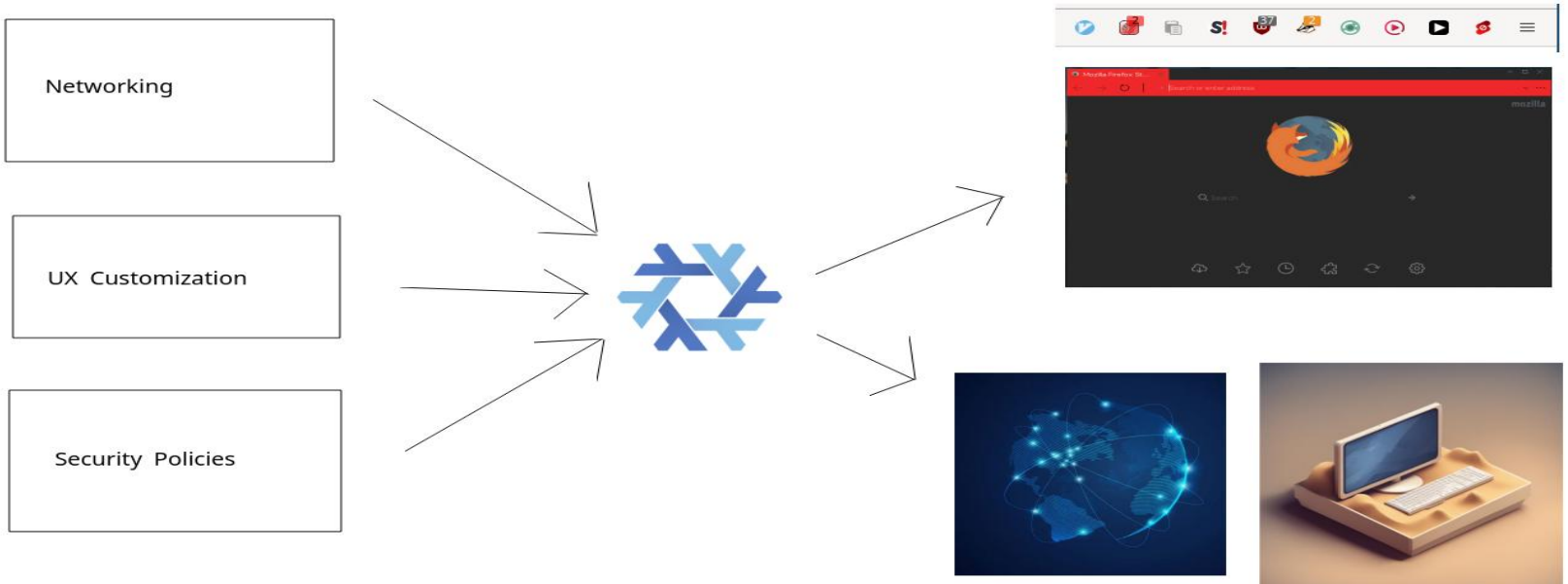
Single Website Firefox VMs with NixOS

Nitin Passa



Disclaimer

This document is being distributed for informational and educational purposes only and is not an offer to sell or the solicitation of an offer to buy any securities or other instruments. The information contained herein is not intended to provide, and should not be relied upon for, investment advice. The views expressed herein are not necessarily the views of Two Sigma Investments, LP or any of its affiliates (collectively, "Two Sigma"). Such views reflect the assumptions of the author(s) of the document and are subject to change without notice. The document may employ data derived from third-party sources. No representation is made by Two Sigma as to the accuracy of such information and the use of such information in no way implies an endorsement of the source of such information or its validity.
The copyrights and/or trademarks in some of the images, logos or other material used herein may be owned by entities other than Two Sigma. If so, such copyrights and/or trademarks are most likely owned by the entity that created the material and are used purely for identification and comment as fair use under international copyright and/or trademark laws. Use of such image, copyright or trademark does not imply any association with such organization (or endorsement of such organization) by Two Sigma, nor vice versa.

