

Sophia Alpha Project Report

Department of finance | One Canberra Avenue, Forrest. ACT, 2603

Digital records transformation initiative

Alexis Harper

2020

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Action | Completed By | Reviewed By |
| 2020/06/22 | Document creation | AH |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Project Overview 5](#_Toc44952751)

[Background 5](#_Toc44952752)

[Purpose 5](#_Toc44952753)

[Order of Priorities 5](#_Toc44952754)

[Objectives (and how they will be measured) 5](#_Toc44952755)

[Project Scope 5](#_Toc44952756)

[Deliverables 5](#_Toc44952757)

[Constraints 5](#_Toc44952758)

[Assumptions 5](#_Toc44952759)

[Business Case 5](#_Toc44952760)

[Project Benefits 5](#_Toc44952761)

[Alternate Options 5](#_Toc44952762)

[Cost and Timescale 5](#_Toc44952763)

[Cost/Benefit Analysis 5](#_Toc44952764)

[Risk Identification 5](#_Toc44952765)

[Risk Prevention 5](#_Toc44952766)

[Risk Management 7](#_Toc44952767)

[Roles and responsibilities 10](#_Toc44952768)

[Stakeholders 10](#_Toc44952769)

[Project Plan 10](#_Toc44952770)

[Initial Project Plan 10](#_Toc44952771)

[Work planning 10](#_Toc44952772)

[Milestones/Schedule 10](#_Toc44952773)

[Future Planning 10](#_Toc44952774)

[Project Control: Monitoring mechanisms, communication channels and schedules 10](#_Toc44952775)

[Quality Control 11](#_Toc44952776)

[Software Requirement Specification 12](#_Toc44952777)

[Architectural Design 13](#_Toc44952778)

[Architecture 13](#_Toc44952779)

[Drupal 13](#_Toc44952780)

[Amazon 15](#_Toc44952781)

[Server Design 15](#_Toc44952782)

[Language 15](#_Toc44952783)

[Libraries 16](#_Toc44952784)

[Neptune Sync 16](#_Toc44952785)

[Subsystems 16](#_Toc44952786)

[Querier 16](#_Toc44952787)

[Graph Generation 16](#_Toc44952788)

[Restful Interface 16](#_Toc44952789)

[Search System 17](#_Toc44952790)

[Data Imports 17](#_Toc44952791)

[User management 17](#_Toc44952792)

[Architectural View Decomposition 17](#_Toc44952793)

[Use-Case View 17](#_Toc44952794)

[Design View 18](#_Toc44952795)

[Process View 20](#_Toc44952796)

[Component View 20](#_Toc44952797)

[Deployment View 20](#_Toc44952798)

[Data Management 21](#_Toc44952799)

[Data Stores 22](#_Toc44952800)

[Data Persistence 22](#_Toc44952801)

[Neptune Data Schema 22](#_Toc44952802)

[Data Dictionary 22](#_Toc44952803)

[Appendix 22](#_Toc44952804)

Table of Figures

[Figure 1 Drupal data flow diagram 13](file:///C:\Users\AHarp\DoF\report\Sophia%20Alpha%20-%20Project%20Report.docx#_Toc44952805)

[Figure 2: Drupal Data-flow Diagram 14](file:///C:\Users\AHarp\DoF\report\Sophia%20Alpha%20-%20Project%20Report.docx#_Toc44952806)

[Figure 3 Sophia Alpha deployment diagram 15](#_Toc44952807)

[Figure 4: Serialization Overview 17](#_Toc44952808)

[Figure 5: Drupal Domain Diagram 18](#_Toc44952809)

[Figure 6 Neptune Sync class diagram 19](#_Toc44952810)

[Figure 7 The package diagram for Drupal 20](#_Toc44952811)

[Figure 8 The package diagram for the EC2 server 20](#_Toc44952812)

[Figure 9: AWS Drupal Hosting Pattern 21](#_Toc44952813)

# Project Overview

### Background

### Purpose

### Order of Priorities

### Objectives (and how they will be measured)

### Project Scope

#### Exclusions From Scope

### Deliverables

### Constraints

### Assumptions

# Business Case

### Project Benefits

### Alternate Options

### Cost and Timescale

[TODO]

### Cost/Benefit Analysis

[TODO]

### Risk Identification

### Risk Prevention

Stay abreast of and involved in GA product management developments. Build relationships with eCat team. Relationships with NCI. Maintain relevance to user base and scientific areas.

### Risk Management

#### Risk Strategy

-describe mitigations

#### Risk Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Profile | Likelihood | Impact | Mitigation |
| Stakeholder vision changed | Non-technical | High | Low | Reduce |
| Misaligned vision | Non-technical | Medium | Critical | Avoid |
| Staffing changes | Non-technical | Medium | Medium | Reduce |
| GovCMS issues | Technical | Medium | Medium | Transfer |
| Not meeting reputation | Non-technical | Medium | Low | Reduce |
| Drupal modules | Technical | Low | Low | Reduce |
| External Libraries | Technical | Medium | Medium | Reduce |
| Loss of funding | Non-technical | Medium | Critical | Avoid |
| Scope creep | Non-technical | High | Low | Reduce |
| Lack of business value | Non-technical | Low | Medium | Reduce |
| Drupal technical dept | Technical | Medium | Low-Medium | Reduce |
| External Interference | Non-technical | Medium | Medium | Reduce |
| Challenging the status quo | Non-technical | High | Medium | Reduce |
| Computation limits | Technical | High | Medium | Accept |

