# Technical Solution Specification Document

for

# NTT PC – Web Arena

Version 1.0



**Emerio GlobeSoft Pte. Ltd**

Emerio House, 50 Ubi Crescent #01-05 Ubi Tech Park Singapore 408568 Tel: (+65) 63492999 Fax: (+65) 6349296

Http://www.emeriocorp.com

# Table Of Contents

## Document Acceptance and Release Notice

This is the baseline version 1.0 of the Technical Specification Document of the NTT PC – Web Arena project.

| Emerio | | |
|---|---|---|
| **Name** | **Designation** | **Signature** |
| **Prepared By** | | |
| Venkateswarlu Sayana | Lead Solution Architect | |
| | | |
| **Reviewed By** | | |
| Mohan Kumar P | Vice President | |

| NTT PC | | |
|---|---|---|
| **Name** | **Designation** | **Signature** |
| **Reviewed By** | | |
| NTT PC Team | | |
| **Approved By** | | |
| | | |

## Summary of Document Changes

This document is subject to Technical solution design specification. For identification of amendments, each page contains a version number, section identifier and page number within the section.

Changes will only be issued as complete replacement sections covered by a Release Note signed by the Approving Authority. All replaced sections will be destroyed immediately. This document will not be released for use, unless it has been duly authorized.

an NTT Communications Company

## Version History

| # | Version No. | Version Date | Remarks |
|---|---|---|---|
| 1 | 0.1 | 10/09/2018 | Draft Version of the document |
| 2 | 0.2 | 25/09/2018 | Added contents to Design Overview, Layered Architecture, System Environment sections. |
| 3 | 0.3 | 04/10/2018 | Incorporated review comments given by Mohan |
| 4 | 1.0 | 11/10/2018 | Base lined version |
| | | | |
| | | | |

# 1. Introduction

Currently, NTT PC Communications offers Web Arena and infra service of NTT PC to their domestic customers and their internal usage. The existing solution is into monolithic architecture and proposed to migrate to MicroServices architecture implementing CI/CD process to enhance their application for high performance and high availability. It shall follow/succeed the current VPS used by SuitePRO V4's internet connection and VPS cloud. Each module shall access the information using the web based APIs' provided by NTT PC as part of this project.

## 1.1. Document Purpose

This document outlines the technical solution design of the NTT PC Web Arena application and its interaction with the other MicroService-applications. It describes the high level design, patterns to follow, detailed design for common functionality and frameworks and the application functionality in detail. All system information and requirements documented herein are solution and compiled by Emerio Global Soft Pte Ltd. (Emerio) project team from the various stakeholders of this project.

This document shall be taken as the Blueprint by Emerio project team for designing, building, testing and deploying of the NTT PC APIs'. The NTT PC Project Management team shall agree with all the information provided in this document.

## 1.2. Document scope

The scope of this document includes Technical Design covering:

- High Level Design
- Patterns and frameworks
- Common functionality
- Application functionality in detail

## 1.3. Intended Audience

The following table lists the key users of this document, together with the reason for this use.

| User | Use |
|---|---|
| Developers, Leads | Referenced during the detail design |
| Solution Architect | Review to ensure meets the solution outline |

## 1.4. Related Documents

Documents referred to during preparation of the Application Architecture are listed in the table below:

| # | Title | Version | Issue Date | Comments |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | NTT PC Web Arena Community Detailed BRS | | | |
| 2 | NTT PC Web Arena Functional Specification ( ) | | | |

## 1.5. Involved Parties

Emerio Development Team and NTT PC

## 1.6. Document Reference Personnel

Queries regarding this document may be directed to:

| User | Use |
|---|---|
| Venkateswarlu Sayana | Lead Solution Architect |
| Chandrasekaran P | Project Manager |

# 2. NTT PC Overview

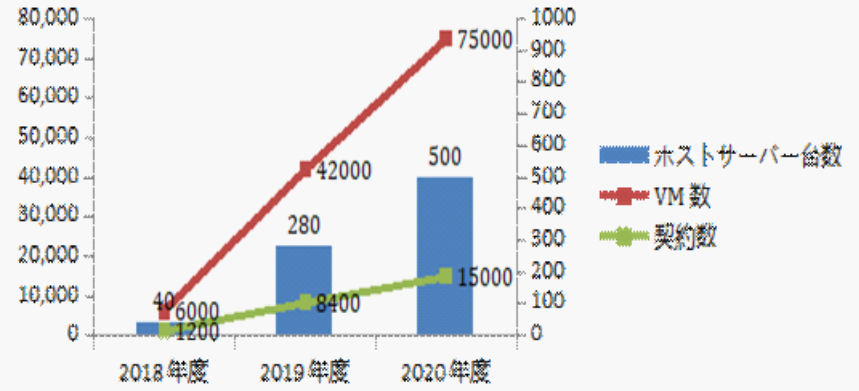
- NTT PC is established in the year 1985 (NTT PC is older than NTT Com).
- NTT PC's business consists of NW-Service (about 60% Internet and VPN) and Server-Service (about 20%
- https://www.NTT PC.co.jp/english/overview.html

## 2.1. Business Target (Plan)

NTT PC provides simple IaaS completely at a lower price to customers. It follows/succeeds the current VPS (SuitePRO V4's internet connection and VPS cloud). The first release is planned to happen in Japan (Domestic) and then extend to APAC region.

**Target/business scale:**

| | 2018 | 2019 | 2020 |
|---|---|---|---|
| **Number of Host Servers** | 40 | 280 | 500 |
| **VM count** | 6,000 | 42,000 | 75,000 |
| **Contract count** | 1,200 | 8,400 | 15,000 |




# 3. What is Web Arena?

NTT PC's hosting brand "Web Arena", web server housing service, was launched in 1997. Now, Web Arena provides "mail and web hosting", "VPS" and "domain name service" to about 80 thousand customers.

NTT PC's revenue of 2017 is over ¥ 2.5 billion (+3% than previous year). The reason for this growth is cost-effective VPS services.

URL: https://web.arena.ne.jp/vps-cloud/ Start price ¥360.

# 4. NTT PC VM and Container Info

NTT PC Web Arena (SimpleIAAS) is a Cloud platform to provide the cost-effective, scalable solutions and services to their (domestic as well APAC region spread) customers.

Provide virtual server (KVM) and container and charges will be based on usage

- Will not collect NW usage fee
- Billing will be done for virtual server and container when it is used. When instance is abolished, billing will not be done. (If an instance is there, billing will be done for that instance)
- Planning to additionally include various options pertaining to the above
  - Load balancer, DNS, additional storage, snapshot backup
  - Migration function among plans
  - IP V4

Provide Virtual server and container types (Menu)

- <VM Type S> Virtualized KVM, Memory 1GB ~ 16GB

Accommodates KVM based VM in a single server (SSD loading) and share CPU core.

When there are server defects, there is no failover. When it cannot be recovered, users can recover using snapshot

- < VM Type S2 > Virtualized KVM, Memory 1GB ~ 16GB

Accommodates KVM based VM in a single serve (SSD loading) and has a specific CPU core.

When there are server defects, there is no failover. When it cannot be recovered, users can recover using snapshot

- < VM Type HA > Virtualized KVM, 4GB ~ 64GB, shared CPU core

When there are server defects, host will do a failover and it will be automatically restarted

- < VM Type HA2 > Virtualized KVM, 4GB ~ 64GB, specific CPU core
- < Container > Virtualized Docker, 256MB ~ 64GB

When there are server defects, container failover will be done using K8S etc., automatically

# 5. System Design Overview

## 5.1. System Overview

NTT PC Web Arena (Simple IAAS) is designed for simplicity and providing easy-to-scale customer services. NTT PC Web Arena helps the Development teams to easily manage their infrastructure services in a more efficient and faster manner.

NTT PC Web Arena with its services helps to build, deploy and scale faster applications to the domestic customers in Japan and plans to expand in APAC region.

