**Declaration**

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks ("..."). Neither in part nor in its entirety have I made use of another student's work and pretended that it is my own. I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

I have not asked anybody to contribute to this project in the form of code, text or drawings EXCEPT as follows.

I received guidance and support from my father Mr Frank Carnovale in these areas:

- Suggestions on focus areas related to deployment of commercial web application systems.
- Specific suggestion and coding instruction on how to deploy Perl library "Mojolicious" and related plugins in order to "kick-start" the application development phase.

Date 24/4/17

Signature D. Carnovale

DANIEL CARNOVALE K1336511

CI6300 – INDIVIDUAL PROJECT

# Uni-Hood Report

Daniel Carnovale

*Project Type : Build*

**K1336511**

**23/04/2017**

# Contents

# Abstract

Many low-cost cloud-computing resources and new free open-source software components are now available to software developers. These have reduced the costs that were associated with the creation of web-based software systems. Although a large team of IT professionals and expensive hardware and communications resources are no longer necessary, and can be replaced by just a few individuals, it's still essential for those individuals to be able to identify which tools and resources will meet their needs. They must also know how to effectively combine these tools to build their proposed system.

This project is an attempt to itemise and instruct (in a "HOWTO" fashion) one such combination of tools and resources that will allow a complete web application system to be built with minimal code, costs and manpower assuming the reader is starting from a standard Windows workstation with no specific tools pre-installed. We address some of the considerations that are essential for real-world feasibility, such as Installation of components, Security, Source control, Issue management and Disaster Recovery. To illustrate these procedures, I have deployed a system called "Uni-Hood" whose ultimate goal is to allow a university community to review and comment on courses, their modules and lecturers in a more structured way than the typical forums and social networks.

# Introduction

## Audience

This work is particularly important given that the increasing diversity of Open-Source components now available makes it difficult for students and individuals to get started with building a system such as Uni-Hood. This project will help them begin to realise the cost-savings that are being achieved via cloud-based computing.

Readers should be able to use these instructions to build a completely different application. It is my hope that many individuals and organisations would find the HOWTO deliverables of this project valuable because of the significant time-saving when starting a project using similar tools.

## Overview of Deliverables

Appendix A is the primary deliverable of this project work, and consists of a series of HOWTOs. These are designed to be simple, practical and readable instructions that allow a particular technical objective to be achieved, and yet do not assume a high existing level of expertise to understand and execute.

The next most significant deliverable is the running Web Application system itself at http://www.uni-hood.co.uk. This is a dedicated Amazon EC2 Instance which I have been using for this project.

The research on infrastructure, technology choices, installation and documenting the HOWTOs, repeating steps to verify their reliability etc. was a very large task and did not allow me to apply the level of web/database application programming that I have learnt from my Programming and Database courses. I felt it was more appropriate to focus on infrastructure and HOWTOs as this would provide more unique value in this body of work.

GitHub Issues were used as a means of documenting my research and any tips relevant to the HOWTO document or the topic area in general. These can be found at https://github.com/darkquake93/unihood/issues for inspection (each issue can be clicked on and their comments, status etc. will be displayed).

## Literature Review

The paper “Cloud Computing’s Effect On Enterprises [1]” details the current and developing stage of Cloud Computing, the associated costs and of course its impact on enterprises. This provides an insight on the rationale for moving to cloud-based resources and hence a justification for our goals in this project.

The paper “A Web-Based geographical information system prototype on Portuguese traditional food products [2] ” discussing PostgreSQL on page 14 provides (while brief) a concise and informative outline of the PostgreSQL Database system and strengthens a beginner’s understandings of its capabilities in general.

“User Guide - PostgreSQL and phpPgAdmin - Powered by Kayako Help Desk Software [3]” is a useful guide to the web-based phpPgAdmin tool, which played a vital role in Physical Database Design and Implementation.

The definitive reference for instructions on creation and management of the web server instance is “Amazon Elastic Compute Cloud (EC2) Documentation [5]”. This is exhaustive and covers many advanced topics beyond the scope of this report. One of our goals is to demonstrate that the initial build of a project such as Uni-Hood does not require such volume and detail.

In order to provide an easier alternative to such detailed reference resources, there is a long tradition of “HOWTOs” in the open-source community. These started as simple text documents, but are now more typically found as well formatted web pages and blog entries. A good example of a HOWTO that is directly related to this report is “HOWTO: Get started with Amazon EC2 [6]”. This is indeed a useful guide to the creation of an instance under EC2, however it seems to assume the reader is already working in the context of a local Linux machine and therefore has SSH functionality. It is also somewhat limited about creation of database services and subsequent steps.

In technical forums and blogs, many similar guides can be found, however the range of available web-building tools and resources (such as Amazon AWS Free Tier and Google Single Sign-On) is evolving so rapidly that any typical example may be out of date by the time a reader finds it. A similar caution should be applied to this report; in 10 months’ time a different tool set or procedure may be preferable.

# Analysis

## Methodology

The deliverables for this project were a series of instructions covering a variety of technical components, where the treatment of each technical component involved applying a specific methodology covering the evaluation, selection and ultimately the documentation of the component.

In more detail, this methodology involved proposing a certain technology / technologies (e.g. the choice of PostgreSQL as a database), utilising that technology to achieve a certain aim in a phase for the project (e.g. a reliable database technology has been made available) and if successful, noting this down as a preliminary step under a new HOWTO document. To verify that this document could achieve the same results each time, these preliminary HOWTO steps were repeated and tested several times. If the same result was produced then the next step would be constructed in a similar way, in relation to the next part of the current aim. If however the results were different, then this step was reviewed and modified on a trial-and-error basis until it was "stable".

As the project developed this verification procedure was found to be in fact not so straightforward, as it was found that on repeating steps in subsequent sessions, the "proper path through" for one day would not necessarily work for another. This is because assumptions about the exact technical context or false memories about the exact sequence of prerequisite steps played a significant factor in the invalidation of previous documentation.

The problem was addressed by ensuring that as many documented procedures as possible, even the earliest ones, were re-tried at later stages of the project, under constantly evolving technical conditions.

Ultimately when an entire HOWTO was completed, the steps undertaken to reach this phase in the project were reset to its initial state (e.g. in the case of PostgreSQL via termination of the entire EC2 instance, thus requiring re-installation of PostgreSQL when re-building the instance). From this point, the steps would once again be followed to observe whether the end-goal could still be achieved.

## Documentation Strategy

I had to make a decision as to whether the HOWTOs should be maintained in the .doc format or the special .md format (incorporating GitHub markup). I ultimately decided to stay with the .doc standard for general files, both because it is the widely accepted and expected format and it would take a while to write the markup for everything, and may not be worth the prevention of layout issues (a small chance for which .doc files may be slightly altered unexpectedly). For technical documentation I decided to stick with .md format though only the Readme.md file satisfied this scenario.

## Server hosting In House VS Cloud

A strong candidate for selection of the server was to set up and manage a Linux installation locally, and work against it as distinct from working against a remote cloud-based service. I opted for cloud-based because it's easily accessible from anywhere and is not affected by hardware or availability problems with local machinery.

In terms of choosing a cloud provider, I chose the Amazon cloud-based environment due to it being a very well known company; offering great customer service and reliability, a free tier program and having a wide range of virtualisation tools and file storage available to its users. The EC2 and S3 services are the most relevant for this project, and when combined can immensely strengthen a virtual Amazon instance. As an additional benefit, through hosting the virtual server in the cloud, it is both accessible anywhere.

## Registering for Amazon Web Services

The aim to minimise the costs associated with the project was a priority when setting up a new Amazon account which would be linked to the Amazon Web Services. After some research, I discovered there was a "Free Tier" period that is included in the creation of a new account, so I could conveniently utilise this instead of my current Amazon account. To distinguish this from my main account, it was necessary to link it to a fresh Gmail account also.

After linkage between the Gmail account and Amazon account was completed (through simply providing it as the email address during sign-up), I started the process of linkage between the Amazon account and the Amazon Web Services. It is a requirement of the AWS sign-up process to link this account as the services will be tied to it. The sign-up process does not provide you with full access to the AWS services right away, as it is seen as a kind of request form for which you will need to wait on a response from and/or keep in contact with Amazon themselves regards if and when they allow you to use these services.

This took longer than expected however, as there was some small issue between my bank (Barclays) accepting or managing the small transaction used to verify this linkage. I contacted Amazon support and started this as a small case, for which after a few message exchanges was resolved very quickly. I was impressed with the speed and quality of support from Amazon, and it helped me proceed with little disruptions to the actual creation of the Uni-Hood virtual machine.

Regards the associated email, any email address can be used for making your AMAZON account (the AMAZON WEB SERVICES ACCOUNT is different), Gmail was my preference. Make sure that this email is valid and not previously associated with Amazon or there may be conflicts. I suggest as a best practice to create a new one for this purpose.

It took 3 days to resolve the above support issues, after which the Amazon Web Services were linked to my account and this free tier program was then available for use. It greatly helped minimise the costs when using virtual machines. Options which are eligible for this are indicated clearly on each step of the instance creation process, so there is strong assurance that the right choice is being made cost-wise.

I have elaborated on this in the “Amazon Free Tier Account Creation” HOWTO.

## Hack attempt prevention, disabling clear-text password login access

It was helpful to consider from the outset whether any newly created Amazon machine was vulnerable to any form of attack. I gathered advice on this and determined that we are protected from this because Amazon Linux by default does not allow password-based SSH access. It is only possible to access the instance using a privately created key-pair.

## OS (Windows VS Linux)

I chose Linux as the development platform because it appears to be very much the leading platform for Cloud-Based technologies, especially as regards Open-Source components.

## Selection of Technologies

Project analysis involved the use of forums, help pages on the web and advice from colleagues to start investigating available technologies and procedures. I considered technologies including GitHub, MySQL, PuTTY, PHP, Python, Ruby / Perl, HTML5, CSS, Javascript, Bootstrap, Angular, JQuery UI, mySQL, PostgreSQL, Ansible and Docker, some of which were utilised in the project. The Amazon services EC2, S3, EBS and Elastic IPs were all implemented into the project as utilisation of the wide range of Amazon technologies was the main direction I wanted to head into for this area, and I intended to make full use of their “free tier” procedure / offer.

This was essentially an “agile” approach in that I chose to investigate tools and make quick assessments on which ones to deploy, without performing a very exhaustive research on the merits of each. I continued use of a tool or resource if it could be quickly and easily deployed to achieve an end, however when this did occur I made notes on how each such tool was deployed. This resulted in the recording of sufficient material to turn these notes into fully elaborated HOWTOs.

## Docker consideration

The Docker Virtualisation environment was also considered, however it required a further level of configuration compared to that of the Amazon EC2 instances. The idea was abandoned however noted down as a suitable fallback in case there was some issue with Amazon (however very unlikely).

## Choice of Amazon as a Cloud Provider

Amazon already has a very high reputation of offering great Customer Service as well as being a massive marketing platform. After having used their S3 service in particular their storage seems very stable and offers a large amount of data to be stored so it seemed a strong candidate to make use of this as a cloud system for my project. Linking the storage facility with the virtualisation environment resulted in a very strong cloud-based solution.

## Choice of Virtual Machine Hosting service

I chose Amazon EC2 because I was already impressed with solely their Amazon Simple Storage Service (S3) at first, after having experimented with it for a while for general purposes, so had no

doubt the EC2 service would serve just as well. It presented me with a very simple interface, quick response times, great support and so on and only had the slight issue with linkage of these services to my account however these were resolved very quickly.

## Choice of Source Control (GitHub vs Dropbox)

I chose GitHub as a repository for all screenshots, source code, version control aspects and documentation (issues etc) needs. I manage it locally using the Windows application "TortoiseGit" which allows me to easily set up a new Git repo, and have access to many Git features though mainly I simply work off the local files, then commit and push them to the master repository link on GitHub. I also on the web interface manage "issues" which are small paragraphs of current bugs, reminders or just general notes regarding the project. I also have the opportunity to revert back to a past commission if something goes wrong in the current stage.

Dropbox on the other hand was a great repository for note taking also, and all important notes and even Putty files and similar were encrypted using the "Boxcryptor" app, which means that if my Dropbox account was compromised the encrypted files could only be read through an additional layer of security through this app. Locally, it's very easy for me to login to the Boxcryptor app and view these files on a separate virtual drive safe from the eyes of the public.

## Access from Local Workstation to Remote Server

I decided to use the Windows Putty application because I am already very familiar with using it to connect to remote servers through the SSH (Secure Shell) protocol and by default it provides an excellent terminal interface to work off. Linux, being a terminal-based operating system itself, works very well with Putty and tips for connection through it are linked to from the connection window on the Amazon EC2 Management Console.

## Choice of Linux distribution (Amazon Linux)

I felt it was safe to assume that at this stage Centos 6 and Amazon Linux are similar in terms of feel and functionality. Any instructions that were followed were followed in reference to steps outlined specifically for Centos 6, though in the rare case they would fail then the steps for Centos 5 would be followed as a fallback.

Amazon's "Free Tier" service allows this Operating System to be used for free or very low-cost during the first year.

## Selection and purchase of the Domain Name

Several titles for this project were considered before any implementation was done, though once Uni-Hood was decided and searched for to verify it was a unique name the next step was to "morph" it into a domain name format, and purchase it. GoDaddy was the main go-to here, so a domain name of uni-hood.co.uk was bought for a very cheap price of £0.99. (GoDaddy Receipt #: 1031408714).

## Consideration of static (elastic) instead of dynamic ip

Every time an instance is restarted a different IP number is given, which is inconvenient yet free. Ultimately, a small fee is required for a single elastic (static) IP number which is then assigned permanently to your domain name using your DNS provider management settings (e.g. Zoneedit).

Obtaining an elastic IP address was a very simple procedure as well as its association with the instance. These are both covered in the “Creating and Associating Elastic IP to Instance” HOWTO document.

This procedure is completely free as it is a single Elastic IP attached to an instance, and Amazon states “You can have one Elastic IP (EIP) address associated with a running instance at no charge.” This reference also details all the pricing for standard instance uptimes, outside of the Free Tier. “EC2 Instance Pricing – Amazon Web Services (AWS),” [4]. My finding here was that ultimately, to reduce costs, the instance should be kept STOPPED but not TERMINATED.

## Consideration of EBS service

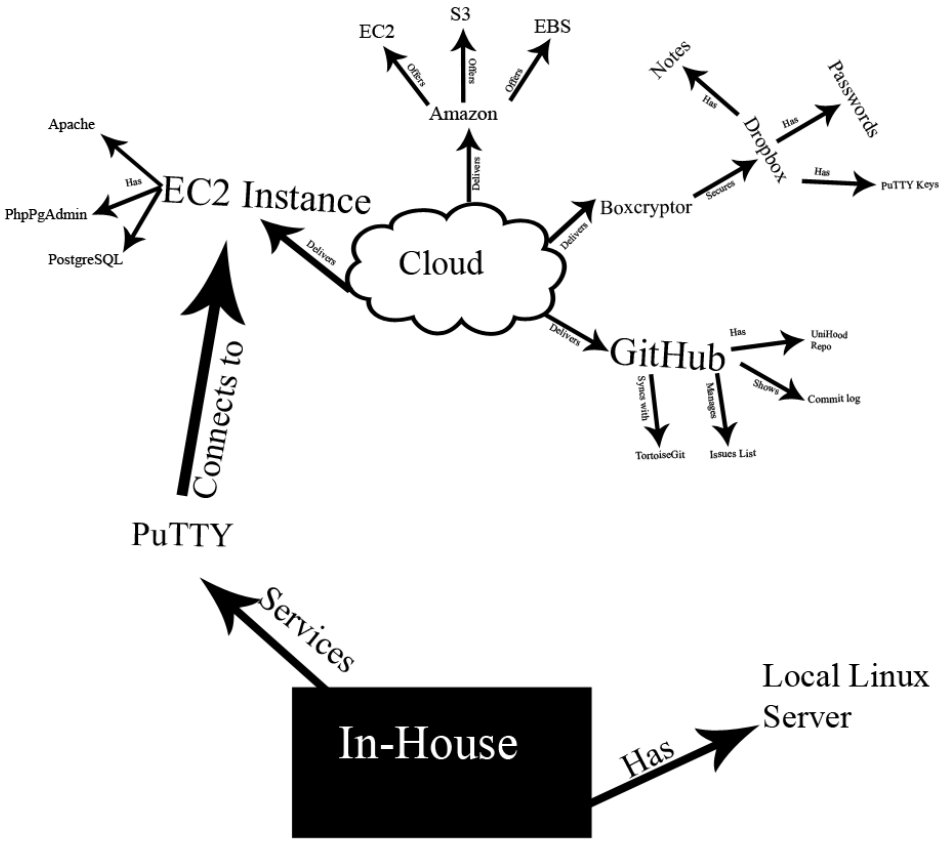
I wanted to investigate how to manage large amounts of data as a side-objective in case I needed to utilise this later, which led me to the experimentation of EBS Volume creation, discussed further under the HOWTO Deliverables “Creating / Removing EBS Volumes” and “Managing EBS Volumes”.

## Selection of Amazon Machine Image (AMI)

At this stage it was necessary to choose an “Amazon Machine Image (AMI)”. The default choice is Amazon’s own distribution of Linux titled “Amazon Linux”. I observed under experimentation that this distribution behaves very similarly to the Centos 6.X version Operating Systems which I know to be a simple, clean, standard starting point.

# Design

## Schematic of System Components



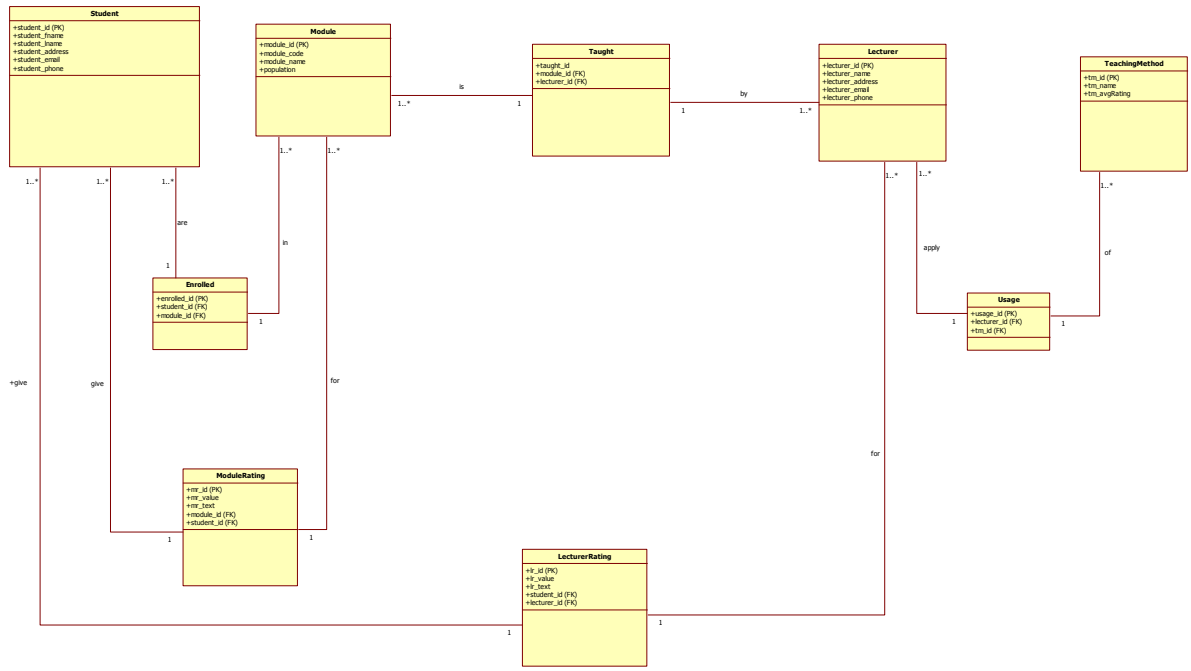
## Summary of Technologies Utilised

We present a summary of component areas, eventual choices and relevant notes.

| Component | Choice | Notes |
|---|---|---|
| Cloud Service Provider | Amazon Web Services (AWS) | Dominant cloud services provider in Industry |
| Virtualisation Service | AWS Elastic Compute Cloud (EC2) | Free-tier allows this type of research at minimal cost |
| Secure Data Archive Repository | AWS Simple Storage Service (S3) | Highly redundant and secure service, allows upload and download of very large files, very low cost |
| Linux Distribution | Amazon Linux | Default and well supported O.S. under EC2, zero licensing cost |
| Public IP Address Arrangement | AWS Elastic IP Service | A chargeable concept in which static public IP numbers can be assigned to EC2 instances. |
| Online Storage Facilty | AWS Elastic Block Storage (EBS) | Low-cost For adding extra drives to currently running Virtual Machine instances. |
| Source Control Service | GitHub (http://github.com) | Cloud-Based Git Server, used as repository of all code and project documentation. Free service for Open Source project work (but all project content is exposed to public). Monthly charge for private repositories. |
| Database | Postgres | Free and open source RDBMS. |
| Javascript Library | JQuery | Industry standard library for dynamic control of web pages |
| CSS Theming and Styling Library | JQuery-UI | No longer “leading edge” but convenient as bundled in with chosen Mojolicious Plugin library (“Tables”) |
| Data Design Tool | StarUML | Windows-based graphical application for presentation of data modelling and data design. Licensing costs range up to USD99 for commercial use. This project used a University-hosted instance. |

## StarUML Class Diagram

After a significant level of experimentation with the "StarUML" application through my previous University years, I was able to quickly apply my knowledge to this project, and through the click-and-drag nature of the graphical environment I was able to quickly construct a basis for the actual database structure. I then was able to export this as an image which I committed & pushed to GitHub, so I could always refer to it whilst implementing the database.




