Deploying a smart contract node

We will see how to deploy a node capable of launching smart contracts. As part of this tutorial, we use a test blockchain, whose list of master nodes is: 167.86.103.31:8080,5.189.168.49:8080,173.212.229.88:8080,62.171.153.36:8080,167.86.124.188:8080

The administrators of this blockchain are members of the Kalima Systems team. For any administrative manipulation, such as authorizing your node for example, please contact: [jerome.delaire@kalima.io](mailto:jerome.delaire@kalima.io)

# Expand node

We consider in this tutorial that the smart contract node is already developed. If this is not the case, you can develop it by following the tutorial SmartContractNode.

# Install the node

To work, the node must be installed on a linux machine with a public IP address, because the contracts will be deployed on this machine by a devOps chain via SSH. If you want to install a smart contract node in a machine that does not have a public IP address , we are able to offer an alternative solution.

As part of this tutorial, we will explain how to deploy a smart node contract on a server. The node that will be deployed is from the SmartContractNode sample code, but you can of course do the same with your own node.

First of all, we need to export the code implementing the smart contract node in the format .jar

In eclipse, this is done simply by clicking on 🡪 Java 🡪 export file 🡪 runnable jar. You then choose the correct configuration and location of your jar on your local machine. As part of our SmartContractNode example, two parameters will have to be passed when launching the node: The configuration file of the Kalima node, as well as the name of the git directory on which the contracts will be stored. For example, the jar can be launched via the following command:

java -jar SmartContractNode.jar node.config ContractsTest

Then you have to connect to your linux machine (here, a vps). It is important to remember that the latter must have a public IP to allow external access to the DevOps chain in SSH.

Once this is done, you simply need to integrate your jar on your linux machine (by command line for example; on a vps, we use the ***ssh command nom\_d utilisateur@adresse*** to connect in ssh on the specified user and at the specified address, then we use the scp command ***nameOfFileSend nomUtilisateur@AdresseServeur:~/RenameTheFile*** to transmit files).

Note that in addition to the jar, do not forget to transmit the node.config file to your machine.

# Create a git directory for contracts

The contracts are stored on a git directory, and then the encrypted contracts are deployed to the smart contract nodes. So you need to create a git directory for your smart contracts . This git directory must be readable by the devOps string.

As part of this tutorial, you can create a directory on our git server: <https://git.kalimadb.com/>

It is then necessary to give read access to the gocd user, and to indicate the link of the directory.

Recall that in argument we must **only** indicate the name of the directory. For more information about triggering smartcontracts, see the "from scratch" part of the SmartContractNode documentation.

# Copy the SSH public key

For contracts to be deployed on your machine, the devOps chain must be able to access them. To do this, you must copy the SSH public key of the devOps string to your machine. As part of this tutorial, the key has been provided to you by Kalima Systems, if not, please contact [jerome.delaire@kalima.io](mailto:jerome.delaire@kalima.io)

Move to the folder containing the key, and then use this command to copy the key:

ssh-copy-id -f -i id\_rsa.pub <user>@<ip\_cible>

With:

* user The user of your machine 🡺(for example, rcs)
* ip\_cible🡺 Your machine's IP address

# Connect the node to the Kalima blockchain

To function, the node must be authorized on the blockchain on an address list of our choice with at least one authorization on /Kalima\_Scripts (to retrieve encryption information and contract signatures).

This authorization step is done by an admin, we authorize the node via a temporary serialId (valid for 5 minutes). In order for a blockchain administrator to authorize your smart node contract, you must provide them with:

* The IP address of the machine that will host the node
* The user of the machine for whom the contracts will be deployed. If the user is rcs for example, the contracts will be deployed in /home/rcs/contracts/
* The url of the git repository
* The list of smart contracts that your node will be able to launch

Once all the steps are done, we can launch our jar with the command: [***java***](https://prograide.com/etiquetada/java) ***-***[***jar***](https://prograide.com/etiquetada/jar) ***filename.jar chemin\_du\_fichier\_node\_config nom\_du\_répertoire\_git***