

# UPLAT

## System Overview and Used Technologies



Revizyon : 1.00  
Tarih : 02.03.2020



Supported by ROSIN - ROS-Industrial Quality-Assured Robot Software Components.  
More information: [rosin-project.eu](http://rosin-project.eu)



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732287.



Virtual Robotic Laboratory and Learning Materials for ROSin by Inovasyon Muhendislik Ltd. Sti. is licensed under CC BY-NC-ND 4.0. To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc-nd/4.0>

## DOKÜMAN REVİZYON SAYFASI

REV. NO	TARİH	SAYFA NO.	AÇIKLAMA
1.00	02/03/2020	Tümü	V1

### CONTENTS

1. Theia IDE	5
1.1. Installation	5
Requirements	5
1.2. Setup	5
1.3. Building	6
1.4. Running	6
2. Docker Containers	6
2.1. Comparing Containers and Virtual Machines	7
2.2. Installation on Ubuntu	7
2.3. Uninstall	7
2.4. Docker Compose	8
2.4.1. Installation	8
2.5. Docker Hub	8
2.6. Docker CLI Commands	8
2.6.1. <i>docker</i> (Base Command)	8
2.6.2. <i>docker run</i>	9
2.6.3. <i>docker container</i>	10
3. NGINX	10
3.1. Installation	12
4. Git	12
4.1. Installation	12
4.2. Some Basic Commands	12
5. ROS	12
6. Gazebo Web	13
6.1. Architecture of Gazebo	13
7. Jupyter Notebook	13
7.1. Installing Jupyter	14
7.2. Starting the notebook server	14
7.3. Creating a new notebook document	14
8. Stack	15
8.1. MERN Stack	16
9. MongoDB	17
9.1. Installing MongoDB	17
9.2. NoSQL	17
9.3. MongoDB Compass	17
9.3.1. Install MongoDB Compass	19
10. Node.js	19

10.1.	Features of Node.js	20
10.2.	Install Node.js	20
11.	Express.js	21
11.1.	Installing Express	21
12.	React	22
12.1.	Virtual DOM	22
12.2.	Component	22
12.3.	Props	22
12.4.	JSX	22
12.5.	Hooks	22
12.6.	Install React	22
12.7.	Why use React?	23
13.	Fuse React Template	23
13.1.	Installing Fuse React	23
14.	Amazon Web Service (AWS)	24
14.1.	AWS Identity and Access Management (IAM)	24
14.1.1.	IAM Features	25
14.2.	AWS EC2	25
14.2.1.	AWS EC2 Bulut Sunucusu Türleri	26
	General Purpose	26
	Compute Optimized	26
	Memory Optimized	26
	Accelerated Computing	26
	Storage Optimized	26
14.2.2.	Launch an Instance	27
14.3.	Systems Manager (SSM)	34
14.4.	AWS Elastic Load Balancing (ELB)	35
14.5.	AWS Certificate Manager (ACM)	36
14.6.	Route53	36
14.7.	Simple Email Service (SES)	37
14.8.	Amazon Simple Storage Service (S3 Buckets)	37
14.9.	CloudFront	38
14.10.	AWS SDK for JavaScript	38
6.10.1	Getting Started in Node.js	38
14.11.	AWS Command Line Interface (AWS CLI)	40
6.11.1	Installing	40
6.11.2	Usage	40

## 1. Theia IDE

Theia is an IDE that supports many programming languages that can work on the cloud and desktop, developed based on various web technologies and VS Code. Theia is developed by various communities under the leadership of the Eclipse Foundation, a not-for-profit organization. Thanks to its modular structure, extensions can be easily added by developers. Theia was developed based on VS Code and supports VS Code extensions directly. Unlike VS Code, Theia can work in the cloud environment. Because the Language Server Protocol is used in this project, it can offer intelligent editing support for more than 60 languages. Also, thanks to the Linux terminal integrated in theia, desired linux commands can be executed. Theia user interface consists of flexible draggable layouts thanks to PhosphorJS.

On Ubuntu machines that will be given to the user in the developed ROS Education Platform Uplat, Theia IDE will be presented as installed. Theia IDE will run on the 3000 port of the machine on which it is installed and will be presented to the user through an iframe in the user interface of the platform. The user will be able to write code for ROS through this IDE and run the command he wants, such as running the written code or running the ROS commands, thanks to the integrated terminal.

### 1.1. Installation

Requirements

You'll need node in version 10:

```
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.5/install.sh | bash  
  
nvm install 10(node version)
```

and yarn

```
npm install -g yarn
```

Also make sure your python --version points to a Python 2.x installation.

### 1.2. Setup

Start with creating a new empty directory and moving into it:

```
mkdir my-app  
cd my-app
```

Create package.json in this directory

```
{  
  "private": true,  
  "dependencies": {  
    "typescript": "latest",  
    "@theia/typescript": "next",  
    "@theia/navigator": "next",  
    "@theia/terminal": "next",  
    "@theia/outline-view": "next",  
    "@theia/preferences": "next",  
    "@theia/messages": "next",  
    "@theia/git": "next",  
    "@theia/file-search": "next",  
    "@theia/markers": "next",
```

```
"@theia/preview": "next",
"@theia/callhierarchy": "next",
"@theia/merge-conflicts": "next",
"@theia/search-in-workspace": "next",
"@theia/json": "next",
"@theia/textmate-grammars": "next",
"@theia/mini-browser": "next"
},
"devDependencies": {
"@theia/cli": "next"
}
}
```

### 1.3. Building

First, install all dependencies.

```
yarn
```

Second, use Theia CLI to build the application.

```
yarn theia build
```

*yarn* looks up *theia* executable provided by *@theia/cli* in the context of our application and then executes the build command with *theia*. This can take a while since the application is built in production mode by default, i.e. obfuscated and minified.

### 1.4. Running

After the build is finished, we can start the application:

```
yarn theia start
```

You can provide a workspace path to open as a first argument and *--hostname*, *--port* options to deploy the application on specific network interfaces and ports, e.g. to open /workspace on all interfaces and port 8080:

```
yarn theia start /my-workspace --hostname 0.0.0.0 --port 8080
```

Open the application by entering the printed address in a new browser page.

## 2. Docker Containers

Docker is software that provides virtualization at the operating system level. It is used to run software packages called Docker containers. It is a standard software unit that packs containers and all of its dependencies. Because the Docker Container packs its code and dependencies, moving the software to other environments can be done quickly and safely. Docker containers are created from docker images. Docker Container Images are a system image that contains everything that an application needs, such as code, runtime, system tools, system libraries and necessary settings.

Docker containers are created from Docker images run by the Docker Engine. Containers created in this way always work the same regardless of the infrastructure it works on. In addition, these containers are completely isolated from the infrastructure on which they are located. Therefore, they can neither communicate with the infrastructure nor other containers unless requested.

In the ROS Training Platform, we developed, either containers or virtual machines would be used to present the ROS environment to the user. It was decided to use a Docker container as the start-up time and system resource consumption were less than virtual machines. To

realize this, the necessary software for the platform such as ROS, Gazebo, Theia IDE and Jupyter Notebook was installed in the Ubuntu Xenial image. Later, the image of this container was taken and thus the image required for the platform was created. If a user starts a training, we need to create a container from related image and present it to the user interface. Also, ROS, Gazebo, related launch file, Jupyter Notebook and Theia IDE must be run after these containers are started. For this, the necessary commands were entered in a script and the container was run with this script. Another task that needs to be done when operating containers is the opening of ports 8080 for Gazebo, 3000 for Theia IDE and 8888 for Jupyter Notebook.

## 2.1. Comparing Containers and Virtual Machines

Containers and virtual machines isolate and share similar resources, but containers virtualize the operating system while virtual machines virtualize the hardware. As containers perform an abstraction that packs code and dependencies together, virtual machines abstract physical hardware and turn one machine into multiple machines. Virtual machines use more system resources than containers because they get the exact copy of the operating system. As seen in Figure 1, Docker virtualizes processes only, while virtual machines virtualize the operating system. For this reason, Docker Containers consume less system resources.

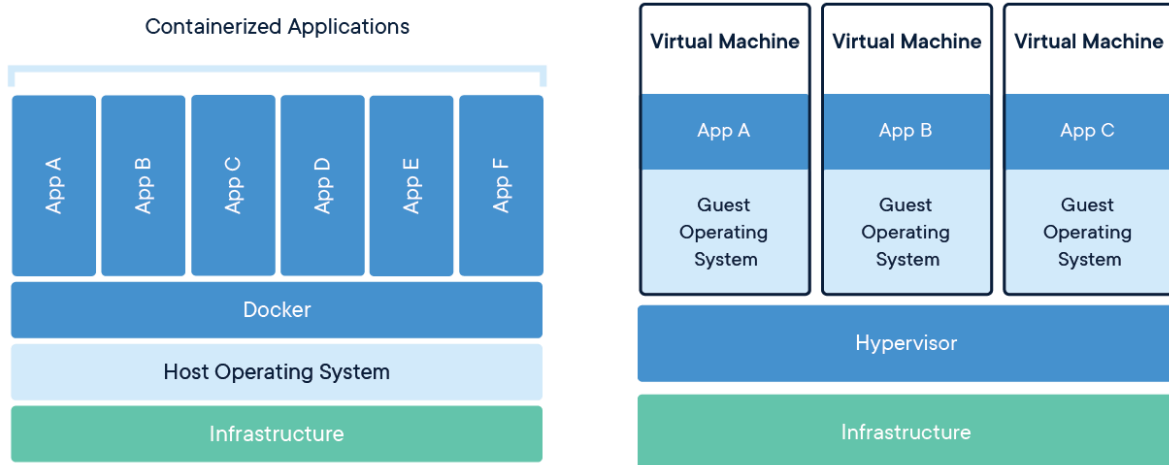


Figure 1: Docker vs Virtual Machine

## 2.2. Installation on Ubuntu

Update the *apt* package index.

```
sudo apt-get update
```

Install the latest version of Docker Engine- Community and containerd, or go to the next step to install a specific version:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Verify that Docker Engine- Community is installed correctly by running the hello-world image.

```
sudo docker run hello-world
```

If you would like to use Docker as a non-root user, you should now consider adding your user to the "docker" group with something like:

```
sudo usermod -aG docker your-user
```

## 2.3. Uninstall

Uninstall the Docker Engine- Community package:

```
sudo apt-get purge docker-ce
```

Images, containers, volumes, or customized configuration files on your host are not automatically removed. To delete all images, containers, and volumes:

```
sudo rm -rf /var/lib/docker
```

### 2.4. Docker Compose

It is used to start many containers with a single command. Configuration settings for these containers are made using the YAML file. Thanks to Docker Compose, all you need to do is to create a `docker-compose.yml` file containing the necessary settings and then run it with the `docker-compose up` command.

A sample `docker-compose.yml` file:

```
version: '3'
services: web:
  build: .
  ports: - "5000:5000"
  volumes:
    - ./code
    - logvolume01:/var/log
  links: - redis
  redis
  image: redis
  volumes: logvolume01: {}
```

#### 2.4.1. Installation

Run this command to download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.4/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Test the installation.

```
docker-compose --version
```

### 2.5. Docker Hub

Docker Hub is Docker's repository service. It is a service where developers can upload the images they create and access these images anywhere. Images can be uploaded publicly and privately. On Ubuntu Linux, docker images are uploaded to the Docker Hub with the `docker push` command. When needed, these images can be downloaded with the `docker pull` command. To log into Docker Hub, `docker login` command is applied and then login with username and password information. Docker `logout` command is used to log out.