#### Detailed descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Stakeholder vision change | | | ID: R1 |
| Profile: Non-technical | **Likelihood: High** | **Impact: Low** | |
| Description: It is not uncommon to have stakeholder mind change on what they want, when they want it and what is more important than other requests. Traditionally this can cause a lot of wastage if this source of change is not pre-empted and handled rapidly. Time can be wasted on functionality the client no longer wants or priorities can be mis-aligned leading to stakeholder disappointment in the project and amplifying R8. | | | |
| Mitigation Strategy: R*educe;* by adopting an agile development practice we can quickly respond to change by re-examining the project and ourselves on a fortnightly basis. Further by keeping in strong contact with important stake holders and showing them the product via a fortnightly showcase to ensure the product is aligned with stakeholder intent. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Misaligned vision | | | ID: R2 |
| Profile: Non-technical | **Likelihood: Medium** | **Impact: Critical** | |
| Description: As a research and development style project it is essential, we have a strong internal vision of what the finish project should be. This is needed to ensure the project remains lean and does not fall off track to “nice-to-haves” and developer pet projects. It is not uncommon in this style of development to constantly fall into a spiral of “what are we even doing again” furthering the need for a strong central vision to rally and boost morale of the team. This Vision should ideally belong to one person who has a strong identity of what is needed. | | | |
| Mitigation Strategy: *Avoid*; as this project relies heavily on stakeholders’ interest and confidence, it is vital that this risk is avoided at all cost. Failing to mitigate this risk will likely bring about R8. A vision paragraph needs to be constructed among the internal team and presented to external stakeholders to sign off on. This vision needs to be assigned to a team member who will have ownership over the vision and ensure its constantly impressed on the team. Further, the vision statement needs to be integrated into team rituals and constantly reflected upon such in backlog revision (How does the story meet this vision) and sprint planning (do these goal align and bring about the vision). | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Staffing changes | | | ID: R3 |
| Profile: Non-technical | **Likelihood: Medium** | **Impact: Medium** | |
| Description: As the project grows due to stakeholder uptake and confidence the internal team will inevitably expand. Further, due to the experimental nature of this project, the team will most likely be filled with non-permanent workers. Thus, it is essential we have a strategy for on boarding a growing team and having protections in place for sudden exiting of team members and replacing them. | | | |
| Mitigation Strategy: R*educe,* as is the standard in project management, its important to have as much lead time and planning for a new role and on-boarding. Recruiting only one person at a time and leaving a size-able period of time passing between recruitments. This is to ensure that team culture remain intact and can naturally spread to the recruit. Further Recruiting too rapidly will put a substantial stress on the team due to on-boarding process, mentoring and over-all integration into the team and office. It should be kept in mind that recruitment takes place outside of work-intense periods.  To safeguard and mitigate work velocity due to an unplanned exit of a team member work should be constantly documented ad-hoc during development and refined in later sprint tasks. Work document should take place in; code comments, git logs, works diaries, technical notes, comments, and descriptions in the team planner. Of note, chat logs in gov-teams and emails are not suitable to document work. These documents should be collated into this document in their relevant section or appendices when relevant. Care should be taken that at least two people have admin credentials for all system infrastructure. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: GovCMS issues | | | ID: R4 |
| Profile: Technical | **Likelihood: Medium** | **Impact: Medium** | |
| Description: As this is a Department of Finance project, we should endeavour to integrate GovCMS into our project. We can also take advantage of its CI/CD build process, Git-lab features, and advance hosting features when the product become live. However, GovCMS enforces a layer of infrastructure and architecture that must be navigated and adhered to. Even Utilizing GovCMS as a PAAS, developing on the platform becomes more costly and risks unseen blockers due to the scope and constrictions embedded by the infrastructure. | | | |
| Mitigation Strategy: *Transfer;* we have set-up a plan with GovCMS that offers a dedicated support desk service which should be utilize as a first step of triaging a GovCMS related issue. Further, we should invest in training a developer around GovCMS development | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Not meeting reputation | | | ID: R5 |
| Profile: Non-technical | **Likelihood: Medium** | **Impact: Low** | |
| Description: As experienced during the first few sprints of the project, we have built almost an expectation of rapid productions and putting out compelling functionality each sprint. We need to carefully manage this expectation as the pace set by early sprint is unsustainable and focused on low hanging fruits. Failure to manage this reputation into a more feasible and sustainable image could results in R8 | | | |
| Mitigation Strategy: *Reduce,* stretch goals should be discussed internally only as the team is more familiar around the current strains and workload of one another. Stretch goals should never be held as a commitment and only as a nice extra if it is reached. Sprints that deliver viable business value should be evenly interwoven with sprints that are more back-end/planning/documentation focused as to not create large gaps for external stakeholders to think nothing is being done. Changes to the sprint plan (both additions and subtractions) should be discussed with the team and the feasibility of such an action examined. Feature requests from stakeholders should never be committed to individually but rather noted and brought up in an internal team meeting for a group consensus. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Drupal Modules | | | ID: R6 |
| Profile: Technical | **Likelihood: Low** | **Impact: Low** | |
| Description: As contributed Drupal modules are open source and do not go through standards/code reviews nor are examined by the security team unlike the Drupal core code, they pose a risk to the development of the project. Situations can arise that a module integrated and firmly embedded into the project several sprints ago are suddenly abandoned leading to a lack of bug fixes, security fixes and a non-existent Drupal core upgrade path requiring the functionality to be ripped out and replace with custom code or a new module. As installing a module modifies the database, filesystem, and config files. A poorly made module has the capability to corrupt these mediums in a non-recoverable way. | | | |
| Mitigation Strategy: *Reduce;* Before any module is integrated into the project a brief feasibility study should be conducted on it. The following should be observed in this report:   * Release type (dev, alpha, RC, release) * Number of sites currently run on in Drupal 8 * Date of last update * The issue tracker (how many issues are open, how quickly are they resolved, how many are critical) * The git repo (doe the code look healthy, are standards followed, does it utilize the API) * Module reputation if available * State of funding and if it is still being funded if available   A stance of less code smell and a minimal viable set should be taken when integrating modules. That is, to use the least amount of modules possible to obtain functionality and to use modules that indicate they are in a healthy state. | | | |
|  | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: External Libraries | | | ID: R7 |
| Profile: Technical | **Likelihood: Medium** | **Impact: Medium** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Loss of Funding | | | ID: R8 |
| Profile: Non-technical | **Likelihood: Medium** | **Impact: Critical** | |
| Description: | | | |
| Mitigation Strategy: *Avoid;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Scope creep | | | ID: R9 |
| Profile: Non-technical | **Likelihood: High** | **Impact: Low** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Lack of business value | | | ID: R10 |
| Profile: Non-technical | **Likelihood: Low** | **Impact: Medium** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Drupal technical dept | | | ID: R11 |
| Profile: Technical | **Likelihood: Medium** | **Impact: Low-Medium** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: External Interference | | | ID: R12 |
| Profile: Non-Technical | **Likelihood: Medium** | **Impact: Medium** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Challenging the status quo | | | ID: R13 |
| Profile: Non-Technical | **Likelihood: High** | **Impact: Medium** | |
| Description: | | | |
| Mitigation Strategy: *Reduce;* | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: Computation limits | | | ID: R14 |
| Profile: Technical | **Likelihood: High** | **Impact: High** | |
| Description: | | | |
| Mitigation Strategy: *Accept;* | | | |

