Deployment scenario

- Using a node in sandbox for DAPPs development
- Deploy a node and a baker for personal baking

How, Why ?

Compiling from source is *not* recommended for deployment

- Need to learn the Ocaml tool chain
- Need to deal with changing dependencies
- Need to follow ongoing development

Tools we are going to use

- Docker : containers
- Ansible : provisioning
- Snap : Binary distribution

None of this is officially documented.

Docker images

From the beginning Tezos provides docker images.

We have 3 flavors

- tezos/tezos : image with shell script entrypoint
- tezos/tezos-debug : similar to `tezos/tezos` but with debug symbols
- tezos/tezos-bare : image with direct access to Tezos binaries

These are created by the gitlab CI and constantly kept up-to-date

Inspect gives us same basic information

```
$ docker pull tezos/tezos-bare:master`
$ docker image inspect tezos/tezos-bare:master \
  | jq '.[].Config.Labels'`
```


```
{ "arch": "x86_64",
  "distro": "alpine",
  "distro_long": "alpine-3.10",
  "maintainer": "contact@nomadic-labs.com",
  "operatingsystem": "linux",
  "org.label-schema.build-image":
    "registry.gitlab.com/tezos/opam-repository:5f775f195",
  "org.label-schema.description": "Tezos node",
  "org.label-schema.docker.schema-version": "1.0",
  "org.label-schema.name": "Tezos",
  "org.label-schema.url": "https://www.nomadic-labs.com",
  "org.label-schema.vcs-ref": "0ce1742ef2",
  "org.label-schema.vcs-url": "https://gitlab.com/tezos/tezos",
  "org.label-schema.vendor": "Nomadic Labs" }
```

In particular : **build ref** and **vcs ref**

- commit from which the image is built
  https://gitlab.com/tezos/tezos/commit/0ce1742ef2
- base build image
  registry.gitlab.com/tezos/opam-repository:5f775f195983dc7adb0c

These two information are useful when reporting bugs.

How to recreate these images

From the source code, you can re-create these images:

- The script `scripts/create_docker_image.sh` rebuild all dependencies and images from scratch. This is the same script used by the gitlab CI
- It takes a few Gb of disk space and allow recreate custom images
- Useful to test a development branch, or to deploy a custom node.

Interlude : Docker GC

- Sometimes docker eats far too much disk space.
- Follow docker best practices when writing docker files
- Ensure docker data directory is mounted in a separate partition to avoid saturating your machine

Clean old image regularly

```
$ docker image prune --all --filter "until=1000h" -f
```

Purge dangling temporary images and layers :

```
$ docker system prune -f
```

Run the main docker image

```
$ docker run -it tezos/tezos:master --help

Available commands:

The following are wrappers around the Tezos binaries.
To call the Tezos binaries directly you must override the
entrypoint using --entrypoint . All binaries are in
/usr/local/bin and the Tezos data in /var/run/tezos

Daemons:
- tezos-node [args]
  Initialize a new identity and run the Tezos node.

- tezos-baker [keys]
- tezos-baker-test [keys]
- tezos-endorser [keys]
- tezos-endorser-test [keys]
```

Run the main docker image (cont)

```
Clients:
- tezos-client [args]
- tezos-signer [args]
- tezos-admin-client

Commands:
  - tezos-upgrade-storage
  - tezos-snapshot-import
    Import a snapshot. The snapshot must be available
    in the file /snapshot
    Using docker run, you can make it available
    using the command :
       docker run -v <yourfilename>:/snapshot
          tezos/tezos tezos-snapshot-import
    <yourfilename> must be an absulute path
```

Node initialization

```
$ docker run -it tezos:latest tezos-node
```

This command will :

- create a node identity (the wrapper script does this automatically)
- run a **mainnet** node in `full` mode and start the bootstrap process

Attention !

- It will take a long time to sync (~5 days)
- It will consume a lot of disk space (~40 GB)
- The bootstrapping procedure needs at least 8 GB of ram
- The node will the run using 4 GB

Importing snapshots and run a `full` node

Download a recent snapshot

```
$ wget https://www.lambsonacid.nl/babylonnet/latest \
  -O full

# for today only
$ http://192.168.100.53/latest -O full
```

Import the snapshot :

```
$ docker run -v $PWD/full:/snapshot tezos/tezos:master \
  tezos-snapshot-import`
```

Run the node :

```
$ docker run tezos/tezos:master tezos-node \
  --network babylonnet
```

We need a volume !

