

Эксплуатация приложения

Михаил Дьяченко

Эксплуатация приложения

- Использование чужого кода
- Совместное использование кода
- Доставка приложения на сервер и пользователям
- Автоматизация

Зависимости



npm

```
$ npm init
```

```
$ npm install express
```

```
$ npm install --save express
```

```
$ npm install --save-dev mocha
```

```
$ npm install express@1.0.0
```

```
$ npm uninstall express
```

```
$ npm list
```

Semantic Versions

3.8.0-alpha.1

- **major** – новые возможности без сохранения обратной совместимости
- **minor** – новые возможности с сохранением обратной совместимости
- **patch** – исправления ошибок, рефакторинг
- **pre-release** – версия в разработке

Фиксируйте версии

package.json

```
{  
  "name": "average",  
  "version": "1.0.0",  
  "description": "Calculate average number",  
  "license": "MIT",  
  "dependencies": {  
    "lodash": "4.17.4"  
  },  
  "devDependencies": {  
    "eslint": "4.0.0"  
  }  
}
```


Advanced Range Syntax

"express": "1.2.3",

"express": ">1.2.3",

"express": ">=1.2.3",

"express": "~1.2.3", // >=1.2.3 <1.3.0

"express": "^1.2.3", // >=1.2.3 <2.0.0

"express": "latest",

"express": "alpha",

"express": "git://github.com/expressjs/express.git",

"express": "git://github.com/expressjs/express.git#4.13.4",

"express": "git://github.com/expressjs/express.git#master",

"express": "git://github.com/expressjs/express.git#f3d99a4",

"express": "expressjs/express#f3d99a4"

Обновляйте устаревшие

npm outdated

```
$ npm outdated
```

Package	Current	Wanted	Latest
mocha	2.3.3	2.4.0	2.4.5
nodemon	1.8.0	1.8.0	1.9.1
lodash	3.10.1	3.10.1	4.6.1
supertest	1.1.0	1.1.0	1.2.0

```
$ npm update
```

Находите уязвимости

npm audit

```
$ npm audit
```

found 10 vulnerabilities (7 moderate, 3 high)
in 2144 scanned packages

5 vulnerabilities require manual review

npm audit

```
$ npm audit
```

Low	Regular Expression Denial of Service
Package	<code>braces</code>
Dependency of	<code>ava [dev]</code>
Path	<code>ava > chokidar > anymatch > micromatch > braces</code>
More info	https://npmjs.com/advisories/786

npm audit

```
$ npm audit fix
```

package-lock.json

```
# Устанавливает зависимости по данным package.json  
# Создает package-lock.json по данным из package.json  
$ npm install
```

```
# Добавляет зависимость необходимую для приложения  
$ npm install express
```

```
# Добавляет зависимость необходимую для разработки  
$ npm install --save-dev pify
```


package-lock.json

```
"pify": {  
  "version": "2.3.0",  
  "resolved": "https://registry.npmjs.org/pify/-/pify-2.3.0.tgz",  
  "integrity": "sha1-7RQaasBDqEnqWISY59yosVMw6Qw=",  
  "dev": true  
}
```

```
# Устанавливает зависимости строго по данным package-lock.json  
$ npm ci
```

Codestyle

```
let foo = 1,  
    bar = 2,  
    baz = 3;    VS    let foo = 1;  
                  let bar = 2;  
                  let baz = 3;
```

CES

VS

TABS



Использование let и const

```
const pi = 3.141592653589;
```

```
const e = 2.71828182;
```

```
const φ = 1.618033988;
```

```
for(let i = 0; i < 10; i++) {
```

```
    console.log(i);
```

```
}
```

Вложенность блоков

```
if (user.isAuthenticated) {  
  if (notes.length > 0) {  
    for(let i = 0; i < note.length; i++) {  
      console.log(notes[i]);  
    }  
  } else {  
    console.log('Notes not found!')  
  }  
}
```

Цикломатическая сложность

```
function renderNotes(res, user, notes) {  
  if (!user.isAuthenticated) {  
    res.sendStatus(403);  
  } else if (notes) {  
    res.render('notes', notes);  
  } else {  
    res.sendStatus(404);  
  }  
}
```

Цикломатическая сложность – количество
независимых путей

Цикломатическая сложность

```
function renderNotes(res, user, notes) {  
  if (!user.isAuthenticated) {  
    res.sendStatus(403);  
  } else if (notes) {  
    res.render('notes', notes);  
  } else {  
    res.sendStatus(404);  
  }  
}
```