Full use of the `docker push` command:

```
docker push <hub-user>/<repo-name>:<tag>
```

Full use of the `docker pull` command:

```
docker pull <hub-user>/<repo-name>:<tag>
```

### 2.6. Docker CLI Commands

#### 2.6.1. `docker` (Base Command)

**`docker build`:** Build an image from a Dockerfile.



**docker commit:** Create a new image from a container's changes.

**docker container:** Manage containers.

**docker create:** Create a new container.

```
docker create -v /home/docker:/docker --name docker ubuntu
```

**docker exec:** Run a command in a running container.

```
docker exec -it --env DISPLAY=:1.0 -w /root ubuntu_bash bash
```

**docker image:** Manage images.

**docker images:** List images.

**docker inspect:** Return low-level information on Docker objects.

**docker kill:** Kill one or more running containers.

**docker login:** Log in to a Docker registry.

**docker logout:** Log out from a Docker registry.

**docker network:** Manage networks.

**docker ps:** List containers.

**docker pull:** Pull an image or a repository from a registry.

**docker push:** Push an image or a repository to a registry.

**docker rm:** Remove one or more containers.

**docker run:** Run a command in a new container.

**docker save:** Save one or more images to a tar archive (streamed to STDOUT by default).

**docker start:** Start one or more stopped containers.

**docker stats:** Display a live stream of container(s) resource usage statistics.

**docker stop:** Stop one or more running containers.

**docker tag:** Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE.

**docker volume:** Manage volumes.

For more detailed information, visit <https://docs.docker.com>.

### 2.6.2. docker run

Docker starts isolated processes from the operating system with the docker run command from an image. These processes have their own file system, their own network and their own process tree.

The basic docker run command takes this form:

```
docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]
```

**--detached (-d) mode:** The container runs in the background.

**--tty(-t):** Allocate a pseudo-tty

**--interactive(-i):** Keep STDIN open even if not attached. Runs fully interactive using --interactive and --tty mode together (-it).

**--name:** Sets name the container

```
docker run --name my-redis -d redis
```

**--dns[]:** Set custom DNS servers

**--restart no:** Do not automatically restart the container when it exits. This is the default.

**--restart on-failure** [:max-retries]: Restart only if the container exits with a non-zero exit status. Optionally, limit the number of restarts retries the Docker daemon attempts.

```
docker run --restart=on-failure:10 redis
```

When the restart policy is selected as above, it restarts up to 10 times.

**--restart always:** Always restart the container regardless of the exit status. When you specify always, the Docker daemon will try to restart the container indefinitely. The container will also always start on daemon startup, regardless of the current state of the container.

**--restart unless-stopped:** Always restart the container regardless of the exit status, including on daemon startup, except if the container was put into a stopped state before the Docker daemon was stopped.

**--rm:** By default, the file system is not deleted even when exiting a container. Container status is set exited. Instead, you can add the --rm option if you want it to be deleted at the same time when exiting the container.

**CMD [ARGS]:** It is a command added to the end of the docker run command. This command specifies which process to run when the container is started. This is optional because a CMD command may have been entered while creating the Dockerfile.

**--entrypoint="":** Overwrite the default entrypoint set by the image

**--env:** Environment value can be assigned to the container created with this option.

**--volume(-v) "path":** With this option, a file sharing area is created between the host and the container. In this way, data can be shared between the container and the host, or between the container and the container.

**--user:** The default user is root when a container is launched. However, using the --user="username" command, the container can be launched with a specific user.

**--workdir(-w):** The default working environment is root (/). This location can be changed using --workdir.

**--publish(-p):** Publish a container's port(s) to the host.

```
docker run -t -i --rm -w /path/to/dir/ -v /doesnt/exist:/foo -p 127.0.0.1:80:8080/tcp --env MYVAR2=foo ubuntu bash
```

### 2.6.3. docker container

Manage containers.

**docker container ls:** List containers.

**docker container prune:** Remove all stopped containers.

**docker container inspect:** Display detailed information on one or more containers.

For more detailed information, visit <https://docs.docker.com> .

## 3. NGINX

nginx is an HTTP web server. In addition, there are mail proxy server, reverse proxy and load balancer tasks. The biggest plus in other web servers is consuming very low resources. However, this causes nginx to be less flexible than its competitors. nginx is flexible enough for us, as we use it only to forward HTTP requests in the Uplat project. Also, nginx can be configured for SSL certificates.

nginx has one master process and several worker processes. Master process manages worker processes according to the configuration file. Worker processes manage incoming http requests.

Nginx systemd commands;

Start nginx:

```
sudo systemctl start nginx
```

Stop nginx

```
sudo systemctl stop nginx
```

Restart nginx:

```
sudo systemctl restart nginx
```

Reload nginx:

```
sudo systemctl reload nginx
```

After any changes made in the configuration file, nginx should be restarted. For this, you can use restart, reload commands or start commands after stop. The difference between the restart and reload commands; When using the restart command, all processes are stopped and restarted completely, while using the reload command, the old worker processes are closed after the new worker processes are started.

nginx forwards the HTTP request to the location specified in the configuration file. Therefore, nginx listens on the specified port and sends HTTP requests from that port to the specified address.

Below is a simple example of this.

```
server {
    listen 80;
    root /data/up1;
    location /images/ {
        proxy_pass http://localhost:8080;
    }
}
```

You can see the server block added to the configuration file above. Accordingly, nginx listens on port 80. Forwards requests starting with /images/ to port 80 to the address "http://localhost: 8080". If the incoming request's URI information is in the form that matches the location block, the proxy\_pass directive changes the incoming URI as specified. Thus, the request goes to a different address.

A simple HTTP/HTTPS server example:

```
server {
    listen      80;
    listen      443 ssl;
    server_name www.example.com;
    ssl_certificate www.example.com.crt;
    ssl_certificate_key www.example.com.key;
    ...
}
```

For detailed information, you can visit <https://docs.nginx.com/nginx/> and <https://nginx.org/en/docs>.

Because of the Uplat project's frontend software runs on the 3000 port, requests to 80 port must be forwarded to 3000 port. We realized this process thanks to a server configuration file prepared for nginx.

Server configuration file is as follows.

```
server {
    listen 80;
    server_name theuplat.com www.theuplat.com;
    location / {
        proxy_pass http://127.0.0.1:3000;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_redirect off;
    }
}
```

```
proxy_set_header X-Forwarded-For $remote_addr;  
proxy_set_header X-Real-IP $remote_addr;  
proxy_set_header X-NginX-Proxy true;  
}  
}
```

### 3.1. Installation

```
sudo apt update  
sudo apt install nginx
```

We can check with the systemd init system to make sure the service is running by typing:

```
systemctl status nginx
```

## 4. Git

Git version control and source code management system. Thanks to this system, the software development process has become easier as a team. This system was written to be used in the development of the Linux kernel. With this system, all versions during the development of a software are saved. In this way, any version can be returned at any time. GitHub service was used while the Uplat project was being developed. The point to be considered here is a git version control tool, while GitHub is a service that uses git.

### 4.1. Installation

First, use the apt package management tools to update your local package index. With the update complete, you can download and install Git:

```
sudo apt update  
sudo apt install git
```

You can confirm that you have installed Git correctly by running the following command:

```
git --version
```

### 4.2. Some Basic Commands

Create new repository:

```
git init --bare /path/to/repository
```

Clone a repository:

```
git clone username@hostname:/path/to/repository
```

Commit and share:

```
cd /path/to/repository  
#(edit some files)  
git commit -a # Commit all changes to the local version of the repository  
git push origin master # Push changes to the server's version of the repository
```

## 5. ROS

The Robot Operating System (ROS) is not an exact operating system it is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms.

Ros is one of important main part in Uplat. Ros is running in docker container predefined hosts. In the first stage, ros kinetic was used in the system. In the future, it is planned to add other ros versions.

## 6. Gazebo Web

Gazebo is an open source multi-robot simulator developed for both indoor and outdoor spaces. It has the ability to simulate multiple robots, sensors and objects in three-dimensional environment. Gzweb is a WebGL client for Gazebo. Like gzclient, it's a front-end graphical interface to gzserver and provides visualization of the simulation. However, Gzweb is a thin client in comparison, and lets you interact with the simulation from the comfort of a web browser. This means cross-platform support, minimal client-side installation, and support for mobile devices.

### 6.1. Architecture of Gazebo

Gazebo is made up of several libraries.

- Physics: Updates the physical state of the simulation
- Staging: Visualizes the simulation state.
- Sensor: Creates sensor data.
- Migration: Manages communication between transactions.
- Graphical User Interface (GUI): Provides visualization and manipulation of the simulation.

These libraries are used by two main processes.

- Server (gzserver): Runs physics cycles and generates sensor data.
- Client (gzclient): Enables visualization of user interaction and simulation.

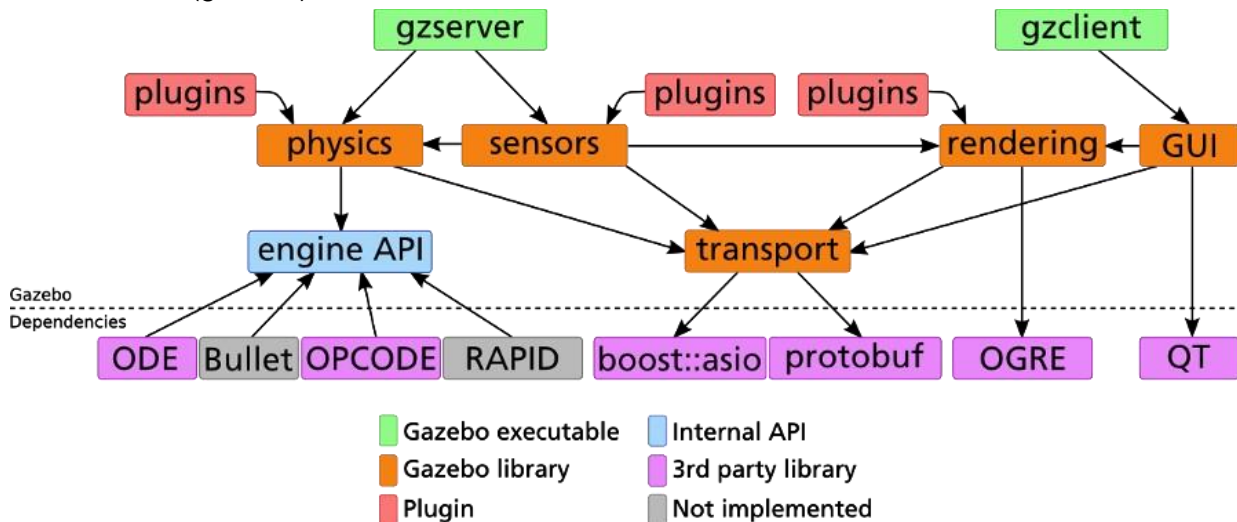


Figure 2: Gazebo's Architecture

Gazebo web is one of important main part for Uplat. Gazebo web is running in docker container predefined port. In our platform gazebo web is running on container's <http://localhost:8080/>.

## 7. Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include; data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

## 7.1. Installing Jupyter

**Conda:** If you use conda, you can install it with:

```
conda install -c conda-forge jupyterlab
```

**Pip:** If you use pip, you can install it with:

```
pip install jupyterlab
```

## 7.2. Starting the notebook server

You can start running a notebook server from the command line using the following command:

```
jupyter notebook
```

This will print some information about the notebook server in your console, and open a web browser to the URL of the web application (by default, <http://127.0.0.1:8888>).

You can create new notebooks from the dashboard with the New Notebook button, or open existing ones by clicking on their name. You can also drag and drop .ipynb notebooks and standard .py Python source code files into the notebook list area.

## 7.3. Creating a new notebook document

A new notebook may be created at any time, either from the dashboard, or using the File ► New menu option from within an active notebook. The new notebook is created within the same directory and will open in a new browser tab. It will also be reflected as a new entry in the notebook list on the dashboard. Figure 3 shows New menu option. In this option users can choose which python version want to use.