By now you have realized that nothing is saved in the container. We need to create a *named volume*

```
$ docker volume create tezos-node-volume

$ docker run \
  -v tezos-node-volume:/var/run/tezos \
  -v $PWD/full:/snapshot tezos/tezos:master \
  tezos-snapshot-import

$ docker run -v tezos-node-volume:/var/run/tezos \
  tezos/tezos:master tezos-node \
    --network babylonnet
```

Bingo ! The node is synchronizing and the data is going to be saved in a volume.

All this is very boring. Let's script it

docker-compose to the rescue.

- It's a docker orchestrator
- It allows to manage multiple images
- It needs a simple *yaml* configuration file.

```
$ sudo pip install docker-compose
```

or `apt-get install ...`

docker-compose.yml (babylonnet)

```yaml
version: "3"

volumes:
  data:
  client:

services:

  node:
    image: tezos/tezos:master
    hostname: node
    command: tezos-node --network babylonnet --rpc-addr 0.0.0.0
    ports:
      - 18732:8732
    volumes:
      - data:/var/run/tezos
    restart: unless-stopped
```

docker-compose.yml (import)

One shot command

```
import:
  image: tezos/tezos:master
  command: tezos-snapshot-import --network babylonnet
  volumes:
    - ./full:/snapshot
    - data:/var/run/tezos
```

Lets spin it up

We import the snapshot we downloaded

```
$ docker-compose up import
```

We start the node

```
$ docker-compose up -d node
```

We can check the logs

```
$ docker-compose logs node
```

Tezos Client (reminder)

- We install a snap binary
- snap is a container based sw distribution platform
  $ wget https://gitlab.com/abate/tezos-snapcraft/-/raw/master/snaps/tezos_5.1.0_multi.snap $ sudo snap install tezos_5.1.0_multi.snap –dangerous

And now finally we can talk with our node or with any Tezos node out there

```
$ export PATH=/snap/bin/:$PATH
$ tezos.client man
```

Tezos Client (cont)

We can keep track of the level that our local node has reached :

```
$ tezos.client -A localhost -P 18732 bootstrapped
```

Notice we connect to the node on `localhost:18732`

Or check the balance :

```
$ tezos.client get balance for bob
```

(We'll play with this in a moment)

Tezos Client configuration (reminder)

Instead of carrying these options with us, we can commit to the client configuration file

```
$ tezos.client config init -A localhost -P 18732
```

And check its content

```
$ tezos.client config show
```

```
{ "base_dir": "/home/abate/snap/tezos/x1/.tezos-client",
  "node_addr": "localhost", "node_port": 18732, "tls": false,
  "web_port": 8080, "confirmations": 0 }
```

Sandbox image: Running a sandbox node and a baker locally

- Useful to run experiments
- DAPPs development
- Self contained
- Runs a node and a baker
- `time-between-blocks` can be configured

```
$ docker run -it \
  -p 127.0.0.1:18731:18731 \
  nomadiclabs/tezos-sandbox:py --time-between-blocks 7
```

```
$ tezos.client -A localhost -P 18731 bootstrapped
```

Note the different rpc port !

Personal baker. One node, one baker, one signer

- We are going to bake on babylonnet
- Getting tokens from the faucet : https://faucet.tzalpha.net/
- Front end node + baker
- Baker account plus delegation
- Signer running in isolation from the baker

We add the baker

```
baker:
  image: tezos/tezos:master
  hostname: baker
  command: tezos-baker --max-priority 128
  environment:
    - PROTOCOL=005-PsBabyM1
  links:
    - node
  extra_hosts:
    signer: 172.20.0.1
  networks:
    - tezos
  volumes:
    - data:/var/run/tezos/node:ro
    - client:/var/run/tezos/client
  restart: unless-stopped
```

We added some network information !

- Docker has different network modes.
- By default all containers run on the same bridge.
- We create a special, private network for our node/baker
- `172.20.0.1` is the gateway for this network
- More details in a moment . . .

