Lecture 11: DevOps

COMP3006: Full-Stack Development

David Walker

School of Engineering, Computing and Mathematics
david.walker@plymouth.ac.uk

Semester 1, 2020-21

Introduction

Today's topics

1. DevOps
2. Analytics

Session learning outcomes

By the end of today's lecture you will:

- Describe the usage and benefits of continuous integration and continuous deployment
- Analyse the best way to release new application features to your user-base

Outline

Different levels of continuous

Continuous integration

- Automatically building and testing your software on a regular basis
- Daily builds, build on every commit

Continuous delivery

- Trust your software due to continuous integration
- Release new version after every commit
- May be more than customers want
  - Always release-ready code

Continuous deployment

- Update your application silently in the background
  - Cloud-based SaaS
- Provide automatic updates
  - Mobile, desktop

Wider benefits

Customers

- Get software updates faster
- Documented process informs users of changes and allows them to influence the development process

Management

- Progress is immediately visible

Developers

- Removes the 'release window' concepts, which minimises delays

System administrators

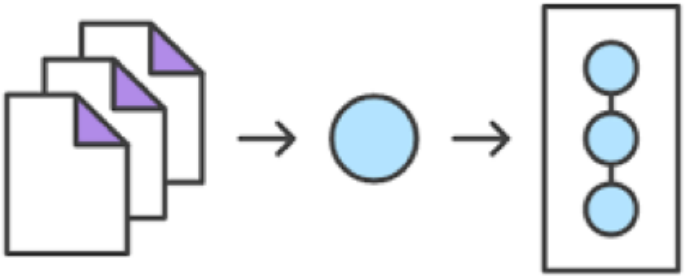
- Freed from deployment work
- Can focus on deployment analysis

System administrators

- Maintain live system performance
- Analyse performance after each deployment
- Simplify analysis and maintenance
  - Don't release entangled features
  - Use atomic releases
  - Single feature roll back

Atomic commits

- Only commit changes related to a single feature
- Collect related changes (files) on a staging area
  - **git add**
- Commit only the changes on the staging area
  - **git commit**



Travis

- An automation server
- Watches your Git
- Takes actions on commit
  - Analyse code
  - Compile
  - Run tests
  - Run scenarios
  - Deploy code to live server
- Other automation servers are available

Setup

- Connect your GitHub account to Travis – let Travis access your repos
- Add `.travis.yml` to your repo
- Whenever you `git push` Travis will automatically build your project and test it automatically

A Travis Example

djw213 / HelloCI build passing

Current Branches Build History Pull Requests More options

main Added secure key for Heroku.

Commit 194127b
Compare 78563d0..194127b
Branch main

David Walker

#13 passed
Ran for 49 sec
2 hours ago

Restart build

Node.js: node
AMD64

Job log View config

Remove log Raw log

```
1                                                                                  0.07s
                                                                                   1.5s
                                                docker_mtu_and_registry_mirrors
                                                                      resolvconf
2  $ git clone --depth=50 --branch=main https://github.com/djw213/HelloCI.git djw213/HelloCI   git.checkout   0.47s
6
7                                                                                  0.01s
8  $ nvm install node                                                  nvm.install   4.24s
14
15 Setting up build cache                                              cache.1
21
22                                                                     cache.npm
25 $ node --version
26 v15.3.0
                                                                                   Top
```

.travis.yml

Add information about your project and its dependencies

```
language: node_js
node_js:
- node
install:
- npm install -g mocha
- npm install chai
```