NixOS Makes VMs Easy

firefox-vm.nix

```
{pkgs, ... }:
let
  config = (pkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [{
      imports = [
        "${pkgs}/nixos/modules/virtualisation/qemu-vm.nix"
        ./base-system.nix    # hostname + xorg + users
        ./autostart-ff.nix
      ];
      virtualisation.memorySize = 8192;
    }];
  }).config;
in
config.system.build.vm
```

NixOS Makes VMs Easy

firefox-vm.nix

```
{pkgs, ... }:
let
  config = (pkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [{
      imports = [
        "${pkgs}/nixos/modules/virtualisation/qemu-vm.nix"
        ./base-system.nix    # hostname + xorg + users
        ./autostart-ff.nix
      ];
      virtualisation.memorySize = 8192;
    }];
  }).config;
in
config.system.build.vm
```

Standard lib function to eval system config

NixOS Makes VMs Easy

firefox-vm.nix

```
{pkgs, ... }:
let
  config = (pkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [{
      imports = [
        "${pkgs}/nixos/modules/virtualisation/qemu-vm.nix"
        ./base-system.nix    # hostname + xorg + users
        ./autostart-ff.nix
      ];
      virtualisation.memorySize = 8192;
    }];
  }).config;
in
config.system.build.vm
```

Standard lib function to eval system config

Autostart Firefox with XDG

NixOS Makes VMs Easy

firefox-vm.nix

```
{pkgs, ... }:
let
  config = (pkgs.lib.nixosSystem {
    system = "x86_64-linux";
    modules = [{
      imports = [
        "${pkgs}/nixos/modules/virtualisation/qemu-vm.nix"
        ./base-system.nix    # hostname + xorg + users
        ./autostart-ff.nix
      ];
      virtualisation.memorySize = 8192;
    }];
  }).config;
in
config.system.build.vm
```

Standard lib function to eval system config

Autostart Firefox with XDG

A script to run system as VM

Autostart Firefox with XDG

autostart-ff.nix

```
{pkgs, ...}: {
  environment.systemPackages = [
    (pkgs.writeTextFile {
      name = "xdg-autostart-firefox";
      destination = "/etc/xdg/autostart/xdg-autostart-firefox.desktop";
      text = ''
          [Desktop Entry]
          Name=xdg-autostart-firefox
          Type=Application
          Terminal=false
          Exec=${pkgs.firefox}/bin/firefox https://nixos.org
        '';
    })
  ];
}
```

Autostart Firefox with XDG

autostart-ff.nix

```
{pkgs, ...}: {
  environment.systemPackages = [
    (pkgs.writeTextFile {
      name = "xdg-autostart-firefox";
      destination = "/etc/xdg/autostart/xdg-autostart-firefox.desktop";
      text = ''
          [Desktop Entry]
          Name=xdg-autostart-firefox
          Type=Application
          Terminal=false
          Exec=${pkgs.firefox}/bin/firefox https://nixos.org
        '';
    })
  ];
}
```

Command to run and url to visit

Launch the VM with the Generated Script

config.system.build.vm -> run-firefoxvm-vm

```
#! /nix/store/<sha>-bash-5.2-p15/bin/bash
...
if test -n "$NIX_DISK_IMAGE" && ! test -e "$NIX_DISK_IMAGE"; then
    echo "Disk image do not exist, creating the virtualisation disk
    ↪  image..."
    ...
fi

# Start QEMU.
exec /nix/store/<sha>-qemu-host-cpu-only-8.0.3/bin/qemu-kvm -cpu max \
    -name firefoxvm \
    -m 8192 \
    ...
    $QEMU_OPTS \
    "$@"
```

VM Per Website: Let's Try the Naive Approach

Utility Function to Generate VM

```
mkBrowserVm = name: url:  (pkgs.lib.nixosSystem {
  system = "x86_64-linux";
  modules = [{
    imports = [ ... ];
    networking.hostName = "firefoxvm-${name}";
    _module.args.url = url; # convenience to reduce LOC
  }];
}).config.system.build.vm;
```

VM Per Website: Let's Try the Naive Approach

Utility Function to Generate VM

```
mkBrowserVm = name: url: pkgs.lib.nixosSystem {
  system = "x86_64-linux";
  modules = [{
    imports = [ ... ];
    networking.hostName = "firefoxvm-${name}";
    _module.args.url = url; # convenience to reduce LOC
  }];
}).config.system.build.vm;
```

Take a name for the VM and URL to visit

VM Per Website: Let's Try the Naive Approach

Utility Function to Generate VM

```
mkBrowserVm = name: url: pkgs.lib.nixosSystem {
  system = "x86_64-linux";
  modules = [{
    imports = [ ... ];
    networking.hostName = "firefoxvm-${name}";
    _module.args.url = url; # convenience to reduce LOC
  }];
}).config.system.build.vm;
```