### Roles and responsibilities

### Stakeholders

# Project Plan

[TODO]

### Initial Project Plan

### Work planning

### Milestones/Schedule

## Future Planning

[TODO]

### Project Control: Monitoring mechanisms, communication channels and schedules

### Quality Control

# Software Requirement Specification

[TODO]

# Architectural Design

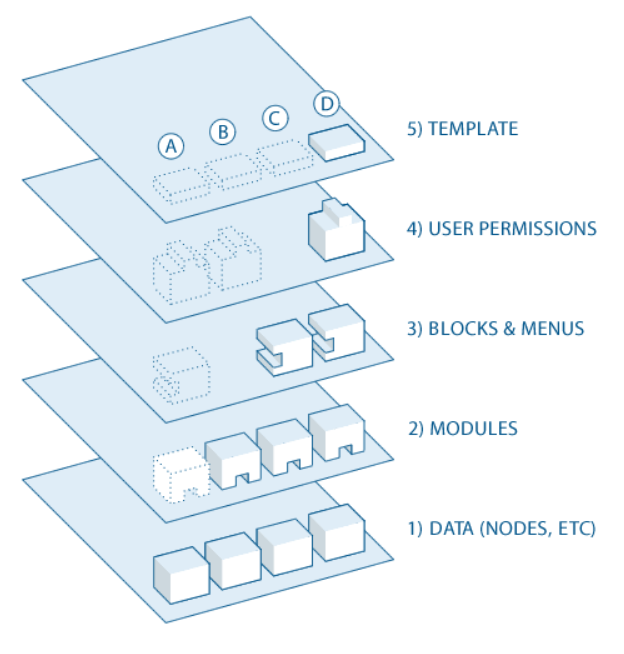
## Architecture

### Drupal

Drupal is an open source Content Management System (CMS) that is actively supported and is used all over the world include the government of United States of America, France and Australia. Large commercial companies such as the BBC and NBC also utilize Drupal.

Drupal also has a history of pre-emptive or rapid responses to security threats while maintaining stable branches with ongoing, open source development.

As an open source project, Drupal has the support of 4,500 people, companies and organisations that have contributed to its development. These contributions can be split into three distinct areas, these area’s being Core, Modules and Templates.

The core component are pieces of plug and play functionality contained in packages called modules that are provided as default with the installation of Drupal. These Modules can be enabled or disabled at will to switch on or off entire functionality throughout the site. Functionality such as content moderation, caching, menus and Search features. These core modules come with a guarantee of support, stability and security.

Modules are community contributed but have been deemed either not developed enough or outside the scope of what Drupal core should provide. The bulk of custom functionality comes from these modules with over 40,000 modules contributed to Drupal, nearly any functionality that is needed, is provided for or exist as a base and requires just a little bit of tweaking. These modules offer various levels of security, support and stability and are often use at your own risk.

Figure Drupal data flow diagram

Ref: https://www.drupal.org/docs/8/understanding-drupal-8/overview

Lastly, Templates are user interface (UI) skins for the site. Themes determine where things are displayed on each page and how they are displayed. It provides polish layer onto of the system that combines Html, JavaScript and CSS that attaches to dynamically assign html classes and display blocks. These themes often provide a base look and feel for a site while allowing for easy customization to alter it into what you are after.

Drupal can be seen as an advance graphical, customizable database API/layer. As at its core, its main roles I to display information contained in the database. However, it’s the systems, tools, API and even transformations that are contained within Drupal that makes Drupal so powerful. An example of this can be seen in the image to the left.

Data is stored in objects called nodes. These nodes group the data and may be embedded or linked to other nodes to enrich the data of the linked/embedded node.

