**Projects**

* [Projects](https://darey.io/project-listings/)
* [Docs](https://www.darey.io/projects/)
* [(STEP 17) PROJECT 7: Devops Tooling Website Solution](https://www.darey.io/docs/devops-tooling-website-solution/)

**DEVOPS TOOLING WEBSITE SOLUTION**

 CompletedSubmit Project for Review

In previous [Project 6(<https://www.darey.io/docs/project-6-step-1/>) you implemented a WordPress based solution that is ready to be filled with content and can be used as a full fledged website or blog. Moving further we will add some more value to our solutions that your DevOps team could utilize. We want to introduce a set of DevOps tools that will help our team in day to day activities in managing, developing, testing, deploying and monitoring different projects.

The tools we want our team to be able to use are well known and widely used by multiple DevOps teams, so we will introduce a single DevOps Tooling Solution that will consist of:

1. [Jenkins](https://www.jenkins.io/) – free and open source automation server used to build [CI/CD](https://en.wikipedia.org/wiki/CI/CD) pipelines.
2. [Kubernetes](https://kubernetes.io/) – an open-source container-orchestration system for automating computer application deployment, scaling, and management.
3. [Jfrog Artifactory](https://jfrog.com/artifactory/) – Universal Repository Manager supporting all major packaging formats, build tools and CI servers. Artifactory.
4. [Rancher](https://rancher.com/products/rancher/) – an open source software platform that enables organizations to run and manage [Docker](https://en.wikipedia.org/wiki/Docker_(software)) and Kubernetes in production.
5. [Grafana](https://grafana.com/) – a multi-platform open source analytics and interactive visualization web application.
6. [Prometheus](https://prometheus.io/) – An open-source monitoring system with a dimensional data model, flexible query language, efficient time series database and modern alerting approach.
7. [Kibana](https://www.elastic.co/kibana) – Kibana is a free and open user interface that lets you visualize your [Elasticsearch](https://www.elastic.co/elasticsearch/) data and navigate the [Elastic Stack](https://www.elastic.co/elastic-stack).

**Note:** Do not feel overwhelmed by all the tools and technologies listed above, we will gradually get ourselves familiar with them in upcoming projects!

Side Self Study

Read about [Network-attached storage (NAS)](https://en.wikipedia.org/wiki/Network-attached_storage), [Storage Area Network (SAN)](https://en.wikipedia.org/wiki/Storage_area_network) and related protocols like NFS, (s)FTP, SMB, iSCSI. Explore what [Block-level storage](https://en.wikipedia.org/wiki/Block-level_storage) is and how it is used by Cloud Service providers, know the difference from [Object storage](https://en.wikipedia.org/wiki/Object_storage).  
On the [example of AWS services](https://dzone.com/articles/confused-by-aws-storage-options-s3-ebs-amp-efs-explained) understand the difference between Block Storage, Object Storage and Network File System.