## 5.3. System Architecture

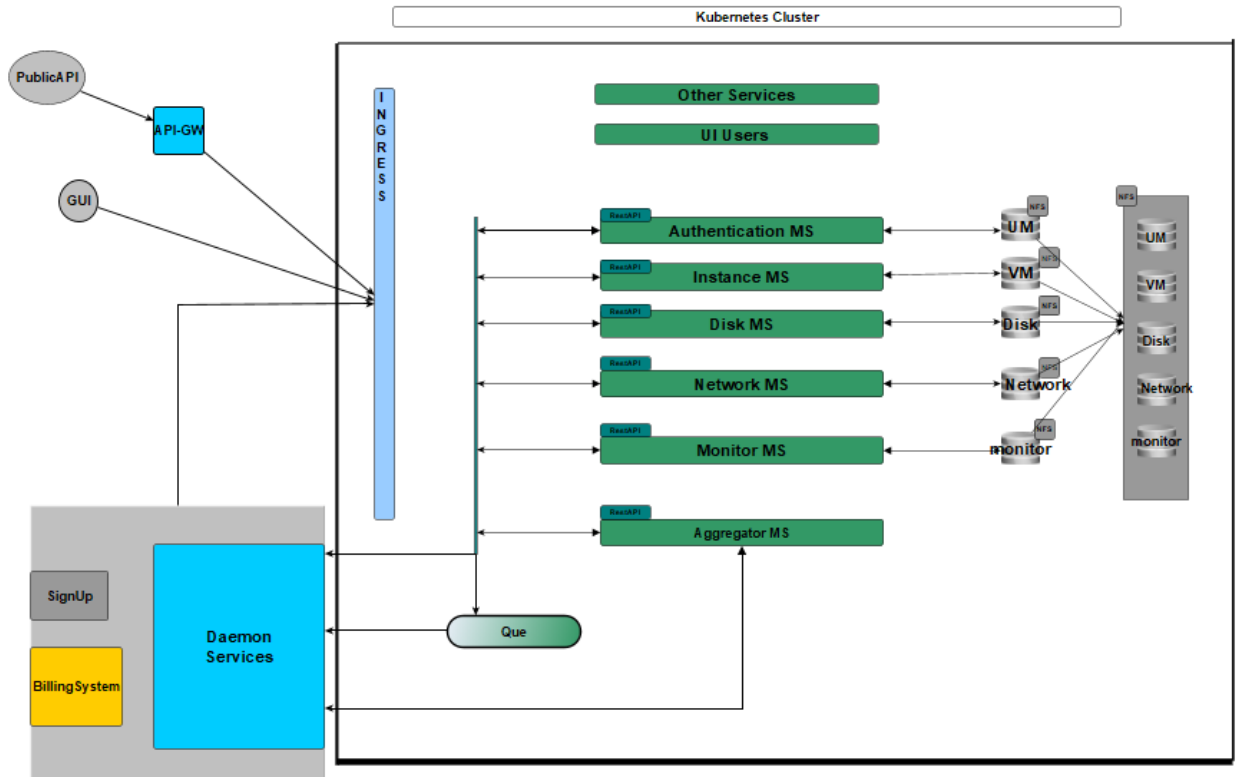



**Figure 1 – NTT PC Web Arena MicroService Overview**

NTT PC Web Arena is Cloud Platform suite of independently deployable, small, modular services in which, each service runs a unique process and communicates through a well-defined, lightweight mechanism to serve business customers.

In Figure 1, MicroServices like Authentication, (Virtual Machine) Instance, Disk, Monitoring, and Network etc. are used.

NTT PC Web Arena GUI implemented Laravel with Lumen framework and NTT PC-We Arena backend MicroServices are built with Lumen.

Public API can access MS (MicroServices) through GateAPI and also GUI users can access MS directly.

All the Front end calls (NTT PC GUI) has to go through GateAPI (aggregator) URL. All the users are first authenticated by gate API (aggregator) and then respective MicroServices which are built using Lumen are invoked.
User Management MicroService (Authentication MS) is responsible for authenticating the user by generating valid authentication token.

Network Management MicroService is responsible for managing and administrating network security system to monitor and controls incoming and outgoing network traffic based on pre-determined security rules by sending requests to the firewall endpoint (user perform to list, create, or delete Firewalls as well as modify access rules). Examples: Firewall fault analysis, performance, provisioning of networks to maintaining the quality of service.

VM Management MicroService (Instance) is responsible for creating VM. User request parameters are sent to the back-end server to authenticate the information to create an Instance with SPEC (CPU, Disk, OS and memory).The backend system accepts the input for creating VM API and sends back the response with status message.

Storage management MicroService is responsible for managing data storage.
Examples: Disk Management and Snapshot management

## 5.4. NTT PC Web Arena Portal – MVC Architecture

NTT PC Web Arena uses Model-View-Controller, which is an architectural industry-standard web development framework for the creation of scalable and extensible NTT PC Web Arena project.

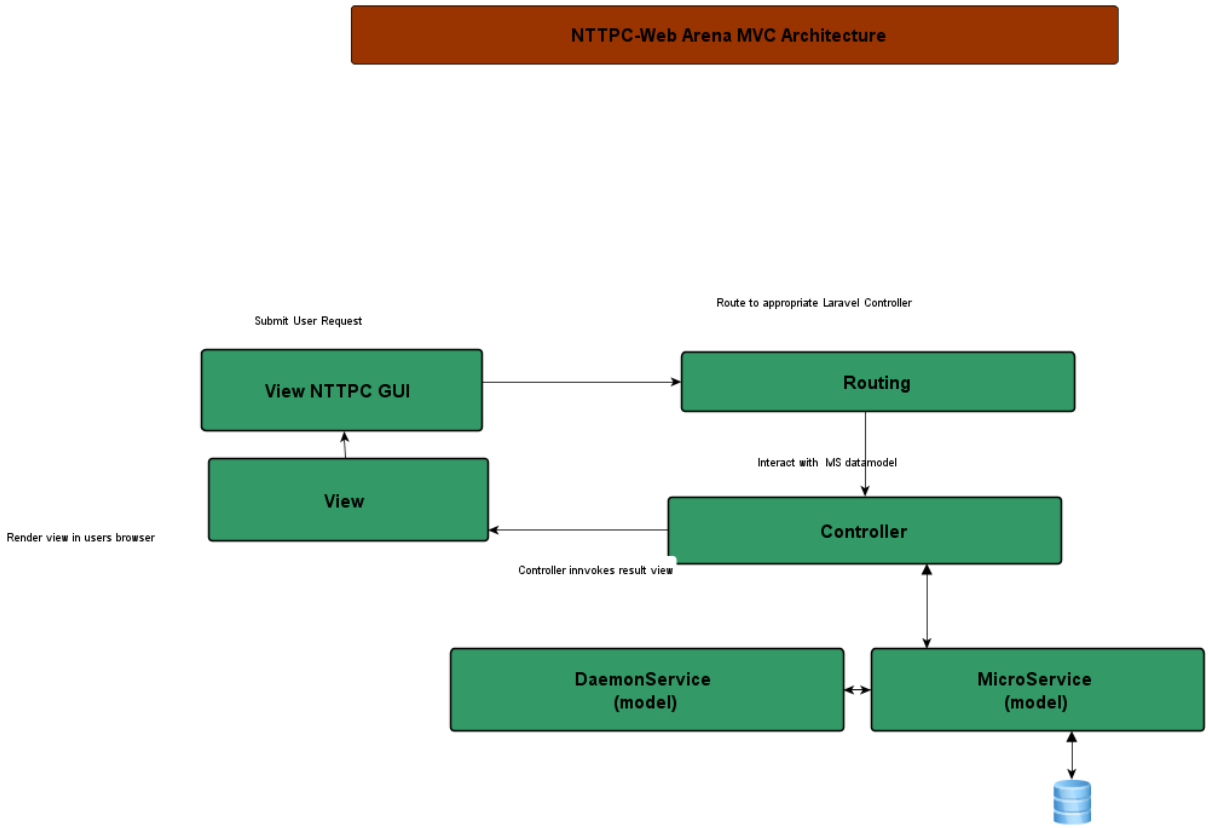



**Figure 2 – NTT PC Web Arena MVC Architecture flow**

Example: NTT PC Web Arena Laravel controllers

An overview of the interactions required to display data on the frontend of the NTT PC Web Arena GUI Portal is illustrated in the Figure 2 given above:

- After receiving an HTTP request from the GUI user, the request consults the Handler Mapping to call the appropriate NTT PC Web Arena GUI Controllers.
- The GUI Controller takes the request and calls the appropriate Service call to Micro services and call methods based on used GET or POST method.
- The Micro-service calls the corresponding methods to sets model data based on defined business logic and returns view through the NTT PC Web Arena GUI controllers.
- The NTT PC Web Arena GUI controllers belongs to the interface between the model (Service business logic) and View components to process all the business incoming and logic request the help NTT PC Web Arena GUI controllers components we are able to manipulate data using the Service component and receive the final results.

# 6. Design Overview

## 6.1. Layered Design and Implementation Technologies

The solution is structured as a multi-tier NTTPS-Web Arena (Simple ISAS) application based on Laravel-Lumen Frameworks along with associated techniques and technologies that run on Apache HTTP Server 2.4.x, MySQL database.

They expose Micro services that are consumed by the NTTPS-Web Arena GUI to retrieve and utilize the content to create VMs'.

Key technology components that Laravel-Lumen framework are as follows:

- Object Oriented (OOP) is mixed up with procedural programming.
- Micro-Service architecture (MSA)
- Laravel Framework
- ORM for Persistence Framework (Laravel)
- Lumen MicroServices

Several clearly defined layers exist and are implemented using PHP, Laravel, Lumen, Apache HTTP Server, and several other Open Source technologies.

The following diagram represents the high-level architecture of the application and how components fit into the various layers.

