**Description**

Develop a series of small scripts and configuration files (3 scripts/conf files + 3 scripts, see below) to fetch weather forecasts from the Norwegian Meteorological Institute and republish them on a Web server running on a Raspberry Pi.

#### Expected outcome

1) to create a software that generates and updates a static website, and 2) set up a Raspberry Pi to serve that static website.

Once the project runs, it will be possible to connect with a browser to http://<IP-OF-RASPBERRY-PI/index.html and to read the list of available forecasts for the next hours. (See picture A for an example.)

It will also be possible to click on each city name and see a page with the weather forecast for that city. Each city-specific page will contain the name of the city and the associated meteorological information. (See picture B for an example.)

Picture A:

Et bilde som inneholder tekst

Automatisk generert beskrivelse

Picture B:

Et bilde som inneholder tekst

Automatisk generert beskrivelse

#### Components

The top-level directory of the project must have a README file that describes the project, including what each file does

The project comprises different components, each of which can be developed independently from the others.

In summary:

1. The scraping script reads the meteorological information and extracts the temperature and weather forecast for various cities.
2. The pages script reads information about the extracted meteorological information and creates the necessary HTML pages.
3. The overview script generates a summary HTML page that links to the other generated pages.
4. The repo update script pushes all new files and the updated files to a GitHub repository.
5. The nginx configuration file tells nginx where to find the HTML files to server over HTTP.
6. The crontab entries run the scripts at regular intervals.
7. The notification script sends a notification with the forecast.
8. The deployment script configures a blank installation of Raspberry Pi OS with all that is necessary for the project to work.

All the script must be written in Bash: example of commands (wget, curl, sed, grep, loops, variables etc. )

##### Scraping script

Web scraping means extracting data from a Web page by 1) downloading the HTML page, 2) using text processing tools to clean it, 3) using text processing tools to extract the data you are looking for.

The scraping script does the following actions every time it is run:

1. It fetches the current weather forecast from <https://api.met.no/weatherapi/locationforecast/2.0/classic?lat=LATITUDE&lon=LONGITUDE> for the cities of Gjøvik, Oslo, Trondheim, Bergen and Tromsø
2. It extracts the temperature forecast for the current hour, the next hour and the hour after that.
3. It extracts the next precipitation forecast.
4. It creates a new directory (specific to this session) where it will store the files with the meteorological data.
5. For each city, it creates an information file with 6 lines:
   1. the name of the city;
   2. the temperature forecast for the current hour;
   3. the temperature forecast for the next hour;
   4. the temperature forecast for the next after that;
   5. the precipitation forecast;
   6. the date and time when the meteorological data has been scraped.
   7. On even days also scrape data for Kristiansand.
   8. Also retrieve information about humidity for each city and add it as a seventh line in the information files.

###### Example of output

$ ls scraped-weather/   
2022-10-24-1400/ 2022-10-24-1900/ 2022-10-25-0800/   
$ ls scraped-weather/2022-10-24-1900/   
bergen.txt gjoevik.txt olso.txt tromsoe.txt trondheim.txt   
$ cat scraped-weather/2022-10-24-1900/bergen.txt  
Bergen  
3  
0  
-4  
cloudy   
2022-10-24 19:03   
$

##### Pages script

The pages script first reads all the information stored in the files created by the scraping script and creates a series of HTML pages, one for each city.

Each HTML page contains

* the name of the city
* the date in which it the meteorological data has been scraped last
* the meteorological data that has been scraped last
* a link to the overview page (See "overview script" below)
* Also show temperature and forecasts of the past 2 days.

###### Example of output

$ ls cities/   
2022-10-24-1400/ 2022-10-24-1900/ 2022-10-25-0800/  
$ ls cities/2022-10-25-0800/   
bergen.html gjoevik.html olso.html tromsoe.html trondheim.html   
$ cat cities/2022-10-25-0800/olso.html   
<!DOCTYPE html>   
<html>   
<head>   
<title>Weather forecast for Gjøvik</title>   
[…]   
</head>   
<body>   
<h1>Weather forecast for Gjøvik</h1>  
<p>Temperatures:</p>  
<ul>  
<li>08:00: 3</li>  
<li>09:00: 5</li>  
<li>10:00: 7</li>  
</ul>  
<p>Forecast: cloudy</p>  
<p>Fetched on 2022-10-25 08:03</p>   
[…]

##### Overview script

The overview script generates a HTML overview page (index.html) of all the available city pages.

The overview page contains the a list of city names.

Each city name is linked to the latest HTML page created by the pages script (See "pages script" above).

###### features

Show the current temperature and forecast after the city name.

###### Example of output

$ ls   
index.html cities/ scraped-weather/   
$ cat index.html   
<!DOCTYPE html>   
<head>…</head>   
<body>  
<p>Forecasts updated at 2022-10-25 08:00.</p>  
<ul>   
<li><a href=”cities/2022-10-25-0800/bergen.html”/>Bergen</a><li>   
<li><a href=”cities/2022-10-25-0800/gjoevik.html”/>Gjøvik</a><li>   
[…]

##### Repo update script

The repo update script commits all the new and changed files to the local git repository. It also pushes the new commit to the centralized repository.

The repo update script must be able to run unattended when launched by cron.

##### Nginx configuration file

In order to serve the generated HTML files, you must properly set up Nginx via a configuration file.

The nginx configuration file should follow the Debian/Ubuntu conventions for files in /etc/nginx/sites-available.

###### features

* Instead of setting up nginx to read the generated overview page from the disk, set up nginx to generate the overview page on the fly using CGI via fcgiwrap.

##### Crontab entries

The meteorological data must be scraped every 6 hours.

To achieve this, set up one or more entries in the system crontab to run the right scripts at the right time.

###### tasks

* Create a systemd timer unit (two files) instead of a crontab entry.

##### Notification script

The notification script sends a notification with the current forecast via the ntfy.sh service.

Whenever the script is run, it reads the latest scraped meteorological data and sends a notification via the ntfy.sh service with a summary of the latest forecasts for all cities.

The notifications must be sent to a topic that starts with "idg1100-562963".

##### Deployment script

The deployment script configures a blank installation of Raspberry Pi OS with all that is necessary for the project to work.

The deployment script takes care of, among other things,

* fetching the git repository from GitHub.
* placing all scripts in the right directories.
* generating all directories needed for the script to work properly.
* copying the nginx configuration file for the website and enabling it.
* setting up the crontab.

#### Constraints

* `awk` cannot be used.
* `perl` cannot be used.
* In `sed` only the `s` command can be used (e.g., `sed -e "s/old/NEW"`)

#### Tips

* Use the current date to name the directories and the files where you put the information about the news articles
  + For example:
    - PROJDIR/scraped-weather/20221028-1600/bergen.txt
  + (Read the “FORMAT” section in the manual page of `date`.)
* Set bash to use strict evaluation mode to catch problems (`set -eu`)
* Familiarize yourself with the `-o` / `--only-matching` option of grep.
* Familiarize yourself with the `-A` and `-B` options of grep.
* Enable the extended regexp syntax in sed and grep using the `-E` option.