NixOS 치트시트

기본 개념

NixOS는 선언적(declarative) 구성 모델을 사용하는 Linux 배포판입니다. 시스템의 전체 구성이 `/etc/nixos/configuration.nix` 파일 하나에 정의되며, 이를 통해 재현 가능하고 신뢰할 수 있는 시스템을 구축할 수 있습니다. Nix 패키지 관리자를 기반으로 합니다.

시스템 관리

- `nixos-rebuild switch`: `/etc/nixos/configuration.nix` 파일의 변경사항을 적용하여 시스템을 업데이트합니다.
- `nixos-rebuild boot`: 시스템을 업데이트하지만, 지금 바로 적용하는 대신 다음 부팅 시에 적용되도록 합니다.
- `nixos-rebuild test`: 시스템을 테스트 빌드하지만, 적용하지는 않습니다.
- `nixos-rebuild switch --upgrade`: 채널을 최신 버전으로 업데이트한 후 시스템을 업데이트합니다.
- `nixos-rebuild --rollback`: 이전 세대의 시스템 구성으로 롤백합니다.
- `nix-channel --list`: 현재 구독 중인 채널 목록을 보여줍니다.
- `nix-channel --update`: 모든 채널을 최신 버전으로 업데이트합니다.

패키지 관리 (Nix-Env)

- `nix-env -iA nixos.<package_name>`: 패키지를 설치합니다. (예: `nix-env -iA nixos.firefox`)
  - `nix-env -iA`는 `nix-env --install --attr`의 축약형입니다.
- `nix-env -e <package_name>`: 패키지를 제거합니다.
- `nix-env -q`: 설치된 모든 패키지를 나열합니다.
- `nix-env --upgrade`: 설치된 모든 패키지를 업그레이드합니다.
- 참고: 영구적인 시스템 구성을 위해서는 `configuration.nix`의 `environment.systemPackages`에 패키지를 추가하는 것이 권장됩니다. `nix-env`는 주로 임시적인 프로필 관리에 사용됩니다.

Nix 셸 (임시 환경)

- `nix-shell -p <package1> <package2>`: 특정 패키지들이 포함된 임시 셸을 시작합니다. 이 셸을 빠져나가면 시스템은 원래 상태로 돌아갑니다.
- `nix-shell`: 현재 디렉토리에 `shell.nix` 또는 `default.nix` 파일이 있는 경우, 해당 파일에 정의된 개발 환경을 로드합니다.
- `nix-shell --pure`: 호스트 환경의 영향을 받지 않는 순수한 셸을 시작합니다.

Nix Store 및 가비지 컬렉션

- Nix는 모든 패키지를 `/nix/store`에 고유한 해시 경로로 저장합니다.
- `nix-store --query --requisites /run/current-system`: 현재 시스템의 모든 종속성을 나열합니다.
- `nix-collect-garbage`: 더 이상 어떤 프로필이나 구성에서도 참조되지 않는 패키지들을 삭제합니다.
- `nix-collect-garbage -d`: 모든 세대(롤백 가능한 구성)를 삭제하고, 현재 세대에서 사용하지 않는 패키지들을 모두 정리합니다. (주의: 롤백이 불가능해짐)

`configuration.nix` 예시 조각

```
{ config, pkgs, ... }:

{
  imports =
    [ ./hardware-configuration.nix ];

  boot.loader.grub.enable = true;
  boot.loader.grub.device = "/dev/sda";

  networking.hostName = "nixos";
  networking.useDHCP = false;
  networking.interfaces.enp0s3.useDHCP =
true;

  # 시간대 설정
  time.timeZone = "Asia/Seoul";

  # 시스템 패키지
  environment.systemPackages = with
pkgs; [
    vim
    wget
    git
    firefox
  ];

  # 사용자 계정
  users.users.jules = {
    isNormalUser = true;
    extraGroups = [ "wheel" ]; # sudo 권
한
  };

  # 서비스 활성화
  services.openssh.enable = true;
}
```

Nix 언어 기본

- let-in 블록: `let x = 1; y = 2; in x + y`
- 함수: `x: x + 1`
- 집합 (Attribute Set): `{ name = "value"; nested = { ... }; }`
- `with pkgs; [ git vim ]`는 `[ pkgs.git pkgs.vim ]`와 같습니다.