# aReadme document for MVP project

Minimal Viable Product project consists in development of SW for Color blockchain running on cloud environment in various data centers support Color platform. This SW allows developer to design and debug dApps for Color platform before the final version of Color platform be completed.

Installation of Minimal Viable Product is the first stage. Now we consider this process in details as follows. Our instruction is applicable without changes for debian-based linux. It could be applied to other operation systems with some changes.

At first let us describe HW requirements.

### Hardware requirements for Minimal Viable Product

Minimal version can be installed on following configuration:

* System with VT-x or AMD-v virtualization support (check in BIOS)
* Minimum 16 GB of memory
* Minimum 100 GB of hard disk space
* Minimum 1 CPU with 8 cores

It is higly recommended to use SSD hard disk.

### Content of Minimal Viable Product

1. Kubernetes (System for control docker containers – Open Source SW, see: [https://kubernetes.io](https://kubernetes.io/))
2. Kong (API platform for micro services – Open Source SW, see: [https://konghq.com](https://konghq.com/))
3. Kubernetes Ingress Controller for Kong allows to use Kong as a load balancer of Kubernetes – Open Source SW (see <https://github.com/Kong/kubernetes-ingress-controller>)
4. Mongo DB – OpenSource nonSQL DB (see: [https://mongodb.com](https://mongodb.com/))
5. Consensus algorithm and its implementation taken from Hyperledger Fabric

The principal scheme of MVP is shown on the following figure.

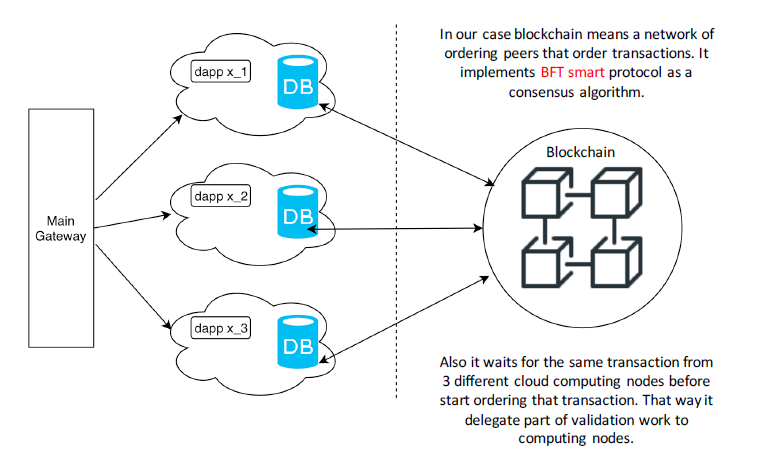


Figure 1. Scheme of MVP

### Kubernetes installation

Following instructions are for debian-based system. Following instructions should be executed from project folder. Full path to project folder should not contain non-latin symbols.

1. Update system packages

sudo apt-get update

1. Install virtualbox

sudo apt-get install -y virtualbox-5.2

1. Install docker

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository "deb [arch=amd64] <https://download.docker.com/linux/ubuntu> $(lsb\_release -cs) stable"

sudo apt-get install -y docker-ce

1. Setup docker to be sudoless

sudo groupadd docker

sudo gpasswd -a $USER docker

newgrp docker

1. Install kubectl

sudo apt-get install -y apt-transport-https

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl

1. Install minikube

curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.30.0/minikube-linux-amd64 && chmod +x minikube && sudo cp minikube /usr/local/bin/ && rm minikube

1. Install docker-compose

sudo apt-get install -y docker-compose

1. Install mongo client

sudo apt-get install -y mongodb-clients

### Running mvp stand with tests

This chapter heavily uses project files

1. Setup hyperldger fabric network

./setup\_network.sh

sudo cp fabric-samples/bin/configtxgen /usr/bin/

1. Start hyperledger network. This commands should be executed in different terminal

newgrp docker

docker-compose -f test/fixtures/docker-compose-2orgs-4peers-tls.yaml up –force-recreate

1. Create python virtual environment

sudo apt-get install python3-venv -y

python3 -m venv env

source env/bin/activate

pip install –upgrade pip

1. Install python requirements

pip install -r unfiltered\_requirements.txt

1. Initialize chaincode

python initialize\_chaincode.py

1. Start first minikube cluster

minikube start -p test

* 1. If there is a following error: “Error restarting cluster: restarting kube-proxy: waiting for kube-proxy to be up for configmap update: timed out waiting for the condition.” Execute next commands:

minikube delete -p test

minikube start -p test

1. Install Kong as load balancer

cat all-in-one-postgres.yaml | kubectl create -f -

1. Build dapp and polling service images, using minikube inner docker registry

eval $(minikube -p test docker-env)

docker build -t wallet:v0.1 -f Dockerfile\_wallet .

docker build -t polling:v0.1 -f Dockerfile\_polling .

1. Create MongoDB deployment

cat mongo.yaml | kubectl create -f -

1. Initiate MongoDB cluster (with one node) to allow transactional mode. Wait some time before this commands while mongodb is still starting up. (You can track this progress using “kubectl get pods” command).

export MONGO\_IP=$(minikube -p test service mongo --url --format "{{ .IP }}" | head -1)

export MONGO\_PORT=$(minikube -p test service mongo --url --format "{{ .Port }}" )

mongo ${MONGO\_IP}:${MONGO\_PORT} --eval "rs.initiate()"

1. Create Kubernetes deployment and service resources

cat wallet-application.yaml | kubectl create -f -

cat polling.yaml | kubectl create -f -

1. Create ingress rule for dapp

cat wallet\_ingress\_rule | kubectl create -f -

1. Create second minikube cluster. Wait while “kubectl get pods” doesn’t show all pods as running.

minikube start -p test2

1. Repeat previous steps for second cluster

cat all-in-one-postgres.yaml | kubectl create -f -

eval $(minikube -p test2 docker-env)

docker build -t wallet:v0.1 -f Dockerfile\_wallet .

docker build -t polling:v0.1 -f Dockerfile\_polling .

cat mongo.yaml | kubectl create -f -

export MONGO\_IP=$(minikube -p test2 service mongo --url --format "{{ .IP }}" | head -1)

export MONGO\_PORT=$(minikube -p test2 service mongo --url --format "{{ .Port }}" )

# Wait here some time while mongodb is starting up (You can track this progress using “kubectl get pods” command).

mongo ${MONGO\_IP}:${MONGO\_PORT} --eval "rs.initiate()"

cat wallet-application.yaml | kubectl create -f -

cat polling.yaml | kubectl create -f -

cat wallet\_ingress\_rule | kubectl create -f -

1. Edit test.py and tester.py files, changing moscow\_host and stpeter\_host variables to results of next commands:

minikube -p test -n kong service kong-proxy --url | head -1

minikube -p test2 -n kong service kong-proxy --url | head -1

1. Run tests. Wait while “kubectl get pods” doesn’t show all pods as running.

python test.py

Important comments:

* To kill first cluster use command “minikube delete -p test”. To kill second use command “minikube delete -p test2”. This allows to begin from scratch in case of problems.
* You may need to vary parameter replicas in wallet-application.yaml. This could be needed in case of huge load of processor. If number of replicas is too low you may also need to increase sleeping times in test.py and tester.py.