Linux Migration Compass

IaaS Setup and Web Framework Documention

Release version 1.0 – 09/04/2025

This document provides a comprehensive guide to setting up the Linux Migration Compass IaaS project.

Name: Muhammad Talha Ijaz
ID: 3541463

VM IP Address: http://172.171.244.182/
Domain: https://linuxmigration.tech/

GitHub Link to resources: https://github.com/Talha-Ijaz-Qureshi/ICT171-Assignment2

Table of Contents

Project Specification

IaaS Description

The project is hosted on Microsoft Azure, using a B1s tier VM. The domain provider is .Tech. Packages installed and used by this project on the VM are Apache2, certbot.

Web Framework

The website is built using the SvelteKit framework. Configured with Typescript. Packages installed and used by the framework are carbon-components-svelte and Iconify. Project Specification

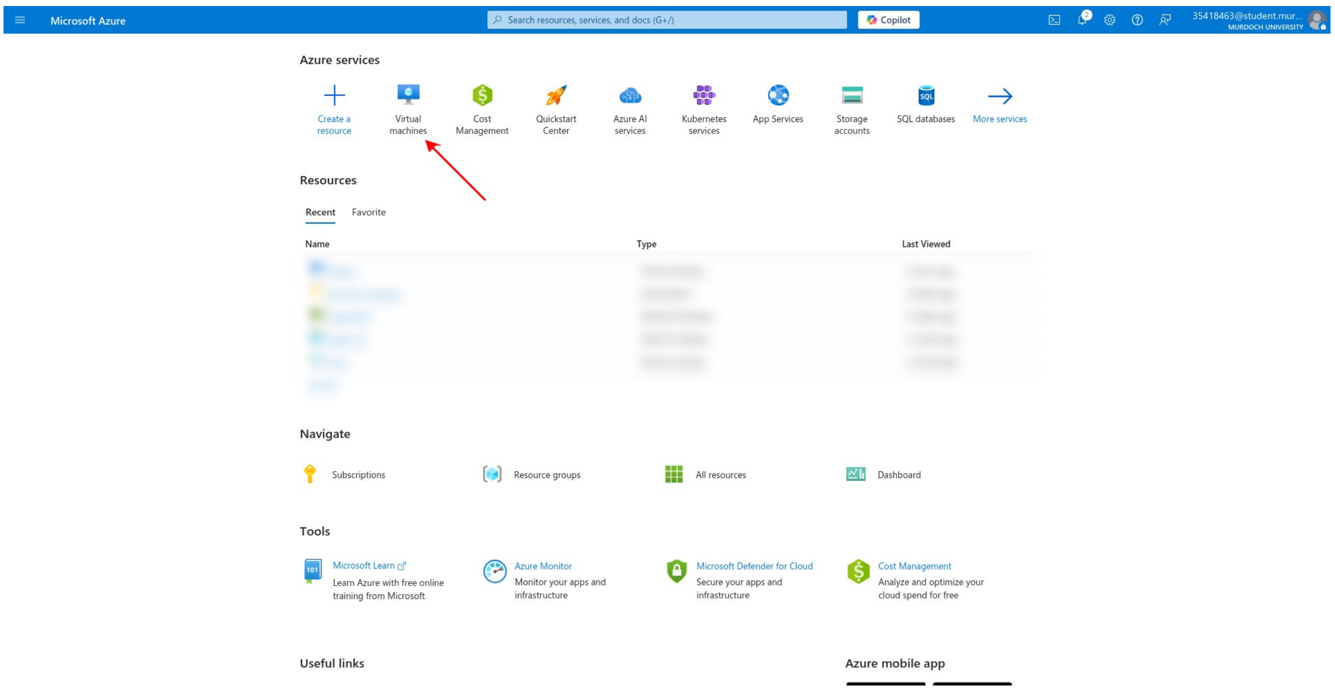
Project Description

The Linux Migration Compass is a comprehensive guide for users to transition to Linux. The website is built using SvelteKit, making use of Svelte’s high reactivity paired with modern components to create a highly efficient, fast and sleek website. The framework has a bespoke content rendering pattern developed from scratch. Allowing for more customization and personalization opportunities, eliminating the need for known but older solutions like WordPress and etc.

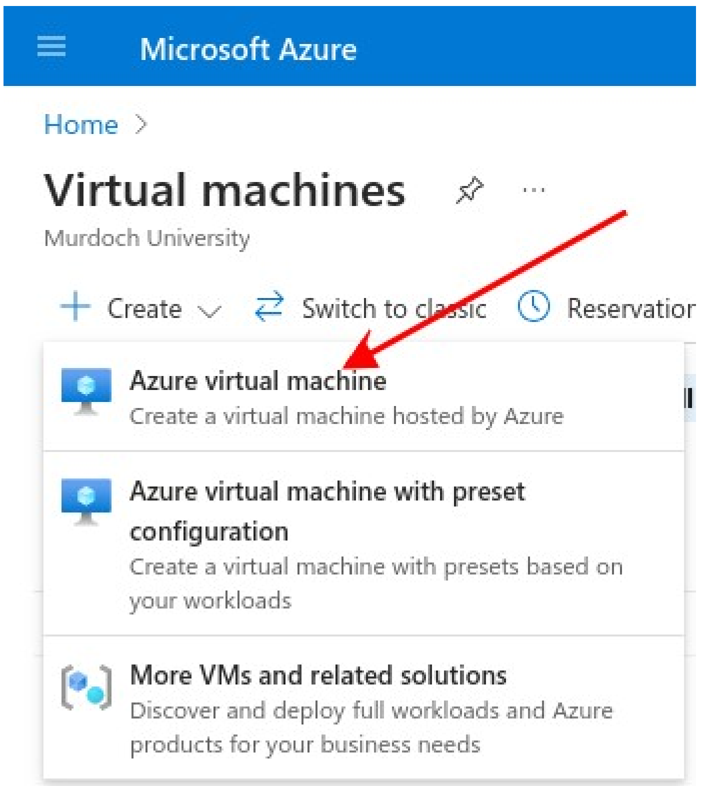
Setting Up The Virtual Machine

Creating a Virtual Machine

To create a VM. Go to your azure portal https://portal.azure.com/#home



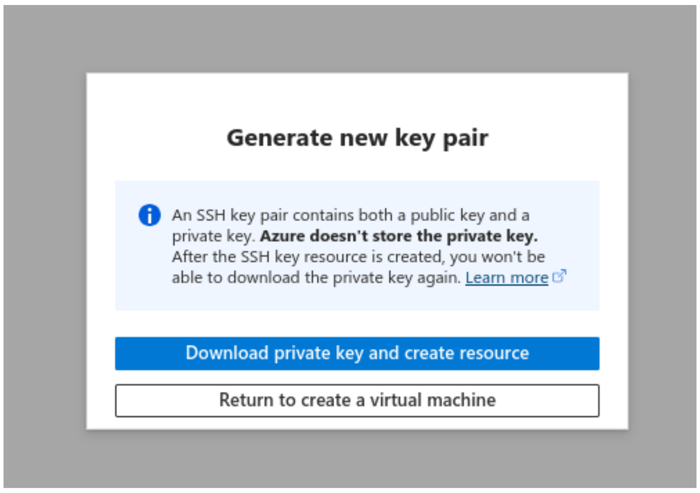
Click “Create” then click “Azure virtual machine”


Microsoft Azure
Home
Virtual machines
Murdoch University
Create
Switch to classic
Reservation
Azure virtual machine
Create a virtual machine hosted by Azure
Azure virtual machine with preset configuration
Create a virtual machine with presets based on your workloads
More VMs and related solutions
Discover and deploy full workloads and Azure products for your business needs

You will then be prompted with a page to enter your virtual machine's options and preferences. The recommended important fields may be filled with the following information.

