






# Testnet config

 Date	
 Slides	
  Courses	<u>IDP</u>
 Lecture video link	

This script will generate the deployment inventory dynamically, based on the contents of:

1. /testnet/env/shared-config.yml
2. /testnet/env/<deployment>/hosts.ini

```
.
├─ ansible
│   ├── ansible.cfg
│   ├── config
│   ├── debug_vars_dump.yml
│   ├── icos_network_redeploy.yml
│   ├── icos_node_backup_base_checkpoint.yml
│   ├── icos_node_purge_checkpoints.yml
│   ├── icos_node_recover_base_checkpoint.yml
│   ├── icos_node_stress.yml
│   ├── ic_p8s_network_destroy.yml
│   ├── ic_p8s_network_update.yml
│   ├── ic_p8s_service_discovery_destroy.yml
│   ├── ic_p8s_service_discovery_install.yml
│   ├── inventory
│   └─ roles
├─ ansible.cfg
├─ BUILD.bazel
├─ config
│   └─ ssh_authorized_keys
├─ default.nix
├─ dfinity_owned.yml
├─ docs
│   ├── alerts
│   ├── AnalyzingDivergedStates.md
│   ├── index.adoc
│   └─ ReplayTool.md
├─ env
│   ├── benchmarklarge
│   ├── benchmarksmall01
│   ├── benchmarkxsmall01
│   ├── benchmarkxsmall02
│   ├── cdhotfix01
│   ├── cdhotfix02
│   └─ cdhotfix03
```



```
|— nns_state_deployment.sh
|— nns-tools
```

## inventory.py

This script will generate the deployment inventory dynamically, based on the contents of:

1. /testnet/env/shared-config.yml
2. /testnet/env/<deployment>/hosts.ini

## Host.ini

tells the parameters for the testnet.

```
[physical_hosts]
[physical_hosts:vars]
# Resources per node
ic_disk_gb=500
ic_cores=30
ic_memory_gb=256

# Note: ipv6 addresses of these nodes can be obtained by *executing* `./hosts --nodes`
[nns]
benchmarklarge.0.0 ic_host="ch1-spm16"
benchmarklarge.0.1 ic_host="fr1-spm16"
benchmarklarge.0.2 ic_host="sf1-spm16"
benchmarklarge.0.3 ic_host="zh1-spm05"
benchmarklarge.0.4 ic_host="ch1-spm16"
benchmarklarge.0.5 ic_host="fr1-spm16"
benchmarklarge.0.6 ic_host="sf1-spm16"
benchmarklarge.0.7 ic_host="zh1-spm05"
benchmarklarge.0.8 ic_host="ch1-spm17"
benchmarklarge.0.9 ic_host="fr1-spm17"
benchmarklarge.0.10 ic_host="sf1-spm17"
benchmarklarge.0.11 ic_host="zh1-spm06"
benchmarklarge.0.12 ic_host="ch1-spm18"
benchmarklarge.0.13 ic_host="fr1-spm18"
benchmarklarge.0.14 ic_host="sf1-spm18"
benchmarklarge.0.15 ic_host="zh1-spm07"
benchmarklarge.0.16 ic_host="ch1-spm19"
benchmarklarge.0.17 ic_host="fr1-spm19"
benchmarklarge.0.18 ic_host="sf1-spm19"
benchmarklarge.0.19 ic_host="zh1-spm08"
benchmarklarge.0.20 ic_host="ch1-spm20"
benchmarklarge.0.21 ic_host="fr1-spm20"
benchmarklarge.0.22 ic_host="sf1-spm20"
```

```

benchmarklarge.0.23 ic_host="zh1-spm09"
benchmarklarge.0.24 ic_host="ch1-spm21"
benchmarklarge.0.25 ic_host="fr1-spm21"
benchmarklarge.0.26 ic_host="sf1-spm21"
benchmarklarge.0.27 ic_host="zh1-spm10"
benchmarklarge.0.28 ic_host="ch1-spm22"
benchmarklarge.0.29 ic_host="fr1-spm22"
benchmarklarge.0.30 ic_host="sf1-spm22"
benchmarklarge.0.31 ic_host="zh1-spm11"
benchmarklarge.0.32 ic_host="ch1-spm16"
benchmarklarge.0.33 ic_host="fr1-spm16"

[subnet_1]
benchmarklarge.1.34 ic_host="sf1-spm16"
benchmarklarge.1.35 ic_host="zh1-spm05"
benchmarklarge.1.36 ic_host="ch1-spm16"
benchmarklarge.1.37 ic_host="fr1-spm16"
benchmarklarge.1.38 ic_host="sf1-spm16"
benchmarklarge.1.39 ic_host="zh1-spm05"
benchmarklarge.1.40 ic_host="ch1-spm17"
benchmarklarge.1.41 ic_host="fr1-spm17"
benchmarklarge.1.42 ic_host="sf1-spm17"
benchmarklarge.1.43 ic_host="zh1-spm06"
benchmarklarge.1.44 ic_host="ch1-spm18"
benchmarklarge.1.45 ic_host="fr1-spm18"
benchmarklarge.1.46 ic_host="sf1-spm18"

[boundary]
benchmarklarge.boundary.47 ic_host="ch1-spm16"
benchmarklarge.boundary.48 ic_host="fr1-spm16"
benchmarklarge.boundary.49 ic_host="sf1-spm16"
benchmarklarge.boundary.50 ic_host="zh1-spm05"
benchmarklarge.boundary.51 ic_host="ch1-spm17"
benchmarklarge.boundary.52 ic_host="zh1-spm06"
benchmarklarge.boundary.53 ic_host="ch1-spm18"
benchmarklarge.boundary.54 ic_host="fr1-spm18"
[boundary:vars]
system_domains=benchmarklarge.testnet.dfinity.network
application_domains=benchmarklarge.testnet.dfinity.network
cert_name=sized-testnet.dfinity.network

[aux]
benchmarklarge.aux.55 ic_host="ch1-spm16"

[nodes:children]
nns
subnet_1
boundary
aux

[prometheus]
# General prometheus config is in shared-config.yml
[prometheus:vars]
# Note: The port must be different for each deployment. See /testnet/README.md
ic_p8s_service_discovery_metrics_addr=[2a05:d01c:d9:2b84:e1df:81b7:9c18:a85b]:8013

```

# Main entryptpoint

It is the `testnet/tools/icos_deploy.sh` . In order to run this script, you have to first get the git head hash of the disk image to be used using `./gitlab-ci/src/artifacts/newest_sha_with_disk_image.sh`

```
root@litecoingold:~/shouvik/ic# ./gitlab-ci/src/artifacts/newest_sha_with_disk_image.sh origin/master
From https://github.com/dfinity/ic
* branch          master      -> FETCH_HEAD
62bf2e969433a5a221d684b339187d8f752576d0
```

## Steps:

### 1. Displays local ipv4 and ipv6 address info

```
-----
**** Local IPv4 address information:

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    inet 127.0.0.1/8 scope host lo
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    altname enp7s0
    inet 172.16.163.1/12 brd 172.31.255.255 scope global dynamic eno1

-----
**** Local IPv6 address information:

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN qlen 1000
    inet6 ::1/128 scope host
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP qlen 1000

-----
```

### 2. Destroy the previous deployments

```
**** Start destroying old deployment (log /tmp/icos-deploy.sh.o5y2Aq/destroy.log)
set -x;
ansible ()
{
    ansible-playbook $(ANSIBLE_ARGS[@]) "$@"
}
declare -a ANSIBLE_ARGS=([0]="-e" [1]="bn_media_path=/root/shouvik/ic/artifacts/boundary-guestos/snall01/62bf2e969433a5a221d684b339187d8f752576d0" [2]="-l" [3]="/root/shouvik/ic/testnet/env/snall01/hosts" [4]="-e" [5]="bn_image_type=" [6]="-e" [7]="ic_git_revision=62bf2e969433a5a221d684b339187d8f752576d0" [8]="-e" [9]="ic_media_path=/root/shouvik/ic/artifacts/guestos/snall01/62bf2e969433a5a221d684b339187d8f752576d0" [10]="-e" [11]="ic_boundary_node_image=boundary")
cd "/root/shouvik/ic/testnet/ansible"
ansible icos network redeploy.yml -e ic.state=destroy
```