3

Бесполезное code review

```
exports.list = function (req, res) { // Используй стрелочные
  let notes = Note.findAll(); // Используй const
  let data = { // И здесь
    notes: notes,
    meta: req['meta'] // Здесь можно так: req.meta
  };

  res.render('notes', data);
};
```

Codestyle **помогает**

Упростить чтение кода всей команды

Избежать конфликтов

Сделать ревью кода полезнее

Избежать типичных ошибок в коде

Сделать код качественнее

Codestyle.md

Variable declaration

- * Each variable should be declared:
 - * using a var statement;
 - * only once in the current scope;
 - * on a new line;
 - * as close as possible to the place where it's first used.
- * Each var statement should have only one variable declared in it.

.editorconfig



.editorconfig

```
[*]  
indent_size = 4  
indent_style = space
```

```
[*.json]  
indent_size = 2
```

editorconfig.org

ESLint

```
$ npm install --save-dev eslint
```

.eslintrc.json

```
{  
  "rules": {  
    "no-unused-vars": 2,  
    "max-len": [1, 100],  
    "max-params": [2, 3]  
  }  
}
```

```
0 - off  
1 - warning  
2 - error
```

eslint.org/docs/rules

.eslintrc.json

Готовые наборы правил

```
$ npm install --save-dev eslint-config-xo
```

```
{  
  "extends": "xo",  
  "rules": {  
    "max-len": [2, 100],  
    "max-params": [2, 3]  
  }  
}
```

npms.io/search?q=eslint-config

.eslintignore

```
/build
```

```
**/*.min.js
```

```
/node_modules
```

Игнорирование правил в коде

```
function onError(err, req, res, next) {  
  /* eslint no-unused-vars: 0 */  
  /* eslint max-params: [2, 4] */  
  
  res.sendStatus(500);  
}
```

Проверка кода

```
$ node_modules/.bin/eslint .
```

```
/Workspace/urfu-2017/notes-app-example/index.js
```

```
  3:29  error  Missing semicolon          semi
```

```
 20:34  error  Missing semicolon          semi
```

Автоматическое исправление

```
$ node_modules/.bin/eslint . --fix
```

ESLint ♥ VSCode

```
38
39 // Подключаем шаблонизатор [eslint] Missing semicolon. (semi)
40 app.set('view engine', 'hbs')
41
42 // Подключаем директорию с шаблонами
43 app.set('views', viewsDir);
44
45 // Логируем запросы к приложению в debug-режиме
46 if (config.get('debug')) {
47     app.use(morgan('dev'));
48 }
49
```

Stylelint

```
$ npm install --save-dev stylelint
```


.stylelintrc.json

```
{  
  "extends": "stylelint-config-standard",  
  "rules": {  
    "color-hex-case": "lower"  
  },  
  "ignoreFiles": [  
    "build/*"  
  ]  
}
```

stylelint.io/user-guide/rules/

TSLint

~~TSLint~~

ESLint + TypeScript

```
$ npm install --save-dev @typescript-eslint/parser
```

```
$ npm install --save-dev @typescript-eslint/eslint-plugin
```

```
{  
  "extends": "xo",  
  "parser": "@typescript-eslint/parser",  
  "plugins": ["@typescript-eslint"],  
  "rules": {  
    "max-len": [2, 100],  
    "@typescript-eslint/no-unused-vars": 2,  
  }  
}
```

Скрипты

Запуск комплексных команд в виде простых
запоминающихся алиасов к ним

Makefile

start:

```
node index.js
```

lint:

```
node_modules/.bin/stylelint public/*.css  
node_modules/.bin/eslint *.js
```

test:

```
node_modules/.bin/mocha test/
```

```
$ make test
```

Building Web Software With Make

Mark McDonnell

Gulp

```
const gulp = require('gulp');
const eslint = require('gulp-eslint');

gulp.task('lint', () => {
  gulp
    .src('*.js')
    .pipe(eslint())
});
```

```
$ gulp lint
```

Building With Gulp
Callum Macrae

NPM Scripts

```
{  
  "name": "awesome-notes",  
  "dependencies": {  
    "mocha": "4.0.0"  
  },  
  "scripts": {  
    "test": "node_modules/.bin/mocha test/",  
  }  
}
```

```
$ npm run test
```

```
$ npm test
```