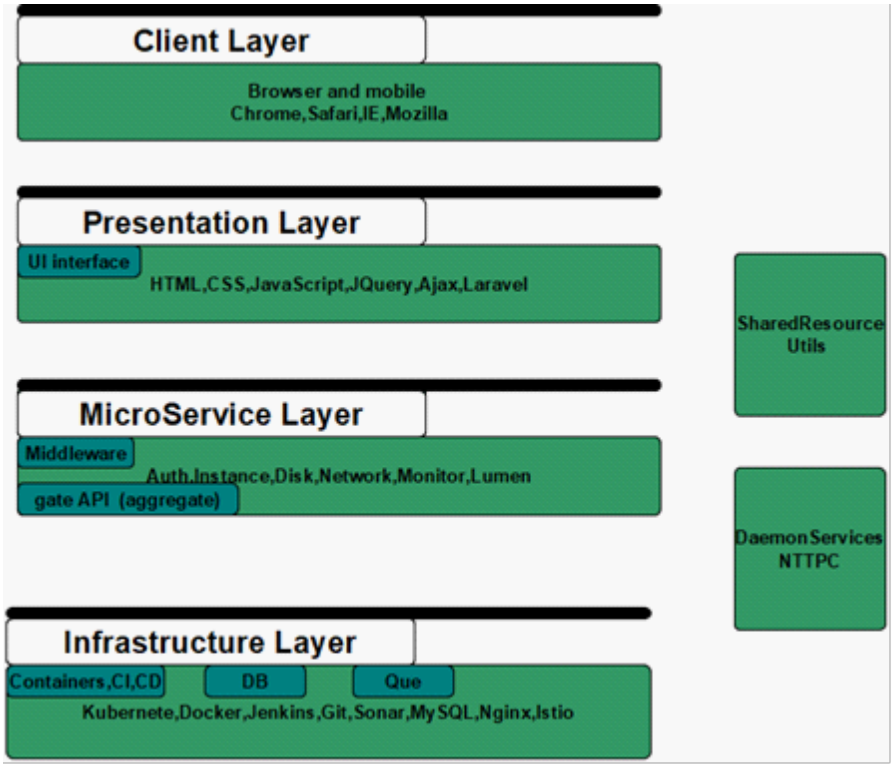



**Figure 3 – Layered Design of NTT PC Web Arena solution**

The building blocks listed in the Figure 3 are organized in an architecture that consists of several clearly defined layers.
The layers in the NTT PC Web Arena application are implemented using PHP, Laravel, Apache Http, and several other open source technologies.

## 1. Client Layer

The Client Layer interacts with remote application clients via MS and API gate services, while the Presentation layer interacts with browsers; together these two layers operate in a parallel relationship.

## 2. Presentation Layer

Presentation Layer consists of web user interfaces. It is implemented as an MVC Web application using Laravel MVC, in conjunction with jQuery used to support the view. The authentication is being handled in this layer via OAuth with MS service.

## 3. MicroService Layer

MicroService Layer consists of the Service models and is the heart of the NTT PC Web Arena server-side application. Models encapsulate the core business data and logic as well as their dependent functionality in an object-oriented manner. The domain models are classified into several types, most of which are PHP beans.

- Entities and value objects are themselves just PHP Objects (plain objects) representing attributes and entities from the database implemented as Lumen (Eloquent built-in ORM) mapped objects. The Lumen toolset helps to generate the objects and mappings from the MS of MySQL database.
- DAO's are implemented as Laravel managed beans that wrap the underlying Laravel ORM persistence implementation. Transaction management is declarative and managed by Laravel, and uses the local Laravel transaction manager.
- DAO's provide flexibility to change an application's persistence mechanism over time without the need to re-engineer application logic that interacts with the Data Access Object tier; implementing a DAO design pattern reduces coupling between Business logic and Persistence logic.
- GateAPI encapsulating the various NTT PC external systems (NTT PC Signup, Daemon Service gateway public) to interact with MS for accessing the NTT PC Web Arena platform.
- Delegate Services is mainly responsible for mapping remote business functionality as local and improving service composition in an MS environment. They encapsulate the 'Remoting' mechanisms and perform necessary data model conversions, which in turn hides the details of distributed computing (such as remote interfaces and special data models) from clients.
- Delegate Services is Lumen managed-beans, wrapping the client end of the MicroService RestAPI Webservices calls.

### 4. Infrastructure Layer

Infrastructure Layer consists of infrastructure applications as Apache HTTP web servers, NGINX, Kubernetes containers, database servers (My SQL), and other supporting infrastructure.

**Shared Utilities** are those components common and reusable across the application:

- **Developer Tests**: Automated testing is implemented using PHPUnit which provides functionality for setting up tests, verifying data MS in the database and other useful re-usable components. Since the project runs within the Kubernetes container, tests can be run inside or outside.
- **Daemon Process**: The communication between Daemon services and the MS will be in synchronous and asynchronous modes.
- **Common**: This is responsible for common utilities used within the application; the following functions have been identified to be shared across the NTT PC Web Arena application.
    - ➢ **Security:** The application interfaces with Oauth to provide user authentication. Within the application, PHP Security and its integration with Lumen MVC is used to provide authentication. The user authentication information is stored on User management DB.
    - ➢ **Common Libraries:** The application logs to a log file using Log4 PHP via a wrapper around the commons-logging info. Utilities libraries used are the combination of NTT PC Web Arena specific utilities classes and open source libraries such as PHP logging
    - ➢ **Error Handling:** Custom error handling is implemented in NTT PC application.

## 7. System Environment

The NTT PC Web Arena application is built on the standard cloud platform structure as described in the following table.

| Specification | Description |
|---|---|
| **Windows Browsers** | Internet Explorer 10 / 11, Firefox 62<br>Chrome on Window69.0.x, Chrome on MacOS 69.0.x, Chrome on Linux69.0.x |
| **Mac Browsers** | Safari 12, Firefox 62.x.x, Chrome (69.0.x) |
| **DB** | MySQL Version 14.14, Distribution 5.7.18 |
| **WebServer** | Apache HTTP Server 2.4.x |
| **Programming and Web** | PHP7.0, Laravel 5.6.x (Lumen 5.6.x) |
| **IDE, API, Data format** | PHP Storm IDE, Visual Studio, JSON, |
| **HTTP Server and Authentication** | Apache HTTP Server 2.4.x, OAUTH2.0, NGINX |
| **UI** | HTML, CSS, JavaScript, jQuery, Ajax, Laravel, RabbitMQ |

## 7.1. Hardware Sizing Requirements

| Environment | Specification |
|---|---|
| **DEV** | • 2 vCPUs, 16 GB memory, 25GB HDD - For Application server (1 count)<br>• 2 vCPUs, 8 GB memory, 50GB HDD - For DB server (1 count)<br>• 2 vCPUs, 8 GB memory, 10GB HDD - For RabbitMQ server (1 count) |
| **QA** | • 2 vCPUs, 16 GB memory, 25GB HDD - For Application server (1 count)<br>• 2 vCPUs, 8 GB memory, 50GB HDD - For DB server (1 count)<br>• 2 vCPUs, 8 GB memory, 10GB HDD - For RabbitMQ server (1 count) |
| **UAT** | • 2 vCPUs, 16 GB memory, 25GB HDD - For Application server (1 count)<br>• 2 vCPUs, 8 GB memory, 50GB HDD - For DB server (1 count)<br>• 2 vCPUs, 8 GB memory, 10GB HDD - For RabbitMQ server (1 count) |
| **PROD** | • 2 vCPUs, 16 GB memory, 25GB HDD - For Application server (1 count)<br>• 2 vCPUs, 8 GB memory, 50GB HDD - For DB server (1 count)<br>• 2 vCPUs, 8 GB memory, 10GB HDD - For RabbitMQ server (1 count) |

## 7.2. CI and CD Specifications Requirements

| Item | Description |
|---|---|
| **Version Control** | Gitlab |
| **Continuous Integration** | Jenkins version 2.x |
| **Source Code Review** | SonarQube 6.4 |
| **Automation Testing** | Selenium 3.14.x, Java1.8 |

| | |
|---|---|
| **Docker** | Docker Hub, Docker version 17.03.2-ce |

## 7.3. OS Specifications

| Specification | Description |
|---|---|
| **Operating System** | Windows 8.1, Windows 10.X |
| **Operating System** | Ubuntu 16.x LTS, Ubuntu 14.x LTS |
| **Operating System** | macOS 10.13.x |
| **Mobile** | Mobile web |

## 7.4. Dependencies

The NTT PC Web Arena application is dependent on several systems and interfaces.

# 8. Technical Dataflow Design

## 8.1. Overview

The following sections describe the design of the NTT PC Web Arena application.