1. Virtual machine name: linuxmigration
2. Region: East US
3. Zone: zone 3 only
4. OS Image: Ubuntu 22.04 x64
5. Instance: B1s
6. Inbound Ports: select HTTP, HTTPS and SSH (must)

Review any information, then click "Review + Create". You will then be prompted with the option to download private keys. Click "Download private key and create resource", this is needed for us to connect to our VM via our local system.

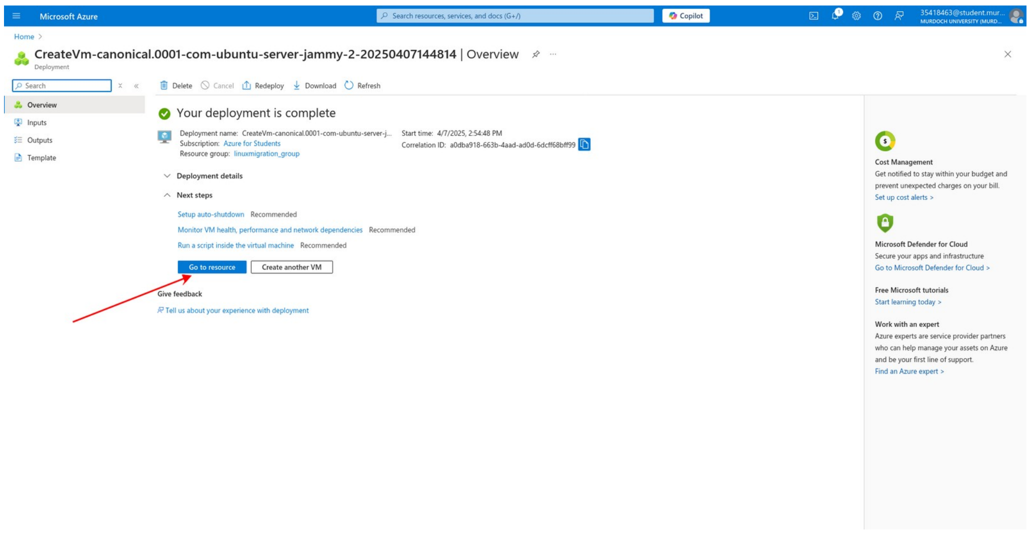


Generate new key pair

An SSH key pair contains both a public key and a private key. **Azure doesn't store the private key.** After the SSH key resource is created, you won't be able to download the private key again. Learn more

Download private key and create resource
Return to create a virtual machine

Once the VM has been successfully deployed. We can now go to our VM's configuration.



CreateVm-canonical.0001-com-ubuntu-server-jammy-2-20250407144814 | Overview

Deployment

Your deployment is complete

Deployment name: CreateVm-canonical.0001-com-ubuntu-server-j...
Subscription: Azure for Students
Resource group: linuxmigration_group

Start time: 4/7/2025, 2:54:48 PM
Correlation ID: a0dba918-663b-4aad-ad0d-6dcff68bff99

Deployment details

Next steps

Setup auto-shutdown Recommended
Monitor VM health, performance and network dependencies Recommended
Run a script inside the virtual machine Recommended

Give feedback

Tell us about your experience with deployment

Cost Management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >

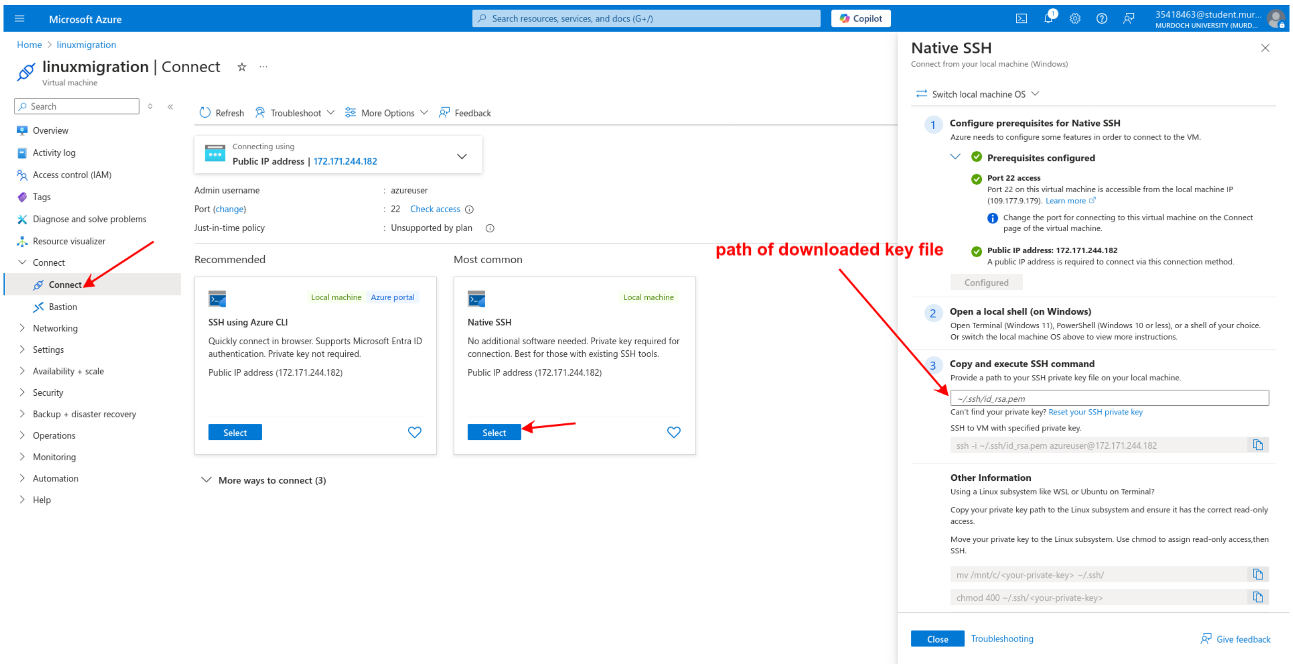
Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials
Start learning today >

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
Find an Azure expert >

Connecting to Our Virtual Machine

Under Connect > Connect, select Native SSH



Connecting using
Public IP address | 172.171.244.182

Admin username : azureuser
Port (change) : 22 Check access
Just-in-time policy : Unsupported by plan

SSH using Azure CLI
Quickly connect in browser. Supports Microsoft Entra ID authentication. Private key not required.
Public IP address (172.171.244.182)

Native SSH
No additional software needed. Private key required for connection. Best for those with existing SSH tools.
Public IP address (172.171.244.182)

Native SSH
Connect from your local machine (Windows)

1 **Configure prerequisites for Native SSH**
Azure needs to configure some features in order to connect to the VM.