First try : start node and baker

Recap

```
$ docker-compose up import
$ docker-compose up -d node
$ tezos.client -A localhost -P 18733 config update
$ tezos.client bootstrapped
```

Now we can start the baker

```
$ docker-compose up baker
```

But, we have to get some baking rights first

- Faucet : https://faucet.tzalpha.net/
- Get some tokens
- Import tokens in your wallet

```
$ tezos.client activate account alice with faucet.json
$ tezos.client get balance for alice
```

`alice` is a *local* handler, you can all be `alice`

Possible messages you are going to see

The node is probably still syncing :

```
Waiting for the node to be bootstrapped before injection...
```

The node is waiting for the operation to be injected into the chain :

```
Waiting for the operation to be included...
```

We need to wait a bit . . .

A bit of key management (reminder)

Let's create a new key using the `tezos-signer` and set bob as delegate

```
$ tezos.signer gen keys bob --encrypted
$ tezos.signer launch local signer
```

Let's add bob to our wallet

```
$ tezos.signer list known addresses
bob: tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG

$ tezos.client add address bob \
  tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG

$ tezos.client import secret key bob \
  unix:///$HOME/.tezos-signer/socket?\
    pkh=tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG"
```

Delegation

Bob needs some funds to cover it's baking services

```
$ tezos.client transfer 100 from alice to bob \
  --burn-cap 0.257
```

Now we can register bob as delegate

```
$ tezos.client register key bob as delegate
```

Finally we can delegate our tokens to bob (that is our baker account)

```
$ tezos.client set delegate for alice to bob
```

We are all set. `bob` is the key for our baker, `alice` has delegated all her tokens to `bob`

Docker networking detour

In order to connect from the container running the baker to the host we need to setup the docker networking (I've already shown you the `tezos` network)

The same as in `docker-compose` can be accomplish using the `docker` command

```
$ docker network create -d bridge \
  --subnet 172.20.0.0/16 \
  --gateway 172.20.0.1 tezos
```

Then test if we can effective connect to the `signer`

```
$ docker container run --rm -it \
  --add-host="signer:172.20.0.1"
  --net=tezos --user=root tezos/tezos-bare:master sh
```

Hello signer . . .

```
$ ping signer
PING signer (172.20.0.1): 56 data bytes
64 bytes from 172.20.0.1: seq=0 ttl=64 time=0.165 ms
64 bytes from 172.20.0.1: seq=1 ttl=64 time=0.080 ms
^C
--- signer ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.080/0.122/0.165 ms
```

Using docker-compose (as before)

We add the network configuration

```
networks:
  tezos:
    driver: bridge
    ipam:
        driver: default
        config:
          - subnet: 172.20.0.0/16
            gateway: 172.20.0.1
```

Using docker-compose (cont)

```
baker:
  image: tezos/tezos:master
  hostname: baker
  command: tezos-baker --max-priority 128
  environment:
    - PROTOCOL=005-PsBabyM1
  links:
    - node
  volumes:
    - data:/var/run/tezos/node:ro
    - client:/var/run/tezos/client
  extra_hosts:
    signer: 172.20.0.1
  networks:
    - tezos
  restart: unless-stopped
```

We can finally run our signer and baker

Launch the signer on the host :

```
$ tezos.signer launch http signer -a 172.20.0.1
```

Let the baker know about the signer :

```
$ TZ1ADDR=tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG \
  docker-compose up configure-signer
```

Run the baker

```
$ docker-compose up baker
```

We are in business. Eventually we're going to bake depending on how many rolls we have. But When ?

Checking our baking rights

Using TzStats :

https://babylonnet.tzstats.com/tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG

From the command line :

```
$ tezos.client rpc get \
"/chains/main/blocks/head/helpers/baking_rights\
  ?delegate=tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG\
  &all"
```

Documentation

- https://tezos.gitlab.io/whitedoc/proof_of_stake.html?highlight=rights#delegation
- RPC: https://tezos.gitlab.io/api/rpc.html#get-block-id-helpers-baking-rights

Ansible all the things

- Ansible is a provisioning tool.
- It depends only by python3 and ssh to work.

First let's install ansible on your machine.

```
$ pip3 install ansible
```

We are going to provision a VM with ansible basically repeating the work we've done so far from a script.