Figure 3: Creating a New Notebook

Blank jupyter notebook interface shown on figure 3. When you create a new notebook document, you will be presented with the notebook name, a menu bar, a toolbar and an empty code cell.

**Notebook name:** The name displayed at the top of the page, next to the Jupyter logo, reflects the name of the .ipynb file.

**Menu bar:** The menu bar presents different options that may be used to manipulate the way the notebook functions.

- **File:** actions related to files and directories
- **Edit:** actions related to editing documents and other activities
- **View:** actions that alter the appearance of JupyterLab



- **Run:** actions for running code in different activities such as notebooks and code consoles
- **Kernel:** actions for managing kernels, which are separate processes for running code
- **Tabs:** a list of the open documents and activities in the dock panel
- **Settings:** common settings and an advanced settings editor
- **Help:** a list of JupyterLab and kernel help links

**Toolbar:** The tool bar gives a quick way of performing the most-used operations within the notebook, by clicking on an icon.

**Code cell:** The default type of cell.

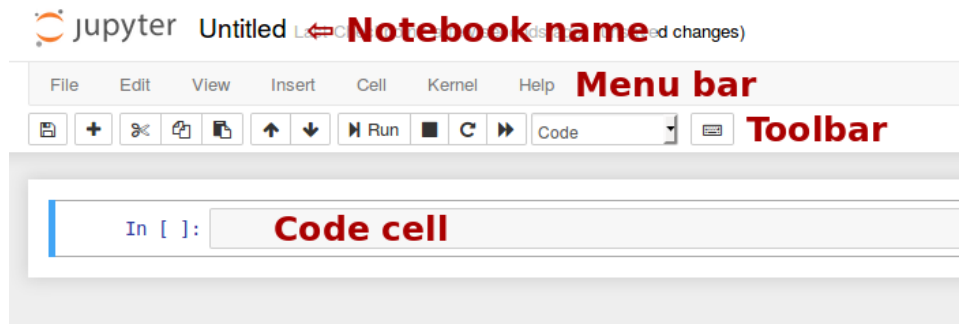


Figure 4: Rename notebook

Users can change already created file name. Rename notebook process shown on figure 5.

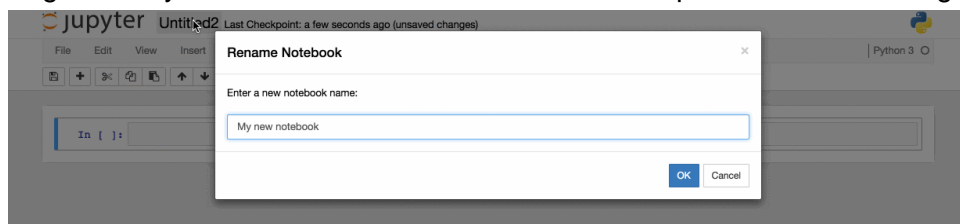


Figure 5: Rename notebook

Created jupyter file shown on Figure 4

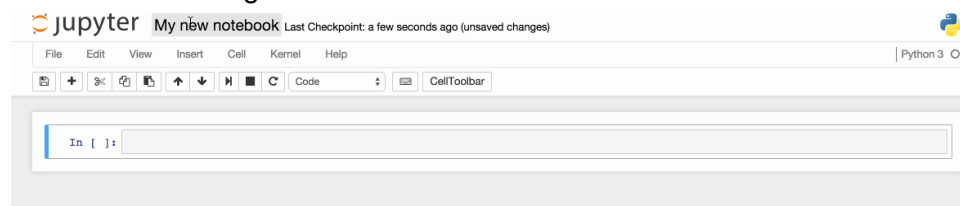


Figure 6: Created Jupyter Notebook

In uplat Jupyter Notebook is used to provide training document to the user. In our platform Jupyter Notebook is running on container's <http://localhost:8888/ipynb> .

## 8. Stack

Software Stack or solution stack is a whole set of software required to create the development platform that software developers need to be able to develop. A “stack” refers to any combination of programming languages and technologies or a combination of software products. Technically there are two types of development stacks: Firstly, a technology stack & Secondary, an application stack. In simple terms, the technology stack is a more cross-

disciplinary term in any software development process. There are too many solutions stacks in the development period. Some most known solution stacks in below;

- LAMP ->Linux, Apache, PostgreSQL, PHP
- MEAN ->MongoDB, Express.js, Ember.js and Node.js.
- MERN ->MongoDB, Express.js, React.js, Node.Js
- XAMPP -> Linux, Mac OS X, Windows/Apache/MySQL/PHP, Perl)
- LNMP / LEMP -> Linux, Nginx, MySQL or MariaDB, Perl, Php or Python

In the UPLAT development stage, the MERN stack is using.

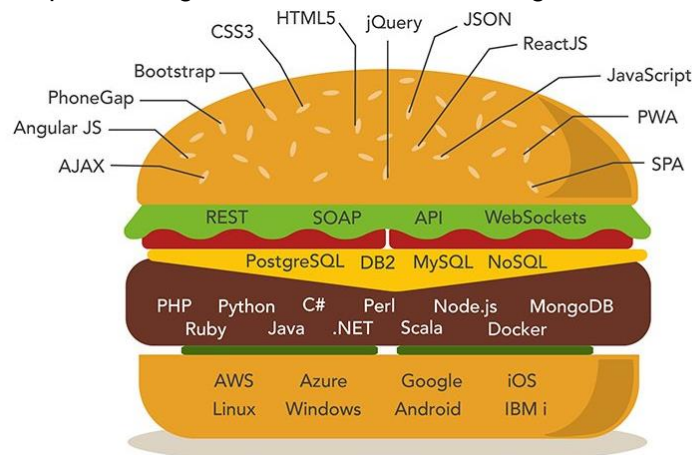


Figure 7: Simple Solution Stack Structure

## 8.1. MERN Stack

MERN Stack is a Javascript Stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises of 4 open-source technologies namely: MongoDB, Express, React and Node.js. It is designed to make the development process smoother and easier. Among these technologies Mongo DB is a database system, Node JS is a back-end runtime environment, Express JS is a back-end web framework and React is a front-end framework [1].

Each of these 4 powerful technologies provides an end-to-end framework for the developers to work in and each of these technologies play a big part in the development of web applications. The main advantage for developers using the MERN stack is that every line of code is written in JavaScript. This is a programming language that's used everywhere, both for client-side code and server-side code. With one language across tiers, there's no need for context switching.

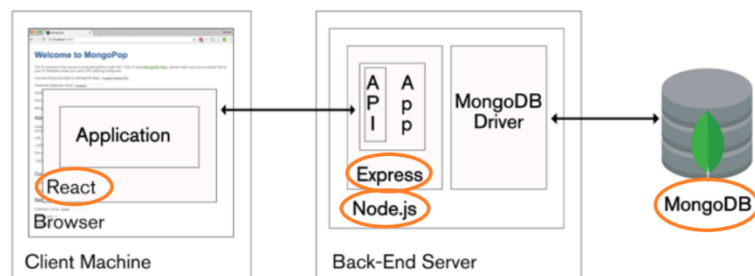


Figure 8: MERN Stack



## 9. MongoDB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with schema. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL). It is a document-based NoSQL database. It keeps the data structurally in Json format and in documents. Queries can be written in any field or by range [2].

MongoDB make concessions some ACID (Atomicity, Consistency, Isolation, Durability) features of relational databases to provide a more flexible data model that can be scaled horizontally. This makes MongoDB an excellent choice for high performance, low latency usage instances that require horizontal scaling beyond the reach of a single instance. But RDBMS's makes no concessions of ACID features because RDBMS's first aim is not fast inquiry. Other main differences shown on table 1. In addition, a lot of information about MongoDB can be found in the user manual in the [link](#).

Table 1: MongoDb and RDBMS Main Differences

RDBMS	MongoDB
database	database
Table	collections
row	documents
column	field
index	index
table joins	\$lookup, embedded documents
Primary keys	Primary key (_id)
aggregation	aggregation pipeline
transactions	transactions
Nested table or object	Embedded document
SQL	NoSQL

### 9.1. Installing MongoDB

Step 1: Import the public key used by the package management system

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
```

Step 2: Create a list file for MongoDB according to your version of Ubuntu.

```
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu
ubuntu_version/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-
4.2.list
```

Step 3: Install the MongoDB packages. For more information please visit <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/>