### 3. Builds USB Sticks for IC nodes

(create an image for later use, maybe for snapshotting)

In this step it downloads the canisters, release files etc. Initially it tries to get it from the s3-bucket, but then it gets it from the “fallback options) which is doing rclone from

somewhere (not sure how it works)

```
curl: (7) Couldn't connect to server
Falling back to the direct mode ...
2023/04/23 14:01:29 INFO : SHA256SUMS: Copied (new)
2023/04/23 14:01:29 INFO : cow_safety.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : counter.wat.gz: Copied (new)
2023/04/23 14:01:29 INFO : candid-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : canister-creator-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : cycles-minting-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : genesis-token-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : governance-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : governance-mem-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : governance-canister_test.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : http_counter.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-btc-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-ckbtc-kyt.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-ckbtc-minter.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-icrc1-archive.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-ckbtc-minter_debug.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-icrc1-index.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : identity-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-icrc1-ledger.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : inter_canister_error_handling.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ic-nervous-system-common-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : json.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : kv_store.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ledger-archive-node-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ledger-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : lifeline_canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : ledger-canister_notify-method.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : memory-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : mem-utils-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : nan_canonicalized.wasm.gz: Copied (new)
2023/04/23 14:01:29 INFO : nns-ui-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : panics.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : pmap_canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : response-payload-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : proxy_canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : registry-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : root-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-governance-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-governance-mem-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-governance-canister_test.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-root-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-test-dapp-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-swap-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : statesync-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : sns-wasm-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : stable.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : time.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : upgrade-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : wasm.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : test-notified.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO : xnet-test-canister.wasm.gz: Copied (new)
2023/04/23 14:01:30 INFO :
Transferred:      10.645M / 10.645 MBytes, 100%, 6.885 MBytes/s, ETA 0s
Transferred:       51 / 51, 100%
Elapsed time:      1.7s
```

## 4. Build USB sticks for boundary nodes

```
**** Build USB sticks for boundary nodes - (Sun 23 Apr 2023 02:01:40 PM CEST)
set -x; err () { echo "[S(date + '%Y-%m-%d %H:%M:%S%Z')]]: $*"; is=2; HOSTS=(zhi.spm02.zhi.dfinity.network); CERT_NAME=sized:testnet.dfinity.network; mkdir -p /root/shouvik/ic/artifacts/boundary-guestos/small01/62b72e969433a5a221d684b339187d8f752576d0/certs; if [[ -x $(CERT_NAME) ]]; then err ".boundary.vars.cert_name" was not defined" else (for HOST in $(HOSTS[@]); do echo >2 "S(date - -rfc-3339=seconds)"; copying $CERT_NAME from server $HOST" scp -B -o "ConnectTimeout 30" -o "UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no" -r "${HOST}:/etc/letsencrypt/live/${CERT_NAME}/" /root/shouvik/ic/artifacts/boundary-guestos/small01/62b72e969433a5a221d684b339187d8f752576d0/certs/" && exit done) || { err "failed to find certificate $CERT_NAME on any designated server" exit 1 } fi; echo >2 "2023-04-23 14:01:41+02:00: Running build-deployment.sh" /root/shouvik/ic/artifacts/boundary-guestos/scripts/build-deployment.sh --input="/root/shouvik/ic/artifacts/boundary-guestos/small01/62b72e969433a5a221d684b339187d8f752576d0/small01.json" --output="/root/shouvik/ic/artifacts/boundary-guestos/small01/62b72e969433a5a221d684b339187d8f752576d0/pos-public-key.pem" --cert="/root/shouvik/ic/artifacts/boundary-guestos/small01/62b72e969433a5a221d684b339187d8f752576d0/pos-public-key.pem"
```

It is able to download the required files from somewhere. However, there's some error," then err ".boundary.vars.cert\_name' was not defined' else (for HOST in \$(HOSTS[@]) not sure where we have to define it.