Prerequisites configured

Port 22 access
Port 22 on this virtual machine is accessible from the local machine IP (109.177.9.179). Learn more

Change the port for connecting to this virtual machine on the Connect page of the virtual machine.

Public IP address: 172.171.244.182
A public IP address is required to connect via this connection method.

2 **Open a local shell (on Windows)**
Open Terminal (Windows 11), PowerShell (Windows 10 or less), or a shell of your choice. Or switch the local machine OS above to view more instructions.

3 **Copy and execute SSH command**
Provide a path to your SSH private key file on your local machine.

~/.ssh/id_rsa.pem

Can't find your private key? Reset your SSH private key

SSH to VM with specified private key.

```
ssh -i ~/.ssh/id_rsa.pem azureuser@172.171.244.182
```

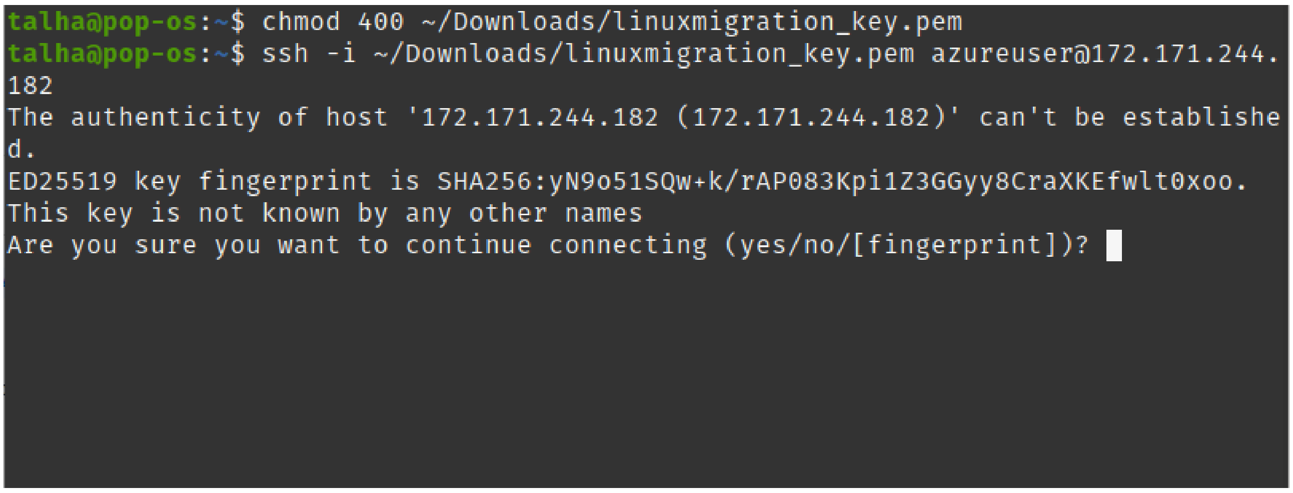
Other Information
Using a Linux subsystem like WSL or Ubuntu on Terminal?

Copy your private key path to the Linux subsystem and ensure it has the correct read-only access.

Move your private key to the Linux subsystem. Use chmod to assign read-only access,then SSH.

```
mv /mnt/c/<your-private-key> ~/.ssh/
chmod 400 ~/.ssh/<your-private-key>
```

Switch between your local machine's OS. And enter the path the key file we downloaded earlier. You will then be able to copy a command you can enter into your terminal.



```
talha@pop-os:~$ chmod 400 ~/Downloads/linuxmigration_key.pem
talha@pop-os:~$ ssh -i ~/Downloads/linuxmigration_key.pem azureuser@172.171.244.
182
The authenticity of host '172.171.244.182 (172.171.244.182)' can't be establishe
d.
ED25519 key fingerprint is SHA256:yN9o51SQw+k/rAP083Kpi1Z3GGyy8CraXKEfwlt0xoo.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? 
```

Enter "yes" this will add the VM's SSH fingerprint to our local machine, or as a known trusted device.

You will now be logged into the VM, indicated by the shell prompt reading "azureuser@linuxmigration:~$"

Initiating the Web Server

Downloading Apache

To download apache, run the following commands:

```
sudo apt update
sudo apt install apache2
```

Enter “y” when prompted “Do you want to continue?”

Now the Apache web server should have started. To test this, visit the VM’s Public IP.

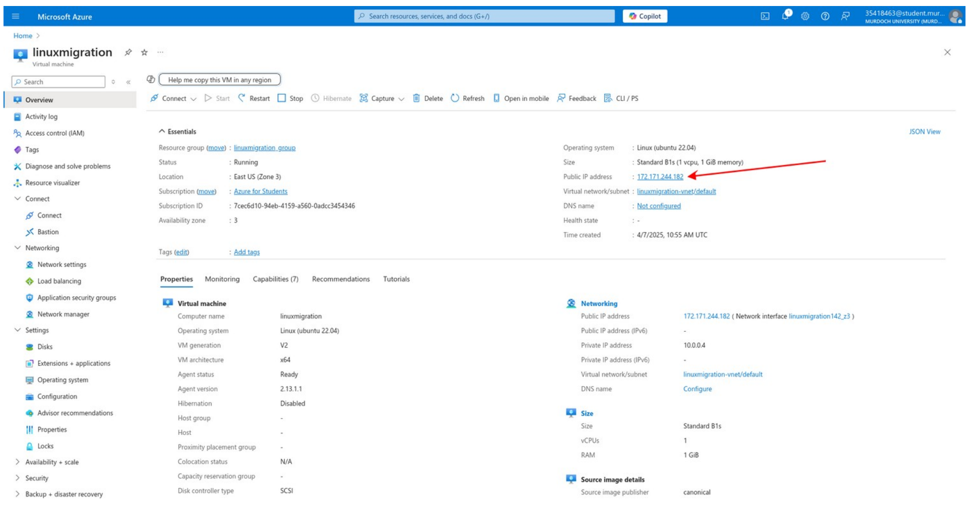
http://172.171.244.182/

If you see the apache default page. The web server has initiated.