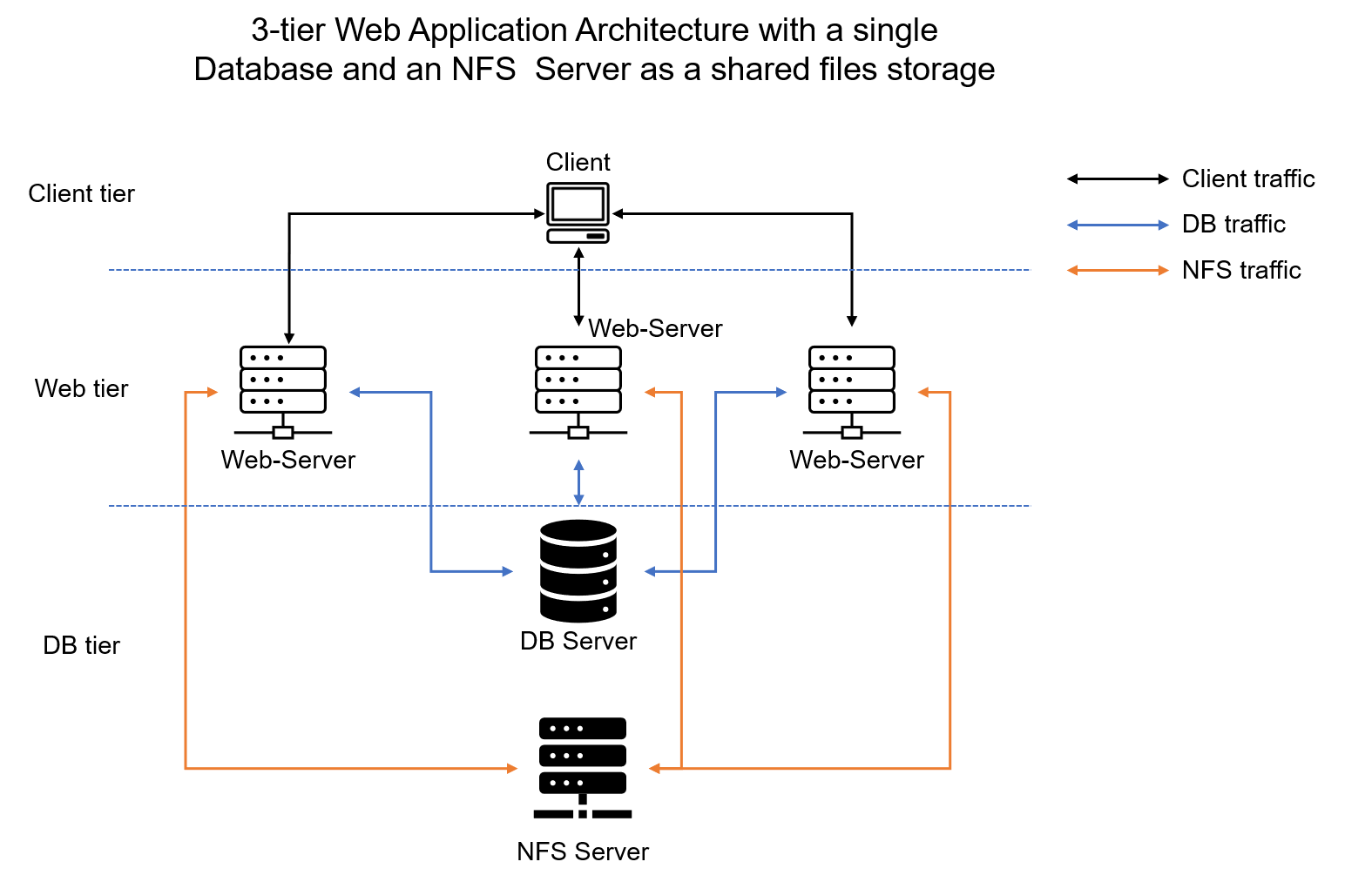
Setup and technologies used in Project 7

As a member of a DevOps team, you will implement a tooling website solution which makes access to DevOps tools within the corporate infrastructure easily accessible.

In this project you will implement a solution that consists of following components:

1. **Infrastructure**: AWS
2. **Webserver Linux**: Red Hat Enterprise Linux 8
3. **Database Server**: Ubuntu 20.04 + MySQL
4. **Storage Server**: Red Hat Enterprise Linux 8 + NFS Server
5. **Programming Language**: PHP
6. **Code Repository**: [GitHub](https://github.com/darey-io/tooling.git)

On the diagram below you can see a common pattern where several stateless Web Servers share a common database and also access the same files using [Network File Sytem (NFS)](https://en.wikipedia.org/wiki/Network_File_System) as a shared file storage. Even though the NFS server might be located on a completely separate hardware – for Web Servers it look like a local file system from where they can serve the same files.



It is important to know what storage solution is suitable for what use cases, for this – you need to answer following questions: what data will be stored, in what format, how this data will be accessed, by whom, from where, how frequently, etc. Base on this you will be able to choose the right storage system for your solution.

Instructions On How To Submit Your Work For Review And Feedback

To submit your work for review and feedback – follow [**this instruction**](https://darey.io/docs/how-to-submit-your-work-for-review-and-feedback-2/).

# STEP 2 — CONFIGURE THE DATABASE SERVER

By now you should know how to install and configure a MySQL DBMS to work with remote Web Server

1. Install MySQL server
2. Create a database and name it tooling
3. Create a database user and name it webaccess
4. Grant permission to webaccess user on tooling database to do anything only from the webservers subnet cidr

Step 3 — Prepare the Web Servers

We need to make sure that our Web Servers can serve the same content from shared storage solutions, in our case – NFS Server and MySQL database.  
You already know that one DB can be accessed for reads and writes by multiple clients. For storing shared files that our Web Servers will use – we will utilize NFS and mount previously created Logical Volume lv-apps to the folder where Apache stores files to be served to the users (/var/www).

This approach will make our Web Servers stateless, which means we will be able to add new ones or remove them whenever we need, and the integrity of the data (in the database and on NFS) will be preserved.

