EWF Authority Nodes - Requirements and Procedures - EWC Edition

Markus Keil, Friedrich Raschwitz

January 2019

Abstract

To provide a stable and secure blockchain for all affiliates and customers, it is important that all parties that run authority nodes follow the procedures and guidelines written in this document. Authority nodes are not used for interaction with the chain. If an affiliate wants to access the chain via RPC through a central node it has to be set up separately as non-authority.

Contents

1 Nomenclature

EWF Network Operations (NetOps) Group of people on EWF side with the responsibility to keep the blockchain secure and operational at all times.

EWF Governance Operations (GovOps) Group of people on EWF side that decide on governance questions.

NodeControl A system component that carries out operational tasks on a validator node on behalf of NetOps

Bootnode Parity node that runs in fullnode configuration and is part of the bootnode section of the chainspec file. New clients that join the chain will contact a bootnode to discover other nodes on the network that are known to the bootnode.

Validator Node Parity node that seals transactions into new blocks based on the AURA consensus algorithm.

Genesis Node A special validator node. The genesis nodes are the first ones on the network and operated by EWF. These nodes will bootstrap the blockchain with the genesis block.

Fullnode A simple node on the network that don't seal new blocks. These nodes are not in the scope of these guidelines

2 System Design

The system of the validator nodes and their supporting components are designed to provide security and stability. The basic layout is shown in Figure 1.

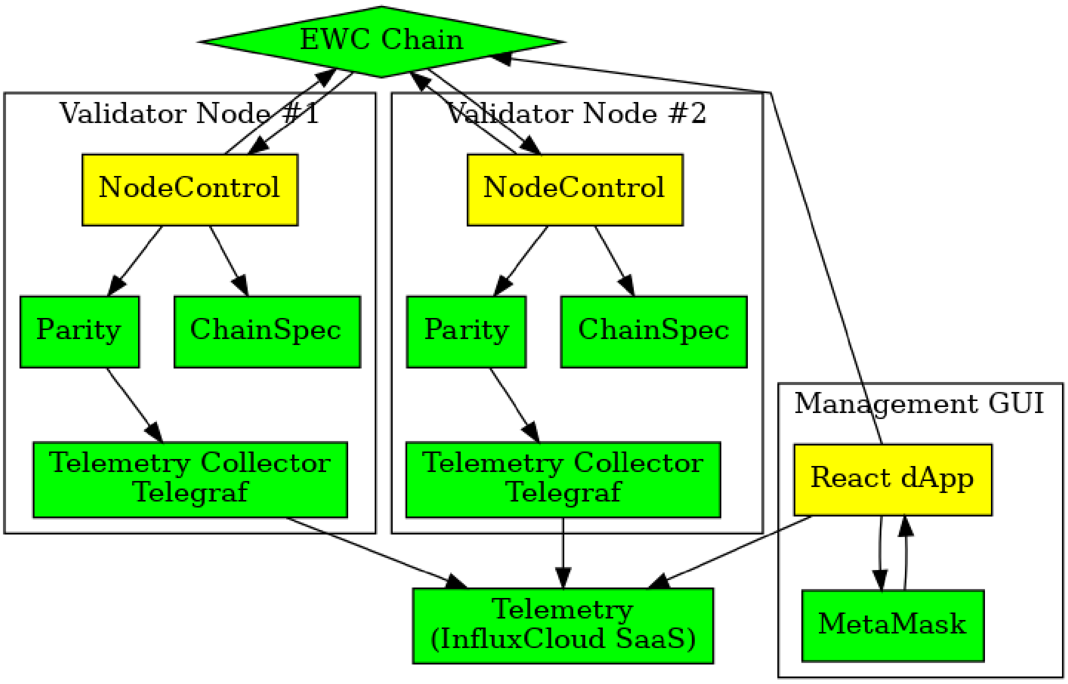

Figure 1: Validator System Design

2.1 System Components

These system components are tailor made or where developed for the validator network.