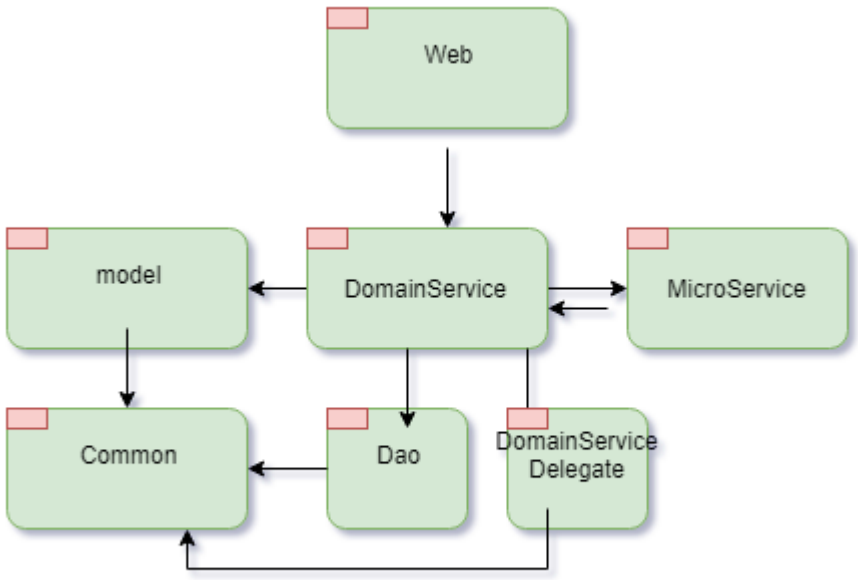



**Figure 4: Dataflow NTT PC**

## 8.2. Web

Web Contains the Laravel UI view, Lumen API, JQuery and JavaScript. Handles view, navigation and delegates to the Domain Services (Laravel components).

## 8.3. MicroServices

All of the backend API calls can be made using MicroServices (Lumen). MS request directly delegates to the Domain Services.

## 8.4. DomainServices

DomainServices invoke respective services based on the kind of request coming to the Web Server.

## 8.5. DomainServiceDelegates

Ensures services related beans are loaded in the application.

## 8.6. DomainDAO

DAO implemented using Laravel, provides the DAO classes to the Domain Services package to retrieve and manipulate data in the MS databases.

## 8.7. DomainModel

DomainModel PHP objects represent the data model.

## 8.8. Common

Contains common classes and logs etc.

## 8.9. Sync and ASync Operation Flow

### 1. Sync Operation Flow to Start Instance

Aggregate MicroService (Gate-API) mainly aggregates several data from several MicroServices and updates several databases. Each MS requests should use HTTPS (through “Daemon Proxy API”) to communicate Daemons in secure way.

To start VM

- User authenticated against authenticated MicroService and then calls jobVMstart () method on Instance MicroService to start job and then updates VMS table.
- Daemon calls aggregate api to collect Disk information.
- If everything goes well daemon return success and shows in running state to User.
- As per the needs Daemon calls the aggregate MS to get network and monitoring information.

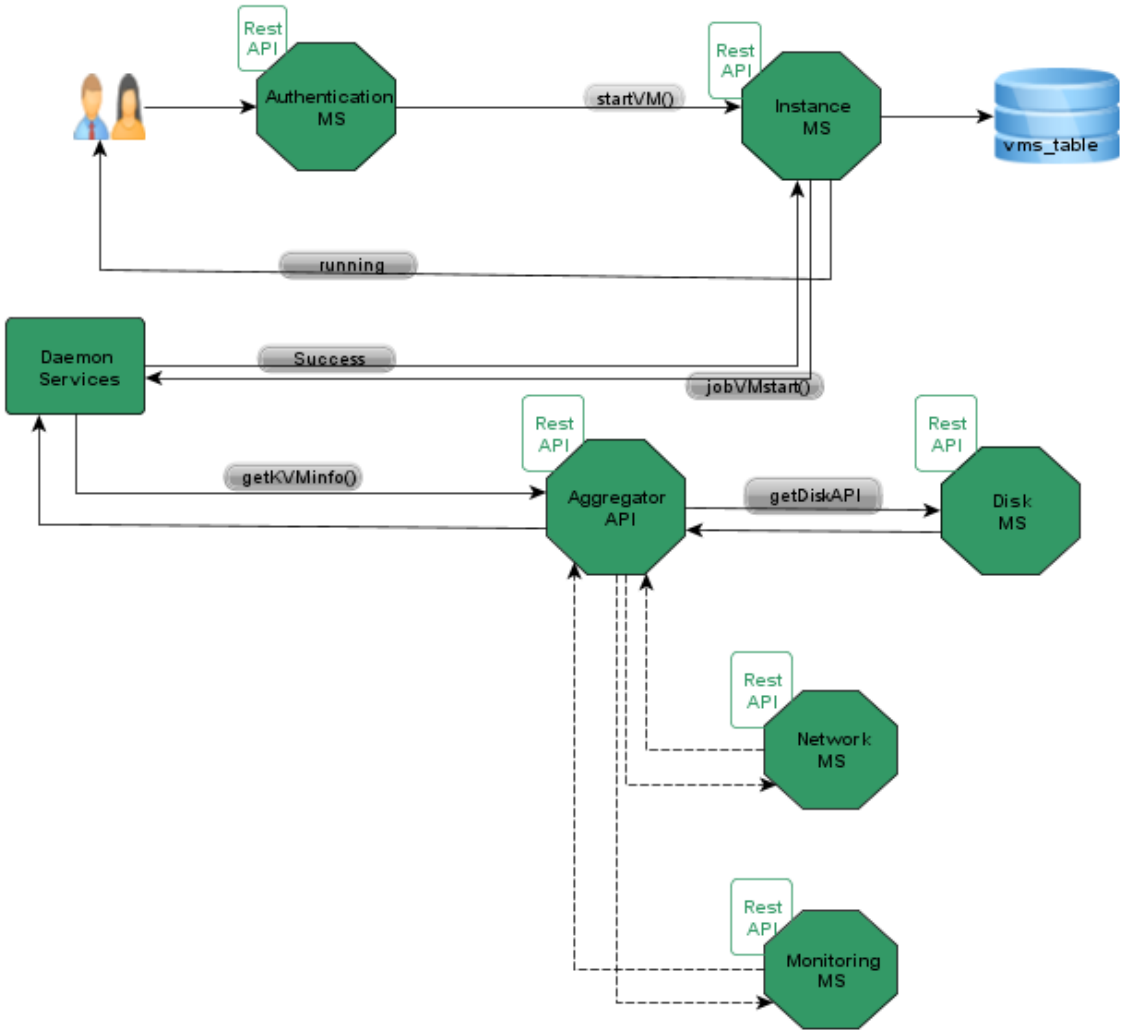



**Figure 5: Sync Dataflow NTT PC**

## 2. Async Flow to create Instance from OS Image

Aggregate MicroService (Gate-API) helps to aggregate data from several databases and updates several databases. Each MS can request to Daemons using HTTPS via Daemon Proxy API.
Once the user credentials are validated, Auth MicroService will provide the access token and then will get validated in the middleware and allow user request (Calls the instance Specification, OS Image and SSH key API on Instance MS) to call Instance MS to create an Instance.

Corresponding DB tables Vms (To reserve an unused instance), Host (To choose most vacant server), IPv4 (To reserve an IP to assign to the instance) are updated and then calls the Disk MS (to create a boot disk to launch a server) to get the disk information and then updates the Disk and Instance DB tables.

- When creating Instance JobInitOs methods are called through Async Queue (once the order is sent through queue, it performs without waiting for host server response.) to daemon.

- Daemon call (get_kvm_resource)aggregate api to collect KVM resource details.
- Daemon call (Get_kvm_information)aggregate api to collect KVM information.
- Daemon call(get_disk_information) aggregate api to collect Disk information.
- Daemon call (change_otherstatus_os ) aggregate api to Change KVM OS installation status.
- Daemon call(change_otherstatus) aggregate api to Change KVM Other status.
- Daemon call (change_disk_status) aggregate api to Change disk status.

