Guide to create Lektor website and deploy it to VPS server

Step 1: To make scripts on Lektor, installing python is required, so you have to install it.

```
sudo apt update
sudo apt install python3-pip python3-venv
```

Step 2: On the terminal, install Lektor.

```
sudo apt install lektor
```

Step 3: To create a simple Lektor project,

```
lektor quickstart
```

Once you run that command, you will have to enter the project name, the path you want to store your project, if you want to create a blog on the website(an example blog). Then you have to enter the author name(your name) of the project and you're done.

Step 4: You run your project to see the website.

```
cd yourproject
lektor server
```

This will allow you to run the project and see the project on your localhost:5000.

Step 5: Prepare Dreamhost VPS for SSH Access

Create an SSH key pair on your local machine.

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

By default, this will generate a public and private key pair: *~/.ssh/id_rsa* and *~/.ssh/id_rsa.pub.*

Copy the public key (`id_rsa.pub`) to your DreamHost VPS.

```
ssh-copy-id user@your_vps_ip
```

Install Lektor on the DreamHost VPS if not installed.

SSH into your VPS.

```
ssh user@your_vps_ip
```

Install Python and Lektor on your VPS.

```
sudo apt update
sudo apt install python3-pip
pip3 install lektor
```

Step 6: Set Up GitHub Repository with Lektor Project

Push Lektor project to GitHub repository.

```
git init
git add .
git commit -m "Initial commit"
git remote add origin https://github.com/yourusername/your-lektor-project.git
git push -u origin main
```

Step 7: Set Up Travis CI for Deployment

Go to Travis CI and sign in with your Github account.
Authorize Travis CI to access your repository and enable the repository for Travis CI builds.

Step 8: Create the `.travis.yml` Configuration File

Create a *.travis.yml* file in the project's root directory. This file defines how Travis CI will run the build and deployment process.

.travis.yml file

```
language: python
Python:
    - "3.11" # Choose the Python version that works for your project

Install:
    - pip install lektor # Install Lektor on Travis CI

Before_deploy:
    - lektor build # Build the Lektor website

Deploy:
    provider: script
    script: bash deploy.sh # Run deploy script
    On:
        branch: main # Deploy when pushing to the main branch
```

Step 9: Create the *deploy.sh* Script

Create a *deploy.sh* file in the project's root directory. This script will handle the actual deployment to your DreamHost VPS using rsync over SSH.

deploy.sh file

```
#!/bin/bash

# Variables
DEPLOY_DIR="/home/username/public_html/website"  # Adjust to your web root
SERVER_USER="your-vps-username"   # VPS username
SERVER_HOST="your-vps-ip"   # VPS IP address
SSH_KEY_PATH="$HOME/.ssh/id_rsa"   # Path to your private SSH key

# Fail on errors
set -e

# Build the Lektor site
echo "Building the Lektor website..."
lektor build
```

```
# Deploy using rsync
echo "Syncing files to DreamHost VPS..."
rsync -avz --delete --rsh="ssh -i $SSH_KEY_PATH" output/
$SERVER_USER@$SERVER_HOST:$DEPLOY_DIR

echo "Deployment completed successfully!"
```

Step 10: Add the SSH Key to Travis CI
To authenticate the deployment script, you need to encrypt your SSH private key and add it to Travis CI. Encrypt your SSH private key using the Travis CLI.

```
travis encrypt-file ~/.ssh/id_rsa --add
```

This command will encrypt your private key and automatically add it to `.travis.yml` as an environment variable. Travis will automatically decrypt the key before running the deployment script.

Step 11: Push Changes to GitHub
Add the `.travis.yml` and `deploy.sh` files to your GitHub repository:

```
git add .travis.yml deploy.sh
git commit -m "Add Travis CI configuration and deploy script"
git push origin main
```

Push changes to the main branch of GitHub repository, and Travis CI should automatically build and deploy Lektor site to DreamHost VPS.

Step 12: You can add static themes, from Lektor to change the way your website looks. On https://github.com/lektor/lektor-themes.git you will find all the 4 themes. In order to add themes, make a themes folder in your lektor project. Then use *cd themes* to be in that folder. Choose one of the themes and clone it in the themes folder.
For example, downloading nix theme

```
git clone https://github.com/rlaverde/lektor-theme-nix.git
```

Once you do download the themes, make sure you complete the configurations of the themes that are listed in the README file of the theme repository.

Then, in the *yourproject.lektorproject file.* Make sure you add the name of the themes in that file. You can change the file manually by double clicking on the file in file explorer. Or, use nano or vim in the terminal to edit the file.

```
nano yourproject.lektorproject

        [project]
        themes = lektor-theme-nix
```

Save the file, then MAKE SURE TO DELETE the *assets, models and templates* folder in your project directory.

Step 13: Navigate to the `templates/` folder of your theme

```
cd themes/lektor-theme-nix/templates
```

Open the file to edit

```
nano layout.html
```

Add or modify the HTML structure.
For example, add a footer section:

```
<footer>

   <p>© 2024 My Website. All rights reserved.</p>

</footer>
```

Save the file `CTRL+O`, then `CTRL+X` to exit the editor.

Step 14: Update CSS or Add New Styles

Navigate to the assets/ folder to find CSS files:

```
cd themes/lektor-theme-nix/assets
```

Open and edit the CSS file, or create a new one for custom styles:

```
nano style.css
```

Add custom styles to the CSS file.

```
footer {
   background-color: #333;
   color: #fff;
   text-align: center;
   padding: 10px 0;
}
```

Save the file `CTRL+O`, then `CTRL+X` to exit the editor.

Link the updated CSS file in the `layout.html` file:

Open `layout.html` again:

```
nano ../templates/layout.html
```

Add the following line inside the `<head>` section:

```
<link rel="stylesheet" href="{{ this.url_to('style.css') }}">
```

Save the file `CTRL+O`, then `CTRL+X` to exit the editor.

Step 15: Chamfering Content for Different Times of the Day

To dynamically modify content based on the time (e.g., morning or afternoon), use **Jinja2 logic** within your templates:

Open the template file (e.g., layout.html or page.html):

```
nano layout.html
```

Add Jinja2 conditional logic to display different content:

```
{% set current_hour = now().hour %}
{% if current_hour < 12 %}
   <h1>Good Morning!</h1>
   <p>Welcome to the website. Here's what's happening this morning.</p>
{% else %}
   <h1>Good Afternoon!</h1>
   <p>Enjoy the afternoon updates and features.</p>
{% endif %}
```

Save the file and rebuild the project:

```
lektor build
```

Step 16: Add Dynamic Assets for Morning and Afternoon

You can also serve different images, styles, or scripts depending on the time:

Modify your templates to load assets conditionally:

```
{% if current_hour < 12 %}
   <img src="{{ this.url_to('morning-banner.jpg') }}" alt="Morning Banner">
{% else %}
   <img src="{{ this.url_to('afternoon-banner.jpg') }}" alt="Afternoon Banner">
{% endif %}
```

Place the corresponding files (morning-banner.jpg and afternoon-banner.jpg) in the assets/ directory.

Drag the image to the fold name them morning-banner.jpg and afternoon-banner.jpg

Step 17: Run

Run the server to view changes in real-time:

```
lektor build
lektor server
```