2.1.1 NodeControl

A small deamon that will carry out updates on behalf of NetOps. See 3.

2.2 Other Components

These components are standardize open-source components.

2.2.1 Telemetry Collector

The system uses Telegraf to collect the telemetry from on the nodes. This collected data is then send via the InfluxDB wire protocol to the also locally running telemetry signer.

2.2.2 Telemetry SaaS Backend

To gather and visualize the telemetry a SaaS provider is used.

2.2.3 Blockchain Client

The blockchain client is the main component of the system as it provides the connection to the blockchain and also carries out signing and validation duties. The probably used software will be Parity-Ethereum from Parity Tech running the AURA Proof-of-Authority engine.

3 NodeControl

NodeControl is a small management application that runs on each validator node. It is in charge of carrying out simple update tasks. It will listen to the local blockchain client via the HTTP RPC interface. The ethereum address of the validator account is used as a node identifier. NodeControl will only act on events directly directed to its assigned address. This allows granular deployment of updates.

The trigger process is shown in Fig. 2.

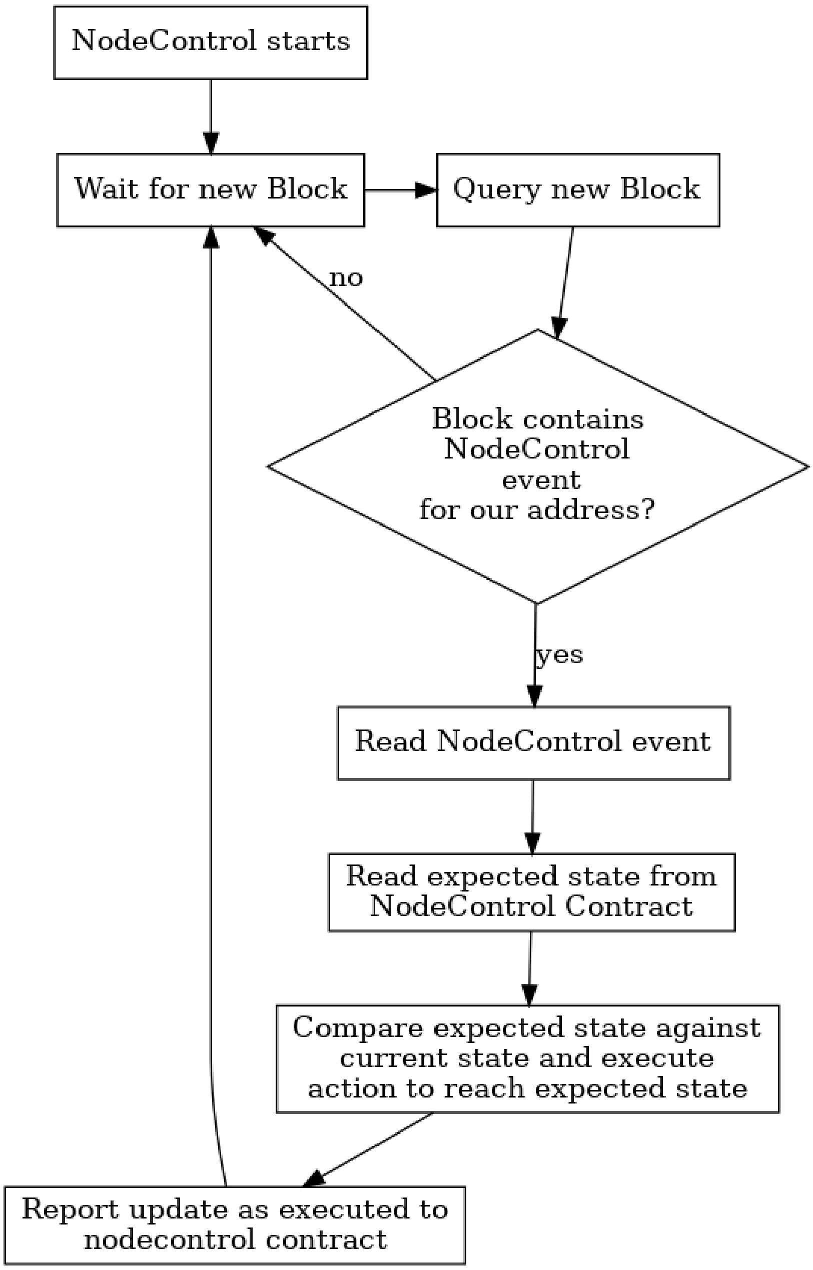
NodeControl starts
Wait for new Block
Query new Block
no
Block contains NodeControl event for our address?
yes
Read NodeControl event
Read expected state from NodeControl Contract
Compare expected state against current state and execute action to reach expected state
Report update as executed to nodecontrol contract

