aarch64-build-box / configuration.nix

mweinelt Unpin kernel 5.15 and move to latest 7fafd5f · 2 months ago

306 lines (273 loc) · 8.46 KB

```
1   { pkgs, modulesPath, lib, ... }: {
2     imports = [
3       (modulesPath + "/profiles/all-hardware.nix")
4       (modulesPath + "/profiles/minimal.nix")
5
6       ({ pkgs, config, ...}: { # Hardware Tuning
7         boot = {
8           consoleLogLevel = 7;
9           initrd.availableKernelModules = [ "ahci" "hisi-rng" ];
10
11          kernelParams = [
12            "cma=0M" "biosdevname=0" "net.ifnames=0" "console=ttyAMA0,115200
13            "initrd=initrd"
14          ];
15          kernelPackages = pkgs.linuxPackages_latest;
16          #
17          kernel.sysctl."kernel.hostname" = "${config.networking.hostName}.$
18        };
19
20        nix.nrBuildUsers = config.nix.settings.max-jobs * 2;
21        nix.settings.max-jobs = 64;
22        nixpkgs.system = "aarch64-linux";
23      })
24
25      ({ # Go fast: networking
26        networking.hostName = "aarch64";
27        networking.domain = "nixos.community";
28        networking.dhcpcd.enable = false;
29        networking.defaultGateway = {
30          address = "147.28.143.249";
31          interface = "bond0";
32        };
33        networking.defaultGateway6 = {
34          address = "2604:1380:4641:c900::e";
35          interface = "bond0";
```

```
36          };
37          networking.nameservers = [
38            "147.75.207.207"
39            "147.75.207.208"
40          ];
41
42          networking.firewall.logRefusedConnections = false;
43
44          networking.bonds.bond0 = {
45            driverOptions = {
46              mode = "802.3ad";
47              xmit_hash_policy = "layer3+4";
48              lacp_rate = "fast";
49              downdelay = "200";
50              miimon = "100";
51              updelay = "200";
52            };
53            interfaces = [
54              "eth2" "eth3"
55            ];
56          };
57
58          networking.interfaces.bond0 = {
59            useDHCP = false;
60            macAddress = "14:30:04:ea:87:26";
61
62            ipv4 = {
63              routes = [
64                {
65                  address = "10.0.0.0";
66                  prefixLength = 8;
67                  via = "10.70.108.142";
68                }
69              ];
70              addresses = [
71                {
72                  address = "147.28.143.250";
73                  prefixLength = 30;
74                }
75                {
76                  address = "10.70.108.143";
77                  prefixLength = 31;
78                }
79              ];
80            };
81
82            ipv6 = {
83              addresses = [
84                {
85                  address = "2604:1380:4641:c900::f";
86                  prefixLength = 127;
87                }
```

```
88              ];
89            };
90          };
91        })
92
93        ({pkgs, ...}: { # Config specific to this purpose
94          services.openssh = {
95            enable = true;
96            settings.KbdInteractiveAuthentication = false;
97            settings.PasswordAuthentication = false;
98            extraConfig = ''
99              MaxSessions 65
100           '';
101         };
102         security.sudo.wheelNeedsPassword = false;
103
104         boot.initrd.postDeviceCommands = "${pkgs.writeScript "post-device-cc
105           #!/bin/sh
106
107           set -eu
108           set -o pipefail
109
110           PATH="${pkgs.coreutils}/bin:${pkgs.util-linux}/bin:${pkgs.gnugrep}
111
112           exec ${./post-devices.sh}
113         ''}";
114
115         services.openssh.hostKeys = [
116           { path = "/persist/ssh/ssh_host_ed25519_key"; type = "ed25519"; }
117           { path = "/persist/ssh/ssh_host_rsa_key"; type = "rsa"; }
118         ];
119
120         environment.systemPackages = [
121           pkgs.btop
122           pkgs.git
123           pkgs.htop
124         ];
125
126         systemd.services.nix-daemon = {
127           environment.LD_PRELOAD = "${pkgs.libeatmydata}/lib/libeatmydata.so
128         };
129
130         nix = {
131           gc = {
132             automatic = true;
133             options = "--max-freed $((64 * 1024**3))";
134           };
135
136           settings = {
137             cores = 0;
138             experimental-features = [ "flakes" "nix-command" ];
139             sandbox = true;
140             trusted-users = [ "@wheel" "@trusted" ];
```

```
141           };
142         };
143       })
144
145       ({pkgs, ...}: {
146         nix = {
147           nixPath = [
148             # Ruin the config so we don't accidentally run
149             # nixos-rebuild switch on the host
150             (let
151               cfg = pkgs.writeText "configuration.nix"
152                 ''
153                   assert builtins.trace "Hey dummy, you're on your server! U
154                   {}
155                 '';
156              in "nixos-config=${cfg}")
157
158              # Copy the channel version from the deploy host to the target
