

Bastion Secured Cloud

«Bastion Secured Cloud» (Далее BSC) – продукт для организации корпоративной виртуальной рабочей среды, предоставляющий сервис совместной и безопасной работы с файлами.

BSC позволяет хранить и передавать чувствительную информацию как внутри корпоративной сети, так и вашим партнёрам с удобством мобильного приложения и безопасностью хранилища со сквозным (e2e) шифрованием.

1. Краткое описание структуры администрируемого комплекса

С помощью BSC организации получают надежный инструмент облачного хранения и совместной работы с информацией по принципу on- premise на собственной инфраструктуре. При этом решение позволяет обмениваться данными с внешними пользователями за периметром организации.

В состав продукта входят следующие приложения, позволяющие работать в частном облаке (Рисунок 1) :

- Система хранения данных – система, обеспечивающая безопасное хранение и обмен корпоративными документами и файлами между сотрудниками;
- Система визуализации интерфейса и взаимодействия с хранилищем данных - позволяет пользователю работать в любом браузере с файлами и взаимодействовать с системой хранения в безопасном режиме.

2. Исполнение в аппаратно-программном комплексе

2.1 Аппаратные требования

Требования к устройству для запуска и тестирования:

- Соединение с Интернет
- Докер на Linux
- Папку с установочными файлами
- Устройство с минимальными характеристиками:
 - CPU
 - RAM
 - Disk
- ПО установленное на устройстве:
 - Дистрибутив Linux (Ubuntu любой [актуальной версии](#))
 - Docker
 - Docker-compose
 - make
 - wget

2.2 Подготовка среды

- Для установки **Docker** последовательно выполняем команды

1. `$ sudo apt-get update`
2. `$ sudo apt-get install apt-transport-https ca-certificates curl software-properties-common`
3. `$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
4. `$ sudo apt-key fingerprint 0EBFCD88`
5. `$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
6. `$ sudo apt-get update`
7. `$ sudo apt-get install docker-ce`
8. `$ sudo docker -v`

Если после 8 пункта мы получили версию Docker, значит установка завершена успешно.

- Для установки **Docker-compose** последовательно выполняем команды

1. `$ sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose`
2. `$ sudo chmod +x /usr/local/bin/docker-compose`
3. `$ docker-compose -version`

Если после 3 пункта мы получили версию Docker-compose, значит установка завершена успешно

- Для установки **make** последовательно выполняем команды

1. `$ sudo apt update`
2. `$ sudo apt install make`
3. `$ make -version`

Если после 3 пункта мы получили версию make, значит установка завершена успешно

- Для установки wget, войдите через SSH от имени пользователя root и выполните команды:

- a. `$ apt-get install wget`
- b. Нажмите Y: `$ dpkg -l | grep wget`

2.3 Установка на физическое оборудование

1. На дистрибутиве Linux Ubuntu скачиваем папку с установочными файлами и сохраняем её в директорию build, переходим в эту папку. Выполняем команду: `$ cd build`
2. Поднимаем локально базу и S3 хранилище. Выполняем команду:
`$ docker-compose up -d db minio`

```
/tmp/build
> ls
Makefile          api.tar          docker-compose.yml etc          sql            tools           web

/tmp/build
> docker-compose up -d db minio
Creating network "bsc_net" with the default driver
Creating volume "bsc_data" with default driver
Creating bscminio ... done
Creating bsc_db ... done
```

3. Скачиваем вспомогательные инструменты: переходим в папку tools:

`$ cd tools`

Выполняем команды:

- > `wget https://dl.min.io/client/mc/release/linux-amd64/mc`
- > `chmod +x mc`
- `$ wget https://github.com/golang-migrate/migrate/releases/download/v4.14.1/migrate.linux-amd64.tar.gz`
- > `tar xfv migrate.linux-amd64.tar.gz`
- > `mv migrate.linux-amd64 migrate`
- > `chmod +x migrate`

```
/tmp/build
> cd tools

/tmp/build/tools
> wget https://dl.min.io/client/mc/release/darwin-amd64/mc
--2021-02-19 16:43:13-- https://dl.min.io/client/mc/release/darwin-amd64/mc
Resolving dl.min.io Cdl.aiin.io)... 178.128.69.202
Connecting to dl.min.io Cdl.min.io(178.128.69.202):443... connected.
HTTP request sent* awaiting response... 200 OK
Length: 21681560 CZ1M) fapplication/octet-stream]
Saving to: 'mc'

mc
100% [----->] 20.68M ZZ1KB/s in 4m 17s

2021-02-19 16:47:31 QBZ.S KB/s) - 'mc' saved CZ1681560/Z1681560]

/tmp/build/tools ZSSs
> chmod +x mc

/tmp/bu i1d/tools
> wget https://github.com/golang-migrate/migrate/releases/download/v4.14.1/migrate.darwin-amd64.tar.gz
```

```
/tmp/build/tools
> tar xfv migrate.darwin-amd64.tar.gz
x ./migrate.darwin-amd64

/tmp/bu i1d/tools
> mv migrate.darwin-amd64 migrate

/tmp/bu i1d/tools
> chmod +x migrate
```

4. Поднимаем рабочее окружение: возвращаемся на уровень выше \$ cd ..

Выполняем команду:

\$ make migrate minio

```
/tmp/build
> make migrate minio
./tools/migrate -source file://sql -database "postgres://user:pass@127.0.0.1:5432/bsc?sslmode=disable" up

1/u init (52.02259ms)
2/u create_acinin (65.850482ms)
3/u sessions (89.978352ms)
4/u settings (108.985481ms)
5/u tfa_confirm (125.498842ms)
6/u alter_files (140.783902ms)
7/u remove_default_folders (156.07168ms)
8/u url_shares (177.150138ms)
9/u s3_name_index (191.750493ms)
10/u file_delete_trigger (208.332907ms)
11/u reg_confirm (226.401032ms)
12/u pass_reset (191.594759ms)
13/u smtp_settings (192.718723ms)
14/u shares (183.7682ms)
15/u unique_file_name (180.598382ms)
16/u update_deleted_at (180.660154ms)
17/u common_access (184.817013ms)
18/u files_full_path (183.335362ms)
19/u files_enlarge_name (179.253893ms)
20/u user_is_active (177.578368ms)
21/u settings_shares_password (175.278866ms)
22/u settings_access_tfa_only (171.814579ms)
23/u url_shares_mode (174.203139ms)
24/u url_shares_fix_expires_at (173.439077ms)
25/u files_user_null (169.857033ms)
26/u url_shares_set_expires_at (167.426806ms)
27/u change_len_file_deleted_name (162.926276ms)
./tools/mc config host add bsc http://127.0.0.1:9000 access_123 secret_123 --api s3v4
Added 'bsc' successfully.
./tools/mc mb bsc/bsc-files

Bucket created successfully 'bsc/bsc-files'.
```

```
/tmp/build >

/tmp/build >
```

5. Устанавливаем докер образ. Выполняем команду:

\$ docker load -i api.tar

```
/tmp/build
> docker load -i api.tar
96217705cb94: Loading layer [
5f972858d21a: Loading layer [
Loaded image: bsc_api:latest
s] 2.048kB/2.048kB
n] 30.16MB/30.16MB
```