During the next steps we will do following:

* Configure NFS client (this step must be done on all three servers)
* Deploy a Tooling application to our Web Servers into a shared NFS folder
* Configure the Web Servers to work with a single MySQL database

1. Launch a new EC2 instance with RHEL 8 Operating System
2. Install NFS client

sudo yum install nfs-utils nfs4-acl-tools -y

1. Mount /var/www/ and target the NFS server’s export for apps

sudo mkdir /var/www

sudo mount -t nfs -o rw,nosuid <NFS-Server-Private-IP-Address>:/mnt/apps /var/www

1. Verify that NFS was mounted successfully by running df -h. Make sure that the changes will persist on Web Server after reboot:

sudo vi /etc/fstab

add following line

<NFS-Server-Private-IP-Address>:/mnt/apps /var/www nfs defaults 0 0

1. Install [Remi’s repository](http://www.servermom.org/how-to-enable-remi-repo-on-centos-7-6-and-5/2790/), Apache and PHP

sudo yum install httpd -y

sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm

sudo dnf install dnf-utils http://rpms.remirepo.net/enterprise/remi-release-8.rpm

sudo dnf module reset php

sudo dnf module enable php:remi-7.4

sudo dnf install php php-opcache php-gd php-curl php-mysqlnd

sudo systemctl start php-fpm

sudo systemctl enable php-fpm

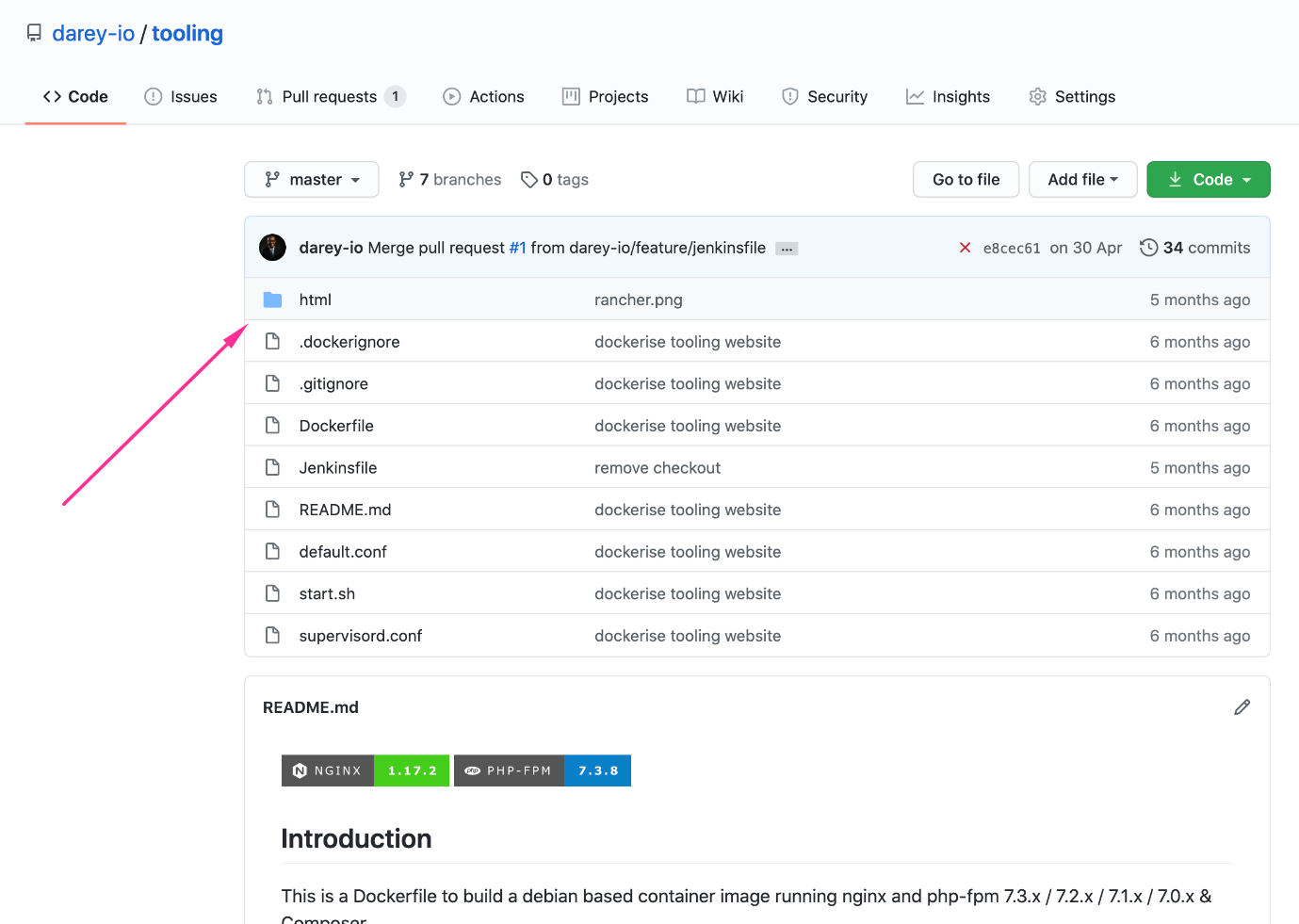
setsebool -P httpd\_execmem 1

**Repeat steps 1-5 for another 2 Web Servers.**

1. Verify that Apache files and directories are available on the Web Server in /var/www and also on the NFS server in /mnt/apps. If you see the same files – it means NFS is mounted correctly. You can try to create a new file touch test.txt from one server and check if the same file is accessible from other Web Servers.
2. Locate the log folder for Apache on the Web Server and mount it to NFS server’s export for logs. Repeat step №4 to make sure the mount point will persist after reboot.
3. Fork the tooling source code from [Darey.io Github Account](https://github.com/darey-io/tooling.git) to your Github account. (Learn how to fork a repo [here](https://youtu.be/f5grYMXbAV0))
4. Deploy the tooling website’s code to the Webserver. Ensure that the **html** folder from the repository is deployed to /var/www/html

