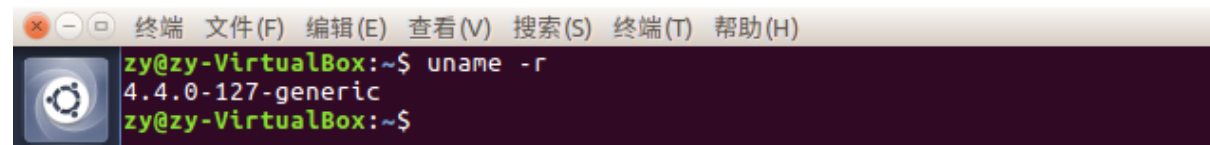


## Docker Ubuntu 安装

1. Docker 要求 Ubuntu 系统的内核版本高于 3.10，查看本页面的前提条件来验证你的 Ubuntu 版本是否支持 Docker。

通过 `uname -r` 命令查看你当前的内核版本



```
zy@zy-VirtualBox:~$ uname -r
4.4.0-127-generic
zy@zy-VirtualBox:~$
```

2. 获取最新版本的 Docker 安装包

`wget -qO- https://get.docker.com/ | sh`

输入当前用户的密码后，就会下载脚本并且安装 Docker 及依赖包

3. 当要以非 root 用户可以直接运行 docker 时，需要执行 `sudo usermod -aG docker runoob` 命令，然后重新登陆

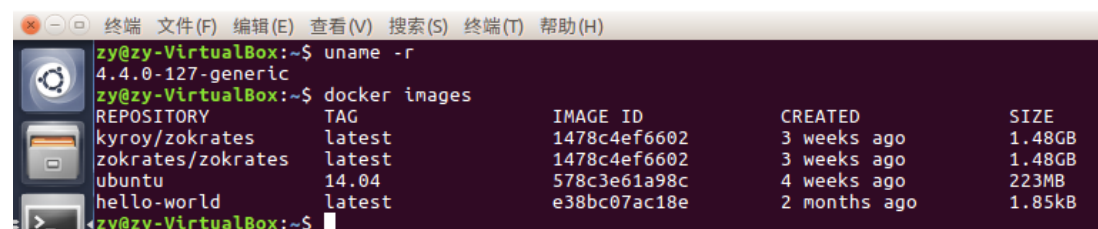
## 部署 zokrates

推荐使用该方法开始 zokrates

`docker run -ti zokrates/zokrates /bin/bash`

初次执行该条命令 docker 会自动从 docker 镜像仓库中下载 zokrates，默认是从 Docker Hub 公共镜像源下载。

可以通过 `docker images` 命令查看已下载镜像

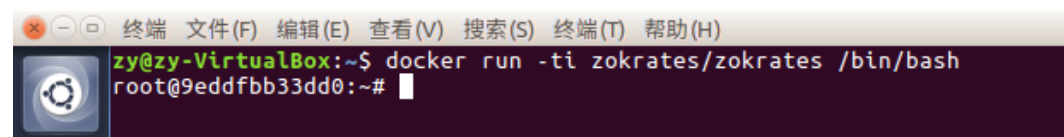


```
zy@zy-VirtualBox:~$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
kyroy/zokrates	latest	1478c4ef6602	3 weeks ago	1.48GB
zokrates/zokrates	latest	1478c4ef6602	3 weeks ago	1.48GB
ubuntu	14.04	578c3e61a98c	4 weeks ago	223MB
hello-world	latest	e38bc07ac18e	2 months ago	1.85kB

```
zy@zy-VirtualBox:~$
```

然后进入交互模式

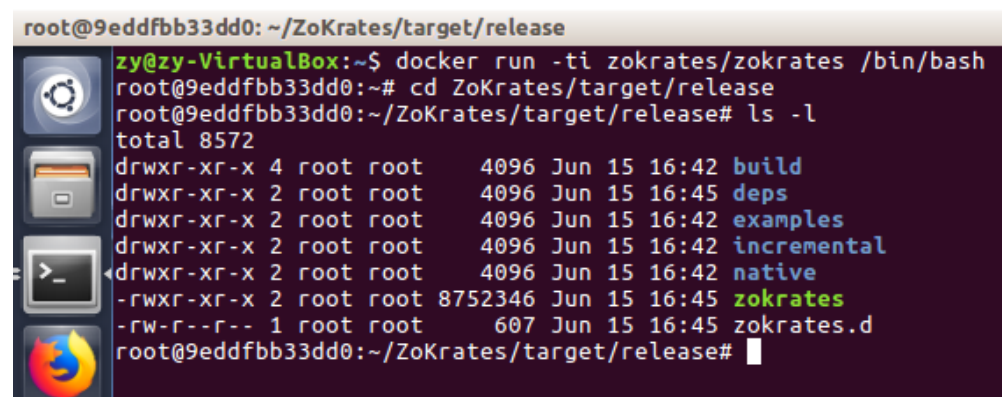


```
zy@zy-VirtualBox:~$ docker run -ti zokrates/zokrates /bin/bash
root@9eddfbb33dd0:~#
```

在交互模式下

`cd ZoKrates/target/release`

该文件夹下有可执行文件 zokrate:



```
root@9eddfbb33dd0: ~/ZoKrates/target/release
```

```
zy@zy-VirtualBox:~$ docker run -ti zokrates/zokrates /bin/bash
root@9eddfbb33dd0:~# cd ZoKrates/target/release
root@9eddfbb33dd0:~/ZoKrates/target/release# ls -l
```

permissions	links	owner	group	size	date	time	file
drwxr-xr-x	4	root	root	4096	Jun 15	16:42	build
drwxr-xr-x	2	root	root	4096	Jun 15	16:45	deps
drwxr-xr-x	2	root	root	4096	Jun 15	16:42	examples
drwxr-xr-x	2	root	root	4096	Jun 15	16:42	incremental
drwxr-xr-x	2	root	root	4096	Jun 15	16:42	native
-rwxr-xr-x	2	root	root	8752346	Jun 15	16:45	zokrates
-rw-r--r--	1	root	root	607	Jun 15	16:45	zokrates.d

```
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

Zokrates 提供命令行界面。您可以通过运行 `./zokrates` 查看可用子命令的概述:

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ./zokrates
ZoKrates 0.1
Jacob Eberhardt, Dennis Kuhnert
Supports generation of zkSNARKs from high level language code including Smart Contracts for proof verification on the
Ethereum Blockchain.
'I know that I show nothing!'

USAGE:
  zokrates <SUBCOMMAND>

FLAGS:
  -h, --help      Prints help information
  -V, --version    Prints version information