Figure 2: NodeControl Trigger Process

4 Genesis Network

The Genesis Network is a special set of validators and full nodes that will be used during launch. Hosts of the genesis network will be geographically distributed across different AWS regions. These regions will be linked via VPC peering, so traffic inside the genesis network is not public facing.

The genesis network will be run by the EnergyWeb Foundation.

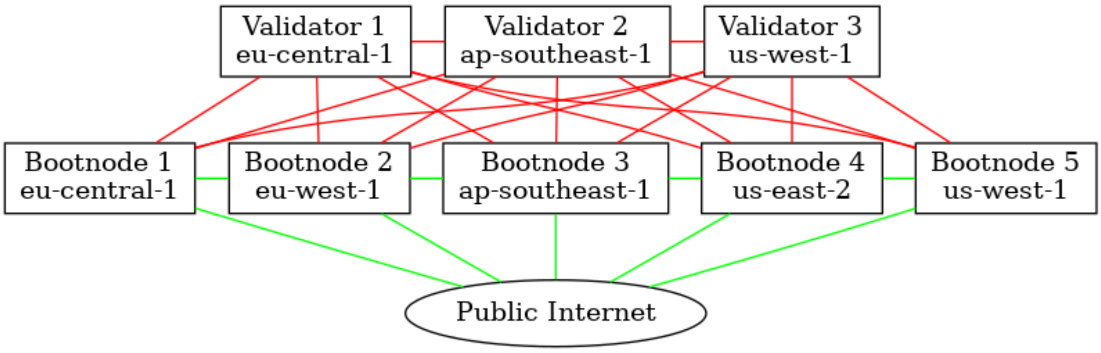

Figure 3: Genesis Network Layout

4.1 Validators

The genesis validators will be starting the chain and also their validator account addresses are hard coded as constructor arguments to the validator contract in the chain specification. The hosts running those validator nodes adhere to the same security standards as normal validators with the addition that they don't have any connection to the public internet blockchain wise.
The following outgoing traffic is permitted though an AWS NAT gateway:

- DNS (53/UDP+TCP)
- HTTP (80/TCP) - mainly for sys updates
- HTTPS (443/TCP) - mainly for sending telemetry

Blockchain traffic is only allowed inside the AWS VPC to the Bootnodes. The validators will have a special configuration file for parity to facilitate that along with the following security group settings

- Allow P2P traffic (30303/tcp+udp) inbound only from the bootnodes via RFC1918 inside the VPC
- Allow P2P traffic (30303/tcp+udp) outbound only to the bootnodes via RFC1918 inside the VPC
- Allow SSH traffic only inbound via a jump host inside the VPC over RFC1918

A preferred EC2 Instance type would be c5.xlarge. These nodes also don't have dedicated public IP's

4.2 Bootnodes

The genesis bootnodes are regular fullnodes (state pruned, no tracing, no fatdb, no warp). They accept connections from the public internet and the validator VPC's on their blockchain P2P port.
Enode addresses of those bootnodes will be part of the general public chainspec.

5 Threat model

This section describes potential attack vectors to the system and how they are either mitigated by the system design, automatic intervention or human intervention.