159              "nixpkgs=/run/current-system/nixpkgs"
160           ];
161         };
162
163         system.extraSystemBuilderCmds = ''
164           ln -sv ${pkgs.path} $out/nixpkgs
165         '';
166         environment.etc.host-nix-channel.source = pkgs.path;
167       })
168
169       {
170         options.ofborg.package = lib.mkOption {
171           description = "Ofborg package";
172           type = lib.types.package;
173         };
174       }
175       ({pkgs, config, ...}: {
176         users.users.gc-of-borg = {
177           description = "GC Of Borg Workers";
178           home = "/var/lib/gc-of-borg";
179           createHome = true;
180           group = "gc-of-borg";
181           uid = 402;
182         };
183         users.groups.gc-of-borg.gid = 402;
184
185         systemd.services = let
186           builder = id: {
187             enable = true;
188             after = [ "network.target" "network-online.target" ];
189             wants = [ "network-online.target" ];
190             wantedBy = [ "multi-user.target" ];
191
192             path = with pkgs; [
```

```
193               nix
194               git
195               curl
196               bash
197             ];
198
199             serviceConfig = {
200               User = "gc-of-borg";
201               Group = "gc-of-borg";
202               PrivateTmp = true;
203               WorkingDirectory = "/var/lib/gc-of-borg";
204               Restart = "always";
205               RestartSec = "10s";
206             };
207
208             preStart = ''
209               mkdir -p ./nix-test-rs-${id}
210             '';
211
212             script = ''
213               export HOME=/var/lib/gc-of-borg;
214               export NIX_REMOTE=daemon;
215               export NIX_PATH=nixpkgs=/run/current-system/nixpkgs;
216               git config --global user.email "graham+cofborg@grahamc.com"
217               git config --global user.name "GrahamCOfBorg"
218               export RUST_BACKTRACE=1
219
220               ${config.ofborg.package}/bin/builder /persist/ofborg/config-${
221             '';
222           };
223
224           makeNBuilders = count:
225             let
226               toMake = pkgs.lib.range 1 count;
227               services = map
228                 (n: { "grahamcofborg-builder-${toString n}" = builder (toStr
229                 toMake;
230             in pkgs.lib.foldr (x: y: x // y) {} services;
231         in makeNBuilders 16;
232       })
233
234       ({ pkgs, ... }: {
235         systemd.services.clone-nixpkgs = {
236           wantedBy = [ "multi-user.target" ];
237           wants = [ "network-online.target" ];
238           after = [ "network-online.target" ];
239           serviceConfig.Type = "oneshot";
240           startAt = "daily";
241           script = ''
242             if [ ! -d /tmp/nixpkgs.git ]; then
243               ${pkgs.git}/bin/git clone --bare https://github.com/nixos/nixp
244             fi
```

```
245           '';
246         };
247       })
248
249       ({ config, modulesPath, lib, pkgs, ... }: {
250         fileSystems."/" = {
251           fsType = "btrfs";
252           label = "root";
253           options = ["compress=lzo" "noatime" "discard=async"];
254         };
255         fileSystems."/persist" = {
256           fsType = "ext4";
257           label = "persist";
258           neededForBoot = true;
259         };
260         boot.loader.grub.enable = false;
261         system.build.bootStage2 = lib.mkForce config.system.build.bootStage1
262         boot.kernelParams = let
263           stage2module = import (modulesPath + "/system/boot/stage-2.nix") {
264             inherit config lib pkgs;
265           };
266           stage2Init = pkgs.runCommand "init.sh" {} ''
267             substitute ${stage2module.config.system.build.bootStage2} $out \
268               --replace-fail 'systemConfig=@systemConfig@' '
269                 for o in $(</proc/cmdline); do
270                   case $o in
271                     rdinit=*)
272                       set -- $(IFS==; echo $o)
273                       rdinit=$2
274                       ;;
275                     *)
276                       ;;
277                   esac
278                 done
279                 systemConfig=''${rdinit%/init}
280               '
281             chmod +x $out
282           '';
283         in [
284           "boot.trace"
285           "init=${stage2Init}"
```

aarch64-build-box / configuration.nix ↑ Top

Code Blame Raw

```
290           # /etc/NIXOS tag.
291           touch /etc/NIXOS
292           PATH=/run/current-system/sw/bin /nix/.nix-netboot-serve-db/registe
293           ${config.nix.package}/bin/nix-env -p /nix/var/nix/profiles/system
294         '';
295         boot.initrd.preDeviceCommands = ''
296           ln -s ${config.system.modulesTree}/lib /lib
297         '';
```

```
298           system.build.initialRamdisk = lib.mkForce (pkgs.writeText "initrd" '
299         })
300
301         ./users.nix
302         ./monitoring.nix
303         ./motd.nix
304         ./armv7l.nix
305       ];
306     }
```