NPM Scripts

```
{  
  "name": "awesome-notes",  
  "dependencies": {  
    "mocha": "4.0.0"  
  },  
  "scripts": {  
    "test": "mocha test/",  
  }  
}
```

```
$ npm run test
```

```
$ npm test
```

Команды

```
{  
  "scripts": {  
    "clean": "rm -rf node_modules/"  
  }  
}
```

Помним о мультиплатформе!

```
{  
  "devDependencies": {  
    "rimraf": "2.5.2"  
  },  
  "scripts": {  
    "clean": "rimraf node_modules/"  
  }  
}
```

Последовательные команд

```
{  
  "scripts": {  
    "check:lint": "eslint .",  
    "check:test": "mocha test/",  
    "check": "npm run check:lint && npm run check:test"  
  }  
}
```

Последовательные команд

```
{  
  "devDependencies": {  
    "npm-run-all": "1.5.2"  
  },  
  "scripts": {  
    "check:lint": "eslint .",  
    "check:test": "mocha test/",  
    "check": "npm-run-all check:lint check:test"  
  }  
}
```

Параллельные команды

```
{  
  "scripts": {  
    "lint:css": "stylelint **/*.css",  
    "lint:js": "eslint .",  
    "lint": "npm run lint:css & npm run lint:js"  
  }  
}
```

Параллельные команды

```
{  
  "scripts": {  
    "lint:css": "stylelint **/*.css",  
    "lint:js": "eslint .",  
    "lint": "npm-run-all --parallel lint:css lint:js"  
  }  
}
```


Группы команд

```
{  
  "scripts": {  
    "lint:css": "stylelint **/*.css",  
    "lint:js": "eslint .",  
    "lint": "npm-run-all --parallel lint:*"  
  }  
}
```

Настройки

```
{  
  "name": "app",  
  "config": {  
    "report": "nyan"  
  },  
  "scripts": {  
    "test": "mocha test/ -R $npm_package_config_report"  
  }  
}
```

```
$ npm test --app:report=markdown
```

Внешние скрипты

```
// scripts/favicon.js
const { readFileSync, writeFileSync } = require('fs');
const toIco = require('to-ico');

const logo = readFileSync('logo.png');

toIco(logo).then(data => writeFileSync('favicon.ico', data));

{
  "scripts": {
    "favicon": "node scripts/favicon.js"
  }
}
```

Аргументы

```
{  
  "scripts": {  
    "dev": "node app/index.js",  
  }  
}
```

```
$ npm run dev -- debug
```

```
$ node app/index.js debug
```

Hooks

```
"scripts": {  
  "prepublish": "npm run test"  
}
```

Hooks

```
"scripts": {  
  "precommit": "npm run test",  
  "prepush": "npm run test",  
  "prepublish": "npm run test"  
}
```

```
$ npm install --save-dev husky
```