5.1 Telemetry

As telemetry needs to flow from the validators to a virtual single ingress point on the SaaS provider. It is not protected by decentralization. This telemetry helps to detect abnormalities in the operation of the validators caused by either system malfunction or deliberate attacks.

Sending tampered data If an attacker manages to disturb node operation, he might try to disguise this by sending normal looking telemetry to the telemetry ingress on behalf of the attacked node. NetOps would detect the faulty node with increased time delay as telemetry data would contradict actual node behavior.
Acknoledged: The current SaaS system won't protect against this attack.

Denial-of-Service against the telemetry ingress An attacker might try to bring down the telemetry ingress completely to "blind" the EWF NetOps team about the status of the validator network.
Acknoledged: The SaaS provider should be able to handle an attack against his infrastructure. But this is not proofed.

Phishing for telemetry An attacker might try to receive telemetry from a node by re-routing telemetry traffic to an attacker system that mimics the ingress (DNS spoofing, MITM). The attacker can gain knowledge about for example the systems load patterns or how a system might respond to an attack.
Acknoledged: The SaaS provider should be able to handle an attack against his infrastructure.

DNS-Spoof/MitM against GUI An attacker might manage to DNS Spoof/MitM the connection between the browser of a member from the NetOps team and the telemetry backend. This way the attacker could present a faked system state of the validator network to NetOps.
Acknoledged: The SaaS provider is not able to provide signed data. Security relies on an untampered HTTPS connection.

Unauthorized access to telemetry data Telemetry data should only be available to the EWF NetOps Team. An attacker might try to directly call the telemetry backend to receive the data.
Acknoledged: The SaaS provider has to authenticate the users correctly.

5.2 Node Control

The Node control component is used to carry out updates to the validator nodes. These updates need to be legitimized by the EWF NetOps and GovOps teams. The component has the potential to disable a node due to a faulty or malicious update .

Send fake update event An attacker might try to fool the NodeControl into carrying out an update script that is not approved.
Prevention: NodeControl will talk to its local blockchain client most of the time via IPC. If it has to talk to outside nodes because the local client is down, it'll use incubed to verify responses.

Compromised Payload An attacker might MitM traffic to any server NodeControl would download a payload from (mainly used for the chain spec file update process).
Containment: Each update will have the SHA256 hash of its agreed payload on chain. The payload downloaded by NodeControl is verified against this hash. If the hashes don't match NodeControl will not carry out the command and report a faulty update to the NodeControl Contract.

5.3 Other Attack Vectors

This section consolidates attack vectors that are either not specific to a single component or targeted against miscellaneous components.

Attack time servers The Aura PoA consensus algorithm highly depends on accurate system clocks. An attacker could try to manipulate the system clock by intercepting requests made by the operating system via NTP (Network Time Protocol). If the attacker shifts the system time by +block time or -block time the validator will start signing during the wrong slot. Other validators then won't accept that block because it is out-of-turn and vice-versa the manipulated node would mark all incoming blocks as invalid as from its point of view all other validators appear to be syncing out-of-turn.
Mitigation: Some validator nodes should use high-precision hardware clocks such as GPS-based ones or DCF-77 clocks instead of NTP for their time source. This way not the whole network is affected. Normal operation could be restored by removing all non-precision-clock nodes from the network. This could mean reduced chain performance.

6 Validators run by Affiliates

This section will go through the requirements needed to run a standard validator by an EWF affiliate.

6.1 Hardware/VM selection for Validators

Affiliate can choose to run the node either On-Premise on their own hardware or at one of the supported cloud-providers.

6.1.1 On-Premise

A on-premise node should have these specs or higher. These resources should be reserved for the validator node and not shared with other workloads.