Take a name for the VM and URL to visit

Output the VM start script

VM Per Website: Let's Try the Naive Approach

Utility Function to Generate VM

```
mkBrowserVm = name: url: pkgs.lib.nixosSystem {
  system = "x86_64-linux";
  modules = [{
    imports = [ ... ];
    networking.hostName = "firefoxvm-${name}";
    _module.args.url = url; # convenience to reduce LOC
  }];
}).config.system.build.vm;
```

Take a name for the VM and URL to visit

Pass url as argument to all modules

Output the VM start script

VM Per Website: Templating is Convenient

autostart-ff.nix

```
{pkgs, url, ...}: {
  environment.systemPackages = [
    (pkgs.writeTextFile {
      name = "xdg-autostart-firefox";
      destination = "/etc/xdg/autostart/xdg-autostart-firefox.desktop";
      text = ''
          [Desktop Entry]
          Name=xdg-autostart-firefox
          Type=Application
          Terminal=false
          Exec=${pkgs.firefox}/bin/firefox ${url}
        '';
    })
  ];
}
```

VM Per Website: Templating is Convenient

autostart-ff.nix

```
{pkgs, url, ...}: {
  environment.systemPackages = [
    (pkgs.writeTextFile {
      name = "xdg-autostart-firefox";
      destination = "/etc/xdg/autostart/xdg-autostart-firefox.desktop";
      text = ''
          [Desktop Entry]
          Name=xdg-autostart-firefox
          Type=Application
          Terminal=false
          Exec=${pkgs.firefox}/bin/firefox ${url}
        '';
    })
  ];
}
```

Visit the configured URL

VM Per Website: VM all the Sites!

site-vms.nix

```nix
let
  sites = {
    discourse = "https://discourse.nixos.org";
    nixos = "https://nixos.org";
    github = "https://github.com/nixos/nixpkgs";
  };
  vms = pkgs.lib.mapAttrsToList mkBrowserVm sites;
in
pkgs.symlinkJoin {
  name = "site-vms";
  paths = vms;
}
```

VM Per Website: VM all the Sites!

site-vms.nix

```nix
let
  sites = {
    discourse = "https://discourse.nixos.org";
    nixos = "https://nixos.org";
    github = "https://github.com/nixos/nixpkgs";
  };
  vms = pkgs.lib.mapAttrsToList mkBrowserVm sites;
in
pkgs.symlinkJoin {
  name = "site-vms";
  paths = vms;
}
```

Generate VMs using our utility function.

VM Per Website: VM all the Sites!

site-vms.nix

```nix
let
  sites = {
    discourse = "https://discourse.nixos.org";
    nixos = "https://nixos.org";
    github = "https://github.com/nixos/nixpkgs";
  };
  vms = pkgs.lib.mapAttrsToList mkBrowserVm sites;
in
pkgs.symlinkJoin {
  name = "site-vms";
  paths = vms;
}
```

Generate VMs using our utility function.

Join the VMs into one result

VM Per Website: Naive Approach is Slow

- Build took 30s with 2 VMs. 48s with 4 VMs. Roughly linear.
- With dozens, build was taking minutes, even with no changes to VMs.

Important Observations on Slow Builds

- A NixOS system is built per website
- pkgs.lib.nixosSystem performs full eval

Improvement 1: Use Flakes to Cache VM Build

flake.nix

```
{
  inputs.nixpkgs.url = "nixpkgs/nixos-23.05";
  outputs = { self, ... }@inputs: {
    nixosConfigurations.firefoxvm = inputs.nixpkgs.lib.nixosSystem {
      specialArgs = { inherit inputs; };
      system = "x86_64-linux";
      modules = [./nixos-vm.nix];
    };
  };
}
```

Improvement 1: Use Flakes to Cache VM Build

flake.nix

```
{
  inputs.nixpkgs.url = "nixpkgs/nixos-23.05";
  outputs = { self, ... }@inputs: {
    nixosConfigurations.firefoxvm = inputs.nixpkgs.lib.nixosSystem {
      specialArgs = { inherit inputs; };
      system = "x86_64-linux";
      modules = [./nixos-vm.nix];
    };
  };
}
```

Standard flake output for a NixOS system

Improvement 1: Use Flakes to Cache VM Build

flake.nix

```
{
  inputs.nixpkgs.url = "nixpkgs/nixos-23.05";
  outputs = { self, ... }@inputs: {
    nixosConfigurations.firefoxvm = inputs.nixpkgs.lib.nixosSystem {
      specialArgs = { inherit inputs; };
      system = "x86_64-linux";
      modules = [./nixos-vm.nix];
    };
  };
}
```