Advanced front-end automation with npm

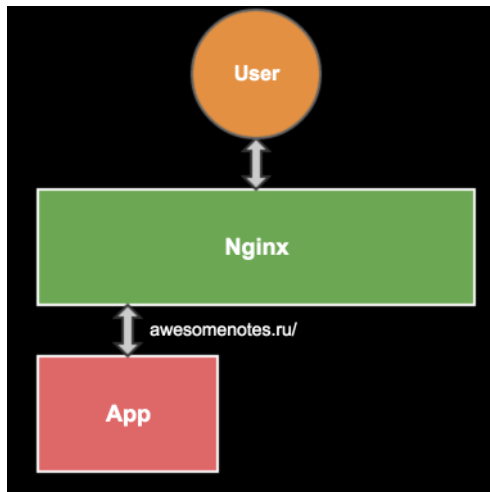
Kate Hudson

How to Use npm as a Build Tool

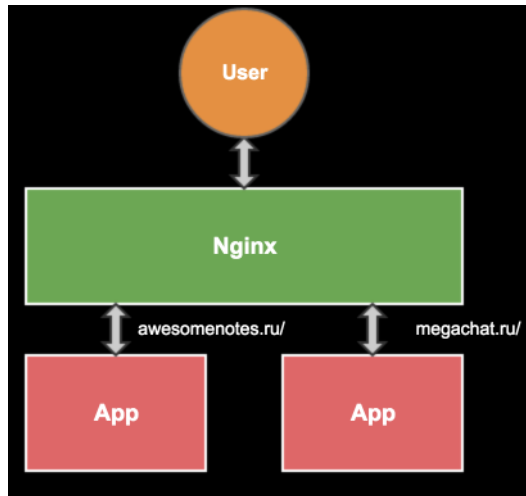
Keith Cirkel

Перерыв

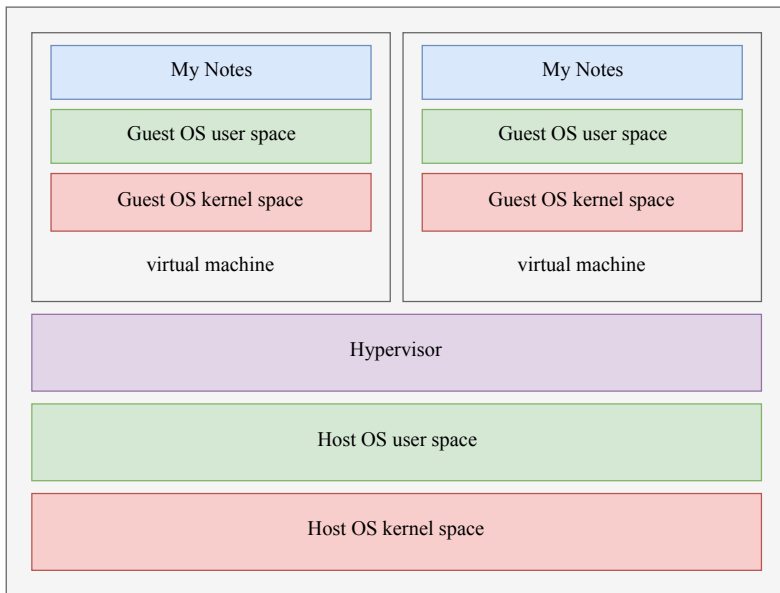
Развёртывание сервиса



Share machine



Virtualization



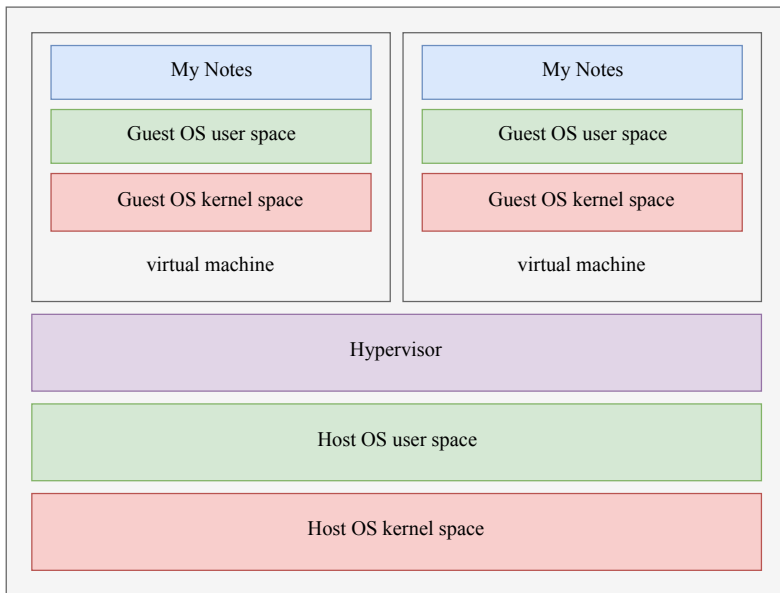
Полная изоляция

Разделение ресурсов (cpu, mem, disk)

