Change Log

| **Revision** | **Submission Date** | **Affected Sections or Pages** | **Change Summary** |
| --- | --- | --- | --- |
| Initial | 06/11/2019 | All | Initial issue of document. |
| 1 | 07/10/2019 | All | Addressing review comments |

Table of Contents

1 Overview 5

1.1 Purpose 5

1.2 Terminology and Notation 5

1.3 References 5

2 Minimum System Requirements 6

2.1 Software Requirements 6

2.2 Supported Browsers 6

2.3 Hardware Requirements 6

2.4 TCP Port Requirements 6

3 Installation 7

3.1 Overview 7

3.2 Prerequisites 7

3.3 Basic Steps 7

3.4 Authentication 8

3.5 Artifactory Settings 8

4 Configuration 8

4.1 Basic Configuration 8

4.2 Analytics Configuration 9

4.3 Tyk Configuration 10

5 Deployment 11

5.1 Steps 11

5.2 Configuration 11

6 Troubleshooting & Defect Reporting 12

# Overview

## Purpose

This document will guide a user through installing AERIE. AERIE is a new software system being developed by the MPSA element of Multi-mission Ground System and Services (MGSS), a subsystem of AMMOS (Advanced Multi-mission Operations System). AERIE will support activity planning, sequencing, and spacecraft analysis of mission operations. This guide will be updated as features are added.

## Terminology and Notation

|  |  |
| --- | --- |
| Term | Meaning |
| AMMOS | Advanced Multi-Mission Operations System |
| MGSS | Multi-Mission Ground Systems and Services |
| NEST | Front End components of AERIE |
| YAML | Markup language |
| Docker | Software platform for building applications based on containers |
| Artifactory | Repository manager |
| LTS | Long-Term Support. Refers to a version of software that has long-term support. |
| NPM | Node Package Manager |

## References

Table 1: Applicable JPL Rules documents

|  |  |
| --- | --- |
| Title | DocID |
| Software Development | 57653 |

Table 2: Applicable MGSS documents

|  |  |
| --- | --- |
| Title | DocID |
| MGSS Implementation and Task Requirements | DOC-001455 (Rev C) |
| Aerie Software Requirements | DOC-002388 |
| Aerie Concept of Operations | DOC-002387 |
| NEST Users Guide | DOC-002380 |
| NEST Software Design Document (SDD) | DOC-002312 |
| NEST Software Requirements Document (NEST SRD) | DOC-001934 |

# Minimum System Requirements

## Software Requirements

|  |  |
| --- | --- |
| Name | Version |
| NodeJS | 10.16.0 LTS |
| NPM | 6.10.0 |
| Open JDK | 8.x |
| Docker | 10.09.x |
| Git | 2.x |

## Supported Browsers

|  |  |
| --- | --- |
| Name | Version |
| Chrome | Latest LTS |
| Firefox | Latest LTS |

## Hardware Requirements

|  |  |
| --- | --- |
| Hardware | Details |
| CPU | 2 gigahertz (GHz) frequency or above |
| RAM | 4 GB at minimum |
| Display Resolution | 2560-by-1600, recommended |
| Internet Connection | High-speed connection, at least 10Mbps |

## TCP Port Requirements

All ports below should be configured to accept incoming connection from clients in Aerie.

|  |  |
| --- | --- |
| Service | Port |
| Adaptation | 27182 |
| Adaptation-runtime | 27184 |
| Nest | 8080 |
| Plan | 27183 |
| Simulation | 27185 |
| Sequencing | 27186 |
| Tyk Gateway | 8081 |

# Installation

## Overview

The Aerie project uses Docker Compose to configure and run the entire project. Docker Compose utilizes YAML configuration files which are located in the root of the project. The YAML configs include everything that is required to run Aerie with Docker and Docker Compose.

“Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a Compose file to configure your application’s services. Then, using a single command, you create and start all the services from your configuration.”

In most cases, Docker Compose will pull down pre-built Docker containers for each of the servers which make up the AMPSA application. These containers are built at runtime, but will eventually be retrieved from our local JPL installation of DockerHub which is hosted on Artifactory. Artifactory is a repository that hosts project artifacts such as build results and Docker images.

Once the necessary containers are pulled down, Docker Compose will orchestrate the network of services and start them up.

Important note: This demo is NOT designed for production use or performance testing. It has not been optimized in any way. It WILL be slow and should not be considered for release. Per Tyk recommendations, a production configuration would have each component on a separate machine, please see the Tyk documentation (<https://tyk.io/docs/>) for more detailed information.

## Prerequisites

To start the services using Docker Compose you must have Git, Docker and Docker Compose installed.

## Basic Steps

The basic steps are:

1. Clone the repository using Git
2. Log in to Docker Hub
3. Create an Artifactory settings.xml file for local development
4. Configure services (if running the full configuration)
5. Start the services using the appropriate configuration file

## Authentication

Before the local and full configurations can be run, you will need to login to Artifactory:

docker login -u $USERNAME cae-artifactory.jpl.nasa.gov:16001

Where $USERNAME is your LDAP username. You will be prompted to enter your password. Once successful, you are logged in and can pull down MPSA images.

NOTE: Access is intended for MPSA Aerie Developers. If you have tried to login and you get an access denied message, please contact [seq.support@jpl.nasa.gov](mailto:seq.support@jpl.nasa.gov) to request access.