## Database Design Strategy

To support the creation of this application, it was necessary to design an application database in the realm of the chosen functionality area (namely Uni-Hood).

The "StarUML" application was particularly useful here as it was much faster and easier than sketching the tables on paper in some respects, however each approach has their advantages. I was able to export this work as an image I could always refer to, while maintaining a top-down approach in that classes with no foreign key linkages were at the top of the diagram, and the rest below them. This made it very easy to start off with creation of the top classes and work downward, instead of having a "foreign key trap" of sorts in that a column depended on another column however that column depended on the one referencing it, or the existence of another table that could not yet be created due to a similar situation. Database design and physical implementation go hand-in-hand, in that it took me several iterations to perfect the structure of the physical database whilst changing the design diagram the whole time in sync. Once at a good stage however, it was ready to be utilised in the main web application later down the track.

## Physical Database Implementation

The phpPgAdmin environment was my choice for the database to run under. Initially there was confusion regarding the PostgreSQL version to use (9,91,92,93, etc). In the end, I decided to go with the latest one (95) but later found serious compatibility problems with phpPgAdmin. A workaround was a quick edit to the “Connection” php service file, the link for which is in the Appendix.

Firstly however it was necessary to install the PostgreSQL and Httpd packages on the virtual machine instance as well as some additional packages. The whole database management environment is included in these packages, so after starting them and appending /phpPgAdmin to the instance ip you will be presented with an interface to which implementation of the database diagram (from StarUML) can be applied. Creation of the database and tables through the interface is fairly straightforward, however logically you must pay attention to the usage of foreign keys and make sure they match the flow of the diagram. Once satisfied with the table creation and their related columns within the tables, test data can be inserted as rows and reviewed within a table’s overview page. You may encounter a scenario where only ID’s are displayed instead of the actual piece of data they are referencing, so a more visual interface allowing foreign data to be recognised was desirable. The open-source “Tables” facility was adapted to account for my data instead, and foreign key values were correctly resolved to the values they linked to and visualised. All database data from the original phpPgAdmin environment, including the test data, can be exported to a “dump file” and this process is outlined in the “Backup/Restore Test Data” HOWTO. This file was then able to be imported into the new facility, as both environments were capable of recognising these SQL dump files.

I tried to convert the UML Diagram from StarUML straight into SQL queries, however though I attempted to install certain add-ons, or find certain functionalities (different across StarUML versions) or whatever seemed necessary to enable this feature I could not manage to proceed. In the end I had to create these statements manually when required, as mainly the graphical interface already allowed creation of these properties.

## Data Sources

Sample test data had to be improvised according to the evolution of the project. The reasoning for this is that it is always impossible to speculate how the final database structure will come together, so as it evolves with the project so will the test data. No alternatives for the test data were to be discussed nor were they necessary; although this is a practical project it is also an Open Source one as stated earlier, which means the data cannot be limited to just one organisation (in this case Kingston University).

Once the database structure was set up, it was a simple process to insert the sample data through Structured Query Language (SQL) statements. SQL is very powerful when used correctly, and as mentioned above the data was improvised as there was no physical source of a student / lecturer list etc available, in addition to the high chance it would not be compatible with the current database structure. Hence when a query worked for a table, it was repeated with modified values. Finally, the open-source “Tables” application was able to present this data in a more logical or meaningful way.

Test data was unchanged across database variations, in other words it was used consistently and unmodified so as to work within a true testing environment. The test data as well as the whole database tables and their columns were frequently exported as dump files as mentioned earlier.

# Implementation

## Installation of PostgreSQL

PostgreSQL is a Database Management package and its installation is as straightforward as that of the Apache or phpPgAdmin packages (using apt-get or similar, outlined in the "Installing and Configuring Postgres, setting up DB Owner" and "Installing Apache and PHP, and Managing the WebServer" HOWTOs. It supplies all the standard Database aspects including columns and their tables, a range of different datatypes, import/export functionality, multiple character sets and so on. When this is utilised within the Apache webserver context it allows a powerful way to put the Database diagram from StarUML into effect.

## Standard Connection Method : Console Access

There is normally no "console" access as expected in stand-alone workstations or in VPlayer-type environments; alternate instructions were provided upon clicking the "connect" link for the instance, however the easiest way is to simply identify and copy the instance's IP. I needed to initiate a bash prompt connection onto the newly created instance which is discussed further in the "Key Pair / Putty" HOWTO.

## Optional Connection Method : Mobile Console Access

An extra convenient option for managing your AWS instance is to use the AWS Console Mobile Application. I found that sometimes the instance state is a bit sticky or completely doesn't work using the desktop interface resulting in being charged for an instance I thought stopped. With the mobile application it is not only easy to review whether your instance is running but also (I found) have more reliability regards changing its state. Instructions on setting this up can be found in the "AWS Console for Mobile" HOWTO.

## Source Control from Windows : TortoiseGit

Under Windows, GitHub source-control can be accessed from the TortoiseGit application. This is covered in more depth in the "TortoiseGit for Windows" HOWTO.

## Secure / Authenticated Command Line Access : Putty and PPK file

PuTTY is a Windows Application for connecting to a virtual Amazon EC2 instance, the procedure for which can be found under the "Connecting to the Instance" HOWTO. Once it is installed and your PPK file is available, it's convenient to make the program Pageant activate this file when you login to Windows. This tip and the configuration of PuTTY and the PPK file are discussed in the "Setting up Putty and Encrypted PPK File" HOWTO. The "SSH KeyPair Generation" HOWTO outlines the generation of the private key used in the setup steps. The passphrase allows you to authenticate yourself one time only at the start of each Windows session (e.g. beginning of day etc.) and will subsequently allow seamless and secure connectivity to your EC2 instance without having to re-enter a conventional password at every login.

## Optional GitHub Management Tool : OctoDroid

Similar to the AWS Console app for Mobile (above) the OctoDroid app allows easy viewing and modification of your GitHub repositories and general user preferences. Your repositories are listed initially, and after selection several insights become visible. More information is found in the "OctoDroid for Mobile" HOWTO document.

## Instance Management : Stopping and Restarting the EC2 Instance

Sometimes you will wish to stop and / or restart your Amazon EC2 instance so as to save costs, or even to re-initialise services or similar requirements. This is explained under the "Stopping and Restarting the EC2 Instance" HOWTO document.

## Application Development

A complex application programming phase is beyond the scope of this project. To help kick-start what will become a fully developed web application, I obtained advice and assistance in the selection and implementation of a suitable database / web application framework. This led to the installation of the Perl "Mojolicious" ("Mojolicious - Perl real-time web framework [7]") library plus some plugins which allowed almost immediate web-based browsing and management of the relational database. Readers of this report could consider this library or similar offerings in other web application languages such as PHP or Python. In my own case, I am not yet an experienced web application programmer, but the strategy I adopted here has given great visibility to all of the setup work covered in this report.

## SQL Injection Attack Prevention

The delivered web app is clearly a database-backed application, and is the type of app which is typically subject to SQL Injection Attacks. This vulnerability can be completely prevented by ensuring that the application logic uses bind variables whenever constructing SQL Queries. The Mojolicious framework and its plugins have allowed all database access to be managed internally using bind variables and it was never necessary to write any database management code that would have constructed literal SQL statements.

# Testing & Evaluation

Testing for this project falls into two categories:

A) Do the primary deliverables (HOWTOs) achieve their purpose?

As discussed previously in the methodology, I have a very high level of confidence that HOWTOs are genuinely useful, however it must be accepted that this is a subjective evaluation and can only really be determined by others.

B) Does the Web Application demonstrate that the system being build is a truly database-backed system?

Testing consisted primarily of using the generated web app to create and browse structured data relevant to the Uni-Hood application. It was convenient to make use of phpPgAdmin on many occasions to both load up and review consistency of information presented by the web app. For usability testing I exposed the application to several users for sample feedback. The most significant message coming back from this form of testing was that while the system succeeded in presenting and maintaining database information, it did not have the actual processing functionality required to fully deal with the original requirements. This was an expected result because the web framework is a CRUD style (Create, Read, Update, Delete) database navigational interface.

# Critical Review

I consider that this project was successful in that the primary deliverable (i.e. the HOWTOs in particular) will serve as an informative and valuable resource for the defined audience. I believe anyone would greatly benefit from my HOWTOs, and also become aware of any steps that are not immediately obvious to them when executing the current task. The dual numeric sequence / screenshot nature of these HOWTOs will allow users to follow the steps efficiently.

The literature review considered sources for Cloud Computing, installation and configuration of tools such as PostgreSQL, phpPgAdmin, StarUML etc and Amazon Services such as Elastic IPs. These sources significantly boosted the development of these HOWTOs and remain strong assets that have built on this knowledge. This report does not necessarily improve those resources, but as intended, serves as a single compilation of steps drawn from many sources.

I have not particularly made improvements regards utilising these technologies, but rather adapted them to my own project's requirements. Although I managed to implement a significant level of these I did not manage to develop them to their full potential given the time frame and resources available for the project.

From the initial to the final stage of the project, generally the components came together nicely though naturally, and as previously mentioned, a few iterations of each HOWTO were necessary. Some problems regarding differing versions of software, incompatibility and so on slowed down the production of the HOWTOs although in some cases was beneficial as they could be noted down as precautions. I handled each of these problems efficiently and had several internet sources to rely on as I proceeded.

I found the PhpPgAdmin Database tool very useful not just in terms of managing a database, but in terms of seeing the project's structure and data design model in action and as a backup strategy in terms of exporting the SQL files. I would strongly recommend it to anyone studying a similar project matter, it was very much an invaluable resource.

Through easy mistakes regarding instance up-time the ability to execute an entirely zero-cost exercise was not achieved. As shown in (bil-1 to bil-3) for Amazon Billing per month (only two are shown for the usual amount I was billed over the previous months, and one from the current month), I wasn't maximising the benefits on the Free Tier usage. You can see the difference however when the price almost halves for the month of April (though not over yet, but I have paid attention to my instance's uptime.

## Ethical Considerations

The system uses Open-Source and/or free components as much as possible. Resources such as the Amazon Free-Tier are encouraged for student use, so for these reasons I see this system as having no ethical conflict. Regarding the application system "Uni-Hood", if it does actually get exposed to the community for use then this potentially raises questions of privacy and confidentiality of the users involved.

## Intellectual Property Rights / Access to Code

All of my own work and code is itself Open-Source and visible to any GitHub user, and I have not duplicated any similar work, hence I do not believe there are any issues regarding intellectual property rights. As is the convention with GitHub projects, I am allowing my work to be publicly available and freely fork-able by these users, with no limits in place such as freedom of distribution.

## Data Protection

A feature of my method and an illustration of the modern approach to this type of project is that all code and documentation is committed to GitHub, giving in effect a high confidence level that not only all work is easily accessible from the cloud, but a full audit trail is available. In order to protect from the extremely unlikely scenario of a catastrophic failure of the GitHub ecosystem, the system itself being built works from its own local Git repository, which is a clone of the GitHub repository. Regards database backup and recoverability concepts, note that even the test database dump file is managed and version controlled in GitHub.

# Future Directions for the Project

For Uni-Hood to reach Production status, some or all of the following missing features are required. In the context of this report, this means that there would be opportunities to write HOWTOs for each of these areas because they are all common to most modern web apps.

## Regular System updates

Encorcing prevention of vulnerabilities regarding security issues has been strongly recommended for a while now, and a reminder is always present. It is necessary to run a “yum-update” command on a machine running a Linux OS. This helps to enforce the prevention of vulnerabilities regarding security issues that may have come about, due to flaws or loopholes in older versions of the installed packages.

When this service enters the production phase, the plan would be to run a “yum update” via a Crontab job.

### Testing of HOWTOs

The HOWTOs have to be futher “field-tested” by publishing them and putting them in front of other people.

### Google Single Sign-On

It is essential for users of this application to be authenticated. Rather than implementing the traditional username / password scheme, it was my intention to adopt a Google Single Sign-On technique which allows reliable authentication of users without the application needing to see or manage their passwords.

“Integrating Google Sign-In into your web app [8]”

### HTTPS Certificate

It is necessary to purchase and apply an HTTPS certificate to ensure all data between user’s browser and the server is encrypted, as well as communication.

### Backup/Disaster Recovery

While our application software is already fully under Source-Control at GitHub, the database needs to be subject to a regular backup regime and a full disaster recovery procedure should be documented.

### Configuration Automation : e.g. Ansible

New tools such as Ansible make it possible to automate some of the steps in these HOWTOs. It should be investigated to see if this has a positive or negative impact on setup complexity as documented so far. An initial attempt was made to prepare an Ansible configuration file, this was discontinued however after I became aware that Ansible cannot be driven from Windows and is therefore outside our assumptions about the starting environment.

### Scalability

One of the strengths of a Cloud-Based solution is that under massive load it should be possible to scale the number of instances accordingly. Procedures for this should be investigated and documented.

### Alternative Cloud Providers

It's possible that alternative Cloud Providers such as Azure, Mega and so on are easy to learn or are better value for money now or in the future. Ideally a similar project should be completed using these services.

# Appendix A: HOWTOs – Attached PDF

Note: On spelling / punctuation I am following the classic UNIX / LINUX style of writing “HOWTOs”, reference:

http://www.tldp.org/

# Appendix B: Screenshots – Attached PDF

# Bibliography

[1] R.Saleem, 'Cloud Computing's Effect On Enterprises', Lund University, Jan-2011. https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=1764306&fileOId=1764311 [Accessed: 21-Apr-2017].

[2] Ye, Z., 2009. A Web-Based geographical information system prototype on Portuguese traditional food products. https://run.unl.pt/handle/10362/2318 [Accessed: 19-Apr-2017].

[3] User Guide - PostgreSQL and phpPgAdmin - Powered by Kayako Help Desk Software [WWW Document] https://support.eapps.com/index.php?/Knowledgebase/Article/View/68/66/user-guide---postgresql-and-phppgadmin [Accessed: 20-Apr-2017].

[4] EC2 Instance Pricing – Amazon Web Services (AWS) [WWW Document], n.d. . Amaz. Web Serv. Inc. https://aws.amazon.com/ec2/pricing/on-demand/ [Accessed 21-Apr-2017].

[5] Amazon Elastic Compute Cloud (EC2) Documentation [WWW Document], n.d. . Amaz. Web Serv. Inc. URL //aws.amazon.com/documentation/ec2/ [Accessed 17-Apr-2017].

[6] Hull, S., 2012. How-to: Get started with Amazon EC2 [WWW Document]. InfoWorld. URL http://www.infoworld.com/article/2615510/cloud-computing/how-to--get-started-with-amazon-ec2.html [Accessed 17-Apr-2017].

[7] Mojolicious - Perl real-time web framework [WWW Document], n.d. URL http://mojolicious.org/ [Accessed 21-Apr-2017].

[8] Integrating Google Sign-In into your web app https://developers.google.com/identity/sign-in/web/sign-in [Accessed 21-Apr-2017].

CI6300 – INDIVIDUAL PROJECT

# Uni-Hood Report

## Appendix A: HOWTOs

**Daniel Carnovale**

**23/04/2017**

# Contents

## Amazon "Free Tier" Account Creation

**NOTE: You may already have an Amazon "shopping" account, used for purchasing books, DVDs etc. It's possible to let this account to be the basis for your usage of Amazon Web Services, bearing in mind that any billing for the project will be sent to your personal account. Billing refers to anything beyond Amazon's "free tier". If you prefer billing on the project to be kept separate (e.g. Academic / corporate requirements) then best practice is to create a new Amazon account first.**

1. Create a new Amazon account if necessary
2. Tie the Amazon account to the Amazon Web Services Free Tier, by clicking "Create a Free Account" here: https://aws.amazon.com/start-now/
3. After following the standard prompts, you will be notified that services will be available soon.
4. Once services are available, make sure you are signed into the Web Console from the "Sign In to the Console" link here: https://aws.amazon.com/console/
5. Try to use some services, the easiest test is the "S3" service to share very large files privately or publicly

## Creating and Associating Elastic IP to Instance

1. Log into the Amazon Web Services Management Console https://aws.amazon.com/console/
2. Obtain a new Elastic IP Address by clicking “"Networking & Security" on the left, click on "Elastic IPs" then click "Allocate New Address" (ela-1).
3. Click "Associate Address" filling in the Instance and Private IP fields which should present suggestions upon clicking them. Click “Associate” (ela-2)
4. The elastic IP is now associated with the instance and can always be used even when the instance is restarted

## AWS Console for Mobile

1. Download the AWS Console Mobile Application which can be found on the Google Play store: https://play.google.com/store/apps/details?id=com.amazon.aws.console.mobile (aws-1)
2. Open on your mobile and sign in as usual
3. You should see a similar screen to (aws-2), from which you should click the “EC2 Instances” row
4. From here, you can manage your instance/s through clicking its row, and the appropriate option. Or simply to review its status (aws-3)

## TortoiseGit for Windows

1. Download the application from https://tortoisegit.org/download/ (tor-1) and make sure you choose the right architecture as highlighted in the screenshot
2. Installation should go smoothly, go with all default installation options
3. If you try to run the application this dialog will show (tor-2) making clear that its usage is as a context menu
4. On github.com, create a new account if you have not already done so and then create a new repository (tor-3)
5. Name it something meaningful, initialise it with a readme (as this is a fresh repo) and continue (tor-4)
6. Click Clone or download, and copy this link for use with TortoiseGit (tor-5)
7. Right click an empty area on a drive in Windows Explorer, and click “Git Clone...” (tor-6)
8. The url and details should already be filled out, so just click “ok” (tor-7)
9. It should be cloned successfully, and ready to have files pushed (tor-8)
10. To test committing and pushing a file, create a new text document in the repository (tor-9)
11. Right-click the Git repo folder, and click “Git Commit -> master...” (tor-10)
12. Type a meaningful message, check All files, select the “Commit & Push” option from the drop down before you click that button. Add the author date if you wish (tor-11)
13. This should complete successfully, and you can also manage the Git Repo on Github.com (tor-12 & tor-13)

## OctoDroid

1. Download the OctoDroid for GitHub Mobile Application which can be found on the Google Play store: https://play.google.com/store/apps/details?id=com.gh4a&hl=en_GB (oct-1)
2. Open on your mobile and sign in as usual
3. Select your repository for viewing / modification
4. From here, you can manage your instance/s through clicking its row, and the appropriate option. Or simply to review its status (oct-2)
5. The commit trail here is very useful as an example to quickly check if your latest commit has gone through (oct-3)

## SSH KeyPair Generation

1. Log into the Amazon Management Console https://aws.amazon.com/console
2. Choose EC2 from the high-level list of services under the “Compute” Category (key-1)
3. Click Key Pairs on the left-hand menu
4. Click “Create Key Pair”
5. Choose a simple new Key Pair name, typically named after yourself (key-2)
6. It will automatically download this as a .pem file, used in the later steps

## Amazon EC2 Instance Creation

1. Click Launch Instance on the top-left of the Amazon Management Console (EC2).
2. For the purposes of this project, Amazon Linux 64-bit should be selected (indicated as part of the Free-Tier program) (EC2-1).
3. The 1GB t2 micro memory instance type should be selected (EC2-2).
4. Leave Instance Details as default, as any changes here could bring about complications or uncertainty in the future when dealing with network details and such (EC2-3).
5. Change the root storage disk size to 30 gb. This should be more than enough space for the simple needs we require, and is of course also eligible for the Free Tier (EC2-4).
6. At this point you are able to name or “Tag” your instance. There are several other tags available to set here though the only one to be concerned with for our purposes is the Name tag. I have called my instance “vlinux” but any other simple name will also be appropriate (EC2-5).
7. Choose the “Select an existing Security Group” radio button, and then select the only group there (named default). (EC2-6).
8. Proceed to launch the instance and create a new key pair, at the same time assigning it a name. Don’t forget to click Download Key Pair here! You can’t download it anywhere else or call up this dialog box again. (EC2-7 & EC2-8).

## Setting up Putty and Encrypted PPK File

1. Download and Install PuTTY from this site: http://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html (put-1) using the full msi Windows installer package for your current system architecture

2. Open PuTTYGen and Load in your .pem file downloaded during "SSH KeyPair Generation" by clicking "Load" highlighted in (put-2)

3. Invent a memorable passphrase and enter it, and again to confirm it and finally generate a .ppk PUTTY Private Key file from the pem file by clicking "Generate" (put-3)

4. Click Save private key and you will be prompted to choose a location to save this key to, I suggest saving it to the same place as the pem file.