## 5.icos\_redeploy.yml is run with create state

once the prev steps are done, we run the icos\_redeploy.yml. it is used to deploy the icos to the nodes.

However, before that, ansible uses inventory to manage the nodes. The inventory defines the managed nodes that are automated, with groups so that we can run automation tasks on multiple hosts at the same time. Once the inventory is defined, patterns is used to select the hosts or groups you want Ansible to run against.

The file can be found here: [testnet/ansible/inventory/inventory.py](#)

The inventory.py takes care of assigning the ipv6 address to the nodes

```
root@litecoingold:~/shouvik/ic/testnet/env/small01# ./hosts --nodes
small01.0.0: 2a00:fb01:400:42:5000:c9ff:fe17:cc3a
small01.1.1: 2a00:fb01:400:42:5000:3fff:fe04:d39b
small01.aux.3: 2a00:fb01:400:42:5000:dcff:fe1e:89dd
small01.boundary.2: 2a00:fb01:400:42:5000:9eff:fed8:2d4
```

This is done through the taking the first part of the ipv6 address from the shared config.

```

et > env > ! shared-config.yml
#
# Example use:
# ansible-playbook -i env/${network}/hosts,env/shared

prometheus:
  vars:
    stage: all
  hosts:
    prometheus.testnet.dfinity.network:

data_centers:
  ch1:
    vars:
      ipv6_prefix: "2607:f6f0:3004:1"
      ipv6_subnet: "/64"
  dml:
    vars:
      ipv6_prefix: "2604:6800:258:1"
      ipv6_subnet: "/64"
  fr1:
    vars:
      ipv6_prefix: "2001:4d78:40d"
      ipv6_subnet: "/64"
  ln1:
    vars:
      ipv6_prefix: "2a0b:21c0:4003:2"
      ipv6_subnet: "/64"
  sel:
    vars:
      ipv6_prefix: "2600:c00:2:100"
      ipv6_subnet: "/64"
  sf1-old:
    vars:
      ipv6_prefix: "2607:fb58:9005:42"
      ipv6_subnet: "/64"
  sf1:
    vars:
      ipv6_prefix: "2602:fb2b:100:10"
      ipv6_subnet: "/64"
  zh1:
    vars:
      ipv6_prefix: "2a00:fb01:400:42"
      ipv6_subnet: "/64"

```

Valeriy Zama



However, this is just the prefix.

In `testnet/env/small01/hosts.ini` , the way nodes are defined is `zh1-spm01` . This automatically generates the the complete ipv6 address

This is not yet clear why and how the `spm01` translates into the address

```
testnet > env > small01 > hosts.ini
Or Ricon, 6 months ago | 5 authors (Valeriy Zamaraiev and others)
1 # Note: ipv6 addresses of these nodes can be obtained by *executing*
2 [nns]
3 small01.0.0 ic_host="zh1-spm02"
4
5 [subnet_1]
6 small01.1.1 ic_host="zh1-spm02"
7
8 [boundary]
9 small01.boundary.2 ic_host="zh1-spm02"
10 [boundary:vars]
11 system_domains=small01.testnet.dfinity.network
12 application_domains=small01.testnet.dfinity.network
13 cert_name=sized-testnet.dfinity.network
14
15 [aux]
16 small01.aux.3 ic_host="zh1-spm02"
17
18 [nodes:children]
19 nns
20 subnet_1
21 boundary
22 aux
23
24 [prometheus]
25 # General prometheus config is in shared-config.yml
26 [prometheus:vars]
27 # Note: The port must be different for each deployment. See /testnet/README.md
28 ic_p8s_service_discovery_metrics_addr=[2a05:d01c:d9:2b84:e1df:81b7:9c18:a85b]:8051
```