This data then flows upwards through the module through the Drupal API, being able to be hooked in and altered at any step of the way. Next, the data is transformed by any modules that hooks into this API, this could be a module that receives the GPS co-ordinates of a location and returns a Google maps display of the location as a result. Or possibly takes in data and validates it against an external source. The module layer is often the best place to interact and transform the data coming from nodes.

Next, blocks and menus are added to the picture, providing forms, links, embedded subsystems (i.e. blocks) and a base UI. Blocks are a powerful tool in reducing the amount of work that needs to be done and automating the work that does need to be done. For example, if you need to accept a terms of service agreement when submitting data, a block can be used to build, present and capture the response. Once this block is built, it can be embedded on any page that requires this small subsystem.

Figure : Drupal Data-flow Diagram

User permission is an in-depth system that allows you to assign users to roles and roles to permission sets. These permission sets allow what a user can do or see along each step of this data flow. Admins may be excepted from accepting the both mention terms of service agreement and thus can be removed from what they interact with. Or even more simply, a certain user type may not have permission to view a specific set of nodes.

Lastly, while the blocks and layer stage provided what extra elements are added to the end view of this node, they have no positioning, look or structure. Thus, the final layer adds this information to provide a complete webpage. Using HTML, JavaScript and CSS, the current state of the node is a bunch of information with no display is transformed into a themed webpage.

#### Drupal 8

Drupal currently has two main versions co-existing. The first being Drupal 7, which was released on January 5th, 2011 and the second being Drupal 8 which was released on November 19th,2015. While both are considered stable and actively get development and security updates, they are vastly different. Drupal 7 is considered a mature and stable release supporting over 14,000 modules and being used by nearly 700,000 sites as of June 2020.

Drupal 8 however, despite being nearly three years old from release is considered in a state for early adopters supporting 5,812 modules and is deployed on just over 200,000 sites worldwide. As such, functional coverage is much lower on Drupal 8, requiring more work to fill missing functionality or less activity and progress when filing bug reports.

Functionally, Drupal 7 and 8 are also different in terms of their core workings. For example, separation f content and configurations, allowing for easier deployment, an improve entity API, object orientated design patterns in its working and API’s, HTML5 support, focus on data transformations and a RESTFUL interface. All these improvements push Drupal 8 away from Drupal 7’s content management ecosystem into more of a data hub space. It should be made clear that migrating from one version to the other is a massive undertaking.

Despite the adversity and lack of support of adopting Drupal 8 early, we selected to use Drupal 8 due to each future proofing platform and its push towards being a data hub which aligns more with our needs.

### Amazon

The entire Sophia-Alpha system is deployed through Amazon Web Service (AWS) using Elastic Compute Cloud (EC2) as the server host, Relational Database Service (RDS) for database hosting, and Simple Email Service (SES) for its SMTP server.

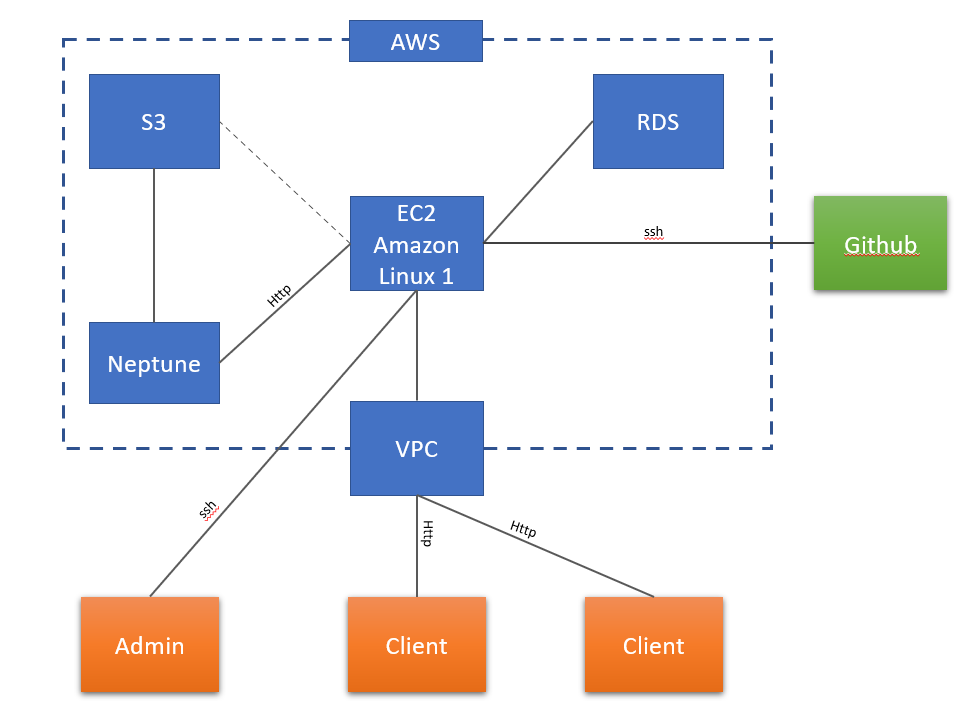


Figure Sophia Alpha deployment diagram

Development and maintenance of the server is conducted through SSH and SCP with an identity file (containing a private key) which is required for all connections.

### Server Design

The server design takes the form of a traditional LAMP (**L**INUX, **A**PACHE, **M**ySQL, **P**HP), albeit locked into Amazons provisions of these technologies, thus the stack details are as follows:

* x64 AWS 4.14
* Apache(AMZN) 2.4
* 5.5 MariaDB
* PHP 7.2

### Language

The dominate language is PHP (7.1) as this is the language Drupal is coded in and thus also, the API. Accordingly, all programmatic aspects of modules are done in PHP though other language wrappers do exist (but aren’t currently used). There are also many opportunities for administrators to inject PHP through the UI for various tasks, such as formatters, blocks, rules and actions.