5. Add PPK file to the Windows Startup folder by simply dragging it in, the easiest way to access this folder is to paste "`%appdata%\Microsoft\Windows\Start Menu\Programs\Startup`" into the address bar of Windows Explorer (put-4)

6. Double click this file to test (also executed on startup), enter passphrase

7. PuTTY should now be authenticated with this file, and you can proceed to connect to any EC2 instance which was activated using the SSH keypair corresponding to the .pem file

## Stopping and Restarting the EC2 Instance

1. Optionally download and install the AWS Console Mobile app, I have found this is more responsive than desktop in terms of managing the instance state. Link to app on Google Play Store: https://play.google.com/store/apps/details?id=com.amazon.aws.console.mobile&hl=en_GB
2. Log into your AWS account either on Mobile or Desktop into Management Console. Desktop url: https://aws.amazon.com/console
3. Enter the EC2 Instances sub-menu.
4. Select the instance(s) you want to stop.
5. Tap the Stop button, the instance will now perform a Shut Down
6. Similarly, you can tap the Start button if the instance was stopped, in effect having restarted the instance.
7. At any point, call up Instance Settings > Get System Log to verify the state of the instance console

## Connecting to the Instance

1. On the instance list on the EC2 page, select your instance.
2. On the description tag below, copy the public IP number (con-1).
3. Open Putty and feel free to save this as a new session: right-click the tray icon and click new session.
4. Paste the ip into the Host Name field, click on Data under the Connection category on the left column and enter ec2-user into the Auto-login username.
5. Click back on the session tab and click save but do not open yet.
6. Right click on the tray icon again and click add key, and select the ppk file you downloaded earlier.
7. Right click on the tray icon once more and click Saved Sessions > (your sessions name). The connection should be established.
8. To perform admin tasks, type “sudo –s”
9. To overview all available packages, whether installed or not installed, type “yum list | more”

## Creating / Removing EBS Volumes

1. Launch this instance, and then click Volumes under the ELASTIC BLOCK STORE category on the left column. Click Create Volume on the top-left corner of this window. (EC2-9).
2. For this volume, choose the same availability zone as the current root drive is allocated to. (EC2-10).
3. Right click this new volume, and click Attach Volume. (EC2-11).
4. Click on the instance field, it should show your instance id as the only item in the list that pops up. Note the Device field value here, this is important. (EC2-12).

## Managing EBS Volumes

1. You will first need to make a new file system with the location of the drive (sdf in my case, also displayed in the details of the drive when clicked on in the AWS window).

   In the terminal, type “mkfs /dev/sdf” where sdf is the device name field from step 12.

   Make a new directory which maps to this drive, I called mine “bigdata”.

   - mkdir /mnt/bigdata
   - mount /dev/sdf /mnt/bigdata

   Check disk space using the “df” command.

   Verify that the 100gb extra EBS volume has been mounted on the directory you specified above. You will notice this in the same output from the “df” command, as it also displays the size of each drive as well as the directory they are mounted on.(EC2-16).

2. To confirm the drive can be written to, a simple test can be performed. You can change the working directory to the one which the drive is mounted on, list its contents, write the current date to a new file called “x”, output the contents of this file and then list the contents of the folder. The screenshot outlines how to do this. (EC2-17).

## Installing and Configuring Postgres, setting up DB Owner

1. sudo –s  # to become root
2. yum install –y postgresql8-server
3. service postgresql initdb
4. chkconfig postgresql on
5. su - postgres
6. cd /var/lib/pgsql/data
7. # edit the file pg_hba.conf
8. # duplicate the line with “local all all ident”
9. # the first line should have its method set to md5
10. # comment out the entries whose connection type is “host”
11. ctrl+d # to log out of postgres user
12. service postgresql start
13. su – postgres
14. psql # this opens up the postgreSQL command line SQL interpreter
15. alter user postgres with password ‘x’; # supply your own secure password here in place of x
16. create user unihood with password ‘unihood’; # similarly here
17. create database unihood;
18. grant all privileges on database unihood to unihood;
19. ctrl+d # to log out of postgres user
20. ctrl+d # again to log out of root
21. # you should now be back to ec2-user
22. psql –U unihood unihood # supply the suitable password for the unihood database user (first ‘unihood’ is username and second ‘unihood’ is database
23. # if you see a prompt similar to unihood=> you now have a valid postgres database owner

# Screenshots: pak-1 shows the result of a yum-list command listing postGreSQL92. Note that we now recommend falling back to postGreSQL8 for compatibility with phpPgAdmin. pak-2 shows a typical output from the initDB step

## Installing Apache and PHP, and Managing the WebServer

1. Ensure PostgreSQL is installed and running, type “service postgresql status” to report the current status of the postgreSQL service
2. yum install –y phpPgAdmin # this will install phpPgAdmin, php itself and the Apache webserver as well as many required library packages.
3. Once installed, the httpd service (i.e. Apache) must be started in order to use the management console. Type “service httpd start” to start it. You can verify it is running through observing the output of “service httpd status”.
4. Go to the URL “(instance ip)/phpPgAdmin”. You should be presented with the management console.
5. Click on the left “PostgreSQL” and log in with the DB user you set up previously.
6. The database should be listed, expand it and proceed to: Schemas > public > Tables categories. Tables can be managed here for the web app.

## Backup/Restore Test Data

1. Click Export on the top-right of the phpPgAdmin management console.
2. Select Structure and data, Format : SQL and the Download option.
3. Click Export, and save the dump file somewhere safe.
4. To restore, DROP THE DATABASE FIRST (bak-1) and re-create, select the database from the tree controller on the left, click the SQL tab at the top, then “Choose file” and then locate and open the dump file you saved. (bak-2)
5. Not only will test data be restored, but the whole database including tables, their structure, foreign keys and so on.

CI6300 – INDIVIDUAL PROJECT

# Uni-Hood Report

## Appendix B: Screenshots

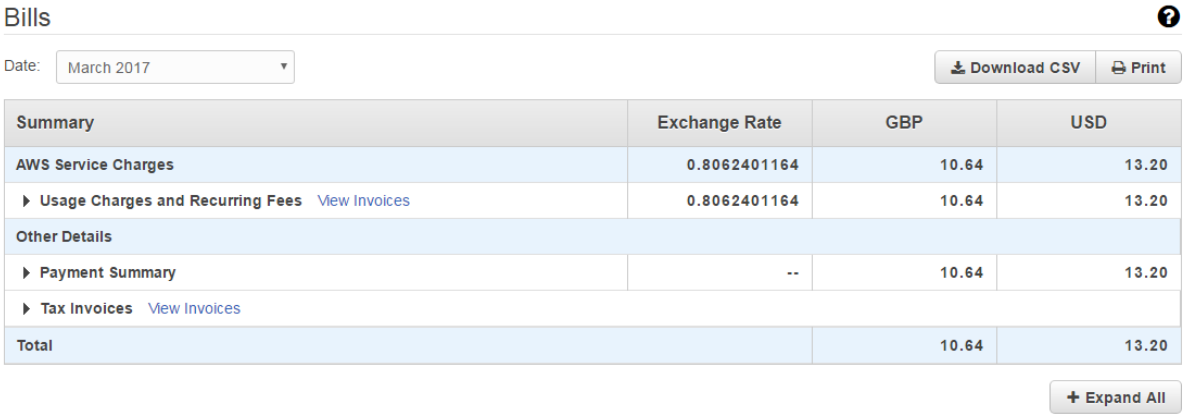
**Daniel Carnovale**

**23/04/2017**

# Contents

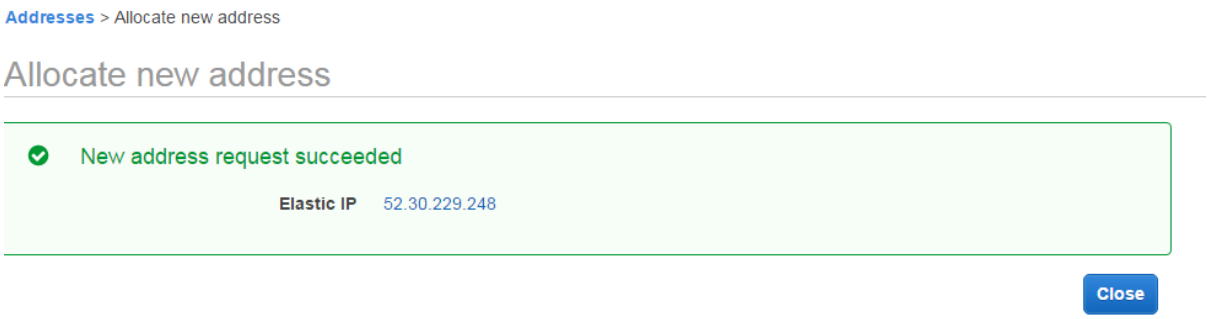
# Amazon Billing

## bil-1

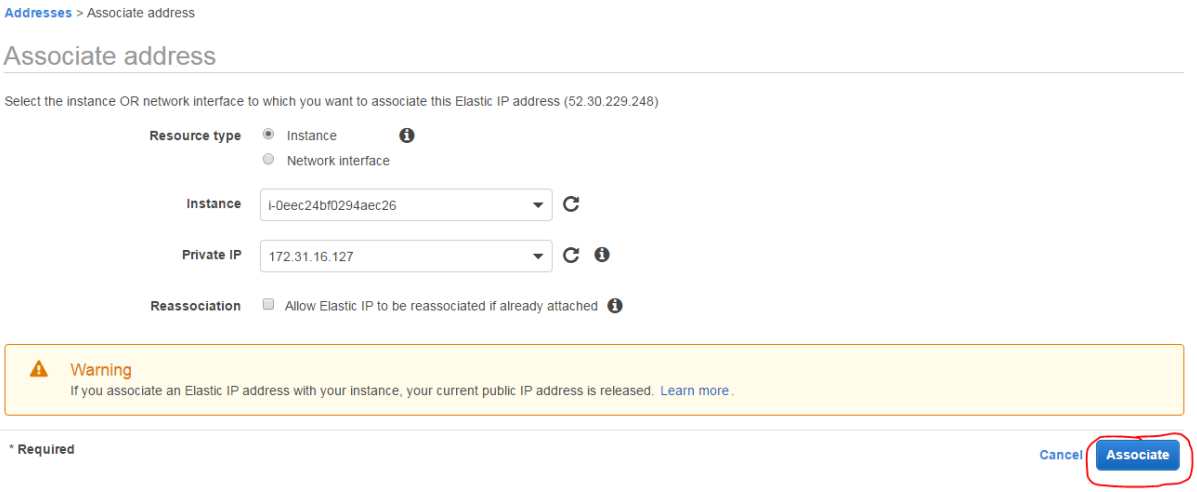


# Creating and Associating Elastic IP to Instance

## ela-1

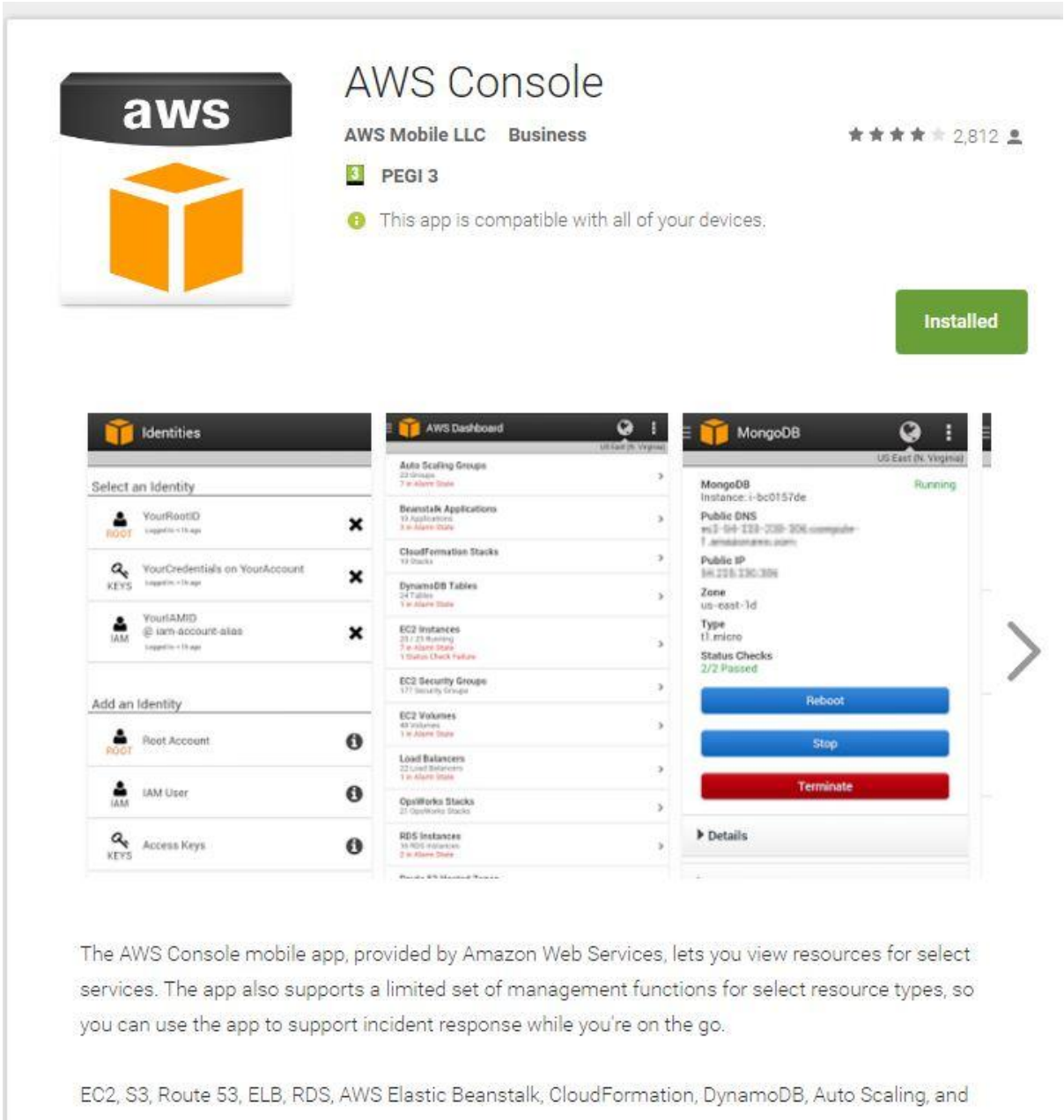



## ela-2

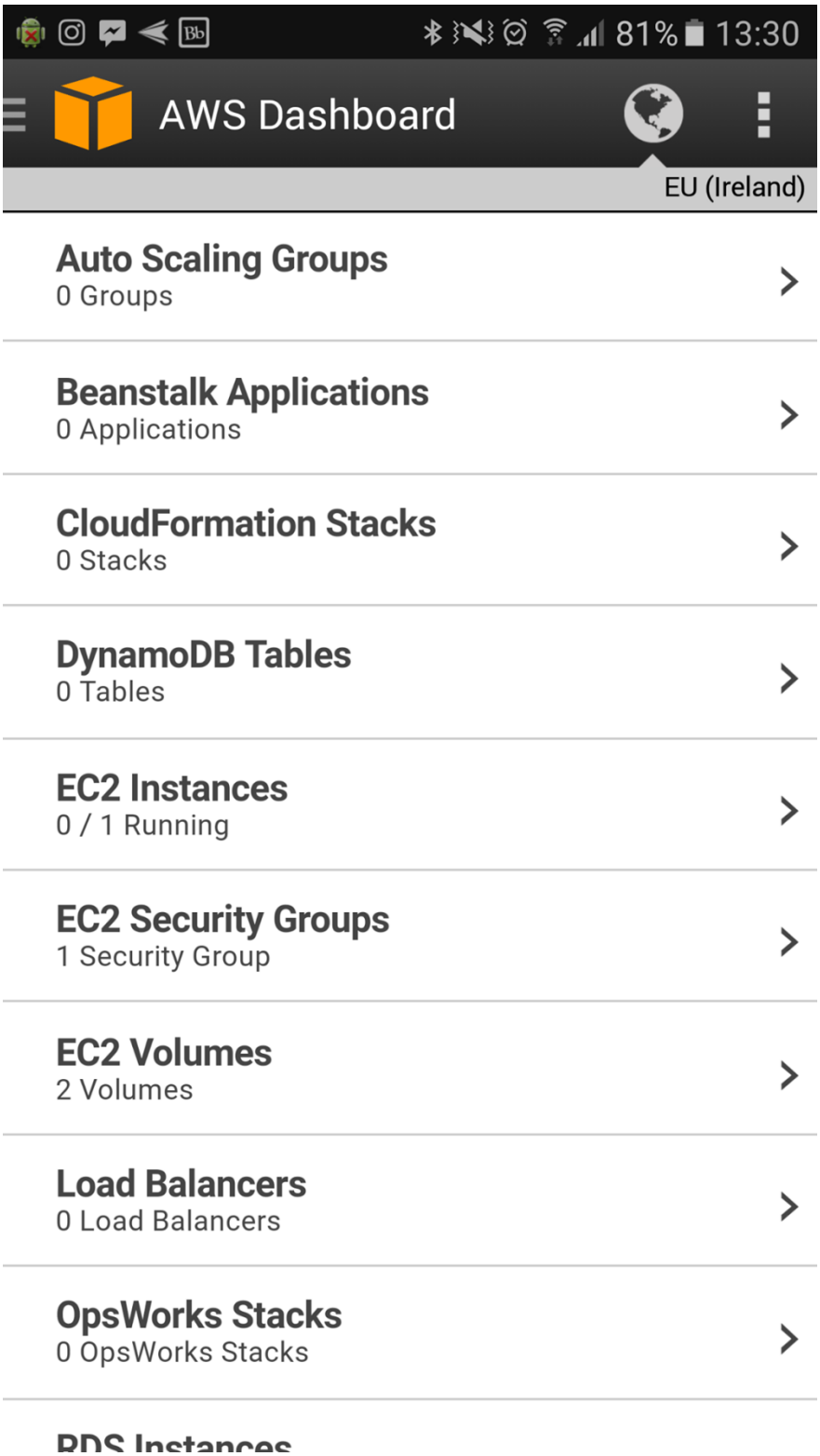


# AWS Console for Mobile

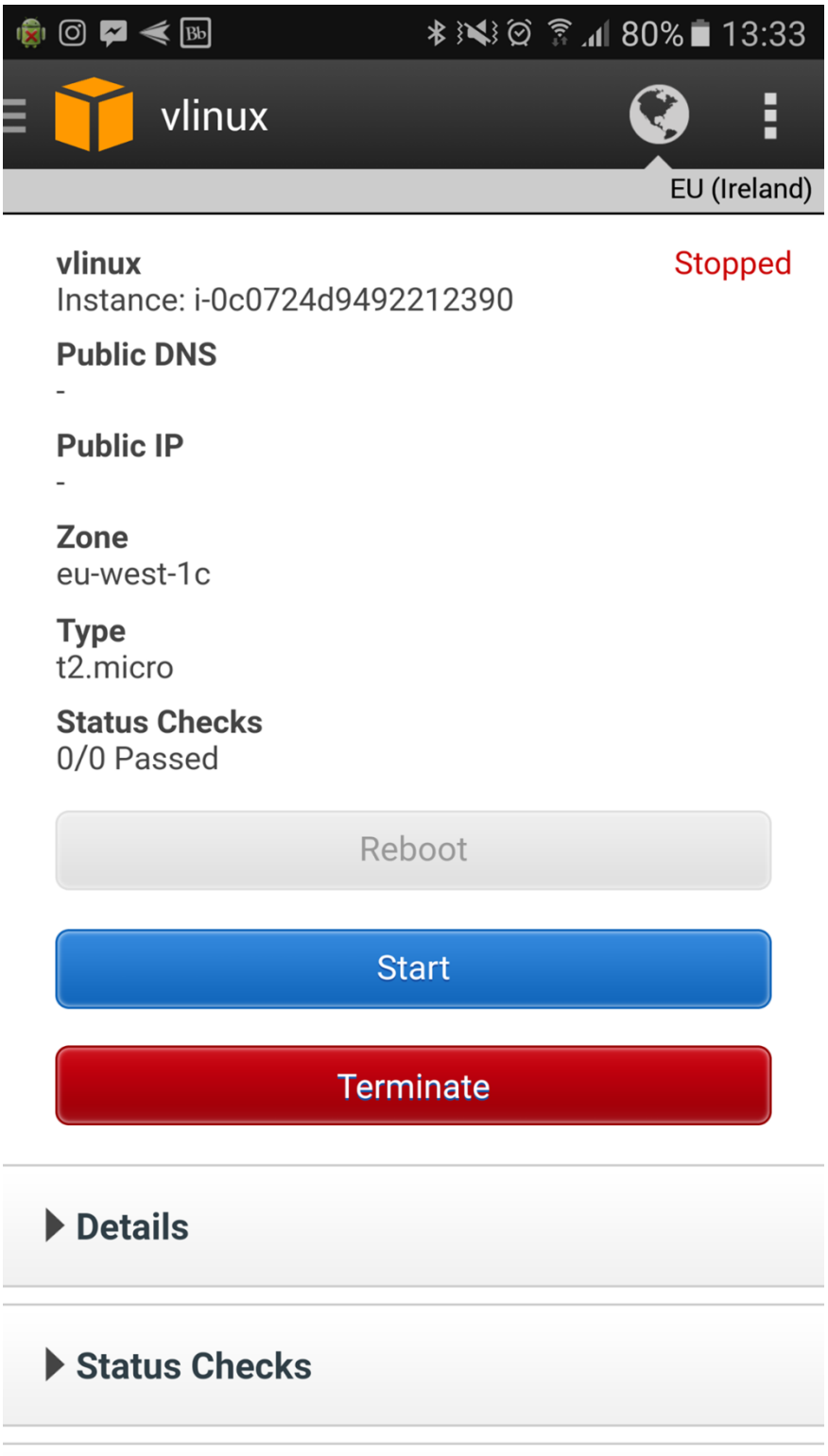
## aws-1



## aws-2

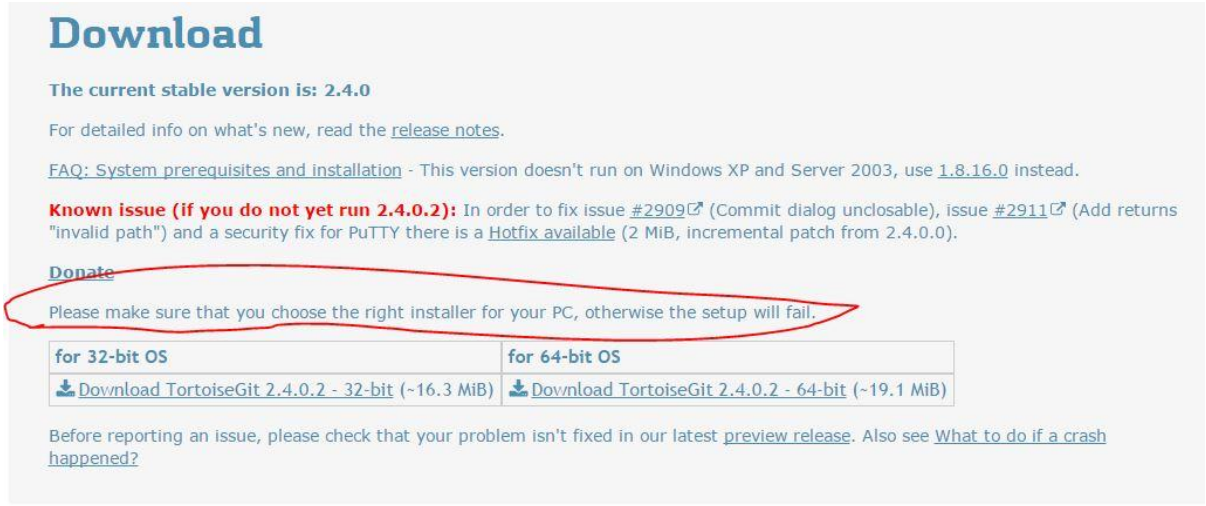

81% 13:30
AWS Dashboard
EU (Ireland)
Auto Scaling Groups
0 Groups
Beanstalk Applications
0 Applications
CloudFormation Stacks
0 Stacks
DynamoDB Tables
0 Tables
EC2 Instances
0 / 1 Running
EC2 Security Groups
1 Security Group
EC2 Volumes
2 Volumes
Load Balancers
0 Load Balancers
OpsWorks Stacks
0 OpsWorks Stacks
RDS Instances