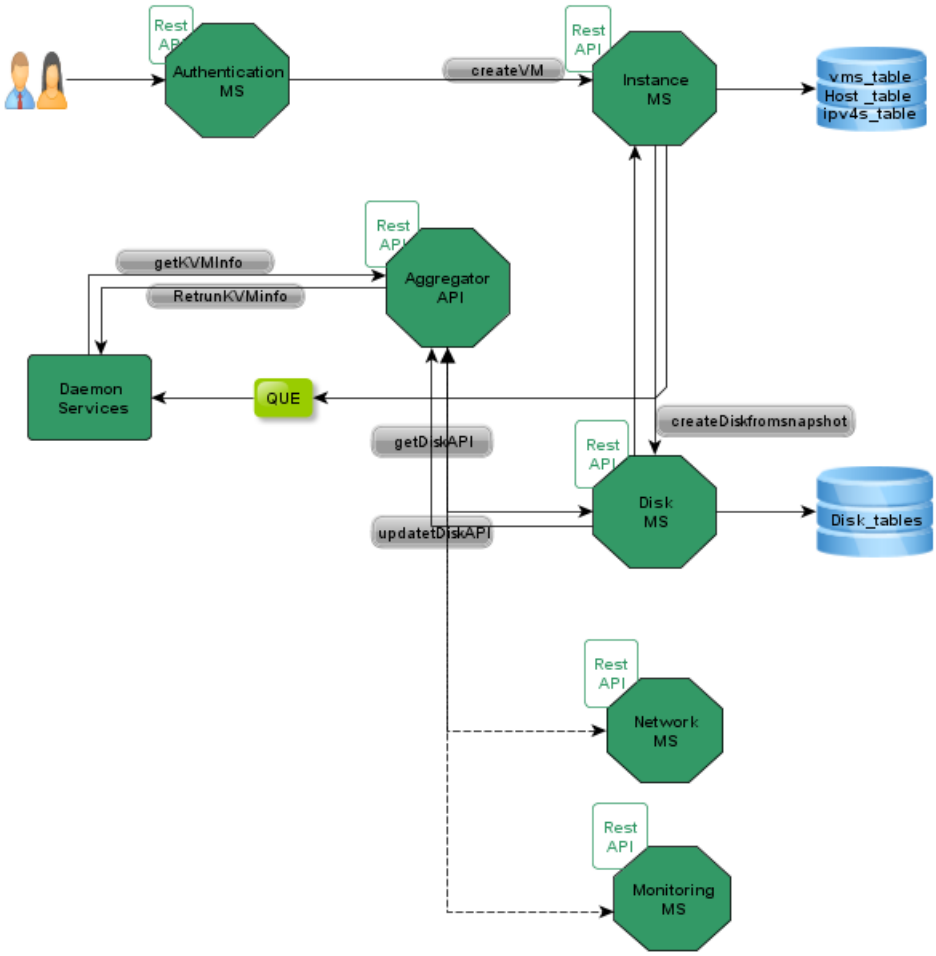



**Figure 6: ASync Dataflow NTT PC**

# 9. Source Layout

## 9.1. Project Directory Hierarchy

The source code of the NTT PC Web Arena application is organized into a hierarchy of modules, under the */App* directory. The hierarchy consists of sub-directories, dividing NTT PC Web Arena into various loosely-coupled sub-projects.

The sub-directories divide the NTT PC application into modules that contain different technical building blocks following the layered architecture described in previously in the Layered Design section.

These sub-directories are as follows:

The NTT PC Web Arena building lumen framework is light and simple. Below is the project structure

```
├── app
├── artisan
├── bootstrap
├── composer.json
├── composer.lock
├── phpunit.xml
├── public
├── readme.md
├── server.php
├── storage
├── tests
└── vendor
```

- **/app**: contains all the main source code for the project.
- **artisan**:commands *Laravel-ers*s
- **/bootstrap**: project starts here. The main **$app**instance of the project.
- **/public**: well, same thing in **Laravel** again.
- **/storage**: contains compiled Blade templates, file based sessions, file caches, and other files.
- **/tests**: all the tests located here.
- **/vendor**: contains Dependency Manager for PHP
- **server.php**: to start the php server from **artisan** command.

/app directory

The**/app** directory looks like.

app

```
├── Console
│   ├── Commands
│   └── Kernel.php
├── Exceptions
│   └── Handler.php
├── Http
│   ├── Controllers
│   │   └── *Controller.php
│   ├── Middleware
```

```
│   │   └── *Middleware.php
│   └── routes.php
├── Jobs
│   └── Job.php
└── Providers
    └── AppServiceProvider.php
```

## 9.2. Exceptions

Exceptions contains exception handlers of the application and stores the exceptions thrown by the application

## 9.3. Middleware

User authentication token is validated in the Middleware application by passing parameters with middleware (MS logic).

## 9.4. Views

Views are stored in resources/views

## 9.5. Controllers

All controller classes are placed under

- “app/Http/Controllers/UserController.php”,
- "app/Http/Controllers/VMController.php”, etc.

## 9.6. Project Layout

The base namespace package for the PROJECT NTT PC Web Arena code base is NTT PC Web Arena.

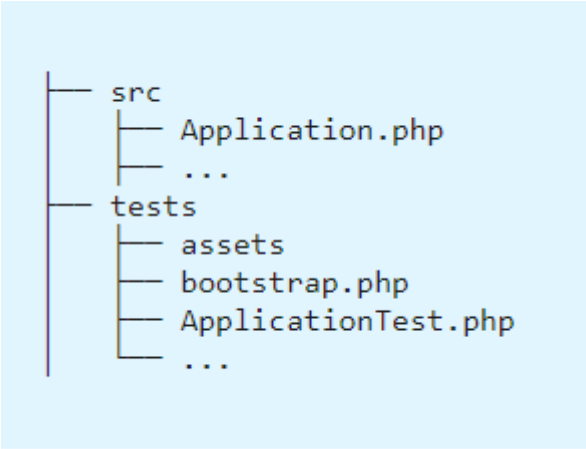
# 10. Testing Strategy with TDD

Automated testing is implemented using PHPUnit6, which provides functionality for setting up tests, verifying data in the MicroServices, DB and other useful re-usable components. Since the NTT PC Web Arena project runs within the Apache container/Kubernetes, tests can be run inside or outside Apache Containers. PHPUnit tool helps developers to perform unit testing and practice test-driven development.

Below are some of the PHPUnit assertions helps to write tests:

*assertTrue()*
*assertFalse()*
*assertEquals()*
*assertNull()*
*assertContains()*
*assertCount()*
*assertEmpty()*

Below is the Sample project layout (Composer project layout)




As per the above src/tests, Development Teams separate the test code from the source code.

All the test codes are not pushed in the Prod environment. Only test codes that are necessary to run the NTT PC Web Arena application are deployed. The test codes provide support to identify the functionalities or requirements that are not working.

The goal of TDD is to have the code clearer, simple and bug-free. Test-Driven Development starts with designing and developing tests for the functionalities of the NTT PC Web Arena application. In TDD approach, we start with the test case implementation, which specifies and validates what code will be implemented.

TDD helps to write and correct the failed tests before implementing new code (before development). This also helps to avoid duplication of code. Here tests are nothing but requirement conditions that need to be tested the business requirements.

## 10.1.Test Driven Development Process Cycle

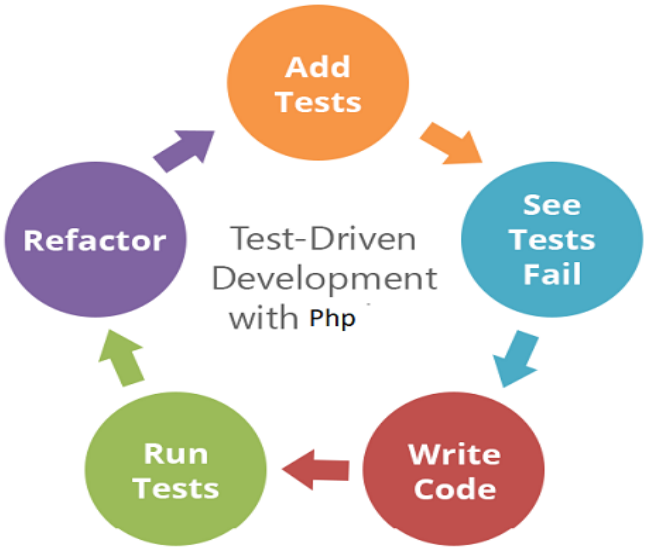



**Figure 7: TDD Cycle**

### 3. TDD in steps

As shown in the above flow

1. Add tests for the functionality.
2. Run tests to fail.
3. Write code according to the identified errors.
4. Run the tests again to see if the test fails or passes
5. Re-factor the code and follow the process again.

## 10.2.Benefits of Test Driven Development

Practicing TDD brings lots of benefits. Some of the benefits are listed below -

- Test case is built, before adding any new feature. This helps to cover the entire code under the test. This is a great benefit of TDD as compared to the code which has no test coverage.
- One should have a specific target before adding new functionality. This means before adding any new functionality, one should be clear about its outcome.
- In the application, one method may depend on the other. When tests are implemented, the team must have clear business requirements about the interfaces between the methods. This allows integrating the methods with the entire application efficiently and helps in making the application modular.

# 11. Build Process

## 11.1. Jenkins

Jenkins is a continuous integration and build management system. It can be integrated with the underlying build system, to perform scheduled builds of the project. In case of NTT PC Web Arena, we are using composer to install dependencies. Jenkins will be integrated with PHP application to perform scheduled builds.

## 11.2. Jenkins Build Job

DevOps team will create individual build jobs for the NTT PC Simple IaaS Dev CI. Whenever the Development team raises new request for branch (development or any bug fixes), DevOps team shall create Jenkins build job for the request.
Jenkins Build URL: http://203.138.94.25:8080/

## 11.3. Jenkins Job Naming standards

DevOps shall configure the Jenkins CI jobs for both Simple IaaS DEV and QA environments.

| Environment | Job Name | Job URL |
|---|---|---|
| QA | qa_Web Arena_<msname> (like qa_Web Arena_ui) | http://203.138.94.25:8080/job/qa_Web Arena_ui/ |
| Dev | dev_Web Arena_<msname> (like qa_Web Arena_ui) | http://203.138.94.25:8080/job/dev_Web Arena_ui/ |

DevOps team shall create individual job for each MicroServices. The MicroService job is created as per above standard. For example, the UI MicroService job name should be **qa_Web Arena_ui**

## 11.4. Helm and Docker images

Helm chart is a collection of files to describe a related set of Kubernetes resources. HelpChart is used to deploy NTT PC Web Arena application stack with HTTP servers, databases, and so on

## 11.5. GITlab

GIT acts as the code repository for the project. All the source code for the project is stored in Git.

- DevOps team shall maintain and control the Server.
- DevOps team shall define roles and give access to the repository.
- Roles shall be defined based on the requirement.