**Note 1:** Do not forget to open TCP port 80 on the Web Server.

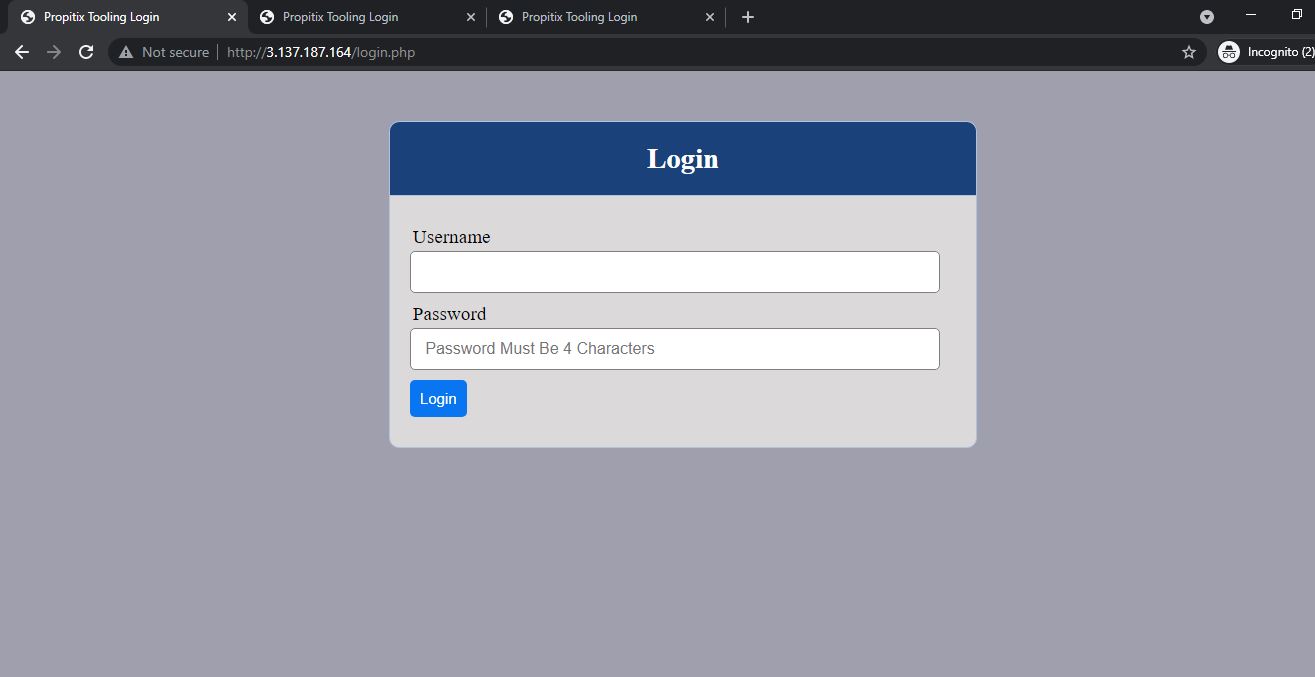
**Note 2:** If you encounter 403 Error – check permissions to your /var/www/html folder and also disable SELinux sudo setenforce 0  
To make this change permanent – open following config file sudo vi /etc/sysconfig/selinux and set SELINUX=disabledthen restrt httpd.



1. Update the website’s configuration to connect to the database (in /var/www/html/functions.php file). Apply tooling-db.sql script to your database using this command mysql -h <databse-private-ip> -u <db-username> -p <db-pasword> < tooling-db.sql
2. Create in MySQL a new admin user with username: myuser and password: password:

INSERT INTO ‘users’ (‘id’, ‘username’, ‘password’, ’email’, ‘user\_type’, ‘status’) VALUES  
-> (1, ‘myuser’, ‘5f4dcc3b5aa765d61d8327deb882cf99’, ‘user@mail.com’, ‘admin’, ‘1’);

1. Open the website in your browser http://<Web-Server-Public-IP-Address-or-Public-DNS-Name>/index.php and make sure you can login into the websute with myuser user.



Congratulations!

You have just implemented a web solution for a DevOps team using LAMP stack with remote Database and NFS servers.