## aws-3


80% 13:33
vlinux
EU (Ireland)
vlinux
Stopped
Instance: i-0c0724d9492212390
Public DNS
-
Public IP
-
Zone
eu-west-1c
Type
t2.micro
Status Checks
0/0 Passed
Reboot
Start
Terminate
Details
Status Checks

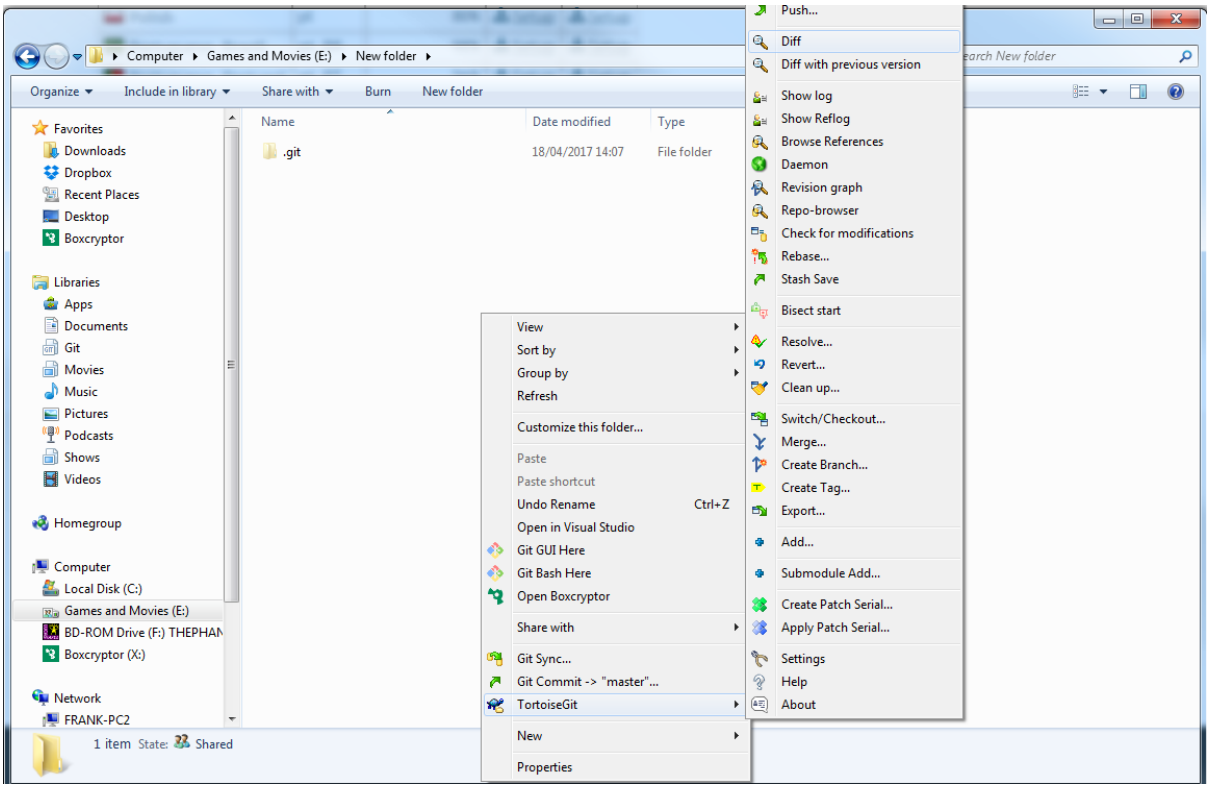
# TortoiseGit for Windows

## tor-1

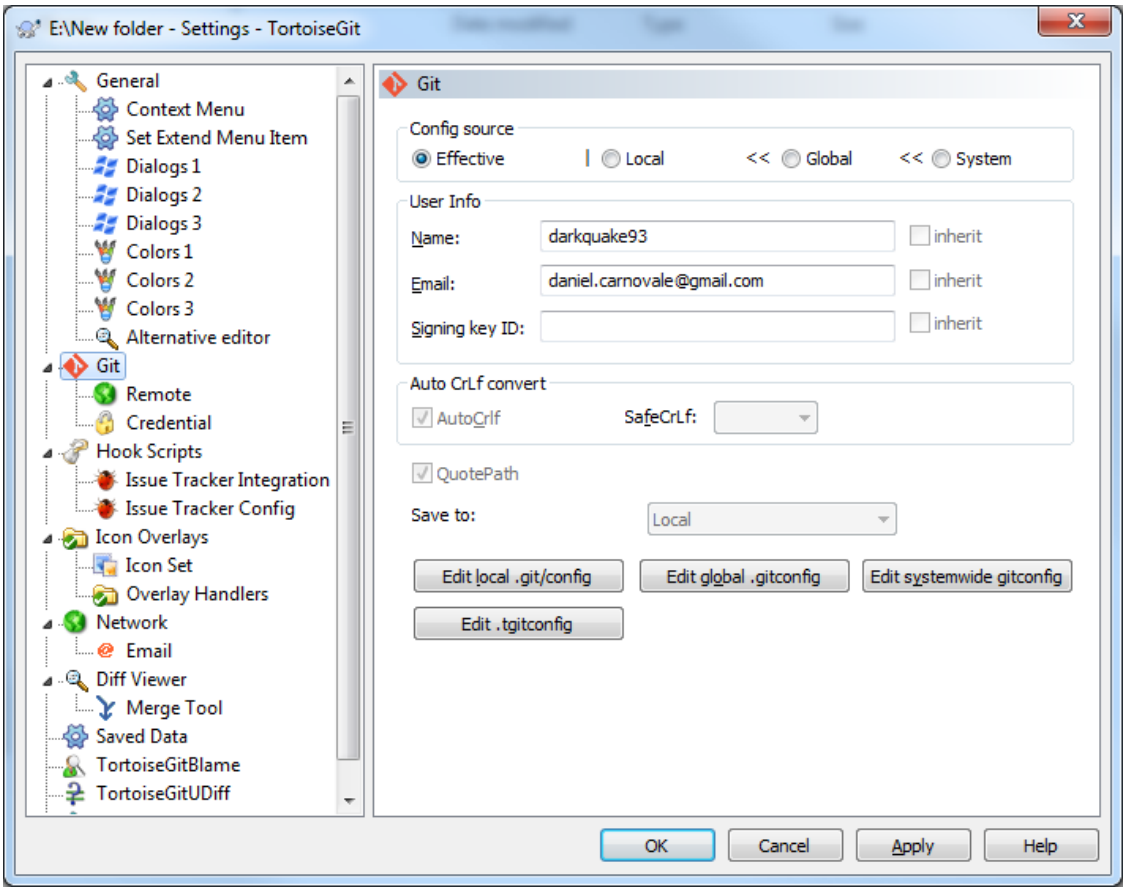



## tor-2

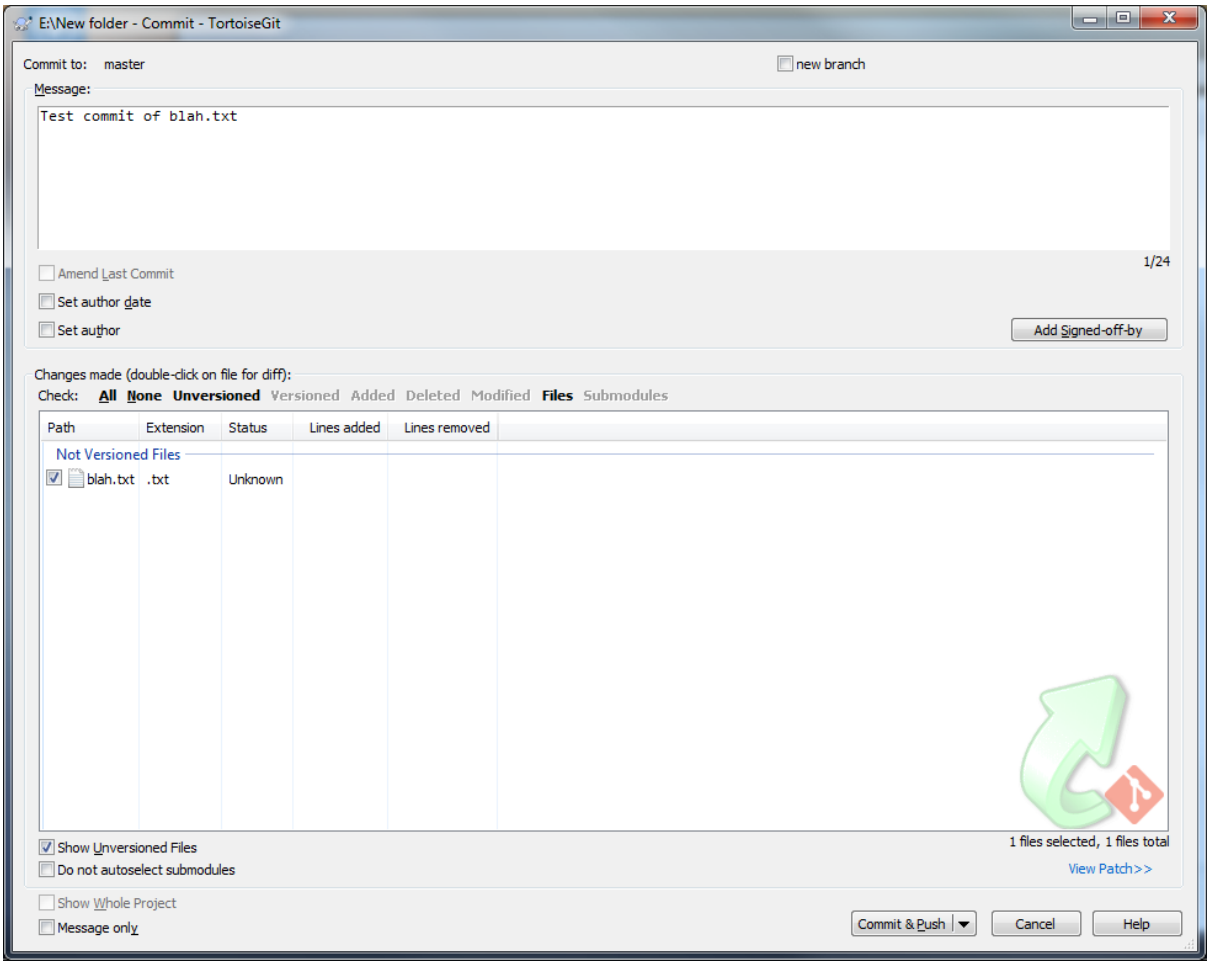
## tor-3


Computer ▸ Games and Movies (E:) ▸ New folder ▸
Organize
Include in library
Share with
Burn
New folder
Favorites
Downloads
Dropbox
Recent Places
Desktop
Boxcryptor
Libraries
Apps
Documents
Git
Movies
Music
Pictures
Podcasts
Shows
Videos
Homegroup
Computer
Local Disk (C:)
Games and Movies (E:)
BD-ROM Drive (F:) THEPHAN
Boxcryptor (X:)
Network
FRANK-PC2
Name
Date modified
Type
.git
18/04/2017 14:07
File folder
1 item State: Shared
View
Sort by
Group by
Refresh
Customize this folder...
Paste
Paste shortcut
Undo Rename Ctrl+Z
Open in Visual Studio
Git GUI Here
Git Bash Here
Open Boxcryptor
Share with
Git Sync...
Git Commit -> "master"...
TortoiseGit
New
Properties
Push...
Diff
Diff with previous version
Show log
Show Reflog
Browse References
Daemon
Revision graph
Repo-browser
Check for modifications
Rebase...
Stash Save
Bisect start
Resolve...
Revert...
Clean up...
Switch/Checkout...
Merge...
Create Branch...
Create Tag...
Export...
Add...
Submodule Add...
Create Patch Serial...
Apply Patch Serial...
Settings
Help
About

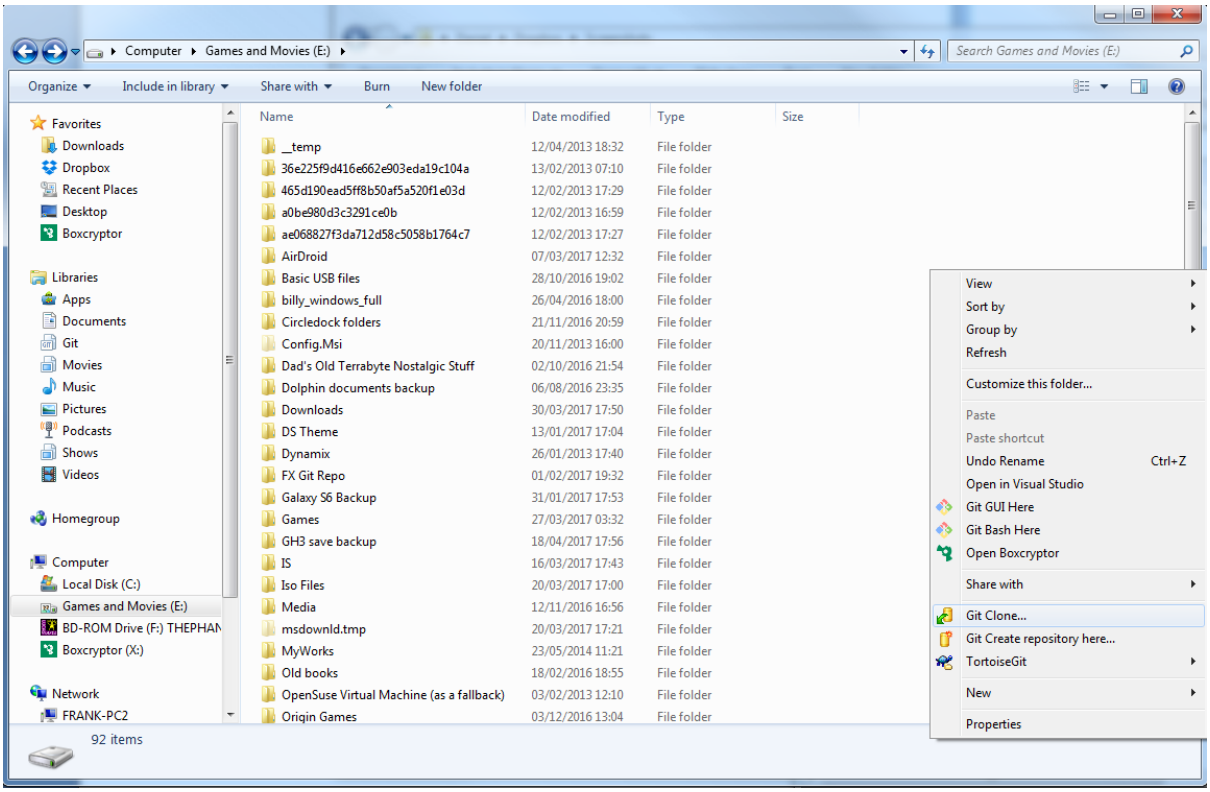
## tor-4


E:\New folder - Settings - TortoiseGit
General
Context Menu
Set Extend Menu Item
Dialogs 1
Dialogs 2
Dialogs 3
Colors 1
Colors 2
Colors 3
Alternative editor
Git
Remote
Credential
Hook Scripts
Issue Tracker Integration
Issue Tracker Config
Icon Overlays
Icon Set
Overlay Handlers
Network
Email
Diff Viewer
Merge Tool
Saved Data
TortoiseGitBlame
TortoiseGitUDiff
Git
Config source
Effective
Local
<< Global
<< System
User Info
Name:
darkquake93
inherit
Email:
daniel.carnovale@gmail.com
inherit
Signing key ID:
inherit
Auto CrLf convert
AutoCrlf
SafeCrLf:
QuotePath
Save to:
Local
Edit local .git/config
Edit global .gitconfig
Edit systemwide gitconfig
Edit .tgitconfig
OK
Cancel
Apply
Help

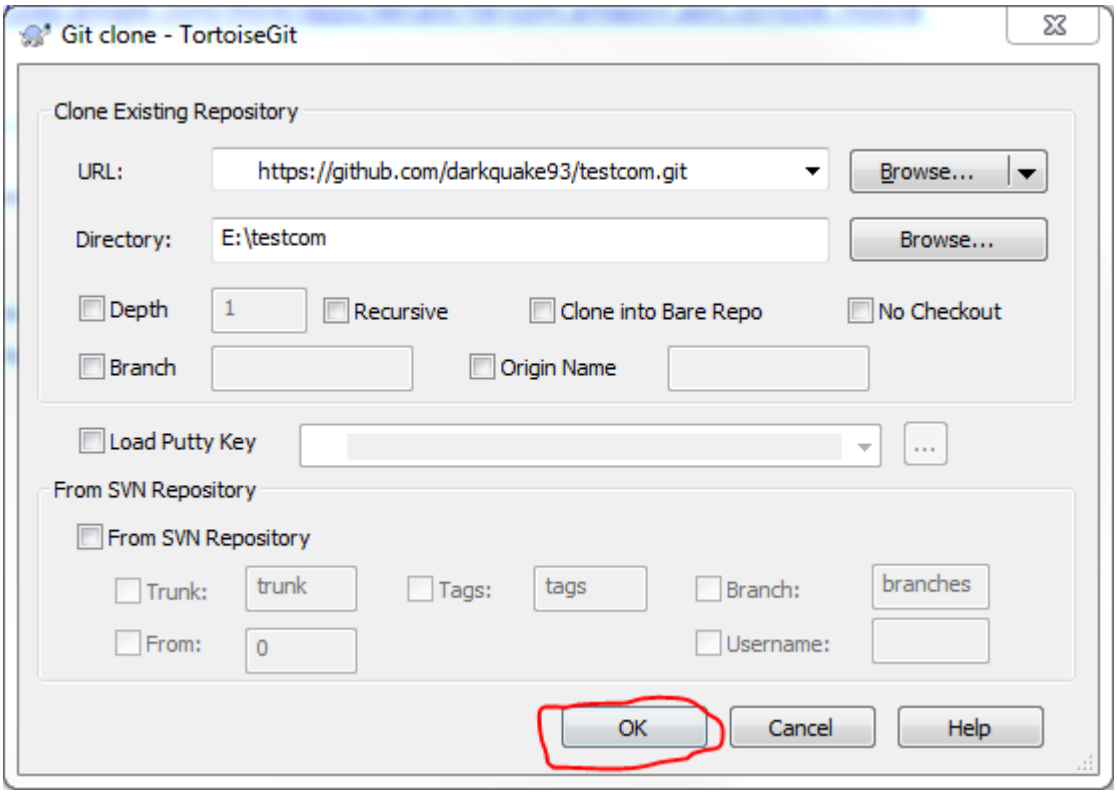
## tor-5


E:\New folder - Commit - TortoiseGit
Commit to: master
new branch
Message:
Test commit of blah.txt
1/24
Amend Last Commit
Set author date
Set author
Add Signed-off-by
Changes made (double-click on file for diff):
Check: All None Unversioned Versioned Added Deleted Modified Files Submodules
Path
Extension
Status
Lines added
Lines removed
Not Versioned Files
blah.txt
.txt
Unknown
1 files selected, 1 files total
Show Unversioned Files
Do not autoselect submodules
View Patch>>
Show Whole Project
Message only
Commit & Push
Cancel
Help