Use the https://d-crs-gitlab1.sphere.ad.jp/Web Arena_SimpleIaaS/ URL to access the GITlab SimpleIASS repository.

## 11.6. Branching Strategy

Branch operation is used to create another line of development. It is useful, when the development process forks off into multiple different directions.
For example, when version 1.0 is released, a branch need to be created so that development of 2.0 features can be kept separate from bug fixes of version 1.0.

In the Simple IaaS project, we shall be adopting the below branching strategy:

1. The DevOps team shall create development branch for each MicroService **wa_<ms>_dev** from Master branch and Development Branch Owner shall create sub branch **wa_<ms>_dev comp1** for their individual component development.
2. The Developer shall validate their code and send merge request to Developer lead. The Lead shall validate and merge code to development branch **wa_<ms>_dev**.
3. After the validation, the code shall be ready to be pushed for QA validation. The Developer shall send merge request to **wa_<ms>_qa** branch.
4. The QA branch owner shall validate and merge it to **wa_<ms>_qa** and QA shall start to validate their code from this branch
5. After completing the QA validation, it shall be ready to be move into pre-production branch **wa_<ms>_stg**.
6. The QA Branch Owner shall send merge request to pre-production branch owner.
7. After the review, the **wa_<ms>_qa** branch code merges to **wa_<ms>_stg**
8. After successful validation on **wa_<ms>_stg** branch, the code shall be ready to be moved into Production **master** branch.
9. Once the code is pushed to Production branch, DevOPS team shall create release tag **wa_<ms>_release_<number>** for reference.
10. The hot fix/bug fix shall go through **wa_<ms>_release_<number.**

The pictorial representation of the Branching Strategy for Simple IaaS is given below.

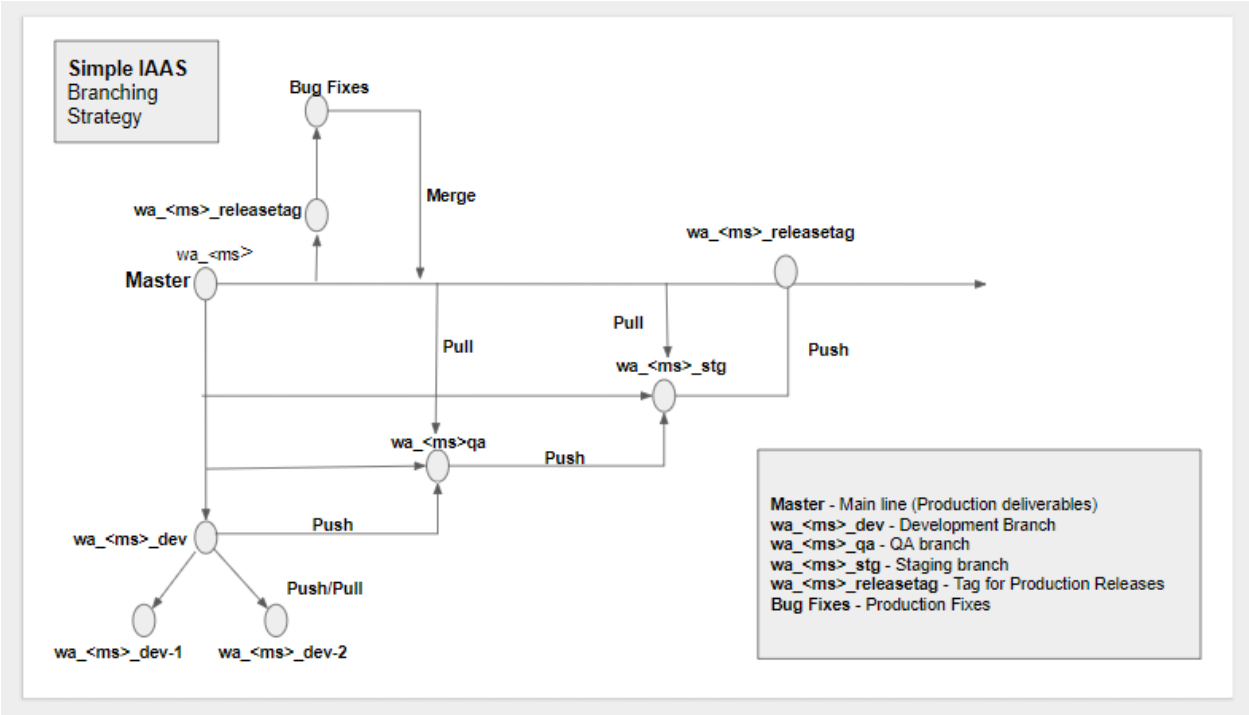



**Figure 8: Branching Strategy**

## 11.7.SonarQube

SonarQube is an open source platform for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on programming languages. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.

## 11.8.SonarQube Integration

SonarQube integration work is implemented in the application.

## 11.9.Kubernetes

Kubernetes is a Container Management System meant to be deployed on Docker-capable clustered environments.

# 12.NTT PC Web Arena (CI/CD) Deployment Process

## 12.1.MicroService Build and Deployment Architecture

The Simple IaaS application is split as MicroServices and performs development activities on specific MicroServices. The DevOps team creates source code repository, Jenkins build and deployment job for individual MicroServices.

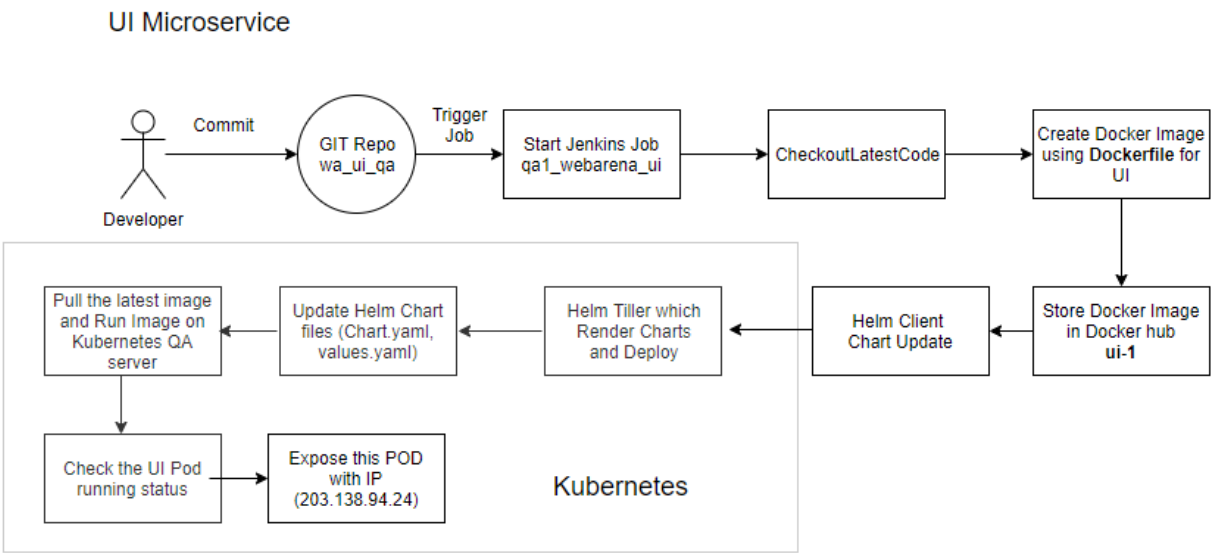
The DevOps team maintains the QA environments, Servers (nodes) under Kubernetes cluster. Jenkins pulls the latest MicroService code from the QA branch (wa_<ms>_qa) and creates Docker image using the Docker file which was created by DevOps and pushes to Docker hub repository.

For QA deployment, there is separate job for deployment. The deployment job connects to the Helm Server and triggers the deployment script. This shall update the helm chart files (Chart.yaml, Values.yaml) with latest build image and perform the **POD** deployment on Kubernetes cluster. The application is exposed outside through Ngnix ingress controller.

The below mentioned MicroServices build and deployment implementation completed on QA server (Kubernetes cluster)

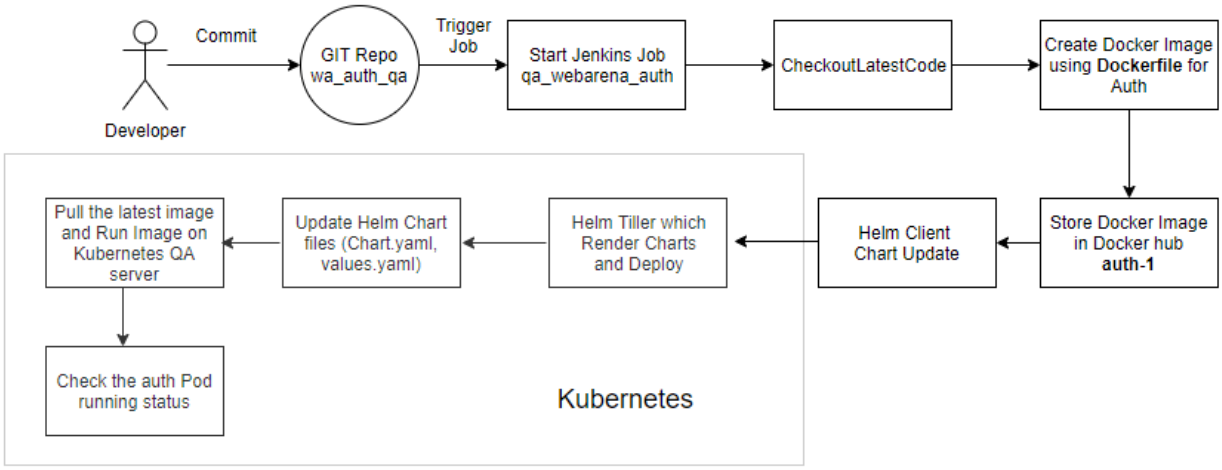
- UI MicroService
- Auth MicroService
- VM MicroService
- Disk MicroService
- Database MicroService
- NTT PC-Operator MicroService

The below pictorial representations provide the detail about the NTT PC SimpleIaaS MicroService image build and Pod deployment on Kubernetes cluster.