Linking our VM to Domain

Verifying Static IP

In your VM’s overview page, click the public IP address



Essentials

Resource group (move)	: linuxmigration_group
Status	: Running
Location	: East US (Zone 3)
Subscription (move)	: Azure for Students
Subscription ID	: 7cec6d10-94eb-4159-a560-0adcc3454346
Availability zone	: 3
Operating system	: Linux (ubuntu 22.04)
Size	: Standard B1s (1 vcpu, 1 GiB memory)
Public IP address	: 172.171.244.182
Virtual network/subnet	: linuxmigration-vnet/default
DNS name	: Not configured
Health state	: -
Time created	: 4/7/2025, 10:55 AM UTC
Tags (edit)	: Add tags

Virtual machine

Computer name	linuxmigration
Operating system	Linux (ubuntu 22.04)
VM generation	V2
VM architecture	x64
Agent status	Ready
Agent version	2.13.1.1
Hibernation	Disabled
Host group	-
Host	-
Proximity placement group	-
Colocation status	N/A
Capacity reservation group	-
Disk controller type	SCSI

Networking

Public IP address	172.171.244.182 (Network interface linuxmigration142_z3)
Public IP address (IPv6)	-
Private IP address	10.0.0.4
Private IP address (IPv6)	-
Virtual network/subnet	linuxmigration-vnet/default
DNS name	Configure

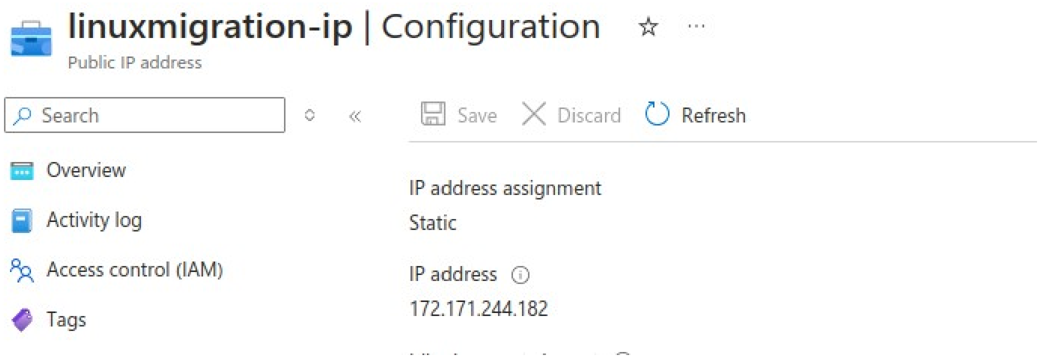
Size

Size	Standard B1s
vCPUs	1
RAM	1 GiB

Source image details

Source image publisher	canonical

Under IP address assignment, ensure it is set to Static.



linuxmigration-ip | Configuration

Public IP address

IP address assignment
Static

IP address
172.171.244.182

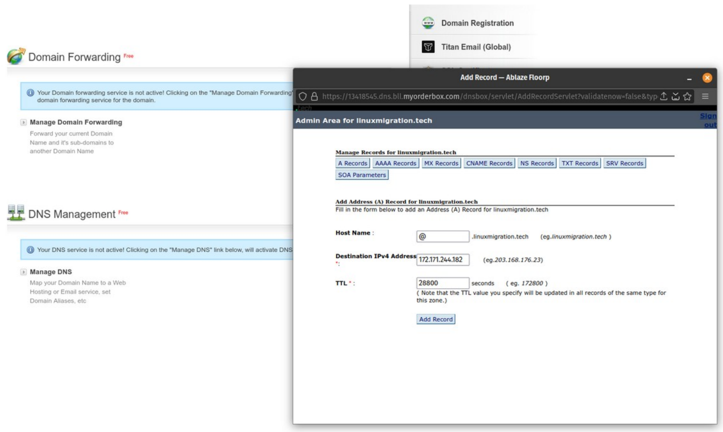
If it is not static. Perform the following steps

1. Select your VM in the home portal.
2. In the left menu, select Properties
3. Under Public IP address\DNS name label, select your IP address.
4. Under DNS name label, enter the prefix “linuxguide”.
5. Select Save at the top of the page.
6. Select Overview in the left menu to return to the VM overview blade.
7. Verify that the DNS name appears correctly.
8. Finally verify the DNS name or the public IP in your browser, the apache default page must appear.

Adding DNS Record to Domain Provider

Go to https://get.tech/, sign in. Click “Manage Orders”, then click on the available domain.

Scroll to the bottom, find “DNS Management” then click “Manage DNS” and enter the following details.



Your Domain forwarding service is not active! Clicking on the "Manage Domain Forwarding" ... domain forwarding service for the domain.

Manage Domain Forwarding
Forward your current Domain Name and it's sub-domains to another Domain Name

Your DNS service is not active! Clicking on the "Manage DNS" link below, will activate DNS

Manage DNS
Map your Domain Name to a Web Hosting or Email service, set Domain Aliases, etc

Add Record — Ablaze Floorp

https://13418545.dns.bll.myorderbox.com/dnsbox/servlet/AddRecordServlet?validatenow=false&typ

Admin Area for linuxmigration.tech

Manage Records for linuxmigration.tech

Add Address (A) Record for linuxmigration.tech
Fill in the form below to add an Address (A) Record for linuxmigration.tech

Host Name : @ .linuxmigration.tech (eg.linuxmigration.tech)

Destination IPv4 Address *: 172.171.244.182 (eg.203.168.176.23)

TTL * : 28800 seconds (eg. 172800)
(Note that the TTL value you specify will be updated in all records of the same type for this zone.)

Host Name: @

Destination IPv4: Our VM’s public IP, [172.171.244.182](172.171.244.182)

TTL: Unchanged

Click “Add Record”

After the record is created it usually takes about an hour for DNS propagate, but it can sometimes take up to 2 days to make changes globally. After some period visit the URL http://linuxmigration.tech/ to ensure the apache default page appears. If so, we have successfully connected our domain to the VM. We can now begin with creating our project.

Creating the Web Framework

Prerequisites

Ensure your system has NodeJS installed. You can download it from https://nodejs.org/en/download depending on your system’s OS.

Run the command:

```
node -v
npm -v.
```

They both must output a version.

Downloading SvelteKit

To download SvelteKit, run the command:

```
npx sv create linux-guide
```

You will then be prompted several options to configure your SvelteKit app, ensure you have select Typescript syntax. Other options are unnecessary for this project. This will download SvelteKit’s scaffold.