Drupal is also highly configuration centric, meaning that a high majority of structural information about the site is stored in configurations. For configurations YAML is the script of choice, again, inherited from what Drupal enforces. However, there is a slight push for JSON/XML and wrappers exist to use these scripts as well.

Templating uses the TWIG language which allows HTML5, JavaScript and programmatic logic to intermingle as a single entity. TWIG defines the visual structure of a pieces of content before CSS is applied to produce a final view.

Lastly, server-side scripting is done in shell. With only a tiny amount of shell scripts used by server admins for automated tasks such as maintenance jobs, back-up, deployment and git interactions.

### Libraries

Currently, only few libraries used in this project and are listed below.

(Core required, Module required, Sophia required)

* Twig: used to provide the Twig language for templating as required by Drupal core
* Zend Framework: used as the framework that provides hooks into Drupal for various controllers, driven by Symphony and thus Composure. Is a dependency for Drupal core.
* Symphony: provides a controller and functionality suit to Zend framework and is a dependency for Zend framework
* Composer: used for providing the main interface into Symphony and the functionality it provides. It is also responsible for defining high level configurations that have external dependencies. Replaces Drush.
* Jquery: used for the front-page slideshow, as a dependency for the slideshow views module.
* Drush (deprecated): used for providing an interface into Symphony. Providing many site admin services. Replaced by Composure.
* PHPunit: used for providing unit test around the majority of the project’s logic
* Echarts: used provides and API for generating dynamic and independent graphs.

### Neptune Sync

## Subsystems

Sophia Alpha can be separated into various high-levelled sub system, these are the querier, Graph generation, RESTful web services, feeds, search and user management. These subsystems are described below.

### Querier

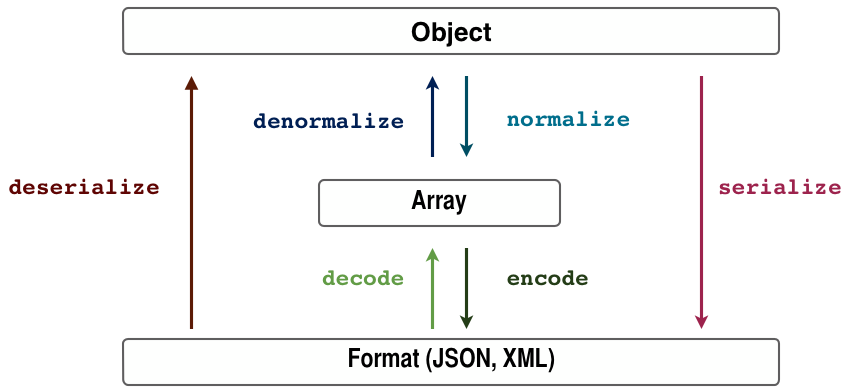
### Graph Generation

### Restful Interface

The RESTful resource interface is provided by four community provided modules, these are:

* Serialization
* RESTful Web Services
* HTTP Basic Authentication
* REST UI

As at its core, Drupal stores content as objects in its data base. Through a process called normalisation, all the relevant data for a content item is collecting from the relevant database tables via its content type schema and constructed into an array. From this array, we can encode the content item into many different formats, including but not limited too JSON and XML as seen in the diagram below.



<https://www.drupal.org/docs/8/api/serialization-api/serialization-api-overview>

Figure : Serialization Overview

This functionality is provided by the *Serialization Module*. In order to allow these serialized outputs to be external facing and bound to a internal URL the *RESTful Web Services modules* is used. This module also allows you to add permissions, which resources can be accessed and how they can be accessed.

As the information on Sophia Alpha is restricted to authorized users the *module HTTP Basic Authentication Module* is used, which allows the web service to accept account authentication token sent in the header of a RESTful request.

Lastly, to provided easy management of the settings provided by the *RESTful Web Services Module*, the REST UI module is used which provides a UI to the configuration, rather than configuring the webservices from a YAML document on the server.

### Search System

### Data Imports

### User management

## Architectural View Decomposition

### Use-Case View

[TODO]