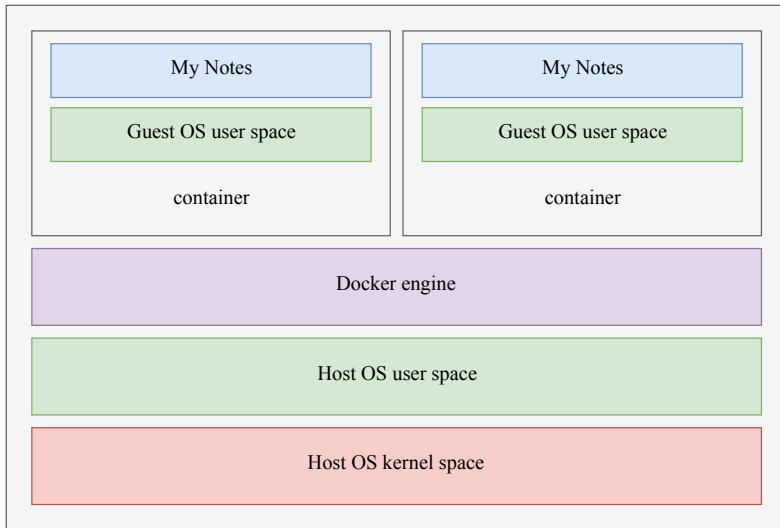
Тяжёлые на подъём

Требовательны к дисковому пространству

Virtualization



Containers



Лёгкие на подъём
Экономят дисковое пространство

Изоляция уязвима

Привязаны к ядру одной OS



Docker Hub – хранилище контейнеров

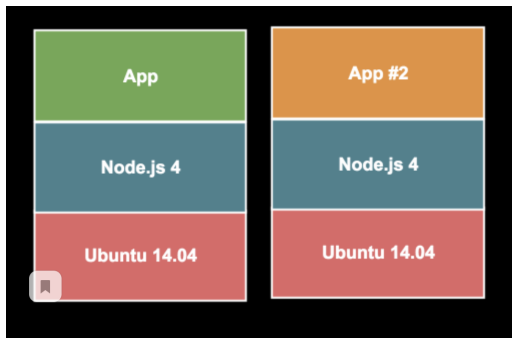
Docker Tools – утилиты для сборки,
публикации и запуска

Docker Images – read-only образы

Docker Hub

Ubuntu, Node.js, MySQL, Mongo

Docker images



Из образов создаются экземпляры
контейнеров

Dockerfile

```
app/  
└─ index.js
```

```
package-lock.json
```

```
Dockerfile
```

```
# Базовый слой
```

```
FROM node:10
```

```
# Копируем всё что нужно из локальной папки в образ
```

```
COPY app /app
```

```
COPY package.json /
```

```
COPY package-lock.json /
```

```
# Устанавливаем зависимости, в образе появится /node_modules
```

```
RUN npm ci --production
```

```
# При старте контейнер начнёт общаться через 80 порт
```

```
EXPOSE 80
```

```
# При старте контейнер выполнит эту команду - запустит наше приложение
```

```
CMD node app/index.js
```

Порт

```
app/  
└─ index.js
```

```
package-lock.json
```

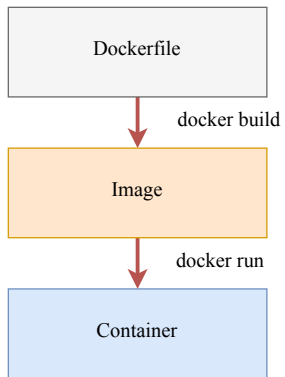
```
import express from 'express';
```

```
const app = express();
```

```
// Указываем порт, через который будем общаться с внешним миром  
app.listen(80);
```


Устанавливаем Docker

Docker



docker build

```
$ docker build --tag my-app .
```

```
Sending build context to Docker daemon 157.8MB
```

```
Step 1/9 : FROM node:10
```

```
Pulling from library/node
```

```
Step 2/9 : COPY app /app
```

```
---> 7acbc4cf9eb3
```

```
...
```

```
...
```

```
...
```

```
Successfully built 90f0c5cc4655
```

```
Successfully tagged my-app:latest
```

docker run

```
$ docker run --publish 8080:80 my-app
```

```
$ docker run --publish 8080:80 90f0c5cc4655
```

`--publish` привязывает порт доступный для внешнего мира, к порту, через который общается контейнер

Теперь можем обратиться к приложению в контейнере по ссылке <http://localhost:8080/>

docker stop

```
$ docker ps
```

CONTAINER ID	IMAGE	...
db1312de4ce1	6e68b41059d8	...

```
$ docker stop db1312de4ce1
```



Облачная PaaS-платформа

Устанавливаем Heroku


```
$ heroku login
```

```
$ heroku help
```

Разворачиваем приложение

```
$ heroku container:login
```

```
$ heroku create my-app
```

```
Creating app... done,  my-app  
https://my-app.herokuapp.com/
```


Разворачиваем приложение

```
$ heroku container:push web
```

```
=== Building web (/app/Dockerfile)
```

```
> Step 1/9 : FROM node:10
```

```
---> 8c10e6cc3f51
```

```
...
```

Your image has been successfully pushed.

You can now release it with the '`container:release`' command.