```
sudo apt-get install -y mongodb-org
```

### 9.2. NoSQL

NoSQL does not mean that there is no SQL above all. It is an abbreviation of "Not Only SQL", which means NoSQL includes SQL and other parts (documents, key-value, models). It is specially designed for specific data models and has flexible schemes for creating modern applications. NoSQL databases have been widely accepted with ease of development, functionality and performance at the appropriate scale. It uses a variety of data models, including document, chart, key-value, in-memory, and search.

### 9.3. MongoDB Compass

MongoDB Compass is the GUI for MongoDB. Compass allows you to analyse and understand the contents of your data without formal knowledge of MongoDB query syntax. In

In addition to exploring your data in a visual environment, you can also use Compass to optimize query performance, manage indexes, and implement document validation. MongoDB compass connect panel shown on Figure 9.

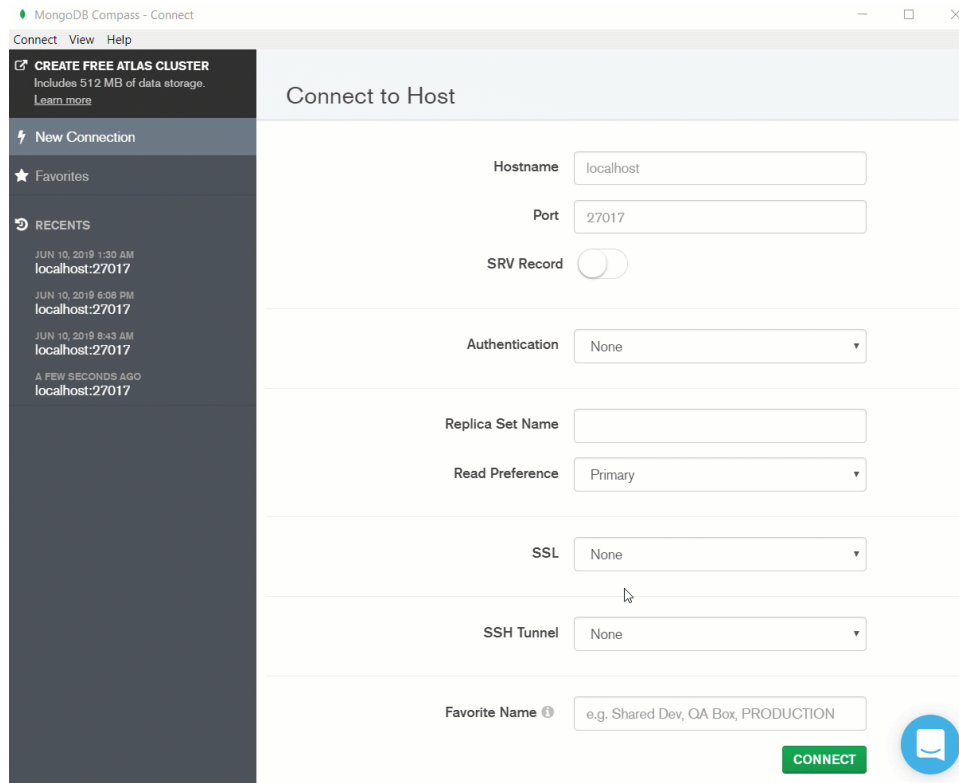


Figure 9: Connect Panel

MongoDB compass simplifies the management of data and provides to create new database. In addition to these compass gives some information about database storage size, collection numbers and etc. Figure 10 show us database information panel.

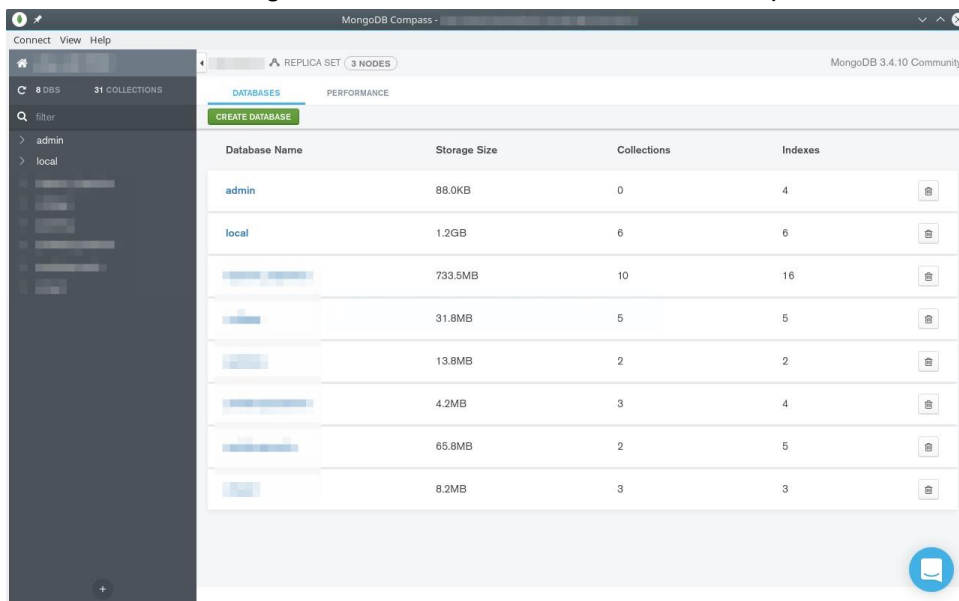


Figure 10: Database Information Panel

MongoDB compass has a query bar. This bar makes possible writing manual query. Query bar shown on figure 11.



Figure 11: Query Bar

Documents panel visualize the documents. In here MongoDB Compass user can change documents, documents types and CRUD process. Document panel shown on Figure 12.

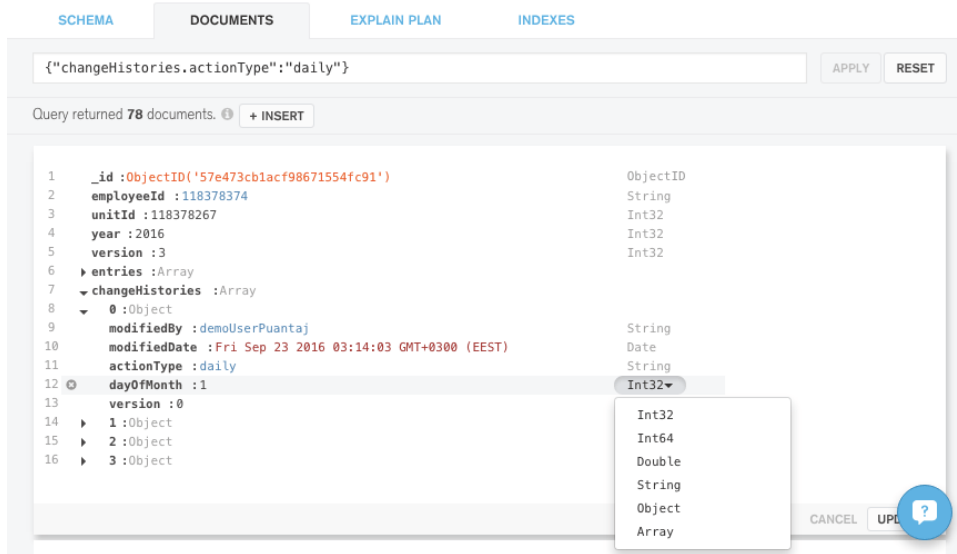


Figure 12: Documents Panel

## 9.3.1. Install MongoDB Compass

Step 1: Download MongoDB compass

```
wget https://downloads.mongodb.com/compass/mongodb-compass_1.15.1_amd64.deb
```

Step 2: install the compass

```
sudo dpkg -i mongodb-compass_1.15.1_amd64.deb
```

Also MongoDB Compass can download this [link](#). In this link users can choose suitable version and platform for user's system. Download panel shown on Figure 13.

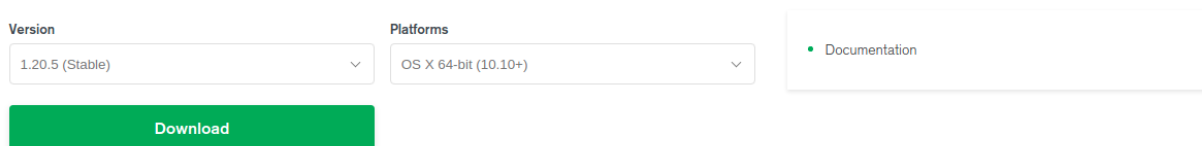


Figure 13: Download Panel

In Uplat, MongoDB used to storing and managing users datas. It provides us flexibility and high performance.

## 10. Node.js

Node.js is a server-side framework structure developed free and open source. Node.js, which stands out with its cross-platform support; It can be used on Windows, Linux, Unix, macOS and many operating system architectures without any problem. In this way, desired internet projects can be developed and web software that offers solutions for real-time needs can be produced.

NodeJS is an environment that enables us to develop server-side and networked applications using the "Google Chrome's v8 JavaScript Engine" with JavaScript language. NodeJS uses a scalable, event-driven asynchronous event driven JavaScript runtime, a non-

blocking I/O working model. Event-driven means that the server only reacts when an event occurs. This allow us to create high performance, highly scalable, “real-time” applications. In addition, web servers can be created easily instead of web server installations such as ISP and Apache. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

### 10.1. Features of Node.js

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven:** All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- **Very Fast:** Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable:** Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering:** Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License:** Node.js is released under the MIT license.

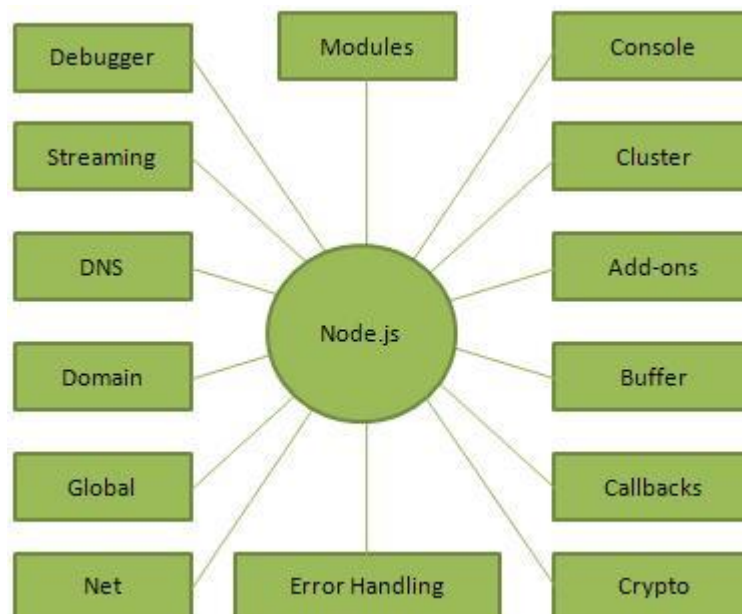


Figure 14: Concept of Node.js

### 10.2. Install Node.js

Step 1: Add Node.js PPA

Based on your OS architecture, download and extract the archive node-v6.3.1-osname.tar.gz into /tmp, and then finally move extracted files into /usr/local/nodejs directory. For example:

```
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
```

Step 2: Install Node.js on Ubuntu

This command also installs many other dependent packages on your system

```
sudo apt-get install nodejs
```

Step 3: Check Node.js and NPM Version

Verify and check the installed node.js version

```
node -v && npm -v
```

## 11. Express.js

Express.js, or simply Express, is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. some of the core features of Express framework

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

Express released as free and open-source software under the MIT License. It has been called the de facto standard server framework for Node.js.

### Advantages of Express.js Module

- It provides the opportunity to develop Web applications easily and quickly,
- Template engines such as Pug, EJS, Handlebars can be used in application development processes,
- Comprehensive applications can be developed with the middleware modules,
- Web requests can be managed with the Route method,
- Database applications such as MySQL, MongoDB, SQL SERVER can be used in applications,
- It is possible to create REST API for different applications, it is easy to manage static files,
- As a routing, Page.js with a simple usage developed based on Express.js can be considered as an option. [3]

Methods that support Express.js: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe , patch, search.

### 11.1. Installing Express

Install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

```
npm install express --save
```

## 12. React

React is a Javascript library developed by Facebook which allows you to build UI components. It facilitates the creation of interactive User Interfaces. It also makes the code easier to understand and launch. React Java Script framework uses server-side rendering to provide a flexible, performance-oriented solution. React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing. Redux and React Router are respective examples of such libraries.

### 12.1. Virtual DOM

Virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change.

### 12.2. Component

React code is made of entities called components. Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass in values that are known as "props".

### 12.3. Props

It is the constant data that enables data transfer between components. They are rendered only once.

### 12.4. JSX

JSX, or JavaScript XML , is an extension to the JavaScript language syntax. Similar in appearance to HTML, JSX provides a way to structure component rendering using syntax familiar to many developers. React components are typically written using JSX, although they do not have to be (components may also be written in pure JavaScript). JSX is similar to another extension syntax created by Facebook for PHP called XHP. An example of JSX code shown on Table 2:

Table 2: JSX example

```
class App extends React.Component {  
  render() {  
    return (  
      <div>  
        <p>Header</p>  
        <p>Content</p>  
        <p>Footer</p>  
      </div>  
    );  
  }  
}
```

### 12.5. Hooks

Hooks are functions that "connect" to React state and life cycle features using functional components. Hooks do not work in classes - they enable us to use React without classes.

### 12.6. Install React

Step 1: Install "create-react-app" using npm



```
npm install create-react-app --global
```

Step 2: Create a new react app by using

```
create-react-app app_name
```

### 12.7. Why use React?

- Features like Virtual DOM, JSX and Components makes it much faster than the rest of the frameworks out there.
- Stands for JavaScript XML. It is an HTML/XML JavaScript Extension which is used in React. Makes it easier and simpler to write React components.
- Components are the building blocks of UI wherein each component has a logic and contributes to the overall UI. These components also promote code reusability and make the overall web application easier to understand.

### 13. Fuse React Template

React is the core of Fuse react template. Fuse React is NOT a traditional admin template, it's an React app. Fuse React written with the React Hooks. Fuse React is a complete React admin template that follows Google's Material Design guidelines. Fuse React admin template uses Material UI as a primary UI library while using Redux for the state management. It has built-in page templates, routing and auth features. It also includes 5 example apps, 20+ pages, lots of reusable react components and more. Fuse React admin template is not only a great kick starter for your project but it also is an extremely good place to learn some of the advanced aspects of the React.

#### 13.1. Installing Fuse React

**Step 1.** Unzip the zip file that you have downloaded from Themeforest. Inside the zip file, you will find the Skeleton Project (Fuse-react-x.x.x-skeleton.zip) along with the Demo Project (Fuse-react-x.x.x-demo.zip), PSD designs and a readme file.

**Step 2.** Extract the contents of the zip file (Fuse-react-x.x.x-skeleton.zip) into a folder that you will work within. For this documentation, we will refer that as "your work folder".

**Step 3.** Open your favorite console application (Terminal, Command Prompt etc.), navigate into your work folder, run the following command and wait for it to finish:

```
yarn
```

This command will install all the required Node.js modules into the node\_modules directory inside your work folder.

**Step 4.** While still in your work folder, run the following command in the console application:

```
yarn start
```

You can check out your console application to get further information about the server. By default, it will run on `http://localhost:3000` but it might change depending on your setup.

Sample project structure of the Fuse React shown on Figure 15.

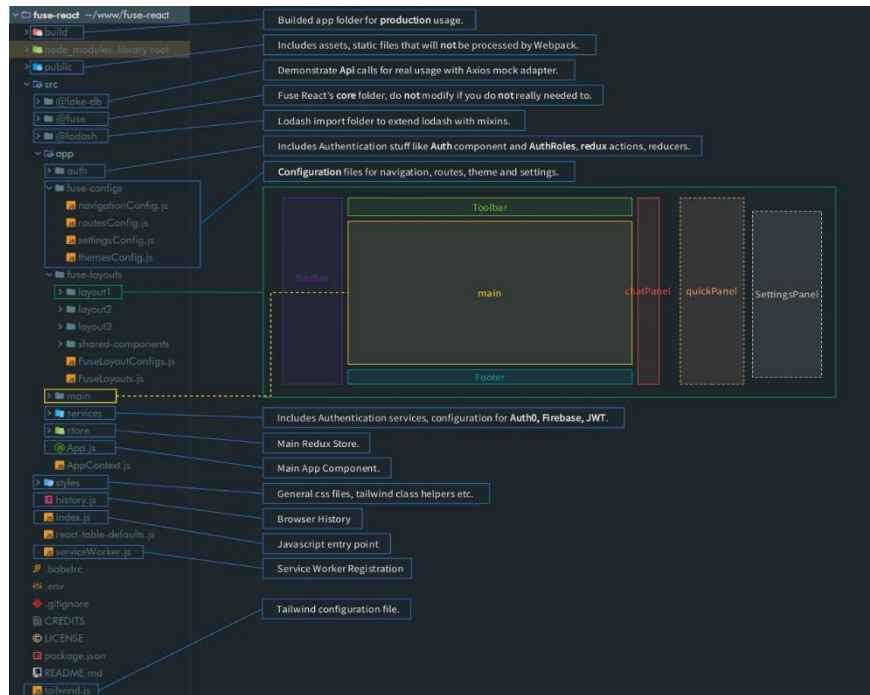


Figure 15: Project Structure of the Fuse React

## 14. Amazon Web Service (AWS)

Amazon Web Service is a cloud service that includes many services. These services briefly cover many technologies such as storage, database, hosting and machine learning, artificial intelligence. AWS has 69 availability areas in 22 geographic regions worldwide.

The Uplat project has been developed on the AWS infrastructure. AWS has been used both for hosting and for preparing the ROS environment that every user need. EC2, Certificate Manager, Route53, ELB and CloudFront services were used together to provide hosting. EC2 service was used to present the ROS environment to the user. AWS SES service was used for the mail service. AWS-SDK, SSM and IAM services were used for dynamic control of AWS services through Uplat's backend software. Route53 service will be used for domain management and Certificate Manager service will be used for SSL certificate.

### 14.1. AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. IAM divided to three part;

For more detailed information please visit [https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html).



**IAM Users:** An IAM user is an entity that you create in AWS. The IAM user represents the person or service who uses the IAM user to interact with AWS. A primary use for IAM users is to give people the ability to sign into the AWS Management Console for interactive tasks and to make programmatic requests to AWS services using the API or CLI.

**IAM Groups:** An IAM group is a collection of IAM users. You can use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users.

**IAM Roles:** An IAM role is very similar to a user, in that it is an identity with permission policies that determine what the identity can and cannot do in AWS. However, a role does not have any credentials (password or access keys) associated with it.

### 14.1.1. IAM Features

IAM gives us some specific features:

- **Shared access to your AWS account:** You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.
- **Granular permissions:** You can grant different permissions to different people for different resources.
- **Identity federation:** You can allow users who already have passwords elsewhere to get temporary access to your AWS account.
- **Secure access to AWS resources for applications that run on Amazon EC2:** You can use IAM features to securely provide credentials for applications that run on EC2 instances. These credentials provide permissions for your application to access other AWS resources.

### 14.2. AWS EC2

Amazon Elastic Compute Cloud (Amazon EC2) is a cloud service that offers machines with scalable processing capacity. With this service, when needed, processing capacity can be increased, security and network needs can be configured, and storage needs can be managed. AWS provides virtual machines called instances with this service. Amazon Machine Images (AMIs) is used to take the image of EC2 instances. New instances can be created quickly from these images. Thanks to the security group, you can determine which ports of instances can be accessed in which IP range. Thanks to elastic IP, static IP can be obtained and assigned to an instance. SSH can be used to connect to an instance on a Linux machine. For this, a key pair is created during the instance launch phase. During connection, it must be connected using the created key pair file.

Application traffic using Elastic Load Balancer can be distributed to many instances. In this way, all application traffic is not loaded in an instance. Amazon CloudWatch can be used to monitor Instance's status. Amazon CloudWatch can be used to view Instance's instant system resource usage. By setting an Auto Scaling group to an instance, you can run new instances automatically when the instance is under high load and then share this load to all instances. There are different pricing options for EC2. These;

- **On-Demand Instances:** You pay what you use without any commitment and down payment.
- **Savings Plans:** You can reduce costs by committing consistent use for 1 or 3 years.
- **Reserved Instances:** You can reduce the cost by making a 1 or 3 year commitment to a specific instance configuration, including the instance type and region.
- **Spot Instances:** It is instance that Amazon sells its idle servers at very low prices. In this way, very low prices can be used in EC2 service.

For more detailed information on payment options, visit <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-purchasing-options.html>.

### **14.2.1. AWS EC2 Instance Types**

#### **General Purpose**

Such servers are balanced in terms of processing capacity, memory and network resources. Recommended for applications that use resources equally, such as web servers and code stores. A1 instances are eligible servers for Arm-based workloads. T2, T3 and T3a instances are Burstable Performance Instances and are recommended for use in processes with sudden CPU increase. M5 instances are ideal for workloads that require a balance of compute, memory, and networking resources including web and application servers, small and mid-sized databases, cluster computing, gaming servers, and caching fleet. M5n and M5dn, instance variants are ideal for applications that can take advantage of improved network throughput and packet rate performance. M5, M5a, M6g and M4 servers are also a good choice for many applications, as they offer a balanced combination of process, memory and network resources.

#### **Compute Optimized**

Cloud servers in this family are optimized for workloads that require computing power. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modelling, dedicated gaming servers and ad server engines, machine learning inference and other compute intensive applications. C4 and C5 instances are low priced servers in this family. C5n instances offer more network bandwidth and more memory than C4 and C5 instances.

#### **Memory Optimized**

Optimized to provide more performance to workloads that use memory a lot. These instances are; R5, R5a, R5n, R4, X1e, X1 and Z1d.

#### **Accelerated Computing**

Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating-point number calculations, graphics processing, or data pattern matching, more efficiently than is possible in software running on CPUs. P3 and P2 instances are designed for general-purpose GPU computing applications. Inf1 instances are built from the ground up to support machine learning inference applications. G4 instances are designed to accelerate machine learning inference and intensive graphics workloads. G3 instances are optimized for graphics-intensive applications. F1 instances offer customizable hardware acceleration with field programmable port arrays (FPGAs).

#### **Storage Optimized**

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications. I3 and I3e instances provides dense Non-Volatile Memory Express (NVMe) SSD instance storage optimized for low latency, high random I/O performance, high sequential disk throughput, and offers the lowest price per GB of SSD instance storage on Amazon EC2. D2 instances feature up to 48 TB of HDD-based local storage, deliver high disk throughput, and offer the lowest price per disk throughput performance on Amazon EC2. H1 instances feature up to 16 TB of HDD-based local storage, deliver high disk throughput, and a balance of compute and memory. For detailed information about EC2 server families, visit <https://aws.amazon.com/tr/ec2/instance-types>.

## 14.2.2. Launch an Instance

To run an instance, you first need to register to AWS. Next, login to AWS Console (<https://eu-central-1.console.aws.amazon.com/console>). An instance can be started after all the information requested by AWS are provided.