SvelteKit Scaffold Directory Layout

```
linux-guide/
├── src/
│   ├── lib/
│   │   └── (Will be used to store components)
│   ├── routes/
│   │   ├── +page.svelte
│   │   └── +layout.svelte
│   │   └── (pages are added here)
│   ├── app.html
│   ├── app.css
│   └── app.d.ts
├── static/
│   ├── favicon.png
│   └── (other static assets like images, fonts, etc.)
├── .gitignore
├── package.json
├── svelte.config.js
├── tsconfig.json
├── vite.config.js
└── README.md
```

Run the command to install any dependencies:

```
cd linux-guide # change directory to the project
npm install
```

Then run the following command to install Carbon Design Components and Iconify.

```
npm install carbon-components carbon-components-svelte sass @carbon/styles
@iconify/svelte
```

The website will have a table of contents, the main content and a preference menu.

Making the Table of Contents File

Create a new folder in 'src/lib/' named 'components'. And in components create a file named 'Toc.svelte'

Copy the following code into the file

```
<script lang="ts"> // Must include lang="ts" for all script tags

// Importing styles, components and themes
import { onMount } from 'svelte';
import { Accordion, AccordionItem } from 'carbon-components-svelte';
import "carbon-components-svelte/css/all.css";
import contentData from '$lib/data/content';
import type { Section } from '$lib/data/content';
import Icon from "@iconify/svelte";

//Declaring our reactive variables
const sections: Section[] = contentData.sections;
```

```
let activeSection: string = '';
let activeSubtopicId: string = '';
let observer: IntersectionObserver | null = null;
let isMenuOpen: boolean = false;
let buttonAnimateClass = "";
// This function formats normal text like What is Pop!_OS? to what-is-pop_os
// This allows us to dynamically create hyperlinks in this file, and it’s
// Corresponding anchor tags on the main content.
// In our main page which we will code soon.
function slugify(text: string): string {
      if (!text) return '';
      return text
      .toLowerCase()
      .trim()
      .replace(/\s+/g, '-')
      .replace(/[^\w-]+/g, '')
      .replace(/--+/g, '-')
      .replace(/^-+/, '')
      .replace(/-+$/, '');
}

// This function handles the smooth scrolling when an item in the table of
// content is clicked

function handleLinkClick(event: Event, href: string) {
      event.preventDefault();
      const targetId = href.slice(1);
      const targetElement = document.getElementById(targetId);

      if (targetElement) {
            const viewportHeight = window.innerHeight;
            const offset = viewportHeight * 0.1;
            const elementTop = targetElement.getBoundingClientRect().top +
            window.scrollY;
            const adjustedTop = elementTop - offset;

            window.scrollTo({
                  top: adjustedTop,
                  behavior: 'smooth'
            });
// This variable is set to False for mobile devices, which closes the menu
            isMenuOpen = false;
      }
}
// This function is implemented for making the component responsive with
// different devices. It toggles the toc menu.
function toggleMenu() {
      isMenuOpen = !isMenuOpen;
      buttonAnimateClass = isMenuOpen ? "btn-animate" : "";

}
// onMount is a life cycle hook which updates and renders components in runtime
// Depending on how we interact.
OnMount(() => {
// This dynamically generates the table of contents from the imported data file
// and
      const subtopicIds: string[] = [];
      sections.forEach(section => {
            section.subtopics.forEach(subtopic => {
                  const subtopicSlug = slugify(subtopic.title);
                  const subtopicId = `${section.id}-${subtopicSlug}`;
```

```
                subtopicIds.push(subtopicId);
            });
        });

// This function allows the table of contents to automatically expand
// the visible section, and collapses the others. And being able to keep
// track of what section is currently being viewed.
        const handleIntersection: IntersectionObserverCallback = (entries) => {
            const intersectingEntries = entries.filter(entry =>
            entry.isIntersecting);
            if (intersectingEntries.length === 0) return;
            let topmostEntry = intersectingEntries[0];
            for (let i = 1; i < intersectingEntries.length; i++) {
                if (intersectingEntries[i].boundingClientRect.top <
                topmostEntry.boundingClientRect.top) {
                topmostEntry = intersectingEntries[i];
                }
            }

            if (topmostEntry.target instanceof HTMLElement) {
                const id = topmostEntry.target.id;
                activeSubtopicId = id;
                const sectionId = id.split('-')[0];
                activeSection = sectionId;
            }
        };
// Configured so the observer fires when the element appears in top 20% of
// the viewport
        const observerOptions: IntersectionObserverInit = {
            rootMargin: "0px 0px -80% 0px",
            threshold: 0.0
        };

        observer = new IntersectionObserver(handleIntersection, observerOptions);
// Begin Observing
        subtopicIds.forEach(id => {
            const el = document.getElementById(id);
            if (el) {
                if (observer) {
                    observer.observe(el);
                }
            } else {
            console.warn(`element with ID '${id}' not found.`);
            }
        });
// Stop observing the element once it is unmounted
        return () => {
            subtopicIds.forEach(id => {
                const el = document.getElementById(id);
                if (el && observer) observer.unobserve(el);
            });
            observer?.disconnect();
        };
});

</script>
<!-- This component dynamically renders the table of contents from our data file
-->
<nav class:open={isMenuOpen} aria-label="Table of contents">
  <h1>Linux Migration Compass</h1>
    <Accordion>
```

```
      {#each sections as section (section.id)}
        <AccordionItem open={activeSection === section.id}>
          <span slot="title">{section.title}</span>
          <ul>
            {#each section.subtopics as subtopic (subtopic.title)}
             {@const subtopicSlug: string = slugify(subtopic.title)}
             {@const subtopicHref: string = `#${section.id}-${subtopicSlug}`}
             <li>
            <a href={subtopicHref} on:click={(event) => handleLinkClick(event,
            subtopicHref)}>
              {activeSubtopicId === `${section.id}-${subtopicSlug}` ? '> ' : ''}
              {subtopic.title}
            </a>
            </li>
           {/each}
          </ul>
        </AccordionItem>
      {/each}
    </Accordion>

</nav>
<!-- Button for toggling the toc on smaller devices -->
<button class="menu-toggle {buttonAnimateClass}" on:click={toggleMenu} aria-
      label="Toggle menu">
      {#if !isMenuOpen}
      <Icon icon="carbon:side-panel-open" width="32" height="32" style="color:
      #78a9ff" />
      {:else}
      <Icon icon="carbon:side-panel-close-filled" width="32" height="32"
      style="color: #78a9ff" />
      {/if}
</button>
</style>
      /* Replace content with CSS for Toc.svelte which is provided at the end of
the document. Toc’s styles are configured to be responsive
 */
</style> <!-- End of file -->
```

The Toc.svelte file dynamically generates a table of content, depending on the data module’s structure. It also ensures users are aware of where they are on the main page, as the generated ID’s of each topic in the table of contents corresponds to the generated ID’s in the main page. Clicking on a topic also smoothly scrolls to the selected topic and updates the table of content’s cursor and collapses/expands appropriately. The component is responsive, and hides away neatly for smaller devices and by default remains visible on larger devices.