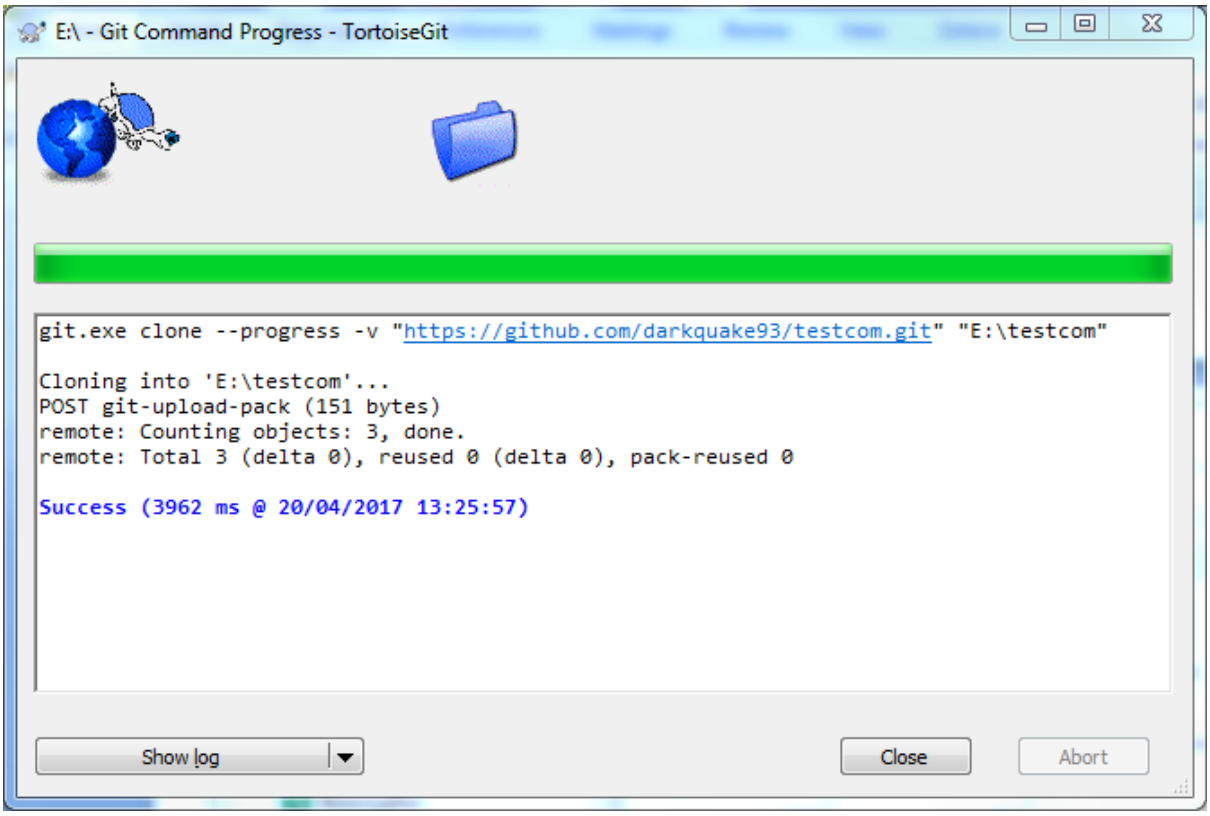
## tor-6


Computer ▸ Games and Movies (E:) ▸
Search Games and Movies (E:)
Organize
Include in library
Share with
Burn
New folder
Favorites
Downloads
Dropbox
Recent Places
Desktop
Boxcryptor
Libraries
Apps
Documents
Git
Movies
Music
Pictures
Podcasts
Shows
Videos
Homegroup
Computer
Local Disk (C:)
Games and Movies (E:)
BD-ROM Drive (F:) THEPHAN
Boxcryptor (X:)
Network
FRANK-PC2
Name
Date modified
Type
Size
__temp 12/04/2013 18:32 File folder
36e225f9d416e662e903eda19c104a 13/02/2013 07:10 File folder
465d190ead5ff8b50af5a520f1e03d 12/02/2013 17:29 File folder
a0be980d3c3291ce0b 12/02/2013 16:59 File folder
ae068827f3da712d58c5058b1764c7 12/02/2013 17:27 File folder
AirDroid 07/03/2017 12:32 File folder
Basic USB files 28/10/2016 19:02 File folder
billy_windows_full 26/04/2016 18:00 File folder
Circledock folders 21/11/2016 20:59 File folder
Config.Msi 20/11/2013 16:00 File folder
Dad's Old Terrabyte Nostalgic Stuff 02/10/2016 21:54 File folder
Dolphin documents backup 06/08/2016 23:35 File folder
Downloads 30/03/2017 17:50 File folder
DS Theme 13/01/2017 17:04 File folder
Dynamix 26/01/2013 17:40 File folder
FX Git Repo 01/02/2017 19:32 File folder
Galaxy S6 Backup 31/01/2017 17:53 File folder
Games 27/03/2017 03:32 File folder
GH3 save backup 18/04/2017 17:56 File folder
IS 16/03/2017 17:43 File folder
Iso Files 20/03/2017 17:00 File folder
Media 12/11/2016 16:56 File folder
msdownld.tmp 20/03/2017 17:21 File folder
MyWorks 23/05/2014 11:21 File folder
Old books 18/02/2016 18:55 File folder
OpenSuse Virtual Machine (as a fallback) 03/02/2013 12:10 File folder
Origin Games 03/12/2016 13:04 File folder
92 items
View
Sort by
Group by
Refresh
Customize this folder...
Paste
Paste shortcut
Undo Rename Ctrl+Z
Open in Visual Studio
Git GUI Here
Git Bash Here
Open Boxcryptor
Share with
Git Clone...
Git Create repository here...
TortoiseGit
New
Properties

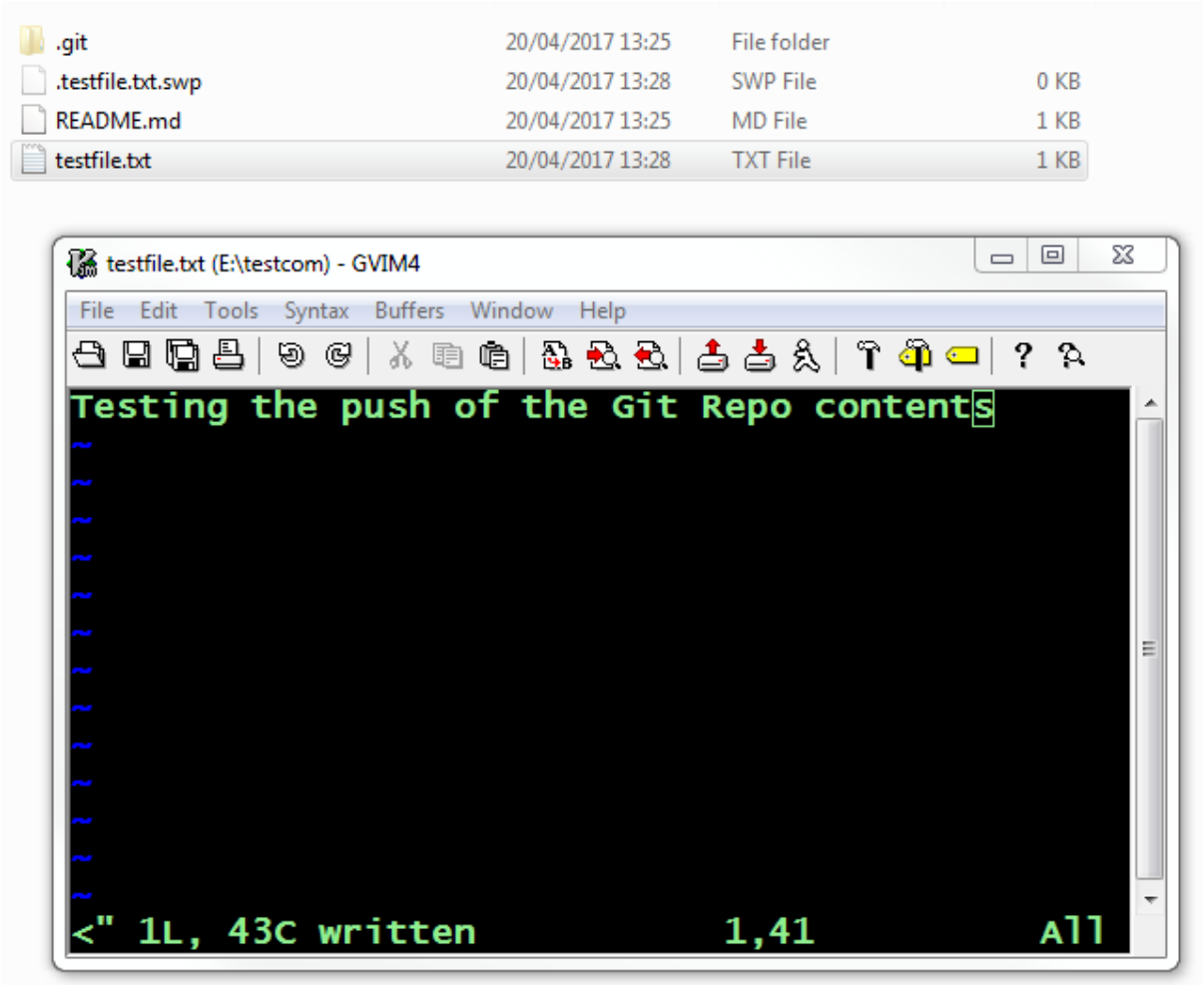
## tor-7


Git clone - TortoiseGit
Clone Existing Repository
URL:
https://github.com/darkquake93/testcom.git
Browse...
Directory:
E:\testcom
Browse...
Depth
1
Recursive
Clone into Bare Repo
No Checkout
Branch
Origin Name
Load Putty Key
From SVN Repository
From SVN Repository
Trunk:
trunk
Tags:
tags
Branch:
branches
From:
0
Username:
OK
Cancel
Help

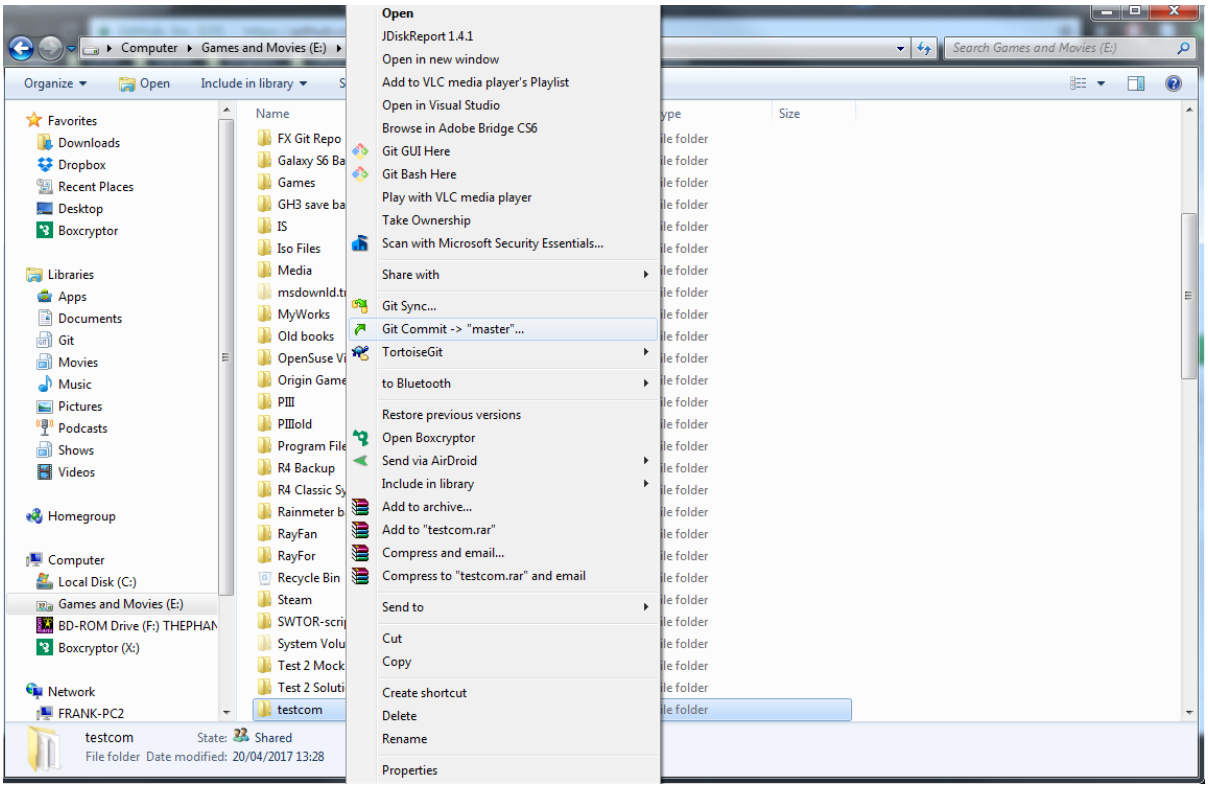
## tor-8


E:\ - Git Command Progress - TortoiseGit
git.exe clone --progress -v "https://github.com/darkquake93/testcom.git" "E:\testcom"
Cloning into 'E:\testcom'...
POST git-upload-pack (151 bytes)
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Success (3962 ms @ 20/04/2017 13:25:57)
Show log
Close
Abort

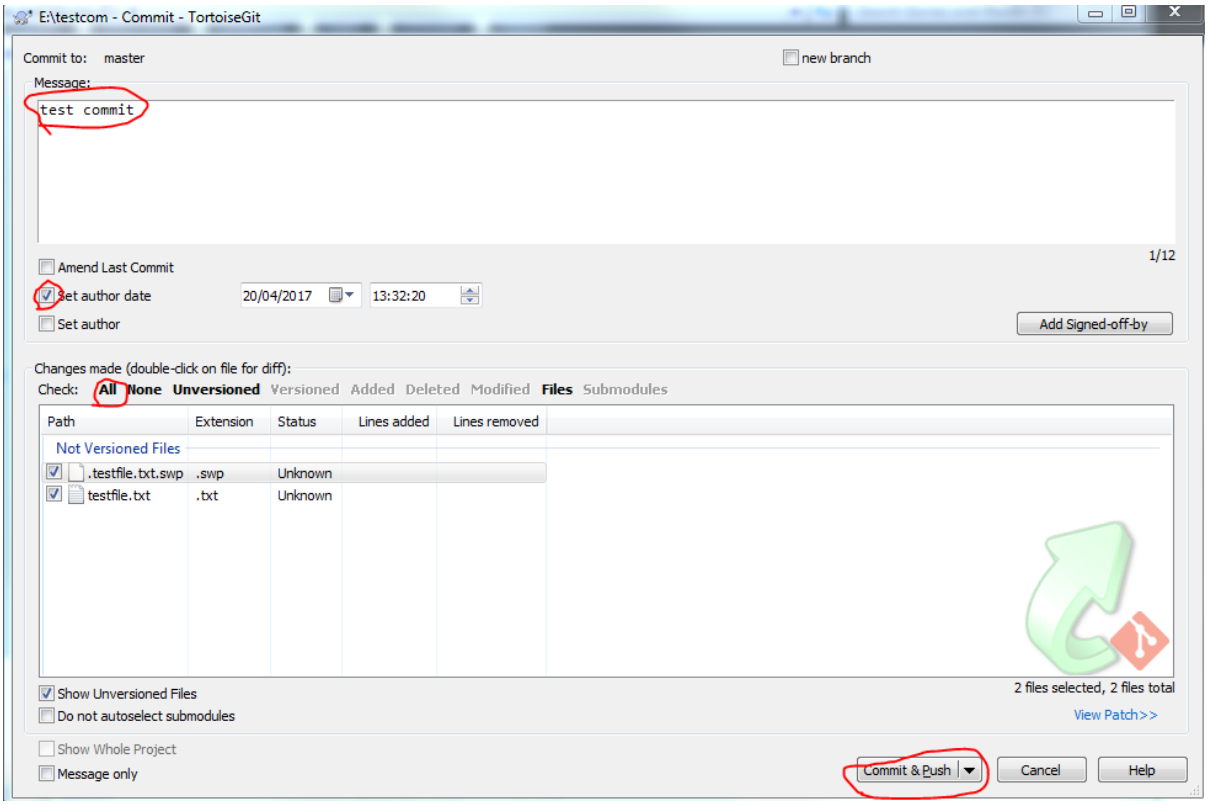
## tor-9


.git 20/04/2017 13:25 File folder
.testfile.txt.swp 20/04/2017 13:28 SWP File 0 KB
README.md 20/04/2017 13:25 MD File 1 KB
testfile.txt 20/04/2017 13:28 TXT File 1 KB
testfile.txt (E:\testcom) - GVIM4
File Edit Tools Syntax Buffers Window Help
Testing the push of the Git Repo contents
<" 1L, 43C written 1,41 All

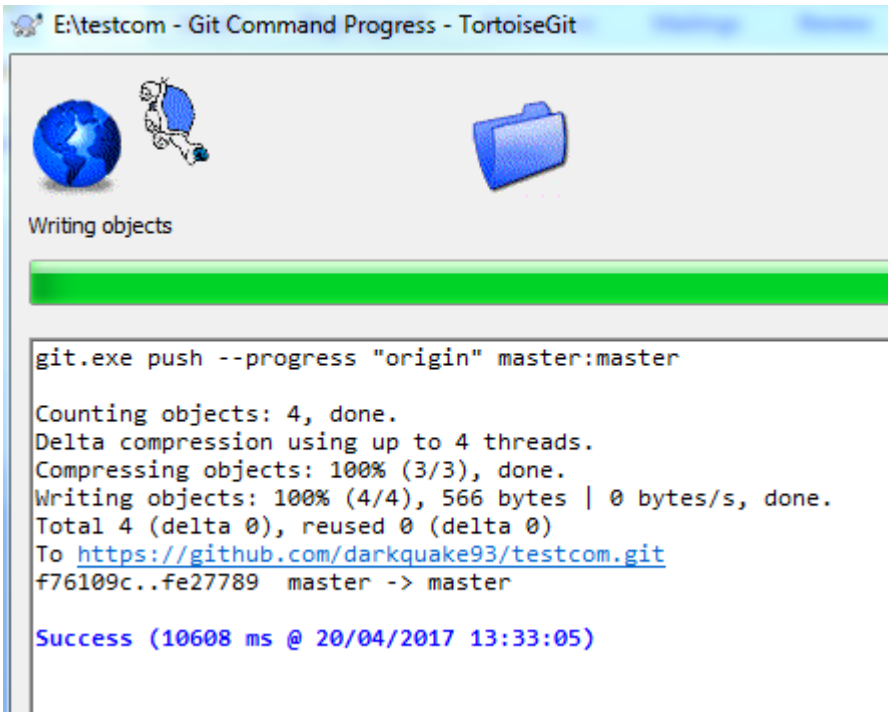
## tor-10


Computer ▸ Games and Movies (E:) ▸
Organize
Open
Include in library
Favorites
Downloads
Dropbox
Recent Places
Desktop
Boxcryptor
Libraries
Apps
Documents
Git
Movies
Music
Pictures
Podcasts
Shows
Videos
Homegroup
Computer
Local Disk (C:)
Games and Movies (E:)
BD-ROM Drive (F:) THEPHAN
Boxcryptor (X:)
Network
FRANK-PC2
Name
FX Git Repo
Galaxy S6 Ba
Games
GH3 save ba
IS
Iso Files
Media
MyWorks
Old books
Origin Game
PIII
PIIIold
R4 Backup
RayFan
RayFor
Recycle Bin
Steam
System Volu
Test 2 Mock
Test 2 Soluti
testcom
testcom State: Shared
File folder Date modified: 20/04/2017 13:28
Open
JDiskReport 1.4.1
Open in new window
Add to VLC media player's Playlist
Open in Visual Studio
Browse in Adobe Bridge CS6
Git GUI Here
Git Bash Here
Play with VLC media player
Take Ownership
Scan with Microsoft Security Essentials...
Share with
Git Sync...
Git Commit -> "master"...
TortoiseGit
to Bluetooth
Restore previous versions
Open Boxcryptor
Send via AirDroid
Include in library
Add to archive...
Add to "testcom.rar"
Compress and email...
Compress to "testcom.rar" and email
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
Search Games and Movies (E:)
Size
le folder