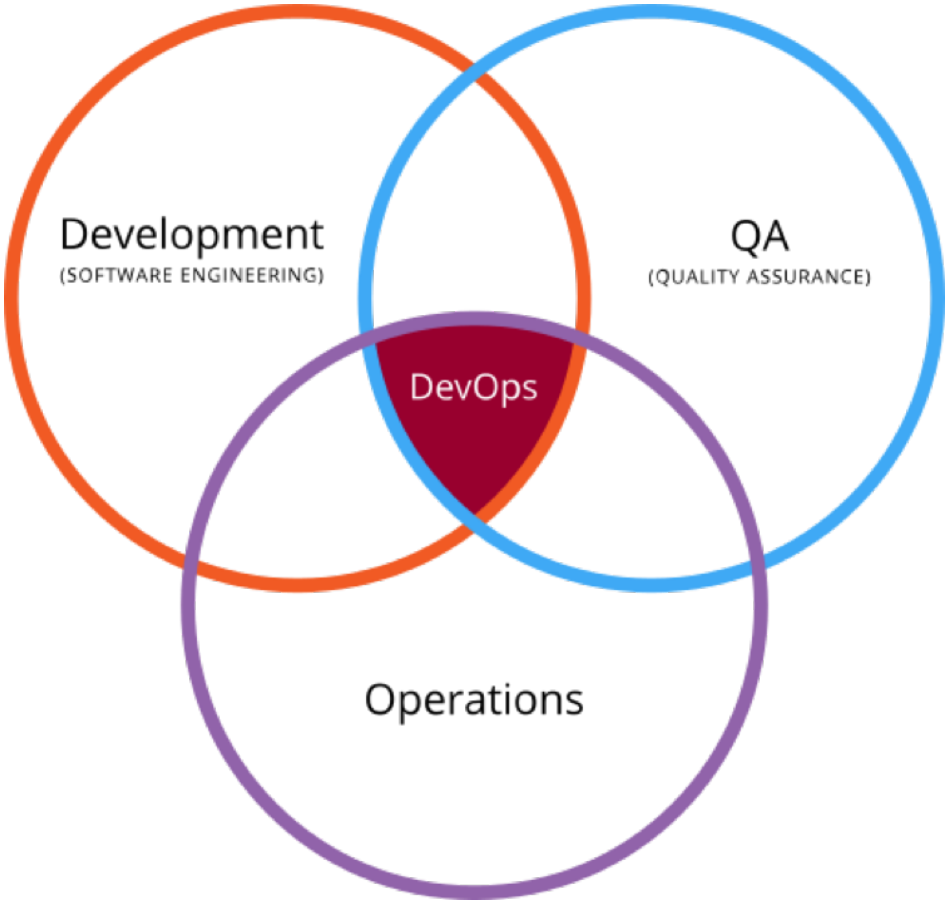
- What language have you used? Specify the version (**node** specifies the latest version of Node)
- Define any prerequisites – include command to install them

Define test script in package.json

```
  "scripts": {
    "test": "mocha -ui tdd test/",
    "start": "node server.js"
  },
```

DevOps

- Development and Operations
- Collaboration and communication of both software developers and other IT professionals



Feature flagging

- Manipulating the availability of features to different users post-deployment
- e.g., a new navigation structure
  - Can be released to just 10% of users (to limit potentially poor experience)
  - Can be used to collect usage data
  - Well-performing features can be rolled out 100%
  - Poorly-performing features can be killed
- Allow comparative A/B tests
- Give users an opt-in choice

Outline

Toggle types

Release toggles

- Transitory toggles for careful deployment

Experiment toggles

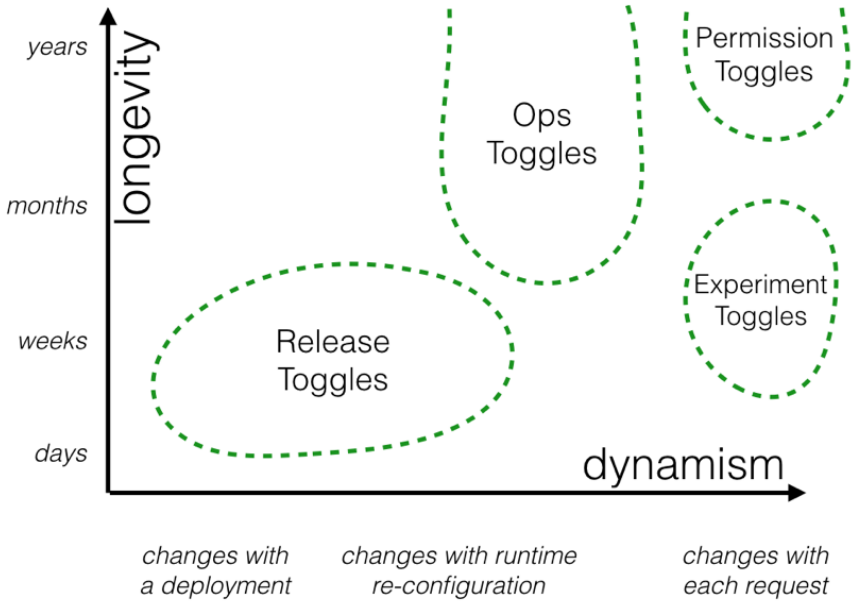
- Short-lived toggles for comparing performance of different features