Разворачиваем приложение

```
$ heroku container:release web
```

```
$ heroku open
```

Полезные команды

```
$ heroku logs
```

```
2019-03-19T11:42:31.796667+00:00 app[api]: Release v2 ...  
2019-03-19T11:42:31.796667+00:00 app[api]: Enable Logpl...
```

```
$ heroku restart
```

```
$ heroku releases
```

```
v6  Deployed web (28f789388337) ... (~ 47m ago)  
v5  Deployed web (ed9e5f3a25b5) ... (~ 49m ago)
```

A Beginner-Friendly Introduction to Containers,
VMs and Docker

Preethi Kasireddy

Architecting Containers

Scott McCarty

Docker Get Started

Dockerizing a Node.js web app

Container Registry & Runtime (Docker Deploys)

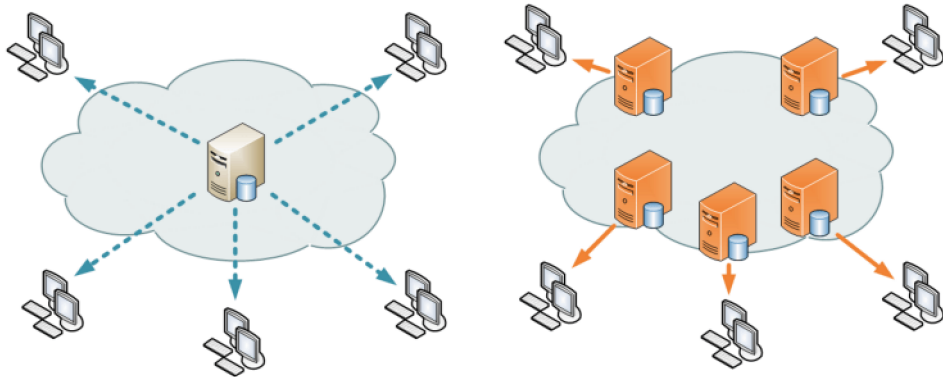
Размещение статистики

Изображения, иконки, таблицы стилей

Не требуют вычислительных ресурсов

Основная задача для статистики – разместить
ближе к пользователю

Content Delivery Network

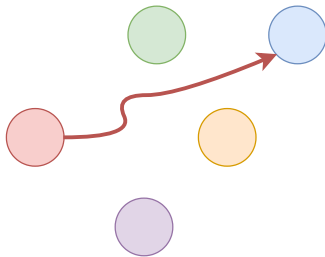


Content Delivery Network

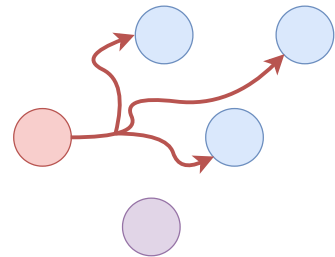
Unicast

Anycast

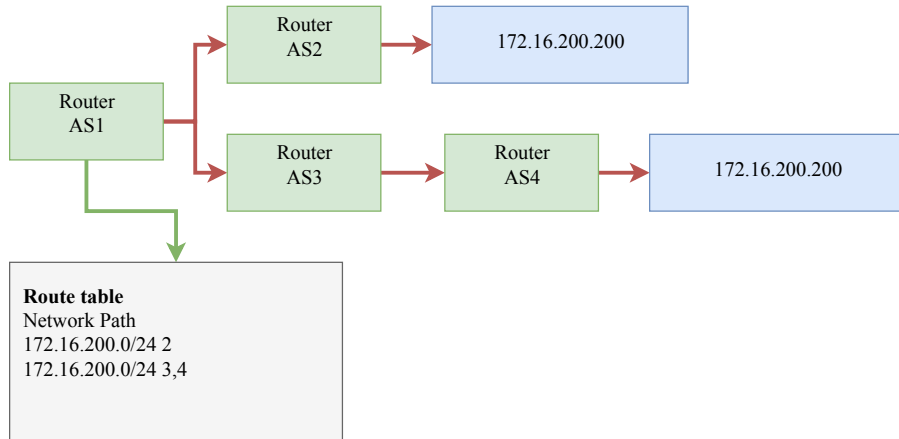
One Machine, One IP



Many Machines, One IP



Border Gateway Protocol



Кеширование

Сжатие текстовой статики

Обеспечение 100% доступности

Количество точек присутствия
Point of Presence

Политика кеширования

Политика устаревания

Surge



```
app/  
└─ index.js  
└─ routes.js  
└─ models  
└─ controllers  
└─ views  
└─ public  
    └─ styles.css  
    └─ favicon.ico
```

