Create new router using open ziti

1. Install binary into the ziti home directory

source /dev/stdin <<< "$(wget -qO- https://raw.githubusercontent.com/openziti/ziti/db71b1a4a6d70feff70cde3962d2c9f9148a0dd5/quickstart/docker/image/ziti-cli-functions.sh)" getZiti

1. Create config.yaml in the binary installed directory like ZITI\_HOME=~/.ziti/quickstart/VM-hstname/ziti-bin/ziti-v0.27.5

Example of config.yaml

v: 3

identity:

cert: "certs/identity.cert.pem"

server\_cert: "certs/internal.chain.cert.pem"

key: "certs/internal.key.pem"

ca: "certs/intermediate-chain.pem"

edge:

csr:

country: US

locality: Charlotte

organization: Netfoundry

organizationalUnit: ADV-DEV

#province: NC

sans:

dns:

- "ubuntu-s-1vcpu-2gb-amd-blr1-01" # local VM Host name

- "localhost"

ip:

- "127.0.0.1"

- "10.47.0.7" # private IP of the local VM on eth0

ctrl:

endpoint: tls:165.232.177.92:8440

# 165.232.177.92 controller public IP, 8440 is ZITI\_CTRL\_PORT

link:

dialers:

- binding: transport

listeners:

- binding: edge

address: tls:0.0.0.0:8444

options:

advertise: 64.227.162.243:8444

# 64.227.162.243 ER public IP 8444 is edge router listener port from controller

maxQueuedConnects: 50

maxOutstandingConnects: 100

connectTimeoutMs: 3000

- binding: tunnel

options:

mode: host #tproxy|host

svcPollRate: 15s

resolver: udp://10.47.0.7:53 #10.47.0.7 is ER private IP

dnsSvcIpRange: 100.64.0.1/10

lanIf: eth0 #Lan interface name

forwarder:

latencyProbeInterval: 10

xgressDialQueueLength: 1000

xgressDialWorkerCount: 128

linkDialQueueLength: 1000

linkDialWorkerCount: 32

1. Create the new folder to install the cert file “mkdir certs” in the ZITI\_HOME path
2. ./ziti edge login “CONTROLLER PUB\_IP”:8441.Where 8441 is the ZITI\_EDGE\_CONTROLLER\_PORT. You have to provide the user name and password which you created at the time of controller installation default username is admin
3. Edit the hosts file in the ER using “vi /etc/hosts add controller public IP to resolve the controller hostname as public IP of the controller like “165.232.177.92 ubuntu-s-2vcpu-2gb-amd-blr1-01” 165.232.177.92 is Public IP of CTRL and ubuntu-s-2vcpu-2gb-amd-blr1-01 Hostname of the CTRL.
4. Create the Edge router using new-router as the name of ER and enroll.jwt is the name of jwt file “./ziti edge create edge-router new-router -t -o enroll.jwt”
5. Assign edge router attributes ./ziti edge update edge-router new-router -a private
6. Register the identity create above enroll.jwt and config.yaml “./ziti-router enroll config.yaml --jwt enroll.jwt”
7. Update the identity using “./ziti edge update identity new-router -a hosts
8. Verify the create edge router using “./ziti edge list edge-routers”
9. Now run the ER using “./ziti-router run config.yaml”

*# 1. Create an identity for the HTTP client and assign an attribute "http-clients". We'll use this attribute when authorizing the clients to*  
*# access the HTTP service*  
ziti edge create identity user http-client -a 'http-clients' -o http.client.jwt   
  
*#2. Create an identity for the HTTP server if you are not using an edge-router with the tunneling option enabled*  
ziti edge create identity user http-server -o http.server.jwt  
  
*#3. Create an intercept.v1 config. This config is used to instruct the client-side tunneler how to correctly intercept*   
*# the targeted traffic and put it onto the overlay.*  
ziti edge create config http.intercept.v1 intercept.v1 '{"protocols":["tcp"],"addresses":["http.ziti"], "portRanges":[{"low":80, "high":80}]}'  
   
*#4. Create a host.v1 config. This config is used instruct the server-side tunneler how to offload the traffic from*   
*# the overlay, back to the underlay.*   
ziti edge create config http.host.v1 host.v1 '{"protocol":"tcp", "address":"'"${http\_server}"'", "port":80}'  
   
#5. Create a service to associate the two configs created previously into a service.  
ziti edge create service http.svc --configs http.intercept.v1,http.host.v1  
  
#6. Create a service-policy to authorize "HTTP Clients" to "dial" the service representing the HTTP server.  
ziti edge create service-policy http.policy.dial Dial --service-roles "@http.svc" --identity-roles '*#http-clients'*  
  
*#7. Create a service-policy to authorize the "HTTP Server" to "bind" the service representing the HTTP server.*  
ziti edge create service-policy http.policy.bind Bind --service-roles '@http.svc' --identity-roles "@${http\_server\_id}"  
  
*#8. Start the server-side tunneller (unless using the docker-compose quickstart) with the HTTP server identity.*  
*# [optional] if you don't use an edge-router as your tunneler, you will need to download and run the tunneler for your OS*  
*# if you are using a ziti-router, skip to step 9 below*  
*#*   
*# This step is dependant on platform. For this demo we'll be using a virtual machine running linux and we'll be using the*  
*# ziti-edge-tunnel binary. Copy the http.server.jwt from step 2 to the server machine. For the example we'll use /tmp/http.server.jwt*  
*#*  
*# enroll the server identity using ziti-edge-tunnel*  
./ziti-edge-tunnel enroll --jwt /tmp/http.server.jwt --identity /tmp/http.server.json  
*# run ziti-edge-tunnel for the client*  
sudo ./ziti-edge-tunnel run -i /tmp/http.server.json  
  
*#9. Start the client-side tunneller using the HTTP client identity.*  
*# This step is dependant on platform. For this demo we'll be using a virtual machine running linux and we'll be using the*  
*# ziti-edge-tunnel binary. Copy the http.client.jwt from step 1 to the client machine. For the example we'll use /tmp/http.client.jwt*  
*#*  
*# enroll the client identity using ziti-edge-tunnel*  
./ziti-edge-tunnel enroll --jwt /tmp/http.client.jwt --identity /tmp/http.client.json  
*# run ziti-edge-tunnel for the client*  
sudo ./ziti-edge-tunnel run -i /tmp/http.client.json