Ops toggles

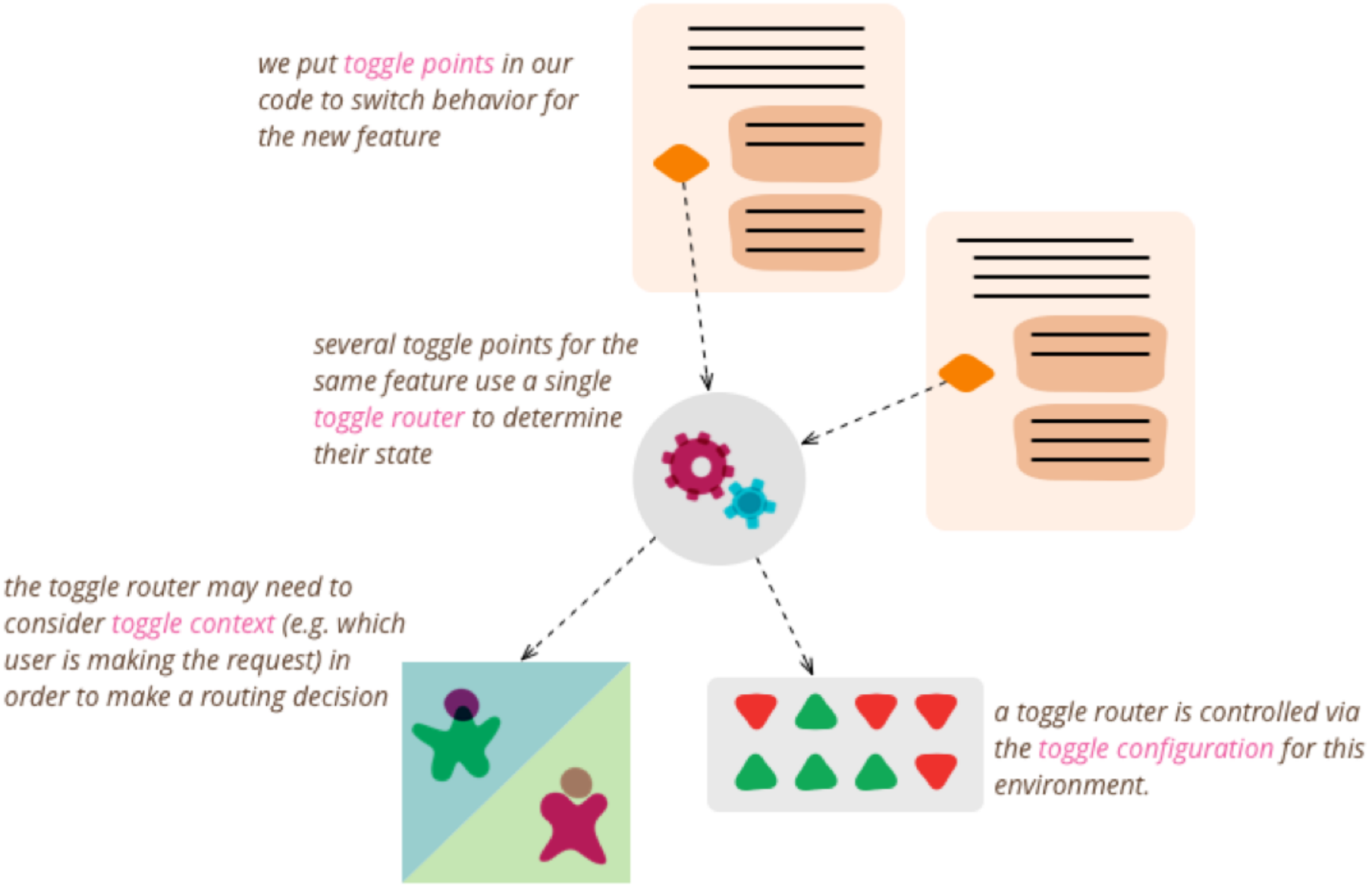
- Long-lived toggles for analysing system performance
- Disabling resource intensive features during high-demand periods