1. Modern Multi-core x64 CPU (at least 4 threads, preferably Xeon-class)
2. 8GB RAM (preferably ECC)
3. Local SSD storage, 250GB free capacity for blockchain, redundant in RAID-1
4. 1 GBit NIC

6.1.2 Amazon AWS

The following EC2 instance sizes are appropriate to run validators:

- m5.xlarge
- m5.2xlarge
- m5a.xlarge
- m5a.2xlarge
- c5.xlarge
- c5.2xlarge

The default EBS storage assigned (normally 8GB) is not large enough to run the node. Make sure to run the node with following EBS storage settings:

- General Purpose SSD (gp2)
- at least 300GB size

6.1.3 Microsoft Azure

The following Azure VirtualMachine sizes are suitable to run a validator:

- D4s_v3
- DS3_v2
- B4ms

Use **Premium SSD** as attached storage with a size of at least **300GB**.

6.2 Reliability and Diversity

EWF NetOps will guarantee a certain reliability and diversity by enforcing heterogeneity of geographical location, Operative System and cloud service. The rationale is that a bug of a specific OS, a failure of a cloud service provider or the backbone internet service provider should not harm the consensus of the Validators.
For this reason, NetOps establishes a set of rules:

- There are at least two Validators running different OS in the system.
- The second most common OS is run by at least 20% of Validators.
- There are at least 15% Validators operating from America, 15% from Europe and 15% from Asia.
- At least two cloud services are used. The second most popular is used by at least 20% of the Validators.

6.3 Supported Operating Systems

The following Linux-based Operating Systems are supported for running a authority node:

- Ubuntu Server 18.04 LTS
- Debian 9.8
- CentOS 7

Affiliates can apply for a certain operating system, but in order to keep a heterogeneous infrastructure EWF NetOps can instruct the Affiliate to use a specific operating system from the above list. The operating system must be installed according to the settings described in this document to qualify the host for becoming part of the authority network.

6.4 Security Requirements

Running an authority nodes requires raised awareness of host and node security as authorities are a main attack surface to disturb operation of the block chain. The following security rules apply:

- No services are permitted to run on the same host that are not part of the authority node package
- All incoming connections on all ports except SSH (22/tcp) and the P2P (30303/tcp+udp) port have to be firewalled on the host with DROP rules. To guarantee proper network etiquette, incoming ICMP has to be accepted.
- SSH access is only allowed for non-root users
- SSH access is only allowed through RSA keys
- Parity RPC endpoints (HTTP, WebSocket) have to be disabled for external use (only docker stack internal)
- System updates have to applied regularly and in a timely manner
- Regular run of rootkit detectors

Most of these rules will be provided in the following set up guides.

6.5 Connectivity Requirements

The following requirements should be met to ensure proper operation:

- Wired connection with 100 MBit/s symmetric link to the internet
- Low latency connection to next internet hop (<5ms)
- No data volume limitations

Even this document requires a 100MBit/s connection, that connection will not be saturated by the node. You can expect 10-30MBit/s [1] when the chain is under load. Traffic will mainly flow on port 30303 (udp/tcp). If chosen to run on a cloud provider, EWF NetOps needs to be consulted to select an appropriate hosting region to ensure node distribution across multiple regions. If chosen to run on-premise, EWF NetOps has to be informed about the hosting location to ensure proper global distribution decisions can be made. The hosting location should be chosen to be as close as possible to one of the major internet exchanges:

- Germany/Continental Europe/USA - DE-CIX / AMS-IX
- United Kingdom - LINX

6.6 AWS Security

When using EC2 instances one can provide at least one additional layer of security using a virtual firewall. The access of administrative tasks outside of the virtual OS also needs to be taken care of.

6.6.1 Access Management Recommendations

1. Do not use the root account for managing EC2 instances, instead assign a new user administrative rights for doing that.
2. Use groups to manage permissions with multiple users and always keep the set of permissions to a minimum
3. Enable multi-factor authentication for administrative accounts
4. Regularly check logs regarding account activity, e.g. using Cloudfront AWS Docs: Access Logs

6.6.2 AWS Firewall

AWS provides an alternative firewall solution for their virtual OS. Instead of configuring the firewall in the OS itself (or additionally), one can configure it one layer above. This firewall is defined by a set of "security groups". Each security group contains rules what packets should be permitted to or from the EC2 instance. That means that instead of having a set of 'Allow' or 'Deny' rules security groups can only contain 'Allow' rules while everything else is denied by default.