Making the Preferences File

Create the file in ‘src/lib/components’ named ‘rightpanel.svelte’

Copy the following code into the file

```
<script lang="ts">
import { Select, SelectItem, Theme } from "carbon-components-svelte";
import Icon from "@iconify/svelte";
// Default theme is g100, or dark
let selected = "g100";
// Reactive variables
let isPreferencesOpen: boolean = false;
let buttonAnimateClass = "";
// Toggles the preferences panel for smaller devices
function togglePreferences() {
      isPreferencesOpen = !isPreferencesOpen;
      buttonAnimateClass = isPreferencesOpen ? "btn-animate" : "";
}
</script>
// Applies the user's choice to theme, which globally applies the selected theme
<Theme theme={selected} />



<!-- Dropdown menu to select theme, binds the value to the variable, which theme
assigns itself to -->
<div class="preferences-panel" class:open={isPreferencesOpen}>
      <h1 style="opacity: 0.8;">Preferences</h1>
      <Select labelText="Carbon theme" bind:selected>
            <SelectItem value="white" text="White" />
            <SelectItem value="g10" text="Gray 10" />
            <SelectItem value="g80" text="Gray 80" />
            <SelectItem value="g90" text="Gray 90" />
            <SelectItem value="g100" text="Dark" />
      </Select>
</div>
<!-- Button for toggling the toc on smaller devices -->
<button class="preferences-toggle {buttonAnimateClass}"
      on:click={togglePreferences} aria-label="Toggle preferences">
      {#if !isPreferencesOpen}
      <Icon icon="carbon:settings-adjust" width="32" height="32" style="color:
      #78a9ff" />
      {:else}
      <Icon icon="carbon:close-filled" width="32" height="32" style="color:
      #78a9ff" />
      {/if}
</button>


<style>
      /* Replace content with CSS for rightpanel.svelte which is provided at the
end of the document. rightpanel.svelte styles are configured to be responsive
 */
</style> <!-- End of file -->
```

The rightpanel.svelte file is a preference menu for users to select their choice of theme, and then applies the selected theme globally across the entire site. The component is responsive, and hides away neatly for smaller devices and by default remains visible on larger devices.

Making the Main Page

Create the file in ‘src/routes/’ named ‘+page.svelte’

Copy the following code into the file

```
<script lang="ts">
// Importing styles, components and themes
import { Content, InlineNotification, CodeSnippet } from 'carbon-components-
svelte';
import content from '$lib/data/content';
import "carbon-components-svelte/css/all.css";
// Formatting text to a format we can assign IDs to, like how we generated
// the slugified ID as hyperlinks in Toc.svelte, here the slugified IDs
// Will correspond to the hyperlinks. This allows for Toc.svelte to link to
// and keep track of the content you are scrolling through
function slugify(text: string): string {
      if (!text) return '';
      return text
            .toLowerCase()
            .trim()
            .replace(/\s+/g, '-')
            .replace(/[^\w-]+/g, '')
            .replace(/--+/g, '-')
            .replace(/^-+/, '')
            .replace(/-+$/, '');
}
// This function parses through the text and converts links to clickable
// hyperlinks
function parseUrls(text: string): string {
      const urlRegex = /(https?:\/\/[^\s]+)/g;
      return text.replace(urlRegex, '<a href="$1" target="_blank" rel="noopener
noreferrer">$1</a>');
}
// Expandable function to format text, can call more parsing functions if
// present
function formatText(text: string): string {
      let formatted = parseUrls(text);
      return formatted;
}
</script>
<!-- Title, visible on mobile devices only -->
<h1 class="hide">Linux Migration<br>Compass</h1>
<!-- This is the main page’s content rendering pattern, it takes the structure
data module and renders them based on the given data →

<Content>
  {#each content.sections as section}
    <section id={section.id}>
      <h2>{section.title}</h2>
      <p>{section.intro}</p>
      {#each section.subtopics as subtopic}
        {@const subtopicSlug: string = slugify(subtopic.title)}
        {@const subtopicId: string = `${section.id}-${subtopicSlug}`}
        <h3 id={subtopicId}>{subtopic.title}</h3>
        {#each subtopic.blocks as block}
          {#if block.type === 'text'}
            <p style="margin: 1em 0 .2em 0;">{@html formatText(block.value)}</p>
          {:else if block.type === 'code'}
```

```
            <CodeSnippet code={block.value} />
          {:else if block.type === 'note'}
            <InlineNotification
              kind={block.kind}
              title={block.message}
              subtitle={block.value}
              hideCloseButton
              lowContrast
            />
          {/if}
        {/each}
      {/each}
      <hr />
    </section>
  {/each}
</Content>


</style>
      /* Replace content with CSS for +page.svelte which is provided at the end
of the document. Main page’s styles are configured to be responsive
 */
</style> <!-- End of file -->
```

The page dynamically renders the content based on the provided structure specification

```
Section (Interface)
├── id: string
├── title: string
├── intro: string
└── subtopics: Subtopic[]
    ├── Subtopic (Interface)
    │   ├── title: string
    │   └── blocks: Block[]
    │       ├── Block (Interface)
    │       │   ├── type: BlockType (Union Type)
    │       │   │   ├── 'text'
    │       │   │   ├── 'code'
    │       │   │   └── 'note'
    │       │   ├── value: string
    │       │   ├── kind?: (Optional Union Type)
    │       │   │   ├── 'error'
    │       │   │   ├── 'info'
    │       │   │   ├── 'info-square'
    │       │   │   ├── 'success'
    │       │   │   ├── 'warning'
    │       │   │   └── 'warning-alt'
    │       │   └── message?: string
    │       └── …
    └── …
```

The rendered content is accordingly the data in the structured module. The Toc is implemented very similarly, while the main page goes ahead a step and renders the content blocks, which hold paragraphs, code snippets, information/warning boxes.

Making the Layout Page

Create the file in ‘src/routes/’ named ‘+layout.svelte’. A layout page in Svelte is what renders other pages and components within it. Pages and components can simply be slotted into this file, which will give us our complete page.