Standard flake output for a NixOS system

System build will now be cached

Improvement 2: QEMU Runtime Variables

QEMU Option:

```
-fw_cfg [name=]<item_name>,string=<string>
```

Improvement 2: QEMU Runtime Variables

QEMU Option:

```
-fw_cfg [name=]<item_name>,string=<string>
```

Creates File:

```
/sys/firmware/qemu_fw_cfg/by_name/<name>/raw
```

Improvement 2: QEMU Runtime Variables

QEMU Option:

```
-fw_cfg [name=]<item_name>,string=<string>
```

Creates File:

```
/sys/firmware/qemu_fw_cfg/by_name/<name>/raw
```

Setting Option:

```
-fw_cfg name=opt/ffurl,string=https://nixos.org
```

Improvement 2: QEMU Runtime Variables

QEMU Option:

```
-fw_cfg [name=]<item_name>,string=<string>
```

Creates File:

```
/sys/firmware/qemu_fw_cfg/by_name/<name>/raw
```

Setting Option:

```
-fw_cfg name=opt/ffurl,string=https://nixos.org
```

Creates File:

```
> cat /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw
https://nixos.org
```

Improvement 2: Use the Runtime Variable in Launch Script

runvm-pkg.nix

```
{self, pkgs, ...}:

pkgs.writeShellScriptBin "ffvm-visit-url" ''
set -euo pipefail

NAME="$1"
URL="$2"

export QEMU_OPTS="-fw_cfg name=opt/ffurl,string=$URL"
export NIX_DISK_IMAGE="$HOME/vms/ffvm-$NAME.qcow2"

${self.nixosConfigurations.firefoxvm.config.system.build.vm}/bin/run-firefoxvm-
''
```

Improvement 2: Use the Runtime Variable in Launch Script

runvm-pkg.nix

Take name and url as args

```
{self, pkgs, ...}:

pkgs.writeShellScriptBin "ffvm-visit-url" ''
set -euo pipefail

NAME="$1"
URL="$2"

export QEMU_OPTS="-fw_cfg name=opt/ffurl,string=$URL"
export NIX_DISK_IMAGE="$HOME/vms/ffvm-$NAME.qcow2"

${self.nixosConfigurations.firefoxvm.config.system.build.vm}/bin/run-firefoxvm-
''
```

Improvement 2: Use the Runtime Variable in Launch Script

runvm-pkg.nix

```
{self, pkgs, ...}:

pkgs.writeShellScriptBin "ffvm-visit-url" ''
set -euo pipefail

NAME="$1"
URL="$2"

export QEMU_OPTS="-fw_cfg name=opt/ffurl,string=$URL"
export NIX_DISK_IMAGE="$HOME/vms/ffvm-$NAME.qcow2"

${self.nixosConfigurations.firefoxvm.config.system.build.vm}/bin/run-firefoxvm-
''
```

Take name and url as args

Set variables for the VM launch script to use

Improvement 2: Use the Runtime Variable in Launch Script

runvm-pkg.nix

```
{self, pkgs, ...}:

pkgs.writeShellScriptBin "ffvm-visit-url" ''
set -euo pipefail

NAME="$1"
URL="$2"

export QEMU_OPTS="-fw_cfg name=opt/ffurl,string=$URL"
export NIX_DISK_IMAGE="$HOME/vms/ffvm-$NAME.qcow2"

${self.nixosConfigurations.firefoxvm.config.system.build.vm}/bin/run-firefoxvm-
''
```

Take name and url as args

Set variables for the VM launch script to use

Start VM, referencing the flake output

Improvement 2: Configure VM to Read Variable

autostart-ff.nix

```
{pkgs, ...}:
let
  ff-visit-url = pkgs.writeShellScriptBin "ff-visit-url" ''
    URL=$(cat /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw)
    ${pkgs.firefox}/bin/firefox "$URL"
  '';
  autostart = pkgs.writeTextFile { ... };
in
{
  environment.systemPackages = [ autostart ];
  systemd.tmpfiles.rules =
    [ "z /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw 0440 root
    ↪  users" ];
}
```

