Національний технічний університет України
«Київський політехнічний інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Розрахунково-графічна робота

по курсу

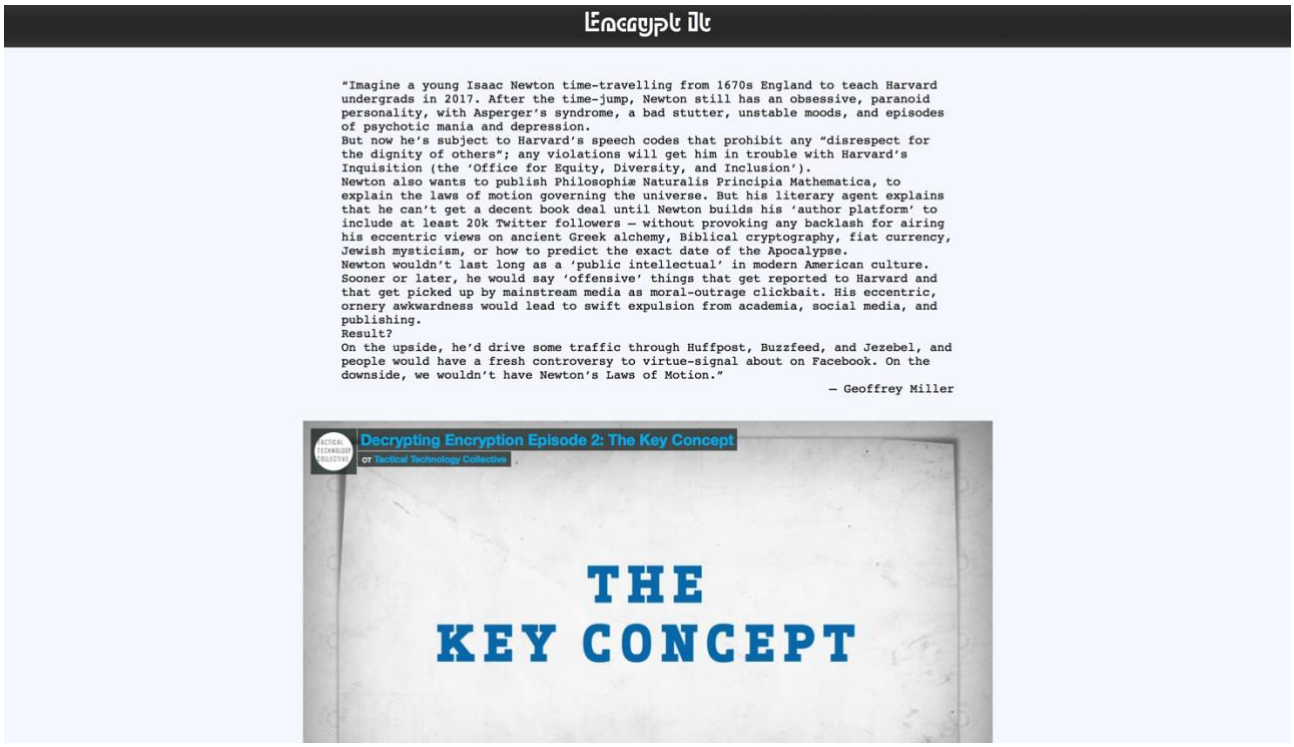
«Інтеграційні програмні системи»

Виконали: студенти 4курсу
ФІОТ гр. ІО-42 та ІО-44
Команда !23.1415 в складі:
Буйницький В.І.
Данішевский А.О.
Євдокимов Д.В.
Салата Я.В.

Київ 2017

1. Опис проекту

Розроблений нами проект дозволяє користувачам виконувати шифрування та дешифрування методами: AES, Blowfish, Twofish, IDEA, MD5, SHA 1, HMAC, RSA Security: RC4.



Скріншот головної сторінки

Серверна частина сервісу розроблена за допомогою засобів IIS Express. База буде знаходиться на віддаленому сервері.

2. Інструмент для контейнеризації. Docker

Docker - інструментарій для управління ізольованими Linux-контейнерами. Docker доповнює інструментарій LXC більш високорівневим API, що дозволяє керувати контейнерами на рівні ізоляції окремих процесів. Зокрема, Docker дозволяє не переймаючись вмістом контейнера запускати довільні процеси в режимі ізоляції і потім переносити і клонувати сформовані для даних процесів контейнери на інші сервери, беручи на себе всю роботу зі створення, обслуговування і підтримки контейнерів.

Сирцевий код Docker написаний мовою Go і поширюється під ліцензією Apache 2.0. Інструментарій базується на застосуванні вбудованих в ядро Linux штатних механізмів ізоляції на основі просторів імен (namespaces) і груп управління (cgroups). Для створення контейнерів використовуються скрипти lxc. Для формування контейнера досить завантажити базовий образ оточення (команда `docker pull base`), після чого можна запускати в ізольованих оточеннях довільні програми (наприклад, для запуску bash можна виконати docker run -i -t base/bin/bash).