Surge

```
$ npm install surge
```

```
$ surge -p ./public -d my-app.surge.sh
```

```
email: email@example.com
token: *****
project path: ./app/public
size: 3 files, 19.2 KB
domain: my-app.surge.sh
upload: [=====] 100%, eta: 0.0s
propagate on CDN: [=====] 100%
plan: Free
users: email@example.com
```


Surge

```
<head>  
  <link rel="stylesheet"  
  href="/styles.css">  
  <link rel="stylesheet"  
    href="https://my-app.surge.sh/styles.css">  
</head>
```

Как работает кеширование

HTTP/1.1 200 OK

Cache-Control: public, max-age=31536000, no-cache

Content-Type: text/css; charset=UTF-8

ETag: d1d3c5c4cdb2568785ba1a366b7fb048

Server: SurgeCDN/0.12.2

```
body {  
    font-family: Arial, sans-serif;  
}
```

Как работает кеширование

GET /styles.css HTTP/1.1

Host: notes-app-operating.surge.sh

If-None-Match: d1d3c5c4cdb2568785ba1a366b7fb048

Если ETag равен If-None-Match,
то ответ от Surge будет очень коротким

HTTP/1.1 304 Not Modified

Знакомство с Content Delivery Network

Webzilla

What is Anycast and How it works

Sarath Pillai

Content Delivery Networks

Rajkumar Buyya

Continuous Integration

Continuous Integration

Code Repository & Version Control

Developer pushes code and automatically triggers server build

Build & Integration Automation

CI server start the build and run test

Continuous Delivery

Release Automation

Store any artifact outcome from the build

Delivery Automation

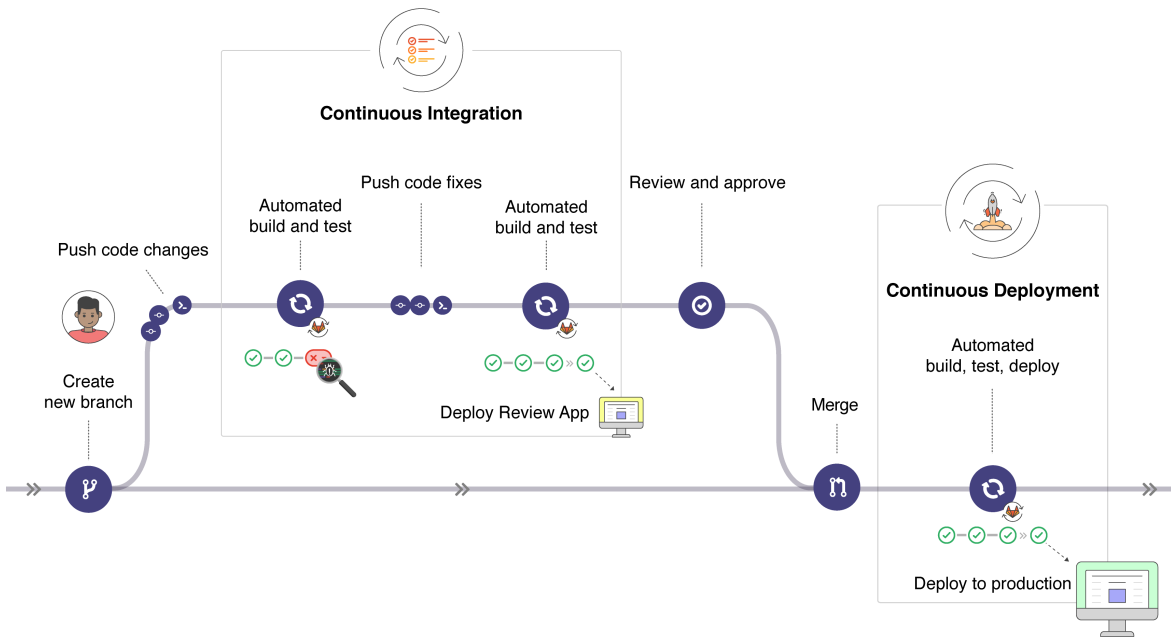
Deploy binaries to any given environment

Continuous Deployment

Production Automation

Promote deployment to production (on-prem, public cloud, hybrid cloud)

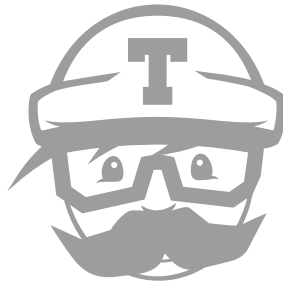
Автоматизация проверки кода и
развёртывания сервиса по факту
изменения кода