Improvement 2: Configure VM to Read Variable

autostart-ff.nix

Autostart with a custom script which reads variable

```
{pkgs, ...}:
let
  ff-visit-url = pkgs.writeShellScriptBin "ff-visit-url" ''
    URL=$(cat /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw)
    ${pkgs.firefox}/bin/firefox "$URL"
  '';
  autostart = pkgs.writeTextFile { ... };
in
{
  environment.systemPackages = [ autostart ];
  systemd.tmpfiles.rules =
    [ "z /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw 0440 root
    ↪  users" ];
}
```

Improvement 2: Configure VM to Read Variable

autostart-ff.nix

```
{pkgs, ...}:
let
  ff-visit-url = pkgs.writeShellScriptBin "ff-visit-url" ''
    URL=$(cat /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw)
    ${pkgs.firefox}/bin/firefox "$URL"
  '';
  autostart = pkgs.writeTextFile { ... };
in
{
  environment.systemPackages = [ autostart ];
  systemd.tmpfiles.rules =
    [ "z /sys/firmware/qemu_fw_cfg/by_name/opt/ffurl/raw 0440 root
    ↪  users" ];
}
```

Autostart with a custom script which reads variable

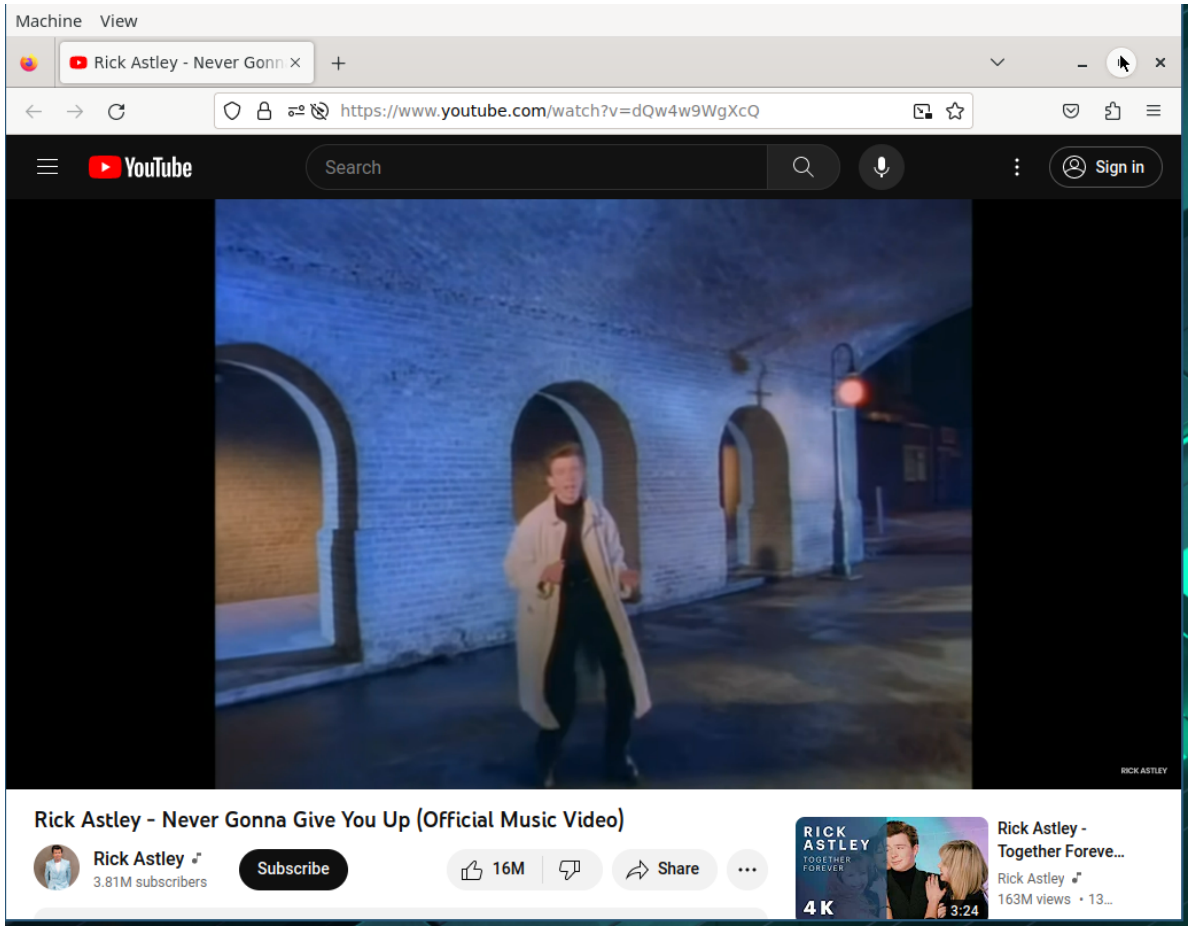
Ensure variable can be read

We have our single site VM!

```
> ./result/bin/ffvm-visit-url greatsong
↪  "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
```