Ansible configuration

Easy if you follow ansible conventions: Hell otherwise

```
$ git clone https://gitlab.com/abate/training2020

$ cat inventory/servers
test-server ansible_ssh_host=<your remote host> \
  ansible_user=ubuntu ansible_user_dir=/home/ubuntu
[node]
test-server

$ cat requirements.yml

    - src: git@gitlab.com:abate/ansible-tezos-docker-run.git
      scm: git
      name: tezos-docker-run
```

Main Ansible configuration file `site.yml`

```
---

- name: initialize nomadic babylonnet node
  hosts: nomadic
  tags: ['install']
  become: yes
  tasks:
    - name: Configure ufw rules
      ufw: rule={{ item.rule }} port={{ item.port }} proto={{ item.p
      with_items:
        - { rule: 'allow', port: '9732', proto: 'tcp' }
        - { rule: 'allow', port: '22', proto: 'tcp' }
      notify: restart ufw
    - name: Allow Remote port forward
      action:
        lineinfile:
          dest="/etc/ssh/sshd_config"
          line="GatewayPorts yes" state="present"
      notify: restart ssh
```

Main Ansible configuration file `site.yml`

```yaml
- name: run a babylonnet docker node
  hosts: nomadic
  roles:
    - role: "tezos-docker-run"
      network: "babylonnet"
      protocol: "005-PsBabyM1"
      snapshot: "https://www.lambsonacid.nl/babylo...FULL"
      history_mode: "full"
      bakers:
        - name: bob
          phk: tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG
  tags: ['tezos-babylonnet-init']
```

Run the Tezos node

```
- name: run babylonnet docker node
  hosts: nomadic
  roles:
    - role: "tezos-docker-run"
      network: "babylonnet"
      protocol: "005-PsBabyM1"
      history_mode: "full"
  tags:
    - tezos-babylonnet-node
```

Configure the signer

```
- name: configure signer
  hosts: nomadic
  roles:
    - role: "tezos-docker-run"
      network: "babylonnet"
      protocol: "005-PsBabyM1"
      history_mode: "full"
      bakers:
        - name: bob
          phk: tz1XBTuhu17722dA2ALQZFf99iyNfksdQdsG
  tags:
    - tezos-babylonnet-signer
```

Start the Baker, Endorser and Accuser

```yaml
- name: start baker
  hosts: nomadic
  roles:
    - role: "tezos-docker-run"
      network: "babylonnet"
      protocol: "005-PsBabyM1"
      history_mode: "full"
      start_baker: true
  tags:
    - tezos-babylonnet-baker
```

A lot of magic, no so magic

Careful Mr. Robinson : **Double baking !!!**

Tezos Docker Run (Ansible role)

- Install/Configure docker and all its dependencies
- Add user tezos
- Create a special docker *tezos network*
- If provided download and init the node with a snapshot
- Start Tezos node
- If provided, configure the baker to use a remote signer
- Launch baker, accuser and endorser

Local signer configuration

We are going to use ssh to provide a secure tunnel to our signer

```
$ cat ~/.ssh/config
Host baker
  HostName 35.181.154.189
  User ubuntu
  RemoteForward 172.18.0.1:6732 127.0.0.1:6732

$ autossh -f baker
```

and we can launch the signer as before :

```
$ tezos.signer launch http signer -a 127.0.0.1
```

But this time listening on **localhost**

Bonus: let's use systemd for the tunnel

```
[Unit]
Description=AutoSSH tunnel service for Tezos baker
After=network.target

[Service]
Environment="AUTOSSH_GATETIME=0"
ExecStart=/usr/bin/autossh -f baker
User=abate

[Install]
WantedBy=multi-user.target
```

Why this is so complicated ?

Security

- Your private keys should never leave your machine
- The signer can be configured to use a ledger
- Simple turn-key setup that you can easily adapt to your needs
- Automatic and can be improved together
- Docker images can be easily updated (no recompilation)

An easier way would be to store your keys directly on the baker or on a signer in the cloud, or using a third-party wallet for signing operations.

Can we do more complicated ?

- Multiple front-end node nodes for 99% uptime
- Split your rolls among multiple bakers/delegates

Can we do it **less** complicated ?

- Delegate your rolls to a baking service.

Question?

Figure: ?

Contact Us

Pietro Abate pietro.abate@nomadic-labs.com