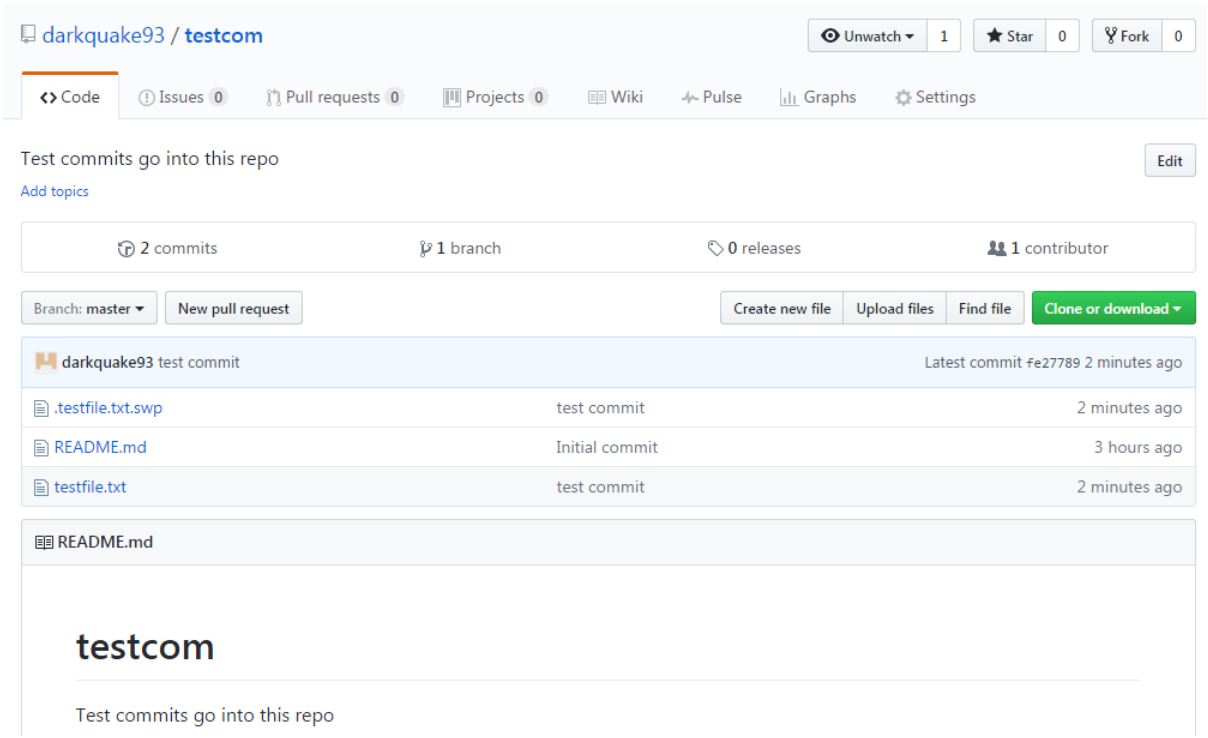
## tor-11


E:\testcom - Commit - TortoiseGit
Commit to: master
new branch
Message:
test commit
1/12
Amend Last Commit
Set author date
20/04/2017
13:32:20
Set author
Add Signed-off-by
Changes made (double-click on file for diff):
Check: All None Unversioned Versioned Added Deleted Modified Files Submodules
Path
Extension
Status
Lines added
Lines removed
Not Versioned Files
.testfile.txt.swp
.swp
Unknown
testfile.txt
.txt
Unknown
2 files selected, 2 files total
Show Unversioned Files
Do not autoselect submodules
View Patch>>
Show Whole Project
Message only
Commit & Push
Cancel
Help

## tor-12

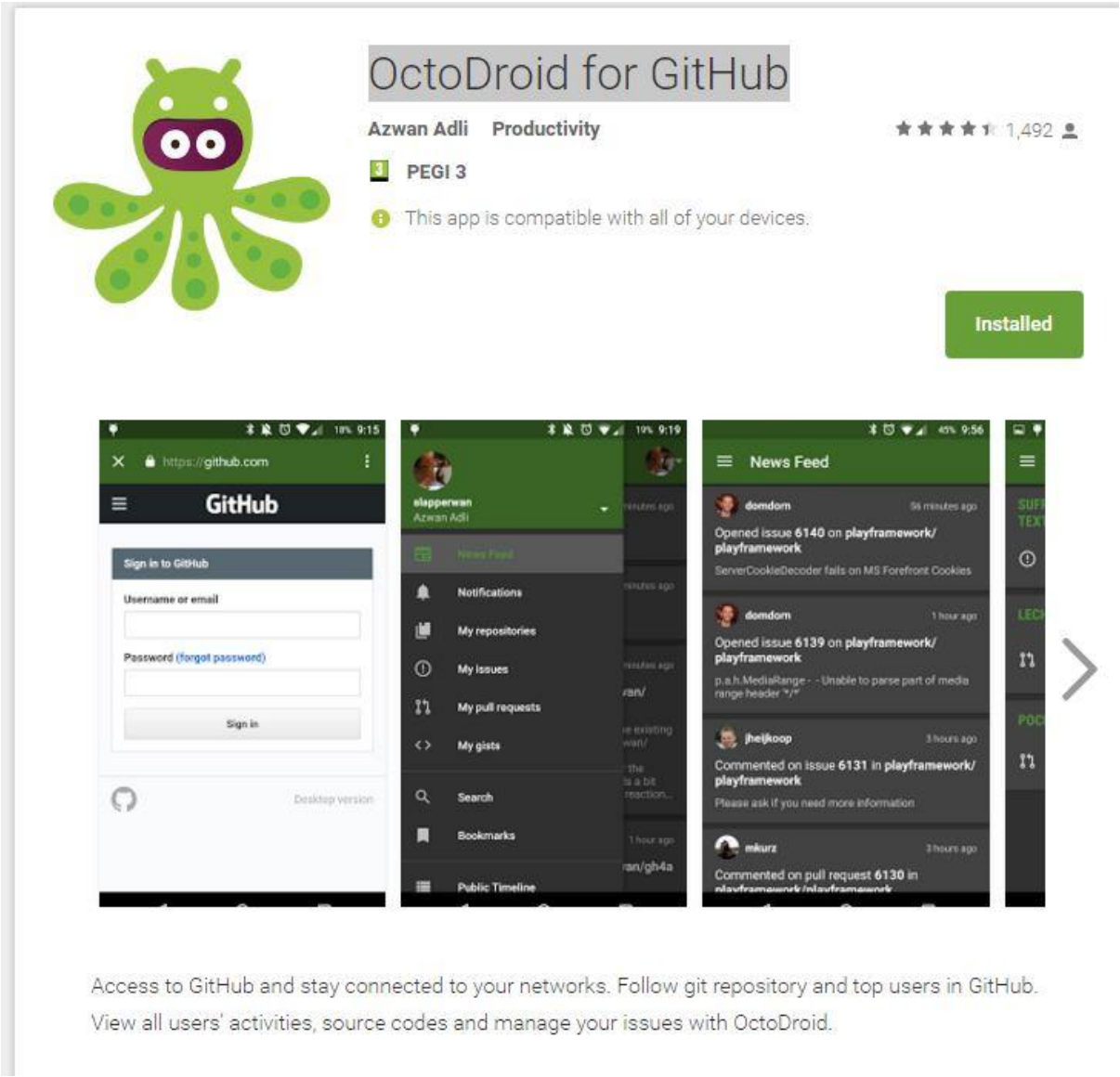

E:\testcom - Git Command Progress - TortoiseGit
Writing objects
git.exe push --progress "origin" master:master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 566 bytes | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To https://github.com/darkquake93/testcom.git
f76109c..fe27789 master -> master
Success (10608 ms @ 20/04/2017 13:33:05)

## tor-13

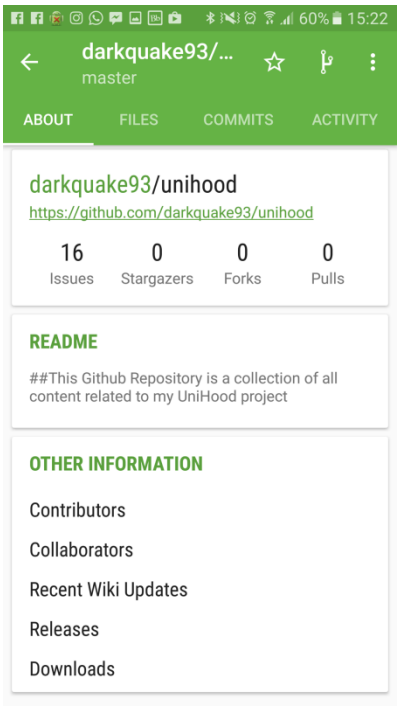

darkquake93 / testcom
Unwatch 1
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Pulse
Graphs
Settings
Test commits go into this repo
Edit
Add topics
2 commits
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download
darkquake93 test commit
Latest commit fe27789 2 minutes ago
.testfile.txt.swp test commit 2 minutes ago
README.md Initial commit 3 hours ago
testfile.txt test commit 2 minutes ago
README.md
testcom
Test commits go into this repo

# OctoDroid

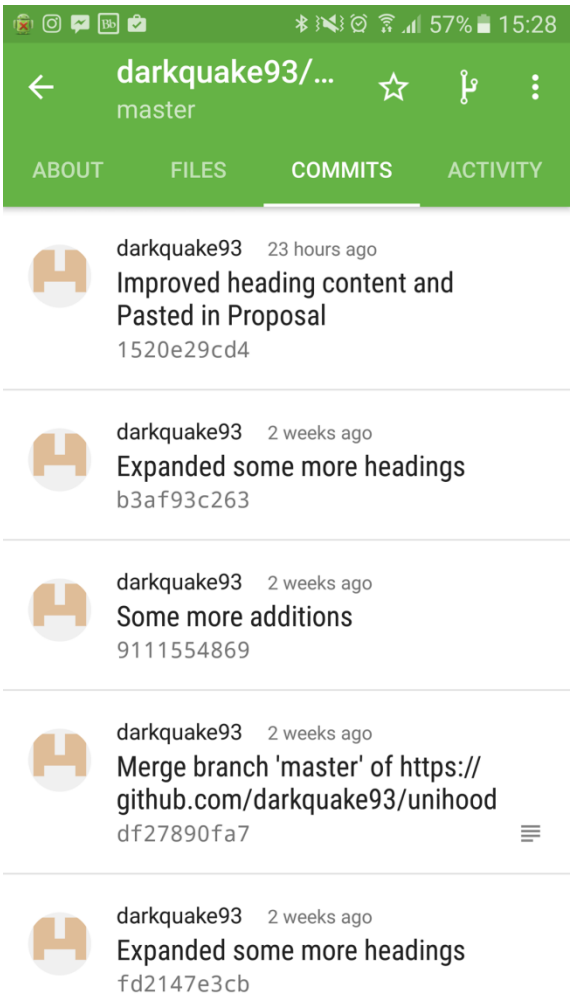
## oct-1



## oct-2

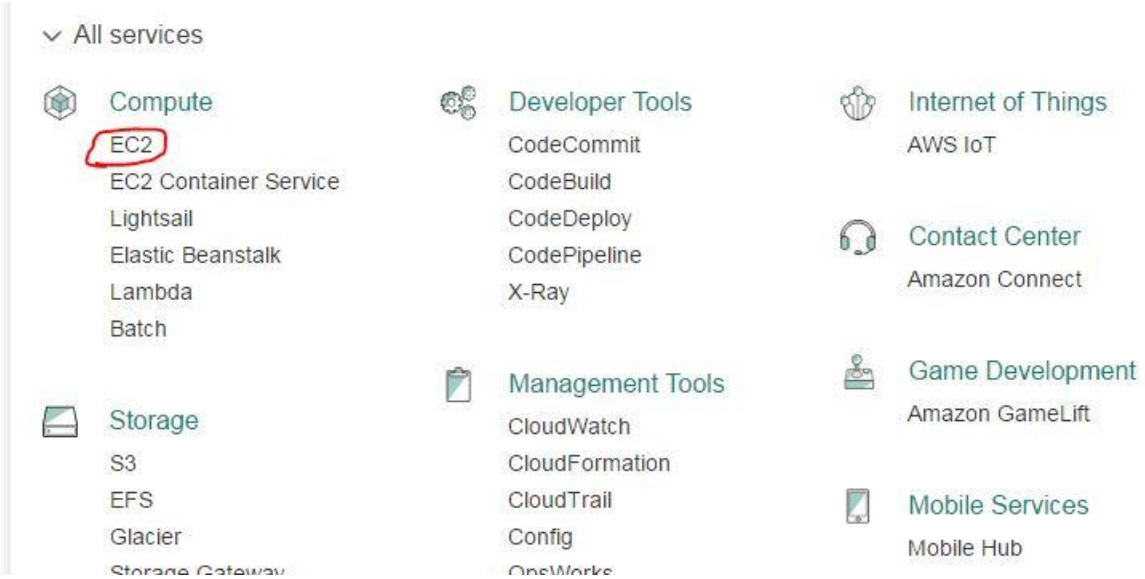

60% 15:22
darkquake93/...
master
ABOUT
FILES
COMMITS
ACTIVITY
darkquake93/unihood
https://github.com/darkquake93/unihood
16
Issues
0
Stargazers
0
Forks
0
Pulls
README
##This Github Repository is a collection of all content related to my UniHood project
OTHER INFORMATION
Contributors
Collaborators
Recent Wiki Updates
Releases
Downloads

## oct-3


57% 15:28
darkquake93/...
master
ABOUT
FILES
COMMITS
ACTIVITY
darkquake93 23 hours ago
Improved heading content and Pasted in Proposal
1520e29cd4
darkquake93 2 weeks ago
Expanded some more headings
b3af93c263
darkquake93 2 weeks ago
Some more additions
9111554869
darkquake93 2 weeks ago
Merge branch 'master' of https://github.com/darkquake93/unihood
df27890fa7
darkquake93 2 weeks ago
Expanded some more headings
fd2147e3cb