Copy the following code into the file

```
<script lang="ts">
// Importing the components we created, carbon components, styles and theme
import Toc from '$lib/components/Toc.svelte';
import RightPanel from '$lib/components/rightpanel.svelte';
import { Grid, Row, Column } from 'carbon-components-svelte';
import "carbon-components-svelte/css/all.css";
import Icon from "@iconify/svelte";
// Getting year to update it in footer
const currentYear = new Date().getFullYear();
</script>
<!-- Serves as a head tag to our final page -->
<svelte:head>
      <title>Linux Migration Compass</title>
<!-- Define viewport settings for correct mobile rendering -->
      <meta name="viewport" content="height=device-height,
            width=device-width, initial-scale=1.0,
            minimum-scale=1.0, maximum-scale=1.0,
            user-scalable=no, target-densitydpi=device-dpi">
</svelte:head>
<!-- Final page layout -->
<Grid>
    <Row>
       <Column sm={4} md={2} lg={4}>
          <Toc /> <!-- Table of contents component -->
       </Column>
       <Column sm={8} md={8} lg={8}>
          <slot /> <!-- main page slots here. Main page and footer will be in
                        the middle column -->
       <footer class="footer">
         <div class="footer-content">
            <h1>Linux Migration Compass</h1>
            <h2>Copyleft - {currentYear} Linux Migration Compass - Talha
            Ijaz</h2>
            <a href="https://github.com/Talha-Ijaz-Qureshi/linux-guide"><h2>View
            Source Code</h2></a>
            <div class="logo">
                  <Icon icon="carbon:shuffle" width="530" height="530"
                  style="color:#78a9ff" />
            </div>
            <div class="lic">
               <h3>
                  License Rationale - GNU General Public License v3.0
               </h3>
               <p>
                  This work is licensed under the GNU General Public License
                  v3.0. You may copy, distribute and modify this work under the
                  terms of the GNU GPL version 3 or any later version.
               </p>
```

```
                <p>
                   This program is distributed in the hope that it will be
                   useful, but WITHOUT ANY WARRANTY; without even the implied
                   warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
                   PURPOSE. See the GNU General Public License for more details.
                </p>
                <p>
                   For more information, see:
                   <ahref="https://www.gnu.org/licenses/gpl-
                   3.0.en.html">https://www.gnu.org/licenses/gpl-3.0.en.html</a>
                </p>
             </div>
             <div class="speshal">
               <Icon icon="carbon:linux" width="32" height="32" style="color:
               #78a9ff" />
               <Icon icon="carbon:earth-filled" width="32" height="32"
               style="color: #78a9ff" />
             </div>
          </div>
       </footer>
       </Column>
       <Column sm={4} md={2} lg={2}>
          <RightPanel /> <!-- our preferences panel component -->
       </Column>
    </Row>
</Grid>

</style>
      /* Replace content with CSS for +layout.svelte which is provided at the
end of the document. Layout page styles are configured to be responsive
 */
</style> <!-- End of file →
```

Adding a Favicon

Under ‘static/’ create a new svg file named fav.svg. And enter the following content

```
<svg xmlns="http://www.w3.org/2000/svg" width="530" height="530" class="logo"
viewBox="0 0 32 32">
<path fill="#78a9ff" d="M22.59 19.41L26.17 23h-6.62l-4.37-7l4.37-7h6.62l-3.58
3.59L24 14l6-6l-6-6l-1.41 1.41L26.17 7h-6.62a2 2 0 0 0-1.69.94L14 14.11l-3.86-
6.17A2 2 0 0 0 8.45 7H2v2h6.45l4.37 7l-4.37 7H2v2h6.45a2 2 0 0 0 1.69-.94L14
17.89l3.86 6.17a2 2 0 0 0 1.69.94h6.62l-3.58 3.59L24 30l6-6l-6-6Z"></path>
</svg>
```

Then go to ‘src/app.html’ and edit the

```
<link rel="icon" href="%sveltekit.assets%/favicon.png" />
to
<link rel="icon" href="%sveltekit.assets%/fav.svg" />
```

Testing the Framework’s Dynamic Rendering

Adding the Data Module File

We have now successfully created our Web Framework. The web contents are available to download at https://github.com/Talha-Ijaz-Qureshi/linux-migration-compass-content

We have already accounted for declarations and importing in our framework, now we can simply add the ‘data’ file into ‘src/lib/’

Now lets run our framework. In your terminal, ensure you’re currently in your project directory. Then run the following command

```
npm run dev
```

Then enter the local host server link into your browser

http://localhost:<port number>/

You should now be able to see the finalized website. To test our content rendering pattern scripting, we can change the data file and see the framework rendering content dynamically based on the data provided. In the data folder we downloaded there are two data modules; content.ts and testcontent.ts, the default file the framework is using is ‘content.ts’, go to your +page.svelte, and Toc.svelte files, and change the imports

In Toc.svelte

```
import contentData from '$lib/data/content';
to
import contentData from '$lib/data/testcontent';

(you do not need to change this, as this statement is only importing the data
type definitions, which are the same for both files.)
import type { Section } from '$lib/data/content';
```

In +page.svelte

```
import content from '$lib/data/content';
to
import content from '$lib/data/testcontent';
```

You will now observe how our content rendering pattern scripting dynamically renders content for any structured data module that matches our structure specification

Deployment to Our VM

Preparing the Framework

To deploy our framework, we must first configure our adapter settings. In the root directory of the project, find 'svelte.config.js' Edit the contents so it matches this.

```
import adapter from '@sveltejs/adapter-static';
import { vitePreprocess } from '@sveltejs/vite-plugin-svelte';

/** @type {import('@sveltejs/kit').Config} */
const config = {
      preprocess: vitePreprocess(),

      kit: {
            adapter: adapter({
            pages: 'build',
            assets: 'build',
            fallback: 'index.html'
            })
      }
};

export default config;
```

Then install the static adapter, in your terminal run:

```
npm install --save-dev @sveltejs/adapter-static
```

Now run the following command to build the app:

```
npm run build
```

In the root directory you will observe a new folder called build appears, the build files are what we will use to deploy the framework.

Transferring the Build to Our VM

Now SSH into our VM incase we've been logged out and update it and ensure Apache is running:

ssh -i ~/Downloads/linuxmigration_key.pem azureuser@172.171.244.182
sudo apt update && sudo apt upgrade
sudo systemctl status apache2

On your local machine, transfer the build files into our VM:

scp -i ~/Downloads/linuxmigration_key.pem -r <path to project>/linux-guide/build
azureuser@172.171.244.182:/home/azureuser/

Check whether the build folder appears in our VM:

```
ls
```

The build has been copied to our VM.

Setting up the Build for Apache

If ‘build’ is printed. Then transfer the build files to Apache’s Directory:

```
sudo mv /home/username/build/* /var/www/html/
```

Then set permission for Apache to read the files:

```
sudo chown -R www-data:www-data /var/www/html/
sudo chmod -R 755 /var/www/html/
```

Then finally restart Apache

```
sudo systemctl restart apache2
```

Our final website should now be accessible at the VM’s public IP, and our domain.

http://172.171.244.182

http://linuxmigration.tech/

Configuring Domain for Apache

We need to configure apache to recognize our domain linuxmigration.tech. For this, open up apache’s config file:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Then modify the config file with the content:

```
<VirtualHost *:80>
    # Existing records here
      ..
      ..
</VirtualHost>
```

to include ServerName:

```
<VirtualHost *:80>
      ServerName linuxmigration.tech
      # Existing records here
      ..
      ..
</VirtualHost>
```

Test for any syntax errors in apache config:

```
sudo apache2ctl configtest
```

And finally restart Apache

```
sudo systemctl restart apache2
```

SSL Certification for Domain

Installing Certbot

To get HTTPS on our domain, we can install the SSL certificate for our domain with certbot.

```
sudo apt update
sudo apt install certbot python3-certbot-apache -y
```

Then run certbot:

```
sudo certbot –apache
```

You will then be prompted several options to setup the SSL certificate.