SUBCOMMANDS:
  compile          Compiles into flattened conditions. Produces two files: human-readable '.code' file and
                   binary file
  compute-witness  Calculates a witness for a given constraint system, i.e., a variable assignment which
                   satisfies all constraints. Private inputs are specified interactively.
  export-verifier  Exports a verifier as Solidity smart contract.
  generate-proof   Calculates a proof for a given constraint system and witness.
  help            Prints this message or the help of the given subcommand(s)
  setup           Performs a trusted setup for a given constraint system.
```

执行一个实例(位于 `~/ZoKrates/examples/add.code`)

① `./zokrates compile -i ~/ZoKrates/examples/add.code`

编译一个 `.code` 文件

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ./zokrates compile -i ~/ZoKrates/examples/add.code
Compiling /root/ZoKrates/examples/add.code
Compiled program:
def main(a):
    b = (a + 5)
    c = (((a + b) + a) + 4)
    d = (((a + c) + a) + b)
    return ((b + c) + d)
Compiled code written to 'out',
Human readable code to 'out.code'.
Number of constraints: 4
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

这个命令会创建一个编译好的 `out.code` 文件

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ls
build  deps  examples  incremental  native  out  out.code  zokrates  zokrates.d
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

② 计算在 `out.code` 中找到的已编译程序和程序的参数的见证。见证是包括计算结果的变量的有效分配。

`./zokrates compute-witness -a 5`

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ./zokrates compute-witness -a 5
Computing witness for:
Witness: {"a": 5, "b": 10, "c": 24, "d": 44, "~one": 1, "~out_0": 78}
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

这行命令创建一个 `witness` 的文件

③ 为 `out.code` 中的已编译程序生成可信设置, 在 `proving.key` 和 `verifying.key` 上创建一个证明密钥和一个验证密钥。

`./zokrates setup`

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ./zokrates setup
Performing setup...
def main(a):
    b = (a + 5)
    c = (((a + b) + a) + 4)
    d = (((a + c) + a) + b)
    return ((b + c) + d)
num variables: 6
num constraints: 1
num inputs: 2
Swap is not beneficial, not performing
* QAP number of variables: 5
* QAP pre degree: 1
* QAP degree: 4
* QAP number of input variables: 2
* G1 window: 3
* G2 window: 1
* G1 elements in PK: 49
* Non-zero G1 elements in PK: 28
* G2 elements in PK: 7
* Non-zero G2 elements in PK: 2
* PK size in bits: 8670
* G1 elements in VK: 4
* G2 elements in VK: 5
* VK size in bits: 3948
Verification key in Solidity compliant format:
vk.h = Pairing.G2Point([0x2674d0f3eefa93280aae4b4343766c74b9f40ce3c7e6b4897e09bd0c027b99, 0x10f126534e5bf30073c9ede1ad5069f160cb16d580fd201f2ee
6f93a8e12ab5], [0x1b7bb897ef93b4249db10ef805bd26f4b3ab0d44cd8d75de0c19f7a64c6292c6, 0x3e1546b9baab629730461a1712c1ebc38aa855659e6225f43a6aed53009f185]);
vk.B = Pairing.G1Point([0x2872ccc145d78bcb4ae5ceda992c9e8a3459e296644cc73ca354c0d8ccbe822b, 0x1286a7ed6b929a1220c3147aa00d5612b02b9cb353593a8a3f68
a6c7c8cd5178]);
vk.C = Pairing.G2Point([0x1f0b5b7f6a449fdd5a366626f4a0581d3130352b0c1e07d53ee3f1fd9883cbd, 0x2ed56fd88347e66c0398d50d1d262f5179f528c76c106e7fc88
d4b69bce0aea], [0x14ffea01f3aeffad4957f724c97c09ac37fd5a9757e48bb9f2a46e98f51f7, 0x1db51b52e8bb8a5c9a0b4cf708ea56a81f150d444315f13b337aae2a6f320f13]);
vk.gamma = Pairing.G2Point([0x2c1e904320d0a999614b7170e2865066450be841a0106b02b28762ce58bbaab4, 0x8090bb0cea30e3d670d30e42afe0c85844c9ffbf6bf
a6e7acd6a10e239c], [0x1ea5bbea5318fb67ac55060ea04c39f5ce66298094470f6ac73e86ac4546f74, 0x2f5ce24fd247a94d79190d7339de8d1ed687b2ee46939527844df79d6f5239a]);
vk.gammaBeta1 = Pairing.G1Point([0x7808a316ff1c8474362d63a436d35cabaf48efc189fa35921b88e1656dcf698, 0x10320d2746493d4de0396b04dab399b2e9e73a66051b
dc5d7dbed159eb78c3]);
vk.gammaBeta2 = Pairing.G2Point([0x2369e54650809fcfb7affed5af5a59a5d0edf6baa0386359409b5b0e988ecf756, 0x8d0e9470c7d94f493560c67ba5460d78c01b5af887
54bcff4d283c18d93410c], [0x1cadb18238b79ccb810f896b3027d99391602f1e3efa463c8928e40bec425cd3, 0x2cc95072943a3f268c5ed9c2629988534fcc3702af76d2368cacad0d0cdeb5]);
vk.Z = Pairing.G2Point([0x1d03169c016fb76d02eb121f59ebc1b5e109a63a101a7b1394fb59d230fc94, 0xece72cb7793f039efc386d1fd5d1827d5f2becbb0b1c582334a
48e3185965fe], [0x1b04db4bb37c2a0b0b4ca5da591632c388908a4ba065389b95771ef84d66d099, 0xd0a4cae0eb121fe202fccbbe0c82f8c1bbf4edc19b168d405bdc143253eb7]);
```

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ls -l
total 8596
drwxr-xr-x 4 root root      4096 Jun 15 16:42 build
drwxr-xr-x 2 root root      4096 Jun 15 16:45 deps
drwxr-xr-x 2 root root      4096 Jun 15 16:42 examples
drwxr-xr-x 2 root root      4096 Jun 15 16:42 incremental
drwxr-xr-x 2 root root      4096 Jun 15 16:42 native
-rw-r--r-- 1 root root        310 Jul  9 07:11 out
-rw-r--r-- 1 root root         98 Jul  9 07:11 out.code
-rw-r--r-- 1 root root       2164 Jul  9 07:31 proving.key
-rw-r--r-- 1 root root         67 Jul  9 07:31 variables.inf
-rw-r--r-- 1 root root       2183 Jul  9 07:31 verification.key
-rw-r--r-- 1 root root         36 Jul  9 07:19 witness
-rwxr-xr-x 2 root root    8752346 Jun 15 16:45 zokrates
-rw-r--r-- 1 root root         607 Jun 15 16:45 zokrates.d
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