Основні можливості Docker:

- Можливість розміщення в ізольованому оточенні різнорідної начинки, що включає різні комбінації виконуваних файлів, бібліотек, файлів конфігурації, скриптів, файлів jar, gem, tar тощо
- Підтримка роботи на будь-якому комп'ютері на базі архітектури x86_64 з системою на базі ядра Linux, починаючи від ноутбуків, закінчуючи серверами та віртуальними машинами. Можливість роботи поверх немодифікованих сучасних ядер Linux (без

накладення патчів) і в штатних оточеннях всіх великих дистрибутивів Linux, включаючи Fedora, RHEL, Ubuntu, Debian, SUSE, Gentoo і Arch;

- Використання легковагих контейнерів для ізоляції процесів від інших процесів і основної системи.
- Оскільки контейнери використовують свою власну самодостатню файлову систему, не важливо де, коли і в якому оточенні вони запускаються.
- Ізоляція на рівні файлової системи: кожен процес виконується у повністю окремій кореневій ФС;
- Ізоляція ресурсів: споживання системних ресурсів, таких як витрата пам'яті і навантаження на CPU, можуть обмежуватися окремо для кожного контейнера з використанням cgroups;
- Ізоляція на рівні мережі: кожен ізольований процес має доступ тільки до пов'язаного з контейнером мережевого простору імен, включаючи віртуальний мережевий інтерфейс і прив'язані до нього IP-адреси;
- Коренева файлова система для контейнерів створюється з використанням механізму copy-on-write (окремо зберігаються тільки змінені і нові дані), що дозволяє прискорити розгортання, знижує витрату пам'яті і економить дисковий простір;
- Всі стандартні потоки (stdout/stderr) кожного виконуваного в контейнері процесу накопичуються і зберігаються у вигляді логу;
- Змінена файлова система одного контейнера може використовуватися як основа для формування нових базових образів і створення інших контейнерів, без необхідності оформлення шаблонів або ручного налаштування складу образів;
- Можливість використання інтерактивної командної оболонки: до стандартного вводу будь-якого контейнера може бути прив'язаний псевдо-tty для запуску shell.
- Підтримка використання різних систем зберігання, які можуть підключатися як плагіни. Серед підтримуваних драйверів зберігання заявлені aufs, device mapper (використовуються снапшоти LVM), vfs (на основі копіювання директорій) і Btrfs. Очікується поява драйверів для ZFS, Gluster і Ceph;
- Можливість створення контейнерів, що містять складні програмні стеки, через зв'язування між собою вже існуючих контейнерів, що містять складові частини формованого стека. Зв'язування здійснюється не через злиття вмісту, а через забезпечення взаємодії між контейнерами (створюється мережевий тунель).

3. Сервер безперервної інтеграції. Travis-ci

Як виявилося, термін «continuous integration» досить старий. Він був введений Мартіном Фаулером (Martin Fowler) у 2000-му році і викладений у статті «Continuous Integration» і по-російськи звучить як «безперервна інтеграція». Це частина процесу розробки, в якій розробляється проект збирається / тестується в різних середовищах виконання автоматично і безперервно. Задумувалася дана методика для найбільш швидкого виявлення помилок / протиріч інтеграції проекту, а отже зниження витрат на наступні простої.

Принцип досить простий: на окремій машині працює якась служба, в обов'язки якої входить отримання вихідного коду проекту, його збірка, тестування, логування, а також можливість надати для аналізу дані виконання перерахованих операцій.

Звичайно ж, безкоштовний сир буває тільки в мишоловці і за зручність необхідно платити: виділити окремий сервер і підтримувати його в робочому стані, забезпечити наявність необхідних програмних комплексів, налаштувати середовища виконання, робити резервні копії даних і т.д. Все це вимагає чимало часу і ресурсів. І цілком логічним здається можливість делегувати цю відповідальність на сторонні сервіси. От якраз таким і є travis-ci - «хостинг безперервної інтеграції для open source співтовариства». Настав час подивитися на нього ближче.

Travis-ci підтримує безліч мов програмування серед яких є і ruby (що не дивно, тому що спочатку він розроблявся для ruby-проектів). Почати користуватися сервісом дуже просто. Потрібно всього лише зробити кілька кроків, які детально описані у власному Гайд проекту. Я лише опишу процес в цілому.

Короткий перелік багів, які шукає утиліта:

- Проблеми продуктивності, пов'язані з розміткою інтерфейсу
- Невикористані ресурси
- Невідповідності розмірів масивів (коли масиви визначені у множинних конфігураціях).
- Проблеми доступності та інтернаціоналізації («магічні» рядки, відсутність атрибуту contentDescription і т.д)
- Проблеми з іконками (невідповідності розмірів, порушення DRY)
- Проблеми зручності використання (Наприклад, не зазначений спосіб введення для текстового поля)
- Помилки в маніфесті