1. Enter the email muhammadtalhaijaz2005@gmail.com,

2. Accept the terms and service

3. Deny sharing email

4. Select our domain

Certbot will soon issue an SSL certificate to our domain.

SSL Certificate Verification

Once certbot has completed issuing, go to apache’s config file to see any new SSL configurations:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

You should see the following records in <VirtualHost *:443>

```
<VirtualHost *:80>

      # Existing records here
      ..
      ..
      SSLCertificateFile /etc/letsencrypt/live/linuxmigration.tech/fullchain.pem

      SSLCertificateKeyFile
/etc/letsencrypt/live/linuxmigration.tech/privkey.pem

      Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
```

Restart Apache

```
sudo systemctl restart apache2
```

Finally, our website should now be SSL certified and HTTPS will appear in the URL.

https://linuxmigration.tech/

https://linuxmigration.tech/

About Linux Migration Compass

License

Author

Original Author and Repository Owner:

Muhammad Talha Ijaz - https://github.com/Talha-Ijaz-Qureshi/linux-guide

Documentation Information

Documentation Release Verison: 1.0

Release Date: 09/04/2025

Project Information and Acknowledgments

The codes and scripts written in this project are all the original author’s own original idea and property.

The JS framework is SvelteKit by https://svelte.dev/

The Icons used are from https://icon-sets.iconify.design/

The UI components used are from https://svelte.carbondesignsystem.com/

Code Substitutions

This section has all the codes that were needed to be substituted into places as aforementioned.

Toc.svelte CSS Styles

```
<style>
@import
url('https://fonts.googleapis.com/css2?
family=IBM+Plex+Sans:wght@300;400;600&fa
mily=IBM+Plex+Serif:wght@300;400;600&dis
play=swap');

h1 {
font-size: 1.5rem;
margin-bottom: .5em;
margin-top: 0;
margin-right: auto;
margin-left: auto;
font-family: 'IBM Plex Serif', serif;
font-weight: 600;
color: #78a9ff;
font-stretch: 100%;
}

nav {
padding: 1rem 0;
border-right: 1px solid var(--cds-
border-subtle, #e0e0e0);
width: fit-content;
position: fixed;
top: 0;
left: 0;
height: 100%;
display: flex;
align-items: center;
justify-content: flex-start;
flex-direction: column;
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
z-index: 1001;
backdrop-filter: blur(1em);
-webkit-backdrop-filter: blur(1em);
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
box-shadow: 2px 0 5px rgba(0, 0, 0,
0.1);

}

.btn-animate {
transform: translateX(85vw);
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
}
```

```
.menu-toggle {
display: none;
position: fixed;
top: 1rem;
left: .5rem;
font-size: 1.5rem;
background: none;
border: none;
cursor: pointer;
color: #78a9ff;
z-index: 100;
}
@media (min-width: 1200px) {
nav {
transform: translateX(0);
}
.menu-toggle {
display: none;
}
}

@media (max-width: 1199px) {
nav {
transform: translateX(-100%);
border-right: 1px solid var(--cds-
border-subtle, #e0e0e0);

}
nav.open {
transform: translateX(0);
}
}

ul {
list-style: none;
padding-left: 1rem;
margin-top: 0.5rem;
}

li {
margin: 0.5rem 0;
}

a {
color: var(--cds-link-primary, #0f62fe);
text-decoration: none;
font-size: var(--cds-body-compact-01-
font-size, 0.875rem);
}
```

```
a:hover {
text-decoration: underline;
color: var(--cds-link-primary-hover,
#0043ce);
}
```

```
@media (max-width: 1199px) {
.menu-toggle {
display: block;
}
}
</style>
```

rightpanel.svelte CSS Styles

```

.preferences-toggle {
display: none;
position: fixed;
top: 4rem;
left: .5rem;
z-index: -100;
font-size: 1.5rem;
background: none;
border: none;
cursor: pointer;
color: #78a9ff;
z-index: 100;
transform: translateX(0);
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
}

.btn-animate {
transform: translateX(85vw);
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
}

.preferences-panel {
position: fixed;
top: 0em;
right: 5em;
height: 100%;
width: 250px;
padding: 1rem;
box-shadow: -2px 0 5px rgba(0, 0, 0,
0.1);
transition: transform 200ms cubic-
bezier(1, 0, 0.01, 1);
z-index: 1000;
backdrop-filter: blur(1em);
-webkit-backdrop-filter: blur(1em);
}
.preferences-toggle {
display: none;
}
```

```
@media (min-width: 768px) and (max-
width: 1199px) {
.preferences-panel {
position: fixed;
left: 0;
transform: translateX(-100%);
border-right: 1px solid var(--cds-
border-subtle, #e0e0e0);
box-shadow: 2px 0 5px rgba(0, 0, 0,
0.1);

width: 250px;

}
.preferences-panel.open {
transform: translateX(0);
}
.preferences-toggle {
display: block;
}

}

@media (max-width: 767px) {
.preferences-panel {
transform: translateX(-100%);
width: 300px;
position: fixed;
left: 0;
border-right: 1px solid var(--cds-
border-subtle, #e0e0e0);
box-shadow: 2px 0 5px rgba(0, 0, 0,
0.1);
}
.preferences-panel.open {
transform: translateX(0);
}
.preferences-toggle {
display: block;
}
}
```

+page.svelte CSS Styles

```
@import
url('https://fonts.googleapis.com/css2?
family=IBM+Plex+Sans:wght@300;400;600&fa
mily=IBM+Plex+Serif:wght@300;400;600&dis
play=swap');
```

```

:global(ol) {
margin: .1em;
margin-left: 2em;
line-height: 2;
```

```css
}
.hide {
display: none;
}
h1 {
font-size: 1.5rem;
margin-left: 2rem;
margin-bottom: 0rem;
font-family: 'IBM Plex Serif', serif;
font-weight: 600;
color: #78a9ff;
font-stretch: 100%;
position: relative;
}
h2 {
font-size: 2.5rem;
/* color: #c6c6c6; */
margin: 2rem 0 1rem;
font-family: 'IBM Plex Serif', serif;
opacity: .9;
}
h3 {
font-size: 1.5rem;
margin: 1.5rem 0 0.5rem;
opacity: .9;
}

hr {
border: 0;
border-top: 1px solid #393939;
margin: 2rem 0;
}
p {
line-height: 2;
}

@media (max-width: 1199px) {
h1 {
font-size: 1.2rem;
margin-left: 2rem;
}
h2 {
font-size: 1.9rem;
}
h3 {
font-size: 1.5rem;
}
p {
font-size: 0.95rem;
line-height: 1.7;
}
.hide {
display: block;
}
}
```