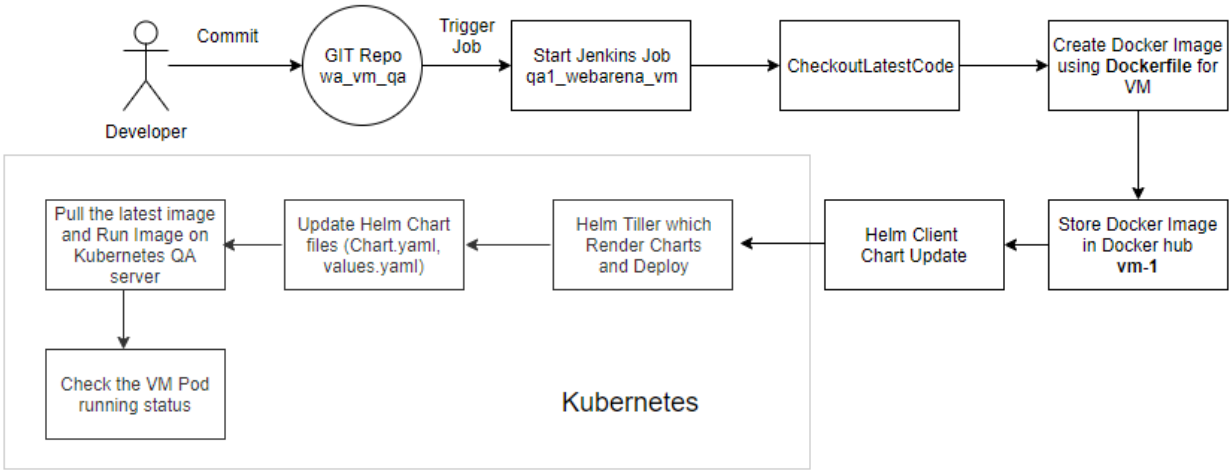


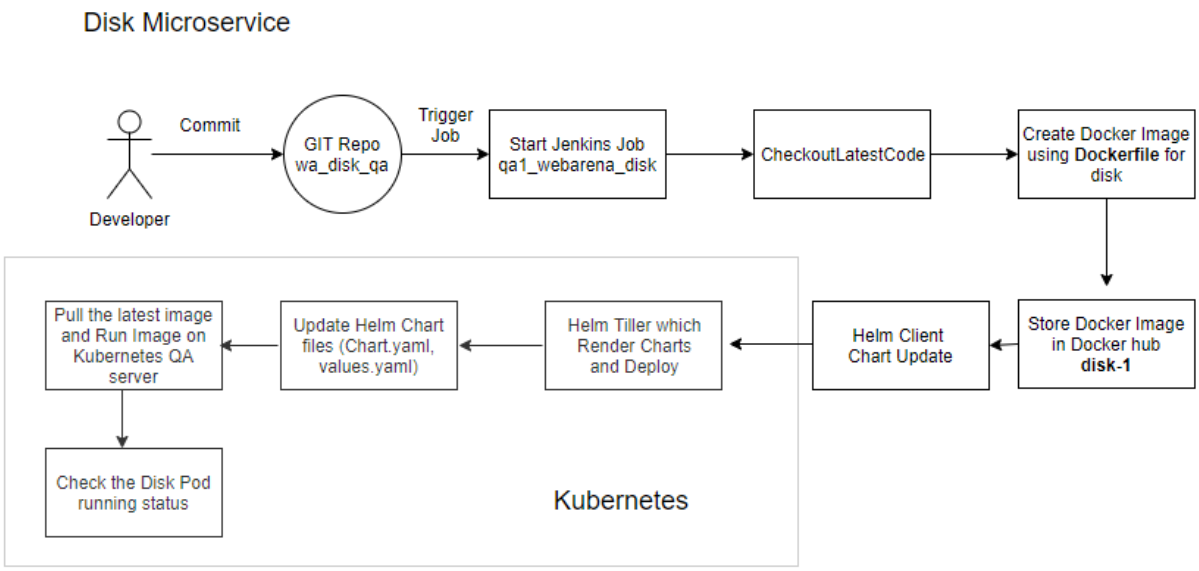

The UI MicroService is exposed through Nginx ingress Controller, https://203.138.94.24

Auth Microservice




VM Microservice






NTT PC Web Arena application shall be deployed in four environments namely DEV, QA, UAT and PROD.

## 12.2.Development Environment

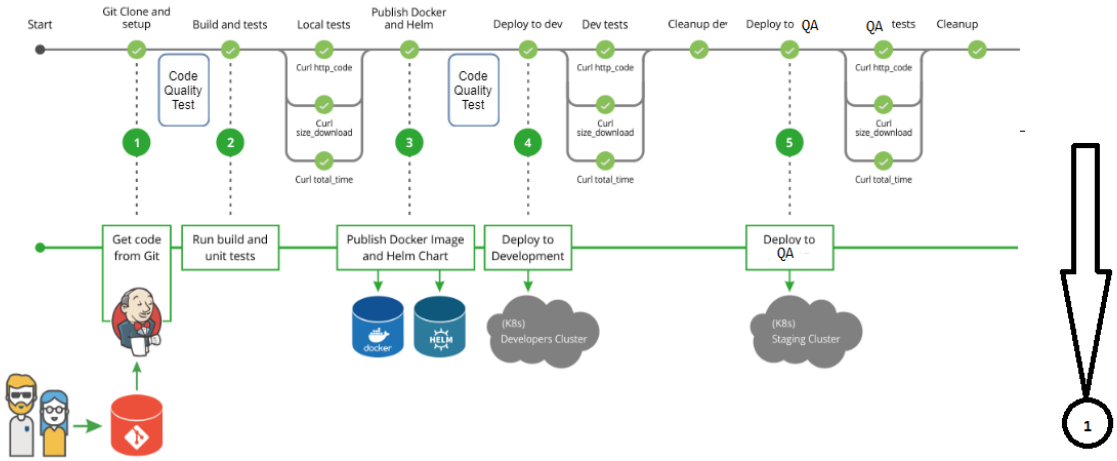



**Figure 9: Continuous Integration and Deployment Flow**

- Developer pushes code to Git, which triggers a Jenkins build webhook.
- Jenkins pulls the latest code changes.
- Run build and unit tests
- Jenkins runs the build.
- Application’s Docker image is created during the build.- Tests run against a running Docker container.

- Publish Docker image and Helm Chart.
- Application's Docker image is pushed to the Docker registry.
- Helm chart is packed and uploaded to the Helm repository.
- Deploy to Development.
- Application is deployed to the Kubernetes development cluster or namespace using the published Helm chart.
- Tests run against the deployed application in Kubernetes development environment.

## 12.3.QA Environment

- Jenkins pulls the latest code changes in QA
- Run build and unit tests
- Jenkins runs the build.
- Application's Docker image is created during the build.- Tests run against a running Docker container.
- Publish Docker image and Helm Chart
- Application's Docker image is pushed to the Docker registry.
- Helm chart is packed and uploaded to the Helm repository.
- Deploy to QA
- Application is deployed to Kubernetes QA cluster or namespace using the published Helm chart.
- Run tests against the deployed application in the Kubernetes QA environment.