6. Поднимаем бекенд. Выполняем команду: \$ docker-compose up api

```
docker-compose (docker-compose)
h-fm"}
bsc.api 1 2021-02-19T13:50:21.861Z      DEBUG registered route      {"path": "/archive/generate", "method": "POST", "handler": "file.(\"Handler\").Ge
nerateArchive-fm"}
bsc.api 1 2021-02-19T13:50:21.8612      DEBUG registered route      {"path": "/settings/domatns", "method": "PUT", "handler": "settings.(\"Handler)
.UpdateDomains-fm"} bsc.api 1 2021-02-
19T13:50:21.8612      DEBUG registered route      {"path": "/settings/domains/check", "method": "PUT", "handler": "settings.(\"Ha
ndler\").UpdateDomainsCheck-fm"} bsc.api 1 2021-
02-19T13:50:21.861Z      DEBUG registered route      {"path": "/settings/smp", "method": "PUT", "handler": "settings.(\"Handler\").Up
dateSMTP-fm"} bsc.api 1 2021-02-
19T13:50:21.861Z      DEBUG registered route      {"path": "/user/tfa", "method": "PUT", "handler": "user.(\"Handler\").SettingsTFA
-fm"}
bsc.api 1 2021-02-19T13:50:21.861Z      DEBUG registered route      {"path": "/user/password", "method": "PUT", "handler": "user.(\"Handler\").Passwo
rdChange-fm"}
bsc.api 1 2021-02-19T13:50:21.8612      DEBUG registered route      {"path": "/file/id/trash", "method": "PUT", "handler": "file.(\"Handler\").MoveT
oTrash-fm"} bsc.api 1 2021-02-19T13:50:21.8612
DEBUG registered route      {"path": "/file/id/restore", "method": "PUT", "handler": "file.(\"Handler\").Res
tore-fm"} bsc.api 1 2021-02-19T13:50:21.8612
DEBUG registered route      {"path": "/auth/refresh", "method": "PUT", "handler": "user.(\"Handler\").Refresh
-fm"} bsc.api 1 2021-02-19T13:50:21.861Z
DEBUG registered route      {"path": "/file/id", "method": "DELETE", "handler": "file.(\"Handler\").DeleteCo
mpletely-fm"} bsc.api 1 2021-02-19T13:50:21.861Z
DEBUG registered route      {"path": "/file/id/access", "method": "DELETE", "handler": "file.(\"Handler\").R
evokeAccess-fm"}
bsc.api 1 2021-02-19T13:50:21.861Z      DEBUG registered route      {"path": "/user/logout", "method": "DELETE", "handler": "user.(\"Handler\").Logou
t-fm"}
bsc.api 1 2021-02-19T13:50:21.8612      DEBUG registered route      {"path": "/auth/refresh", "method": "DELETE", "handler": "user.(\"Handler\").Refr
esh-fm"} bsc.api 1 2021-02-19T13:50:21.8612
DEBUG registered route      {"path": "/share/shareID", "method": "DELETE", "handler": "file.(\"Handler\").De
leteURLShare-fm"} bsc.api 1 2021-02-
19T13:50:21.8612      DEBUG registered route      {"path": "/trash/clear", "method": "DELETE", "handler": "file.(\"Handler\").Clear
Trash-fm"} bsc.api 1 2021-02-19T13:50:21.861Z
DEBUG registered route      {"path": "/auth/refresh", "method": "HEAD", "handler": "user.(\"Handler\").Refres
h-fm"} bsc.api 1 2021-02-19T13:50:21.861Z
DEBUG registered route      {"path": "/auth/refresh", "method": "OPTIONS", "handler": "user.(\"Handler\").Ref
resh-fm"}
bsc.api 1 2021-02-19T13:50:21.861Z      DEBUG registered route      {"path": "/auth/refresh", "method": "CONNECT", "handler": "user.(\"Handler\").Ref
resh-fm"} bsc.api 1 2021-02-19T13:50:21.8612
DEBUG registered route      {"path": "/auth/refresh", "method": "TRACE", "handler": "user.(\"Handler\").Refre
sh-fm"}
bsc.api 1 2021-02-19T13:50:21.8632      WARN starting server {"addr" : ":8080"}
bsc.api 1 2021-02-19T13:50:21.8632      WARN starting server {"addr" : ":8081"}
```

7. Поднимаем фронтенд, Для этого переходит в новое окно терминала и выполняем команды: \$ cd web
\$ docker-compose up --build