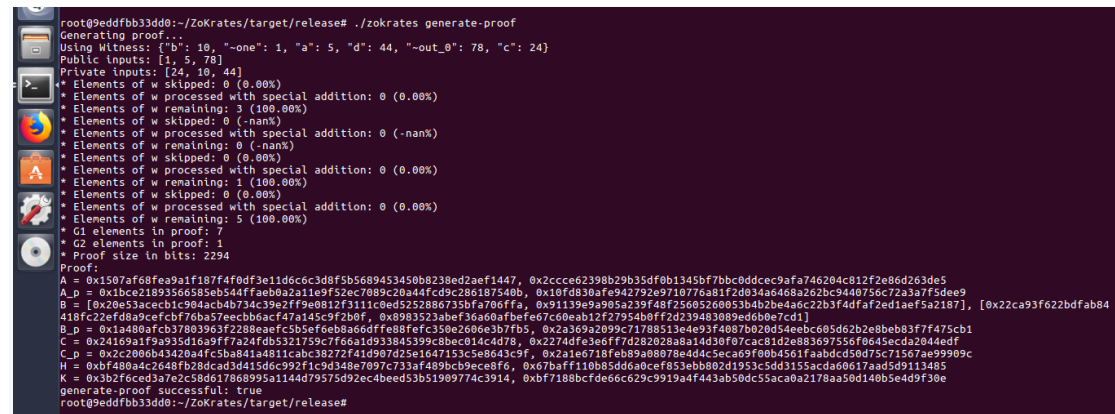
④使用./verifying.key 上的验证密钥，生成一个 Solidity 合约，以便在验证已编译程序 out.code 的计算证明。在./verifier.sol 创建验证者合约。

[./zokrates export-verifier](#)

```
root@9eddfbb33dd0:~/ZoKrates/target/release# ls -l
total 8608
drwxr-xr-x 4 root root      4096 Jun 15 16:42 build
drwxr-xr-x 2 root root      4096 Jun 15 16:45 deps
drwxr-xr-x 2 root root      4096 Jun 15 16:42 examples
drwxr-xr-x 2 root root      4096 Jun 15 16:42 incremental
drwxr-xr-x 2 root root      4096 Jun 15 16:42 native
-rw-r--r-- 1 root root        310 Jul  9 07:11 out
-rw-r--r-- 1 root root         98 Jul  9 07:11 out.code
-rw-r--r-- 1 root root       2164 Jul  9 07:31 proving.key
-rw-r--r-- 1 root root         67 Jul  9 07:31 variables.inf
-rw-r--r-- 1 root root       2183 Jul  9 07:31 verification.key
-rw-r--r-- 1 root root      11430 Jul  9 07:41 verifier.sol
-rw-r--r-- 1 root root         36 Jul  9 07:19 witness
-rwxr-xr-x 2 root root    8752346 Jun 15 16:45 zokrates
-rw-r--r-- 1 root root         607 Jun 15 16:45 zokrates.d
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

⑤使用./proving.key 中的证明密钥，生成用于计算已编译程序 out.code 的证明，从而验证 witness。

`./zokrates generate-proof`