### Design View

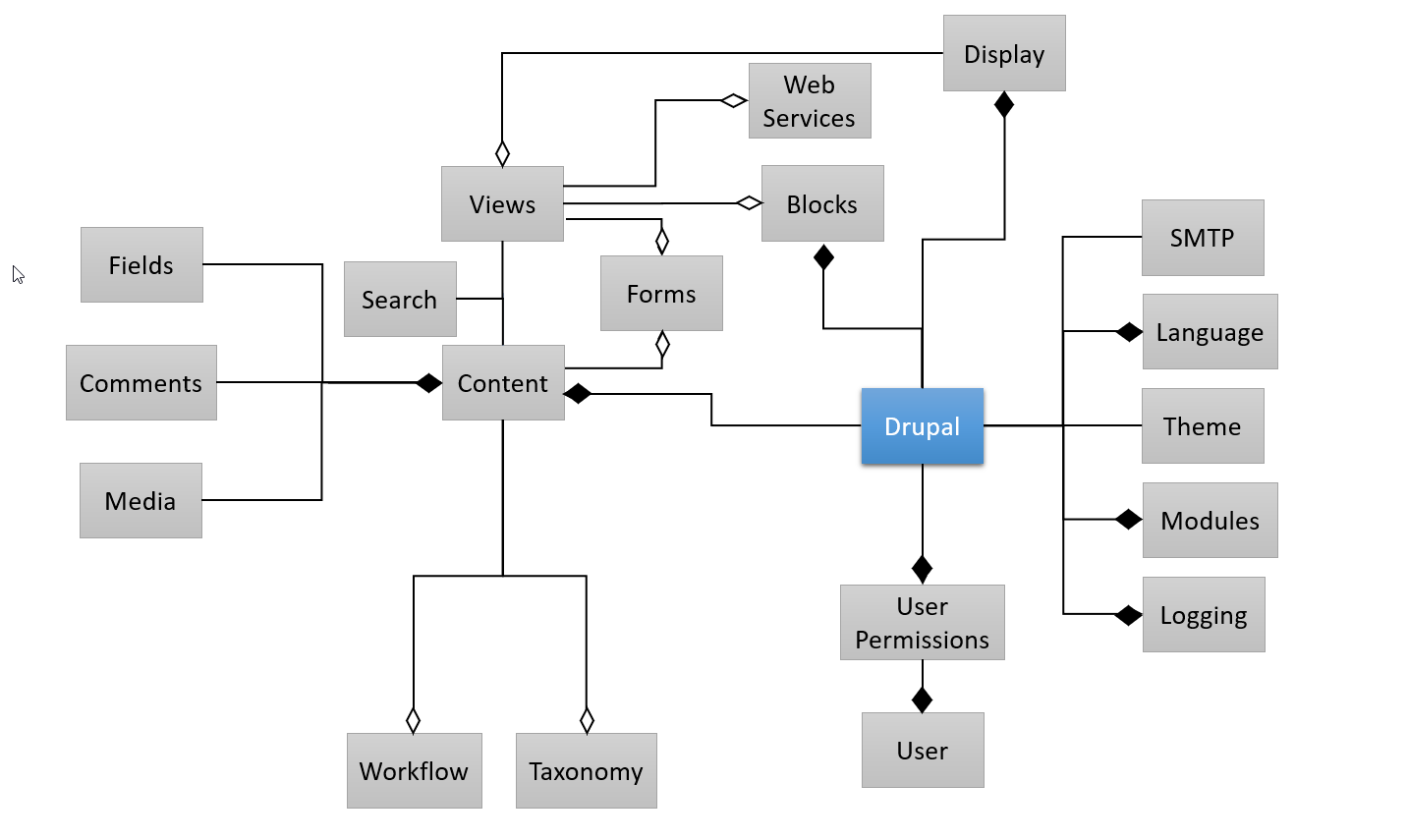


Figure : Drupal Domain Diagram

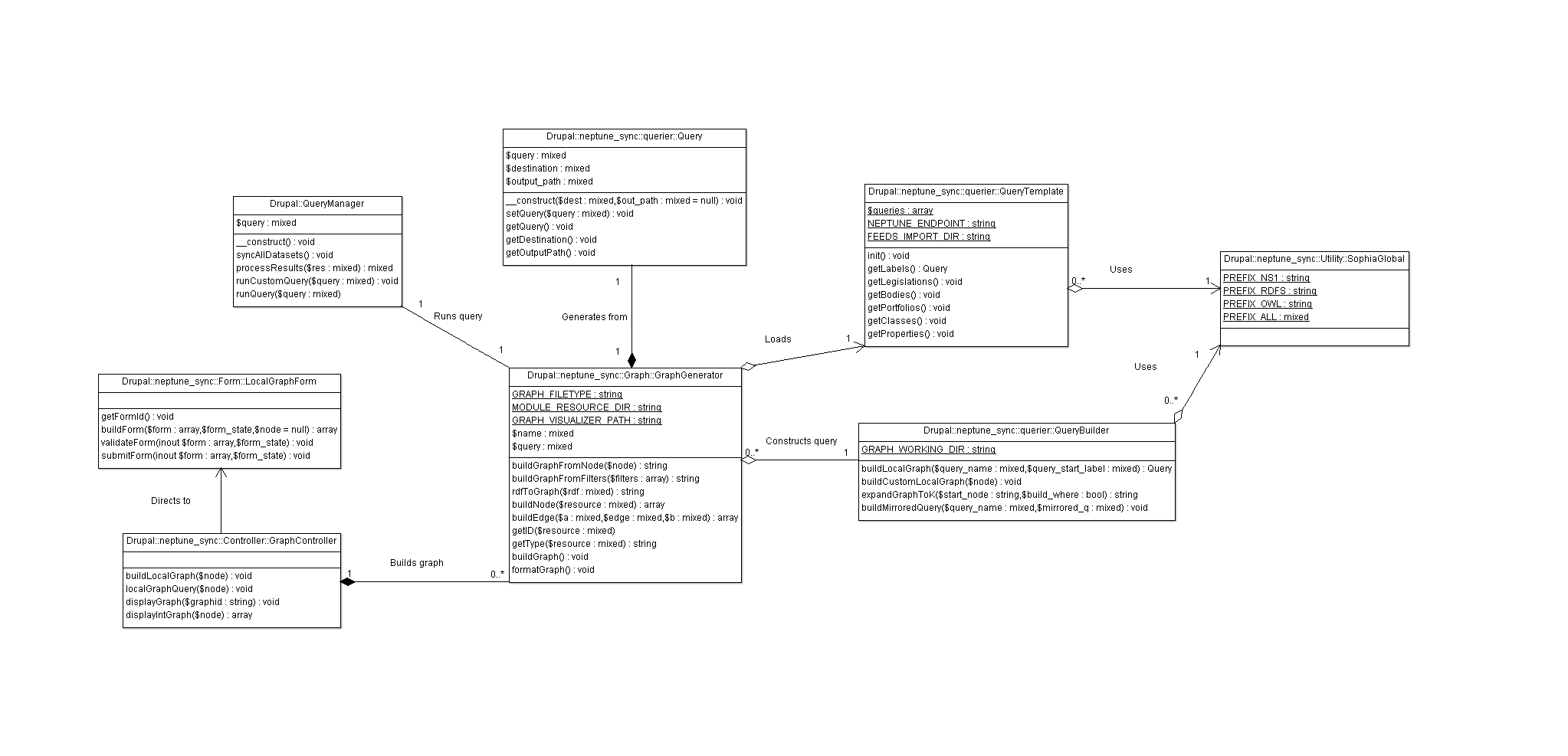


Figure Neptune Sync class diagram

#### API

### Process View

[TODO]

