-dir=data.etcd --name ${THIS_NAME} \
    --initial-advertise-peer-urls http://${THIS_IP}:2380 --listen-peer-urls http://${THIS_IP}:2380 \
    --advertise-client-urls http://${THIS_IP}:2379 --listen-client-urls http://${THIS_IP}:2379 \
    --initial-cluster ${CLUSTER} \
    --initial-cluster-state ${CLUSTER_STATE} --initial-cluster-token ${TOKEN}

# For machine 2
THIS_NAME=${NAME_2}
THIS_IP=${HOST_2}
etcd --data-dir=data.etcd --name ${THIS_NAME} \
    --initial-advertise-peer-urls http://${THIS_IP}:2380 --listen-peer-urls http://${THIS_IP}:2380 \
    --advertise-client-urls http://${THIS_IP}:2379 --listen-client-urls http://${THIS_IP}:2379 \
    --initial-cluster ${CLUSTER} \
    --initial-cluster-state ${CLUSTER_STATE} --initial-cluster-token ${TOKEN}

# For machine 3
THIS_NAME=${NAME_3}
THIS_IP=${HOST_3}
etcd --data-dir=data.etcd --name ${THIS_NAME} \
    --initial-advertise-peer-urls http://${THIS_IP}:2380 --listen-peer-urls http://${THIS_IP}:2380 \
    --advertise-client-urls http://${THIS_IP}:2379 --listen-client-urls http://${THIS_IP}:2379 \
    --initial-cluster ${CLUSTER} \
    --initial-cluster-state ${CLUSTER_STATE} --initial-cluster-token ${TOKEN}
```

Interacting with your cluster

```shell
export HOST_1=10.240.0.17 (CHANGE THIS)
export HOST_2=10.240.0.18 (CHANGE THIS)
export HOST_3=10.240.0.19 (CHANGE THIS)
export ENDPOINTS=$HOST_1:2379,$HOST_2:2379,$HOST_3:2379

$ etcdctl --endpoints=$ENDPOINTS member list
```

Note that you're going to run each command on a separate VM. You can just have three terminals open to each VM, where you ssh into each node. Once you've started those three nodes, you should then log into any of your VMs and run the etcdctl command to interact with and manipulate your etcd cluster. When you log in again, you may need to add the etcd and etcdctl programs to your path again.

A set of commands you can use to interact with your cluster is available at https://etcd.io/docs/v3.5/tutorials/how-to-check-cluster-status/

In particular, we're going to be building a table of the status of the cluster. To do this, ensure that the ENDPOINTS environment variable is setup as above, and run

```shell
$ etcdctl --write-out=table --endpoints=$ENDPOINTS endpoint
status
```

You should see a table similar to:

```
+-----------------+-----------------+---------+---------+-----------+------------+-----------+-------
|    ENDPOINT     |        ID       | VERSION | DB SIZE | IS LEADER | IS LEARNER | RAFT TERM | RAFT I
+-----------------+-----------------+---------+---------+-----------+------------+-----------+-------
| 10.240.0.17:2379 | 4917a7ab173fabe7 |  3.5.0  |   45 kB |      true |      false |         4 |      1
| 10.240.0.18:2379 | 59796ba9cd1bcd72 |  3.5.0  |   45 kB |     false |      false |         4 |      1
| 10.240.0.19:2379 | 94df724b66343e6c |  3.5.0  |   45 kB |     false |      false |         4 |      1
+-----------------+-----------------+---------+---------+-----------+------------+-----------+-------
```

This will show you which node is the leader, which RAFT term the system is in, etc. Review this information and familiarize yourself with the different fields.

Experimenting with emulated failures

Once your cluster is set up, you're going to use etcdctl's "put" command to set a key/value pair in your cluster. Recall that the whole point of etcd is to store mutable state in a fault tolerant manner. Make sure to pass the –endpoints=$ENDPOINTS command line option to ensure etcdctl can find your cluster.

Next, you're going to use the AWS web interface to select the leader node, and you'll set the VM's "Instance state" to "stopped", which shuts the VM down. You'll use etcdctl's status command to show which node took over as the new leader (and note any change in the RAFT term). You can "get" the key value pair you stored earlier to verify that it was not lost. Finally, you can restart the VM, and restart etcd on that node, and see it rejoin the cluster.

You may refer to the https://etcd.io/docs/v3.5/tutorials/writing-to-etcd/ and https://etcd.io/docs/v3.5/tutorials/reading-from-etcd/ for the syntax.

Documenting the experiment

We set up a form in Gradescope that will walk you through the experiment. You will paste information from your VM and etcd environment into this form, and paste the output of the etcdctl commands into that form. This is all you have to do. There is no autograder–you are simply carrying out these steps and recording some results from your experiments.

(optional) Extending TritonTube to use etcd

The steps above are all that is required for Lab 9. However, to fully realize the benefits of RAFT for TritonTube, we encourage you to optionally extend the Metadata interface in your TritonTube server to store the metadata in etcd. Go includes a client that makes this pretty easy. A short example of using the client code is available at https://github.com/etcd-io/etcd/blob/main/client/v3/README.md

Asking for and receiving help

For this project, you are required to go through these steps in your own AWS account. However, we encourage you to help your fellow students with how to setup VMs, how to install etcd and work with Linux, and how to do the experiment. So the main thing is getting to the point where you can do the steps outlined in the GradeScope form submission. So asking for and giving help is fine for lab 9.

###