1. Here all services of AWS can be viewed and started. To start the EC2 instance, click on the EC2 service shown in Figure 16.

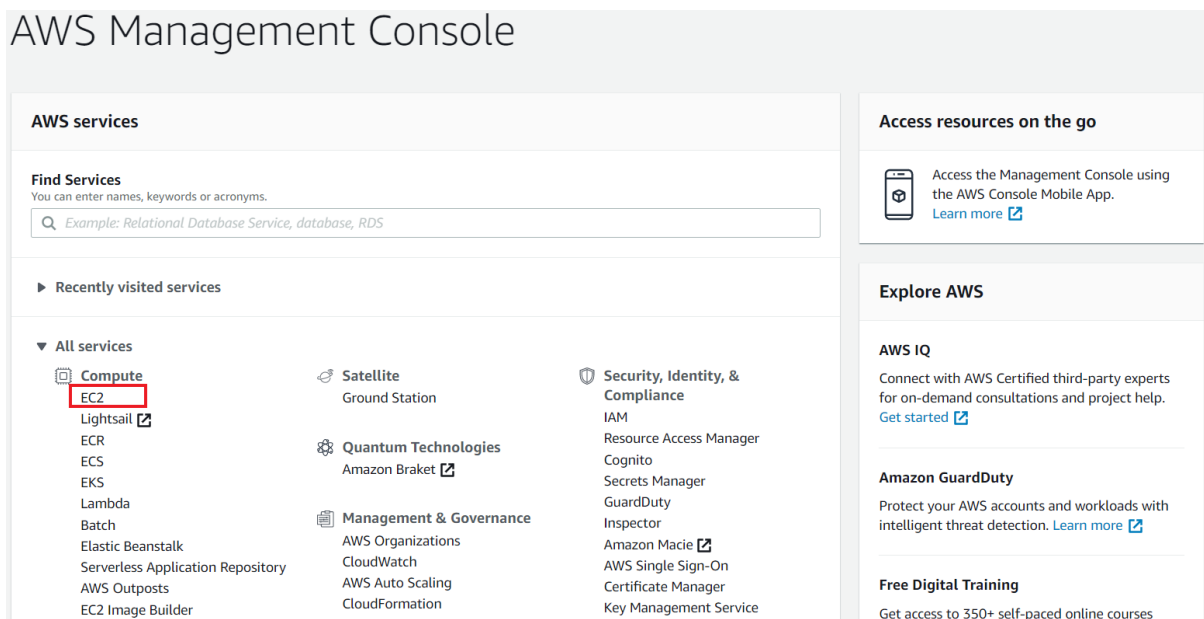


Figure 16: AWS Management Console Home

2. In the screen that opens, you can see information about your EC2 usage. Click the Launch instance button shown in Figure 17.

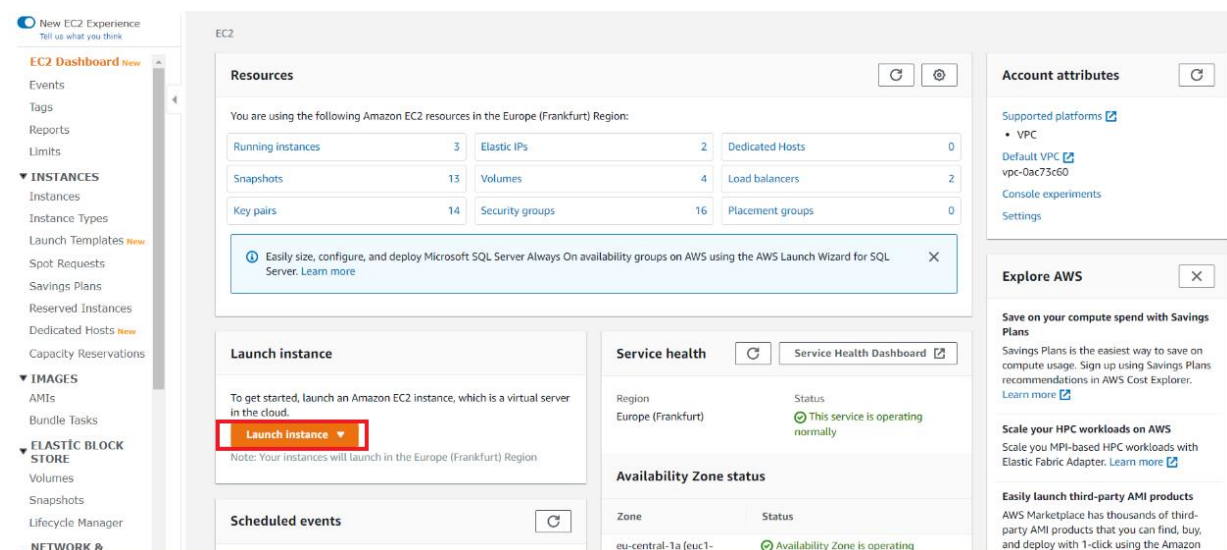


Figure 17: EC2 Dashboard

- In this step, Amazon Machine Image selection is made according to the instance to be created. You can choose from many images prepared by AWS. Or you can start an instance with an image you have prepared yourself. Or you can start an instance with one of your own images. You can access these images in the left panel shown in Figure 18.

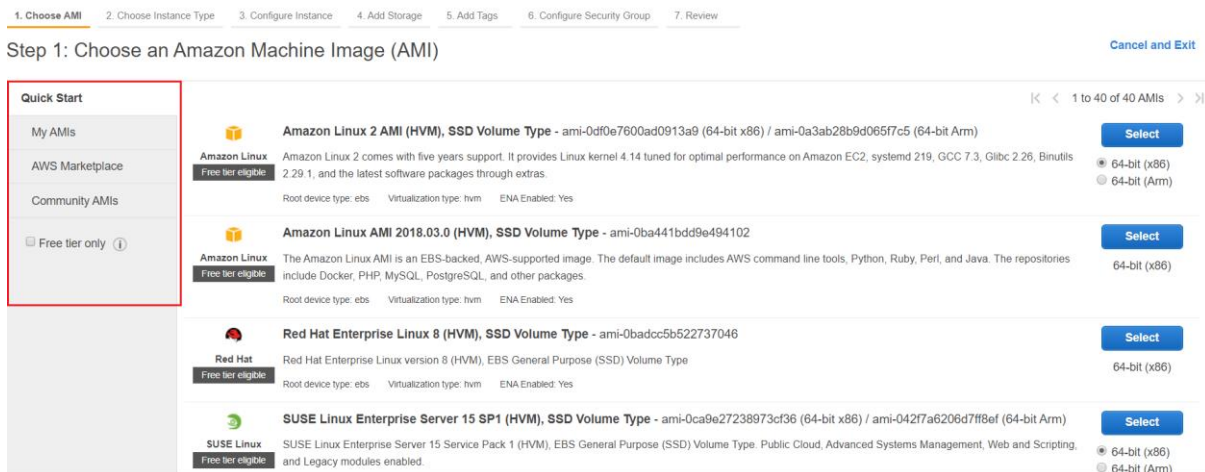


Figure 18: Image Panel

As we will create Ubuntu 18.04 LTS instance, select the image shown in figure 19.

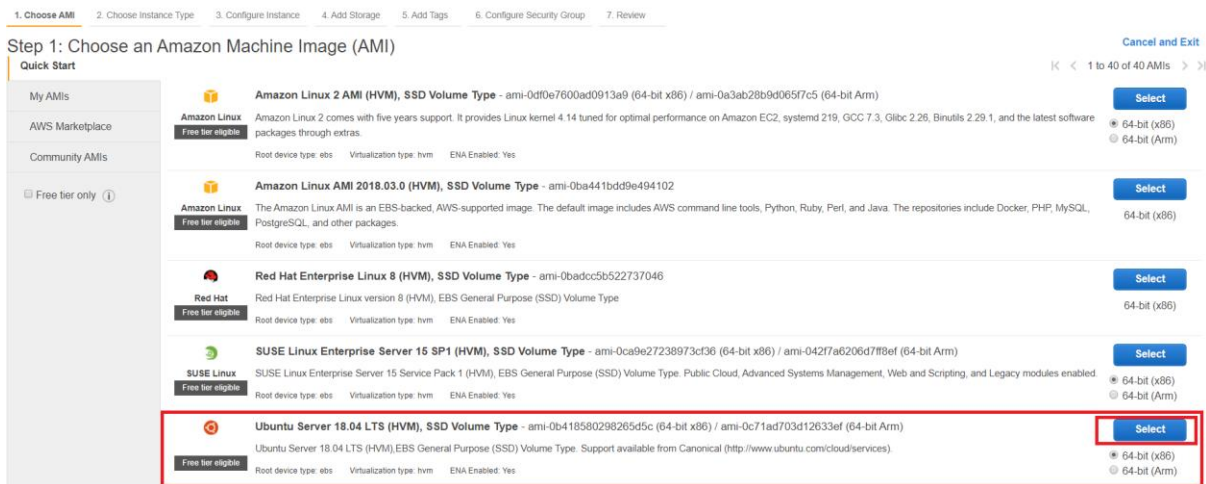


Figure 19: Ubuntu 18.04 Image

- You should decide which instance to choose according to the purpose of use and mark this instance. We will choose the free tier t2.micro instance. Then you can jump to the Review stage by clicking on the Review and Launch button shown in Figure 20, or you can continue the configuration by pressing the Next: Configure Instance Details button. We will click on Next: Configure Instance Details button.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

**Step 2: Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All Instance types Current generation Show/Hide Columns

Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	General purpose	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t2.xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	General purpose	t3a.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes
<input type="checkbox"/>	General purpose	t3a.micro	2	1	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous **Review and Launch** Next: Configure Instance Details

Figure 20: Instance Types

- In this step, as you will see in Figure 21, there are many configuration settings. For more information on these settings, we recommend reviewing the AWS EC2 documentation (<https://docs.aws.amazon.com/ec2/index.html>). We will not make any changes in this section. Click on the Next: Add Storage button.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 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-0ac73c60 (default) Create new VPC

Subnet: No preference (default subnet in any Availability Zone) Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

Placement group: ☐ Add instance to placement group

Capacity Reservation: Open Create new Capacity Reservation

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

T2/T3 Unlimited: ☐ Enable Additional charges may apply

File systems: Add file systems Create new file system

Cancel Previous **Review and Launch** Next: Add Storage

Figure 21: Configure Instance Details



- At this step, storage settings can be made. The storage capacity can be increased by increasing the number in the size section marked in Figure 22. This can be done after the instance is created. On Volume Type section, you can change the disc type. Then click on the Next: Add Tags button.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 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	Encryption
Root	/dev/sda1	snap-0ce94def453a46677	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GiB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

Figure 22: Add Storage

- This step can add tags for volume or instance. Tag is defined as key-value pairs. You can add a new tag with the Add another tag button shown in Figure 23. Then click on the Next: Configure Security Group button.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

**Step 5: Add Tags**  
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

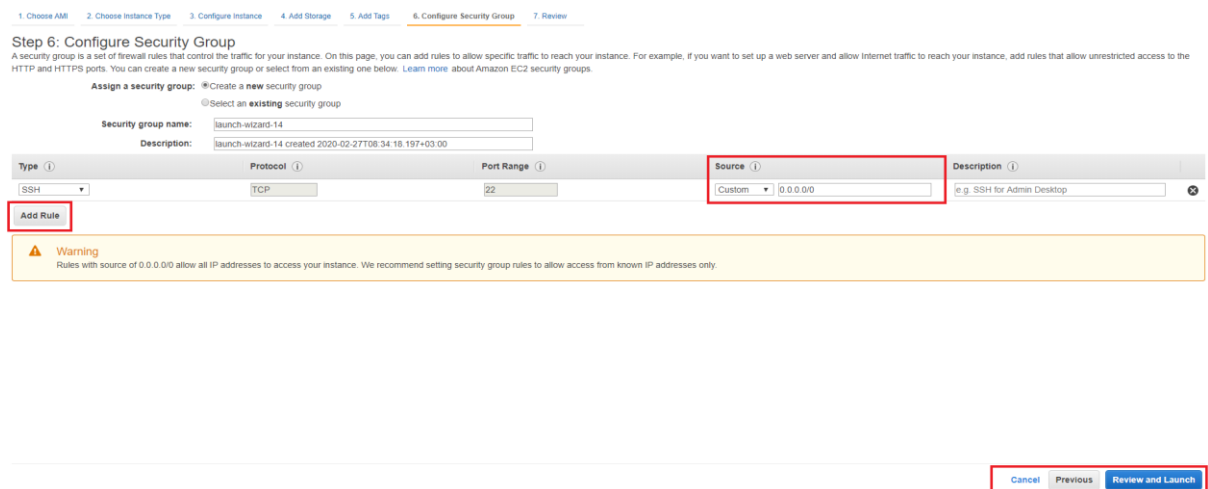
Key (128 characters maximum)	Value (256 characters maximum)	Instances	Volumes
		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Security Group](#)

Figure 23: Add Tags

8. At this stage, you can set where the TCP ports of the instances can be accessed. TCP port 22 is open by default for the SSH connection. Another port can be configured by clicking the Add Rule button shown in Figure 24. In the Source section, it is determined which IP addresses or IP addresses can be connected in the added port. Then click the Review and Launch button.



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 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 name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

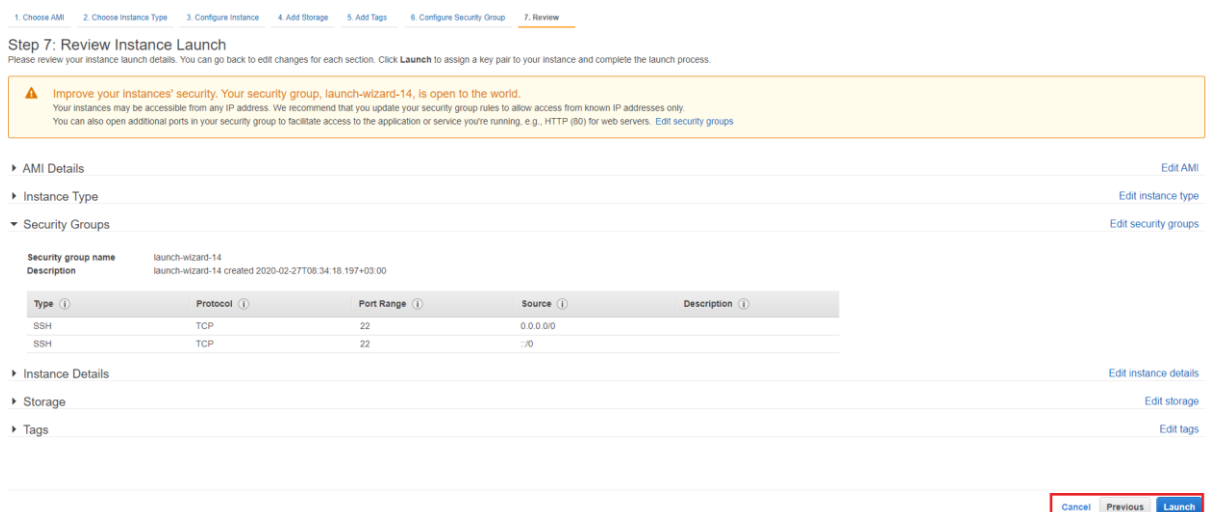
[Add Rule](#)

**Warning**  
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Previous](#) [Review and Launch](#)

Figure 24: Security Groups

9. In this step, you can see the list of all the settings made so far, as shown in figure 25. You can make changes to all the settings you want by pressing the Previous button. After pressing the launch button, your instance begins to be created.



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 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, launch-wizard-14, 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](#)
- Instance Type [Edit instance type](#)
- Security Groups [Edit security groups](#)
  - Security group name: launch-wizard-14
  - Description: launch-wizard-14 created 2020-02-27T08:34:18.197+03:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	:/0	
- Instance Details [Edit instance details](#)
- Storage [Edit storage](#)
- Tags [Edit tags](#)

Figure 25: Review Instance Launch

Then a key pair is created in the window shown in figure 26 or an existing key pair is selected. The created key pair is downloaded and stored. This key pair is used when establishing the SSH connection. Instances are created by clicking the Launch instances button.

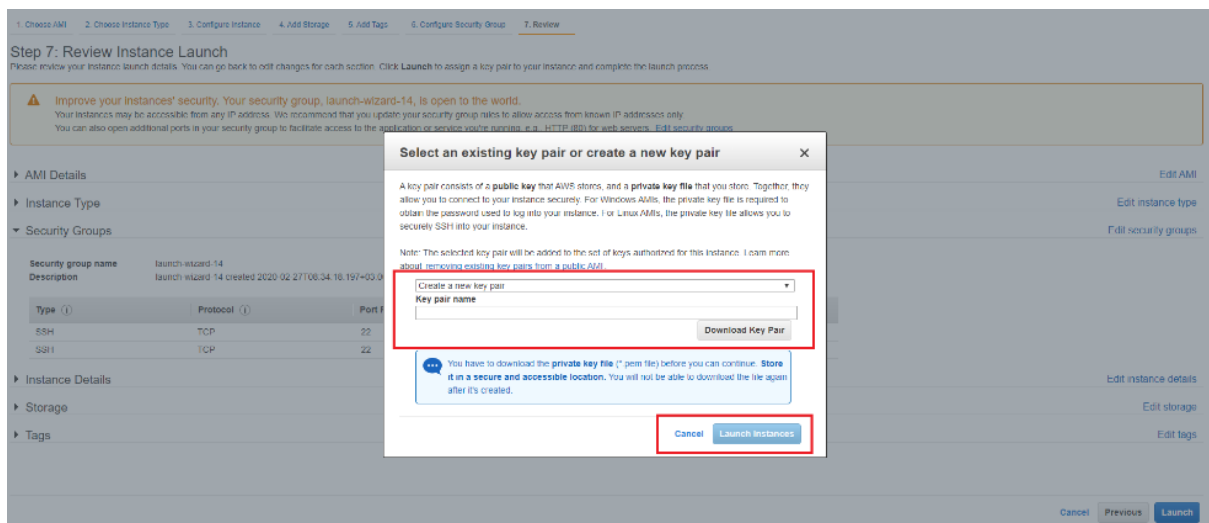


Figure 26: Create Key Pair

10. At this stage, an information page is opened as shown in figure 27. Information on the instance can be accessed on this page. By clicking the View Instances button, all instances can be viewed and managed.

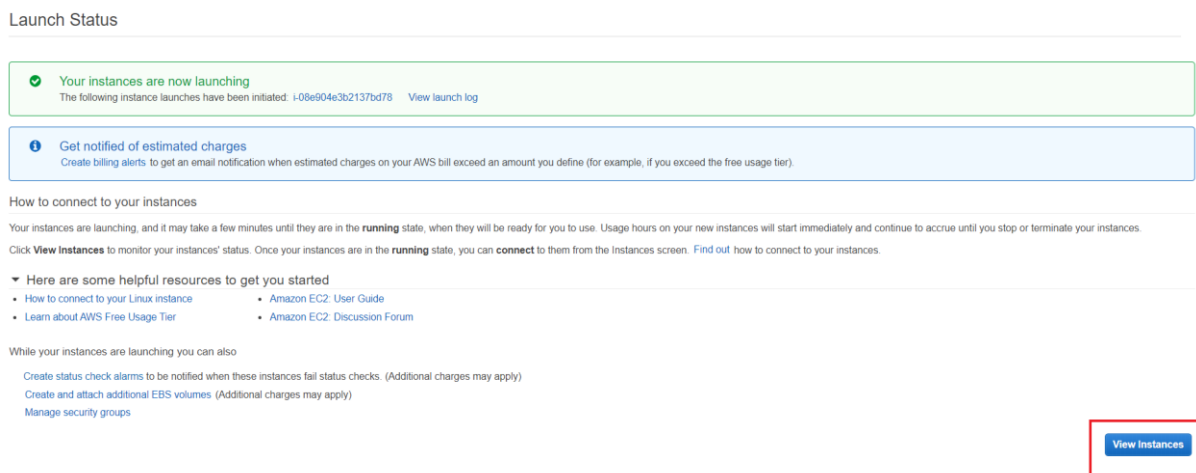


Figure 27: Launch Status



11. On the page in Figure 28, the launched instances are managed. On this page, the status of instances is displayed, new settings are made, or their status can be changed. You can get information about how to connect by clicking on the Connect button shown in Figure 28. With the action button, new settings can be made about the selected instances or these instances can be turned off.

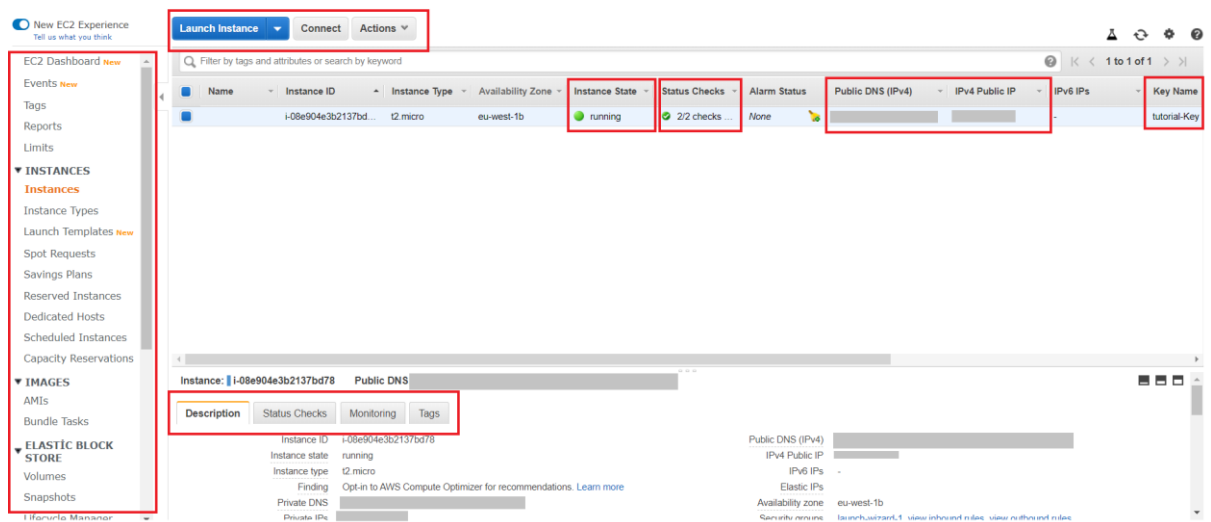


Figure 28: Instances

For more detailed information please visit <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide>.

### 14.3. Systems Manager (SSM)

AWS Systems Manager is an AWS service that you can use to view and control your infrastructure on AWS. Using the Systems Manager console, you can view operational data from multiple AWS services and automate operational tasks across your AWS resources. Furthermore, AWS Systems Manager Agent (SSM Agent) is Amazon software that can be installed and configured on an Amazon EC2 instance, an on-premises server, or a virtual machine (VM). SSM Agent makes it possible for Systems Manager to update, manage, and configure these resources. SSM Agent must be installed on each instance you want to use with Systems Manager.

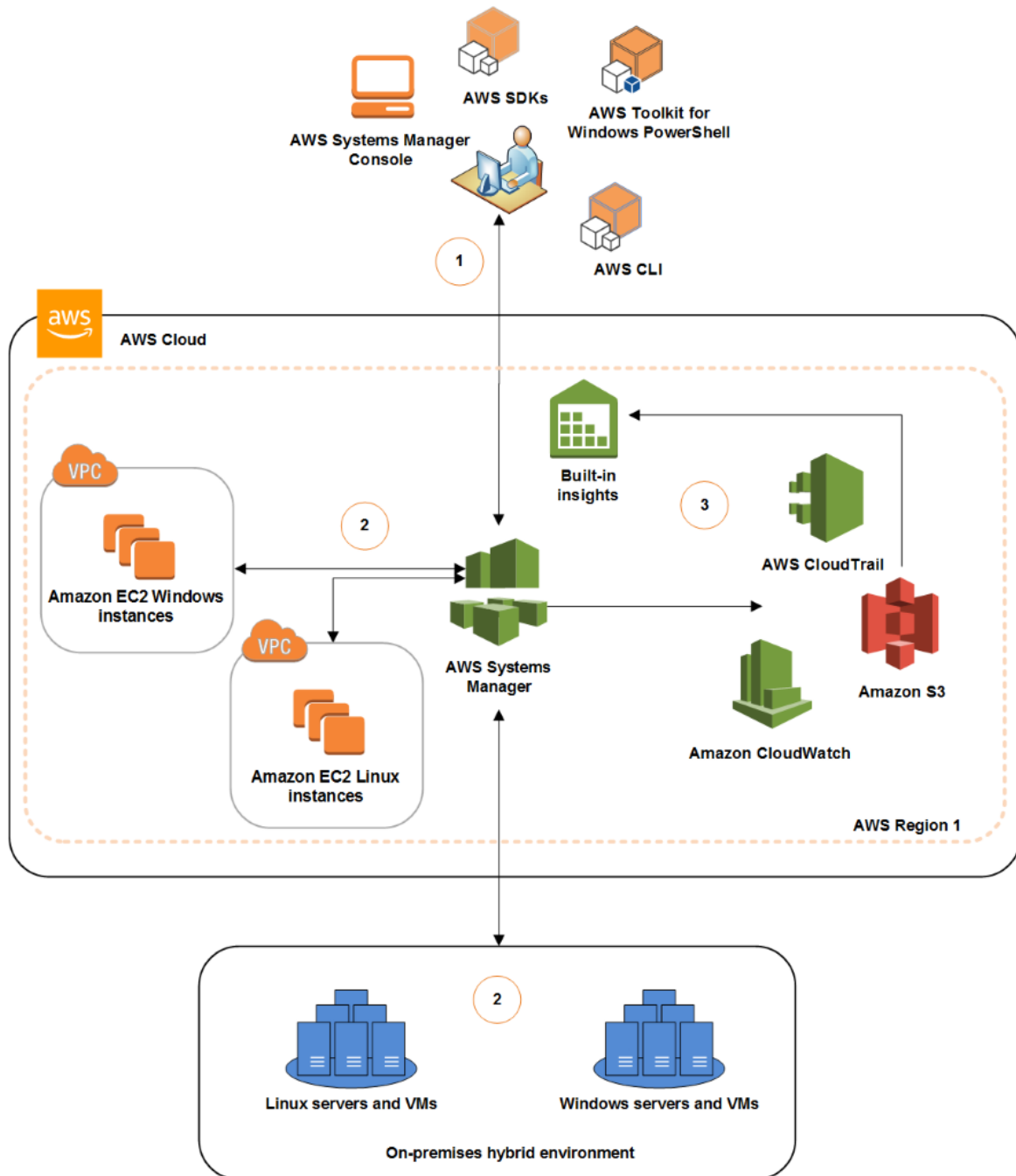


Figure 29: General Example of Systems Manager Process Flow

For more detailed information you can visit [AWS Systems Manager](https://aws.amazon.com/systems-manager/).