### Component View

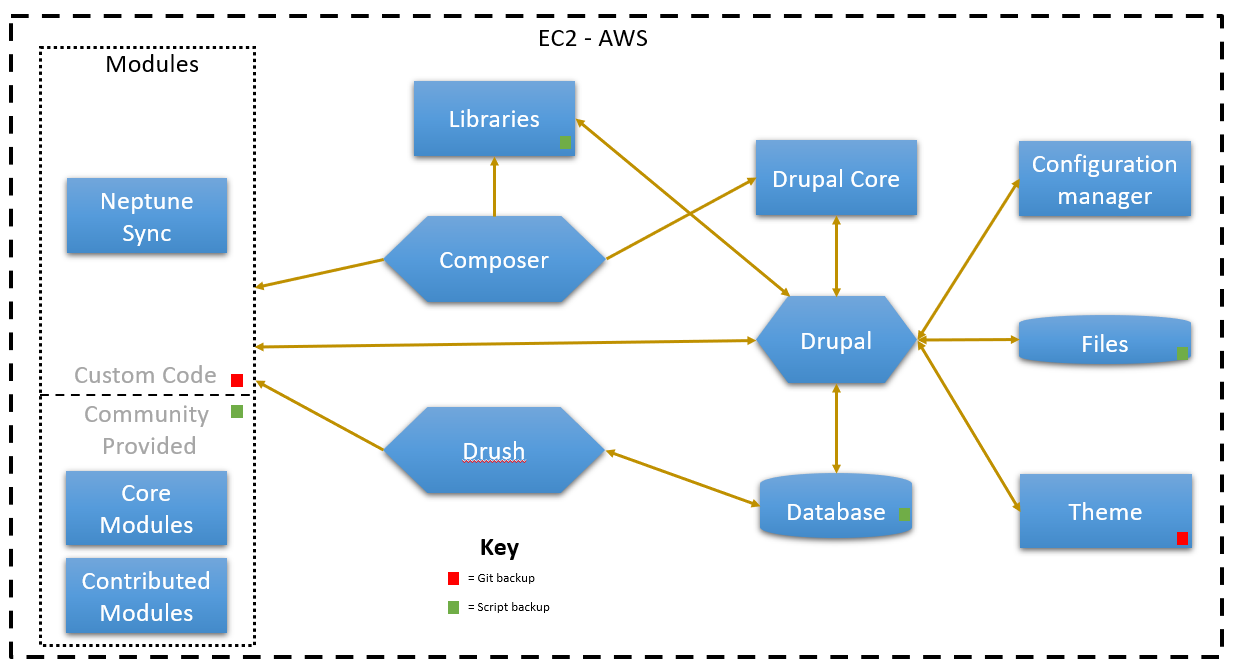


Figure The package diagram for Drupal

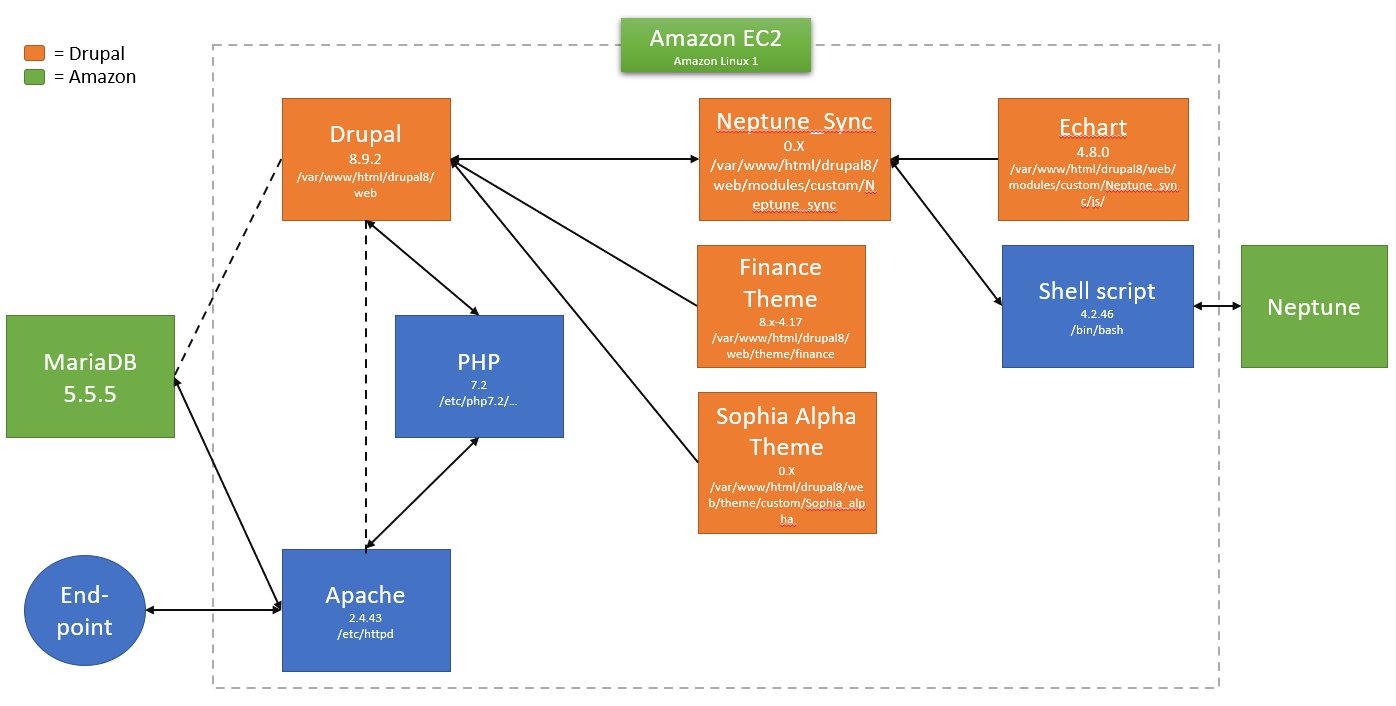


Figure The package diagram for the EC2 server

### Deployment View

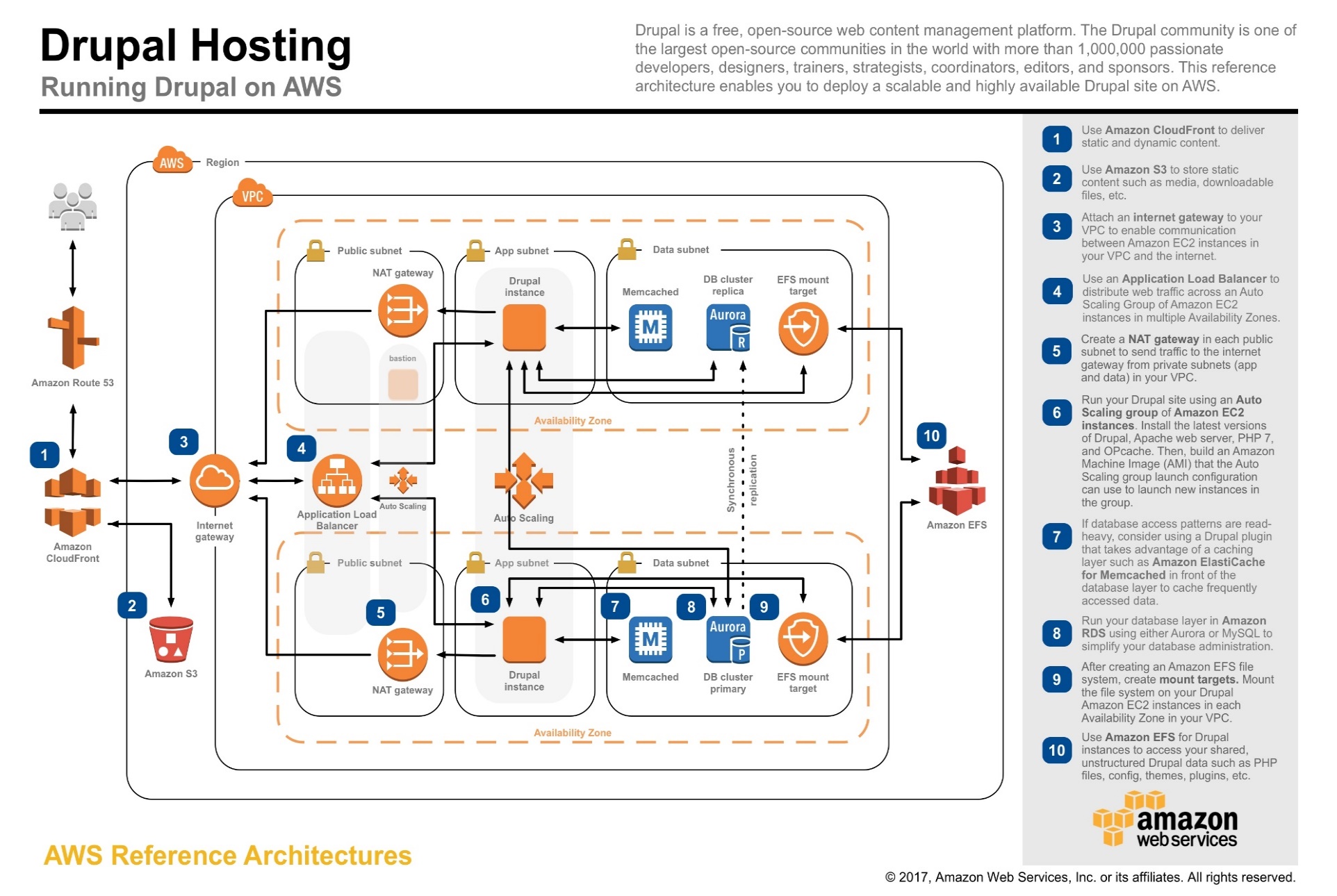


Figure : AWS Drupal Hosting Pattern

## Data Management

[TODO]

\*data strategy

### Data Stores

#### Neptune

#### RDS

##### Config

#### S3

#### EC2

##### Composer

##### Core

##### Modules

##### Theme

##### Libraries

### Data Persistence

### Neptune Data Schema

### Data Dictionary

# Appendix