```
root@9eddfbb33dd0:~/ZoKrates/target/release# ./zokrates generate-proof
Generating proof...
Using Witness: {"b": 10, "-one": 1, "a": 5, "d": 44, "-out_0": 78, "c": 24}
Public inputs: [1, 5, 78]
Private inputs: [24, 10, 44]
* Elements of w skipped: 0 (0.00%)
* Elements of w processed with special addition: 0 (0.00%)
* Elements of w remaining: 3 (100.00%)
* Elements of w skipped: 0 (-nan%)
* Elements of w processed with special addition: 0 (-nan%)
* Elements of w remaining: 0 (-nan%)
* Elements of w skipped: 0 (0.00%)
* Elements of w processed with special addition: 0 (0.00%)
* Elements of w remaining: 1 (100.00%)
* Elements of w skipped: 0 (0.00%)
* Elements of w processed with special addition: 0 (0.00%)
* Elements of w remaining: 5 (100.00%)
* G1 elements in proof: 7
* G2 elements in proof: 1
* Proof size in bits: 2294
Proof:
A = 0x1507af687e9a1f187f4f0df3e11d6c6c3d8f5b5689453450b8238ed2aef1447, 0x2ccce62398b29b35df0b1345bf7bbc0ddcec9afa746204c812f2e86d263de5
A_p = 0x1bce21893566585eb544f4faeb0a2a11e9f52ec7089c20a44fcd9c286187540b, 0x10fd830afe942792e9710776a81f2d034a6408a262bc9440756c72a3a7f5dee9
B = [0x20e53aceb1c904acb4b734c39e2ff9e0812f3111c0ed5252886735bfa706ffa, 0x91139e9a905a239f48f25605260053b4b2be4a6c22b3f4dfaf2ed1aef5a2187], [0x22ca93f622bdfab84
410fc22efda09cfcfb70ba57ecbb6ac47a145c9f2b0f, 0x8983523abef36a60a0bfe67c6beab12f7954b0ff72d239483089ed6b0e7cd1]
B_p = 0x1a40a1cb37003963f2280eafcc505efebba90dfef88f8fc350e260e3b77b5, 0x2a369a2099c17188513e4e93f4087b02d054eebc005d62b2e8be83ff7f475cb1
C = 0x24109a1f9a935d16a9ff7a24fdb5321759c7f6eaid933845399c8bec014c4d78, 0x2274dfe3e6ff7d282028a8a14d38f07cac81d2e883697556f0645ecd2044edf
C_p = 0x2c2006b4320a4fc5ba841a4811cab38272f41d907d25e1647153c5e8643c9f, 0x2a1e6718feb89a88078e4d4c5eca69f80b4561faabdc5d075c71567ae99909c
H = 0xbf480a4c2648f02b8cd3d415dc992f1c9d348e7097c733af489bcb9ece8f6, 0x67baff110b85dd0a0cef853ebb02d1953c5dd3155acda60617aad59113485
K = 0x302f6ced3a7e2c58d617d08995a1144d79575d92ec4beed53b51909774c3914, 0xbf7108bcfd6e6c029c919a4f443ab50dc55aca0a2178aa50d140b5e4d9f30e
generate-proof successful: true
root@9eddfbb33dd0:~/ZoKrates/target/release#
```

⑥通过验证者合约，可以检查此证明。例如，使用 web3，调用将如下所示

`sudo docker cp 9eddfbb33dd0:/root/ZoKrates/target/release/verifier.sol /opt/soft/`

//将 verifier.sol 从 docker 容器中移出

//如何进入上次启动的容器

//docker ps -a

//docker start 9eddfbb33dd0

//docker exec -ti 9eddfbb33dd0 /bin/bash

//不然重新 docker run 的话会重新开一个容器

利用 truffle 及 ganache-cli 部署该合约

truffle console

>>Verifier.at(<verifier contract address>).verifyTx(A, A\_p, B, B\_p, C, C\_p, H, K, [...publicInputs, ...outputs])

//[5,78]

ZoKrates 调用的 libsnark 接口

//在 ZoKrates/lib 中的 wraplibsnark.cpp wraplibsnark.hpp 中再次封装

//下面是主要调用的函数

```
#include "libsnark/algebra/curves/alt_bn128/alt_bn128_pp.hpp"
```

// alt\_bn128\_pp 是一种椭圆曲线

```
#include "libsnark/zk_proof_systems/ppzksnark/r1cs_ppzksnark/r1cs_ppzksnark.hpp"
```

//包含 zksnark 主要函数

① `r1cs_ppzksnark_constraint_system<alt_bn128_pp> createConstraintSystem(const uint8_t* A, const uint8_t* B, const uint8_t* C, int constraints, int variables, int inputs)`

//创建约束系统

```
// r1cs_constraint_system<Fr<alt_bn128_pp>> cs;
```

//调用类 r1cs\_constraint\_system 默认构造函数

② `r1cs_ppzksnark_keypair<alt_bn128_pp> generateKeypair(const r1cs_ppzksnark_constraint_system<alt_bn128_pp> &cs)`

//调用 r1cs\_ppzksnark\_generator<alt\_bn128\_pp>(cs);

//生成 verification\_key 和 proving\_key

//下面是两个最主要的函数

③ `bool_setup(const uint8_t* A, const uint8_t* B, const uint8_t* C, int constraints, int variables, int inputs, const char* pk_path, const char* vk_path)`

// alt\_bn128\_pp::init\_public\_params(); 初始化椭圆曲线参数

// r1cs\_constraint\_system<Fr<alt\_bn128\_pp>> cs; 初始化约束系统

// cs = createConstraintSystem(A, B, C, constraints, variables, inputs); 为 cs 赋值

//r1cs\_ppzksnark\_keypair<alt\_bn128\_pp> keypair= r1cs\_ppzksnark\_generator<alt\_bn128\_pp>(cs); 生成 verification\_key 和 proving\_key

//serializeProvingKeyToFile(keypair.pk, pk\_path);

//serializeVerificationKeyToFile(keypair.vk, vk\_path); 将 pk、vk 输出到文件中

④ `bool_generate_proof(const char* pk_path, const uint8_t* public_inputs, int public_inputs_length, const uint8_t* private_inputs, int private_inputs_length)`

// r1cs\_ppzksnark\_proof<alt\_bn128\_pp> proof = r1cs\_ppzksnark\_prover<alt\_bn128\_pp>(pk, primary\_input, auxiliary\_input); 产生证明