## 12.4.UAT Environment

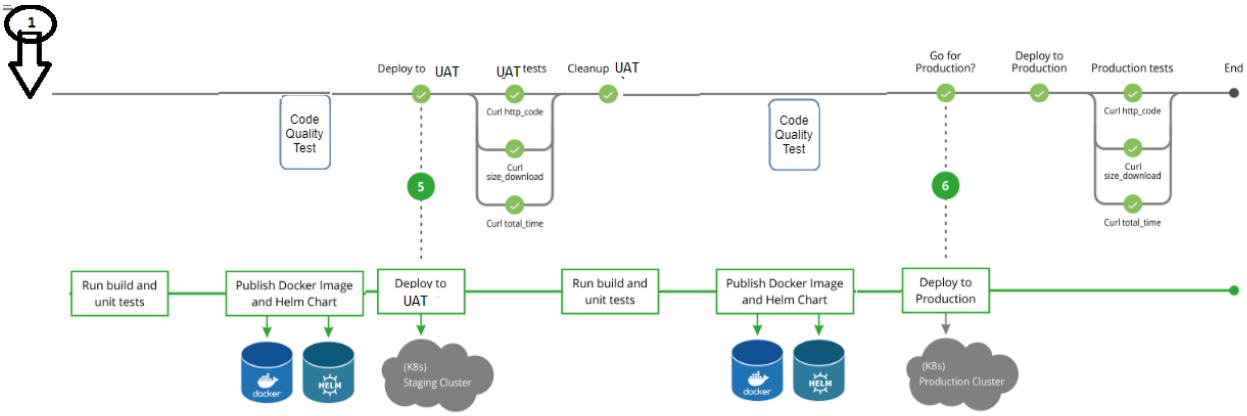


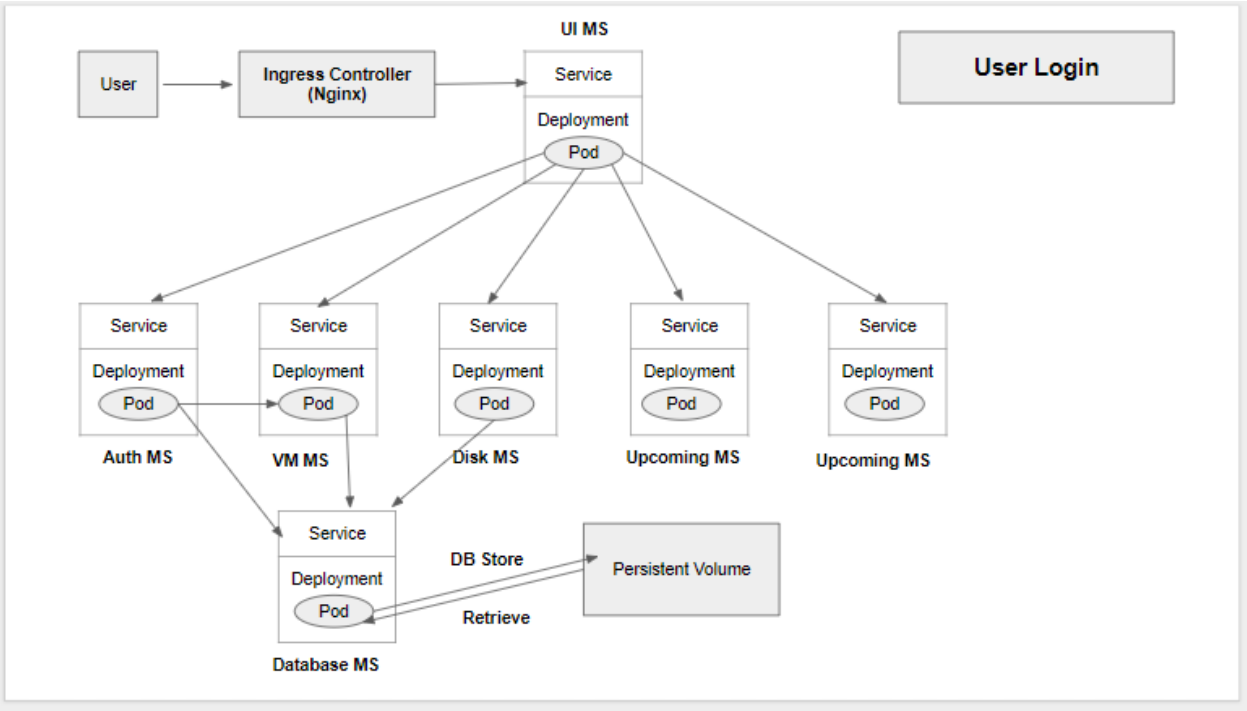
**Figure 10: Continuous Integration and Deployment Architecture**

- Jenkins pulls the latest code changes in UAT
- Run build and unit tests
- Jenkins runs the build.
- Application's Docker image is created during the build.
- Tests run against a running Docker container.
- Publish Docker image and Helm Chart
- Application's Docker image is pushed to the Docker registry.
- Helm chart is packed and uploaded to the Helm repository.
- Deploy to UAT
- Application is deployed to Kubernetes UAT cluster or namespace using the published Helm chart.
- Run tests against the deployed application in the Kubernetes UAT environment.

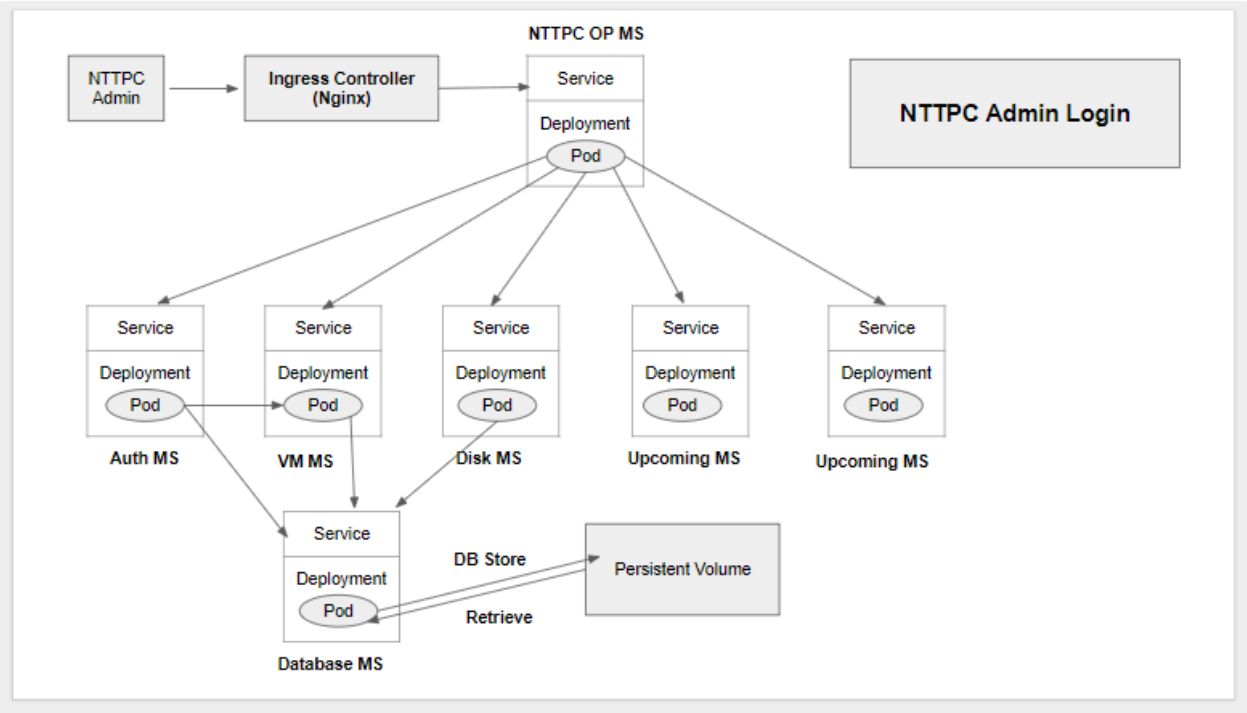
## 12.5.Production Environment

- Deploy to Production
- The application is deployed to the production cluster, if the NTT PC Web Arena application meets the defined criteria.
- (manual approval step)
- Sanity tests run against the deployed prod NTT PC Web Arena application. if required, we can perform a rollback

## 4. Kubernetes Architecture for Simple IaaS Application (User Login)




## 5. Kubernetes Architecture for Simple IaaS Application (NTT PC Operator Login)




# 13. Appendix A – Glossary

Definitions, acronyms and abbreviations used as part of this document are mentioned below:

| Abbreviation / Acronym | Expansion |
|---|---|
| API | Application Program Interface |
| CIDR | Classless Inter-Domain Routing |
| CPU | Central Processing Unit |
| DB | Database |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| ISP | Internet Service Provider |
| ICMP | Internet Control Message Protocol |
| JSON | JavaScript Object Notation |
| NTT PC | NTT PC Communication Co. Ltd |
| Oath | Authorization and Authentication |
| OS | Operating System |
| SSH | Secure Shell |
| SMS | Short Messaging Service |
| SPEC | Specification |
| TFA | Two Factor Authentication |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| VPS | Virtual Private Server |
| UI | User Interface |
| XML | Extensible Mark-up Language |
| URL | Uniform Resource Locator |
| CSS | Cascading Style Sheets |
| CURD | Create, Read, Update and Delete |
| MVC | Model View Controller |
| MS | MicroServices |
| Apache | Apache HTTP Server |
| MSQL | My Structured Query Language |
| PHP | Personal Home Page |
| ORM | Object Relational Mapping |
| WAMP | Windows, Apache, MySQL, and PHP |

| Abbreviation / Acronym | Expansion |
|---|---|
| DBMS | Data Base Management System |
| SQL | Structured Query Language |
| CI | Continuous Integration |
| CD | Continuous Deployment |
| SCM | Software Configuration Management |
| QA | Quality Assurance |
| Dev | Development |
| Stg | Staging |
| HDD | Hard Disk Drive |