That also means that by default SSH access needs to be allowed in one security group. More information on that can be found here:
AWS Docs: authorizing-access-to-an-instance

6.6.3 EC2 Key Pairs

EC2 key pairs are basically just RSA key pairs generated and distributed using AWS-specific commands. In general one can follow the guide in the SSH section. However, these commands may be more convenient to some users, more information here:
AWS Docs: ec2-key-pairs

[1] TODO: need to verify

6.7 Setup Procedure

To setup a new authority node read this whole document and then use this procedure to carry out the set up. The full process is shown in Figure **??**

1. Choose hosting provider (on-premise or qualified cloud provider) and favored operating system
2. Consult with EWF NetOps on location and operating system
3. EWF NetOps will provide the location and OS to use, along with the official installation script for the chosen operating system
4. Install operating system according to this document
5. Deploy blockchain node using the script given by NetOps
6. Contact EWF NetOps to confirm installation and incoming telemetry
7. Send the validator account information to the EWF Governance team

7 Operating system installation

The following section provide a comprehensive guide for installation of one the supported operating systems. All further deployment procedures are based on the installation results.

7.1 Ubuntu Server 18.04 LTS

7.1.1 On-Premise

Procedure based on version 18.04.2.

- Download the ISO from https://www.ubuntu.com/download/server.
- Boot the ISO
- Select **English** as language
- Choose a convenient keyboard layout
- Choose **Install Ubuntu**
- Let the network auto-configure -or- configure manually if needed. The system needs an internet connection.
- Select no proxy and keep the mirror address.
- Select **Use an entire disk** and confirm
- Choose user name and host name in next screen. Choose a strong password.
- Select **Install OpenSSH Server** but don't import keys
- Don't select any snaps and continue
- Finish installation and let it boot to the prompt
- Login as the created user and run a full system update using sudo apt update && sudo apt dist-upgrade -y

7.1.2 AWS

The Ubuntu AMI Id's for all regions in AWS can be found in Table 1

Also Ubuntu AMI's are listed at https://cloud-images.ubuntu.com/locator/ec2/. Search for **ebs 18.04 amd64** to get the correct version.

7.1.3 Azure

The URN for the image is Canonical:UbuntuServer:18.04-LTS:latest

7.2 Debian 9.8

7.2.1 On-Premise

- Download the NetInst ISO from https://www.debian.org/distrib/netinst
- Boot the ISO
- Select **Install** from the boot screen
- Select **English** as language
- Select Location based on actual location of the host

Region	AMI ID
ap-northeast-1	ami-0eb48a19a8d81e20b
ap-south-1	ami-007d5db58754fa284
ap-southeast-1	ami-0dad20bd1b9c8c004
ca-central-1	ami-01b60a3259250381b
eu-central-1	ami-090f10efc254eaf55
eu-north-1	ami-5e9c1520
eu-west-1	ami-08d658f84a6d84a80
sa-east-1	ami-09f4cd7c0b533b081
us-east-1	ami-0a313d6098716f372
us-west-1	ami-06397100adf427136
cn-northwest-1	ami-09b1225e9a1d84e4c
cn-north-1	ami-09dd6088c3e46151c
us-gov-west-1	ami-66bdd307
us-gov-east-1	ami-7bd2340a
ap-northeast-2	ami-078e96948945fc2c9
ap-southeast-2	ami-0b76c3b150c6b1423
eu-west-2	ami-07dc734dc14746eab
us-east-2	ami-0c55b159cbfafe1f0
us-west-2	ami-005bdb005fb00e791
ap-northeast-3	ami-0babd61cf592f1c03
eu-west-3	ami-03bca18cb3dc173c9