Автоматизация рутины

Неизбежное тестирование кода

Быстрая доставка до конечного
пользователя



travis-ci



circle-ci

gitlab ci

Создаем файл .gitlab-ci.yml

```
image: node:10.15.3
```

```
stages:
```

- test

```
test_unit:
```

```
  stage: test
```

```
  script:
```

- npm install
- npm run test

```
only:
```

- pushes

Теперь на каждое обновление репозитория, например, **git push**, CI будет выполнять перечисленные команды

gitlab ci pipeline

The screenshot displays the GitLab CI/CD interface. On the left is a sidebar with navigation options: Project overview, Repository, Issues (0), Merge Requests (0), CI / CD (selected), Pipelines, Jobs, Schedules, and Charts. The main content area shows a pipeline summary with the following details:

- Status: All 1, Pending 0, Running 1, Finished 0
- Buttons: Run Pipeline, Clear Runner Caches, CI Lint
- Table headers: Status, Pipeline, Triggerer, Commit, Stages
- Table row:
 - Status: running
 - Pipeline: #118072160 latest
 - Triggerer: Michael Dyachenko's avatar
 - Commit: master ~ 1f7058fc first
 - Stages:
 - Action: ✕

Для того, чтобы CI мог разворачивать приложение в Heroku и публиковать статику в Surge, нам нужно поделиться нашими секретами

Заходим на страницу настроек CI проекта

The screenshot shows the GitLab interface for a project named 'heroku-gitlab'. The left sidebar contains navigation options: Project (Details, Activity, Cycle Analytics), Issues (0), Merge Requests (0), CI / CD, Operations, Packages, Wiki, Snippets, and Settings (highlighted with a red box). A red arrow points from the 'Settings' box to the 'CI / CD' option in the 'Repository' section of the main content area. The main content area shows the project details, including the name 'heroku-gitlab' and Project ID '13845063'. Below this, there is a section titled 'The repository for this project is empty' with instructions on how to create files directly in GitLab using options like 'Add README', 'Add CHANGELOG', and 'Add CONTRIBUTING'. At the bottom, there are instructions for setting up a new repository, including global setup and cloning the repository.

GitLab Projects Groups Activity Milestones Snippets

H heroku-gitlab

Project

- Details
- Activity
- Cycle Analytics

Issues 0

Merge Requests 0

CI / CD

Operations

Packages

Wiki

Snippets

Settings

Sam Barros > heroku-gitlab > Details

H heroku-gitlab Project ID: 13845063

Add license

How do I deploy my code to Heroku using GitLab CI/CD?

The repository for this project is empty

You can create files directly in GitLab using one of the following options.

New file Add README Add CHANGELOG Add CONTRIBUTING

Command line instructions

can also upload existing files from your computer using the instructions below.

global setup

```
git config --global user.name "Sam Barros"
git config --global user.email "samarony.barros@gmail.com"
```

Setup a new repository

```
git clone git@gitlab.com:samaronybarros/heroku-gitlab.git
cd heroku-gitlab
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

Добавляем секреты для Heroku и Surge

The screenshot shows the GitLab CI/CD configuration page for a project named 'heroku-gitlab'. The page is divided into several sections: General pipelines, Auto DevOps, Runners, and Variables. The Variables section is highlighted with a red box and contains a table of environment variables.

Variables

Environment variables are applied to environments via the runner. They can be protected by only exposing them to protected branches or tags. Additionally, they can be masked so they are hidden in job logs, though they must match certain regex requirements to do so. You can use environment variables for passwords, secret keys, or whatever you want. You may also add variables that are made available to the running application by prepending the variable key with `K8S_SECRET_`. [More information](#)

Type	Key	Value	State	Masked	Scope
Variable	HEROKU_API_KEY	*****	Protected <input checked="" type="checkbox"/>	Masked <input checked="" type="checkbox"/>	All environments
Variable	HEROKU_APP_PI	*****	Protected <input checked="" type="checkbox"/>	Masked <input checked="" type="checkbox"/>	All environments
Variable	HEROKU_APP_S	*****	Protected <input checked="" type="checkbox"/>	Masked <input checked="" type="checkbox"/>	All environments
Variable	Input variable ke	Input variable	Protected <input checked="" type="checkbox"/>	Masked <input checked="" type="checkbox"/>	All environments

Buttons: Save variables, Reveal values

Gitlab CI

Вопросы?