## 14.4. AWS Elastic Load Balancing (ELB)

ELB is the AWS service that distributes incoming application and network traffic to many destinations. Thanks to this service, users can use many EC2 instances for the same application and share incoming network traffic to these instances. In this way, you increase the availability and error tolerance. The most effective use scenario can be observed when used with an autoscaling group. According to this scenario, when the load of existing instances increases, this load is directed to new instances created by the autoscaling group. ELB supports three

different load balancer types. These are Application Load Balancers: Network Load Balancers, and Classic Load Balancers. You need to choose the load balancer type that suits your needs.

**Application Load Balancers:** It is ideal for balancing the load of HTTP and HTTPS traffic due to its advanced request routing capability. Application load balancer running on the 7th layer directs the traffic to the destinations according to the content of the request.

**Network Load Balancer:** Network Load Balancer is ideal for balancing the load of Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Transport Layer Security (TLS) traffic, where very high performance is required.

**Classic Load Balancer:** Classic Load Balancer provides basic load balancing features across multiple Amazon EC2 instances and operates at both the request and connection level.

### 14.5. AWS Certificate Manager (ACM)

This service allows creating and managing SSL/TLS certificates for applications running on AWS. With ACM, you can import certificates issued by other certification authorities (CAs) and manage them. Certification authorities are organizations that generate digital certificates. You can create wildcard SSL certificates through ACM. This way, you protect an unlimited number of subdomains. You cannot apply ACM certificates to the web page you want. To be able to apply, you must use one of the AWS services integrated with ACM. AWS services integrated with ACM; Elastic Load Balancing are Amazon CloudFront, AWS Elastic Beanstalk, Amazon API Gateway and AWS CloudFormation services. Certificates produced by ACM are valid for 13 months and these are X.509 version 3 certificates. ACM is available in all AWS regions. However, for a certificate to be valid worldwide, the certificate of this domain must be obtained in all AWS regions. Also, the certificate must be obtained in the US East (N. Virginia) region to use ACM with the Amazon CloudFront service. The AWS Certificate Manager service is free to use. Users only pay for the service they use with ACM. For more information, visit <https://docs.aws.amazon.com/acm/>.

### 14.6. Route53

Amazon Route53 is a highly available and scalable Domain Name System (DNS) web service. Like every service of AWS, Route 53 can work integrated with other services such as EC2 instances, Elastic Load Balancers or S3 buckets. Also; It enables us to direct our users and services from outside AWS to services within AWS. In order to use Route 53, you do not have to buy domains on Amazon, you can also manage the domains you have bought from other companies (eg Godaddy, Name etc.) with Route 53 service. You can also create routing policies using Route 53. The forwarding policies you can create are as follows:

- **Simple Routing:** By using Simple Routing policies, you can ensure that your users access your web page as you do on your Windows or Linux DNS servers. When you use simple routing, you cannot create more than one record with the same name and same type, but you can enter more than one data in the same record.
- **Weighted Routing:** You can use this policy if you have more than one server hosting the same application and you want to assign weight to the traffic directed to these servers.
- **Geoproximity Routing:** Thanks to geoproximity routing, you can ensure that your users can access your resources according to the geographical location.
- **Latency-based Routing:** By using Latency-Based Routing, you can ensure that users are directed to the region with the lowest latency.
- **Failover Routing:** By using Failover Routing policies, you can ensure the establishment of active-passive structures. In this way, you can ensure that your users are automatically directed to the server that responds in case a server is not responding.
- **Geolocation Routing:** Thanks to Geolocation Routing policies, you can ensure that your users coming from a certain region are directed to the servers in a certain region.

When you use geolocation routing, you can make your users continent-based or country-based routing and restriction.

- **Multivalue Answer Records:** Thanks to multivalue records, Route 53 can make multiple returns to DNS queries.

Route 53 service on AWS is a service that can be preferred as it provides monitoring of our servers and services as well as providing many features that we cannot do on normal DNS servers. For more detailed information you can visit [Amazon Route 53](#).

### 14.7. Simple Email Service (SES)

Amazon Simple Email Service (Amazon SES) is a cloud-based email sending service designed to help digital marketers and app developers send marketing, notification, and transactional emails. It is a reliable, low-cost service for businesses of all sizes that communicate with their customers via email.

When you send an email, you are sending it through some type of outbound email server. That email server might be provided by your Internet service provider (ISP), your company's IT department, or you might have set it up yourself. The email server accepts your email content, formats it to comply with email standards, and then sends the email out over the Internet. The email may pass through other servers until it eventually reaches a receiver (an entity, such as an ISP, that receives the email on behalf of the recipient). The receiver then delivers the email to the recipient. The following diagram illustrates the basic email-sending process.



Figure 30: Basic e-mail sending process

When you use Amazon SES, Amazon SES becomes your outbound email server. You can also keep your existing email server and configure it to send your outgoing emails through Amazon SES so that you don't have to change any settings in your email clients. The following diagram shows where Amazon SES fits in to the email sending process.



Figure 31: Amazon SES e-mail sending process

### 14.8. Amazon Simple Storage Service (S3 Buckets)

Amazon S3 or Amazon Simple Storage Service is an object storage service that offers scalability, data accessibility and performance features for the internet environment. It's designed to make web-scale computing easier for developers. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers [2].

## 14.9. CloudFront

Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

- If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately.
- If the content is not in that edge location, CloudFront retrieves it from an origin that you've defined—such as an Amazon S3 bucket, a MediaPackage channel, or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

CloudFront is integrated with AWS – both physical locations that are directly connected to the AWS global infrastructure, as well as other AWS services.

For more detailed information please visit <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html>.

## 14.10. AWS SDK for JavaScript

The AWS SDK for JavaScript provides a JavaScript API for AWS services. The AWS SDK for JavaScript provides JavaScript with AWS services and resources in both browser scripts and Node.js applications. Describes how to set up the SDK, connect to AWS services, and access AWS service features. Also provides Node.js and browser code examples for working with popular Amazon Web Services.

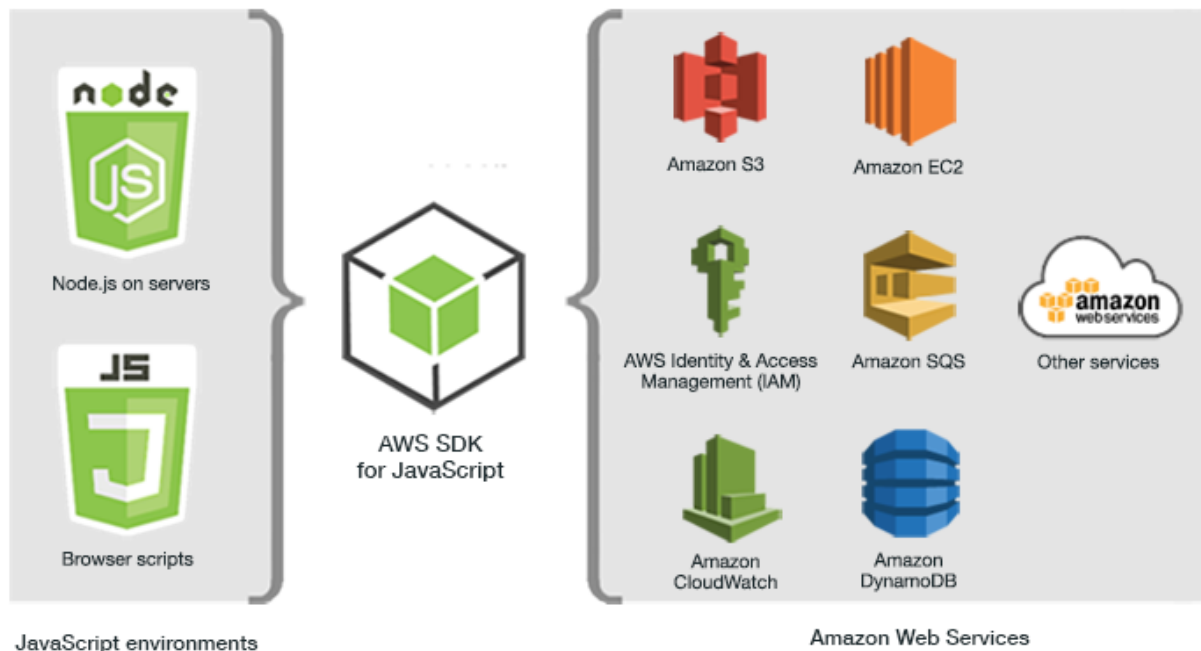


Figure 32: AWS SDK schema for Amazon Web Services

### 6.10.1 Getting Started in Node.js

#### Step 1: Configure Your Credentials:

Credentials must be provided to access the account and resources via Node.js. You can see and update credentials with the code snippet below.