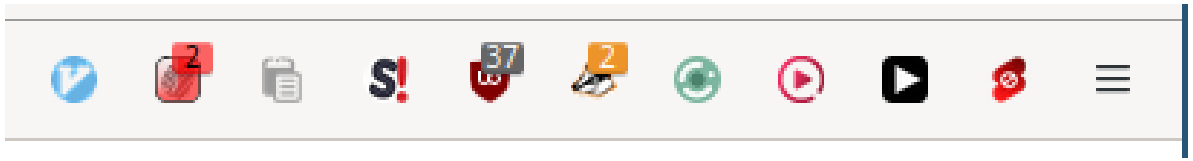
We have our single site VM!

```
> ./result/bin/ffvm-visit-url greatsong
↪  "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
```



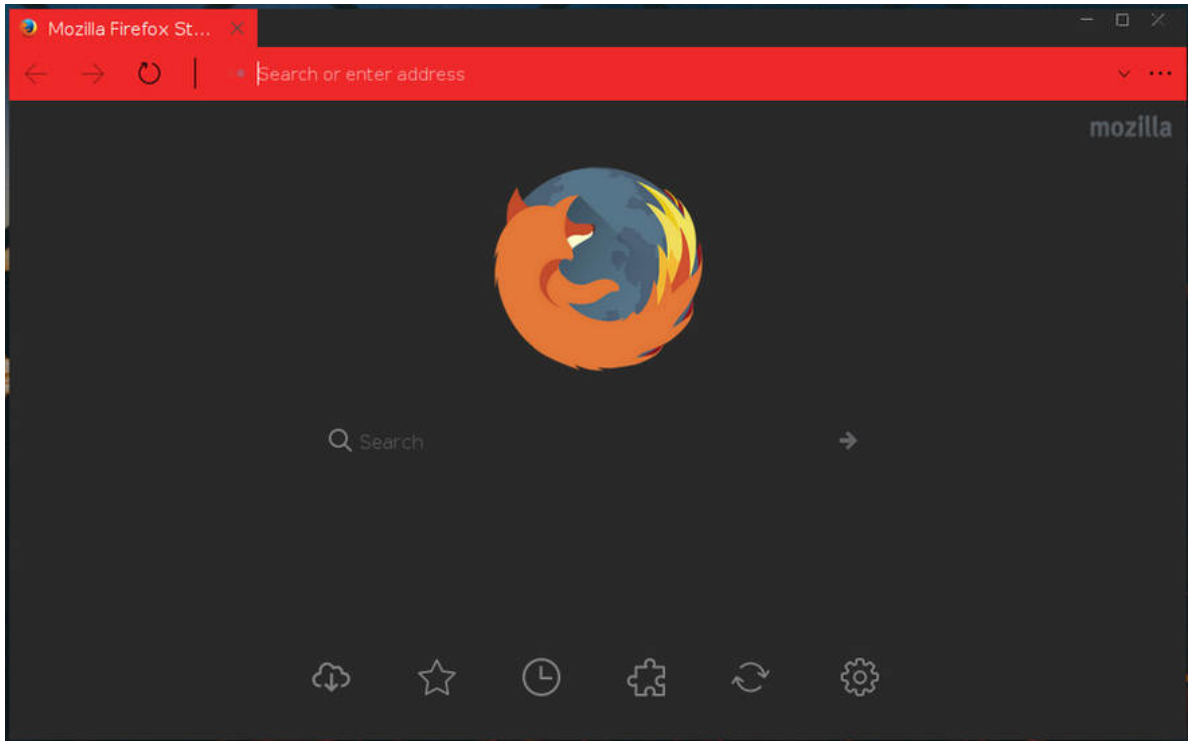
Customize

- Firefox with Addons



Customize

- Firefox with Addons
- User-chrome Styling



Customize

- Firefox with Addons
- User-chrome Styling
- Networking: VPN Configuration

Customize

- Firefox with Addons
- User-chrome Styling
- Networking: VPN Configuration