Permission toggles

- Enabling high-value content for premier users



Feature toggle infrastructure



Google Analytics

Understand site users to evaluate content/product performance

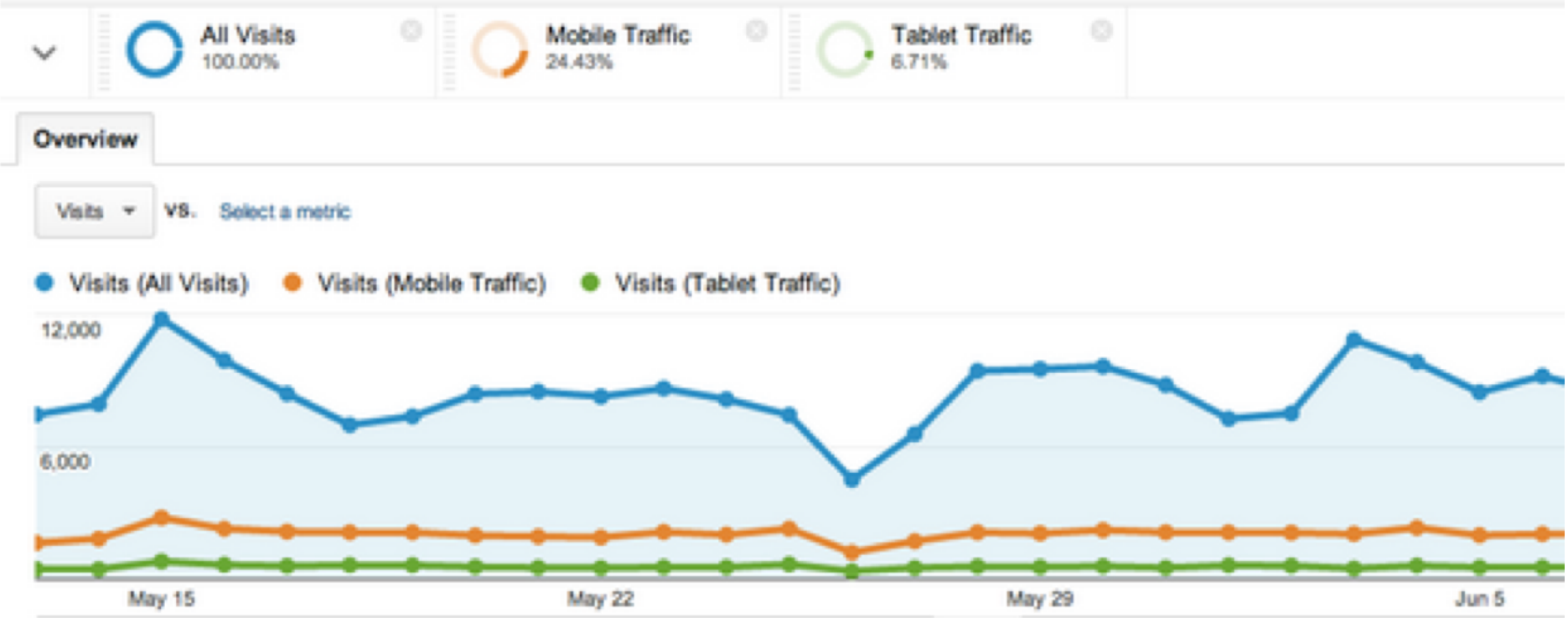
- **analytics.js**
  - Sends a **pageview** for each page your users visit
  - Google Analytics processes this data and can infer a great deal of information

Information from Google Analytics

- Information from **pageview** data
  - Total time spent on site
  - Time spent on each page (and order each page visited)
  - Internal links clicked
- Information from IP address, user agent string, initial page inspection
  - Geographic location of a user
  - What browser/OS are being used
  - Screen size and whether Flash or Java are installed
  - The referring site

User segmentation

Different user segments have different behaviours



appmetrics.js

A library for defining your own events and reporting to DevTools and Google Analytics

- Name events
- Record time and duration of events

```
// Each metric name should be unique.
var metric = new Metric("my_event");

// Mark name will be "mark_my_event_start".
metric.start();

// Mark name will be "mark_my_event_end".
metric.end();

metric1.sendToAnalytics("my_event");
```

Summary

DevOps Pipeline

- Pipeline automation
- Benefits for all

Analytics

- How are users using a website?

Next

- MEAN Stack, Part III: Angular