```
var AWS = require("aws-sdk");
AWS.config.getCredentials(function(err) {
  if (err) console.log(err.stack);
  // credentials not loaded
  else {
    console.log("Access key:", AWS.config.credentials.accessKeyId);
    console.log("Secret access key:", AWS.config.credentials.secretAccessKey);
  }
});
AWS.config.update({
  accessKeyId: YOUR_ACCESS_KEY_ID,
  secretAccessKey: YOUR_SECRET_ACCESS_KEY,
  region: YOUR_REGION
});
```

### Step 2: Install the SDK and Dependencies:

You can download this sdk using NPM(Node Package Manager).

```
npm install aws-sdk
```

This command installs the SDK for JavaScript in your project, and updates package.json to list the SDK as a project dependency . These package and their associated code are installed in the node\_modules subdirectory of your projects.

### Step 3: Write the Node.js Code:

Create a new file named example.js to contain the example code. Let's run an EC2 instance with this code.

```
const AWS = require('aws-sdk');

AWS.config.update({
  accessKeyId: YOUR_ACCESS_KEY_ID,
  secretAccessKey: YOUR_SECRET_ACCESS_KEY,
  region: YOUR_REGION
});
const ec2 = new AWS.EC2({ apiVersion: '2016-11-15'});

var params = {
  BlockDeviceMappings: [
    {
      DeviceName: "/dev/sdh",
      Ebs: {
        VolumeSize: 100
      }
    }
  ],
  ImageId: "ami-abc12345",
  InstanceType: "t2.micro",
  KeyName: "my-key-pair",
  MaxCount: 1,
  MinCount: 1,
  SecurityGroupIds: [
    "sg-1a2b3c4d"
  ],
  SubnetId: "subnet-6e7f829e",
```



```
TagSpecifications: [
  {
    ResourceType: "instance",
    Tags: [
      {
        Key: "Purpose",
        Value: "test"
      }
    ]
  }
];
ec2.runInstances(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else console.log(data);              // successful response
});
```

### Step 4: Run the Sample:

Type the following command to run the sample.

```
node example.js
```

For more detailed information you can visit [AWS SDK for JavaScript](#).

## 14.11. AWS Command Line Interface (AWS CLI)

The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. The AWS CLI provides direct access to the public APIs of AWS services. You can explore a service's capabilities with the AWS CLI and develop shell scripts to manage your resources. Or, you can take what you learn to develop programs in other languages by using the AWS SDKs.

### 6.11.1 Installing

Follow these steps from the command line to install the AWS CLI on Linux.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Confirm the installation.

```
aws --version
```

### 6.11.2 Usage

The AWS Command Line Interface (AWS CLI) uses a multipart structure on the command line that must be specified in this order:

1. The base call to the aws program.
2. The top-level command, which typically corresponds to an AWS service supported by the AWS CLI.
3. The subcommand that specifies which operation to perform.
4. General CLI options or parameters required by the operation. You can specify these in any order as long as they follow the first three parts. If an exclusive parameter is specified multiple times, only the last value applies.

```
aws <command> <subcommand> [options and parameters]
```

You can get help on the command line to see the supported services



aws help

the operations for a service,

aws autoscaling help

and the parameters for a service operation.

aws autoscaling create-auto-scaling-group help

For more detailed information you can visit [AWS Command Line Interface](#).

### 15. References

- [1] <https://blog.hyperiondev.com/index.php/2018/09/10/everything-need-know-mern-stack/>
- [2] <https://www.gencayildiz.com/blog/nosql-ve-mongodb-nedir/>
- [3] <https://ceaksan.com/tr/express-js-nedir/>