## Artifactory Settings

You need an Artifactory settings.xml file with your JPL credentials so you can download and use Artifactory packages locally. Follow these steps to set this file up:

1. Go to <https://cae-artifactory.jpl.nasa.gov/artifactory/webapp/#/artifacts/browse/tree/General/maven-libs-snapshot-local> and log in with your JPL credentials
2. Click "Set Me Up"
3. Type in your JPL password
4. Use the settings shown below. 
5. Click "Generate Maven Settings" and then "Generate Settings" which will download a settings.xml file
6. Move settings.xml into ~/.m2/, replacing the old settings.xml if it exists.

# Configuration

## Basic Configuration

The basic command that you will use to start services with Docker Compose is:

docker-compose up

This would start the default configuration which is tailored toward local dev. To start additional services, you will use a specific configuration like this:

docker-compose [-f <docker-compose-ARG>.yml] up -d

The <docker-compose-ARG>.yml roughly corresponds to the environment that you may require, as outlined below. The -d option will run the services in the background. To get a feel for how the service works though, you can omit it.

The services may take several minutes to start. Once the logging output from docker-compose has slowed down, you should be able to access the services as per their individual documentation.

Press cmd+c or ctrl+c to stop all services if docker-compose is running in the foreground. Or to stop an application which has been started in the background, run the following:

docker-compose [-f <docker-compose-ARG>.yml] stop

Where the <docker-compose-ARG>.yml file corresponds to the environment which you started the application with.

### docker-compose.yml

This configuration is intended for local development where you do not want to actively develop the core services. This could be useful as a starting place for local development where the developer may comment out sections that they are working on. In this environment images are retrieved from Artifactory.

### docker-compose-full.yml

This is the entire stack, including logging, analytics, the full API gateway dashboard, and all of the core services. This is intended for use in a production environment, with a host that has a healthy amount of RAM, CPU, and disk space. It can be run locally, but may not provide great performance.

**Additional configuration required**

Analytics and Tyk Dashboard have additional configuration to make them work. See the relevant sections below to get them set up properly.

## Analytics Configuration

In the analytics directory there is a file called analytics.env-example. This file needs to be renamed to analytics.env. This will eventually enable the ability to create a custom database settings. For now, however, leave the settings as is.

Once the service has been started, you will need to run the setup script in the analytics directory.

cd analytics

./setup.sh

This will initialize the database that will be used by Matomo.

## 

## Tyk Configuration

To start the Tyk stack you will first need to do some minor configuration. Then you can use the tyk and tyk\_local files. See step 3, below.

1. Set up your /etc/hosts file to include the IP of your docker daemon:

127.0.0.1 www.tyk-portal-test.com

127.0.0.1 www.tyk-test.com

Note that the IP may be different depending on your installation, Windows users may find it running on 10.x.x.x, it is important the URL stays the same because our setup.sh assumes this is the one you are using.

1. Add your dashboard license

Open the tyk\_dashboard/tyk\_dashboard.conf file and add your license string to the "license\_key": "" section.

1. Start the full stack

docker-compose -f docker-compose-full.yml up

1. Bootstrap the instance (creates a test user):

cd ../tyk\_dashboard

chmod +x setup.sh

./setup.sh

Make sure you save the passwords somewhere. Optionally, create a user with the methods used in the setup.sh file.

1. Log in with the credentials provided.

The setup script will provide a username and password, as well as the URL of your portal, please note that this will be running on port 3000, not port 80.

# Deployment

This document describes how to deploy the Aerie services and Nest front-end via Docker. All of these instructions should be carried out on the machine you are installing to.

## Steps

1. Make sure Docker and Git are installed, and that Docker is running. docker-compose should be automatically installed with the installation of Docker:

which docker

which docker-compose

which git

docker info

1. Clone the Aerie repository:

git clone git@github.jpl.nasa.gov:MPS/aerie.git

cd aerie

1. Log into the Artifactory Docker repository:

docker login cae-artifactory.jpl.nasa.gov:16001/gov/nasa/jpl/ammos/mpsa/aerie

1. Use Docker Compose to start the system:

docker-compose -f docker-compose.yml up --build

1. To stop and remove all the containers run:

docker-compose down

## Configuration

The docker-compose files are parameterized with the .env file in the root of the Aerie repository.

|  |  |
| --- | --- |
| Environment Variable | Description |
| DOCKER\_TAG | A Docker Tag of the Aerie version you are deploying. It has the form: [BRANCH\_NAME]+b[BUILD\_NUMBER].r[SHORT\_GIT\_COMMIT\_HASH].[yyyyMMdd]. For example this is a real tag: develop-b122.rcb8493e.20190529. For a list of Docker image tags, first log into Artifactory. Note each service has a tag of a single version. For example the nest, adaptation, plan, and sequencing services all have a tag of develop-b122.rcb8493e.20190529. |
| DOCKER\_URL | The URL of a Docker repository. Defaults to Artifactories docker-develop-local repository. |

# Troubleshooting & Defect Reporting

Please contact Chris Camargo ([ccamargo@jpl.nasa.gov](mailto:ccamargo@jpl.nasa.gov)) for any issues you have when deploying Aerie.