Somehow, once all this is done, in the `artifacts` folder, the complete mapping of the nodes to the physical mapping is created.a

```
acts > boundary-guestos > small01 > dac024ceaaded49d3af84519e53fb8d5f08ce2b7 > {} list.json > {} _meta > {} hostvars > {} small01.0.0 > mac_addre
{
  "_meta": {
    "hostvars": {
      "small01.0.0": {
        "ansible_host": "2a00:fb01:400:42:5000:c9ff:fe17:cc3a",
        "guest_hostname": "zh1-spm02",
        "guest_number": 1,
        "ic_host": "zh1-spm02",
        "inventory_dir": "/home/shouvik/Work/Personal/ic/testnet/env/small01",
        "inventory_file": "/home/shouvik/Work/Personal/ic/testnet/env/small01/hosts.ini",
        "ipv6_address": "2a00:fb01:400:42:5000:c9ff:fe17:cc3a",
        "mac_address": "52:00:c9:17:cc:3a",
        "node_index": 0,
        "subnet_index": 0
      },
      "small01.1.1": {
        "ansible_host": "2a00:fb01:400:42:5000:3fff:fe04:d39b",
        "guest_hostname": "zh1-spm02",
        "guest_number": 2,
        "ic_host": "zh1-spm02",
        "inventory_dir": "/home/shouvik/Work/Personal/ic/testnet/env/small01",
        "inventory_file": "/home/shouvik/Work/Personal/ic/testnet/env/small01/hosts.ini",
        "ipv6_address": "2a00:fb01:400:42:5000:3fff:fe04:d39b",
        "mac_address": "52:00:3f:04:d3:9b",
        "node_index": 1,
        "subnet_index": 1
      },
      "small01.aux.3": {
        "ansible_host": "2a00:fb01:400:42:5000:dcff:fe1e:89dd",
        "guest_hostname": "zh1-spm02",
        "guest_number": 3,
        "ic_host": "zh1-spm02",
        "inventory_dir": "/home/shouvik/Work/Personal/ic/testnet/env/small01",
        "inventory_file": "/home/shouvik/Work/Personal/ic/testnet/env/small01/hosts.ini",
        "ipv6_address": "2a00:fb01:400:42:5000:dcff:fe1e:89dd",
        "mac_address": "52:00:dc:1e:89:dd",
        "node_index": 3,
        "subnet_index": "aux"
      },
      "small01.boundary.2": {
        "ansible_host": "2a00:fb01:400:42:5000:9eff:fed8:2d4",
        "guest_hostname": "zh1-spm02",
        "guest_number": 4,
        "ic_host": "zh1-spm02",
        "inventory_dir": "/home/shouvik/Work/Personal/ic/testnet/env/small01"
      }
    }
  }
}
```