Table 1: Ubuntu 18.04 LTS AWS AMI Id's

- Chose a convenient keyboard layout
- Let the network auto-configure -or- configure manually if needed. The system needs an internet connection.
- Name your host. Change it from *debian* to something else
- Choose a strong root password
- create the user account and choose a strong password
- select the proper timezone
- For the partitions use **Guided - use entire disk**
- Select **All files in one partition**
- Finish partitioning and write changes to disk
- Select **No** when ask to scan more disks
- Choose a mirror close to the host
- Opt-out of the package survey
- on the **Software Selection** select only **SSH Server** and **standard system utilities**
- Install the grub bootloader to MBR and use the primary disk for that
- Finish installation and let it boot to the prompt
- Login as root and run a full system update using apt update && apt dist-upgrade -y
- Reboot

Region	AMI ID
eu-north-1	ami-043a919b6dc7c51cc
ap-south-1	ami-0b6490868957ce747
eu-west-3	ami-0cb185e7696ffe300
eu-west-2	ami-0ef10a4062f24d89d
eu-west-1	ami-035c67e6a9ef8f024
ap-northeast-2	ami-0fa1392d5d545f9e8
ap-northeast-1	ami-0c4290d7ce45d7bbe
sa-east-1	ami-0bc0ce4ab8b82305c
ca-central-1	ami-0857efbad274a1a89
ap-southeast-1	ami-04c9740a9ed018dba
ap-southeast-2	ami-0b91189c4f9f5cd9e
eu-central-1	ami-05449f21272b4ee56
us-east-1	ami-0f9e7e8867f55fd8e
us-east-2	ami-00c5940f2b52c5d98
us-west-1	ami-0afda78f1d0272d99
us-west-2	ami-01d07e14f082b3ba1

Table 2: Debian 9.8 AWS AMI's

7.2.2 AWS

The AMI Id's for all regions in AWS can be found in Table 2

You can retrieve this list also from https://wiki.debian.org/Cloud/AmazonEC2Image/Stretch

7.2.3 Azure

The URN for the image is credativ:Debian:9:latest

7.3 CentOS 7

7.3.1 On-Premise

- Download the minimal ISO from https://www.centos.org/download/
- Boot the ISO
- Confirm the automatic boot option **Test this media & install CentOS 7**
- Choose **English** as language
- On the installation summary choose "Installation destination" and confirm "automatic partinioning"
- Back on the installation summary screen click on "Network & Hostname"
- change the hostname
- enable the network interface and make sure it is configured properly
- Click **Done** to get back to the summary and click **Begin Installation**
- During installation set a root password
- Finish installation and let it boot to the prompt
- Login as root and run a system update with yum update

Region	AMI ID
ap-northeast-1	ami-25bd2743
ap-northeast-2	ami-7248e81c
ap-south-1	ami-5d99ce32
ap-southeast-1	ami-d2fa88ae
ap-southeast-2	ami-b6bb47d4
ca-central-1	ami-dcad28b8
eu-central-1	ami-337be65c
eu-west-1	ami-6e28b517
eu-west-2	ami-ee6a718a
eu-west-3	ami-bfff49c2
sa-east-1	ami-f9adef95
us-east-1	ami-4bf3d731
us-east-2	ami-e1496384
us-west-1	ami-65e0e305
us-west-2	ami-a042f4d8

Table 3: CentOS 7 AWS AMI's

7.3.2 AWS

The AMI Id's for all regions in AWS can be found in Table 3

You can retrieve the list also from https://wiki.centos.org/Cloud/AWS#head-78d1e3a4e6ba5c5a3847750d88266916ffe69648

7.3.3 Azure

The URN for the image is OpenLogic:CentOS:7.5:latest

8 Telemetry

Authority nodes have to send automatic telemetry data to NetOps. This helps detecting attacks or other network disturbances early. The following telemetry is collected and send to the telemetry SaaS provider:

- CPU usage
- Memory usage
- Disk usage
- Number of connected blockchain peers
- Current block information (hash, number of tx, blocktime)
- Network throughput
- Network error rate
- Service status for SSH, Docker and the Parity container
- Statisatics per docker container