# SSH KeyPair Generation

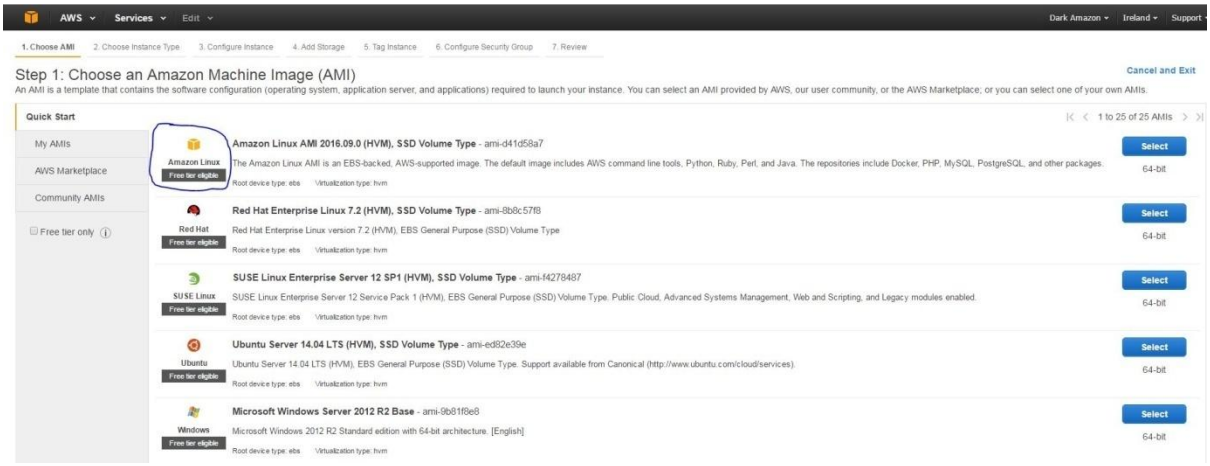
## key-1



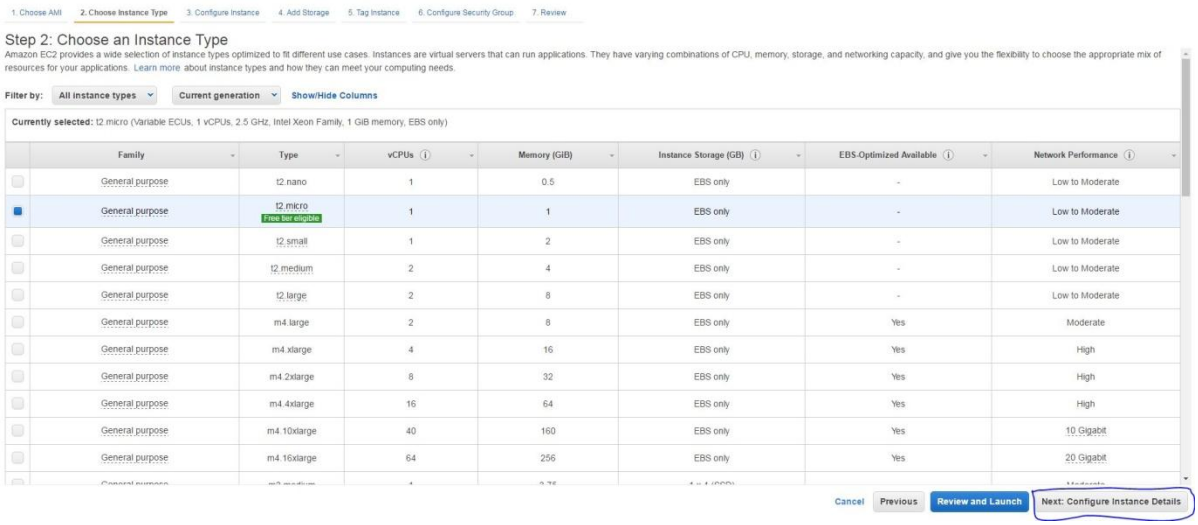

## key-2

# Amazon EC2 Instance Creation

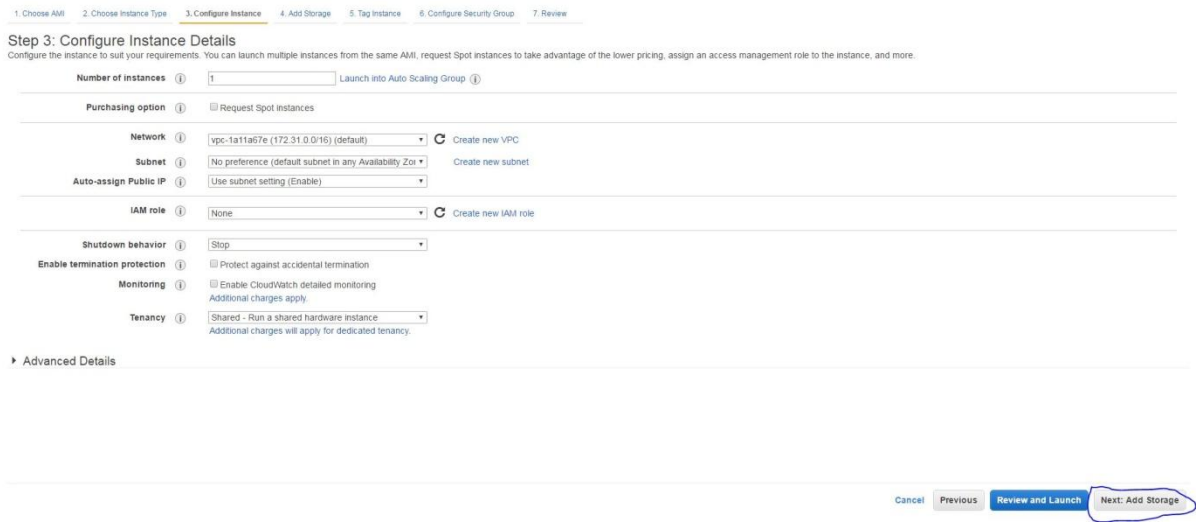
## ec2-1



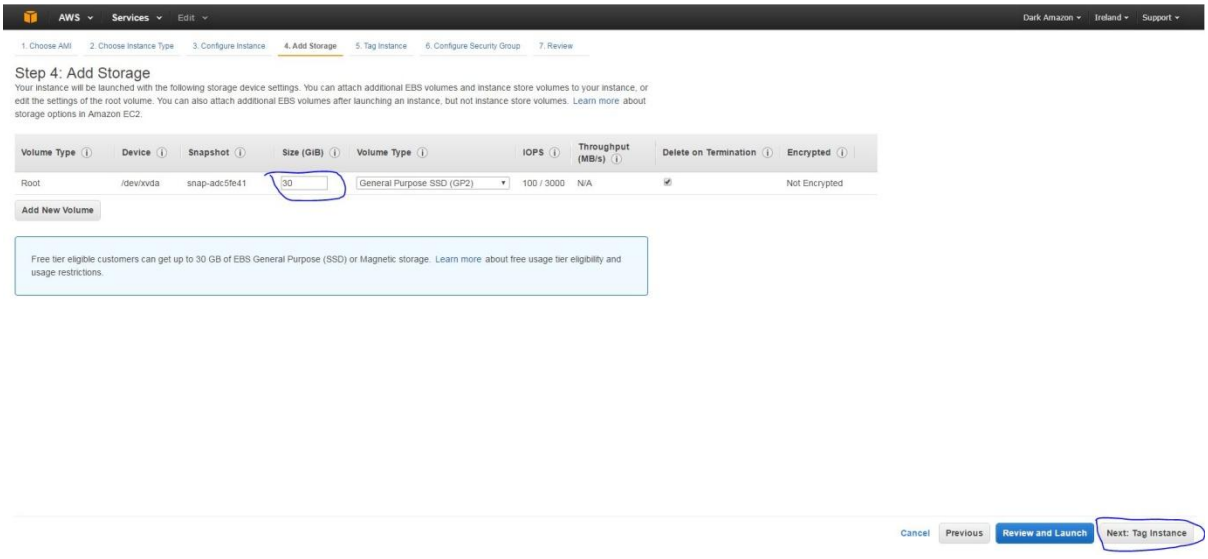

## ec2-2



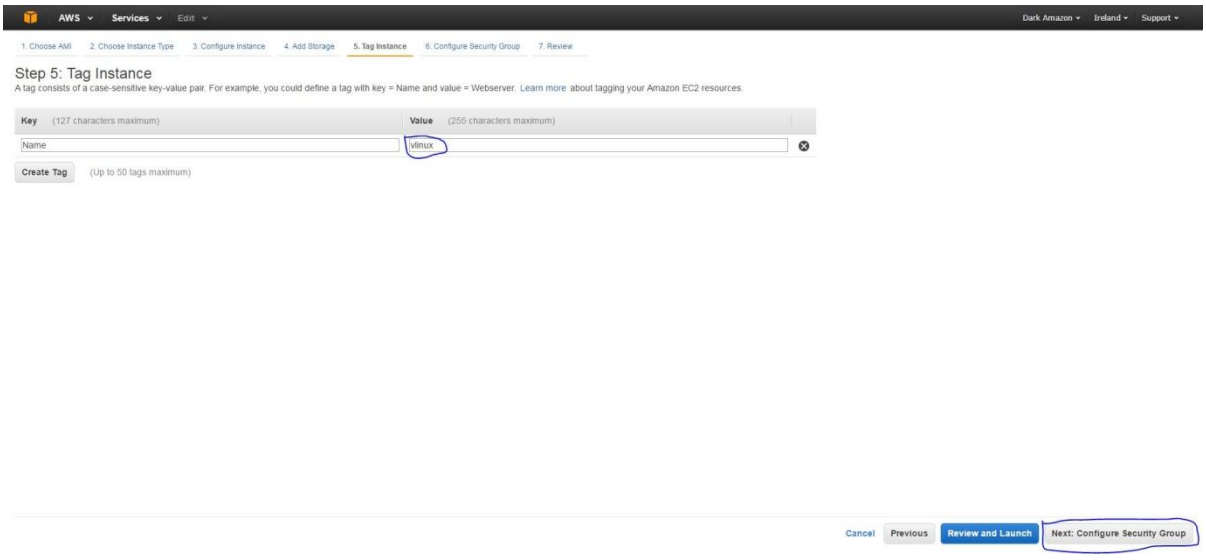
## ec2-3


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 3: Configure Instance Details
Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.
Number of instances
1
Launch into Auto Scaling Group
Purchasing option
Request Spot instances
Network
vpc-1a11a67e (172.31.0.0/16) (default)
Create new VPC
Subnet
No preference (default subnet in any Availability Zo
Create new subnet
Auto-assign Public IP
Use subnet setting (Enable)
IAM role
None
Create new IAM role
Shutdown behavior
Stop
Enable termination protection
Protect against accidental termination
Monitoring
Enable CloudWatch detailed monitoring
Additional charges apply.
Tenancy
Shared - Run a shared hardware instance
Additional charges will apply for dedicated tenancy.
Advanced Details
Cancel
Previous
Review and Launch
Next: Add Storage

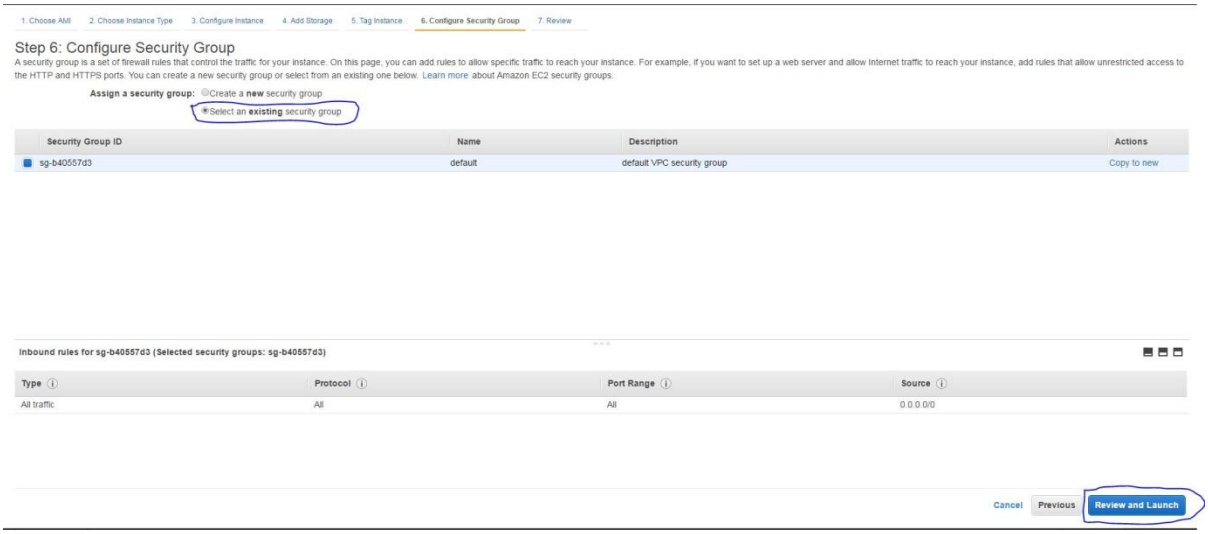
## ec2-4


AWS
Services
Edit
Dark Amazon
Ireland
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 4: Add Storage
Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.
Volume Type
Device
Snapshot
Size (GiB)
Volume Type
IOPS
Throughput (MB/s)
Delete on Termination
Encrypted
Root
/dev/xvda
snap-adc5fe41
30
General Purpose SSD (GP2)
100 / 3000
N/A
Not Encrypted
Add New Volume
Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.
Cancel
Previous
Review and Launch
Next: Tag Instance

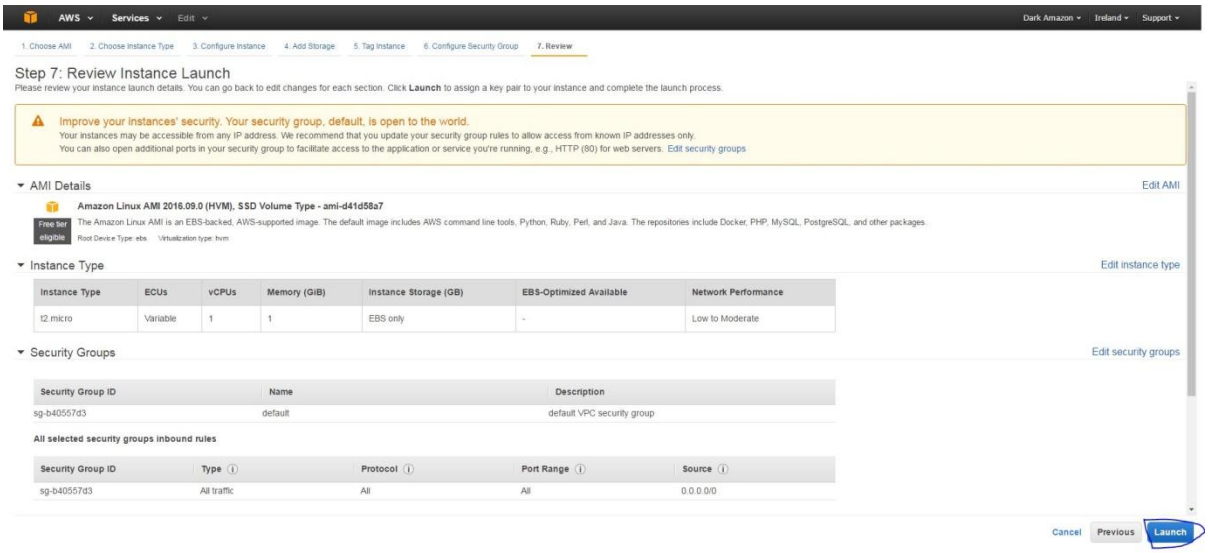
## ec2-5


AWS
Services
Edit
Dark Amazon
Ireland
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 5: Tag Instance
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.
Key (127 characters maximum)
Value (255 characters maximum)
Name
vlinux
Create Tag
(Up to 50 tags maximum)
Cancel
Previous
Review and Launch
Next: Configure Security Group

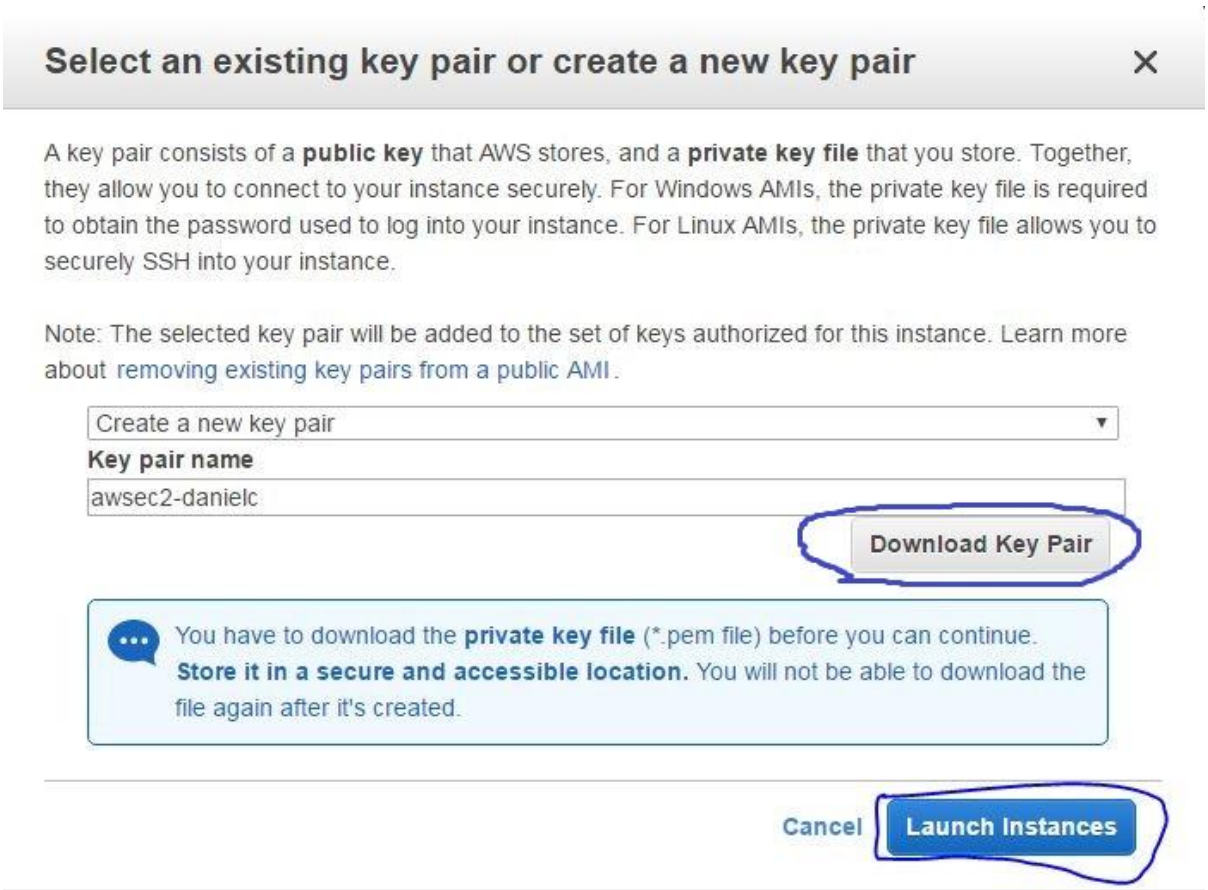
## ec2-6


1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 6: Configure Security Group
A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.
Assign a security group:
Create a new security group
Select an existing security group
Security Group ID
Name
Description
Actions
sg-b40557d3
default
default VPC security group
Copy to new
Inbound rules for sg-b40557d3 (Selected security groups: sg-b40557d3)
Type
Protocol
Port Range
Source
All traffic
All
All
0.0.0.0/0
Cancel
Previous
Review and Launch

## ec2-7

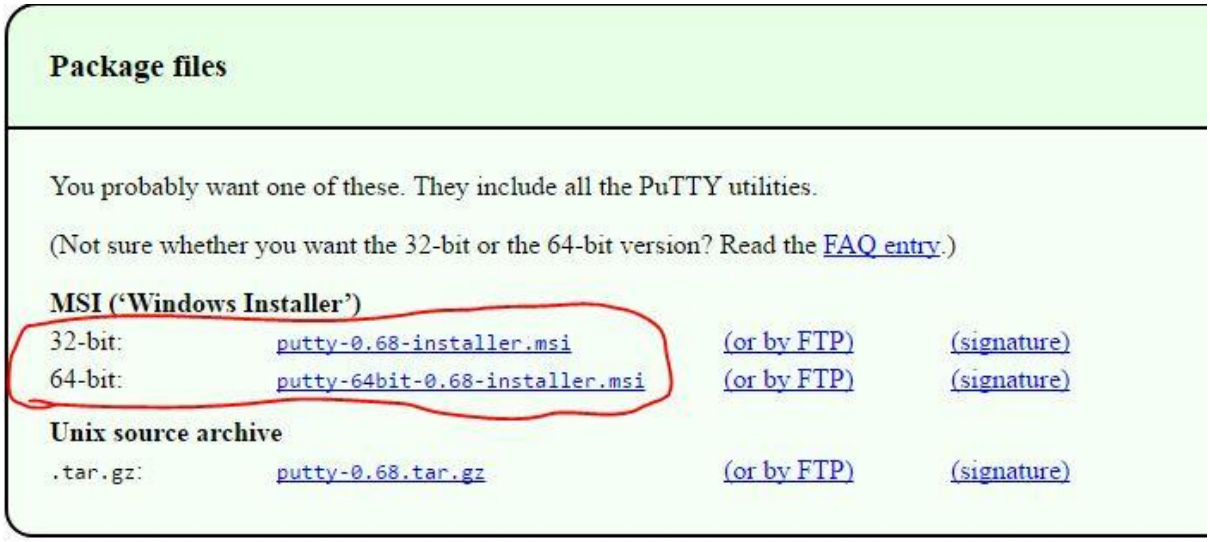

AWS
Services
Edit
Dark Amazon
Ireland
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Tag Instance
6. Configure Security Group
7. Review
Step 7: Review Instance Launch
Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.
Improve your instances' security. Your security group, default, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups
AMI Details
Edit AMI
Amazon Linux AMI 2016.09.0 (HVM), SSD Volume Type - ami-d41d58a7
Free tier eligible
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root Device Type: ebs Virtualization type: hvm
Instance Type
Edit instance type
Instance Type
ECUs
vCPUs
Memory (GiB)
Instance Storage (GB)
EBS-Optimized Available
Network Performance
t2.micro
Variable
1
1
EBS only
-
Low to Moderate
Security Groups
Edit security groups
Security Group ID
Name
Description
sg-b40557d3
default
default VPC security group
All selected security groups inbound rules
Security Group ID
Type
Protocol
Port Range
Source
sg-b40557d3
All traffic
All
All
0.0.0.0/0
Cancel
Previous
Launch

## ec2-8

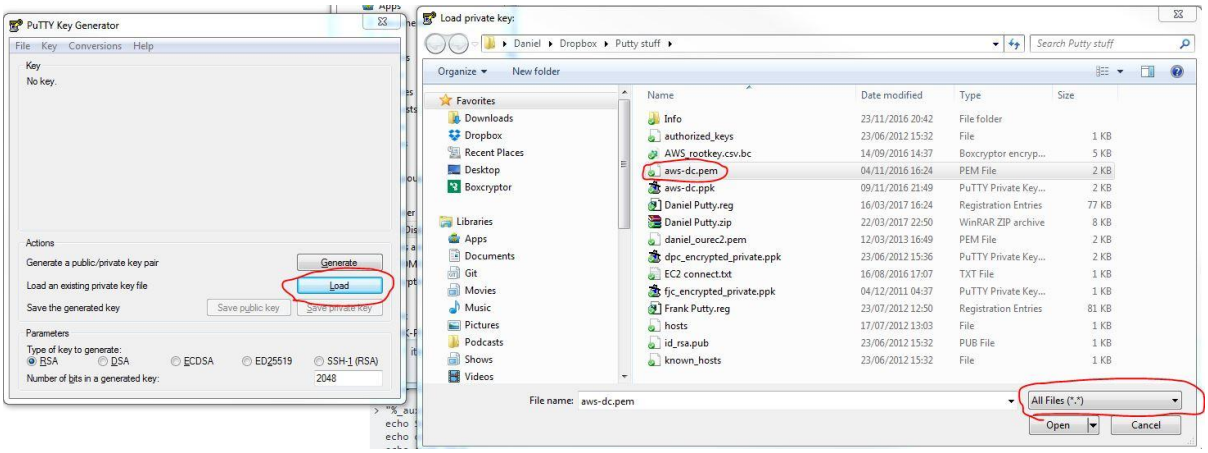

Select an existing key pair or create a new key pair
A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.
Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.
Create a new key pair
Key pair name
awsec2-danielc
Download Key Pair
You have to download the private key file (*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.
Cancel
Launch Instances

# Setting up Putty and Encrypted PPK File

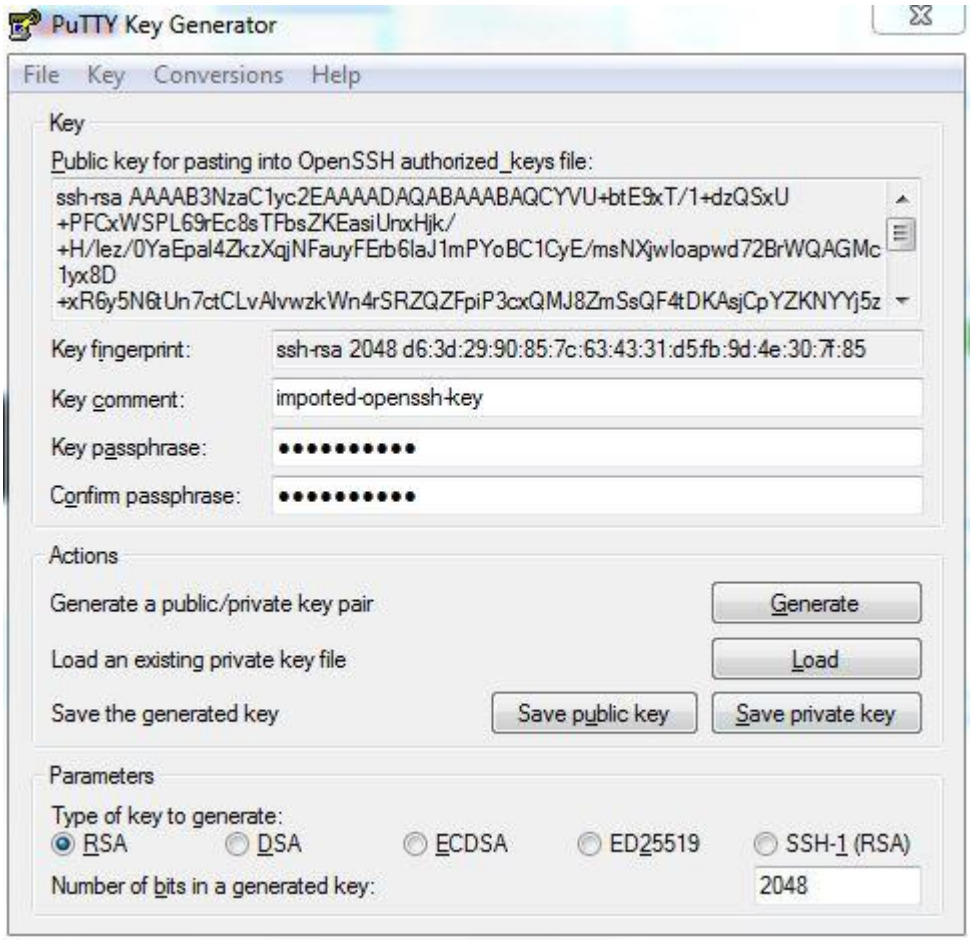
## put-1



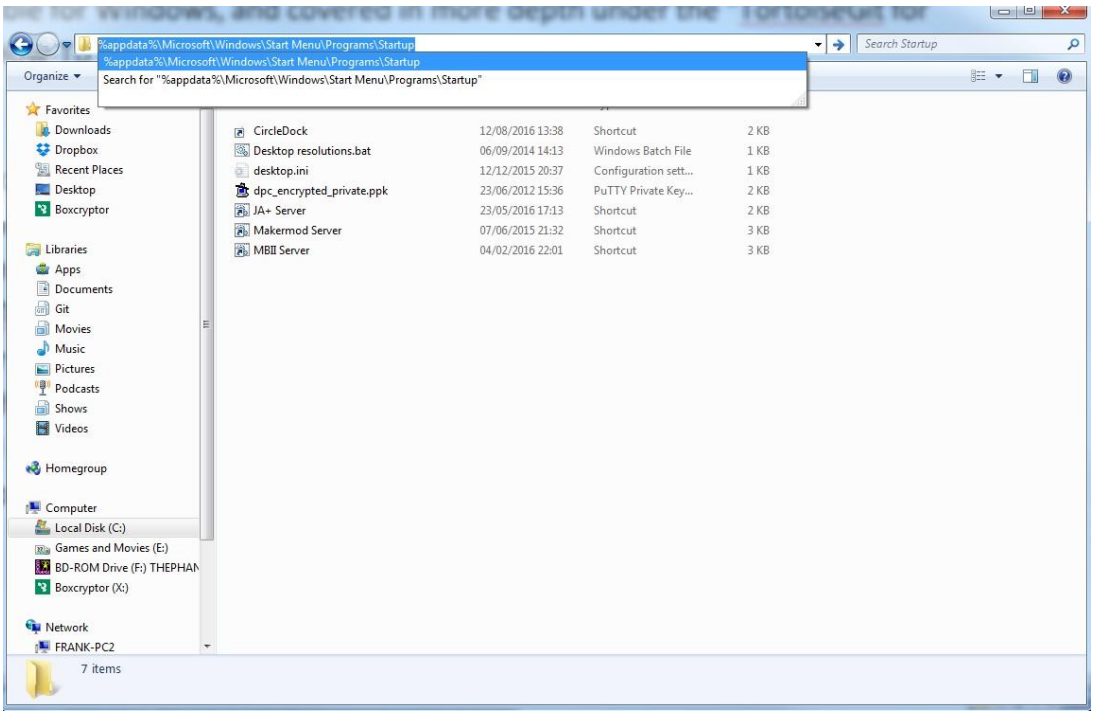

## put-2



## put-3


PuTTY Key Generator
File Key Conversions Help
Key
Public key for pasting into OpenSSH authorized_keys file:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCYVU+btE9xT/1+dzQSxU
+PFCxWSPL69rEc8sTFbsZKEasiUnxHjk/
+H/lez/0YaEpal4ZkzXqjNFauyFErb6laJ1mPYoBC1CyE/msNXjwloapwd72BrWQAGMc
1yx8D
+xR6y5N6tUn7ctCLvAlvwzkWn4rSRZQZFpiP3cxQMJ8ZmSsQF4tDKAsjCpYZKNYYj5z
Key fingerprint:
ssh-rsa 2048 d6:3d:29:90:85:7c:63:43:31:d5:fb:9d:4e:30:7f:85
Key comment:
imported-openssh-key
Key passphrase:
Confirm passphrase:
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
RSA
DSA
ECDSA
ED25519
SSH-1 (RSA)
Number of bits in a generated key:
2048