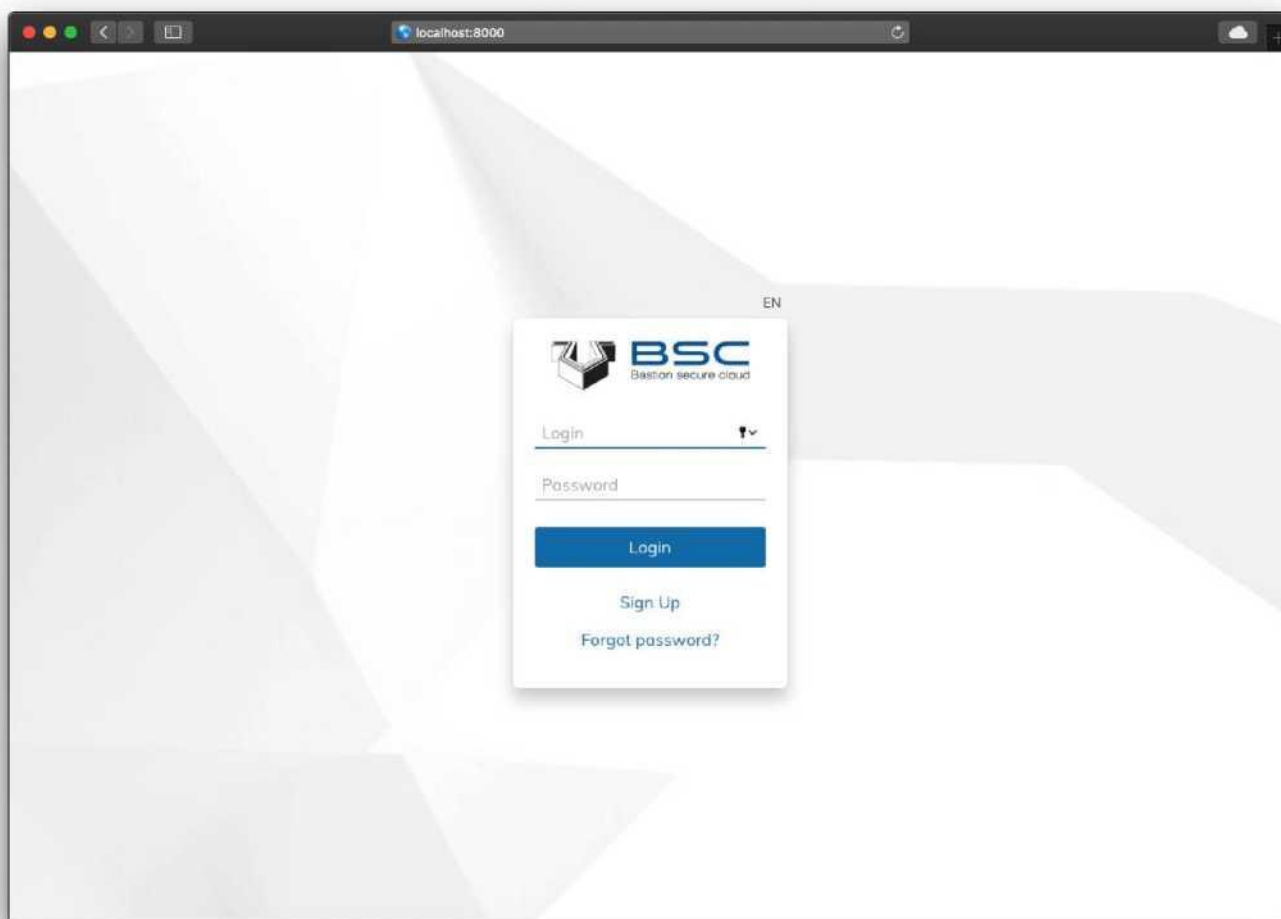
```
/tmp/build > cd web

/tmp/build/web
> docker-compose up --build

Building web
Step 1/2 : FROM nginx
--> f6d0b4767a6c
Step 2/2 : COPY nginx.conf /etc/nginx/conf.d/default.conf
--> Using cache
--> de8fca631203

Successfully built de8fca631203
Successfully tagged bsc_web:latest
Creating bsc_web ... done
Attaching to bsc_web
bsc_web | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
bsc_web | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
bsc_web | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
bsc_web | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
bsc_web | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
bsc_web | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
bsc_web | /docker-entrypoint.sh: Configuration complete; ready for start up
```

Для входа в интерфейс BSC нужно из любого браузера перейти по адресу <http://localhost:8000>. Видим перед собой окно с авторизацией в BSC. Данные для входа: 1. Логин: admin 2. Пароль: admin



9. Также вы можете зарегистрироваться в качестве пользователя
1. В окне входа выбираем пункт **Sign Up**
 2. Проходим регистрацию
 3. Входим с пользовательскими логином и паролем

3 Лицензирование

Представленная версия для работы не требует дополнительной активации, является полнофункциональной.

4. Поддержка

Компания Bastion осуществляет поддержку в рамках договора, срок поддержки устанавливается индивидуально с каждым заказчиком.

5.Контакты

По вопросам поддержки обращайтесь:

- На официальный сайт: bastion-tech.ru
- На электронную почту: support@bastion-tech.ru