One possible way that it is happening is through `/ic-os/boundary-api-guestos/rootfs/opt/ic/bin/setup-nftables.sh` what is used to set up the network configuration, but not sure how this is being used in the testnet Could not find a possible connection from testnet folder to this file.

```
> boundary-api-guestos > rootfs > opt > ic > bin > $ setup-nftables.sh
readonly NETWORK_CONFIG="${BOOT_DIR}/network.conf"

readonly RUN_DIR='/run/ic-node/etc/nftables'
readonly SYSTEM_REPLICAS_FILE="${RUN_DIR}/system_replicas.ruleset"
readonly RULESET_FILE="${RUN_DIR}/defs.ruleset"

ipv6_replica_ips="::/128"
ipv4_http_ips="0.0.0.0/32"
ipv6_http_ips="::/128"
ipv6_debug_ips="::/128"
ipv6_monitoring_ips="::/128"

function csv() {
    local -r arr=("${@}")
    IFS=,
    echo "${arr[*]}"
}

# Read the network config variables from file. The file must be of the form
# "key=value" for each line with a specific set of keys permissible (see code
# below).
function read_variables() {
    if [[ ! -d "${BOOT_DIR}" ]]; then
        err "missing node configuration directory: ${BOOT_DIR}"
        exit 1
    fi
    if [ ! -f "${BN_CONFIG}" ]; then
        err "missing bn_vars configuration: ${BN_CONFIG}"
        exit 1
    fi
    if [ ! -f "${NETWORK_CONFIG}" ]; then
        err "missing network configuration: ${NETWORK_CONFIG}"
        exit 1
    fi
}

# Read limited set of keys. Be extra-careful quoting values as it could
# otherwise lead to executing arbitrary shell code!
while IFS="=" read -r key value; do
    case "$key" in
        "ipv4_http_ips") ipv4_http_ips+="${value}" ;;
        "ipv6_http_ips") ipv6_http_ips+="${value}" ;;
        "ipv6_debug_ips") ipv6_debug_ips+="${value}" ;;
        "ipv6_monitoring_ips") ipv6_monitoring_ips+="${value}" ;;
    esac
done < "${BN_CONFIG}"
```

This is kinda what is used to set up the network configuration, but not sure how this is being used in the `testnet`. Could not find a possible connection from `testnet` folder to this file.

## 6. Run the playbook `icos_network_redeploy.yml`

However, Since the IP configurations are so convoluted, I am not able to find a way to update the IPs as they are being generated through `inventory.py`.

Because of this, the playbook crashes when the trying to ssh into the nodes.

## Some other issues:

```
Valeriy Zamaraiev, 17 months ago | 1 author (Valeriy Zamaraiev)
### Dependencies

In order to run the Ansible deployment from your own machine or any remote
server, the following dependencies have to be met:
- Operating System:
  \ \ \
  Ubuntu 20.04
  \ \ \

:warning: Deployments from MacOS are not supported at the moment.

- Packages:
  \ \ \
apt -y install ansible coreutils jq mtools rclone tar util-linux unzip --no-install-recommends
  \ \ \

If you are not working on a Ubuntu 20.04 based system, you can use the following
office builders.
Please make sure that you initialize the ssh agent before connecting, and to
forward the local ssh credentials.

Check the ssh-agent keys
\ \ \ bash
ssh-add -L
  \ \ \

  \ \ \
# SSH to remote machine using your DFINITY SSH user
ssh -A zh1-spm22.zh1.dfinity.network
  \ \ \
Valeriy Zamaraiev, 17 months ago • Opensourcing more of the IC infrastructure

or
  \ \ \
ssh -A zh1-spm34.zh1.dfinity.network
  \ \ \
```

Also, in order to deploy the ansible deployment from some other machines, it is needed to ssh into the dfinity machine with their credentials. Not sure what's the dependence on this.

However, after a bit of digging, found that one of the important components for the IC system, i.e the NNS needs to be run on that specific machine in the dfinity network. they haven't implemented it yet to be deployed on a local machine yet.

```
Daniel Wong, last month | 2 authors (Maximilian Summe and others)
### nns_dev_testnet.sh

This script creates a testnet with mainnet state using a stable shared identity and modifies it in a few ways for development
purposes.
1. Adds an application subnet.
2. Sets CMC default subnet list to that application subnet.
3. Creates a cycles wallet for our shared principal on the application subnet.
4. Configures SNS-W to create SNS's on application subnet, and to respond to our principal's wallet.
5. Uploads the latest SNS Wasms into SNS-W canister

It then stores all of the variables in a directory (which is output) so they can be easily referenced for
interaction with the subnet

Needs to be run on zh1-spm22.zh1.dfinity.network. (Ideally, we'd
be able to run this locally; implementing that is feasible, but
we haven't done it yet.) Daniel Wong, last month • docs: Elaborated some instructions related to rel...
```

## ToDo:

1. Do more research about the `nns_dev_testnet.sh`
2. See if we can map the ips of our local testbed with the ipv6 addresses of the nodes in their dfinity testbed.