## put-4

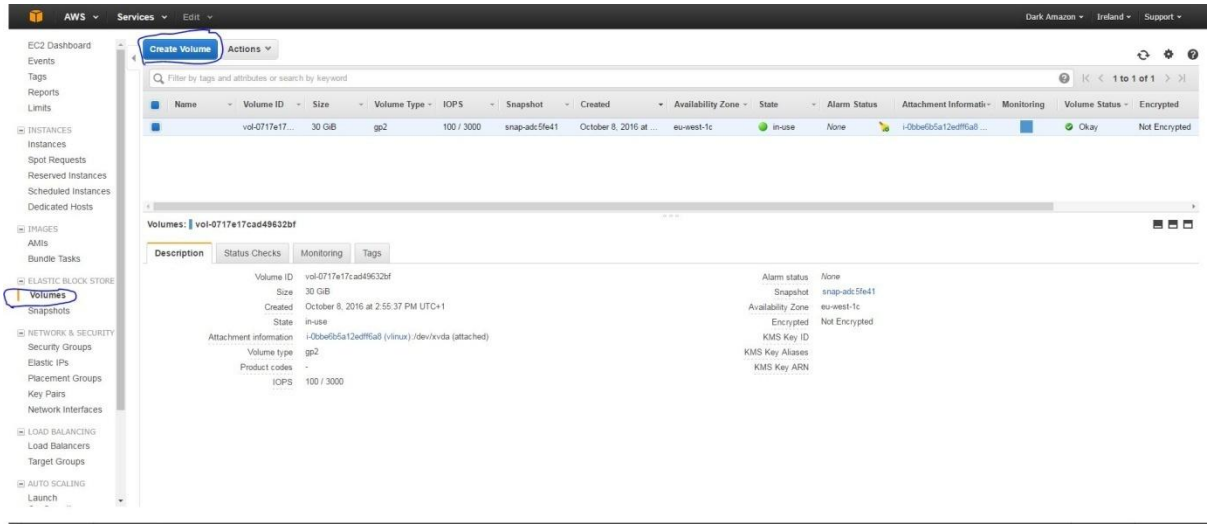

%appdata%\Microsoft\Windows\Start Menu\Programs\Startup
Search for "%appdata%\Microsoft\Windows\Start Menu\Programs\Startup"
Search Startup
Organize
Favorites
Downloads
Dropbox
Recent Places
Desktop
Boxcryptor
Libraries
Apps
Documents
Git
Movies
Music
Pictures
Podcasts
Shows
Videos
Homegroup
Computer
Local Disk (C:)
Games and Movies (E:)
BD-ROM Drive (F:) THEPHAN
Boxcryptor (X:)
Network
FRANK-PC2
CircleDock 12/08/2016 13:38 Shortcut 2 KB
Desktop resolutions.bat 06/09/2014 14:13 Windows Batch File 1 KB
desktop.ini 12/12/2015 20:37 Configuration sett... 1 KB
dpc_encrypted_private.ppk 23/06/2012 15:36 PuTTY Private Key... 2 KB
JA+ Server 23/05/2016 17:13 Shortcut 2 KB
Makermod Server 07/06/2015 21:32 Shortcut 3 KB
MBII Server 04/02/2016 22:01 Shortcut 3 KB
7 items

# Connecting to the Instance

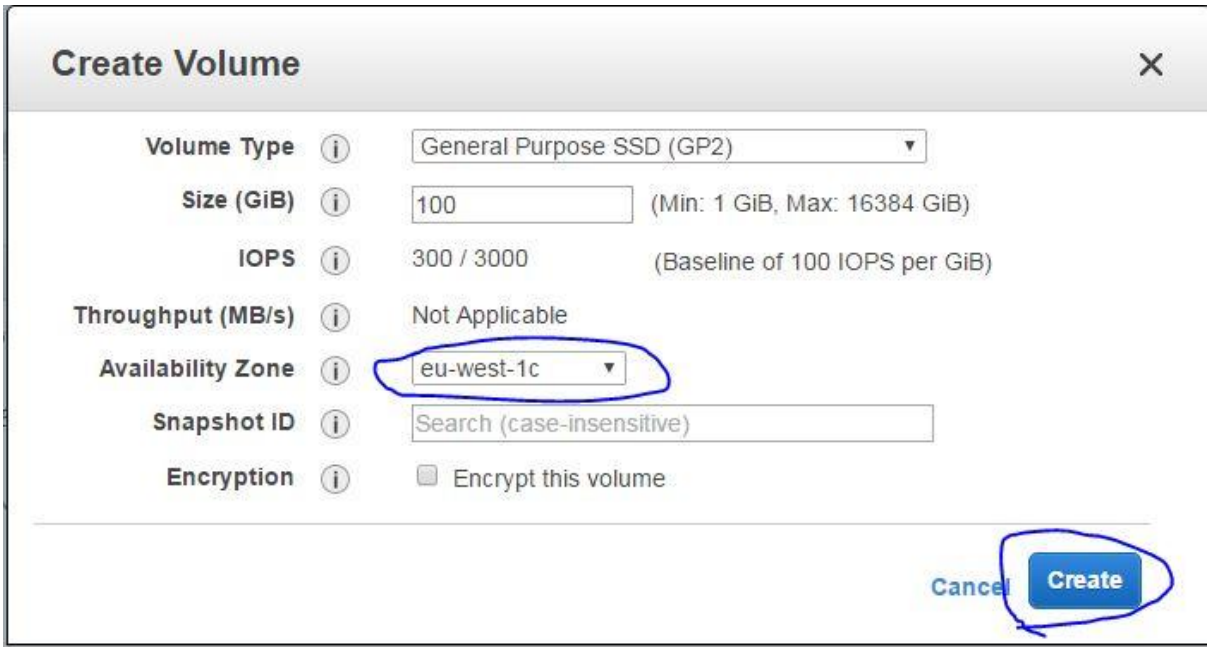
## con-1

# Creating / Removing EBS Volumes

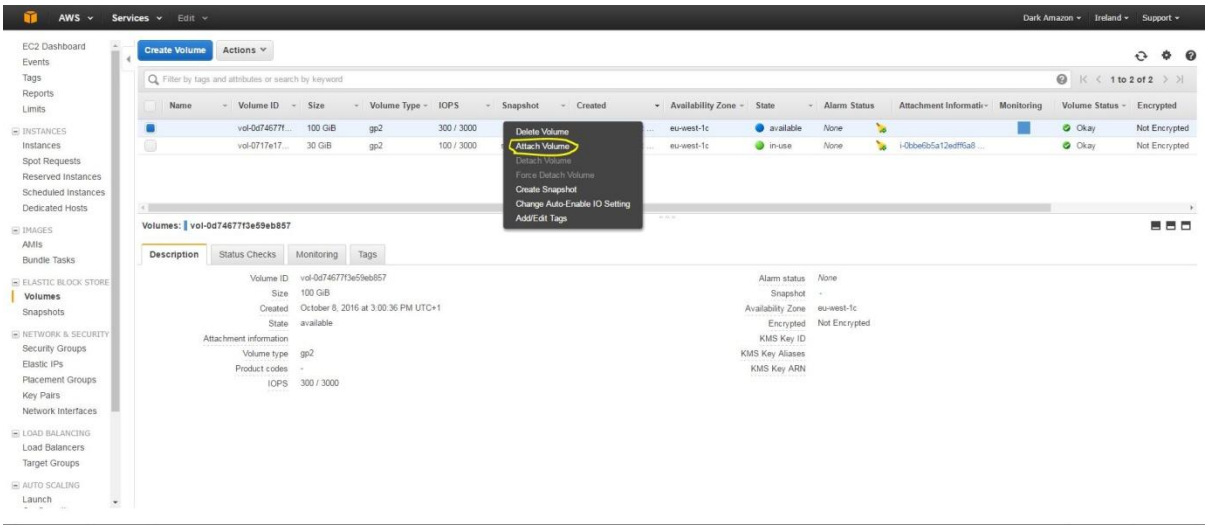
## ec2-9



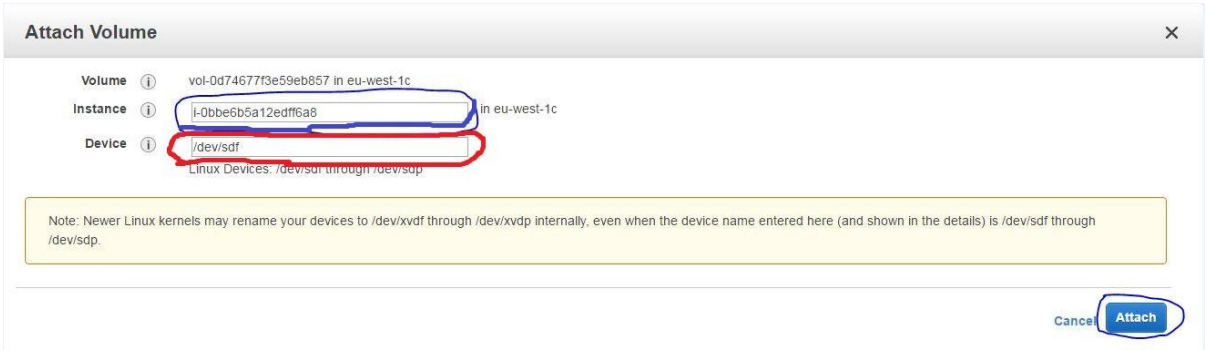

## ec2-10



## ec2-11

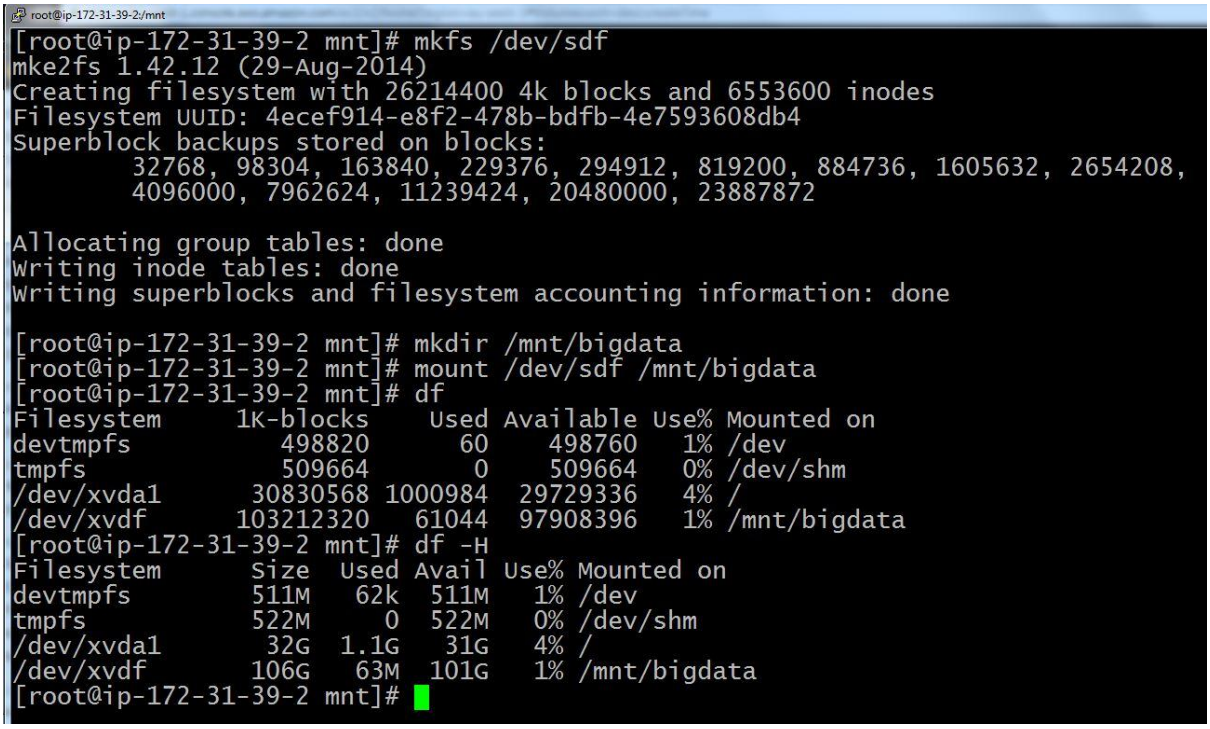

AWS
Services
Edit
Dark Amazon
Ireland
Support
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Spot Requests
Reserved Instances
Scheduled Instances
Dedicated Hosts
IMAGES
AMIs
Bundle Tasks
ELASTIC BLOCK STORE
Volumes
Snapshots
NETWORK & SECURITY
Security Groups
Elastic IPs
Placement Groups
Key Pairs
Network Interfaces
LOAD BALANCING
Load Balancers
Target Groups
AUTO SCALING
Launch
Create Volume
Actions
Filter by tags and attributes or search by keyword
1 to 2 of 2
Name
Volume ID
Size
Volume Type
IOPS
Snapshot
Created
Availability Zone
State
Alarm Status
Attachment Informati
Monitoring
Volume Status
Encrypted
vol-0d74677f...
100 GiB
gp2
300 / 3000
vol-0717e17...
30 GiB
gp2
100 / 3000
eu-west-1c
available
None
eu-west-1c
in-use
None
i-0bbe6b5a12edff6a8 ...
Okay
Not Encrypted
Okay
Not Encrypted
Delete Volume
Attach Volume
Detach Volume
Force Detach Volume
Create Snapshot
Change Auto-Enable IO Setting
Add/Edit Tags
Volumes: vol-0d74677f3e59eb857
Description
Status Checks
Monitoring
Tags
Volume ID vol-0d74677f3e59eb857
Size 100 GiB
Created October 8, 2016 at 3:00:36 PM UTC+1
State available
Attachment information
Volume type gp2
Product codes
IOPS 300 / 3000
Alarm status None
Snapshot
Availability Zone eu-west-1c
Encrypted Not Encrypted
KMS Key ID
KMS Key Aliases
KMS Key ARN

## ec2-12

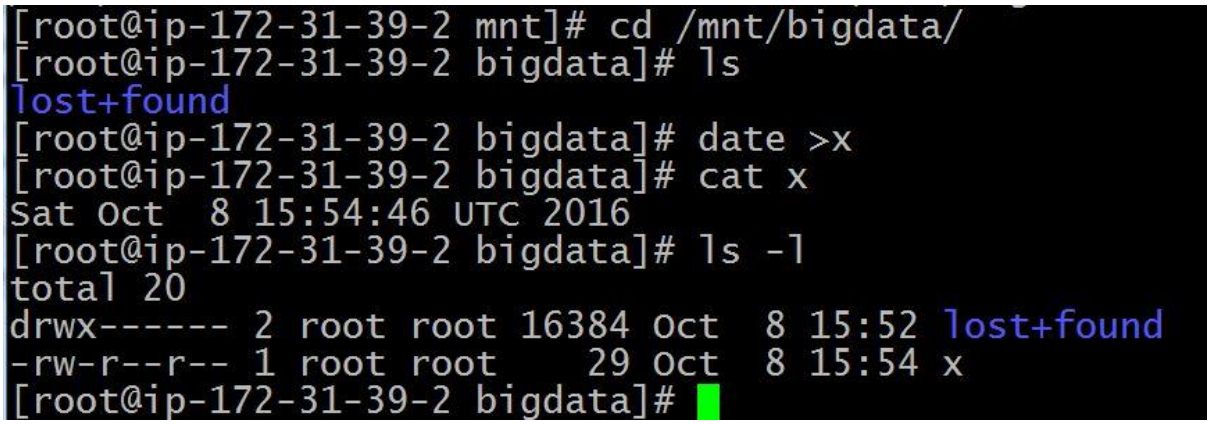

Attach Volume
Volume
vol-0d74677f3e59eb857 in eu-west-1c
Instance
i-0bbe6b5a12edff6a8
in eu-west-1c
Device
/dev/sdf
Linux Devices: /dev/sdf through /dev/sdp
Note: Newer Linux kernels may rename your devices to /dev/xvdf through /dev/xvdp internally, even when the device name entered here (and shown in the details) is /dev/sdf through /dev/sdp.
Cancel
Attach

## Managing EBS Volumes

### ec2-16

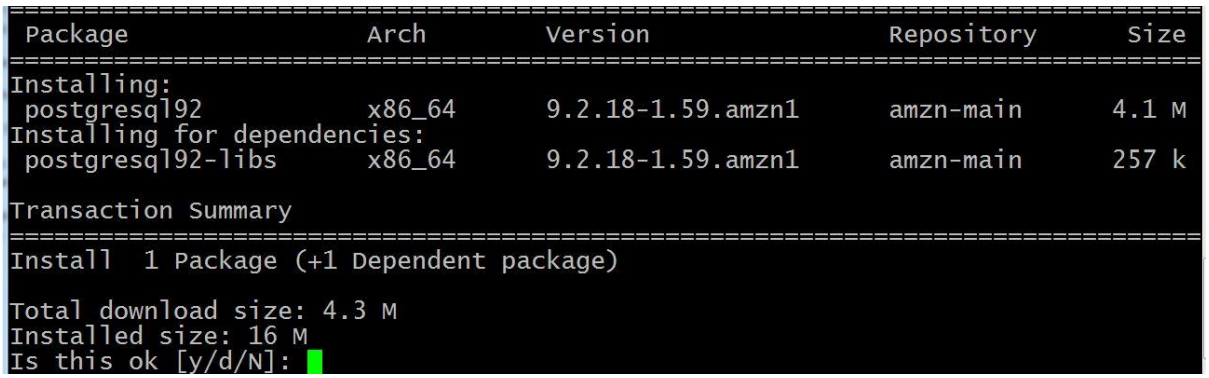



### ec2-17

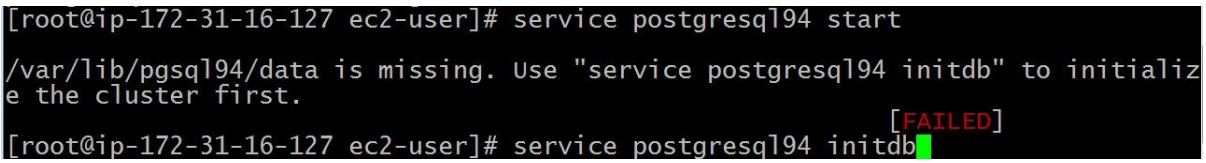


# Installing Apache and PHP, and Managing the WebServer

## pak-1




## pak-2



# Backup/Restore Test Data

## bak-1


## bak-2